Высшее профессиональное образование

А. В. Могилев
Н. И. Пак
Е. К. Хеннер

ПРАКТИКУМ ПО ИНФОРМАТИКЕ

2-е издание

Учебное пособие

Педагогические специальности

ACADEMIA

ВЫСШЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

А. В. МОГИЛЕВ, Н. И. ПАК, Е. К. ХЕННЕР

ПРАКТИКУМ ПО ИНФОРМАТИКЕ

Под редакцией Е. К. Хеннера

2-издание, стереотипное

Москва
ACADEMIA
2005

УДК 076.5
ББК 32.81
М74

Рецензенты:
проректор по информатизации, зав. кафедрой ОмГПУ
д-р пед. наук проф. *М. П. Лапчик*;
зав. кафедрой информатики МГОУ им. М. А. Шолохова,
президент Академии информатизации образования
д-р техн. наук проф. *Я. А. Ваграменко*

Могилев А. В.

М74 Практикум по информатике: Учеб. пособие для студ. высш. учеб. заведений / А. В. Могилев, Н. И. Пак, Е. К. Хеннер; Под ред. Е. К. Хеннера. — 2-е изд., стер. — М.: Издательский центр «Академия», 2005. — 608 с.

ISBN 5-7695-2247-X

Практикум по тематике, уровню сложности и методическим подходам соответствует учебному пособию А. В. Могилева, Н. И. Пака и Е. К. Хеннера «Информатика». Он включает разделы: теоретическая информатика; программное обеспечение ЭВМ; языки и методы программирования; вычислительная техника; компьютерные сети и телекоммуникации; информационные системы; компьютерное моделирование. Даны наборы тренировочных заданий; лабораторные работы; материалы для тестового контроля по основным темам.

Для студентов университетов и педагогических вузов, проходящих подготовку в качестве бакалавров и магистров образования по профилю «Информатика», а также для учителей информатики. Может быть полезным преподавателям информатики в вузах при подготовке и проведении занятий. Может использоваться при реализации образовательных программ в различных вузах, в которых информатика является одним из профилирующих предметов, а также в процессе переподготовки и повышения квалификации учителей информатики в системе повышения квалификации работников образования.

УДК 076.5
ББК 32.81

ISBN 5-7695-2247-X

ПРЕДИСЛОВИЕ

Данный практикум по тематике, уровню сложности и методическим подходам соответствует учебному пособию А.В.Могилева, Н.И.Пака и Е.К.Хеннера «Информатика» (Издательский центр «Академия», 2004, 3-е изд.), является его продолжением и опирается на теоретический материал, изложенный в указанном пособии.

Практикум охватывает все разделы, присутствующие в указанном пособии:

- теоретические основы информатики;
- программное обеспечение;
- языки и методы программирования;
- вычислительная техника;
- компьютерные сети и телекоммуникации;
- информационные системы;
- компьютерное моделирование.

Как и указанное учебное пособие, практикум предназначен для реализации профессиональных образовательных программ в различных вузах, в которых информатика является одной из профилирующих дисциплин.

Авторы надеются, что практикум будет полезен не только студентам педагогических вузов и классических университетов, избравшим обучение информатике сферой своей будущей профессиональной деятельности, но и учителям информатики для самообразования и повышения квалификации. Часть материалов, отраженных в практикуме, можно использовать в школах при углубленном изучении информатики.

Выполнение работ в объеме данного практикума позволит выработать устойчивые навыки во многих областях практической деятельности современной информатики. В силу своего назначения практикум не исчерпывает большинства включенных в него тем, но создает базу для специализации в той или иной сфере информатики.

При выполнении работ практикума доля самостоятельной деятельности студентов должна быть существенно выше, чем при других видах учебной работы; преподаватель в этой ситуации достаточно часто выступает в роли консультанта. Это помогает будущему специалисту научиться самостоятельно осваивать новые знания и навыки, что является одной из важнейших целей обучения.

Практикум поддерживает различные виды деятельности, дополняющей теоретическую подготовку. Он включает темы для рефератов, темы для семинарских занятий, упражнения и задачи, лабораторные работы по основным темам и разделам курса информатики.

Не все темы, изучаемые в курсе информатики, связаны с приобретением узкопрактических навыков, связанных с компьютером, решением задач. Такими являются, например, чисто теоретические темы «Информатика как наука и как вид практической деятельности», «История развития вычислительной техники» и другие. В силу особенностей подготовки той аудитории, которой адресован практикум, и некоторые другие темы не ориентированы на приобретение практических навыков в узком смысле слова. Не требуется от учителя, например, уметь разрабатывать трансляторы, администрировать сетевые операционные системы и тому подобное. Авторы трактуют понятие «практикум» достаточно широко, поэтому по ряду разделов приходится ограничиться указанием тем семинарских занятий, рефератов. Более того, авторы сочли возможным не затрагивать вообще некоторых

вопросов, присутствующих в базовом учебном пособии для ознакомления. Для некоторых таких тем практикум для той категории студентов, которой он адресован, нуждается в специальных программных системах-тренажерах, разработка которых в основном — дело будущего.

В то же время очень многие вопросы информатики требуют от студентов приобретения навыков решения задач, пользования весьма сложными программами, разработки пользовательских программ на нескольких языках программирования. По соответствующим темам практикум включает задачи и упражнения, лабораторные работы. Учитывая состояние наличного компьютерного и программного обеспечения различных вузов, авторы рассчитывали на персональные компьютеры того уровня, который сегодня практически стал общераспространенным. Сложнее вопрос о программных средствах, необходимых для решения задач и выполнения лабораторных работ. Часть из них являются общераспространенными (некоторые операционные системы и их оболочки, пакет Microsoft Office и др.), часть распространяется фирмами-производителями бесплатно — особенно когда речь идет не о новейших версиях (но вполне достаточных для ознакомления с информационными технологиями, заложенными в них). Подчеркнем, что практикум отнюдь не столь специализирован, чтобы, например, при выполнении работы по машинной графике почувствовалась разница между свободно распространяемой версией CorelDraw 5 и весьма дорогой CorelDraw 7 (написано в конце 2000 г.; скорее всего, этот пример быстро станет неактуальным). Некоторые программы, используемые в данном практикуме, могут быть свободно получены из Internet; соответствующие справки приведены в тексте.

Последовательность прохождения тем практикума может отличаться от той, в которой расположен материал в данной книге. Информатика не столь формализована как, например, математика, и многие разделы информатики при изучении можно менять местами. Порядок прохождения курса определяется конкретным вузом.

Для большей части лабораторных работ приводится примерная оценка трудоемкости (в часах). Она исходит из того, что студенты предварительно подготовились к выполнению работы, освоили соответствующий теоретический материал. Эта продолжительность может корректироваться преподавателем, ведущим занятия, путем определения обязательных для исполнения заданий (если в работе их несколько).

В практикум включен также набор тестов для контроля знаний. Эти тесты охватывают не только темы, включенные в практикум, но и те, которые отражены в базовом учебном пособии. Тесты будут полезны как для самооценки знаний по различным темам, так и для подготовки к выполнению работ практикума.

Авторы практикума — преподаватели кафедр информатики Воронежского, Красноярского и Пермского педагогических университетов.

В работе над пособием частично использованы материалы, предоставленные авторам их коллегами. Прежде всего это доцент кафедры информатики Пермского госпедуниверситета А. П. Шестаков (часть практических заданий для гл. 1, 4 и 7); доцент кафедры информатики Вятского госпедуниверситета С. М. Окулов (часть заданий гл. 3, связанных с программированием на Паскале); ст. преподаватель кафедры методики преподавания информатики Красноярского госпедуниверситета Л. Б. Хегай (часть заданий гл. 2, связанных с программами Word, Excel, Access, освоением компьютерной графики); доценты кафедры информатики Пермского госпедуниверситета Е. В. Соснина (гл. 1, лабораторные работы по машинам Поста и Тьюринга) и Т. И. Клигман (гл. 6, работа с ГИС «Карта Москвы»); инженер ИВЦ Пермского госпедуниверситета А. В. Князев (гл. 6, программа-имитатор ГИС). Авторы чрезвычайно им за это благодарны.

Глава 1

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

Теоретические основы информатики — пока не вполне сложившийся и устоявшийся раздел науки. Он возникает на наших глазах, что делает его особенно интересным: нечасто удается наблюдать и даже участвовать в рождении новой науки! Как и теоретические разделы других наук, теоретическая информатика формируется в основном под влиянием потребностей обучения информатике.

Теоретическая информатика — наука в значительной степени математизированная. Она базируется на ряде разделов математики: теории автоматов и алгоритмов, математической логике, теории формальных языков и грамматик, реляционной алгебре, теории информации и др.

Теоретическая информатика старается методами точного анализа ответить на основные вопросы, возникающие при хранении и обработке информации, например, чему равно количество информации, сосредоточенной в той или иной информационной системе, какова наиболее рациональная организация информации для хранения или поиска, а также существуют ли алгоритмы и каковы свойства алгоритмов преобразования информации.

Конструкторы устройств хранения данных проявляют чудеса изобретательности, увеличивая объем и плотность хранения данных на дисках, но в основе этой деятельности лежат теория информации и теория кодирования.

Для решения прикладных задач существуют замечательные программы, но для того, чтобы грамотно поставить прикладную задачу, привести ее к виду, который подвластен компьютеру, надо знать основы информационного и математического моделирования и т.д.

Только освоив эти разделы информатики, можно считать себя специалистом в этой науке. Другое дело — с какой глубиной осваивать; многие разделы теоретической информатики достаточно сложны и требуют основательной математической подготовки.

Большую часть практических занятий по теоретической информатике целесообразно строить в семинарской форме. Полезна подготовка рефератов, чтение докладов.

Важная задача практикума по этому разделу — формирование познавательных интересов, выработка навыков самостоятельной работы с литературой, расширение кругозора в области информатики.

§ 1. ИНФОРМАТИКА КАК НАУКА И КАК ВИД ПРАКТИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ

Рекомендации по проведению занятий

Практические занятия по этой теме проводятся на семинарах. Важную роль играет написание и защита рефератов.

Краткие сведения

Информатика — это наука и сфера практической деятельности, связанная с различными аспектами получения, хранения, обработки, передачи и использования информации.

«Понятие информатики охватывает области, связанные с разработкой, созданием, использованием и материально-техническим обслуживанием систем обработки информации, включая машины, оборудование, организационные аспекты, а также комплекс промышленного, коммерческого, административного и социального воздействия» — такая формулировка была дана в 1978 г. Международным конгрессом по информатике.

С современной точки зрения понятие «информатика» шире, чем используемое в англоязычных странах Computer Science, поскольку включает как прикладную часть («компьютерные науки»), так и теоретическую, связанную с отмеченными выше аспектами действий с информацией.

Информатика включает в себя следующие основные разделы: теоретическую информатику, вычислительную технику, программирование, информационные системы, искусственный интеллект. Она является конгломератом наук, объединенных общим объектом исследования.

Научное ядро информатики относят к фундаментальным наукам, поскольку ее основные понятия носят общенаучный характер, используются во многих других науках и видах деятельности.

Информатика имеет существенные ***социальные аспекты***. Она «включает... комплекс промышленного, коммерческого, административного и социального воздействия». Информатизация, т.е. процесс проникновения информационных технологий во все сферы жизни и деятельности общества, сильно влияет на социальную сферу. В настоящее время в общественном устройстве развитых стран появились черты информационного общества, во все сферы жизни и деятельности членов которого включены средства информатики в качестве орудий интеллектуального труда, переработки любой информации, моделирования реальных и прогнозируемых событий, управления производством, обучения и т.д.

Под влиянием информатизации радикально меняется структура труда, совершается переток людей из сферы прямого материального производства в так называемую информационную сферу. Промышленные рабочие и крестьяне, составлявшие в середине XX века более 2/3 населения, сегодня в развитых странах составляют менее 1/3. К середине 90-х годов численность «информационных работников» (к которым причисляют всех, в чьей профессиональной деятельности доминирует умственный труд) достигла в США 60 %. Добавим, что за те же годы производительность труда в США за счет научно-технического прогресса (ведь информатизация — его главная движущая сила) в целом выросла на 37 %.

Информатизация сильнейшим образом влияет на структуру экономики ведущих в экономическом отношении стран. В числе их лидирующих отраслей промышленности традиционные добывающие и обрабатывающие отрасли оттеснены максимально наукоемкими производствами электроники, средств связи и вычислительной техники — так называемой сферой высоких технологий. Темпы развития сферы высоких технологий и уровень прибылей в ней превышают в 5—10 раз темпы развития традиционных отраслей производства.

Существуют и отрицательные социальные последствия информатизации общества (по крайней мере при оценке с традиционных позиций): более быстрое высвобождение рабочей силы, чем это может освоить общество, повышенная социальная напряженность из-за роста интеллектуальной конкуренции, усиление контроля со стороны государства за каждым членом общества в демократических странах и т.д.

Правовые аспекты информатики связаны с тем, что деятельность программистов и других специалистов, работающих в сфере информатики, все чаще выступает в качестве объекта правового регулирования. Некоторые действия при этом могут быть квалифицированы как правонарушения (преступления). Регулированию подлежат вопросы собственности на информацию, охрана авторских прав на компьютерные программы и базы данных, гарантии сохранения конфиденциальности и секретности определенных видов информации и многое другое. Информатизация социальной сферы, распространение информационных сетей породили как новые виды преступности, так и многочисленные правовые проблемы, правовое регулирование которых далеко от завершения.

В Российской Федерации (как и в других странах) действуют специальные правовые акты, регламентирующие отношения в сфере информации. К ним, в частности, относятся:

- Закон Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных» (1992 г.);
- Указ Президента Российской Федерации «Об основах государственной политики в сфере информатизации» (1994 г., изменения и дополнения — 1995 г.);
- Закон Российской Федерации «Об информации, информатизации и защите информации» (1995 г.);
- Закон Российской Федерации «Об участии в международном информационном обмене» (1996 г.);
- Постановление Правительства Российской Федерации «О сертификации средств защиты информации» (1995 г.);
- Постановление Правительства Российской Федерации «О государственном учете и регистрации баз и банков данных» (1996 г.);
- Постановления Правительства Российской Федерации «О государственном учете и регистрации баз и банков данных» и «Об утверждении положения о государственной системе научно-технической информации» (1997 г.).

Этические аспекты информатики чрезвычайно важны. Далеко не все правила, регламентирующие деятельность в сфере информатики, можно свести к правовым нормам. Очень многое определяется соблюдением неписаных правил поведения для тех, кто причастен к миру компьютеров. Как и в любой другой большой и разветвленной сфере человеческой деятельности, в информатике к настоящему времени сложились определенные морально-этические нормы поведения и деятельности.

Этика — система норм нравственного поведения человека. Всякий раз, собираясь совершить сомнительный поступок в сфере профессиональной деятельности, человек должен задуматься, соответствует ли он этическим нормам, сложившимся в профессиональном сообществе.

Контрольные вопросы

1. Какие определения информатики Вы знаете?
2. Как появился термин «информатика»?
3. Каков объект и предмет исследования информатики?
4. Расскажите о целях и задачах информатики.
5. Что общего и в чем различие информатики и кибернетики?
6. Какое место занимает информатика в системе наук?
7. Какова структура современной информатики? Из каких частей и разделов она состоит?
8. Какие существуют наиболее известные информационные технологии?
9. Дайте определение фундаментальной естественной науки, приведите примеры.
10. Каково различие между естественными и техническими науками? К каким наукам следует отнести информатику?
11. Что такое информационная революция?
12. Назовите процессы, приводящие к созданию информационного общества.
13. Что принято понимать под «информационным обществом»?
14. Каковы основные социальные последствия информатизации общества?
15. Какими нормативными актами регулируются отношения в сфере информатики?
16. В чем состоит авторское право на программные средства и базы данных?
17. В чем состоит имущественное право на программные средства и базы данных?
18. Как осуществляется защита авторских и имущественных прав?
19. Охарактеризуйте виды компьютерных преступлений.
20. Расскажите об этике программистов и этических аспектах Internet.

Проблемные вопросы

1. Как и для чего появилась информатика?
2. Расскажите об информатике как об отрасли, как о науке, как о прикладной дисциплине.
3. Почему компьютеризация хотя и является важным шагом к информационному обществу, но еще не делает его таковым?
4. Какие этические проблемы существуют, по Вашему мнению, в современной информатике?
5. В чем заключается правовое регулирование на информационном рынке?
6. В чем отличие процессов компьютеризации и информатизации?
7. Чем определяется информационный потенциал общества?

Темы для рефератов

1. История развития информатики.
2. Кибернетика — наука об управлении.
3. Информатика и управление социальными процессами.
4. Информационные системы.
5. Автоматизированные системы управления.
6. Автоматизированные системы научных исследований.
7. Составные части современной информатики.
8. Построение интеллектуальных систем.

9. Информатика и математика.
10. Информатика и естественные науки.
11. Компьютер как историогенный фактор.
12. Компьютерная революция: социальные перспективы и последствия.
13. Путь к компьютерному обществу.
14. Информатика в деятельности юриста.
15. Общие приемы правового регулирования информационных отношений.
16. Правонарушения в сфере информационных технологий.
17. Правила этикета при работе с компьютерной сетью.
18. Защита информации в Internet.
19. Информационная основа управления экономикой.
20. Информационный бизнес.

Темы семинарских занятий

1. История развития информатики.
2. Информатика как единство науки и технологии.
3. Структура современной информатики.
4. Место информатики в системе наук.
5. Социальные аспекты информатики.
6. Правовые аспекты информатики.
7. Этические аспекты информатики.

Дополнительная литература

1. *Аветисян Р.Д., Аветисян Д.В.* Теоретические основы информатики. — М.: РГГУ, 1997.

2. *Азимов Ч.Н.* Научно-техническая информация и право. — Харьков: Выща шк., 1987.

3. *Андрундас Е.Ч.* Информационная элита: Корпорации и рынок новостей. — М.: Изд-во МГУ, 1991.

4. *Батурин Ю.М, Жодзинский А.М.* Компьютерная преступность и компьютерная безопасность. — М.: Юрид. лит., 1991.

5. Введение в информационный бизнес: Учеб. пособие / Под ред. В.П.Тихомирова, А.В.Хорошилова. — М.: Финансы и статистика, 1996.

6. *Воробьев Г.Т.* Твоя информационная культура. — М.: Мол. гвардия, 1988.

7. *Воронов Ю.П.* Компьютеризация: Шаг в будущее. — Новосибирск: Наука. Сиб. отд-ние, 1990.

8. *Гаврилов О.А.* Основы правовой информатики. — М.: Ин-т государства и права РАН, 1998.

9. *Гейтс Б.* Дорога в будущее. — М.: Русская редакция, 1996.

10. *Гольгамер Г.И.* Научно-информационная деятельность: Практика и проблемы. — М.: Радио и связь, 1987.

11. *Готт В.С., Семенюк Э.П., Урсул А.Д.* Социальная роль информатики. — М.: Знание, 1987.

12. Закон Российской Федерации «О правовой охране программ для электронных вычислительных машин и баз данных». — М., 1992.

13. Информатика / Под ред. Н.В.Макаровой. — М.: Финансы и статистика, 1997.

14. Информатика в понятиях и терминах. — М.: Просвещение, 1991.

15. Информатика в рабочих профессиях. — М.: Наука, 1990.
16. Информатика и культура. Сб. науч. тр. — Новосибирск: Наука. Сиб. отд-ние, 1990.
17. Информатика. Энциклопедический словарь для начинающих. — М.: Педагогика-Пресс, 1994.
18. Информационные технологии в научных исследованиях и испытаниях. Сб. науч. тр. — Киев: ИК, 1991.
19. *Коновец А.Ф.* НТП и информация. — М.: Знание, 1990.
20. *Крол Э.* Все об Internet: Пер. с англ. — СПб.: АО Балтэк; Киев: Торг.-изд. бюро BHV, 1995.
21. *Коханов В.В.* Информационные процессы в природе, обществе, технике. — Чебоксары: Клио, 1997.
22. *Ловцов Д.* Защита информации в глобальной сети ИНТЕРНЕТ // Информатика и образование. — 1998. — №5. — С.101—108.
23. *Мазур М.* Качественная теория информации. — М.: Мир, 1974.
24. *Майоров С.И.* Информационный бизнес: Коммерческое распространение и маркетинг. — М.: Финансы и статистика, 1993.
25. *Марков С.* Информатика как базовая наука образования // Информатика и образование. — 1998. — №6. — С.3—8.
26. *Морозов И.Ю.* Информатика: Учеб. пособие. Ч.1. — Омск: Изд-во ОмГУ, 1995.
27. Научные основы организации управления и построения АСУ / Под ред. В.Л.Бройло, В.С.Крылова. — М.: Высш. шк., 1990.
28. *Овезов Б.Б.* Автоматизация управления информационными процессами. — Ашхабад: Ылым, 1981.
29. *Першиков В.И., Савинков В.М.* Толковый словарь по информатике. — М.: Финансы и статистика, 1995.
30. Право и информатика / Под ред. Е.А.Суханова. — М.: Изд-во МГУ, 1990.
31. Применение информатики в управлении, обучении и научных исследованиях. — М.: Изд-во МГУ, 1989.
32. Проектирование и использование региональных АСНТИ. — Новосибирск: Наука. Сиб. отд-ние, 1991.
33. *Пушкин В.Г., Урсул А.Д.* Информатика, кибернетика, интеллект. — Кишинев: Штиинца, 1989.
34. *Ракитов А.И.* Информационная революция: Наука, экономика, технология. — М.: Изд-во ИНИОН РАН, 1993.
35. *Ракитов А.И.* Философия компьютерной революции. — М.: Мысль, 1991.
36. *Симонович С.В., Евсеев Г.А., Алексеев А.Г.* Общая информатика. — М.: АСТ-ПРЕСС, 1998.
37. Создание автоматизированных систем информационного обеспечения научных исследований. Сб. науч. тр. — Киев: ИК, 1986.
38. *Страссман П.А.* Информация в век электроники: Пер. с англ. — М.: Экономика, 1987.
39. *Суханов А.П.* Информация и прогресс. — Новосибирск: Наука. Сиб. отд-ние, 1988.
40. *Сухина В.Ф.* Человек в мире информатики. — М.: Радио и связь, 1992.
41. *Урсул А.Д.* Информатизация общества (Введение в социальную информатику): Учеб. пособие. — М.: Высш. шк., 1990.
42. *Цымбал В.П.* Информатика и индустрия информации. — Киев: Выща шк., 1989.
43. *Черри К.* Человек и информация: Пер. с англ. — М.: Связь, 1979.
44. *Шнейдеров В.С.* Занимательная информатика. — СПб.: Политехника, 1994.
45. *Юзвишин И.И.* Информациология, или Закономерности информационных процессов и технологий в микро- и макромирах Вселенной. — М.: Радио и связь, 1996.

§ 2. ИНФОРМАЦИЯ, ЕЕ ВИДЫ И СВОЙСТВА

Краткие сведения

Понятие «информация» является одним из фундаментальных в современной науке вообще и базовым для информатики. Информацию наряду с веществом и энергией рассматривают в качестве важнейшей сущности мира, в котором мы живем. Однако если задаться целью формально определить понятие «информация», то сделать это будет чрезвычайно сложно.

В простейшем бытовом понимании с термином «информация» обычно ассоциируются некоторые сведения, данные, знания и т.п. Информация передается в виде сообщений, определяющих форму и представление передаваемой информации. Примерами сообщений являются музыкальное произведение; телепередача; команды регулировщика на перекрестке; текст, распечатанный на принтере; данные, полученные в результате работы составленной вами программы, и т.д. При этом предполагается, что имеются «источник информации» и «получатель информации».

Сообщение от источника к получателю передается посредством какой-нибудь среды, являющейся в таком случае «каналом связи». Так, при передаче речевого сообщения в качестве канала связи можно рассматривать воздух, в котором распространяются звуковые волны, а в случае передачи письменного сообщения (например, текста, распечатанного на принтере) каналом сообщения можно считать лист бумаги, на котором напечатан текст.

Человеку свойственно субъективное восприятие информации через некоторый набор ее свойств: важность, достоверность, своевременность, доступность, «больше-меньше» и т.д. Использование терминов «больше информации» или «меньше информации» подразумевает некую возможность ее измерения (или хотя бы количественного соотнесения). При субъективном восприятии измерение информации возможно лишь в виде установления некоторой субъективной порядковой шкалы для оценки «больше-меньше». При объективном измерении количества информации следует заведомо отрешиться от восприятия ее с точки зрения субъективных свойств, примеры которых перечислены выше. Более того, не исключено, что не всякая информация будет иметь объективно измеряемое количество.

Чтобы сообщение было передано от источника к получателю, необходима некоторая материальная субстанция — носитель информации. Сообщение, передаваемое с помощью носителя, — сигнал. В общем случае сигнал — это изменяющийся во времени физический процесс. Та из характеристик процесса, которая используется для представления сообщений, называется параметром сигнала.

В случае, когда параметр сигнала принимает последовательное во времени конечное число значений (при этом все они могут быть пронумерованы), сигнал называется дискретным, а сообщение, передаваемое с помощью таких сигналов, — дискретным сообщением. Если же источник вырабатывает непрерывное сообщение (соответственно параметр сигнала — непрерывная функция от времени), то соответствующая информация называется непрерывной. Примеры дискретного сообщения — текст книги, непрерывного сообщения — человеческая речь, передаваемая модулированной звуковой волной; параметром сигнала в последнем случае является давление, создаваемое этой волной в точке нахождения приемника — человеческого уха.

Непрерывное сообщение может быть представлено непрерывной функцией, заданной на некотором интервале. Непрерывное сообщение можно преобразовать

в дискретное (такая процедура называется дискретизацией). Из бесконечного множества значений параметра сигнала выбирается их определенное число, которое приближенно может характеризовать остальные значения. Для этого область определения функции разбивается на отрезки равной длины и на каждом из этих отрезков значение функции принимается постоянным и равным, например, среднему значению на этом отрезке. В итоге получим конечное множество чисел. Таким образом, любое непрерывное сообщение может быть представлено как дискретное, иначе говоря, последовательностью знаков некоторого алфавита.

Возможность дискретизации непрерывного сигнала с любой желаемой точностью (для возрастания точности достаточно уменьшить шаг) принципиально важна с точки зрения информатики. Компьютер — цифровая машина, т.е. внутреннее представление информации в нем дискретно. Дискретизация входной информации (если она непрерывна) позволяет сделать ее пригодной для компьютерной обработки.

Единицы количества информации: вероятностный и объемный подходы

Определить понятие «количество информации» довольно сложно. В решении этой проблемы существуют два основных подхода. Исторически они возникли почти одновременно. В конце 40-х годов XX века один из основоположников кибернетики, американский математик Клод Шеннон, развил вероятностный подход к измерению количества информации, а работы по созданию ЭВМ привели к «объемному» подходу.

Вероятностный подход

Рассмотрим в качестве примера опыт, связанный с бросанием правильной игральной кости, имеющей N граней. Результаты данного опыта могут быть следующие: выпадение грани с одним из следующих знаков: 1, 2, ..., N.

Введем в рассмотрение численную величину, измеряющую неопределенность — **энтропию** (обозначим ее H). Согласно развитой теории, в случае равновероятного выпадания каждой из граней величины N и H связаны между собой **формулой Хартли** $H = \log_2 N$.

Важным при введении какой-либо величины является вопрос о том, что принимать за единицу ее измерения. Очевидно, H будет равно единице при $N = 2$. Иначе говоря, в качестве единицы принимается количество информации, связанное с проведением опыта, состоящего в получении одного из двух равновероятных исходов (примером такого опыта может служить бросание монеты, при котором возможны два исхода: «орел», «решка»). Такая единица количества информации называется «бит».

В случае, когда вероятности P_i результатов опыта (в примере, приведенном выше, — бросания игральной кости) неодинаковы, имеет место **формула Шеннона** $H = -\sum_{i-1}^{N} P_i \times \log_2 P_i$. В случае равновероятности событий $P_i = \frac{1}{N}$, и формула Шеннона переходит в формулу Хартли.

В качестве примера определим количество информации, связанное с появлением каждого символа в сообщениях, записанных на русском языке. Будем считать, что русский алфавит состоит из 33 букв и знака «пробел» для разделения слов. По формуле Хартли $H = \log_2 34 \approx 5{,}09$ бит.

Однако в словах русского языка (равно как и в словах других языков) различные буквы встречаются неодинаково часто. Ниже приведена табл. 1.1 вероятностей частоты употребления различных знаков русского алфавита, полученная на основе анализа очень больших по объему текстов.

Воспользуемся для подсчета H формулой Шеннона: $H \approx 4{,}72$ бит. Полученное значение H, как и можно было предположить, меньше вычисленного ранее. Величина H, вычисляемая по формуле Хартли, является максимальным количеством информации, которое могло бы приходиться на один знак.

Аналогичные подсчеты H можно провести и для других языков, например, использующих латинский алфавит — английского, немецкого, французского и др. (26 различных букв и «пробел»). По формуле Хартли получим $H = \log_2 27 \approx 4{,}76$ бит.

Таблица 1.1

Частотность букв русского языка

i	Символ	$P(i)$	i	Символ	$P(i)$	i	Символ	$P(i)$
1	_	0,175	12	Л	0,035	23	Б	0,014
2	О	0,090	13	К	0,028	24	Г	0,012
3	Е	0,072	14	М	0,026	25	Ч	0,012
4	Ё	0,072	15	Д	0,025	26	Й	0,010
5	А	0,062	16	П	0,023	27	Х	0,009
6	И	0,062	17	У	0,021	28	Ж	0,007
7	Т	0,053	18	Я	0,018	29	Ю	0,006
8	Н	0,053	19	Ы	0,016	30	Ш	0,006
9	С	0,045	20	З	0,016	31	Ц	0,004
10	Р	0,040	21	Ь	0,014	32	Щ	0,003
11	В	0,038	22	Ъ	0,014	33	Э	0,003
						34	Ф	0,002

Рассмотрим алфавит, состоящий из двух знаков 0 и 1. Если считать, что со знаками 0 и 1 в двоичном алфавите связаны одинаковые вероятности их появления ($P(0) = P(1) = 0{,}5$), то количество информации на один знак при двоичном кодировании будет равно $H = \log_2 2 = 1$ бит.

Таким образом, количество информации (в битах), заключенное в двоичном слове, равно числу двоичных знаков в нем.

Объемный подход

В двоичной системе счисления знаки 0 и 1 называют **битами** (*bit* — от английского выражения *Binary digiTs* — двоичные цифры). В компьютере бит является наименьшей возможной единицей информации. Объем информации, записанной двоичными знаками в памяти компьютера или на внешнем носителе информации, подсчитывается просто по числу требуемых для такой записи двоичных символов. При этом, в частности, невозможно нецелое число битов (в отличие от вероятностного подхода).

Для удобства использования введены и более крупные, чем бит, единицы количества информации. Так, двоичное слово из восьми знаков содержит один **байт**

информации. 1024 байта образуют килобайт (Кбайт), 1024 килобайта — мегабайт (Мбайт), а 1024 мегабайта — гигабайт (Гбайт).

Между вероятностным и объемным количеством информации соотношение неоднозначное. Далеко не всякий текст, записанный двоичными символами, допускает измерение объема информации в вероятностном (кибернетическом) смысле, но заведомо допускает его в объемном. Далее, если некоторое сообщение допускает измеримость количества информации в обоих смыслах, то это количество не обязательно совпадает, при этом кибернетическое количество информации не может быть больше объемного.

В прикладной информатике практически всегда количество информации понимается в объемном смысле.

Как ни важно измерение информации, нельзя сводить к нему все связанные с этим понятием проблемы. При анализе информации социального (в широким смысле) происхождения на первый план могут выступить такие ее свойства, как истинность, своевременность, ценность, полнота и т.д. Их невозможно оценить в терминах «уменьшение неопределенности» (вероятностный подход) или числа символов (объемный подход). Обращение к качественной стороне информации породило иные подходы к ее оценке. При **аксиологическом** подходе стремятся исходить из ценности, практической значимости информации, т.е. из качественных характеристик, значимых в социальной системе. При **семантическом** подходе информация рассматривается с точки зрения как формы, так и содержания. При этом информацию связывают с **тезаурусом**, т.е. полнотой систематизированного набора данных о предмете информации. Отметим, что эти подходы не исключают количественного анализа, но он становится существенно сложнее и должен базироваться на современных методах математической статистики.

Понятие информации нельзя считать лишь техническим, междисциплинарным и даже наддисциплинарным термином. Информация — это фундаментальная философская категория. Дискуссии ученых о философских аспектах информации надежно показали несводимость информации ни к одной из этих категорий. Концепции и толкования, возникающие на пути догматических подходов, оказываются слишком частными, односторонними, не охватывающими всего объема этого понятия.

Попытки рассмотреть категорию информации с позиций основного вопроса философии привели к возникновению двух противостоящих концепций — **функциональной** и **атрибутивной.** «Атрибутисты» квалифицируют информацию как свойство всех материальных объектов, т.е. как атрибут материи. «Функционалисты» связывают информацию лишь с функционированием сложных, самоорганизующихся систем.

Можно попытаться дать философское определение информации с помощью указания на связь определяемого понятия с категориями **отражения** и **активности**. Информация есть содержание образа, формируемого в процессе отражения. Активность входит в это определение в виде представления о формировании некоего образа в процессе отражения некоторого субъект-объектного отношения. При этом не требуется указания на связь информации с материей, поскольку как субъект, так и объект процесса отражения могут принадлежать как к материальной, так и к духовной сфере социальной жизни. Однако существенно подчеркнуть, что материалистическое решение основного вопроса философии требует признания необходимости существования материальной среды — носителя информации в процессе такого отражения. Итак, информацию следует трактовать как имманентный (неотъемлемо присущий) атрибут материи, необходимый момент ее самодвиже-

ния и саморазвития. Эта категория приобретает особое значение применительно к высшим формам движения материи — биологической и социальной.

Известно большое количество работ, посвященных ***физической трактовке*** информации. Эти работы в значительной мере построены на основе аналогии формулы Больцмана, описывающей энтропию статистической системы материальных частиц, и формулы Хартли. Соответствующие материалы можно найти в литературе, отраженной в приведенном ниже перечне.

Информацию следует считать особым ***видом ресурса***, при этом имеется в виду толкование «ресурса» как запаса неких знаний материальных предметов или энергетических, структурных или каких-либо других характеристик предмета. В отличие от ресурсов, связанных с материальными предметами, информационные ресурсы являются неистощимыми и предполагают существенно иные методы воспроизведения и обновления, чем материальные ресурсы. В связи с таким взглядом центральными становятся следующие свойства информации: *запоминаемость*, *передаваемость*, *преобразуемость*, *воспроизводимость*, *стираемость*.

Подводя итог сказанному, отметим, что предпринимаются (но отнюдь не завершены) усилия ученых, представляющих самые разные области знания, построить единую теорию, которая призвана формализовать понятие информации и информационного процесса, описать превращения информации в процессах самой разной природы. Движение информации есть сущность процессов управления, которые суть проявление имманентной активности материи, ее способности к самодвижению. С момента возникновения кибернетики управление рассматривается применительно ко всем формам движения материи, а не только к высшим (биологической и социальной). Многие проявления движения в неживых — искусственных (технических) и естественных (природных) системах также обладают общими признаками управления, хотя их исследуют в химии, физике, механике в энергетической, а не в информационной системе представлений. Информационные аспекты в таких системах составляют предмет новой междисциплинарной науки — синергетики.

Высшей формой информации, проявляющейся в управлении в социальных системах, являются знания. Это наддисциплинарное понятие, широко используемое в педагогике и исследованиях по искусственному интеллекту, также претендует на роль важнейшей философской категории. В философском плане познание следует рассматривать как один из функциональных аспектов управления. Такой подход открывает путь к системному пониманию генезиса процессов познания, его основ и перспектив.

Контрольные вопросы

1. Какая форма представления информации — непрерывная или дискретная — приемлема для компьютеров и почему?
2. В чем состоит процедура дискретизации непрерывной информации?
3. Какие определения понятия «информация» Вы знаете?
4. Назовите основные свойства информации.
5. Каким образом возникает, хранится, обрабатывается и передается информация?
6. Какая форма представления информации используется в информатике?
7. Какие виды информационных сигналов Вы знаете?
8. В чем преимущества дискретного представления информации?
9. Может ли человек передать информацию машине? Каким образом? А наоборот?
10. Что такое количество информации?

11. Какой принцип положен в основу измерения количества информации?
12. Каким образом определяется единица количества информации при кибернетическом подходе?
13. Как определяется количество информации в знаковых сообщениях?
14. Каковы основные единицы измерения количества информации?
15. Приведите объемы памяти известных Вам носителей информации.
16. Как определяется понятие энтропии? Как она связана с информацией?
17. Какова связь между энтропией, негэнтропией и информацией?
18. Какие свойства социальной информации важны при ее качественном анализе?
19. Определите информацию как философскую категорию.
20. В чем состоит функциональная концепция информации?
21. В чем состоит атрибутивная концепция информации?
22. Как связана информация с категориями отражения и активности?
23. Расскажите об информационной трактовке социальных процессов.
24. Каковы основные свойства информации как особого вида ресурса?
25. Расскажите о движении информации с точки зрения процессов управления.

Проблемные вопросы

1. Приведите примеры передачи, хранения и обработки информации в природе, технической и общественной деятельности человека.
2. Какие проблемы по хранению и обработке информации решают современные информационные технологии и какие создают?
3. Дайте определение меры неопределенности. Проиллюстрируйте это понятие.
4. Почему информация является философской категорией?
5. Почему нельзя однозначно сопоставить информацию и энтропию?
6. Почему обе концепции информации — как функциональная, так и атрибутивная — являются неполными?

Темы для рефератов

1. Проблема информации в современной науке.
2. Передача информации.
3. Дискретизация непрерывных сообщений.
4. Субъективные свойства информации.
5. Аналоговые ЭВМ.
6. Непрерывная и дискретная информация.
7. Информация и энтропия.
8. Вероятность и информация.
9. Проблема измерения информации.
10. Ценностный подход к информации.
11. Семантическая информация.
12. Атрибутивная и функциональная концепции информации.
13. Информация и эволюция живой природы.
14. Информационные процессы в неживой природе.
15. Отражение и информация.
16. Материя, энергия и информация.
17. Синергетика и информация.
18. Познание, мышление и информация.

19. Картина мира и информация.
20. Свойства информационных ресурсов.
21. Информация и сознание.

Темы семинарских занятий

1. Различные уровни представлений об информации.
2. Непрерывная и дискретная информация.
3. Единицы количества информации: вероятностный и объемный подходы.
4. Философия и информация.
5. Информация и физический мир.

Задачи и упражнения

1. Подсчитайте количество информации, приходящейся на один символ, в следующем тексте экономического содержания:

Организационно-правовые формы предприятий в своей основе определяют форму их собственности, то есть кому принадлежит предприятие, его основные фонды, оборотные средства, материальные и денежные ресурсы. В зависимости от формы собственности в России в настоящее время различают три основные формы предпринимательской деятельности: частную, коллективную и контрактную.

Указание: составьте таблицу, аналогичную табл. 1.1, определив вероятность каждого символа в тексте как отношение количества одинаковых символов каждого значения ко всему числу символов в тексте. Затем по формуле Шеннона подсчитайте количество информации, приходящейся на один символ.

2. Подсчитайте количество информации, приходящейся на один символ, в следующем тексте технического содержания:

Общая технологическая схема изготовления сплавного транзистора напоминает схему изготовления диода, за исключением того, что в полупроводниковую пластинку производят вплавление двух навесок примесей с двух сторон. Вырезанные из монокристалла германия или кремния пластинки шлифуют и травят до необходимой толщины.

3. Подсчитайте количество информации, приходящейся на один символ, в следующем тексте исторического содержания:

С конца пятнадцатого столетия в судьбах Восточной Европы совершается переворот глубокого исторического значения. На сцену истории Европы выступает новая крупная политическая сила — Московское государство. Объединив под своей властью всю северо-восточную Русь, Москва напряженно работает над закреплением добытых политических результатов и во внутренних, и во внешних отношениях.

4. Подсчитайте количество информации, приходящейся на один символ, в следующем тексте естественно-научного содержания:

Новые данные о физиологической потребности организма человека в пищевых веществах и энергии, а также выяснение закономерностей ассимиляции пищи в условиях нарушенного болезнью обмена веществ на всех этапах метаболического конвейера позволили максимально сбалансировать химический состав диет и их энергетическую ценность.

5. Подсчитайте количество информации, приходящейся на один символ, в следующем художественно-литературном тексте:

С любопытством стал я рассматривать сборище. Пугачев на первом месте сидел, облокотясь на стол и подпирая черную бороду своим широким кулаком. Черты лица

его, правильные и довольно приятные, не изъявляли ничего свирепого. Все обходились между собою как товарищи и не оказывали никакого особенного предпочтения своему предводителю.

Дополнительная литература

1. *Абдеев В.Ф.* Философия информационной цивилизации. — М.: ВЛАДОС, 1994.
2. *Абрамов Ю.Ф.* Картина мира и информация. — Иркутск: Изд-во Иркут. ун-та, 1988.
3. *Аветисян Р.Д., Аветисян Д.В.* Теоретические основы информатики. — М.: РГГУ, 1997.
4. *Агеев В.М.* Теория информации и кодирования: Дискретизация и кодирование измерительной информации. — М.: МАИ, 1977.
5. *Айламазян А.К., Стась Е.В.* Информатика и теория развития. — М.: Наука, 1989.
6. *Антонов А.В.* Информация: Восприятие и понимание. — Киев: Наук. думка, 1988.
7. *Бауэр Ф.Л., Гоос Г.* Информатика. Вводный курс: Пер. с нем. — М.: Мир, 1976.
8. *Бриллюэн Л.* Наука и теория информации: Пер. с англ. — М.: Физматгиз, 1960.
9. *Бриллюэн Л.* Научная неопределенность и информация: Пер. с англ. — М.: Мир, 1966.
10. *Брой М.* Информатика: В 3 т. Т.1. Основополагающее введение. — М.: Диалог-МИФИ, 1996.
11. *Быховский А.* Информация и живые организмы // Наука и жизнь. — 1976. — №8.
12. *Величкин А.И.* Теория дискретной передачи непрерывных сообщений. — М.: Сов. радио, 1970.
13. *Гришкин И.И.* Понятие информации. Логико-методологический аспект. — М.: Наука, 1973.
14. *Дмитриев В.И.* Прикладная теория информации. — М.: Наука, 1989.
15. *Дубровский Д.И.* Информация, сознание, мозг. — М.: Высш. шк., 1980.
16. *Ефимов А.Н.* Информация: ценность, старение, рассеяние. — М.: Знание, 1978.
17. *Жалдан М.И., Квитко А.Н.* Теория вероятностей с элементами информатики. Практикум. — Киев: Выща шк., 1989.
18. *Заличев Н.Н.* Энтропия информации и сущность жизни. — М.: Радиоэлектроника, 1995.
19. *Иезуитов А.* О философских основах информатики // Педагогическая информатика. — 1998. — №4. — С.54—65.
20. *Коган И.М.* Прикладная теория информации. — М.: Радио и связь, 1981.
21. *Колмогоров А.Н.* Теория информации и теория алгоритмов. — М.: Наука, 1987.
22. *Котова Е.В.* Энергия и информация. — Киев: Выща шк., 1981.
23. *Кузьмин И.В., Кедрус В.А.* Основы теории информации и кодирования. — Киев: Выща шк., 1986.
24. *Мазур М.* Качественная теория информации. — М.: Мир, 1974.
25. *Орлов В.А., Филиппов Л.И.* Теория информации в упражнениях и задачах. — М.: Высш. шк., 1976.
26. *Павлов Т.* Информация, отражение, творчество. — М.: Прогресс, 1967.

27. *Петрушенко Л.А.* Самодвижение материи в свете кибернетики. — М.: Наука, 1971.
28. *Полтавский Р.П.* Термодинамика информационных процессов. — М.: Наука, 1981.
29. *Пушкин В.Г., Урсул А.Д.* Информатика, кибернетика, интеллект. — Кишинев: Штиинца,1989.
30. *Седов Е.А.* Эволюция и информация. — М.: Наука, 1976.
31. *Стратонович Р.Л.* Теория информации. — М.: Сов. радио, 1975.
32. *Суханов А.П.* Мир информации. — М.: Мысль, 1986.
33. *Тюхтин В.С.* Теория отражения в свете современной науки. — М.: Наука, 1971.
34. *Урсул А.Д.* Информация и мышление. — М.: Знание, 1970.
35. *Урсул А.Д.* Проблема информации в современной науке. Философские очерки. — М.: Наука, 1975.
36. *Цымбал В.П.* Задачник по теории информации и кодирования. — Киев: Выща шк., 1976.
37. *Чернавский Д.С.* Синергетика и информация. — М.: Знание, 1990.
38. *Шеннон К.* Работы по теории информации. — М.: Изд-во иностр. лит., 1966.
39. *Шредингер Э.* Что такое жизнь? С точки зрения физики? / Пер. с англ. — М.: Изд-во иностр. лит., 1947.
40. *Юзвишин И.И.* Информациология, или Закономерности информационных процессов и технологий в микро- и макромирах Вселенной. — М.: Радио и связь, 1996.
41. *Яглом А.М., Яглом И.М.* Вероятность и информация. — М.: Наука, 1973.
42. *Янков М.* Материя и информация. — М.: Прогресс, 1979.

§ 3. СИСТЕМЫ СЧИСЛЕНИЯ

Рекомендации по проведению занятий

Системы счисления — одна из традиционных тем курса информатики, восходящих к программированию ЭВМ первых поколений в машинных кодах. В настоящее время данная тема сохраняет свое значение как весьма типичный случай кодирования информации, а также в связи с широким использованием шестнадцатеричных обозначений в машинно-ориентированных разделах программирования. Знание систем счисления полезно для понимания представления данных в памяти ЭВМ и операций над ними. Системы счисления (особенно по основанию 10) достаточно подробно изучаются в курсах математики и информатики средней общеобразовательной школы. В данном курсе эта тема предполагает повторение уже известных сведений, специализацию в отношении систем счисления по основанию 16, 8 и 2, а также обобщение в плане кодирования информации.

Целесообразно проведение семинарского занятия, подготовка рефератов, посвященных истории и значению позиционных систем счисления. Особое внимание следует уделить формированию стабильных навыков чтения и записи чисел в шестнадцатеричной системе. Полезным является и знакомство с различными приемами перевода чисел в системы счисления по основанию 2, 8 и 16, в том числе с помощью калькулятора или компьютера и встроенного интерпретатора языка BASIC.

Краткие сведения

Перевод чисел из одной позиционной системы счисления в другую. Арифметические операции

При переводе чисел из десятичной системы счисления в систему с основанием $P > 1$ обычно используют следующий алгоритм:

1) если переводится целая часть числа, то она делится на P, после чего запоминается остаток от деления. Полученное частное вновь делится на P, остаток запоминается. Процедура продолжается до тех пор, пока частное не станет равным нулю. Остатки от деления на P выписываются в порядке, обратном их получению;

2) если переводится дробная часть числа, то она умножается на P, после чего целая часть запоминается и отбрасывается. Вновь полученная дробная часть умножается на P и т.д. Процедура продолжается до тех пор, пока дробная часть не станет равной нулю. Целые части выписываются после двоичной запятой в порядке их получения. Результатом может быть либо конечная, либо периодическая двоичная дробь. Поэтому, когда дробь является периодической, приходится обрывать умножение на каком-либо шаге и довольствоваться приближенной записью исходного числа в системе с основанием P.

Пример 1. Перевести данное число из десятичной системы счисления в двоичную (получить пять знаков после запятой в двоичном представлении).

а) $464_{(10)}$; б) $380{,}1875_{(10)}$; в) $115{,}94_{(10)}$.

Решение:

а)			б)						в)					
	464	0		380	0			1875		115	1			94
	232	0		190	0	0		375		57	1	1		88
	116	0		95	1	0		75		28	0	1		76
	58	0		47	1	1		5		14	0	1		52
	29	1		23	1	1		0		7	1	1		04
	14	0		11	1					3	1	0		08
	7	1		5	1					1	1	0		16
	3	1		2	0									
	1	1		1	1									

а) $464_{(10)} = 111010000_{(2)}$; б) $380{,}1875_{(10)} = 101111100{,}0011_{(2)}$;

в) $115{,}94_{(10)} \approx 1110011{,}11110_{(2)}$

(в данном случае было получено шесть знаков после запятой, после чего результат был округлен).

Если необходимо перевести число из двоичной системы счисления в систему счисления, основанием которой является степень двойки, достаточно объединить цифры двоичного числа в группы по столько цифр, каков показатель степени, и использовать приведенный ниже алгоритм. Например, если перевод осуществляется в восьмеричную систему, то группы будут содержать три цифры ($8 = 2^3$). В целой части числа группировка производится справа налево, в дробной части — слева направо. Если в последней группе недостает цифр, дописываются нули: в целой части — слева, в дробной — справа. Затем каждая группа заменяется соответствующей цифрой новой системы. Соответствия приведены в таблице.

P	Соответствия															
2	00	01	10	11												
4	0	1	2	3												
2	00	00	01	01	10	10	11	11								
	0	1	0	1	0	1	0	1								
8	0	1	2	3	4	5	6	7								
2	00	00	00	00	01	01	01	01	10	10	10	10	11	11	11	11
	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Переведем из двоичной системы в шестнадцатеричную число $1111010101,11_{(2)}$.

$\underline{001111010101},\underline{1100}_{(2)} = 3D5,C_{(16)}$.

При переводе чисел из системы счисления с основанием P в десятичную систему счисления необходимо пронумеровать разряды целой части справа налево, начиная с нулевого, и дробной части, начиная с разряда сразу после запятой, слева направо (начальный номер — 1). Затем вычислить сумму произведений соответствующих значений разрядов на основание системы счисления в степени, равной номеру разряда. Это и есть представление исходного числа в десятичной системе счисления.

Пример 2. Перевести данное число в десятичную систему счисления:

а) $1000001_{(2)}$.

$1000001_{(2)} = 1\cdot 2^6 + 0\cdot 2^5 + 0\cdot 2^4 + 0\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 64 + 1 = 65_{(10)}$.

Замечание. Если в каком-либо разряде стоит нуль, то соответствующее слагаемое можно опускать;

б) $1000011111,0101_{(2)}$.

$1000011111,0101_{(2)} = 1\cdot 2^9 + 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-2} + 1\cdot 2^{-4} =$
$= 512 + 16 + 8 + 4 + 2 + 1 + 0{,}25 + 0{,}0625 = 543{,}3125_{(10)}$;

в) $1216,04_{(8)}$.

$1216,04_{(8)} = 1\cdot 8^3 + 2\cdot 8^2 + 1\cdot 8^1 + 6\cdot 8^0 + 4\cdot 8^{-2} = 512 + 128 + 8 + 6 + 0{,}0625 = 654{,}0625_{(10)}$;

г) $29A,5_{(16)}$.

$29A,5_{(16)} = 2\cdot 16^2 + 9\cdot 16^1 + 10\cdot 16^0 + 5\cdot 16^{-1} = 512 + 144 + 10 + 0{,}3125 = 656{,}3125_{(10)}$.

Для выполнения арифметических операций в системе счисления с основанием P необходимо иметь соответствующие таблицы сложения и умножения.

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

×	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Пример 3. Сложить числа:

а) $10000000100_{(2)} + 111000010_{(2)} = 10111000110_{(2)}$;

б) $223{,}2_{(8)} + 427{,}54_{(8)} = 652{,}74_{(8)}$;

в) $3B3{,}6_{(16)} + 38B{,}4_{(16)} = 73E{,}A_{(16)}$.

$$\begin{array}{r} 10000000100 \\ +\ 111000010 \\ \hline 10111000110 \end{array} \qquad \begin{array}{r} 223{,}2 \\ +\ 457{,}54 \\ \hline 652{,}74 \end{array} \qquad \begin{array}{r} 3B3{,}6 \\ +\ 38B{,}4 \\ \hline 73E{,}A \end{array}$$

Пример 4. Выполнить вычитание:

а) $1100000011{,}011_{(2)} - 101010111{,}1_{(2)} = 110101011{,}111_{(2)}$;

б) $1510{,}2_{(8)} - 1230{,}54_{(8)} = 257{,}44_{(8)}$;

в) $27D{,}D8_{(16)} - 191{,}2_{(16)} = EC{,}B8_{(16)}$.

$$\begin{array}{r} 1100000011{,}011 \\ -\ 101010111{,}1 \\ \hline 110101011{,}111 \end{array} \qquad \begin{array}{r} 1510{,}2 \\ -\ 1230{,}54 \\ \hline 257{,}44 \end{array} \qquad \begin{array}{r} 27D{,}D8 \\ -\ 191{,}2 \\ \hline EC{,}B8 \end{array}$$

Пример 5. Выполнить умножение:

а) $100111_{(2)} \times 1000111_{(2)} = 101011010001_{(2)}$;

б) $1170{,}64_{(8)} \times 46{,}3_{(8)} = 57334{,}134_{(8)}$;

в) $61{,}A_{(16)} \times 40{,}D_{(16)} = 18B7{,}52_{(16)}$.

$$\begin{array}{r} 100111 \\ \times\ 1000111 \\ \hline 100111 \\ +\ 100111 \\ 100111 \\ 100111 \\ \hline 101011010001 \end{array} \qquad \begin{array}{r} 1170{,}64 \\ \times\ 46{,}3 \\ \hline 355234 \\ +\ 732470 \\ 474320 \\ \hline 57334{,}134 \end{array} \qquad \begin{array}{r} 61{,}A \\ \times\ 40{,}D \\ \hline 4F52 \\ +\ 1868 \\ \hline 18B7{,}52 \end{array}$$

Контрольные вопросы

1. Какие системы счисления называют позиционными, а какие — непозиционными? Приведите примеры.
2. Что называется основанием системы счисления?
3. Почему для вычислительной техники особенно важна система счисления по основанию 2?
4. Почему произошел переход от двоичных к шестнадцатеричным обозначениям в архитектуре ЭВМ?
5. Какие способы перевода целых десятичных чисел в двоичные и обратно Вы знаете?
6. Каковы правила выполнения арифметических операций над числами в двоичном представлении?

7. Как переводить целые числа из двоичного представления в восьмеричное и шестнадцатеричное представления и обратно?

8. Какое двоичное представление отрицательных целых чисел используется в вычислительной технике?

9. Как представляются в вычислительной технике действительные числа (числа с плавающей запятой)?

10. Дайте определение системы счисления. Назовите и охарактеризуйте свойства системы счисления.

11. Какие символы используются для записи чисел в двоичной системе счисления, восьмеричной, шестнадцатеричной?

12. Чему равны веса разрядов слева от точки, разделяющей целую и дробную части, в двоичной системе счисления (восьмеричной, шестнадцатеричной)?

13. Чему равны веса разрядов справа от точки, разделяющей целую и дробную части, в двоичной системе счисления (восьмеричной, шестнадцатеричной)?

Темы для рефератов

1. Системы счисления Древнего мира.

2. Римская система счисления. Представление чисел в ней и решение арифметических задач.

3. История десятичной системы счисления.

4. Применение в цифровой электронике двоичной, восьмеричной и шестнадцатеричной систем счисления.

Темы семинарских занятий

1. Значение систем счисления для прогресса математики и вычислительной техники.

2. Перевод чисел в двоичную, восьмеричную и шестнадцатеричную системы счисления и арифметические операции над ними.

Задачи и упражнения

1. Переведите в двоичную систему десятичные числа 231, 564, 1023, 4096.

2. Переведите в десятичную систему двоичные числа 10011101, 1100101001110110, 101111001011001011100111.

3. Какое максимальное число можно представить в двоичной системе пятнадцатью цифрами?

4. Переведите в восьмеричную систему двоичные числа 111001, 101110111, 110010101110.

5. Переведите в двоичную систему восьмеричные числа 324, 2367, 53621.

6. Переведите в шестнадцатеричную систему двоичные числа 11010011, 101101101011, 1001011100111101.

7. Переведите в двоичную систему шестнадцатеричные числа 3A, D14, AF4C, F55DD.

8. Сложите, вычтите из большего меньшее, перемножьте и разделите первое на второе числа в двоичном представлении 1101001110011101 и 1001011010110111.

Лабораторная работа

Время выполнения 6 часов.

Задания к лабораторной работе

1. Переведите данное число из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную системы счисления.
2. Переведите данное число в десятичную систему счисления.
3. Сложите числа.
4. Выполните вычитание.
5. Выполните умножение.

Примечание. В заданиях 3—5 проверьте правильность вычислений переводом исходных данных и результатов в десятичную систему счисления. В задании 1 д) получите пять знаков после запятой в двоичном представлении.

Вариант 1

1. а) $860_{(10)}$; б) $785_{(10)}$; в) $149{,}375_{(10)}$; г) $953{,}25_{(10)}$; д) $228{,}79_{(10)}$.
2. а) $1001010_{(2)}$; б) $1100111_{(2)}$; в) $110101101{,}00011_{(2)}$;
   г) $111111100{,}0001_{(2)}$; д) $775{,}11_{(8)}$; е) $294{,}3_{(16)}$.
3. а) $1101100000_{(2)} + 10110110_{(2)}$; б) $101110111_{(2)} + 1000100001_{(2)}$;
   в) $1001000111{,}01_{(2)} + 100001101{,}101_{(2)}$; г) $271{,}34_{(8)} + 1566{,}2_{(8)}$;
   д) $65{,}2_{(16)} + 3CA{,}8_{(16)}$.
4. а) $1011001001_{(2)} - 1000111011_{(2)}$; б) $1110000110_{(2)} - 101111101_{(2)}$;
   в) $101010000{,}10111_{(2)} - 11001100{,}01_{(2)}$; г) $731{,}6_{(8)} - 622{,}6_{(8)}$;
   д) $22D{,}1_{(16)} - 123{,}8_{(16)}$.
5. а) $1011001_{(2)} \cdot 1011011_{(2)}$; б) $723{,}1_{(8)} \cdot 50{,}2_{(8)}$; в) $69{,}4_{(16)} \cdot A{,}B_{(16)}$.

Вариант 2

1. а) $250_{(10)}$; б) $757_{(10)}$; в) $711{,}25_{(10)}$; г) $914{,}625_{(10)}$; д) $261{,}78_{(10)}$.
2. а) $1111000_{(2)}$; б) $1111000000_{(2)}$; в) $111101100{,}01101_{(2)}$;
   г) $100111100{,}1101_{(2)}$; д) $1233{,}5_{(8)}$; е) $2B3{,}F4_{(16)}$.
3. а) $1010101_{(2)} + 10000101_{(2)}$; б) $1111011101_{(2)} + 101101000_{(2)}$;
   в) $100100111{,}001_{(2)} + 100111010{,}101_{(2)}$; г) $607{,}54_{(8)} + 1620{,}2_{(8)}$;
   д) $3BF{,}A_{(16)} + 313{,}A_{(16)}$.
4. а) $1001000011_{(2)} - 10110111_{(2)}$; б) $111011100_{(2)} - 10010100_{(2)}$;
   в) $1100110110{,}0011_{(2)} - 11111110{,}01_{(2)}$; г) $1360{,}14_{(8)} - 1216{,}4_{(8)}$;
   д) $33B{,}6_{(16)} - 11B{,}4_{(16)}$.
5. а) $11001_{(2)} \cdot 1011100_{(2)}$; б) $451{,}2_{(8)} \cdot 5{,}24_{(8)}$; в) $2B{,}A_{(16)} \cdot 36{,}6_{(16)}$.

Вариант 3

1. а) $759_{(10)}$; б) $265_{(10)}$; в) $79{,}4375_{(10)}$; г) $360{,}25_{(10)}$; д) $240{,}25_{(10)}$.
2. а) $1001101_{(2)}$; б) $10001000_{(2)}$; в) $100111001{,}01_{(2)}$;
   г) $1111010000{,}001_{(2)}$; д) $1461{,}15_{(8)}$; е) $9D{,}A_{(16)}$.
3. а) $100101011_{(2)} + 111010011_{(2)}$; б) $1001101110_{(2)} + 1101100111_{(2)}$;
   в) $1010000100{,}1_{(2)} + 11011110{,}001_{(2)}$; г) $674{,}34_{(8)} + 1205{,}2_{(8)}$;
   д) $2FE{,}6_{(16)} + 3B{,}4_{(16)}$.
4. а) $1100110010_{(2)} - 1001101101_{(2)}$; б) $1110001100_{(2)} - 10001111_{(2)}$;
   в) $11001010{,}01_{(2)} - 1110001{,}001_{(2)}$; г) $641{,}6_{(8)} - 273{,}04_{(8)}$;

д) $3CE,B8_{(16)} - 39A,B8_{(16)}$.

5. а) $1010101_{(2)} \cdot 1011001_{(2)}$; б) $1702,2_{(8)} \cdot 64,2_{(8)}$; в) $7,4_{(16)} \cdot 1D,4_{(16)}$.

Вариант 4

1. а) $216_{(10)}$; б) $336_{(10)}$; в) $741,125_{(10)}$; г) $712,375_{(10)}$; д) $184,14_{(10)}$.
2. а) $1100000110_{(2)}$; б) $1100010_{(2)}$; в) $1011010,001_{(2)}$; г) $1010100010,001_{(2)}$; д) $1537,22_{(8)}$; е) $2D9,8_{(16)}$.
3. а) $101111111_{(2)} + 1101110011_{(2)}$; б) $10111110_{(2)} + 100011100_{(2)}$; в) $1101100011,0111_{(2)} + 1100011,01_{(2)}$; г) $666,2_{(8)} + 1234,24_{(8)}$; д) $346,4_{(16)} + 3F2,6_{(16)}$.
4. а) $1010101101_{(2)} - 110011110_{(2)}$; б) $1010001111_{(2)} - 1001001110_{(2)}$; в) $1111100100,11011_{(2)} - 101110111,011_{(2)}$; г) $1437,24_{(8)} - 473,4_{(8)}$; д) $24A,4_{(16)} - B3,8_{(16)}$.
5. а) $101011_{(2)} \cdot 100111_{(2)}$; б) $1732,4_{(8)} \cdot 34,5_{(8)}$; в) $36,4_{(16)} \cdot A,A_{(16)}$.

Вариант 5

1. а) $530_{(10)}$; б) $265_{(10)}$; в) $597,25_{(10)}$; г) $300,375_{(10)}$; д) $75,57_{(10)}$.
2. а) $101000111_{(2)}$; б) $110001001_{(2)}$; в) $1001101010,01_{(2)}$; г) $1011110100,01_{(2)}$; д) $1317,75_{(8)}$; е) $2F4,0C_{(16)}$.
3. а) $1100011010_{(2)} + 11101100_{(2)}$; б) $10111010_{(2)} + 1010110100_{(2)}$; в) $1000110111,011_{(2)} + 1110001111,001_{(2)}$; г) $1745,5_{(8)} + 1473,2_{(8)}$; д) $24D,5_{(16)} + 141,4_{(16)}$.
4. а) $1100101010_{(2)} - 110110010_{(2)}$; б) $110110100_{(2)} - 110010100_{(2)}$; в) $1101111111,1_{(2)} - 1100111110,1011_{(2)}$; г) $1431,26_{(8)} - 1040,3_{(8)}$; д) $22C,6_{(16)} - 54,2_{(16)}$.
5. а) $1001001_{(2)} \cdot 11001_{(2)}$; б) $245,04_{(8)} \cdot 112,2_{(8)}$; в) $4B,2_{(16)} \cdot 3C,3_{(16)}$.

Вариант 6

1. а) $945_{(10)}$; б) $85_{(10)}$; в) $444,125_{(10)}$; г) $989,375_{(10)}$; д) $237,73_{(10)}$.
2. а) $110001111_{(2)}$; б) $111010001_{(2)}$; в) $100110101,1001_{(2)}$; г) $1000010,01011_{(2)}$; д) $176,5_{(8)}$; е) $3D2,04_{(16)}$.
3. а) $1000011101_{(2)} + 101000010_{(2)}$; б) $100000001_{(2)} + 1000101001_{(2)}$; в) $101111011,01_{(2)} + 1000100,101_{(2)}$; г) $1532,14_{(8)} + 730,16_{(8)}$; д) $BB,4_{(16)} + 2F0,6_{(16)}$.
4. а) $1000101110_{(2)} - 1111111_{(2)}$; б) $1011101000_{(2)} - 1001000000_{(2)}$; в) $1000101001,1_{(2)} - 1111101,1_{(2)}$; г) $1265,2_{(8)} - 610,2_{(8)}$; д) $409,D_{(16)} - 270,4_{(16)}$.
5. а) $111010_{(2)} \cdot 1100000_{(2)}$; б) $1005,5_{(8)} \cdot 63,3_{(8)}$; в) $4A,3_{(16)} \cdot F,6_{(16)}$.

Вариант 7

1. а) $287_{(10)}$; б) $220_{(10)}$; в) $332,1875_{(10)}$; г) $652,625_{(10)}$; д) $315,21_{(10)}$.
2. а) $10101000_{(2)}$; б) $1101100_{(2)}$; в) $10000010000,01001_{(2)}$; г) $1110010100,001_{(2)}$; д) $1714,2_{(8)}$; е) $DD,3_{(16)}$.
3. а) $1100110_{(2)} + 1011000110_{(2)}$; б) $1000110_{(2)} + 1001101111_{(2)}$; в) $101001100,101_{(2)} + 1001001100,01_{(2)}$; г) $275,2_{(8)} + 724,2_{(8)}$; д) $165,6_{(16)} + 3E,B_{(16)}$.
4. а) $1011111111_{(2)} - 100000011_{(2)}$; б) $1110001110_{(2)} - 100001011_{(2)}$; в) $110010100,01_{(2)} - 1001110,1011_{(2)}$; г) $1330,2_{(8)} - 1112,2_{(8)}$; д) $AB,2_{(16)} - 3E,2_{(16)}$.
5. а) $110000_{(2)} \cdot 1101100_{(2)}$; б) $1560,2_{(8)} \cdot 101,2_{(8)}$; в) $6,3_{(16)} \cdot 53,A_{(16)}$.

Вариант 8

1. а) $485_{(10)}$; б) $970_{(10)}$; в) $426{,}375_{(10)}$; г) $725{,}625_{(10)}$; д) $169{,}93_{(10)}$.
2. а) $10101000_{(2)}$; б) $101111110_{(2)}$; в) $1010101{,}101_{(2)}$; г) $1111001110{,}01_{(2)}$; д) $721{,}2_{(8)}$; е) $3C9{,}8_{(16)}$.
3. а) $1010100111_{(2)} + 11000000_{(2)}$; б) $1110010010_{(2)} + 110010111_{(2)}$; в) $1111111{,}101_{(2)} + 101010101{,}101_{(2)}$; г) $1213{,}44_{(8)} + 166{,}64_{(8)}$; д) $41{,}4_{(16)} + 3CF{,}D_{(16)}$.
4. а) $1010000000_{(2)} - 1000101010_{(2)}$; б) $1011010101_{(2)} - 110011001_{(2)}$; в) $1001001010{,}11011_{(2)} - 1000111000{,}01_{(2)}$; г) $1145{,}2_{(8)} - 1077{,}5_{(8)}$; д) $380{,}1_{(16)} - 2DC{,}3_{(16)}$.
5. а) $111011_{(2)} \cdot 100000_{(2)}$; б) $511{,}2_{(8)} \cdot 132{,}4_{(8)}$; в) $68{,}4_{(16)} \cdot 37{,}8_{(16)}$.

Вариант 9

1. а) $639_{(10)}$; б) $485_{(10)}$; в) $581{,}25_{(10)}$; г) $673{,}5_{(10)}$; д) $296{,}33_{(10)}$.
2. а) $1011000011_{(2)}$; б) $100010111_{(2)}$; в) $1100101101{,}1_{(2)}$; г) $1000000000{,}01_{(2)}$; д) $1046{,}4_{(8)}$; е) $388{,}64_{(16)}$.
3. а) $1000010100_{(2)} + 1101010101_{(2)}$; б) $1011001010_{(2)} + 101011010_{(2)}$; в) $1110111000{,}101_{(2)} + 1101100011{,}101_{(2)}$; г) $1430{,}2_{(8)} + 666{,}3_{(8)}$; д) $388{,}3_{(16)} + 209{,}4_{(16)}$.
4. а) $1111100010_{(2)} - 101011101_{(2)}$; б) $1011000100_{(2)} - 1000100000_{(2)}$; в) $1101111000{,}1001_{(2)} - 1000000{,}01_{(2)}$; г) $1040{,}2_{(8)} - 533{,}2_{(8)}$; д) $3FB{,}4_{(16)} - 140{,}6_{(16)}$.
5. а) $11111_{(2)} \cdot 10001_{(2)}$; б) $1237{,}3_{(8)} \cdot 117{,}5_{(8)}$; в) $66{,}4_{(16)} \cdot 65{,}8_{(16)}$.

Вариант 10

1. а) $618_{(10)}$; б) $556_{(10)}$; в) $129{,}25_{(10)}$; г) $928{,}25_{(10)}$; д) $155{,}45_{(10)}$.
2. а) $1111011011_{(2)}$; б) $1011101101_{(2)}$; в) $1001110110{,}011_{(2)}$; г) $1011110011{,}10111_{(2)}$; д) $675{,}2_{(8)}$; е) $94{,}4_{(16)}$.
3. а) $11111010_{(2)} + 10000001011_{(2)}$; б) $1011010_{(2)} + 1001111001_{(2)}$; в) $10110110{,}01_{(2)} + 1001001011{,}01_{(2)}$; г) $1706{,}34_{(8)} + 650{,}3_{(8)}$; д) $180{,}4_{(16)} + 3A6{,}28_{(16)}$.
4. а) $111101101_{(2)} - 101111010_{(2)}$; б) $1000110100_{(2)} - 100100111_{(2)}$; в) $1111111011{,}01_{(2)} - 100000100{,}011_{(2)}$; г) $1300{,}44_{(8)} - 1045{,}34_{(8)}$; д) $16A{,}8_{(16)} - 147{,}6_{(16)}$.
5. а) $100111_{(2)} \cdot 110101_{(2)}$; б) $1542{,}2_{(8)} \cdot 50{,}6_{(8)}$; в) $A{,}8_{(16)} \cdot E{,}2_{(16)}$.

Вариант 11

1. а) $772_{(10)}$; б) $71_{(10)}$; в) $284{,}375_{(10)}$; г) $876{,}5_{(10)}$; д) $281{,}86_{(10)}$.
2. а) $1000001111_{(2)}$; б) $1010000110_{(2)}$; в) $101100110{,}011011_{(2)}$; г) $100100110{,}101011_{(2)}$; д) $1022{,}2_{(8)}$; е) $53{,}9_{(16)}$.
3. а) $1100111_{(2)} + 1010111000_{(2)}$; б) $1101111010_{(2)} + 1000111100_{(2)}$; в) $1111101110{,}01_{(2)} + 1110001{,}011_{(2)}$; г) $153{,}3_{(8)} + 1347{,}2_{(8)}$; д) $E0{,}2_{(16)} + 1E0{,}4_{(16)}$.
4. а) $1010101110_{(2)} - 11101001_{(2)}$; б) $1000100010_{(2)} - 110101110_{(2)}$; в) $1010100011{,}011_{(2)} - 1000001010{,}0001_{(2)}$; г) $1517{,}64_{(8)} - 1500{,}3_{(8)}$; д) $367{,}6_{(16)} - 4A{,}C_{(16)}$.
5. а) $1100110_{(2)} \cdot 101111_{(2)}$; б) $1272{,}3_{(8)} \cdot 23{,}14_{(8)}$; в) $48{,}4_{(16)} \cdot 5{,}A_{(16)}$.

Вариант 12

1. а) $233_{(10)}$; б) $243_{(10)}$; в) $830{,}375_{(10)}$; г) $212{,}5_{(10)}$; д) $58{,}89_{(10)}$.
2. а) $1001101111_{(2)}$; б) $1000001110_{(2)}$; в) $111110011{,}011_{(2)}$;
   г) $11010101{,}1001_{(2)}$; д) $1634{,}5_{(8)}$; е) $C2{,}3_{(16)}$.
3. а) $1101111001_{(2)} + 1010010101_{(2)}$; б) $1111001001_{(2)} + 1001100100_{(2)}$;
   в) $100110010{,}011_{(2)} + 110001000{,}011_{(2)}$; г) $1712{,}14_{(8)} + 710{,}4_{(8)}$;
   д) $E6{,}1_{(16)} + 38C{,}8_{(16)}$.
4. а) $1000001110_{(2)} - 100100001_{(2)}$; б) $1101000110_{(2)} - 1001101000_{(2)}$;
   в) $1011001111{,}01_{(2)} - 110100010{,}01_{(2)}$; г) $1734{,}4_{(8)} - 134{,}2_{(8)}$;
   д) $2F2{,}A_{(16)} - 22D{,}A_{(16)}$.
5. а) $1000000_{(2)} \cdot 100101_{(2)}$; б) $103{,}2_{(8)} \cdot 147{,}04_{(8)}$; в) $67{,}4_{(16)} \cdot 54{,}8_{(16)}$.

Вариант 13

1. а) $218_{(10)}$; б) $767_{(10)}$; в) $894{,}5_{(10)}$; г) $667{,}125_{(10)}$; д) $3{,}67_{(10)}$.
2. а) $1111100010_{(2)}$; б) $1000011110_{(2)}$; в) $101100001{,}011101_{(2)}$;
   г) $1001111001{,}1_{(2)}$; д) $1071{,}54_{(8)}$; е) $18B{,}0C_{(16)}$.
3. а) $1000011111_{(2)} + 1111100_{(2)}$; б) $1011100011_{(2)} + 111110110_{(2)}$;
   в) $111111100{,}1_{(2)} + 1011100100{,}1_{(2)}$; г) $1777{,}2_{(8)} + 444{,}1_{(8)}$;
   д) $3EF{,}3_{(16)} + C7{,}4_{(16)}$.
4. а) $1101000100_{(2)} - 101010101_{(2)}$; б) $1110010111_{(2)} - 1011100_{(2)}$;
   в) $1100101111{,}01_{(2)} - 10010001{,}01_{(2)}$; г) $640{,}2_{(8)} - 150{,}22_{(8)}$;
   д) $380{,}68_{(16)} - 50{,}4_{(16)}$.
5. а) $100010_{(2)} \cdot 1100110_{(2)}$; б) $741{,}4_{(8)} \cdot 141{,}64_{(8)}$; в) $B{,}7_{(16)} \cdot D{,}C_{(16)}$.

Вариант 14

1. а) $898_{(10)}$; б) $751_{(10)}$; в) $327{,}375_{(10)}$; г) $256{,}625_{(10)}$; д) $184{,}4_{(10)}$.
2. а) $101110100_{(2)}$; б) $1111101101_{(2)}$; в) $1110100001{,}01_{(2)}$;
   г) $1011111010{,}0001_{(2)}$; д) $744{,}12_{(8)}$; е) $1EE{,}C_{(16)}$.
3. а) $1001000000_{(2)} + 101010110_{(2)}$; б) $11000010_{(2)} + 1001110100_{(2)}$;
   в) $1011101110{,}1_{(2)} + 11100101{,}01_{(2)}$; г) $2015{,}1_{(8)} + 727{,}54_{(8)}$;
   д) $9D{,}8_{(16)} + ED{,}8_{(16)}$.
4. а) $1010000100_{(2)} - 1000001000_{(2)}$; б) $1111110011_{(2)} - 1001101001_{(2)}$;
   в) $101001100{,}101_{(2)} - 100100101{,}1_{(2)}$; г) $1024{,}6_{(8)} - 375{,}14_{(8)}$;
   д) $3E9{,}4_{(16)} - 72{,}6_{(16)}$.
5. а) $1001010_{(2)} \cdot 1001000_{(2)}$; б) $747{,}2_{(8)} \cdot 64{,}14_{(8)}$; в) $56{,}1_{(16)} \cdot 33{,}C_{(16)}$.

Вариант 15

1. а) $557_{(10)}$; б) $730_{(10)}$; в) $494{,}25_{(10)}$; г) $737{,}625_{(10)}$; д) $165{,}37_{(10)}$.
2. а) $101001101_{(2)}$; б) $1110111100_{(2)}$;
   в) $10000001000{,}001_{(2)}$; г) $1000110110{,}11011_{(2)}$;
   д) $147{,}56_{(8)}$; е) $1CA{,}3_{(16)}$.
3. а) $1101100001_{(2)} + 1001101110_{(2)}$; б) $1101010101_{(2)} + 101011001_{(2)}$;
   в) $1101111110{,}011_{(2)} + 1100101101{,}1011_{(2)}$; г) $1771{,}2_{(8)} + 300{,}5_{(8)}$;
   д) $2F2{,}8_{(16)} + E4{,}B_{(16)}$.
4. а) $1111000000_{(2)} - 111101000_{(2)}$; б) $1100110111_{(2)} - 1001110000_{(2)}$;
   в) $1000011110{,}1001_{(2)} - 110000111{,}01_{(2)}$; г) $1436{,}34_{(8)} - 145{,}2_{(8)}$;
   д) $3F5{,}98_{(16)} - 240{,}3_{(16)}$.
5. а) $1011100_{(2)} \cdot 101000_{(2)}$; б) $1300{,}6_{(8)} \cdot 65{,}2_{(8)}$; в) $68{,}A_{(16)} \cdot 9{,}6_{(16)}$.

Вариант 16

1. а) $737_{(10)}$; б) $92_{(10)}$; в) $934{,}25_{(10)}$; г) $413{,}5625_{(10)}$; д) $100{,}94_{(10)}$.
2. а) $1110000010_{(2)}$; б) $1000100_{(2)}$; в) $110000100{,}001_{(2)}$;
   г) $1001011111{,}00011_{(2)}$; д) $665{,}42_{(8)}$;
   е) $246{,}18_{(16)}$.
3. а) $11110100_{(2)} + 110100001_{(2)}$; б) $1101110_{(2)} + 101001000_{(2)}$;
   в) $1100110011{,}1_{(2)} + 111000011{,}101_{(2)}$; г) $1455{,}04_{(8)} + 203{,}3_{(8)}$;
   д) $14E{,}8_{(16)} + 184{,}3_{(16)}$.
4. а) $1000010101_{(2)} - 100101000_{(2)}$; б) $1001011011_{(2)} - 101001110_{(2)}$;
   в) $111111011{,}101_{(2)} - 100000010{,}01_{(2)}$; г) $341{,}2_{(8)} - 275{,}2_{(8)}$;
   д) $249{,}5_{(16)} - EE{,}A_{(16)}$.
5. а) $1001000_{(2)} \cdot 1010011_{(2)}$; б) $412{,}5_{(8)} \cdot 13{,}1_{(8)}$; в) $3B{,}A_{(16)} \cdot 10{,}4_{(16)}$.

Вариант 17

1. а) $575_{(10)}$; б) $748_{(10)}$; в) $933{,}5_{(10)}$; г) $1005{,}375_{(10)}$; д) $270{,}44_{(10)}$.
2. а) $1010000_{(2)}$; б) $10010000_{(2)}$; в) $1111010000{,}01_{(2)}$;
   г) $101000011{,}01_{(2)}$; д) $1004{,}1_{(8)}$; е) $103{,}8C_{(16)}$.
3. а) $1011110101_{(2)} + 1010100110_{(2)}$; б) $1001100011_{(2)} + 1110010010_{(2)}$;
   в) $1111110100{,}01_{(2)} + 110100100{,}01_{(2)}$; г) $755{,}36_{(8)} + 1246{,}5_{(8)}$;
   д) $8D{,}2_{(16)} + 63{,}8_{(16)}$.
4. а) $1100111110_{(2)} - 1101001_{(2)}$; б) $1101111011_{(2)} - 1101110101_{(2)}$;
   в) $1101001010{,}011_{(2)} - 1010011110{,}101_{(2)}$; г) $1632{,}1_{(8)} - 706{,}34_{(8)}$;
   д) $283{,}C_{(16)} - 19C{,}8_{(16)}$.
5. а) $111000_{(2)} \cdot 1101001_{(2)}$; б) $133{,}6_{(8)} \cdot 73{,}4_{(8)}$; в) $46{,}8_{(16)} \cdot B{,}A_{(16)}$.

Вариант 18

1. а) $563_{(10)}$; б) $130_{(10)}$; в) $892{,}5_{(10)}$; г) $619{,}25_{(10)}$; д) $198{,}05_{(10)}$.
2. а) $11100001_{(2)}$; б) $101110111_{(2)}$; в) $1011110010{,}0001_{(2)}$;
   г) $1100010101{,}010101_{(2)}$; д) $533{,}2_{(8)}$; е) $32{,}22_{(16)}$.
3. а) $1100100011_{(2)} + 1101001111_{(2)}$; б) $111101111_{(2)} + 10010100_{(2)}$;
   в) $1010010000{,}0111_{(2)} + 111010100{,}001_{(2)}$; г) $1724{,}6_{(8)} + 1322{,}2_{(8)}$;
   д) $2C7{,}68_{(16)} + 6F{,}4_{(16)}$.
4. а) $111001110_{(2)} - 11011011_{(2)}$; б) $1011000001_{(2)} - 110100001_{(2)}$;
   в) $1011111101{,}1_{(2)} - 111100000{,}01_{(2)}$; г) $1126{,}06_{(8)} - 203{,}54_{(8)}$;
   д) $32B{,}D_{(16)} - 187{,}D8_{(16)}$.
5. а) $1100101_{(2)} \cdot 1001010_{(2)}$; б) $1544{,}4_{(8)} \cdot 16{,}64_{(8)}$; в) $69{,}8_{(16)} \cdot 30{,}8_{(16)}$.

Вариант 19

1. а) $453_{(10)}$; б) $481_{(10)}$; в) $461{,}25_{(10)}$; г) $667{,}25_{(10)}$; д) $305{,}88_{(10)}$.
2. а) $111001010_{(2)}$; б) $1101110001_{(2)}$; в) $1001010100{,}10001_{(2)}$;
   г) $111111110{,}11001_{(2)}$; д) $1634{,}35_{(8)}$; е) $6B{,}A_{(16)}$.
3. а) $101110001_{(2)} + 101111001_{(2)}$; б) $1110001110_{(2)} + 1100110111_{(2)}$;
   в) $10000011010{,}01_{(2)} + 1010010110{,}01_{(2)}$; г) $1710{,}2_{(8)} + 773{,}24_{(8)}$;
   д) $3E7{,}7_{(16)} + 32{,}2_{(16)}$.
4. а) $1111000010_{(2)} - 1110000011_{(2)}$; б) $1110101011_{(2)} - 111000111_{(2)}$;
   в) $1111011010{,}011_{(2)} - 1011100111{,}01_{(2)}$; г) $1650{,}2_{(8)} - 502{,}2_{(8)}$;
   д) $3E0{,}6_{(16)} - 17E{,}9_{(16)}$.
5. а) $1001101_{(2)} \cdot 11111_{(2)}$; б) $1226{,}1_{(8)} \cdot 24{,}4_{(8)}$; в) $36{,}6_{(16)} \cdot 38{,}4_{(16)}$.

Вариант 20

1. а) $572_{(10)}$; б) $336_{(10)}$; в) $68{,}5_{(10)}$; г) $339{,}25_{(10)}$; д) $160{,}57_{(10)}$.
2. а) $1010110011_{(2)}$; б) $1101110100_{(2)}$; в) $1010101{,}101_{(2)}$;
   г) $1101000{,}001_{(2)}$; д) $414{,}1_{(8)}$; е) $366{,}4_{(16)}$.
3. а) $10001000_{(2)} + 1011010010_{(2)}$; б) $111110011_{(2)} + 111110000_{(2)}$;
   в) $1010001010{,}1011_{(2)} + 1101010100{,}011_{(2)}$; г) $711{,}2_{(8)} + 214{,}2_{(8)}$;
   д) $7A{,}58_{(16)} + 2D0{,}9_{(16)}$.
4. а) $110111010_{(2)} - 1110001_{(2)}$; б) $1100001000_{(2)} - 11000100_{(2)}$;
   в) $1111111010{,}01_{(2)} - 1000110010{,}0101_{(2)}$; г) $1060{,}52_{(8)} - 761{,}14_{(8)}$;
   д) $1C0{,}6_{(16)} - 8D{,}2_{(16)}$.
5. а) $11101_{(2)} \cdot 110101_{(2)}$; б) $1106{,}2_{(8)} \cdot 145{,}2_{(8)}$; в) $65{,}4_{(16)} \cdot 55{,}9_{(16)}$.

Вариант 21

1. а) $949_{(10)}$; б) $763_{(10)}$; в) $994{,}125_{(10)}$; г) $523{,}25_{(10)}$; д) $203{,}82_{(10)}$.
2. а) $1110001111_{(2)}$; б) $100011011_{(2)}$; в) $1001100101{,}1001_{(2)}$;
   г) $1001001{,}011_{(2)}$; д) $335{,}7_{(8)}$; е) $14C{,}A_{(16)}$.
3. а) $1110101010_{(2)} + 10111001_{(2)}$; б) $10111010_{(2)} + 10010100_{(2)}$;
   в) $111101110{,}1011_{(2)} + 1111011110{,}1_{(2)}$; г) $1153{,}2_{(8)} + 1147{,}32_{(8)}$;
   д) $40F{,}4_{(16)} + 160{,}4_{(16)}$.
4. а) $1000000100_{(2)} - 101010001_{(2)}$; б) $1010111101_{(2)} - 111000010_{(2)}$;
   в) $1101000000{,}01_{(2)} - 1001011010{,}011_{(2)}$; г) $2023{,}5_{(8)} - 527{,}4_{(8)}$;
   д) $25E{,}6_{(16)} - 1B1{,}5_{(16)}$.
5. а) $1001011_{(2)} \cdot 1010110_{(2)}$; б) $1650{,}2_{(8)} \cdot 120{,}2_{(8)}$; в) $19{,}4_{(16)} \cdot 2F{,}8_{(16)}$.

Вариант 22

1. а) $563_{(10)}$; б) $264_{(10)}$; в) $234{,}25_{(10)}$; г) $53{,}125_{(10)}$; д) $286{,}16_{(10)}$.
2. а) $1100010010_{(2)}$; б) $10011011_{(2)}$; в) $1111000001{,}01_{(2)}$;
   г) $10110111{,}01_{(2)}$; д) $416{,}1_{(8)}$; е) $215{,}7_{(16)}$.
3. а) $10111111_{(2)} + 1100100001_{(2)}$; б) $110010100_{(2)} + 1011100001_{(2)}$;
   в) $10000001001{,}0101_{(2)} + 1010000110{,}01_{(2)}$; г) $1512{,}4_{(8)} + 1015{,}2_{(8)}$;
   д) $274{,}5_{(16)} + DD{,}4_{(16)}$.
4. а) $1000001001_{(2)} - 111110100_{(2)}$; б) $1111000101_{(2)} - 1100110101_{(2)}$;
   в) $1100110101{,}1_{(2)} - 1011100011{,}01_{(2)}$;
   г) $1501{,}34_{(8)} - 1374{,}5_{(8)}$; д) $12D{,}3_{(16)} - 39{,}6_{(16)}$.
5. а) $111101_{(2)} \cdot 1010111_{(2)}$; б) $1252{,}14_{(8)} \cdot 76{,}04_{(8)}$; в) $66{,}68_{(16)} \cdot 1E{,}3_{(16)}$.

Вариант 23

1. а) $279_{(10)}$; б) $281_{(10)}$; в) $841{,}375_{(10)}$; г) $800{,}3125_{(10)}$; д) $208{,}92_{(10)}$.
2. а) $1100111001_{(2)}$; б) $10011101_{(2)}$;
   в) $1111011{,}001_{(2)}$; г) $110000101{,}01_{(2)}$;
   д) $1601{,}56_{(8)}$; е) $16E{,}B4_{(16)}$.
3. а) $1000100001_{(2)} + 1011100110_{(2)}$; б) $1101110011_{(2)} + 111000101_{(2)}$;
   в) $1011011{,}01_{(2)} + 1000101110{,}1001_{(2)}$; г) $665{,}1_{(8)} + 1217{,}2_{(8)}$;
   д) $30C{,}7_{(16)} + 2A1{,}8_{(16)}$.
4. а) $11110010_{(2)} - 10101001_{(2)}$; б) $1110100001_{(2)} - 1011001001_{(2)}$;
   в) $1101001010{,}1_{(2)} - 1011101001{,}11011_{(2)}$; г) $166{,}14_{(8)} - 143{,}2_{(8)}$;
   д) $287{,}A_{(16)} - 62{,}8_{(16)}$.
5. а) $1001001_{(2)} \cdot 100010_{(2)}$; б) $324{,}2_{(8)} \cdot 122{,}12_{(8)}$; в) $F{,}4_{(16)} \cdot 38{,}6_{(16)}$.

Вариант 24

1. а) $744_{(10)}$; б) $554_{(10)}$; в) $269{,}375_{(10)}$; г) $120{,}25_{(10)}$; д) $139{,}09_{(10)}$.
2. а) $101000001_{(2)}$; б) $1110111100_{(2)}$; в) $1001110101{,}011001_{(2)}$; г) $1000010001{,}00011_{(2)}$; д) $1177{,}6_{(8)}$; е) $3FA{,}E8_{(16)}$.
3. а) $10000001010_{(2)} + 11111111_{(2)}$; б) $111011000_{(2)} + 1110111_{(2)}$; в) $111010101{,}101_{(2)} + 11101111{,}001_{(2)}$; г) $251{,}42_{(8)} + 72{,}54_{(8)}$; д) $2CF{,}A_{(16)} + 242{,}4_{(16)}$.
4. а) $1001000100_{(2)} - 100111010_{(2)}$; б) $100001100_{(2)} - 10110011_{(2)}$; в) $1110111100{,}011_{(2)} - 1100000011{,}0111_{(2)}$; г) $1700{,}2_{(8)} - 456{,}44_{(8)}$; д) $1A1{,}8_{(16)} - E0{,}7_{(16)}$.
5. а) $11110_{(2)} \cdot 1100100_{(2)}$; б) $1034{,}6_{(8)} \cdot 43{,}1_{(8)}$; в) $2C{,}4_{(16)} \cdot 6{,}2_{(16)}$.

Вариант 25

1. а) $686_{(10)}$; б) $585_{(10)}$; в) $530{,}6875_{(10)}$; г) $87{,}375_{(10)}$; д) $131{,}82_{(10)}$.
2. а) $110111001_{(2)}$; б) $101111011_{(2)}$; в) $1110111100{,}1_{(2)}$; г) $110000011{,}0111_{(2)}$; д) $742{,}34_{(8)}$; е) $396{,}A_{(16)}$.
3. а) $10000010001_{(2)} + 1000100010_{(2)}$; б) $101011100_{(2)} + 10101111_{(2)}$; в) $1001110000{,}001_{(2)} + 10100101{,}011_{(2)}$; г) $1216{,}2_{(8)} + 2012{,}4_{(8)}$; д) $372{,}18_{(16)} + 251{,}38_{(16)}$.
4. а) $100110110_{(2)} - 11101001_{(2)}$; б) $1010100111_{(2)} - 110000010_{(2)}$; в) $11001101{,}1011_{(2)} - 1001101{,}011_{(2)}$; г) $1254{,}2_{(8)} - 1150{,}54_{(8)}$; д) $2E1{,}8_{(16)} - 19A{,}4_{(16)}$.
5. а) $1101000_{(2)} \cdot 10011_{(2)}$; б) $1411{,}44_{(8)} \cdot 46{,}4_{(8)}$; в) $63{,}8_{(16)} \cdot 8{,}6_{(16)}$.

Дополнительная литература

1. *Аветисян Р.Д., Аветисян Д.В.* Теоретические основы информатики. — М.: РГГУ, 1997.
2. *Бауэр Ф.Л., Гооз Г.* Информатика. Вводный курс: В 2 ч. Ч.2: Пер. с нем. — М.: Мир, 1990.
3. Информатика в понятиях и терминах. — М.: Просвещение, 1991.
4. Информатика. Энциклопедический словарь для начинающих. — М.: Педагогика-Пресс, 1994.
5. Основы информатики и вычислительной техники / А.Г.Гейн, В.Г.Житомирский, Е.В.Линецкий и др. — М.: Просвещение, 1991.
6. *Решетников В.Н., Сотников А.Н.* Информатика — что это? — М.: Радио и связь, 1989.

§ 4. КОДИРОВАНИЕ ИНФОРМАЦИИ

Рекомендации по проведению занятий

Кодирование информации — одна из базовых тем курса теоретических основ информатики, отражающая фундаментальную необходимость представления информации в какой-либо форме. При этом слово «кодирование» понимается не в узком смысле — кодирование как способ сделать сообщение непонятным для всех,

кто не владеет ключом кода, а в широком — как представление информации в виде сообщения на каком-либо языке. Освещение данной темы в курсе информатики возможно под различными углами зрения и на различных уровнях. Самый простой подход состоит в рассмотрении понятия кодирования как представления информации в ознакомительном, общеобразовательном плане. Более продвинутый подход включает изучение теории кодирования, в том числе ряда теорем с доказательствами. В данном практикуме мы ориентируемся на достаточно элементарные сведения о кодировании, имеющие общеобразовательное значение, и оставляем серьезное знакомство с теорией кодирования для специальных курсов.

По теме «кодирование информации» целесообразно проведение как семинарских, так и практических занятий (по решению задач). Весьма полезной является подготовка рефератов. Что же касается проведения лабораторных работ, то в § 5 предложена лишь одна работа, предусматривающая отработку навыков по частному способу кодирования — представлению данных в памяти ЭВМ; это полезно для предварительного формирования знаний по вычислительной технике.

Контрольные вопросы

1. Что такое кодирование информации в общем смысле?
2. Каково место кодирования среди процессов обработки информации?
3. Что называется знаком, абстрактным алфавитом? Приведите примеры.
4. Что такое код? Приведите примеры кодирования и декодирования.
5. Что называется избыточностью кода?
6. Какова избыточность естественных языков? Для чего она служит?
7. Приведите примеры искусственного повышения избыточности кода.
8. В чем состоит содержание 1-й и 2-й теорем Шеннона?
9. Какие коды называются двоичными? Приведите примеры.
10. Какой код используется для кодирования букв латинского алфавита буквами персонального компьютера?
11. Какие коды используются в вычислительной технике для кодирования букв русского алфавита?
12. Как получить прямой и дополнительный коды целого числа?
13. Как представляются действительные числа в памяти ЭВМ?
14. Как кодируется графическая информация, если изображение черно-белое (цветное)?

Темы для рефератов

1. История кодирования информации.
2. Символы и алфавиты для кодирования информации.
3. Кодирование и шифрование.
4. Основные результаты теории кодирования.
5. Современные способы кодирования информации в вычислительной технике.

Темы семинарских занятий

1. Понятие «кодирование информации». Знак. Алфавит. История кодирования и шифрования.

2. Кодирование информации в вычислительной технике.
3. Основные теоремы теории кодирования и их следствия.

Задачи и упражнения

1. Оцените число символов алфавита, кодируемого с помощью двоичных последовательностей длиной:

а) 4 знака; б) 8 знаков; в) 12 знаков; г) 16 знаков.

2. С помощью кодовой таблицы ASCII декодируйте следующее сообщение: 01010100 01001111 00100000 01000010 01000101 00100000 01001111 01010010 00100000 01001110 01001111 01010100 00100000 01010100 01001111 00100000 01000010 01000101.

3. С помощью кодовой таблицы ASCII закодируйте в последовательность шестнадцатеричных чисел слово COMPUTER.

4. Закодируйте и декодируйте любое текстовое сообщение с помощью кода Цезаря— пронумеровав алфавит десятичными цифрами и заменив буквы соответствующими им числами.

5. Закодируйте и декодируйте любое текстовое сообщение, усложнив код Цезаря добавлением к каждому последующему числу, заменяющему букву, некоторое постоянное число.

§ 5. ПРЕДСТАВЛЕНИЕ ДАННЫХ В ПАМЯТИ ЭВМ

Краткие сведения

Для представления информации в памяти ЭВМ (как числовой, так и нечисловой) используется двоичный способ кодирования.

Элементарная ячейка памяти ЭВМ имеет длину 8 бит (байт). Каждый байт имеет свой номер (его называют ***адресом***). Наибольшую последовательность бит, которую ЭВМ может обрабатывать как единое целое, называют ***машинным словом***. Длина машинного слова зависит от разрядности процессора и может быть равной 16, 32 битам и т.д.

Для кодирования символов достаточно одного байта. При этом можно представить 256 символов (с десятичными кодами от 0 до 255). Набор символов персональных ЭВМ IBM PC чаще всего является расширением кода ASCII (American Standart Code for Information Interchange — стандартный американский код для обмена информацией).

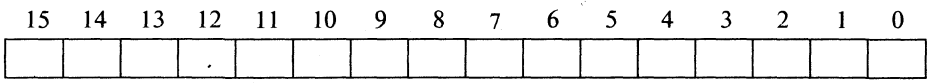
В некоторых случаях при представлении в памяти ЭВМ чисел используется смешанная двоично-десятичная «система счисления», где для хранения каждого десятичного знака нужен полубайт (4 бита) и десятичные цифры от 0 до 9 представляются соответствующими двоичными числами от 0000 до 1001. Например, упакованный десятичный формат, предназначенный для хранения целых чисел с 18 значащими цифрами и занимающий в памяти 10 байт (старший из которых знаковый), использует именно этот вариант.

Другой способ представления целых чисел — ***дополнительный код***. Диапазон значений величин зависит от количества бит памяти, отведенных для их хранения. Например, величины типа Integer (все названия типов данных здесь и ниже представлены в том виде, в каком они приняты в языке программирования Turbo Pascal, в других языках такие типы данных тоже есть, но могут иметь

другие названия) лежат в диапазоне от -32768 (-2^{15}) до 32767 ($2^{15}-1$), и для их хранения отводится 2 байта; типа LongInt — в диапазоне от -2^{31} до $2^{31}-1$ и размещаются в 4 байтах; типа Word — в диапазоне от 0 до 65535 ($2^{16}-1$) (используется 2 байта) и т.д.

Как видно из примеров, данные могут быть интерпретированы как числа со знаками, так и без знаков. В случае представления величины со знаком самый левый (старший) разряд указывает на положительное число, если содержит нуль, и на отрицательное, если — единицу.

Вообще разряды нумеруются справа налево, начиная с 0. Ниже показана нумерация бит в двухбайтовом машинном слове.



Дополнительный код положительного числа совпадает с его ***прямым кодом***. Прямой код целого числа может быть получен следующим образом: число переводится в двоичную систему счисления, а затем его двоичную запись слева дополняют таким количеством незначащих нулей, сколько требует тип данных, к которому принадлежит число. Например, если число $37_{(10)} = 100101_{(2)}$ объявлено величиной типа Integer, то его прямым кодом будет 0000000000100101, а если величиной типа LongInt, то его прямой код будет 00000000000000000000000000100101. Для более компактной записи чаще используют шестнадцатеричный код. Полученные коды можно переписать соответственно как $0025_{(16)}$ и $00000025_{(16)}$.

Дополнительный код целого отрицательного числа может быть получен по следующему алгоритму:

1) записать прямой код модуля числа;

2) инвертировать его (заменить единицы нулями, нули — единицами);

3) прибавить к инверсному коду единицу.

Например, запишем дополнительный код числа (-37), интерпретируя его как величину типа LongInt:

1) прямой код числа 37 есть 00000000000000000000000000100101;

2) инверсный код 11111111111111111111111111011010;

3) дополнительный код 11111111111111111111111111011011 или $FFFFFFDB_{(16)}$.

При получении числа по его дополнительному коду прежде всего необходимо определить его знак. Если число окажется положительным, то просто перевести его код в десятичную систему счисления. В случае отрицательного числа необходимо выполнить следующий алгоритм:

1) вычесть из кода числа 1;

2) инвертировать код;

3) перевести в десятичную систему счисления. Полученное число записать со знаком минус.

Примеры. Запишем числа, соответствующие дополнительным кодам:

а) 0000000000010111. Поскольку в старшем разряде записан нуль, то результат будет положительным. Это код числа 23;

б) 1111111111000000. Здесь записан код отрицательного числа. Исполняем алгоритм:

1) $1111111111000000_{(2)} - 1_{(2)} = 1111111110111111_{(2)}$;

2) 0000000001000000;

3) $1000000_{(2)} = 64_{(10)}$.

Ответ: -64.

Несколько иной способ применяется для представления в памяти персонального компьютера действительных чисел. Рассмотрим представление величин ***с плавающей точкой***.

Любое действительное число можно записать в стандартном виде $M \cdot 10^p$, где $1 \le M < 10$, p — целое. Например, $120100000 = 1{,}201 \cdot 10^8$. Поскольку каждая позиция десятичного числа отличается от соседней на степень числа 10, умножение на 10 эквивалентно сдвигу десятичной запятой на одну позицию вправо. Аналогично деление на 10 сдвигает десятичную запятую на позицию влево. Поэтому приведенный выше пример можно продолжить: $120100000 = 1{,}201 \cdot 10^8 = 0{,}1201 \cdot 10^9 = 12{,}01 \cdot 10^7$... Десятичная запятая «плавает» в числе и больше не помечает абсолютное место между целой и дробной частями.

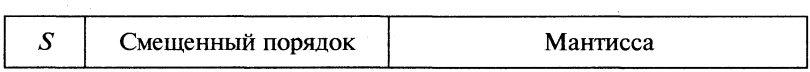
В приведенной выше записи M называют ***мантиссой*** числа, а p — его ***порядком***. Для того чтобы сохранить максимальную точность, вычислительные машины почти всегда хранят мантиссу в нормализованном виде, что означает, что мантисса в данном случае есть число, лежащее между $1_{(10)}$ и $2_{(10)}$ ($1 \le M < 2$). Основание системы счисления здесь, как уже отмечалось выше, — число 2. Способ хранения мантиссы с плавающей точкой подразумевает, что двоичная запятая находится на фиксированном месте. Фактически подразумевается, что двоичная запятая следует после первой двоичной цифры, т.е. нормализация мантиссы делает единичным первый бит, помещая тем самым значение между единицей и двойкой. Место, отводимое для числа с плавающей точкой, делится на два поля. Одно поле содержит знак и значение мантиссы, а другое содержит знак и значение порядка.

Персональный компьютер IBM PC позволяет работать со следующими действительными типами (диапазон значений указан по абсолютной величине):

Тип	Диапазон	Мантисса	Байты
Real	$2{,}9 \cdot 10^{-39} ... 1{,}7 \cdot 10^{38}$	11—12	6
Single	$1{,}5 \cdot 10^{-45} ... 3{,}4 \cdot 10^{38}$	7—8	4
Double	$5{,}0 \cdot 10^{-324} ... 1{,}7 \cdot 10^{308}$	15—16	8
Extended	$3{,}4 \cdot 10^{-4932} ... 1{,}1 \cdot 10^{4932}$	19—20	10

Покажем преобразование действительного числа для представления его в памяти ЭВМ на примере величины типа Double.

Как видно из таблицы, величина это типа занимает в памяти 8 байт. На рисунке показано, как здесь представлены поля мантиссы и порядка:



63 52 0

Можно заметить, что старший бит, отведенный под мантиссу, имеет номер 51, т.е. мантисса занимает младшие 52 бита. Черта указывает здесь на положение двоичной запятой. Перед запятой должен стоять бит целой части мантиссы, но поскольку она всегда равна 1, здесь данный бит не требуется и соответствующий разряд отсутствует в памяти (но он подразумевается). Значение порядка для упрощения вычислений и сравнения действительных чисел хранится в виде ***смещенного числа***, т.е. к настоящему значению порядка перед записью его в память прибавляется смещение. Смещение выбирается так, чтобы минимальному значению поряд-

ка соответствовал нуль. Например, для типа Double порядок занимает 11 бит и имеет диапазон от 2^{-1023} до 2^{1023}, поэтому смещение равно $1023_{(10)} = 1111111111_{(2)}$. Наконец, бит с номером 63 указывает на знак числа.

Таким образом, из вышесказанного вытекает следующий алгоритм для получения представления действительного числа в памяти ЭВМ:

1) перевести модуль данного числа в двоичную систему счисления;

2) нормализовать двоичное число, т.е. записать в виде $M \cdot 2^p$, где M — мантисса (ее целая часть равна $1_{(2)}$) и p — порядок, записанный в десятичной системе счисления;

3) прибавить к порядку смещение и перевести смещенный порядок в двоичную систему счисления;

4) учитывая знак заданного числа (0 — положительное; 1 — отрицательное), выписать его представление в памяти ЭВМ.

Пример. Запишем код числа −312,3125.

1) Двоичная запись модуля этого числа имеет вид 100111000,0101.

2) Имеем $100111000{,}0101 = 1{,}001110000101 \cdot 2^8$.

3) Получаем смещенный порядок 8 + 1023 = 1031. Далее имеем $1031_{(10)} = 10000000111_{(2)}$.

4) Окончательно

0	10000000111	0011100001010000000000000000000000000000000000000000000000
63	52	0

Очевидно, что более компактно полученный код стоит записать следующим образом: $C073850000000000_{(16)}$.

Другой пример иллюстрирует обратный переход от кода действительного числа к самому числу.

Пример. Пусть дан код $3FEC600000000000_{(16)}$ или

0	10000000111	1100011000000000000000000000000000000000000000000000000000
63	52	0

1) Прежде всего замечаем, что это код положительного числа, поскольку в разряде с номером 63 записан нуль. Получим порядок этого числа: $01111111110_{(2)} = 1022_{(10)}$; $1022 - 1023 = -1$.

2) Число имеет вид $1{,}1100011 \cdot 2^{-1}$ или 0,11100011.

3) Переводом в десятичную систему счисления получаем 0,88671875.

Лабораторная работа

Время выполнения 6 часов.

Задания к лабораторной работе

1. Переведите данное число из десятичной системы счисления в двоично-десятичную.

2. Переведите данное число из двоично-десятичной системы счисления в десятичную.

3. Зашифруйте данный текст, используя таблицу ASCII-кодов.

4. Дешифруйте данный текст, используя таблицу ASCII-кодов.

5. Запишите прямой код числа, интерпретируя его как восьмибитовое целое без знака.

6. Запишите дополнительный код числа, интерпретируя его как восьмибитовое целое со знаком.

7. Запишите прямой код числа, интерпретируя его как шестнадцатибитовое целое без знака.

8. Запишите дополнительный код числа, интерпретируя его как шестнадцатибитовое целое со знаком.

9. Запишите в десятичной системе счисления целое число, если дан его дополнительный код.

10. Запишите код действительного числа, интерпретируя его как величину типа Double.

11. Дан код величины типа Double. Преобразуйте его в число.

Вариант 1

1. а) $585_{(10)}$; б) $673_{(10)}$; в) $626_{(10)}$.
2. а) $010101010101_{(2-10)}$; б) $10011000_{(2-10)}$; в) $010000010110_{(2-10)}$.
3. IBM PC.
4. 8A AE AC AF EC EE E2 A5 E0.
5. а) $224_{(10)}$; б) $253_{(10)}$; в) $226_{(10)}$.
6. а) $115_{(10)}$; б) $-34_{(10)}$; в) $-70_{(10)}$.
7. а) $22491_{(10)}$; б) $23832_{(10)}$.
8. а) $20850_{(10)}$; б) $-18641_{(10)}$.
9. а) 0011010111010110; б) 1000000110101110.
10. а) −578,375; б) −786,375.
11. а) 408E130000000000; б) C077880000000000.

Вариант 2

1. а) $285_{(10)}$; б) $846_{(10)}$; в) $163_{(10)}$.
2. а) $000101010001_{(2-10)}$; б) $010101010011_{(2-10)}$; в) $011010001000_{(2-10)}$.
3. Автоматизация.
4. 50 72 6F 67 72 61 6D.
5. а) $242_{(10)}$; б) $135_{(10)}$; в) $248_{(10)}$.
6. а) $81_{(10)}$; б) $-40_{(10)}$; в) $-24_{(10)}$.
7. а) $18509_{(10)}$; б) $28180_{(10)}$.
8. а) $28882_{(10)}$; б) $-19070_{(10)}$.
9. а) 0110010010010101; б) 1000011111110001.
10. а) −363,15625; б) −487,15625.
11. а) C075228000000000; б) 408B9B0000000000.

Вариант 3

1. а) $905_{(10)}$; б) $504_{(10)}$; в) $515_{(10)}$.
2. а) $010010010100_{(2-10)}$; б) $001000000100_{(2-10)}$; в) $01110000_{(2-10)}$.
3. Информатика.
4. 50 72 6F 63 65 64 75 72 65.
5. а) $207_{(10)}$; б) $210_{(10)}$; в) $226_{(10)}$.
6. а) $98_{(10)}$; б) $-111_{(10)}$; в) $-95_{(10)}$.

7. а) $19835_{(10)}$; б) $22248_{(10)}$.
8. а) $18156_{(10)}$; б) $-28844_{(10)}$.
9. а) 0111100011001000; б) 1111011101101101.
10. а) 334,15625; б) 367,15625.
11. а) C07C08C000000000; б) C0811B0000000000.

Вариант 4

1. а) $483_{(10)}$; б) $412_{(10)}$; в) $738_{(10)}$.
2. а) $001101011000_{(2-10)}$; б) $100010010010_{(2-10)}$; в) $010101000110_{(2-10)}$.
3. Computer.
4. 84 88 91 8A 8E 82 8E 84.
5. а) $185_{(10)}$; б) $224_{(10)}$; в) $193_{(10)}$.
6. а) $89_{(10)}$; б) $-65_{(10)}$; в) $-8_{(10)}$.
7. а) $29407_{(10)}$; б) $25342_{(10)}$.
8. а) $23641_{(10)}$; б) $-23070_{(10)}$.
9. а) 0111011101000111; б) 1010110110101110.
10. а) 215,15625; б) −143,375.
11. а) C071760000000000; б) 407FF28000000000.

Вариант 5

1. а) $88_{(10)}$; б) $153_{(10)}$; в) $718_{(10)}$.
2. а) $000110000100_{(2-10)}$; б) $100110000111_{(2-10)}$; в) $100100011000_{(2-10)}$.
3. Printer.
4. 43 4F 4D 50 55 54 45 52.
5. а) $158_{(10)}$; б) $134_{(10)}$; в) $190_{(10)}$.
6. а) $64_{(10)}$; б) $-104_{(10)}$; в) $-47_{(10)}$.
7. а) $30539_{(10)}$; б) $26147_{(10)}$.
8. а) $22583_{(10)}$; б) $-28122_{(10)}$.
9. а) 0100011011110111; б) 1011101001100000.
10. а) −900,546875; б) −834,5.
11. а) 407C060000000000; б) C0610C0000000000.

Вариант 6

1 а) $325_{(10)}$; б) $112_{(10)}$; в) $713_{(10)}$.
2. а) $100101100010_{(2-10)}$; б) $001001000110_{(2-10)}$; в) $011100110110_{(2-10)}$.
3. Компьютеризация.
4. 50 52 49 4E 54.
5. а) $239_{(10)}$; б) $160_{(10)}$; в) $182_{(10)}$.
6. а) $55_{(10)}$; б) $-89_{(10)}$; в) $-22_{(10)}$.
7. а) $17863_{(10)}$; б) $25893_{(10)}$.
8. а) $24255_{(10)}$; б) $-26686_{(10)}$.
9. а) 0000010101011010; б) 1001110100001011.
10. а) −969,15625; б) −434,15625.
11. а) C082B30000000000; б) C086EB0000000000.

Вариант 7

1. а) $464_{(10)}$; б) $652_{(10)}$; в) $93_{(10)}$.
2. а) $000110010010_{(2-10)}$; б) $001100011000_{(2-10)}$; в) $011000010000_{(2-10)}$.
3. YAMAHA.

4. 4D 4F 44 45 4D.
5. а) $237_{(10)}$; б) $236_{(10)}$; в) $240_{(10)}$.
6. а) $95_{(10)}$; б) $-68_{(10)}$; в) $-77_{(10)}$.
7. а) $28658_{(10)}$; б) $29614_{(10)}$.
8. а) $31014_{(10)}$; б) $-24013_{(10)}$.
9. а) 0001101111111001; б) 1011101101001101.
10. а) −802,15625; б) −172,375.
11. а) C085EB0000000000; б) C07D428000000000.

Вариант 8

1. а) $342_{(10)}$; б) $758_{(10)}$; в) $430_{(10)}$.
2. а) $010110010000_{(2-10)}$; б) $011101100101_{(2-10)}$; в) $011100010111_{(2-10)}$.
3. Световое перо.
4. 4C 61 73 65 72.
5. а) $136_{(10)}$; б) $130_{(10)}$; в) $239_{(10)}$.
6. а) $82_{(10)}$; б) $-13_{(10)}$; в) $-77_{(10)}$.
7. а) $27898_{(10)}$; б) $24268_{(10)}$.
8. а) $19518_{(10)}$; б) $-16334_{(10)}$.
9. а) 0000110100001001; б) 1001110011000000.
10. а) 635,5; б) −555,15625.
11. а) C07848C000000000; б) C085394000000000.

Вариант 9

1. а) $749_{(10)}$; б) $691_{(10)}$; в) $1039_{(10)}$.
2. а) $100100010001_{(2-10)}$; б) $001000111001_{(2-10)}$;
   в) $001101100011_{(2-10)}$.
3. Микропроцессор.
4. 88 AD E4 AE E0 AC A0 E2 A8 AA A0.
5. а) $230_{(10)}$; б) $150_{(10)}$; в) $155_{(10)}$.
6. а) $74_{(10)}$; б) $-43_{(10)}$; в) $-21_{(10)}$.
7. а) $18346_{(10)}$; б) $25688_{(10)}$.
8. а) $31397_{(10)}$; б) $-21029_{(10)}$.
9. а) 0110101101111000; б) 1110100100110101.
10. а) 110,546875; б) −743,375.
11. а) C08B794000000000; б) 407CB28000000000.

Вариант 10

1. а) $817_{(10)}$; б) $661_{(10)}$; в) $491_{(10)}$.
2. а) $100001010001_{(2-10)}$; б) $010000000111_{(2-10)}$;
   в) $001001110001_{(2-10)}$.
3. Принтер.
4. 42 69 6E 61 72 79.
5. а) $219_{(10)}$; б) $240_{(10)}$; в) $202_{(10)}$.
6. а) $44_{(10)}$; б) $-43_{(10)}$; в) $-94_{(10)}$.
7. а) $23359_{(10)}$; б) $27428_{(10)}$.
8. а) $21481_{(10)}$; б) $-20704_{(10)}$.
9. а) 0001101010101010; б) 1011110111001011.
10. а) −141,375; б) 145,375.
11. а) 408EA14000000000; б) C07B128000000000.

Вариант 11

1. а) $596_{(10)}$; б) $300_{(10)}$; в) $515_{(10)}$.
2. а) $001100100110_{(2-10)}$; б) $001000010110_{(2-10)}$; в) $010100010010_{(2-10)}$.
3. Дисковод.
4. 49 6E 66 6F 72 6D 61 74 69 6F 6E.
5. а) $237_{(10)}$; б) $160_{(10)}$; в) $253_{(10)}$.
6. а) $122_{(10)}$; б) $-97_{(10)}$; в) $-82_{(10)}$.
7. а) $30469_{(10)}$; б) $21517_{(10)}$.
8. а) $23008_{(10)}$; б) $-23156_{(10)}$.
9. а) 0010111101000000; б) 1011001101110001.
10. а) 576,375; б) −99,375.
11. а) 40864B0000000000; б) C047140000000000.

Вариант 12

1. а) $322_{(10)}$; б) $320_{(10)}$; в) $738_{(10)}$.
2. а) $000110000000_{(2-10)}$; б) $100101010110_{(2-10)}$; в) $011101100001_{(2-10)}$.
3. Pentium 100.
4. 91 A8 E1 E2 A5 AC A0 20 E1 E7 A8 E1 AB A5 AD A8 EF.
5. а) $201_{(10)}$; б) $135_{(10)}$; в) $198_{(10)}$.
6. а) $91_{(10)}$; б) $-7_{(10)}$; в) $-95_{(10)}$.
7. а) $29234_{(10)}$; б) $19909_{(10)}$.
8. а) $25879_{(10)}$; б) $-27169_{(10)}$.
9. а) 0001111001010100; б) 1011010001110010.
10. а) −796,15625; б) 325,15625.
11. а) 4060B00000000000; б) C0846C6000000000.

Вариант 13

1. а) $780_{(10)}$; б) $949_{(10)}$; в) $718_{(10)}$.
2. а) $0001000000010101_{(2-10)}$; б) $100110011001_{(2-10)}$; в) $001101100001_{(2-10)}$.
3. Арифмометр.
4. AC AE A4 A5 AB A8 E0 AE A2 A0 AD A8 A5.
5. а) $188_{(10)}$; б) $213_{(10)}$; в) $217_{(10)}$.
6. а) $89_{(10)}$; б) $-90_{(10)}$; в) $-34_{(10)}$.
7. а) $25173_{(10)}$; б) $25416_{(10)}$.
8. а) $27435_{(10)}$; б) $-22433_{(10)}$.
9. а) 0111110101101100; б) 1111011001100010.
10. а) −142,375; б) 565,15625.
11. а) C086494000000000; б) C083DC6000000000.

Вариант 14

1. а) $164_{(10)}$; б) $1020_{(10)}$; в) $713_{(10)}$.
2. а) $011110000100_{(2-10)}$; б) $001100010001_{(2-10)}$; в) $100101010001_{(2-10)}$.
3. Сканер.
4. A2 EB E7 A8 E1 AB A8 E2 A5 AB EC AD EB A9 20 ED AA E1 AF A5 E0 A8 AC A5 AD E2.
5. а) $127_{(10)}$; б) $199_{(10)}$; в) $187_{(10)}$.
6. а) $57_{(10)}$; б) $-31_{(10)}$; в) $-109_{(10)}$.
7. а) $17689_{(10)}$; б) $20461_{(10)}$.
8. а) $26493_{(10)}$; б) $-30785_{(10)}$.

9. а) 0010110001100110; б) 1010001111010000.
10. а) −550,15625; б) 616,15625.
11. а) 407C360000000000; б) 408B594000000000.

Вариант 15

1. а) $280_{(10)}$; б) $700_{(10)}$; в) $464_{(10)}$.
2. а) $010100110011_{(2-10)}$; б) $100100100101_{(2-10)}$; в) $100010010001_{(2-10)}$.
3. Винчестер.
4. 43 6F 6D 70 75 74 65 72 20 49 42 4D 20 50 43.
5. а) $217_{(10)}$; б) $161_{(10)}$; в) $232_{(10)}$.
6. а) $53_{(10)}$; б) $-24_{(10)}$; в) $-110_{(10)}$.
7. а) $23380_{(10)}$; б) $22620_{(10)}$.
8. а) $24236_{(10)}$; б) $-30388_{(10)}$.
9. а) 0100101101100011; б) 1001001000101100.
10. а) 84,15625; б) −681,375.
11. а) 4075E28000000000; б) C07E980000000000.

Вариант 16

1. а) $728_{(10)}$; б) $383_{(10)}$; в) $202_{(10)}$.
2. а) $001100110011_{(2-10)}$; б) $001101100010_{(2-10)}$; в) $010001000100_{(2-10)}$.
3. IBM PC.
4. 8A AE AC AF EC EE E2 A5 E0.
5. а) $170_{(10)}$; б) $242_{(10)}$; в) $158_{(10)}$.
6. а) $70_{(10)}$; б) $-50_{(10)}$; в) $-90_{(10)}$.
7. а) $21581_{(10)}$; б) $31014_{(10)}$.
8. а) $19903_{(10)}$; б) $-17431_{(10)}$.
9. а) 0011111110001000; б) 1001011111011111.
10. а) 650,375; б) −974,5.
11. а) C05DCA0000000000; б) 408E5B0000000000.

Вариант 17

1. а) $158_{(10)}$; б) $177_{(10)}$; в) $439_{(10)}$.
2. а) $000100110101_{(2-10)}$; б) $001010010011_{(2-10)}$; в) $0001000000100100_{(2-10)}$.
3. Автоматизация.
4. 50 72 6F 67 72 61 6D.
5. а) $172_{(10)}$; б) $247_{(10)}$; в) $216_{(10)}$.
6. а) $104_{(10)}$; б) $-67_{(10)}$; в) $-88_{(10)}$.
7. а) $17134_{(10)}$; б) $17996_{(10)}$.
8. а) $24197_{(10)}$; б) $-19851_{(10)}$.
9. а) 0001010110011011; б) 1001010000111010.
10. а) 423,15625; б) 835,15625.
11. а) 4089794000000000; б) 408B414000000000.

Вариант 18

1. а) $328_{(10)}$; б) $537_{(10)}$; в) $634_{(10)}$.
2. а) $000100000100_{(2-10)}$; б) $010110011001_{(2-10)}$; в) $100000110111_{(2-10)}$.
3. Информатика.
4. 50 72 6F 63 65 64 75 72 65.
5. а) $203_{(10)}$; б) $199_{(10)}$; в) $214_{(10)}$.

6. а) $87_{(10)}$; б) $-50_{(10)}$; в) $-31_{(10)}$.
7. а) $17130_{(10)}$; б) $27910_{(10)}$.
8. а) $26837_{(10)}$; б) $-17264_{(10)}$.
9. а) 0100011000011101; б) 1101001111000101.
10. а) −197,15625; б) −341,375.
11. а) C057D80000000000; б) 406F0C0000000000.

Вариант 19

1. а) $1026_{(10)}$; б) $725_{(10)}$; в) $100_{(10)}$.
2. а) $100110010110_{(2-10)}$; б) $100100110010_{(2-10)}$; в) $000110010000_{(2-10)}$.
3. Computer.
4. 84 88 91 8A 8E 82 8E 84.
5. а) $173_{(10)}$; б) $149_{(10)}$; в) $129_{(10)}$.
6. а) $73_{(10)}$; б) $-117_{(10)}$; в) $-39_{(10)}$.
7. а) $24335_{(10)}$; б) $28591_{(10)}$.
8. а) $19650_{(10)}$; б) $-27052_{(10)}$.
9. а) 0110010000000000; б) 1111111001010100.
10. а) 612,15625; б) −652,546875.
11. а) 40664C0000000000; б) 40684C0000000000.

Вариант 20

1. а) $853_{(10)}$; б) $135_{(10)}$; в) $66_{(10)}$.
2. а) $100001111001_{(2-10)}$; б) $100000010000_{(2-10)}$; в) $001101000100_{(2-10)}$.
3. Printer.
4. 43 4F 4D 50 55 54 45 52.
5. а) $178_{(10)}$; б) $240_{(10)}$; в) $152_{(10)}$.
6. а) $54_{(10)}$; б) $-10_{(10)}$; в) $-43_{(10)}$.
7. а) $18083_{(10)}$; б) $19157_{(10)}$.
8. а) $18477_{(10)}$; б) $-28803_{(10)}$.
9. а) 0101010001100111; б) 1110101001001100.
10. а) 575,375; б) 983,375.
11. а) C088440000000000; б) C0696C0000000000.

Вариант 21

1. а) $206_{(10)}$; б) $382_{(10)}$; в) $277_{(10)}$.
2. а) $011101100101_{(2-10)}$; б) $010001110111_{(2-10)}$; в) $011101010000_{(2-10)}$.
3. Компьютеризация.
4. 50 52 49 4E 54.
5. а) $234_{(10)}$; б) $254_{(10)}$; в) $192_{(10)}$.
6. а) $120_{(10)}$; б) $-110_{(10)}$; в) $-112_{(10)}$.
7. а) $19743_{(10)}$; б) $30381_{(10)}$.
8. а) $30643_{(10)}$; б) $-23233_{(10)}$.
9. а) 0111100111001110; б) 1001100000100111.
10. а) −503,15625; б) 339,375.
11. а) C06EA50000000000; б) C08E230000000000.

Вариант 22

1. а) $692_{(10)}$; б) $844_{(10)}$; в) $1014_{(10)}$.
2. а) $010101100010_{(2-10)}$; б) $100100100111_{(2-10)}$; в) $001001000101_{(2-10)}$.

3. YAMAHA.
4. 4D 4F 44 45 4D.
5. а) $215_{(10)}$; б) $229_{(10)}$; в) $241_{(10)}$.
6. а) $101_{(10)}$; б) $-34_{(10)}$; в) $-56_{(10)}$.
7. а) $23242_{(10)}$; б) $17599_{(10)}$.
8. а) $25657_{(10)}$; б) $-29323_{(10)}$.
9. а) 0010101000011001; б) 1011000010001010.
10. а) 654,546875; б) 494,375.
11. а) C0642C0000000000; б) C082F14000000000.

Вариант 23

1. а) $707_{(10)}$; б) $133_{(10)}$; в) $1023_{(10)}$.
2. а) $001010000011_{(2-10)}$; б) $010000000011_{(2-10)}$; в) $001010000001_{(2-10)}$.
3. Световое перо.
4. 4C 61 73 65 72.
5. а) $136_{(10)}$; б) $202_{(10)}$; в) $207_{(10)}$.
6. а) $85_{(10)}$; б) $-44_{(10)}$; в) $-66_{(10)}$.
7. а) $17949_{(10)}$; б) $27584_{(10)}$.
8. а) $27445_{(10)}$; б) $-31187_{(10)}$.
9. а) 0100011111000100; б) 1011001111110000.
10. а) 446,15625; б) −455,375.
11. а) 408B894000000000; б) C089930000000000.

Вариант 24

1. а) $585_{(10)}$; б) $239_{(10)}$; в) $361_{(10)}$.
2. а) $011010000001_{(2-10)}$; б) $100001010001_{(2-10)}$; в) $001110000111_{(2-10)}$.
3. Микропроцессор.
4. 88 AD E4 AE E0 AC A0 E2 A8 AA A0.
5. а) $162_{(10)}$; б) $224_{(10)}$; в) $206_{(10)}$.
6. а) $73_{(10)}$; б) $-111_{(10)}$; в) $-66_{(10)}$.
7. а) $17189_{(10)}$; б) $22238_{(10)}$.
8. а) $32549_{(10)}$; б) $-23508_{(10)}$.
9. а) 0011100011010100; б) 1001010101100011.
10. а) −279,375; б) −838,15625.
11. а) 4081C94000000000; б) 403D800000000000.

Вариант 25

1. а) $382_{(10)}$; б) $830_{(10)}$; в) $512_{(10)}$.
2. а) $100000100101_{(2-10)}$; б) $010010010100_{(2-10)}$;
   в) $011000000011_{(2-10)}$.
3. Принтер.
4. 42 69 6E 61 72 79.
5. а) $136_{(10)}$; б) $183_{(10)}$; в) $162_{(10)}$.
6. а) $111_{(10)}$; б) $-122_{(10)}$; в) $-61_{(10)}$.
7. а) $21736_{(10)}$; б) $22611_{(10)}$.
8. а) $18894_{(10)}$; б) $-25174_{(10)}$.
9. а) 0000111101011000; б) 1110000000001111.
10. а) 300,546875; б) −400,15625.
11. а) 408EFB0000000000; б) 4078D28000000000.

Дополнительная литература

1. *Аветисян Р.Д., Аветисян Д.В.* Теоретические основы информатики. — М.: РГГУ, 1997.

2. *Агеев В.М.* Теория информации и кодирования: Дискретизация и кодирование измерительной информации. — М.: МАИ, 1977.

3. *Бауэр Ф.Л., Гоос Г.* Информатика. Вводный курс: Пер. с нем. — М.: Мир, 1976.

4. *Брой М.* Информатика: В 3 т. Т. 2. Вычислительные структуры и машинно-ориентированное программирование: Пер. с нем. — М.: Диалог-МИФИ, 1996.

5. *Дмитриев В.И.* Прикладная теория информации. — М., 1989.

6. *Коган И.М.* Прикладная теория информации. — М.: Радио и связь, 1981.

7. *Кузьмин И.В., Кедрус В.А.* Основы теории информации и кодирования. — Киев: Выща шк., 1986.

8. *Мазур М.* Качественная теория информации. — М.: Мир, 1974.

9. *Суханов А.П.* Мир информации. — М.: Мысль, 1986.

10. *Цымбал В.П.* Задачник по теории информации и кодирования. — Киев: Выща шк., 1976.

§ 6. ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ

Рекомендации по проведению занятий

Теория графов — один из фундаментальных разделов дискретной математики. Сведения из этого раздела с 60-х годов традиционно включаются в курсы кибернетики, а затем и информатики, поскольку графы оказались очень продуктивным средством информационного (математического) моделирования структур систем и процессов, представления задач информационного характера.

В курсе информатики целесообразно ограничиться ознакомлением с понятийным аппаратом, основными результатами теории графов и, самое главное, методами представления и характеристики графов с помощью матриц.

Полезна и подготовка рефератов. На семинарских занятиях, помимо обсуждения теоретических вопросов, необходимо решение задач.

Контрольные вопросы

1. Что называется графом? ориентированным графом?
2. Что называется вершинами графа? ребрами?
3. Какие ребра и какие вершины графа называются смежными?
4. Какой граф называется мультиграфом?
5. Какой граф называется полным? дополнением?
6. Что называется петлей? цепью? циклом? путем в графе?
7. Какой граф называется деревом?
8. Что называется суммой графов? пересечением? композицией?
9. Что называется транзитивным замыканием графа? декартовым произведением графов?
10. Что называется степенью графа? Что называется полустепенями исхода и захода графа?

11. Что такое цикломатическое число графа?
12. Что называется хроматическим числом графа? функцией Гранди?
13. С помощью каких матриц можно задать граф?
14. Какой граф называется эйлеровым?
15. В чем состоит содержание теоремы Форда — Фалкерсона?

Темы для рефератов

1. Исторические вехи теории графов.
2. Задачи, сводящиеся к графам.
3. Связность в графах.
4. Графы и отношения на множествах.
5. Теоремы о числах графов.
6. Устойчивость графов.
7. Расстояния и пути в графах.

Темы семинарских занятий

1. Основные понятия теории графов.
2. Операции над графами.
3. Степени и числа графов.
4. Матрицы графов.
5. Расстояния и пути на графах.
6. Задачи, сводящиеся к графам. Транспортные сети и задачи о потоках.

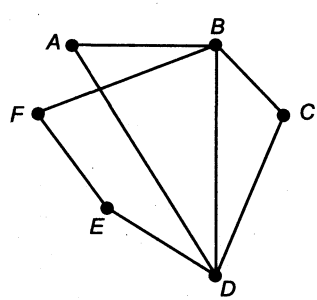
Задачи и упражнения

1. Изобразите графы, имеющие следующие матрицы смежности:

а)
$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

б)
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

2. Получите матрицу смежности графа:



3. Сколько существует различных графов, имеющих n вершин?

4. Пусть граф с вершинами A, B, C, D имеет ребра AB, AC, BD, CD. Используя матрицу смежности этого графа, определить:

а) число маршрутов длины 2 из C в B;

б) число маршрутов длины 3 из A в B;

в) является ли граф связным?

5. Сколько различных ориентированных графов может существовать в заданных N вершинах?

6. Пусть V — множество вершин ориентированного графа. Какова максимальная мощность множества дуг этого графа?

Дополнительная литература

1. *Басакер Р., Саати Т.* Конечные графы и сети: Пер. с англ. — М.: Наука, 1874.

2. *Зыков А.А.* Основы теории графов. — М.: Наука, 1987.

3. *Кристофидес Н.* Теория графов. Алгоритмический подход: Пер. с англ. — М.: Мир, 1978.

4. *Кук Д., Бейз Г.* Компьютерная математика: Пер с англ. — М.: Наука, 1990.

5. *Оре О.* Теория графов: Пер. с англ. — М.: Наука, 1980.

6. Основы кибернетики. Математические основы кибернетики / Под ред. К.А. Пупкова. — М.: Высш. шк., 1974.

7. *Свами М., Тхуласираман К.* Графы, сети и алгоритмы: Пер. с англ. — М.: Мир, 1984.

8. *Татт У.* Теория графов: Пер. с англ. — М.: Мир, 1988.

9. *Уилсон Р.* Введение в теорию графов: Пер. с англ. — М.: Мир, 1977.

10. *Харари Ф.* Теория графов: Пер. с англ. — М.: Мир, 1973.

§ 7. АЛГОРИТМ И ЕГО СВОЙСТВА

Рекомендации по проведению занятий

Тему «Алгоритм и его свойства» традиционно считают главной темой теоретической информатики, вводящей в обширные практические разделы алгоритмизации различные процедурные языки программирования, а также методы программирования. Нельзя не отметить, что в данном случае рассматривается интуитивное понятие алгоритма, а также интуитивно ясные его свойства. Данное обстоятельство приводит к тому, что у разных авторов определения алгоритма несколько различаются, также различаются и наборы свойств алгоритмов. Важно не становиться в данном случае на догматические позиции.

Интересно проследить развитие понятия алгоритма, контекста, в котором оно используется, в последние десятилетия. В математике, например, является традиционным подход к алгоритмам с точки зрения их существования и алгоритмической разрешимости задачи. В 70-е годы в работах по кибернетике алгоритм рассматривается в основном еще в плане процесса вычислений, преимущественно с помощью ЭВМ. В 80-е годы уже в книгах по информатике алгоритм связывается с обработкой информации (данных). Ныне алгоритмы не связываются с обработкой чего-то конкретного и вперед выходят их общие свойства.

Сам по себе алгоритм не может быть введен без исполнителя, два этих понятия являются начальными и неопределяемыми. В случаях, когда исполнитель алгоритма явно не описан, он подразумевается общеизвестным и стандартным. Такой подход к алгоритмам в обучении является методически ошибочным и ведет к затруднениям в усвоении техники программирования на алгоритмических языках.

При изучении данной темы целесообразно применение различных форм занятий — и семинарских, и лабораторно-практических. При выполнении лабораторных работ может быть использован широкий набор программных средств, моделирующих исполнителей на IBM-совместимых ПК. Надо иметь в виду, что основные практические навыки в отношении разработки алгоритмов будут формироваться в дальнейшем при изучении языков и методов программирования. Главная задача проведения занятий по данной теме — пропедевтика изучения в дальнейшем языков и методов программирования ЭВМ. Остается полезной и подготовка рефератов, посвященных истории формирования понятия алгоритма, различиям в трактовке алгоритмов в математике и информатике, известнейшим алгоритмам.

Контрольные вопросы

1. Каково происхождение слова «алгоритм»?
2. Приведите определение алгоритма.
3. Приведите примеры вычислительных алгоритмов, алгоритмов обработки информации и алгоритмов, не направленных на обработку информации.
4. Что такое исполнитель? Приведите примеры.
5. Из каких элементов состоят алгоритмы?
6. Охарактеризуйте способы представления алгоритмов.
7. Какова роль языка в представлении алгоритмов? Что называют «алгоритмическим языком»?
8. В чем состоит свойство дискретности алгоритма?
9. В чем состоит свойство детерминированности (определенности) алгоритма? Можно ли говорить о детерминированности алгоритмов, использующих случайные числа?
10. Что означает свойство направленности (результативности) алгоритма? Можно ли считать алгоритмами процедуры, подразумевающие обработку бесконечных последовательностей чисел?
11. В чем состоит свойство элементарности (локальности) шагов алгоритма?
12. Что означает «массовость алгоритма»?
13. Проинтерпретируйте свойства алгоритмов на приведенных в ответе на вопрос 3 примерах алгоритмов.
14. Каковы основные алгоритмические конструкции?
15. Какие элементы графических схем представления алгоритмов используются для отображения основных алгоритмических конструкций?
16. Каковы основные конструкции алгоритмического языка?

Темы для рефератов

1. История формирования понятия «алгоритм».
2. Известнейшие алгоритмы в истории математики.
3. Проблема существования алгоритмов в математике.
4. Средства и языки описания (представления) алгоритмов.
5. Методы разработки алгоритмов.

Темы семинарских занятий

1. Понятие алгоритма.
2. Средства представления алгоритмов. Основные конструкции алгоритмических языков.
3. Свойства алгоритмов.

Рекомендации по программному обеспечению

Вследствие давних традиций в изучении данной темы имеется большое число программных средств для учебных компьютеров типа «Корвет», «УКНЦ», «Yamaha», остающихся кое-где в эксплуатации, а также для IBM-совместимых компьютеров, моделирующих те или иные исполнители. Многие из них предназначены для обучения информатике детей младшего и среднего школьного возраста (например, программные модели исполнителей «Пылесосик» и «Кенгуренок»). Весьма интересным комплексом является «Роботландия». Однако изучение систем команд этих исполнителей более соответствует задачам курса методики обучения информатике.

Весьма полезным при изучении представления алгоритмов на формальном языке, а также основных алгоритмических конструкций является так называемый Е-практикум или комплекс КУМИР (разработанный на мехмате МГУ в конце 80-х — начале 90-х годов), работающий в среде MS-DOS — файловой оболочки типа Norton Commander. Используя данное программное обеспечение, можно поставить несколько лабораторных работ, обеспечивающих качественное знакомство с представлением алгоритмов на формальном алгоритмическом языке, а также с основными алгоритмическими конструкциями и даже с некоторыми типами данных.

Задачи и упражнения

1. Изобразите алгоритм Евклида для нахождения наибольшего общего делителя положительных чисел a и b с помощью граф-схемы и запишите его на алгоритмическом языке.

2. Изобразите с помощью граф-схемы и запишите на алгоритмическом языке алгоритмы, являющиеся решением следующих задач:

а) пусть задана последовательность $x_1, x_2, x_3, \ldots, x_n$ из n произвольных действительных чисел и число a; требуется подсчитать в этой последовательности количество K чисел $x_i > a$ и количество M чисел $x_i < a$;

б) требуется вычислить сумму $1 + 1/1! + 1/2! + 1/3! + \ldots + 1/n!$ и проверить, что с ростом n эта сумма приближается к основанию натурального логарифма e;

в) с точностью 10^{-5} решить уравнение $x = \sin(x)$, используя метод итераций, т.е. получить последовательность $1, x_1, x_2, \ldots, x_{n-1}, x_n$, где $x_{i+1} = \sin(x_i)$ и $|x_n - x_{n-1}| < 10^{-5}$.

Лабораторные работы

1. Система команд исполнителя алгоритмов.
2. Основные алгоритмические конструкции в языке исполнителя.
3. Разработка алгоритмов на языке исполнителя.

Дополнительная литература

1. *Алферова З.А.* Теория алгоритмов. — М.: Статистика, 1973.
2. *Брой М.* Информатика: В 3 т. Т. 1. Основополагающее введение: Пер. с нем. — М.: Диалог-МИФИ, 1996.
3. *Вирт Н.* Алгоритмы и структуры данных: Пер. с англ. — М.: Мир, 1989.
4. *Гудман С., Хадитниеми С.* Введение в разработку и анализ алгоритмов: Пер. с англ. — М.: Мир, 1981.
5. *Кнут Д.* Искусство программирования для ЭВМ. Т. 1. Основные алгоритмы: Пер. с англ. — М.: Мир, 1976.
6. *Матюшков Л.П., Лихтарович А.А.* Основы машинной математики: Пособие для учителя. — Минск: Нар. Асвета, 1988.
7. Основы кибернетики. Математические основы кибернетики / Под ред. К.А.Пупкова. — М.: Высш. шк., 1974.

§ 8. ФОРМАЛИЗАЦИЯ ПОНЯТИЯ АЛГОРИТМА

Рекомендации по проведению занятий

При изучении данной темы полезны как семинарские занятия, подготовка рефератов, так и лабораторные работы с абстрактными машинами Тьюринга и Поста, а также с нормальными алгоритмами Маркова.

Краткие сведения

Введенного на интуитивном уровне понятия алгоритма и опытным путем установленных свойств алгоритмов вполне достаточно для решения широкого круга математических задач. Достаточно такого подхода и при решении практических задач программирования компьютеров. Однако в тех случаях, когда задача оказывается сложной и ее алгоритмическое решение найти не удается, встает вопрос о ее алгоритмической разрешимости. При этом требуется понятие алгоритма ***формализовать***, т.е. четко определить исполнителя алгоритмов, с помощью которого можно провести доказательство алгоритмической неразрешимости задачи.

В первой половине XX века почти параллельно были разработаны несколько подходов к формализации алгоритмов — рекурсивные функции, абстрактные машины Тьюринга и Поста, λ-исчисление, нормальные алгорифмы (так это традиционно называется в математике) Маркова, впоследствии оказавшиеся эквивалентными. Некоторые из этих подходов оказали воздействие на становление информатики и нашли отражение в производственных языках программирования — таких, как Lisp (λ-исчисление), Рефал (нормальные алгорифмы Маркова). Последний язык используется в исследованиях и разработках в области искусственного интеллекта; кстати, он чуть ли не единственный, разработанный в России и получивший мировое признание.

Формализованные подходы к алгоритмам в информатике обладают высокой образовательной ценностью. Так, абстрактные машины Тьюринга и Поста являются прекрасными средствами освоения алгоритмизации даже при безмашинном ва-

рианте преподавания информатики начиная со средних классов школы. Кроме того, не ослабевает уверенность, что на базе математической теории алгоритмов в ближайшие годы сформируется соответствующий теоретический раздел информатики, так что изучение формальных подходов к алгоритмам несет в себе момент пропедевтики развития науки.

Контрольные вопросы

1. Зачем в математике потребовалось формализовать понятие алгоритма?
2. Какие подходы к уточнению понятия алгоритма существуют?
3. Какие функции называют вычислимыми? Какие функции называют частичными?
4. Какие функции называют частично-рекурсивными?
5. Какой набор простейших функций и элементарных операций используется в теории рекурсивных функций?
6. Какова формулировка тезиса Черча? Что он означает?
7. Каково устройство абстрактной машины Поста? Каковы выполняемые ею команды?
8. Каково устройство абстрактной машины Тьюринга? Каковы выполняемые ею действия?
9. Как описывается машина Тьюринга? Приведите примеры схем машин Тьюринга.
10. Что называется композицией машин Тьюринга?
11. В чем состоит содержание теоремы Тьюринга?
12. Приведите примеры дедуктивных цепочек.
13. Как определяются нормальные алгоритмы Маркова?
14. В чем состоит задача универсального алгоритма?

Темы для рефератов

1. Проблема алгоритмической разрешимости в математике.
2. Основатели теории алгоритмов — Клини, Черч, Пост, Тьюринг.
3. Основные определения и теоремы теории рекурсивных функций.
4. Тезис Черча.
5. Проблемы вычислимости в математической логике.
6. Машина Поста.
7. Машина Тьюринга.
8. Нормальные алгоритмы Маркова и ассоциативные исчисления в исследованиях по искусственному интеллекту.

Темы семинарских занятий

1. Формализация понятия алгоритма и алгоритмическая разрешимость.
2. Основы теории рекурсивных функций.
3. Абстрактные автоматы и уточнение понятия алгоритма. Машины Поста и Тьюринга.
4. Ассоциативные исчисления и нормальные алгоритмы Маркова.

Рекомендации по программному обеспечению

При чтении лекций (если имеется overhead- или мультимедиа-проектор), а также при выполнении лабораторных работ рекомендуется использовать какие-либо из многочисленных программных моделей машин Поста и Тьюринга, разрабатываемых студентами старших курсов при выполнении курсовых и дипломных работ.

Для поддержки изучения нормальных алгоритмов Маркова имеется версия транслятора языка Рефал для 8-разрядных учебных машин типа «Yamaha MSX-2».

Задачи и упражнения

1. Используя рекурсивные функции, постройте:
   а) трехместную функцию сложения;
   б) n-местную функцию сложения.
2. Используя рекурсивные функции, постройте:
   а) двухместную функцию умножения;
   б) трехместную функцию умножения;
   в) n-местную функцию умножения.
3. Постройте частичную двухместную функцию деления.
4. Напишите и самостоятельно исполните программу машины Поста, складывающую два числа, разделенные на ленте:
   а) одним пробелом;
   б) произвольным числом пробелов.
5. Напишите и самостоятельно исполните программу машины Поста, вычитающую два числа, разделенные на ленте:
   а) одним пробелом;
   б) произвольным числом пробелов.
6. Напишите и самостоятельно исполните программу машины Поста, умножающую два числа, разделенные на ленте:
   а) одним пробелом;
   б) произвольным числом пробелов.
7. Напишите и самостоятельно исполните программу машины Поста, делящую одно число на другое, разделенные на ленте одним пробелом.
8. Постройте машины Тьюринга, вычисляющие простейшие арифметические функции.
9. Имея машины Тьюринга, вычисляющие функции g и h, постройте машину, вычисляющую:
   а) суперпозицию этих функций;
   б) функцию, получаемую из g и h примитивной рекурсией.
10. Постройте машину Тьюринга, производящую обращение функции.
11. Постройте машину Тьюринга, вычисляющую:
    а) сумму двух чисел;
    б) разность двух чисел;
    в) произведение двух чисел;
    г) частное двух чисел.
12. Постройте нормальный алгоритм Маркова, реализующий вычитание двух целых чисел, представленных символами 1. Проверьте его работу на примерах.
13. Задайте нормальный алгоритм Маркова, реализующий умножение двух чисел, представленных символами 1.

Лабораторные работы

Лабораторная работа № 1

Машина Поста

Время выполнения 4 часа.

Вариант 1

На ленте машины Поста расположен массив в N отмеченных секциях. Необходимо справа от данного массива через одну пустую секцию разместить массив вдвое больший (он должен состоять из $2N$ меток). При этом исходный массив может быть стерт.

Вариант 2

На ленте машины Поста расположен массив из N меток (метки расположены через пробел). Надо сжать массив так, чтобы все N меток занимали N расположенных подряд секций.

Вариант 3

На информационной ленте машины Поста расположено N массивов меток, отделенных друг от друга свободной ячейкой. Каретка находится над крайней левой меткой первого массива. Определите число массивов.

Вариант 4

Игра Баше. В игре участвуют двое (человек и машина Поста). Напишите программу, по которой всегда будет выигрывать машина Поста. Суть игры заключается в следующем: имеется 21 предмет. Первым ходит человек. Каждый из играющих может брать 1, 2, 3 или 4 предмета. Проигрывает тот, кто берет последний предмет.

Вариант 5

Число k представляется на ленте машины Поста $k + 1$ идущими подряд метками. Одна метка соответствует нулю. Составьте программу прибавления 1 к произвольному числу k. Каретка расположена над одной из меток, принадлежащих заданному числу k.

Вариант 6

Составьте программу сложения двух целых неотрицательных чисел a и b, расположенных на ленте машины Поста. Каретка расположена над одной из меток, принадлежащих числу a. Число b находится правее числа a через несколько пустых секций.

Вариант 7

Составьте программу сложения произвольного количества целых неотрицательных чисел, записанных на ленте машины Поста на расстоянии одной пустой секции друг от друга. Каретка находится над крайней левой меткой левого числа.

Вариант 8

На ленте машины Поста расположен массив из N меток. Составьте программу, действуя по которой машина выяснит, делится ли число на 3. Если да, то после массива через одну пустую секцию поставьте метку V.

Вариант 9

Число k представлено на ленте машины Поста $k + 1$ идущими подряд метками. Найдите остаток от деления целого неотрицательного числа k на 3, если известно, что каретка находится справа от заданного числа.

Вариант 10

Составьте программу нахождения разности двух неотрицательных целых чисел a и b, находящихся на ленте машины Поста. Каретка находится над левой меткой левого числа. Неизвестно, какое число больше: a или b.

Вариант 11

На ленте машины Поста расположен массив из $2N$ отмеченных секций. Составьте программу, по которой машина Поста раздвинет на расстояние в одну секцию две половины данного массива.

Вариант 12

На ленте машины Поста расположен массив из $2N-1$ меток. Составьте программу отыскания средней метки массива и стирания ее.

Вариант 13

На ленте машины Поста расположены два массива. Составьте программу стирания того из массивов, который имеет большее количество меток.

Вариант 14

На информационной ленте машины Поста находятся два массива в M и N меток. Составьте программу выяснения, одинаковы ли массивы по длине.

Вариант 15

Задача В. А. Успенского. На информационной ленте либо вправо, либо влево от секции, над которой расположена каретка, находится массив меток. Расстояние до массива выражается произвольным числом. Необходимо составить программу, работая по которой машина Поста найдет этот массив и установит каретку на начало этого массива.

Вариант 16

Составьте программу умножения двух чисел a и b.

Вариант 17

На ленте машины Поста находится n массивов меток, после последнего массива на расстоянии более трех пустых секций находится одна метка. Массивы разде-

лены тремя пустыми ячейками. Количество меток в массивах не может быть меньше двух. Произвести обработку массивов следующим образом: если количество меток в массиве кратно трем, то стереть метки в данном массиве через одну, иначе — массив стереть полностью. Каретка находится над крайней левой меткой первого массива.

Вариант 18

На ленте изображено n массивов меток, отделенных друг от друга одной свободной ячейкой. Каретка находится над первой меткой первого массива. Определите количество массивов.

Вариант 19

На ленту машины Поста нанесены два массива меток на некотором расстоянии друг от друга. Соедините эти два массива в один. Каретка находится над крайней левой меткой левого массива.

Вариант 20

На ленте машины Поста отмечен массив n меток. Найдите число $2n + 1$ и проверьте, делится ли оно на 3. Если да, то после числа через одну пустую секцию поставьте две метки, если нет — поставьте три метки. Каретка находится над крайней левой отмеченной секцией.

Вариант 21

Дан массив меток. Каретка обозревает первую пустую секцию перед началом массива. Раздвиньте массив так, чтобы после каждой метки была пустая секция.

Вариант 22

Составьте программу сложения произвольного количества чисел, записанных на ленте машины Поста через одну пустую секцию. Каретка обозревает крайнюю левую секцию левого числа.

Вариант 23

Найти НОД двух чисел, находящихся на ленте машины Поста. Между этими числами находится произвольное количество пустых секций. Каретка находится над левой меткой левого числа.

Вариант 24

На ленте машины Поста находятся n массивов меток. Каретка находится где-то над первым массивом. Удалите все массивы с четными номерами (соседние массивы разделены тремя пустыми секциями).

Вариант 25

На информационной ленте машины Поста находится массив меток. Каретка находится где-то над массивом (но не над крайней меткой). Сотрите все метки, кроме крайних, таким образом, чтобы положение каретки при этом не изменилось.

Лабораторная работа № 2

Машина Тьюринга

Время выполнения 4 часа.

Вариант 1

На информационной ленте машины Тьюринга содержится массив символов +. Необходимо разработать функциональную схему машины Тьюринга, которая каждый второй символ + заменит на −. Каретка в начальном состоянии находится где-то над указанным массивом.

Вариант 2

Дано число n в восьмеричной системе счисления. Разработайте машину Тьюринга, которая увеличивала бы заданное число n на 1.

Вариант 3

Дана десятичная запись натурального числа $n > 1$. Разработайте машину Тьюринга, которая уменьшала бы заданное число n на 1. При этом запись числа $n - 1$ не должна содержать левый нуль, например, $100 - 1 = 99$, а не 099. Начальное положение головки — правое.

Вариант 4

Дан массив из открывающихся и закрывающихся скобок. Постройте машину Тьюринга, которая удаляла бы пары взаимных скобок. Например, дано: «) (( ) (( ) », надо получить: «) ... ((. ».

Вариант 5

Дана строка из букв a и b. Разработайте машину Тьюринга, которая переместит все буквы a в левую, а буквы b в правую часть строки. Каретка находится над крайним левым символом строки.

Вариант 6

Даны два целых положительных числа в десятичной системе счисления. Сконструируйте машину Тьюринга, которая будет находить разность этих чисел, если известно, что первое число больше второго, а между ними стоит знак «минус». Каретка находится над левой крайней цифрой левого числа.

Вариант 7

Даны два целых положительных числа в различных системах счисления, одно — в троичной системе, другое — в десятичной. Разработайте машину Тьюринга, которая будет находить сумму этих чисел в десятичной системе счисления.

Вариант 8

Сконструируйте машину Тьюринга, которая выступит в качестве двоично-восьмеричного дешифратора.

Вариант 9

Дана конечная последовательность меток, записанных в клетки ленты подряд, без пропусков. Необходимо разработать машину Тьюринга, которая будет записывать в десятичной системе счисления число этих меток.

Вариант 10

На информационной ленте машины Тьюринга в трех секциях в произвольном порядке записаны три различные буквы: *A*, *B* и *C*. Каретка в начальном состоянии обозревает букву, расположенную справа. Необходимо составить функциональную схему машины Тьюринга, которая сумеет поменять местами крайние буквы.

Вариант 11

Даны два натуральных числа m и n, представленных в унарной системе счисления. Соответствующие наборы символов « | » разделены « − » , вслед за последним символом набора n стоит знак «=». Разработайте машину Тьюринга, которая будет находить разность чисел m и n. При этом результат должен быть записан следующим образом: если $m > n$, то справа от «=» должны стоять знак «+» и набор символов « | » в количестве $m - n$; если $m = n$, то справа от знака «=» должна стоять пустая клетка; если $m < n$, то справа от «=» должны стоять знак «−» и набор символов « | » в количестве $n - m$.

Вариант 12

Даны два натуральных числа n и m, заданных в унарной системе счисления. Числа n и m представлены наборами символов « | » , разделенных « \ ». В конце набора стоит «=». Разработайте машину Тьюринга, которая будет производить деление нацело двух натуральных чисел n и m и находить остаток от деления. При этом результат должен быть записан следующим образом: после «=» должен находиться набор символов « | » частного (он может быть и пустым), после чего ставится знак «(», за которым следует набор символов « | » остатка от деления n на m.

Вариант 13

На ленте машины Тьюринга находится число, записанное в десятичной системе счисления. Умножьте это число на 2, если каретка находится над крайней левой цифрой числа.

Вариант 14

На ленте машины Тьюринга находится целое положительное число, записанное в десятичной системе счисления. Найдите произведение этого числа на число 11. Каретка обозревает крайнюю правую цифру числа.

Вариант 15

На ленте машины Тьюринга находится слово, состоящее из букв латинского алфавита {*a*, *b*, *c*, *d*}. Подсчитайте число букв «*a*» в данном слове и полученное значение запишите на ленту левее исходного слова через пробел. Каретка обозревает крайнюю левую букву.

Вариант 16

На ленте машины Тьюринга находится десятичное число. Определите, делится ли это число на 5 без остатка. Если делится, то запишите справа от числа слово «да», если нет — «нет». Каретка находится где-то над числом.

Вариант 17

На ленте машины Тьюринга записано число в десятичной системе счисления. Каретка находится над крайней правой цифрой. Запишите цифры этого числа в обратном порядке.

Вариант 18

На информационной ленте машины Тьюринга находится десятичное число. Найдите результат целочисленного деления этого числа на 2.

Вариант 19

На информационной ленте машины Тьюринга находится массив, состоящий только из символов *A* и *B*. Сожмите массив, удалив из него все элементы *B*.

Вариант 20

Число n задано на ленте машины Тьюринга массивом меток. Преобразуйте это значение n по формуле:

$$\varphi(n) = \begin{cases} n-2, & \text{если } n > 5, \\ 1, & \text{если } n = 5, \\ 2n, & \text{если } n < 5. \end{cases}$$

Каретка обозревает крайнюю левую метку.

Вариант 21

Число n задано на ленте машины Тьюринга массивом меток. Преобразуйте это значение n по формуле:

$$\varphi(n) = \left[\frac{5n}{4}\right] + 2.$$

Каретка обозревает крайнюю левую метку.

Вариант 22

На ленте машины Тьюринга находится массив $2N$ меток. Уменьшите этот массив в 2 раза.

Вариант 23

Даны два натуральных числа n и m, представленные в унарной системе счисления. Между этими числами стоит знак «?». Выясните отношение m и n, т.е. знак «?» замените на один из подходящих знаков «>», «<», «=».

Вариант 24

Найдите произведение двух натуральных чисел m и n, заданных в унарной системе счисления. Соответствующие наборы символов «|» разделены знаком «*», а

справа от последнего символа правого члена стоит знак « = ». Поместите результат умножения этих чисел вслед за знаком « = ».

Вариант 25

На информационной ленте машины Тьюринга в трех секциях в произвольном порядке записаны три цифры: 1, 2, 3. Каретка обозревает крайнюю левую цифру. Необходимо составить функциональную схему машины Тьюринга, которая расположит эти цифры в порядке возрастания.

Дополнительная литература

1. *Алферова З.А.* Теория алгоритмов. — М.: Статистика, 1973.
2. *Брой М.* Информатика: В 3 т. Т. 1. Основополагающее введение: Пер. с нем. — М.: Диалог-МИФИ, 1996.
3. *Мальцев А.И.* Алгоритмы и вычислимые функции. — М.: Наука, 1986.
4. *Манин Ю.И.* Вычислимая и невычислимое. — М.: Сов. радио, 1980.
5. *Марков А.А., Нагорный Н.М.* Теория алгорифмов. — М.: Наука, 1984.
6. *Матюшков Л.П., Лихтарович А.А.* Основы машинной математики: Пособие для учителя. — Минск: Нар. Асвета, 1988.
7. Машины Тьюринга и вычислимые функции: Пер. с нем. — М.: Мир, 1972.
8. *Миков А.И.* Информатика. Введение в компьютерные науки. — Пермь: ПГУ, 1998.
9. Основы кибернетики. Математические основы кибернетики / Под ред. К.А.Пупкова. — М.: Высш. шк., 1974.
10. *Трахтенброт Б.А.* Алгоритмы и вычислительные автоматы. — М.: Сов. радио, 1974.
11. *Успенский В.А.* Машина Поста. — М.: Наука, 1988.

§ 9. ПРИНЦИПЫ РАЗРАБОТКИ АЛГОРИТМОВ И ПРОГРАММ ДЛЯ РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ

Рекомендации по проведению занятий

Разработка алгоритмов и программ для решения прикладных задач является одним из важнейших направлений деятельности специалиста по информатике и остается наиболее стабильным компонентом подготовки этого специалиста в вузе. Крупный недостаток сложившейся методики обучения разработке алгоритмов и программ состоит в том, что она ориентируется на индивидуальную разработку студентами простых учебных программ на языках программирования высокого уровня, не требующих в силу своей простоты применения специальных методов управления проектами, проектирования и кодирования программ, разделения труда в группе программистов. Это ведет к тому, что программистская подготовка выпускников вузов направлена в основном на обучение программистов-индивидуалов, способных лишь на разработку небольших кустарных программ и не готовых к работе в производственных программистских коллективах.

Настоящий параграф представляет собой попытку заполнить пробел в подготовке по управлению проектами и специальным методам разработки больших программных комплексов производственными коллективами программистов. Во всяком случае, предлагаемые здесь рекомендации могут обеспечить пропедевтику дальнейшей подготовки в этом направлении.

Эффективность обучения достигается комплексным применением различных форм учебной работы по данной теме — лекций, самостоятельной работы студентов, семинарско-практических и лабораторных занятий, а также вычислительной практики, курсового и дипломного проектирования, учебно-исследовательской работы студентов. При проведении семинарских занятий предусматривается подготовка рефератов. Лабораторные занятия не требуют использования компьютеров и программного обеспечения и могут быть проведены в форме деловой игры по управлению программистским проектом, коллективному проектированию и разработке программной системы.

Контрольные вопросы

1. Каков размер программ (в строках исходных кодов), который требуется для решения практических задач?
2. Каков жизненный цикл программных систем? Каковы его этапы?
3. Почему требуется определенная методология разработки программных систем?
4. Как развивались, какие ступени развития прошли методологии разработки программ?
5. Каковы основные общие требования к процессу проектирования программных систем?
6. Что такое декомпозиция? пошаговая детализация?
7. Что называется методом разработки программ сверху вниз? снизу вверх?
8. В чем состоит модульный подход к разработке программ?
9. В чем состоит структурный подход к проектированию программных систем?
10. В чем состоит объектный подход к разработке программ?
11. Какие методы обеспечения правильности программ существуют?
12. Что называется тестированием программ?
13. Что называется доказательным программированием?
14. Что называется документированием программ? Какие виды программной документации существуют?

Темы для рефератов

1. Жизненный цикл программных систем.
2. Методы управления проектами при разработке программных систем.
3. Методы проектирования программных систем.
4. Модульный подход к программированию.
5. Структурный подход к программированию.
6. Объектный подход к программированию.
7. Декларативный подход к программированию.
8. Параллельное программирование.
9. Case-технологии разработки программных систем.
10. Доказательное программирование.

Темы семинарских занятий

1. Жизненный цикл программных систем.
2. Методы проектирования программ. Пошаговая детализация (декомпозиция).
3. Модульный, структурный и объектный подходы к проектированию и программированию.
4. Методы обеспечения правильности программ.

Лабораторные работы

1. Деловая игра по управлению проектом по разработке программной системы.
2. Деловая игра по пошаговой детализации при проектировании программ.
3. Документирование программы.
4. Доказательство правильности программы (алгоритма).

Дополнительная литература

1. *Алагич С., Арбиб М.* Проектирование корректных структурированных программ: Пер. с англ. — М.: Радио и связь, 1984.
2. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979.
3. *Брукс Ф. П.* Как проектируются и создаются программные комплексы: Пер. с англ. — М.: Наука, 1971.
4. *Ван Тассел Д.* Стиль, разработка, эффективность и испытание программ: Пер. с англ. — М.: Мир, 1985.
5. *Вирт Н.* Алгоритмы + структуры данных = программы: Пер. с англ. — М.: Мир, 1985.
6. *Вирт Н.* Систематическое программирование: Пер. с англ. — М.: Мир, 1977.
7. *Громов Г. Р.* Национальные информационные ресурсы. Проблемы промышленной эксплуатации. — М.: Наука, 1985.
8. *Громов Г. Р.* Очерки информационной технологии. — М.: Инфоарт, 1992.
9. *Дал У., Дейкстра Э., Хоор К.* Структурное программирование: Пер. с англ. — М.: Мир, 1975.
10. *Дейкстра Э.* Дисциплина программирования: Пер. с англ. — М.: Мир, 1978.
11. *Йордан Э.* Структурное проектирование и конструирование программ: Пер. с англ. — М.: Мир, 1979.
12. *Квиттнер П.* Задачи, программы, вычисления, результаты: Пер. с англ. — М.: Мир, 1980.
13. *Кнут Д.* Искусство программирования для ЭВМ: В 3 т. — М.: Мир, 1976—1978.
14. *Майерс Г.* Искусство тестирования программ: Пер. с англ. — М.: Финансы и статистика, 1982.
15. *Майерс Г.* Надежность программного обеспечения: Пер. с англ. — М.: Мир, 1980.
16. Программирование на параллельных вычислительных системах: Пер. с англ. / Под ред. Р. Бэбба. — М.: Мир, 1991.
17. *Турский В.* Методология программирования. — М.: Мир, 1981.

18. *Фокс Дж.* Программное обеспечение и его разработка: Пер. с англ. — М.: Мир, 1985.
19. *Холстед М.Х.* Начала науки о программах: Пер. с англ. — М.: Финансы и статистика, 1981.
20. *Хьюз Дж., Митчом Дж.* Структурный подход к программированию: Пер. с англ. — М.: Диалог-МИФИ, 1994.

Тесты к главе 1

Введение в информатику

1.Что понимается под информацией в кибернетике:
1) СУБД; 2) автоматизированная обучающая система;
3) любая совокупность сигналов, воздействий или сведений; 4) килобайты.
2. Что такое кибернетика:
1) наука об общих закономерностях в управлении и связи в различных системах: искусственных, биологических и социальных;
2) наука, изучающая вопросы, связанные со сбором, хранением, преобразованием и использованием информации;
3) наука, изучающая законы механики;
4) раздел науки, изучающей биосистемы.
3. Теоретическая информатика опирается:
1) на законы механики и электричества;
2) законы природы;
3) математическую логику, теорию алгоритмов, теорию кодирования, системный анализ;
4) разделы математики: численный анализ, математический анализ, дифференциальные уравнения.
4. К системному программному обеспечению относятся:
1) новые языки программирования и компиляторы к ним, интерфейсные системы;
2) системы обработки текстов, электронные процессоры, базы данных;
3) решение вопросов об анализе потоков информации в различных сложных системах;
4) поисковые системы, глобальные системы хранения и поиска информации.
5. К прикладному программному обеспечению относятся:
1) новые языки программирования и компиляторы к ним, интерфейсные системы;
2) системы обработки текстов, электронные процессоры, базы данных;
3) решение вопросов об анализе потоков информации в различных сложных системах;
4) поисковые системы, глобальные системы хранения и поиска информации.
6. Вычислительная техника — это:
1) раздел информатики, в котором идет речь о технических деталях и электронных схемах компьютера;
2) раздел информатики, в котором идет речь об архитектуре вычислительных систем, определяющей состав, назначение, принципы взаимодействия устройств;

3) раздел информатики, занимающийся разработкой систем программного обеспечения;
4) раздел информатики, занимающийся вопросами анализа потоков информации.

7. Телематика — это:
1) наука о телекоммуникациях;
2) телеконференция;
3) служба обработки информации на расстоянии (кроме телефона и телеграфа);
4) динамика развития телевидения.

Информационные технологии

1. Информационная технология АСУ — это:
1) система, управляющая работой станка с числовым программным управлением;
2) комплекс технических и программных средств, организующих управление объектами в производстве или общественной сфере;
3) система, помогающая учащимся осваивать новый материал, контролирующая знания;
4) программно-аппаратный комплекс, который позволяет эффективно проектировать механизмы, здания, узлы сложных агрегатов.

2. Информационная технология АСУТП — это:
1) система, управляющая работой станка с числовым программным управлением;
2) комплекс технических и программных средств, организующих управление объектами в производстве или общественной сфере;
3) система, помогающая учащимся осваивать новый материал, контролирующая знания;
4) программно-аппаратный комплекс, который позволяет эффективно проектировать механизмы, здания, узлы сложных агрегатов.

3. Информационная технология АСНИ — это:
1) система, помогающая учащимся осваивать новый материал, контролирующая знания;
2) система, управляющая работой станка с числовым программным управлением;
3) комплекс технических и программных средств, организующих управление объектами в производстве или общественной сфере;
4) программно-аппаратный комплекс, в котором научные приборы сопряжены с компьютером, который производит обработку данных и представляет их в удобной форме.

4. Информационная технология АОС — это:
1) система, управляющая работой станка с числовым программным управлением;
2) система, помогающая учащимся осваивать новый материал, контролирующая знания;
3) программно-аппаратный комплекс, в котором научные приборы сопряжены с компьютером, производящим обработку данных и представляющим их в удобной форме;
4) комплекс технических и программных средств, организующих управление объектами в производстве или общественной сфере.

5. Информационная технология САПР — это:
   1) система, управляющая работой станка с числовым программным управлением;
   2) программно-аппаратный комплекс, в котором научные приборы сопряжены с компьютером, который производит обработку данных и представляет их в удобной форме;
   3) программно-аппаратный комплекс, который позволяет эффективно проектировать механизмы, здания, узлы сложных агрегатов;
   4) комплекс технических и программных средств, организующих управление объектами в производстве или общественной сфере.

Информация

1. Сигнал — это:
   1) сообщение, передаваемое с помощью носителя;
   2) виртуальный процесс передачи информации;
   3) электромагнитный импульс;
   4) световая вспышка.
2. Сигнал будет дискретным в случае:
   1) когда источник вырабатывает непрерывное сообщение;
   2) когда параметр сигнала принимает последовательное во времени конечное число значений;
   3) когда передается с помощью волны;
   4) когда источником посылается всего один бит/с.
3. Сигнал будет непрерывным в случае:
   1) когда параметр сигнала принимает последовательное во времени конечное число значений;
   2) когда источником посылается всего один бит/с;
   3) когда источник вырабатывает непрерывное сообщение;
   4) когда передается с помощью волны.
4. Примером дискретного сигнала является:
   1) видеоинформация; 2) музыка;
   3) человеческая речь; 4) текстовая информация.
5. Примером непрерывного сигнала является:
   1) байт; 2) человеческая речь; 3) буква; 4) текст.
6. Бит — это:
   1) состояние диода: закрыт или открыт; 2) 8 байт;
   3) запись текста в двоичной системе;
   4) наименьшая возможная единица информации.
7. Каково количество информации в сообщении *мама мыла раму:*
   1) 8 байт; 2) 1 байт; 3) 6 бит; 4) 1 Кбайт?
8. Как называется запоминаемая информация:
   1) микроскопической; 2) макроскопической;
   3) пространственной; 4) тождественной?

Кодирование информации

1. Система счисления — это:
   1) подстановка чисел вместо букв;
   2) способ перестановки чисел;

3) принятый способ записи чисел и сопоставления этим записям реальных значений чисел;
4) правила исчисления чисел.

2. Непозиционная система счисления — это:
1) двоичная; 2) восьмеричная;
3) шестнадцатеричная; 4) буквы латинского алфавита.

3. Основанием позиционной системы счисления называется:
1) основание логарифма из формулы перевода чисел в системе;
2) количество правил вычисления в системе;
3) целая часть чисел;
4) число отличных друг от друга знаков, которые используются для записи чисел.

4. Какая запись числа 729,854 в десятичной системе счисления будет верной:
1) 7·10^3 + 2·10^2 + 9·10^1 + 8·10^0 + 5·10^−1 + 4·10^−2;
2) 7·10^2 + 2·10^1 + 9·10^0 + 8·10^−1 + 5·10^−2 + 4·10^−3;
3) 7·10^3 + 2·10^2 + 9·10^1 + 8·10^−1 + 5·10^−2 + 4·10^−3;
4) 7·10^2 + 2·10^1 + 9·10^0 + 8·10^−0 + 5·10^−1 + 4·10^−2?

5. Сложите два числа в двоичной системе счисления: 1101 + 01 равно:
1) 1100; 2) 1110; 3) 1101; 4) 1011.

6. Сложите два числа в двоичной системе счисления: 10101 + 1011 равно:
1) 101010; 2) 010101; 3) 100000; 4) 111111.

7. Умножьте два числа в двоичной системе счисления: 1101·01 равно:
1) 10101; 2) 011011; 3) 10100; 4) 00011.

8. Умножьте два числа в двоичной системе счисления: 01011·101 равно:
1) 1011101; 2) 0101010; 3) 0101111; 4) 0110111.

9. При переводе числа 15 из десятичной системы счисления в двоичную получится число:
1) 1011; 2) 1101; 3) 1101; 4) 1111.

10. При переводе числа 27 из десятичной системы счисления в двоичную получится число:
1) 10011; 2) 11101; 3) 11011; 4) 11110.

11. При переводе числа 35 из десятичной системы счисления в двоичную получится число:
1) 110001; 2) 100011; 3) 111001; 4) 111111.

12. При переводе дробного числа 0,15 из десятичной системы счисления в двоичную получится число:
1) 0,00100110011...; 2) 0,001001001...; 3) 0,010101...; 4) 0,0000100....

13. При переводе дробного числа 0,69 из десятичной системы счисления в двоичную получится число:
1) 0,11011...; 2) 0,010011; 3) 0,101100...; 4) 0,10111....

14. При переводе числа 83,55 из десятичной системы счисления в восьмеричную получится число:
1) 123,4314...; 2) 321,4314...; 3) 123,4134; 4) 312,1432....

15. При переводе дробного числа 14,25 из десятичной системы счисления в двоичную получится число:
1) 1110,01; 2) 1111,10; 3) 001,01; 4) 111,01.

16. При переводе дробного числа 43,32 из десятичной системы счисления в двоичную получится число:
1) 111011,1010...; 2) 101011,010100...;
3) 101011,111...; 4) 010100,0001....

17. При переводе числа 63,42 из десятичной системы счисления в восьмеричную получится число:
1) 70,327; 2) 07,723; 3) 77,327; 4) 70,723.
18. Алфавитом называются:
1) буквы: заглавные и малые, знаки препинания, пробел;
2) множество знаков в произвольном порядке;
3) множество знаков, в котором определен их порядок;
4) множество всех возможных знаков.
19. Правило, описывающее однозначное соответствие букв алфавитов при преобразовании, называется:
1) сообщением; 2) кодом; 3) кодировщиком; 4) декодировщиком.
20. Процедура преобразования сообщения из одного алфавита в другой называется:
1) кодом; 2) кодировщиком; 3) перекодировщиком; 4) перекодировкой.
21. Кодировщиком называется:
1) устройство, обеспечивающее кодирование сообщения;
2) устройство, обеспечивающее декодирование сообщения;
3) правило, по которому производится кодирование;
4) правило, по которому производится декодирование.
22. Декодировщиком называется:
1) устройство, обеспечивающее кодирование сообщения;
2) устройство, обеспечивающее декодирование сообщения;
3) правило, по которому производится кодирование;
4) правило, по которому производится декодирование.
23. Кодирование сообщения происходит:
1) в момент прохождения сообщения по каналам связи;
2) в момент поступления сообщения от источника в канал связи;
3) в момент приема сообщения получателем;
4) в процессе расшифровки сообщения специальной программой.
24. Декодирование сообщения происходит:
1) в момент прохождения сообщения по каналам связи;
2) в момент поступления сообщения от источника в канал связи;
3) в момент приема сообщения получателем;
4) в процессе зашифровки сообщения специальной программой.

Графы

1. Граф задается:
1) множеством точек с координатами;
2) парой множеств: множеством вершин и множеством ребер;
3) множеством вершин;
4) множеством ребер.
2. Если ребра графа определяются упорядоченными парами вершин, то граф называется:
1) семантическим; 2) ориентированным;
3) простым; 4) циклом.
3. Ребра будут параллельными, если:
1) начала и концы ребер совпадают;
2) цепь из этих ребер замкнута;
3) их вершины соединены двумя или более ребрами;

4) они концевые.

4. Петля в графе будет, если:
   1) начала и концы ребер совпадают;
   2) цепь из этих ребер замкнута;
   3) их вершины соединены двумя или более ребрами;
   4) они концевые.

5. Какой граф называется простым:
   1) с замкнутой простой цепью; 2) без петель и параллельных ребер;
   3) если все ребра параллельны; 4) состоящий из одной петли?

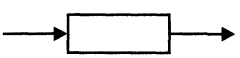
6. Цепь графа — это:
   1) если все определяемые маршрутом ребра смежные;
   2) если ребра в маршруте не образуют петель;
   3) маршрут, в котором все определяемые им ребра различны;
   4) если граф простой.

7. Дерево — это:
   1) неориентированный связный граф;
   2) ориентированный несвязный граф;
   3) граф со смежными вершинами;
   4) ориентированный связный граф.

Алгоритмы

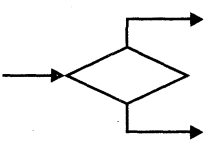
1. Как называется графическое представление алгоритма:
   1) последовательность формул; 2) блок-схема;
   3) таблица; 4) словесное описание?

2. На рисунке представлена часть блок-схемы. Как называется такая вершина:



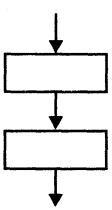
   1) предикатная; 2) объединяющая;
   3) функциональная; 4) сквозная?

3. На рисунке представлена часть блок-схемы. Как называется такая вершина:



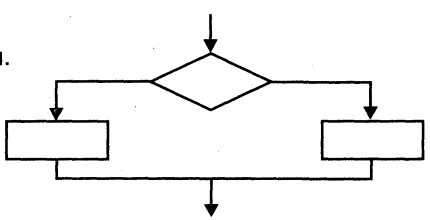
   1) предикатная;
   2) объединяющая;
   3) функциональная;
   4) сквозная?

4. На рисунке представлена часть блок-схемы. Как она называется:



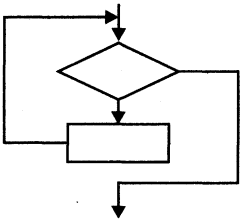
   1) альтернатива;
   2) итерация;
   3) вывод данных;
   4) следование?

5. На рисунке представлена часть блок-схемы. Как она называется:



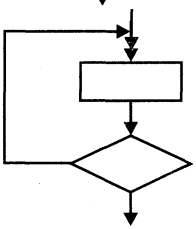
   1) альтернатива;
   2) композиция;
   3) цикл с предусловием;
   4) итерация?

6. На рисунке представлена часть блок-схемы. Как она называется:



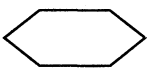
1) альтернатива;
2) композиция;
3) цикл с предусловием;
4) цикл с постусловием?

7. На рисунке представлена часть блок-схемы. Как она называется:



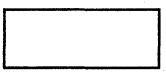
1) альтернатива;
2) композиция;
3) цикл с постусловием;
4) цикл с предусловием?

8. Как называется конструкция блок-схемы, изображенная на рисунке:



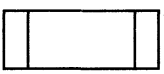
1) выполнение операций;
2) начало-конец алгоритма;
3) вызов вспомогательного алгоритма;
4) ввод/вывод данных?

9. Как называется конструкция блок-схемы, изображенная на рисунке:



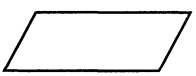
1) выполнение операций;
2) начало-конец алгоритма;
3) вызов вспомогательного алгоритма;
4) ввод/вывод данных?

10. Как называется конструкция блок-схемы, изображенная на рисунке:



1) выполнение операций;
2) начало-конец алгоритма;
3) вызов вспомогательного алгоритма;
4) ввод/вывод данных?

11. Как называется конструкция блок-схемы, изображенная на рисунке:



1) выполнение операций;
2) начало-конец алгоритма;
3) вызов вспомогательного алгоритма;
4) ввод/вывод данных?

12. Свойство алгоритма записываться в виде упорядоченной совокупности отделенных друг от друга предписаний (директив):

1) понятность; 2) определенность; 3) дискретность; 4) массовость.

13. Свойство алгоритма записываться в виде только тех команд, которые находятся в Системе Команд Исполнителя, называется:

1) понятность; 2) определенность;
3) дискретность; 4) результативность.

14. Свойство алгоритма записываться только директивами однозначно и одинаково интерпретируемыми разными исполнителями:

1) детерминированность; 2) результативность;
3) дискретность; 4) понятность.

15. Свойство алгоритма, что при точном исполнении всех предписаний процесс должен прекратиться за конечное число шагов с определенным ответом на поставленную задачу:

1) понятность; 2) детерминированность;
3) дискретность; 4) результативность.

16. Свойство алгоритма обеспечения решения не одной задачи, а целого класса задач этого типа:

1) понятность; 2) определенность;
3) дискретность; 4) массовость.

17. Что называют служебными словами в алгоритмическом языке:

1) слова, употребляемые для записи команд, входящих в СКИ;
2) слова, смысл и способ употребления которых задан раз и навсегда;
3) вспомогательные алгоритмы, которые используются в составе других алгоритмов;
4) константы с постоянным значением?

18. Рекурсия в алгоритме будет прямой, когда:

1) рекурсивный вызов данного алгоритма происходит из вспомогательного алгоритма, к которому в данном алгоритме имеется обращение;
2) порядок следования команд определяется в зависимости от результатов проверки некоторых условий;
3) команда обращения алгоритма к самому себе находится в самом алгоритме;
4) один вызов алгоритма прямо следует за другим.

19. Рекурсия в алгоритме будет косвенной, когда:

1) рекурсивный вызов данного алгоритма происходит из вспомогательного алгоритма, к которому в данном алгоритме имеется обращение;
2) порядок следования команд определяется в зависимости от результатов проверки некоторых условий;
3) команда обращения алгоритма к самому себе находится в самом алгоритме;
4) один вызов алгоритма прямо следует за другим.

20. Команда машины Поста имеет структуру *n K m*, где:

1) *n* — действие, выполняемое головкой; *K* — номер следующей команды, подлежащей выполнению; *m* — порядковый номер команды;
2) *n* — порядковый номер команды; *K* — действие, выполняемое головкой; *m* — номер следующей команды, подлежащей выполнению;
3) *n* — порядковый номер команды; *K*— номер следующей команды, подлежащей выполнению; *m* — действие, выполняемое головкой;
4) *n* — порядковый номер команды; *K* — действие, выполняемое головкой; *m* — номер клетки, с которой данную команду надо произвести.

21. Сколько существует команд у машины Поста:

1) 2; 2) 4; 3) 6; 4) 8?

22. В машине Поста останов будет результативным:

1) при выполнении недопустимой команды;
2) если машина не останавливается никогда;
3) если результат выполнения программы такой, какой и ожидался;
4) по команде «Стоп».

23. В машине Поста некорректным алгоритм будет в следующем случае:

1) при выполнении недопустимой команды;
2) результат выполнения программы такой, какой и ожидался;
3) машина не останавливается никогда;
4) по команде «Стоп».

24. В машине Тьюринга рабочий алфавит:
1) $A = \{a_{40}\ 0,\ b_{40}\ 1,\ c_{40}\ 2,\ \ldots,\ w_{40}\ t\}$;
2) $A = \{a_{40}\ 0,\ a_{40}\ 1,\ a_{40}\ 2,\ \ldots,\ a_{40}\ t\}$;
3) $A = \{a_{40}\ 0,\ a_{41}\ 0,\ a_{42}\ 0,\ \ldots,\ a_{4t}\ 0\}$;
4) $A = \{a_{10}\ 0,\ a_{20}\ 0,\ a_{30}\ 0,\ \ldots,\ a_{90}\ 0\}$.

25. В машине Тьюринга состояниями являются:
1) $\{a_{40}\ 0,\ a_{40}\ 1,\ a_{40}\ 2,\ \ldots,\ a_{40}\ t\}$;
2) $\{q_{41},\ q_{42},\ q_{43},\ \ldots,\ q_{4s}\}$;
3) $\{q_{41},\ q_{42},\ q_{43},\ \ldots,\ q_{4s},\ a_{40}\ 0,\ a_{40}\ 1,\ a_{40}\ 2,\ \ldots,\ a_{40}\ t\}$;
4) $\{q_{40},\ q_{41},\ q_{42},\ \ldots,\ q_{4s}\}$.

26. В машине Тьюринга предписание *L* для лентопротяжного механизма означает:
1) переместить ленту вправо; 2) переместить ленту влево;
3) остановить машину; 4) занести в ячейку символ.

27. В машине Тьюринга предписание *R* для лентопротяжного механизма означает:
1) переместить ленту вправо; 2) переместить ленту влево;
3) остановить машину; 4) занести в ячейку символ.

28. В машине Тьюринга предписание *S* для лентопротяжного механизма означает:
1) переместить ленту вправо; 2) переместить ленту влево;
3) остановить машину; 4) занести в ячейку символ.

29. В алгоритме Маркова ассоциативным исчислением называется:
1) совокупность всех слов в данном алфавите;
2) совокупность всех допустимых систем подстановок;
3) совокупность всех слов в данном алфавите вместе с допустимой системой подстановок;
4) когда все слова в алфавите являются смежными.

30. В ассоциативном счислении два слова называются смежными:
1) если одно из них может быть преобразовано в другое применением подстановок;
2) если одно из них может быть преобразовано в другое однократным применением допустимой подстановки;
3) когда существует цепочка от одного слова к другому и обратно;
4) когда они дедуктивны.

31. В алгоритме Маркова дана цепочка $P\ P_1\ P_2\ \ldots\ P_к$. Если слова $P_1, P_2, \ldots, P_{к-1}$ смежные, то цепочка называется:
1) ассоциативной;
2) эквивалентной;
3) индуктивной;
4) дедуктивной.

32. В алгоритме Маркова дана цепочка $P\ P_1\ P_2\ \ldots\ P_к$. Если слова $P_1, P_2, \ldots, P_{к-1}$ смежные и цепочка существует и в обратную сторону, то слова P и $P_к$ называют:
1) ассоциативными;
2) эквивалентными;
3) индуктивными;
4) дедуктивными.

33. В алгоритмах Маркова дана система подстановок в алфавите $A = \{a, b, c\}$:
abc — c
ba — cb
ca — ab
Преобразуйте с помощью этой системы слово *bacaabc*:
1) *cbc*; 2) *ccbcbbc*; 3) *cbacba*; 4) *cbabc*.

34. В алгоритмах Маркова дана система подстановок в алфавите $A = \{a, b, c\}$:

cb — abc

bac — ac

cab — b

Преобразуйте с помощью этой системы слово *bcabacab:*

1) *ccb*; 2) *cab*; 3) *cbc*; 4) *bcaab.*

35. Способ композиции нормальных алгоритмов будет суперпозицией, если:

1) выходное слово первого алгоритма является входным для второго;
2) существует алгоритм C, преобразующий любое слово p, содержащееся в пересечении областей определения алгоритмов A и B;
3) алгоритм D будет суперпозицией трех алгоритмов A B C, причем область определения D является пересечением областей определения алгоритмов A B и C, а для любого слова p из этого пересечения $D(p) = A(p)$, если $C(p) = e$, $D(p) = B(p)$, если $C(p) = e$, где e — пустая строка;
4) существует алгоритм C, являющийся суперпозицией алгоритмов A и B, такой, что для любого входного слова p $C(p)$ получается в результате последовательного многократного применения алгоритма A до тех пор, пока не получится слово, преобразуемое алгоритмом B.

36. Способ композиции нормальных алгоритмов будет объединением, если:

1) выходное слово первого алгоритма является входным для второго;
2) существует алгоритм C, преобразующий любое слово p, содержащееся в пересечении областей определения алгоритмов A и B;
3) алгоритм D будет суперпозицией трех алгоритмов A B C, причем область определения D является пересечением областей определения алгоритмов A B и C, а для любого слова p из этого пересечения $D(p) = A(p)$, если $C(p) = e$, $D(p) = B(p)$, если $C(p) = e$, где e — пустая строка;
4) существует алгоритм C, являющийся суперпозицией алгоритмов A и B, такой, что для любого входного слова p $C(p)$ получается в результате последовательного многократного применения алгоритма A до тех пор, пока не получится слово, преобразуемое алгоритмом B.

37. Способ композиции нормальных алгоритмов будет разветвлением, если:

1) выходное слово первого алгоритма является входным для второго;
2) существует алгоритм C, преобразующий любое слово p, содержащееся в пересечении областей определения алгоритмов A и B;
3) алгоритм D будет суперпозицией трех алгоритмов A B C, причем область определения D является пересечением областей определения алгоритмов A B и C, а для любого слова p из этого пересечения $D(p) = A(p)$, если $C(p) = e$, $D(p) = B(p)$, если $C(p) = e$, где e — пустая строка;
4) существует алгоритм C, являющийся суперпозицией алгоритмов A и B, такой, что для любого входного слова p $C(p)$ получается в результате последовательного многократного применения алгоритма A до тех пор, пока не получится слово, преобразуемое алгоритмом B.

38. Способ композиции нормальных алгоритмов будет итерацией, если:

1) выходное слово первого алгоритма является входным для второго;
2) существует алгоритм C, преобразующий любое слово p, содержащееся в пересечении областей определения алгоритмов A и B;
3) алгоритм D будет суперпозицией трех алгоритмов A B C, причем область определения D является пересечением областей определения алгоритмов A B и C, а для любого слова p из этого пересечения $D(p) = A(p)$, если $C(p) = e$, $D(p) = B(p)$, если $C(p) = e$, где e — пустая строка;

4) существует алгоритм C, являющийся суперпозицией алгоритмов A и B, такой, что для любого входного слова p $C(p)$ получается в результате последовательного многократного применения алгоритма A до тех пор, пока не получится слово, преобразуемое алгоритмом B.

Структуры данных

1. Какой подход в программировании называется операциональным:
   1) подход, ориентированный на то, что логическая структура программы может быть выражена комбинацией трех базовых структур — следования, ветвления, цикла;
   2) подход, ориентированный на непосредственно выполняемые компьютером операции;
   3) подход, ориентированный на то, что отдельные группы операторов могут объединяться во вспомогательные алгоритмы;
   4) когда задача описывается совокупностью фактов и правил?
2. Какой подход в программировании называется структурным:
   1) подход, ориентированный на непосредственно выполняемые компьютером операции;
   2) подход, не ориентированный на непосредственно выполняемые компьютером операции;
   3) подход, ориентированный на то, что отдельные группы операторов могут объединяться во вспомогательные алгоритмы;
   4) подход, ориентированный на то, что логическая структура программы может быть выражена комбинацией трех базовых структур — следования, ветвления, цикла?
3. Какие данные относятся к неструктурированным:
   1) множества, массивы;
   2) целые числа, действительные числа, логические, символьные;
   3) записи, файлы;
   4) графы, деревья?
4. Какие данные относятся к структурированным:
   1) записи, файлы, множества, массивы;
   2) целые числа, действительные числа, логические, символьные;
   3) списки, стеки;
   4) графы, деревья?
5. Какие данные относятся к динамическим:
   1) записи, файлы, множества, массивы;
   2) целые числа, действительные числа;
   3) логические, символьные;
   4) списки, стеки, графы, деревья?
6. В какой строке таблицы истинности допущена ошибка:
   1) 1;
   2) 2;
   3) 3;
   4) ошибки нет?

X	Y	Not X
0	0	1
0	1	1
1	0	0
1	1	0

7. В какой строке таблицы истинности допущена ошибка:
   1) 2;
   2) 1;
   3) 3;
   4) 4?

X	Y	X and Y
0	0	0
0	1	1
1	0	0
1	1	1

8. В какой строке таблицы истинности допущена ошибка:
   1) 1;
   2) 2;
   3) 3;
   4) 4?

X	Y	X or Y
0	0	0
0	1	1
1	0	0
1	1	1

9. Массивом называется:
   1) набор именованных компонент разного типа, объединенных общим именем;
   2) линейно упорядоченный набор следующих друг за другом компонент;
   3) однородный набор величин одного и того же типа, идентифицируемых вычисляемым индексом;
   4) множество элементов.

10. Записью называется:
   1) набор именованных компонент разного типа, объединенных общим именем;
   2) линейно упорядоченный набор следующих друг за другом компонент;
   3) однородный набор величин одного и того же типа, идентифицируемых вычисляемым индексом;
   4) множество элементов.

11. Очередью называется:
   1) набор именованных компонент разного типа, объединенных общим именем;
   2) линейно упорядоченный набор следующих друг за другом компонент;
   3) однородный набор величин одного и того же типа, идентифицируемых вычисляемым индексом;
   4) множество элементов.

12. Когда доступ к элементам осуществляется следующим образом: новые компоненты могут добавляться только в хвост и значения компонент могут читаться только в порядке следования от головы к хвосту, то эта структура:
   1) массив; 2) очередь;
   3) множество; 4) запись.

13. Когда доступ к элементам осуществляется в любой момент времени и к любому элементу с помощью индексов, то эта структура:
   1) массив; 2) очередь;
   3) множество; 4) запись.

14. Когда доступ к элементам осуществляется только путем проверки принадлежности элемента к структуре, то эта структура:
   1) массив;
   2) очередь;
   3) множество;
   4) запись.

Правильные ответы

Введение в информатику

№	1	2	3	4
1			X	
2	X			
3			X	
4	X			
5		X		
6		X		
7			X	

Информационные технологии

№	1	2	3	4
1		X		
2	X			
3				X
4		X		
5			X	

Информация

№	1	2	3	4
1	X			
2		X		
3			X	
4				X
5		X		
6				X
7		X		
8		X		

Кодирование информации

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1				X	9				X	17			X	
2				X	10			X		18			X	
3				X	11		X			19		X		
4		X			12	X				20				X
5			X		13			X		21	X			
6			X		14	X				22		X		
7		X			15	X				23		X		
8				X	16		X			24			X	

Графы

№	1	2	3	4
1		X		
2		X		
3			X	
4	X			
5		X		
6			X	
7	X			

Алгоритмы

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1		X			14	X				27	X			
2			X		15				X	28			X	
3	X				16				X	29			X	
4				X	17		X			30		X		
5	X				18			X		31				X
6			X		19	X				32		X		
7			X		20		X			33		X		
8		X			21			X		34				X
9	X				22				X	35	X			
10			X		23			X		36		X		
11				X	24			X		37			X	
12			X		25				X	38				X
13	X				26		X							

Структуры данных

№	1	2	3	4	№	1	2	3	4
1		X			8			X	
2				X	9			X	
3		X			10	X			
4	X				11		X		
5				X	12		X		
6				X	13	X			
7	X				14			X	

Глава 2

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ

Изучение программного обеспечения ЭВМ при подготовке учителя информатики предполагает как ознакомление с основными принципами построения современных программных средств, так и выработку практических навыков пользования типичными программами основных классов. В силу этого практическая часть курса может включать следующие виды учебной деятельности:

- проведение семинарских занятий по многим вопросам, по которым проведение узко практических (лабораторных) занятий невозможно или нецелесообразно;
- написание рефератов (с их последующей защитой);
- выполнение лабораторных работ с целью практического освоения ряда программ и приобретения навыков их использования.

Практикум, приведенный в данной главе, рассчитан на поддержку 3—4-семестрового курса «Программное обеспечение ЭВМ». В силу исторически сложившихся причин профессиональная образовательная программа подготовки бакалавра образования по направлению «Естествознание» (в классификаторе 2000 г. направление разделено на два — «Естественно-научное образование» и «Физико-математическое образование») и учителя информатики не охватывает разные разделы программного обеспечения с одинаковой глубиной. Наибольшей детализации подвергается изучение операционных систем персональных ЭВМ (в настоящее время это Windows) и наиболее широко используемое прикладное программное обеспечение общего назначения — системы обработки текстов, компьютерной графики, базы данных, электронные таблицы. При изучении остальных разделов программного обеспечения, описанных в главе 2 базового учебника, реже ставится цель развития практических навыков (это может быть реализовано, например, в спецкурсах), и достаточно бывает ограничиться теоретическим изучением и написанием реферата.

Эти обстоятельства обусловливают неоднородность параграфов данной главы. Для некоторых из них даются краткие сведения, дополняющие базовый учебник или представляющие краткую «выжимку» из теоретического материала. Контрольные вопросы позволят в целом оценить степень готовности к проведению семинарских занятий и лабораторных работ. Литература, указанная в списках, поможет готовить рефераты, получать более детальную информацию о технике работы с программами.

Существенной частью практикума по данному разделу является выполнение лабораторных работ. Цель — выработка и закрепление практических навыков в освоении основных программных средств информационных технологий обработки текстов, графики, вычислений и т.п.

Поскольку существует множество однотипных программных средств и они динамично развиваются, выполнение работ можно осуществлять на имеющемся в наличии программном обеспечении.

В практикуме даны рекомендации по наиболее популярным и распространенным программам, в основном с ориентацией на MS OFFICE, но это не означает необходимость использования именно этих программных продуктов.

§ 1. ОПЕРАЦИОННЫЕ СИСТЕМЫ

Рекомендации по проведению занятий

Операционные системы занимают особое место среди остальных видов программного обеспечения. Это наиболее сложный его вид, охватывающий разнообразные системы и к тому же быстро развивающийся. В соответствии с этим практикум по операционным системам является наиболее сложной и ответственной частью подготовки по программному обеспечению ЭВМ. В настоящем пособии мы ограничиваемся наиболее распространенными (и простыми) операционными системами корпорации Microsoft для персональных IBM-совместимых компьютеров. Практикум носит выраженный пользовательский характер и не ставит перед собой цели подготовить специалистов по разработке системных утилит, драйверов и т.д. В других разделах практикума мы рассмотрим операционные системы UNIX и Novel Netware.

По теме «Операционные системы» целесообразны проведение как семинарских, так и лабораторных занятий, а также подготовка рефератов. Однако число семинарских занятий должно быть небольшим; основное внимание следует уделить практическому овладению операционной системой WINDOWS'95(98).

Краткие сведения

MS DOS

Основные структурные компоненты

Основные структурные компоненты MS DOS (Microsoft Disk Operation System) таковы:

- системный загрузчик (SB);
- базовые модули — io.sys, ms-dos.sys;
- командный процессор (или интерпретатор команд) — command.com;
- драйверы устройств (т.е. программы, поддерживающие их работу);
- утилиты DOS (внешние команды DOS).

Охарактеризуем коротко основные компоненты MS DOS.

Системный загрузчик ОС (SB) — это короткая программа, находящаяся в первом секторе диска с операционной системой. Функция этой программы заключается в считывании в память остальных модулей ОС.

Системный загрузчик проверяет наличие на диске ядра операционной системы, состоящего из файлов с названиями io.sys и ms-dos.sys, загружает их в оперативную память.

Модули ОС (io.sys и ms-dos.sys) загружаются в память загрузчиком ОС и остаются в памяти компьютера постоянно. (Файл io.sys представляет собой дополнение BIOS, а файл ms-dos.sys реализует основные высокоуровневые услуги DOS.)

Командный процессор DOS (command.com) отыскивает и запускает на исполнение файл настройки ОС — config.sys, программу автозапуска (файл autoexec.bat), если она есть, а также обрабатывает команды, введенные пользователем. Некоторые команды пользователя (например, type, dir, copy) командный процессор выполняет сам. Такие команды называют *внутренними*. Для выполнения остальных (внешних) команд пользователя командный процессор ищет на дисках программу с соответствующим именем и если находит ее, то загружает в память и передает ей управление. По окончании работы программы Command.com удаляет программу из памяти и выводит сообщение о готовности к выполнению команд — приглашение DOS. Список команд приведен ниже.

Файловая система MS DOS поддерживает дисководы, обозначаемые латинской буквой и двоеточием, например:

a:, b:, c:

иерархическую систему каталогов, заимствованную у системы UNIX, файлы с именами до 8 символов и расширением до трех.

Внешние команды DOS — это программы, поставляемые вместе с ОС в виде отдельных файлов. Они выполняют действия обслуживающего характера, например, форматирование дискет, проверку дисков и т.д. Список команд см. ниже.

Драйверы устройств — это специальные программы, которые дополняют систему ввода-вывода DOS и обеспечивают обслуживание новых или нестандартное использование имеющихся устройств. Драйверы загружаются в память компьютера при загрузке ОС, их имена указываются в специальном файле (config.sys). Такая схема облегчает добавление новых устройств и позволяет делать это, не затрагивая системные файлы DOS.

Последовательность загрузки MS DOS

При включении электропитания компьютера (или при нажатии на клавишу RESET на корпусе компьютера, или при одновременном нажатии клавиш CTRL, ALT и DEL на клавиатуре) начинают работать программы проверки оборудования, находящиеся в постоянной памяти компьютера. Если они находят ошибку, то выводят сообщения на экран.

После окончания тестирования программа начальной загрузки пытается прочесть с дискеты, установленной на дисководе (A:), программу-загрузчик ОС. Если в дисководе нет дискеты, то загрузка ОС будет производиться с жесткого диска.

После того как с диска, с которого загружается ОС, прочитана программа-загрузчик ОС, эта программа считывает модули ОС (io.sys и ms-dos.sys) и передает им управление.

Далее с того же диска читается файл конфигурации системы config.sys и в соответствии с указаниями, содержащимися в этом файле, загружаются драйверы устройств и устанавливаются параметры ОС. Если такой файл отсутствует, параметры устанавливаются по умолчанию.

После этого с диска, с которого загружается ОС, читается командный процессор (command.com) и ему передается управление. Командный процессор выполняет командный файл (autoexec.bat), если этот файл имеется в корневом каталоге диска, с которого загружается ОС. В этом файле указывают команды и программы, выполняемые при каждом запуске компьютера. Если такой файл не найден, то DOS запрашивает у пользователя текущую дату и время.

После выполнения этого файла процесс загрузки ОС заканчивается. DOS выдает приглашение, показывающее, что она готова к приему команд, например, **C:\>**.

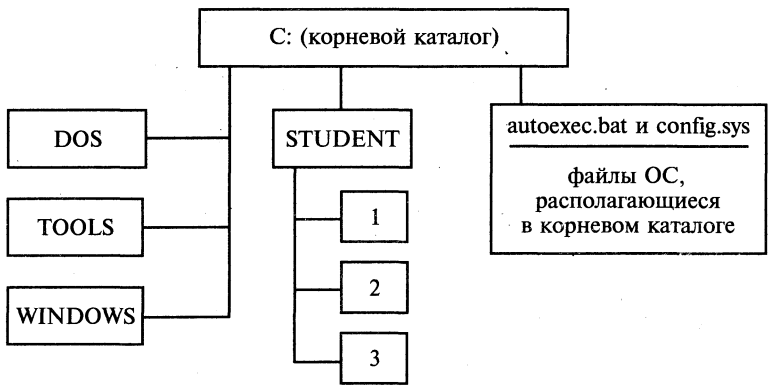
Одними из основных понятий ОС являются файл и каталог (директория). ***Файл*** — это однородная по своему назначению совокупность информации, хранящаяся на диске и имеющая имя.

В ОС MS DOS *имя файла* состоит из двух частей: собственно имени и расширения (т.е. типа файла). Имя файла может содержать не более восьми символов. Нельзя употреблять знаки арифметических операций, пробела, отношений, пунктуации. Расширение имени файла может содержать от одного до трех символов, оно необязательно. Расширение говорит о типе файла. Например, расширение doc обычно имеют документы текстового процессора Word, расширение bas — это файлы с текстами программ на языке Бейсик.

Нельзя давать файлам имена, совпадающие с именами устройств.

Расширение от основного имени отделяется точкой, например, Readme.doc, flag.bmp.

Группу файлов можно объединять в *каталоги*. Кроме файлов, в каталогах могут содержаться подкаталоги. В результате на диске образуется иерархическая система каталогов в виде дерева, например:



В каталогах хранятся также сведения о размерах файлов, сведения о времени создания или последнего изменения файла, атрибуты файла и т.д.

Самым первым является корневой каталог. В данном примере это каталог диска **C:**

Групповые спецификации

При операциях копирования, перемещения, удаления файлов и т.п. для облегчения работы файлы можно объединять в группы. Для объединения файлов используют следующие значки:

* — заменяет любое число символов в имени файла или его расширении;

? — заменяет один произвольный символ.

Примеры:

Обозначение	Пояснения к команде
*.doc	Группа файлов, имеющих расширение DOS
n*g.*	Группа всех файлов, имеющих первую букву в имени файла *n* и последнюю букву *g*. Расширение может быть любым
m?h.*	Группа всех файлов, имя файла которых состоит из трех букв, первая из которых *m*, последняя *h*. Расширение может быть любым

Список внутренних команд

Внутренние команды MS DOS встроены в командный процессор command.com и работают под его управлением. Команды вводятся с клавиатуры, их ввод завершается нажатием клавиши <ВВОД> (<ENTER>). Набирать команды DOS можно как строчными буквами, так и прописными. В формате записи команд в квадратных скобках помечены необязательные параметры.

Команда	Пример	Пояснения к команде
DIR [диск:][путь] [имя файла] [/P] [W] Выводит на экран список директорий и файлов	DIR	Просмотр содержимого в текущем каталоге
	DIR A:\NC	Просмотр содержимого, находящегося на диске A: в каталоге NC
	DIR K*.*	Просмотр списка файлов текущего каталога, начинающихся на букву K
	DIR *.txt	Просмотр списка всех файлов с расширением txt
	DIR A?.*	Просмотр списка файлов с именами из двух знаков, первый из которых буква *a*, и произвольными расширениями
	DIR C:\NC /P	Просмотр содержимого каталога NC, находящегося на диске C:, «порциями» (постранично)
	DIR /W	Выводит информацию в сокращенном виде — только имена файлов и директориев (в 5 столбцов)
DEL [диск:] [путь] имя _файла [/P] Удаляет файлы	DEL A:\NC*.txt	Удаление всех файлов с расширением txt из каталога NC диска A:
	DEL A:\nc*.txt /P	Повторяет предыдущую команду, только перед удалением каждого файла просит подтвердить удаление (нажать клавишу Y («yes»))
	DEL *.pcx	Удаление всех файлов с расширением pcx из текущего каталога
COPY [диск:] [путь\] имя_файла1 [диск:] [путь\] [имя_файла2] Копирует один и более файлов в указанное место	COPY C:\STUDENT\pismo.doc A:\TEKST	Копирование файла с именем pismo.doc из каталога STUDENT диска C: в директорию TEKST, находящуюся на диске A:
	COPY *.doc A:	Копирование всех файлов из текущего каталога в корневой каталог диска A:
	copy con pismo.txt	Создание текстового файла с именем pismo.txt. Ввести необходимый текст. Для сохранения файла и выхода из режима редактирования текста нажать F6 или Ctrl+Z, затем Enter. Появится сообщение о записи файла на диск в текущей директории

Команда	Пример	Пояснения к команде
RENAME [диск:][путь]имя_файла1 имя_файла2 Переименование файла файл1 — старое имя файл2— новое имя	RENAME pismo.doc letter.doc	Переименование файла pismo.doc в текущем каталоге в файл letter.doc
	C:\>RENAME A:\flag.bmp ship.bmp	Переименование файла flag.bmp в корневом каталоге диска A: в файл ship.bmp (текущим является диск C:)
MD [диск:][путь] Создает новый каталог (директорий)	MD C:\STUDENT\NATALI	Создание подкаталога с именем NATALI в каталоге STUDENT диска C:
	mkdir MY	Создание каталога под именем MY в текущем каталоге на текущем диске
CD [диск:] [путь] Позволяет перейти в другой директорий	CD NC	Переход в каталог NC из каталога верхнего уровня
	C:\DOS>CD \WINDOWS	Переход из текущего каталога в каталог WINDOWS. Левая косая черта в начале пути (перед WINDOWS) заставляет идти ОС через корневой каталог
	CD..	Переход на уровень вверх
	CD \	Переход в корневой каталог текущего диска
RD [диск:][путь] Удаление каталога. При удалении каталог должен быть пустым	RD MY	Удаление директории MY, при этом надо находиться на уровень выше удаляемой директории
	RD \GRAFIK\SPIRAL	Каталог SPIRAL является подкаталогом каталога GRAFIK. Удаление каталога SPIRAL из любого места, например, даже если текущим является другой каталог C:\WINDOWS>
DATE Отображает текущую системную дату MS DOS. Если команда DATE вводится без параметров, то она выводит на экран текущую дату. Если нажать Enter, то эта дата останется неизменной. Если необходимо, можно ввести новую дату	DATE 01.09.99	Установление текущей даты
	DATE	Просмотр текущей даты
TIME Индикация/установка системных часов MS DOS	TIME	Вывод текущего системного времени. Последующим вводом с клавиатуры можно поменять показания часов
	TIME 17.35.01	Изменение системного времени

Команда	Пример	Пояснения к команде
VER	VER	Вывод на экран дисплея номера версии MS DOS
TYPE [диск:] [путь] имя_файла Вывод на экран содержимого текстового файла. Если текст большой, его можно просматривать порциями, нажимая Ctrl+S для останова вывода до нажатия любой клавиши, после чего вывод будет продолжен	TYPE A:\TEXT\pismo.txt	Просмотр содержимого файла pismo.txt, находящегося на диске A: в директории TEXT
	TYPE text.txt |more	Вывод на экран содержимого указанного файла постранично, т.е.через каждые 24 строки
EXIT	EXIT	Возврат в прикладную программу из MS DOS. Например, если из Windows переключились в режим MS DOS, для возврата обратно используется данная команда

Имеется еще несколько команд, на практике используемых реже, о которых можно узнать из руководства по операционной системе.

Внешние команды

Помимо команд, распознаваемых и выполняемых командным процессором, в операционной системе имеется большое число утилит — команд, реализованных в виде отдельных программ. Рассмотрим наиболее часто используемые команды.

Команда	Пример	Пояснения к команде
FORMAT.com FORMAT [диск:] /f:V /q /s V — задание объема диска. Максимальный объем форматирования для дискет с двумя отверстиями — 1,44 Мбайт, для дискет с одним отверстием — 720 кбайт	FORMAT A: /q	Быстрое форматирование дискеты, объем по умолчанию 1,44 Мбайт
	FORMAT A: /F:720 /S	Форматирование дискеты, объем 720 Кбайт и копирование файлов ОС на диск
SYS.com SYS [путь] d: Переносит скрытые системные файлы io.sys, msdos.sys и command.com на требуемый диск	SYS A:	Перенос системных файлов из текущей директории (где есть системные файлы) на диск A:
Mem.exe MEM [/c] [/p] Получение информации о распределении памяти компьютера	MEM /c /p	Вывод постранично на экран списка программ, расположенных в оперативной памяти компьютера, указывая их местоположение и размер (значение параметра в десятичном коде)

Окончание таблицы

Команда	Пример	Пояснения к команде
	MEM	Без параметров команды выводит лишь резюме с информацией об имеющемся в распоряжении объеме памяти
PRINT.exe PRINT имя_файла	PRINT TEXT.txt	Вывод на принтер текстового файла с именем TEXT.txt

Большое число утилит MS DOS описано в руководстве по этой системе. Важное значение имеют также драйверы, особенно расширенной оперативной памяти, входящие в состав ОС и позволяющие использовать более чем 640 Кбайт памяти.

Командный файл автозагрузки ОС AUTOEXEC.BAT и файл настройки ОС CONFIG.SYS

Особую роль в системе играют файлы CONFIG.SYS и AUTOEXEC.BAT, читаемые при загрузке системы и задающие ее конфигурацию, загружаемые в память драйверы и резидентные программы, а также дополнительные команды, выполняемые при загрузке системы.

CONFIG.SYS выполняется до загрузки командного процессора и содержит вызовы SYS-драйверов. Загружаемые драйверы устанавливаются командой DEVICE, после которой указывается полное имя файла, содержащего драйвер.

Фактически AUTOEXEC.BAT — это список команд, которые нам пришлось бы вводить вручную в начале каждого сеанса работы.

Примеры:

AUTOEXEC.bat

Команда	Пояснения
@echo off	Отключение режима вывода на экран системных сообщений, за исключением сообщений об ошибках
Path c:\window; c:\windows\command; c:\dos; c:\tools\c:\nc	Установка каталогов, в которых автоматически организуется поиск введенных в командной строке файлов, а затем их запуск
lh keurus.com	Загрузка программы-русификатора в UMB (Upper Memory Blocks) — участки верхней области памяти
lh dfimcom.com	Загрузка драйвера мыши в UMB
prompt pg	Установка формата приглашения ОС в командной строке
lh nc	Загрузка системной оболочки Norton Commander в UMB

CONFIG.sys

Команда	Пояснения
Device=c:\windows\himem.sys	Установка драйвера управления расширенной памятью (HMA — High Memory Area — первые 64 Кбайт расширенной памяти)

Окончание таблицы

Команда	Пояснения
Device=c:\windows\emm.exe	Установка режима использования области верхней памяти (UMB — Upper Memory Blocks)
dos=high,umb	Загрузка ядра ОС в области верхней и расширенной памяти
Lastdrive=z	Установка числа логических дисков
files=40	Число одновременно открываемых файлов

Краткие сведения

Файловая оболочка NORTON COMMANDER

Пакет программ NC

Пакет программ Norton Commander (NC) относится к классу программ-оболочек.

Программа-оболочка — это программа, один из модулей которой, называемый резидентным, постоянно находится в оперативной памяти компьютера и для выполнения каких-либо заданных пользователем функций загружает с диска в свободные области памяти необходимые исполнительные модули.

Применение операционной оболочки NC значительно упрощает управление компьютером, так как процесс ввода команд и их параметров заменяется выбором из предлагаемого списка (меню) возможных значений или нажатием соответствующей функциональной клавиши.

Кроме того, при правильной работе с NC практически не требуется указывать пути к файлам и каталогам, что при работе в DOS часто сопровождается ошибками.

Пакет программ NC позволяет получить в наглядном виде информацию о компьютере, оперативной памяти и ее загрузке, изображать дерево каталогов диска (с возможностью перемещения по нему), осуществлять поиск файлов и каталогов, редактировать текстовые файлы, работать с архивами и т.д.

Основные принципы работы в NC

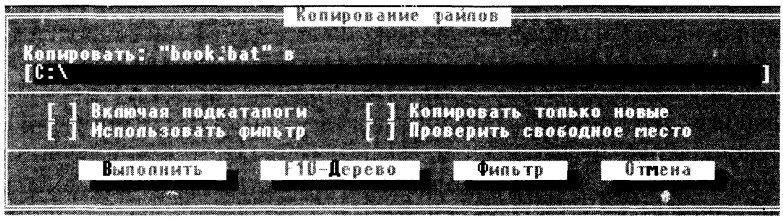
При работе с пакетом информация выводится на экран в окна двух типов: *информационные* и *диалоговые*.

Информационное окно — это окно, которое, как правило, занимает всю площадь экрана и предназначено главным образом для получения информации о различных компонентах вычислительной системы.

Диалоговые окна — это окна, которые предназначены для управления пакетом программ и ввода в них различной управляющей информации (например, окно копирования файлов, окно выбора текущего диска для панели и др.).

Управление компьютером в операционной оболочке NC осуществляется при помощи:

- функциональных клавиш **F1— F10**;
- «горячих» клавиш;
- управляющего меню;
- диалоговых окон;



- непосредственного ввода команд в командную строку;
- ручного манипулятора «мышь».

Получение помощи

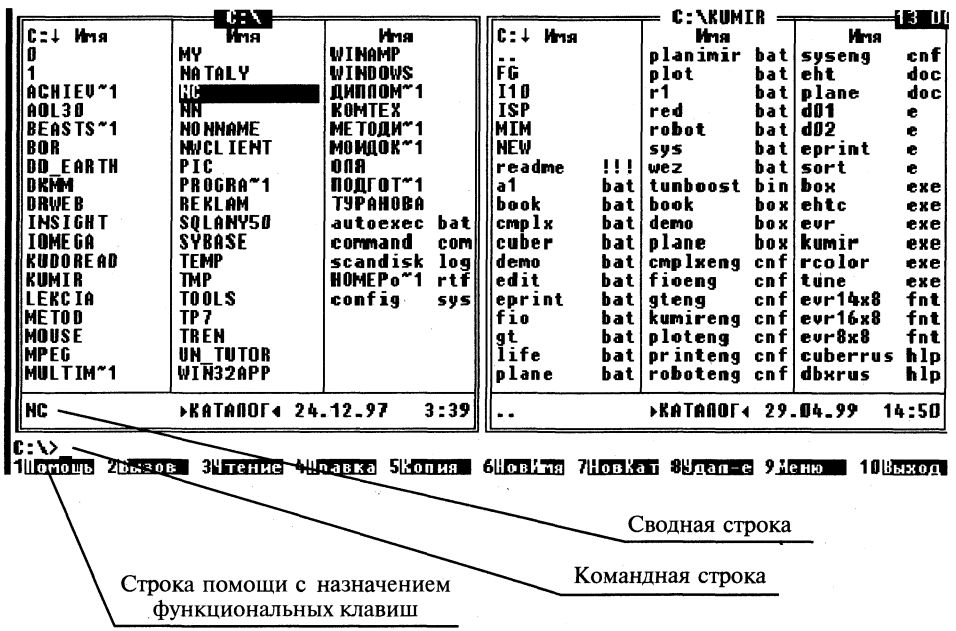
В любой момент работы с NC можно получить подсказку, нажав клавишу **F1**. На экране появляется оглавление. Чтобы вывести информацию по нужной теме, необходимо с помощью клавиш управления курсором выбрать необходимый пункт и нажать ENTER.

Выход из программы Norton Commander

Выход из NC осуществляется нажатием клавиши F10 и последующим подтверждением своего намерения.

Структура информационного окна

После запуска NC на экран выводится информационное окно с двумя панелями.



Одна панель является *активной*, другая *пассивной*. Курсор (полоска контрастного цвета) всегда находится в текущей (активной) панели. Переход с панели на панель осуществляется клавишей TAB. *Сводная строка* содержит информацию о текущем файле или каталоге в полном формате. При выделении одного или нескольких файлов в ней указывается число и суммарный объем выделенных файлов. В зависимости от настройки NC сводная строка может быть отключена.

Вид приглашения в *командной строке* может быть изменен пользователем, но всегда отражает путь к текущему каталогу и автоматически меняется при изменении диска или каталога.

Выбор диска для панели

Для вывода на панель информации того или иного диска необходимо нажать комбинацию клавиш:

Alt и F1 — для выбора диска, отражаемого в левой панели;

Alt и F2 — для выбора диска, отражаемого в правой панели.

В появившемся окне выбрать из списка доступных дисков нужный (с помощью клавиш управления курсором) и нажать Enter.

При указании имени дисковода, в котором не оказалось дискеты, на экране появится окно с сообщением об ошибке и предложением либо повторить операцию после установки диска, либо сменить неверно введенное имя.

Комбинация клавиш Ctrl и R позволяет вывести в текущую панель оглавление смененной в дисководе дискеты, Ctrl и O — погасить панели для просмотра информации.

Основные виды панелей

Панель	Вызов	Описание
Панель с содержимым каталога	Alt и F1 — выбор диска для левой панели Alt и F2 — выбор диска для правой панели	Выводится перечень имен файлов и каталогов. Для визуального отличия файлы обозначаются строчными латинскими буквами, каталоги — прописными. В разрыве верхней линии рамки выводятся имя каталога и путь к нему. Если текущий каталог не корневой, то в первой строке панели вы увидите две точки .. — ссылка на родительский каталог
Панель со сводной информацией	Ctrl и L	По выведенной на эту панель информации можно определить размер свободной оперативной и дисковой памяти, метку активного диска и прочитать краткий комментарий к активному каталогу, записанный в текстовый файл Dirinfo, если таковой имеется

Окончание таблицы

Панель	Вызов	Описание
Панель дерева каталогов	Alt и F10	В этой панели отображается дерево каталогов текущего диска, поэтому можно наглядно увидеть структуру взаимосвязанных каталогов. С помощью дерева каталогов легко выполняются операции перемещения в любой каталог с параллельным просмотром его содержимого, поиск нужного каталога
Панель быстрого просмотра	Ctrl и Q	Позволяет просматривать содержимое текстового файла, выделенного в смежной панели каталога, или получить краткую информацию о выделенном загрузочном модуле или каталоге (объем и число содержащихся в нем файлов и каталогов)
Панель просмотра каталогов	Ctrl и Z	Отображается следующая информация: число выделенных каталогов, включая подкаталоги; общее число файлов в каталогах; полный размер выделенного файла или общий размер всех файлов в каталогах; общее число дискет, необходимых для копирования всех выделенных файлов и каталогов

Работа с файлами, каталогами в операционной оболочке NC

Команда	Описание действий
Смена текущего каталога	Подвести курсор к имени нужного каталога и нажать Enter
Выход из текущего каталога	Установить курсор на две точки, расположенные в верхней части списка каталога, нажать клавишу Enter
Создание каталога	1. Нажать F7 2. В появившемся окне ввести имя каталога и нажать Enter
Выделение одного файла (каталога)	1. Установить курсор на строку с именем файла (каталога) 2. Нажать Insert
Выделение группы файлов (каталогов)	1. Последовательно устанавливать курсор на строки с соответствующими именами и каждый раз нажимать клавишу Insert 2. Ошибочное выделение можно отменить повторным нажатием клавиши Insert в соответствующей строке
Быстрое выделение	1. Нажать клавишу + , расположенную в зоне малой цифровой клавиатуры (называемую часто «серый плюс») 2. В диалоговом окне указать шаблон, идентифицирующий группу для выделения (используя групповые спецификации) 3. Нажать Enter 4. Клавиша − , расположенная в зоне малой цифровой клавиатуры («серый минус»), отменяет выделение
Копирование (перемещение) файлов и каталогов	1. Установить в одной панели исходный каталог, откуда предстоит копировать (перемещать) 2. Установить в другой панели каталог, куда необходимо копировать (перемещать) 3. Выделить в исходном каталоге файлы (каталоги), подлежащие копированию (перемещению) 4. Нажать F5 — для копирования (F6 — для перемещения), при этом исходная панель должна оставаться активной 5. Нажать Enter

Примечание. В верхней части диалогового окна копирования отображено имя копируемого файла (при копировании группы файлов указывается лишь число копируемых файлов) и путь к каталогу, в который должно произойти копирование.

При копировании каталога со всеми входящими в него файлами и каталогами необходимо в диалоговом окне установить флажок *Включая подкаталоги* и нажать клавишу ENTER.

Если при копировании файла в заданный каталог обнаруживается, что такой файл уже существует, на экране выводится диалоговое окно, окрашенное в красный цвет. Устанавливая курсор на один из предложенных режимов, можно осуществить замену файлов по одному, скопировать все файлы без предупреждения, пропустить копирование определенных файлов или отменить дальнейшее копирование.

Команда	Описание действий
Переименование файлов	1. Установить в одной панели исходный каталог, в котором находится файл, подлежащий переименованию 2. Установить в другой панели каталог, куда необходимо переместить переименованный файл 3. Выделить в исходном каталоге необходимый файл 4. Нажать F6, исходная панель должна оставаться активной 5. В диалоговом окне ввести новое имя файла 6. Нажать Enter
Удаление файла (каталога)	1. Выбрать удаляемый файл (файлы) или каталог (каталоги) 2. Нажать F8 3. В диалоговом окне установить флажок напротив опции *Удаление подкаталогов* (для разрешения одновременного удаления файлов и каталогов), нажать Enter 4. Для подтверждения удаления снова нажать Enter
Создание текстового файла	1. Нажать Shift и F4 2. В появившемся окне ввести имя создаваемого файла с расширением .txt и нажать клавишу Enter 3. В открывшемся встроенном редакторе текстов NC создать текстовый документ 4. Нажать F10 для выхода из редактора. Если до этого документ не был сохранен, то появится диалоговое окно, в котором необходимо сделать выбор: *Выйти с сохранением созданного файла*, *Выйти без сохранения*, *Отменить выход*
Редактирование текстового файла	1. Установить курсор на имени файла 2. Нажать F4 3. Внести необходимые изменения в текст 4. Выйти из режима редактирования — нажать F10
Просмотр текстового файла	1. Установить курсор на имени файла 2. Нажать F3 *Примечание.* Если подключена соответствующая программа, то можно просмотреть и графический файл
Правая (Левая)\ Имя (Тип, Время, Размер)	Указывают порядок сортировки файлов
Команды\ Конфигурация	Устанавливает режимы настройки программы NC

Работа с пакетом Norton Commander при помощи управляющего меню

Активизация управляющего меню осуществляется нажатием клавиши F9. Управляющее меню расположено в верхней строке экрана и содержит следующие пункты:

Левая Файл Диск Команды Правая

Перемещая выделение с помощью горизонтальных клавиш управления курсором, можно выбрать требуемый режим и нажатием клавиши Enter раскрыть вертикальное подменю, содержащее перечень команд.

Клавиша Esc служит для выхода из режима работы с управляющим меню.

Изменение вида экрана NC с помощью команд управляющего меню

Команда	Описание действий
Правая (Левая)\ Краткий формат	Устанавливает режим, при котором в панели отображаются только имена файлов
Правая (Левая)\ Полный формат	Для каждого файла указываются его характеристики: размер в байтах, дата и время его создания или последней модификации
Правая (Левая)\ Имя (Тип, Время, Размер)	Указывают порядок сортировки файлов
Команды\ Конфигурация	Устанавливает режимы настройки программы NC

Краткие сведения Windows'95 (98)

Общая характеристика

Windows'95 (98) является высокопроизводительной, универсальной, надежной, многозадачной и многопотоковой интегрированной 32-разрядной операционной системой нового поколения с расширенными сетевыми возможностями, работающей в защищенном режиме (integrated 32-bit protected-mode operating system) и обеспечивающей графический интерфейс с пользователем. Windows'95 (98) представляет собой интегрированную среду, обеспечивающую эффективный обмен информацией между отдельными программами и предоставляющую пользователю широкие возможности по обработке текстовой, графической, звуковой и видеоинформации. Понятие интегрированности подразумевает также совместное использование ресурсов компьютера всеми программами.

Windows'95 (98) обеспечивает работу пользователя в сети, с электронной почтой, с факсом и со средствами мультимедиа, поддерживает большинство приложений DOS и предыдущих версий Windows.

После загрузки Windows'95 (98) на экране появляется изображение, напоминающее рабочий стол (desktop). Так же, как на рабочем столе, на его модели (на экране) размещены значки папок с документами и значки быстрого доступа.

Значительное внимание уделено документо-ориентированной работе с тем, чтобы пользователь в первую очередь уделял внимание документам, а не прикладным программам (документом называется любой файл, обрабатываемый с помощью прикладной программы).

Windows'95 (98) позволяет давать файлам имена, содержащие до 255 символов и включать пробелы, знак плюс, знак равенства, квадратные скобки, точку с запятой и другие знаки препинания. Пробелы, находящиеся в начале и в конце имени, не учитываются. Имя файла можно писать на русском языке. Любые симво-

лы, стоящие после последней точки, рассматриваются как расширение. Расширение имени зависит от приложения, в котором создавался файл. Имя для папки задается так же, как для файла. Однако для папки не задается расширение. Информация о длинных именах файлов в новой операционной системе хранится в виртуальной таблице размещения файлов (Virtual File Allocation Table — VFAT).

Перед тем как выключать питание компьютера, необходимо закрыть все открытые документы и приложения. Выключение питания без закрытия документа может привести к потере данных, повреждению открытых файлов и к трудностям с их открытием при последующих сеансах работы.

После выключения компьютера без правильного выхода из системы возможны нарушения в логической структуре диска. Их исправить можно с помощью программы ScanDisk, расположенной в группе Служебные программы (Accessories).

Для корректного выхода из Windows надо щелкнуть кнопку *Пуск* и команду *Завершить работу* (Shut Down) в появившемся меню. Появится диалоговое окно *Завершение работы* (Shut Down Windows).

Окно содержит четыре кнопки-переключателя: *Выключить компьютер*, *Перезагрузить компьютер*, *Перезагрузить компьютер в режиме эмуляции MS DOS*, *Войти в систему под другим именем*. Все кнопки закрывают все программы. В нижней части окна Shut Down Windows расположены три кнопки *Да*, *Нет*, *Справка*.

Через небольшой промежуток времени после щелчка мышью кнопки *Да* компьютер будет подготовлен к выключению: будут очищены внутренние буферы и кэши дисков, обеспечено сохранение данных. Не следует выключать электропитание до тех пор, пока не появится сообщение: «Теперь питание компьютера можно выключить».

Запуск программ, загрузка документов

Windows'95 (98) предоставляет удобные средства быстрого вызова программ, документов и папок с помощью значков быстрого вызова (shortcut), позволяющих двойным щелчком открыть папку или документ, не запуская предварительно приложение, в котором создавался объект.

Объекты (предметы), с которыми мы контактируем в реальной жизни, обладают определенными свойствами. У каждого предмета свой внешний вид, вес, габариты и т. п. Аналогично объекты Windows имеют свои характеристики (Properties). Можно подобрать внешний вид значка, отображающего файл. Файлы имеют размеры, для них задаются атрибуты и т. п.

Чтобы ускорить открытие часто используемых документов и запуск приложений, можно создать для каждого документа свой значок быстрого вызова и разместить их на рабочем столе в одной или нескольких папках. Можно обеспечить быстрый вызов принтера, установив его значок на рабочей поверхности стола. В этом случае, чтобы распечатать файл, будет достаточно перетащить мышью его значок на значок принтера. Двойной щелчок значка быстрого вызова *Блокнот* (WordPad) на экране дисплея запустит текстовый процессор.

Чтобы установить значок быстрого вызова к папке (файлу), ее необходимо выделить в окне *Мой компьютер* (My Computer) и выбрать команду *Создание ярлыка* (Create Shortcut) из меню *Файл* (File).

Первоначально значок располагается в конце списка окна. Значок можно переместить или скопировать на рабочий стол или в часто используемую папку с помощью мыши методом Drag and Drop (Перетащить и отпустить). Другой вариант

установки значка быстрого вызова в нужной папке или на рабочем столе — перетащить значок программы или документа в нужную папку, нажав правую кнопку мыши, и воспользоваться командой *Создать ярлык* (Create Shortcut(s) Here) из динамического меню, которое появится, когда отпустят правую кнопку. Можно не копировать файл (папку) в другую папку, а вставить в нее значок быстрого вызова к этому файлу. Сначала следует выделить файл, затем активизировать команду *Копировать* (Copy) из меню *Правка* (Edit). После перехода в окно, где предполагается разместить значок быстрого вызова, активизируют команду *Вставить ярлык* (Paste Shortcut) из меню *Правка* (Edit).

Изображение значка быстрого вызова можно изменить с помощью диалогового окна, появляющегося после выделения значка и активизации команды *Свойства* (Properties) из меню *Файл* (File).

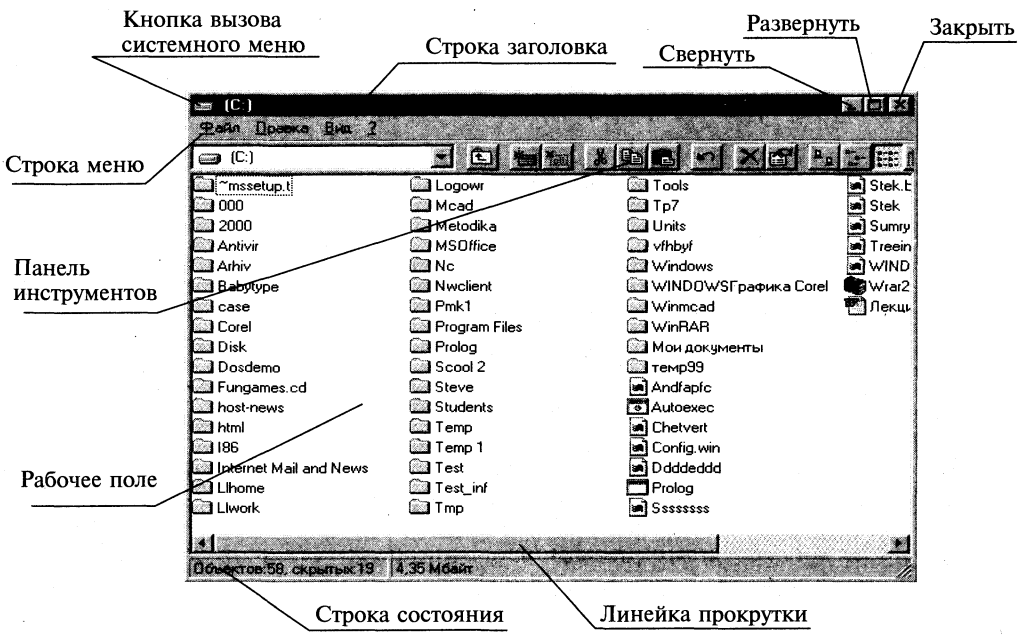
Все значки быстрого вызова связаны с файлами и папками, которые они представляют. При удалении файлов и папок автоматически удаляются и значки. При удалении значка быстрого вызова файл, с которым он связан, не удаляется. Если переименовать папку или файл, надпись к значку не изменится, однако связь между ними сохранится.

Виды и свойства окон

Существенно усовершенствованы в Windows'95 (98) окна. В дополнение к панели команд каждое окно получило панель инструментов.

Панель инструментов можно отобразить в окне папки или удалить командой *Панель инструментов* (Toolbar) из меню *Вид* (View).

Панель содержит раскрывающееся окно списка, в котором представлено имя текущей папки, и кнопки, дублирующие часто используемые команды. После раскрытия окна списка в нем видна древовидная диаграмма папок, имеющихся на компьютере.



Команду можно быстро активизировать, щелкнув кнопку на панели инструментов. При этом нет необходимости сначала выбирать меню. При подводе указателя мыши к кнопке рядом с ним появляется флажок — подсказка с пояснением выполняемой команды.

В зависимости от указанного типа файла Windows'95 (98) позволяет открыть то или иное приложение, использовать определенный набор команд. Для изменения типа файла или значка к нему используют кнопку *Правка* (Edit).

При работе с документом следует максимально увеличить область экрана, где может располагаться интересующая нас информация. Однако большую часть экрана часто занимают различные панели.

Отображение на экране панели задач (taskbar) регулируется с помощью флажков вкладки *Параметры панели задач* (Taskbar Options) команды *Панель задач* (Taskbar) меню *Настройки* (Settings).

Элементы окна

Системное меню — содержит команды для изменения размеров окна, его перемещения, минимизации до размеров значка и закрытия.

Строка состояния — содержит разнообразную справочную информацию, зависящую от текущего режима работы.

Строка горизонтального меню — содержит группы команд, объединенных по функциональным признакам. Вызов справочной информации осуществляется с помощью кнопки «?».

Некоторые кнопки панели инструментов окна:


Вырезать в буфер


Копировать в буфер


Вставить из буфера


Переход на уровень вверх


Удаление объекта


Отмена последнего действия


Кнопки настройки отображения объектов в окне.

Изменение размеров окна

1. При помощи кнопок, расположенных в строке заголовка окна:
   - кнопка *Свернуть* сворачивает окно в кнопку и помещает ее на панели задач. Чтобы снова развернуть окно, необходимо щелкнуть по данной кнопке;
   - кнопка *Развернуть* увеличивает размер окна до максимально возможного;
   - кнопка *Восстановить* изменяет размер окна от максимального до среднего и обратно.

2. При помощи мыши:
   - подвести указатель мыши к границе окна (вертикальной, горизонтальной, углу), которую необходимо изменить (указатель примет вид двойной стрелки);
   - при нажатой левой клавише мыши переместить указатель в нужном направлении.

Способы закрытия окна

1. Комбинация клавиш Alt + F4.
2. Щелчок по кнопке *Закрыть* строки заголовка окна.
3. Активизировать системное меню и выбрать команду *Закрыть*. Выполнить команду *Файл/Закрыть*.

Настройки системы

Они сосредоточены в папке *Панель управления* команды *Настройка главного меню системы*. Ниже приведены некоторые иконки настроек с пояснением их смысла.

Дата и время

Позволяет изменить дату, а также время, отображаемое на *Панели задач*.

Мышь

Позволяет настроить мышь для работы левши, подключить новые указатели мыши, а также изменить скорость перемещения указателя.

Система

Позволяет просмотреть характеристики компьютера (марку процессора, объем оперативной памяти), список установленных устройств, состояние системных ресурсов.

Экран

Позволяет разместить на *Рабочем столе* рисунок, изменить цветовое оформление Windows, разрешение экрана, настроить программу-заставку, а в Windows'98 включить некоторые эффекты отображения информации.

Установка и удаление программ

Вызвав диалоговое окно установки и удаления программ, можно увидеть все программы, успешно установленные в Windows. Если программа в списке отсутствует — значит, установка прошла некорректно. Назначение этого объекта — установка новой программы или удаление уже установленной. Объект позволяет также добавить или удалить какой-либо компонент Windows (например, *Игры*) и создать загрузочный диск.

Установка оборудования

При установке нового оборудования (например, *Сетевой карты*) система должна подобрать для него необходимые драйверы, и если возникают проблемы с определением устройства, можно с помощью *Мастера установки оборудования* или вручную установить драйверы.

Клавиатура

Позволяет изменить сочетание клавиш для переключения раскладок клавиатуры с русской на английскую, а также включить или отключить отображение индикатора языка (Ru/En) на панели задач.

Принтеры

Чтобы выводить информацию на печать, необходимо предварительно установить принтер, используемый по умолчанию в Windows. Для этого в окне папки *Принтеры* расположен специальный значок — *Установка принтера*. Запустив его и отвечая на вопросы мастера установки, можно легко установить принтер. Такие программы, как Excel или Access, не могут сформировать окно предварительного просмотра или отчет, если в системе не установлен принтер.

Для выделения заголовков, смыслового разграничения отдельных фрагментов, при написании формул, индексов используют различные стили и размеры шрифта.

Получить справочные данные о шрифте и увидеть его гарнитуру (стиль) позволяет окно, появляющееся после выбора команды *Открыть* (Open) из меню *Файл* (File) папки *Шрифты* (Fonts) программы *Мой компьютер* (My Computer).

В окне отражаются название (Typeface name), размер файла (File size), версия (Version), фирма-разработчик, демонстрируются образцы нескольких размеров шрифта. Чтобы удалить используемый шрифт, его надо выделить в окне папки *Шрифты* (Fonts) и выбрать команду *Удалить* (Delete) из меню File.

Во многих приложениях гарнитура и размер шрифта задаются в диалоговом окне команды *Шрифт* (Font).

Перечень всех установленных шрифтов приводится в окне списка *Шрифт*. Размер в пунктах выбирается в окне *Размер* (Size). Образец написания выбранного шрифта представляется в демонстрационном поле *Образец* (Sample) в правой части окна. В поле *Цвет* (Color) задается цвет символов.

Начертание (стиль) шрифта задается в окне *Начертание* (Font style): обычный, полужирный, курсив, подчеркнутый.

Меню *Пуск* (Start) позволяет выполнить большой набор работ, связанных с запуском приложений, получением справок, поиском и открытием документов, настройкой системы. Видимо, поэтому разработчики операционной системы рядом с кнопкой *Пуск* поместили подсказку: «Начните работу с нажатия этой кнопки».

Для активизации меню *Пуск* следует щелкнуть кнопку *Пуск* или нажать на клавиши Ctrl+Esc. При остановке указателя мыши на пункте меню со значком треугольника с правой стороны раскрываются окна, содержащие *Подменю* (submenu) и команды. В свою очередь, отдельные пункты появившегося подменю также могут быть отмечены значком треугольника и иметь свои подменю. Каждое подменю содержит группу программ. Для выбора программы необходимо остановить на ней указатель и щелкнуть мышью.

Пункт *Документы* (Document) открывает список с названиями последних документов, с которыми работал пользователь. Список может содержать до 15 наименований документов, независимо от приложения, в котором они создавались. Для открытия документа следует щелкнуть на его названии.

Следует отметить, что некоторые приложения не добавляют имена файлов в список меню Document. Тогда документ можно открыть, запустив то приложение,

в котором он создавался. Как правило, открыть документ позволяет команда *Открыть* (Open) из меню *Файл* (File) соответствующего приложения Windows. Кроме того, документ можно открыть двойным щелчком его значка в окне *Мой компьютер* (My Computer).

Ряд приложений Windows проверяет, имеются ли в памяти компьютера несохраненные данные. При выходе из приложения без команды *Save* (Сохранить) появится предупреждающее сообщение с вопросом о необходимости сохранения последних изменений. При попытке закрыть приложение без указаний, как поступить с открытым документом, появится запрос: «Сохранить изменения, внесенные в документ?» Три кнопки (*Да*, *Нет*, *Отменить*) позволяют сохранить внесенные во время текущего сеанса работы изменения, не вносить изменений или отменить выход из системы.

Наиболее быстрый способ добавить команду/пункт в меню *Пуск* — перетащить мышью значок программы на кнопку меню Start. Новый пункт меню расположится в верхней строке меню. Например, можно создать значок быстрого вызова (Shortcuts) для программы Norton Commander, разместить сначала на рабочем столе (desktop) значок программы Norton Commander, а затем перетащить его мышью на кнопку *Пуск* (Start).

Нередко приходится искать нужный файл/папку, так как забыто его имя или место расположения. Если известна папка, где расположен файл, то можно легко найти его по расширению. Сложнее, если не известно и название, и расширение.

Можно использовать команду *Файлы или Папки* (Files or Folders) из меню *Поиск* (Find), чтобы быстро найти файл или папку на компьютере пользователя или на других компьютерах сети.

Поиск можно выполнять по следующим критериям:

- по имени файла или папки и по цепочке символов, входящих в имя файла или папки;
- по расширению имени файла;
- по дате последней модификации;
- по размеру;
- по отрывку текста из документа или по заголовку какого-либо раздела.

Использование различных вкладок облегчает поиск файлов по определенным критериям.

Для быстрого вызова часто повторяемых команд можно воспользоваться динамическим меню, которое вызывается правой кнопкой мыши.

Динамическое меню содержит часто употребляемые команды. Набор команд зависит от выбранного объекта: значка диска, папки или файла, выделенного текста, панели задач или свободного места экрана. Например, щелчок правой кнопки свободной области рабочего стола вызывает меню, содержащее следующие команды, позволяющие изменить расположение значков на столе или его внешний вид: *Упорядочить значки* (Arrange icons), *Выстроить значки* (Line up icons), *Обновить* (Refresh), *Вставить* (Paste), *Вставить ярлык* (Paste Shortcut), *Создать* (Create), *Свойства* (Properties).

Если щелкнуть правой кнопкой значок диска в окне программы *Мой компьютер* (My Computer), то появятся команды: *Открыть* (Open), *Проводник* (Explore), *Найти* (Find), *Разделение* (Sharing), *Форматировать* (Format), *Вставить* (Paste), *Создать ярлык* (Create Shortcut), *Свойства* (Properties). Так же, как и для диска, команды динамического меню для файла зависят от типа файла и дублируют меню *Файл*.

Правую кнопку мыши удобно использовать для перемещения или копирования файла из одного окна в другое папки *Мой компьютер* или *Проводник*.

После транспортировки значка папки/файла с нажатой правой кнопкой мыши появляется динамическое меню, позволяющее указать цель транспортировки: переместить или скопировать объект, создать значок быстрого вызова.

Папка *Мусорная Корзина* (Recycle Bin) предназначена для удаления ненужных файлов. Основное отличие между выполнением команды *Удалить* (Delete) в Windows'95 (98) и в других программах состоит в том, что в новой версии операционной системы выбор команды приводит не к удалению файла, а к его перемещению в папку *Мусорная Корзина* (Recycle Bin). Файл, попавший в Мусорную Корзину, сохраняется до тех пор, пока Корзина не будет «очищена».

Чтобы удалить любой файл, папку или значок быстрого вызова, можно использовать команду *Удалить* или переместить значок удаляемого объекта мышью на значок *Мусорной Корзины*. Значок перетаскиваемого объекта исчезнет.

Чтобы просмотреть все файлы, находящиеся в *Мусорной Корзине*, необходимо дважды щелкнуть ее значок. Появится окно папки. Строка меню имеет стандартный набор команд.

Для восстановления файла, папки или значка быстрого вызова необходимо щелкнуть имя восстанавливаемого файла. Если надо восстановить несколько файлов, то имена файлов выделяют при нажатой клавише Ctrl. Затем используют команду *Восстановить* (Restore) из меню Файл.

В настоящее время для IBM PC-совместимых компьютеров выпускаются тысячи наименований винчестеров, адаптеров, контроллеров и других изделий. В ряде случаев их установка на компьютере вызывает значительные трудности с точки зрения совместимости, требует больших затрат времени пользователя для выбора положения перемычек (switches).

При инсталляции Windows'95 (98) приложение *Установка* (Setup) определяет адаптеры и драйверы, не поддерживающие новую технологию, и автоматически делает соответствующие записи в системных файлах. Даже если ваш компьютер не полностью поддерживает стандарт Plug and Play, Windows'95 (98) поможет настроить аппаратное обеспечение с помощью диалогового окна Properties с вкладкой Recources рассматриваемого устройства. Для вызова окна устройства сначала следует активизировать значок *System* (Система) *Панели Управления* и двойным щелчком мыши выбрать устройство.

Контрольные вопросы

1. Что называется операционной системой?
2. Каковы компоненты операционной системы MS DOS? Охарактеризуйте назначение каждой из них.
3. Что такое файл? каталог? логический диск? Как они именуются?
4. Какова последовательность операций начальной загрузки системы?
5. Перечислите внутренние команды операционной системы. Приведите примеры их использования.
6. Приведите примеры использования внешних команд ОС.
7. Поясните назначение и приведите примеры файлов config.sys и autoexec.bat.
8. Каково назначение файловых оболочек типа NORTON COMMANDER?
9. Какова структура панелей NORTON COMMANDER?
10. Каковы основные возможности NORTON COMMANDER?

11. Каковы основные отличия операционной системы WINDOWS'95 (98) от MS DOS?
12. Какие опции содержит главное меню WINDOWS'95 (98)?
13. Как производится запуск программ в WINDOWS'95 (98)?
14. Какова структура и свойства окон WINDOWS'95 (98)?
15. Как производится настройка WINDOWS'95 (98)?
16. Что означает «документо-ориентированный»?

Темы для рефератов

1. Эволюция операционных систем компьютеров различных типов.
2. Возникновение и возможности первых операционных систем для персональных компьютеров.
3. Внешние команды MS DOS.
4. История развития операционной системы WINDOWS.
5. Сравнительный анализ операционных систем WINDOWS и MAC OS.
6. Особенности операционной системы WINDOWS NT WORKSTATION.
7. Перспективы развития операционной системы WINDOWS.
8. Особенности и возможности файловых оболочек типа VOLKOV COMMANDER, DOS NAVIGATOR, FAR, DISC COMMANDER и т.п.
9. Утилиты NORTON UTILITS и подобные.

Темы семинарских занятий

1. Понятие и эволюция операционных систем компьютеров. Понятие многопользовательской и многозадачной операционной системы.
2. Развитие операционных систем персональных компьютеров.
3. Основные команды MS DOS и WINDOWS'95(98).
4. Действия при первичной установке WINDOWS'95(98). Инсталляция новых устройств персонального компьютера.

Рекомендации по программному обеспечению

Практические работы по операционным системам могут быть выполнены в учебном классе на основе компьютеров PC Pentium (и выше), поддерживающих ОС WINDOWS'95(98), имеющих дисковод хотя бы для дискет (A:). Ознакомление с командами MS DOS может быть проведено в режиме эмуляции DOS. Также требуется файловый менеджер NORTON COMMANDER или FAR, установленный на компьютерах. Другого программного обеспечения не требуется.

При ознакомлении с настройками WINDOWS появится необходимость восстановления состояния операционных систем на компьютерах. Использование в учебном классе Windows NT с ограничивающими права пользователей профилями с точки зрения целей обучения является в данном случае неподходящим. Для облегчения восстановления систем можно рекомендовать использование образов систем при помощи локальной сети компьютерного класса.

Первичную установку ОС, форматирование дисков лучше продемонстрировать на отдельном, специально выделенном для этого компьютере.

Задачи и упражнения

Задания 1—22 выполняются с помощью команд MS DOS. Для выполнения задания потребуется гибкий диск.

1. Загрузите компьютер в режиме MS DOS. Определите версию ОС.
2. Измените дату на 08.03.2002.
3. Просмотрите установленное время.
4. Зайдите в каталог C:\WINDOWS\COMMAND.
5. Запишите, сколько файлов содержится в каталоге COMMAND.
6. Создайте системную дискету.
7. Перейдите на диск **A:**
8. Создайте на диске **A:** каталог PROBA. В каталоге PROBA создайте подкаталог TEXT.
9. Скопируйте в **A**:\PROBA следующие файлы из каталога C:\NC nc.exe, nc.mnu, nc.ico.
10. Переименуйте файл nc.mnu в nc.txt.
11. Просмотрите содержимое файла nc.txt.
12. Скопируйте файл nc.txt в каталог **A**:\PROBA\TEXT.
13. Удалите файл nc.txt из каталога **A**:\PROBA.
14. Перейдите в C:\NC.
15. Запустите NC.
16. Выйдите из NC.
17. Удалите каталог TEXT на диске **A**:.
18. Отформатируйте дискету.
19. Установите текущую дату.
20. Осуществите «быстрое» форматирование **ДИСК A**:, проверку испорченных областей производить не надо.
21. Осуществите вывод на экран списка программ, расположенных в оперативной памяти компьютера, указывая их местоположение и размер. При выводе информации должна выполняться остановка после заполнения экрана.
22. Просмотрите и прокомментируйте основные команды файлов autoexec.bat и config.sys.

Задания 23—35 выполняются с помощью файловой оболочки NC или FAR.

23. Установите на правой панели полный режим отображения содержимого диска C:, упорядочите информацию по имени.
24. Определите объем свободного места на диске **C**:.
25. Просмотрите дерево каталогов диска **C**:.
26. Установите на левой панели краткий режим отображения содержимого диска **C:**, упорядочите информацию по размеру файлов.
27. Создайте на диске **C**: каталог PROBA.
28. Скопируйте в каталог PROBA файлы каталога STUDENT с расширением PAS и файл корневого каталога command.com.
29. Найдите файл autoexec.bat, просмотрите его содержимое.
30. Создайте в каталоге PROBA текстовый файл my.txt, содержащий информацию о вашей группе.
31. Внесите изменения в файл MY.TXT. Сохраните их.
32. Создайте локальное пользовательское меню для каталога PROBA, включив пункт вызова тестирующей программы DRWEB.
33. Используя созданное меню, запустите программу DRWEB.EXE.

34. Удалите из каталога PROBA все файлы с расширением PAS.
35. Удалите каталог PROBA.

Задания 36—54 выполняются с помощью средств Windows'95 (98).

36. Откройте окно *Мой компьютер*, измените его размеры и положение, минимизируйте, восстановите, затем закройте.
37. С помощью меню пуска запустите приложение *Блокнот*, напишите в нем «Это проверка» и сохраните этот текст в виде файла с названием «Это проверка» в папке *Мои документы*.
38. С помощью приложения *Paint* выполните рисунок на тему текущего времени года и сохраните его в папке *Мои документы*.
39. С помощью приложения *Калькулятор* вычислите 1234·5678.
40. Выполните поиск файла «Это проверка.txt».
41. Открыв папку *Мои документы* в окне *Мой компьютер*, переименуйте файл «Это проверка.txt» в text.txt.
42. С помощью *Мой компьютер* скопируйте файл text.txt методом перетаскивания на дискету A:.
43. Удалите созданные в ходе выполнения заданий файлы в папке *Мои документы*.
44. Измените текущую дату на 8.03.2002.
45. Измените клавиши переключения клавиатуры с английского алфавита на русский.
46. Измените настройку экрана: размер, цвет рабочего стола. Выберите наиболее понравившуюся цветовую схему.
47. Восстановите состояние системы до изменений.
48. Проверьте состояние системы, наличие установленных сетевых компонент.
49. Установите, а затем удалите в системе новое устройство — принтер, модем, CDROM или что-нибудь еще.
50. Установите, а затем удалите в системе новую программу (по выбору преподавателя).
51. Проверьте, какие компоненты WINDOWS установлены на компьютере.
52. Создайте загрузочный диск.
53. Очистите меню *Документы*.
54. Удалите, а затем добавьте в меню запуска какой-либо пункт.

Лабораторные работы

По операционным системам рекомендуется провести не менее трех лабораторных работ по следующим темам:
1. MS DOS (время выполнения работы 4—6 часов).
2. Файловые менеджеры (оболочки) (время выполнения работы 4—6 часов).
3. Windows'95 (98) (время выполнения работы 6—8 часов).

В ходе этих работ должны быть выполнены задания, приведенные выше.

Дополнительная литература

1. *Ахметов К.* Курс молодого бойца. Т. 1, 2. 5-е изд. — М.: Компьютер Пресс, 1998.
2. *Ахметов К.* Практический курс молодого бойца. — М.: Компьютер Пресс, 1999.

3. *Богумирский Б.* Эффективная работа на IBM PC в среде Windows'95. — СПб.: Питер, 1997.
4. *Дженнингс Р.* Windows'95 в подлиннике. — СПб.: BHV — Санкт-Петербург, 1995.
5. *Джордейн Р.* Справочник программиста персональных компьютеров типа IBM PC, XT и AT: Пер. с англ. — М.: Финансы и статистика, 1992.
6. *Кинг А.* Windows'95 изнутри: Пер. с англ. — СПб.: Питер, 1995.
7. *Крол Эд.* Все об INTERNET / Пер. с англ. С. М. Тимачева. — Киев: BHV, 1995.
8. *Лядова Л. Н.* Персональный компьютер: От начинающего пользователя до профессионала. Т. 1, 2. — Пермь: ПГУ, 1998.
9. Microsoft Windows'95. Шаг за шагом. — М.: ЭКОМ, 1996.
10. *Мюллер С.* Модернизация и ремонт персональных компьютеров: Пер. с англ. — М.; СПб.; Киев, 2000.
11. Операционная система MS DOS. — М.: Радио и связь, 1992.
12. Ресурсы Microsoft Windows'95: В 2 т. — М.: ТОО «Channel Trading Ltd.», 1996.
13. *Соловьев Г. Н., Никитин В. Д.* Операционные системы ЭВМ. — М.: Высш. шк., 1989.
14. *Фигурнов В. Э.* IBM PC для пользователя. 6-е изд. — М.: ИНФРА, 1995.
15. *Фойц С.* Windows 3.1: Пер. с нем. — Киев: BHV, 1993.
16. *Шиб Й.* Windows: секреты и советы: Пер. с нем. — М.: БИНОМ, 1996.
17. *Эдингтон Б.* (и группа разработчиков Microsoft Windows'95). Знакомство с Microsoft Windows'95: Пер. с англ. М.: ТОО «Channel Trading Ltd.», 1995.

§ 2. ПОНЯТИЕ О СИСТЕМЕ ПРОГРАММИРОВАНИЯ

Рекомендации по проведению занятий

Учитывая обзорный характер данной темы и то, что многие вопросы, затронутые в ней, являются впоследствии объектами отдельного изучения, целесообразно в общей части курса ограничиться написанием рефератов и проведением семинарского занятия.

Темы для рефератов

1. История языков программирования.
2. Язык компьютера и человека.
3. Объектно-ориентированное программирование.
4. Непроцедурные системы программирования.
5. Искусственный интеллект и логическое программирование.
6. Языки манипулирования данными в реляционных моделях.
7. Макропрограммирование в среде Microsoft OFFICE.
8. «Визуальное» программирование. VISUAL BASIC, C, PROLOG.
9. Все о DELPHI.
10. Программирование на HTML, JAVA.
11. Издательская система TeX как система программирования.
12. Современные парадигмы программирования. Что дальше?

13. Никлаус Вирт. Структурное программирование. Pascal и Modula.
14. Что мы знаем о Fortran?
15. История языка Бейсик.
16. Язык Ассемблера.
17. Алгоритмический язык Ершова.
18. Все о Logo-мирах.
19. История программирования в лицах.
20. Язык программирования ADA.
21. Язык программирования PL/1.
22. Язык программирования Algol.
23. Язык программирования Си.
24. О фирмах-разработчиках систем программирования.
25. Языки программирования в СУБД.
26. О системах программирования для учебных целей.

Тема семинарских занятий

Трансляция программ и сопутствующие процессы.

Дополнительная литература

1. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции: Пер. с англ. Т. 1. — М.: Мир, 1978.

2. *Брой М.* Информатика: В 3 т. Т. 3. Структуры систем и системное программирование: Пер. с нем. — М.: Диалог-МИФИ, 1996.

3. *Брукс Ф.П.* Как проектируются и создаются программные комплексы: Очерки по системному программированию: Пер. с англ. — М.: Наука, 1979.

4. *Дайтибегов Д.М., Черноусов Е.А.* Основы алгоритмизации и алгоритмические языки. — М.: Финансы и статистика, 1992.

5. *Доорс Дж., Рейблен А., Вадера С.* Пролог — язык программирования будущего: Пер. с англ. — М.: Финансы и статистика, 1990.

6. *Илюшечкин В.М., Костин А.Е.* Системное программное обеспечение. — М.: Высш. шк., 1991.

7. Логический подход к искусственному интеллекту: От классической логики к логическому программированию: Пер. с фр. / А.Тей, П.Гриболон, Ж.Луи и др. — М.: Мир, 1990.

8. *Львовский С.М.* Набор и верстка в пакете LaTeX. — М.: Космосинформ, 1994.

9. Программное обеспечение ЭВМ: В 11 кн. / Под ред. В.Ф.Шаньгина. — М.: Высш. шк., 1987. Кн. 1. *Костин А.Е.* Структура и функционирование ЭВМ. Кн. 2. *Илюшечкин В.М.* и др. Системное программное обеспечение.

10. *Уэйт М., Прата С., Мартин Д.* Язык Си: Пер. с англ. — М.: Мир, 1988.

11. *Федоров А.* Создание Windows-приложений в среде Delphi. — М.: Компьютер Пресс, 1995.

12. Электронные вычислительные машины: В 8 кн. / Под ред. А.Я.Савельева. — М.: Высш. шк., 1993. Кн. 4, 5 «Языки программирования».

13. Языки программирования Ада, Си, Паскаль. Сравнение и оценка: Пер. с англ. / Под ред. А.Фьюэра и Н.Джехани. — М.: Радио и связь, 1989.

§ 3. ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ОБЩЕГО НАЗНАЧЕНИЯ

Рекомендации по проведению занятий

Учитывая обзорный характер данной темы и то, что многие вопросы, затронутые в ней, являются впоследствии объектами отдельного изучения, целесообразно ограничиться написанием рефератов и проведением семинарского занятия.

Темы для рефератов

1. Программные системы обработки текстов под MS DOS.
2. Программные системы обработки текстов под WINDOWS.
3. Электронные таблицы под MS DOS.
4. Электронные таблицы под WINDOWS.
5. Программные системы обработки графической информации под MS DOS.
6. Программные системы обработки графической информации под WINDOWS.
7. Современная компьютерная графика. CorelDraw и Photoshop.
8. Компьютерная анимация. 3D Max и другие.
9. Программные системы обработки сканированной информации.
10. Программные системы «переводчики».
11. Мультимедиасистемы. Компьютер и музыка.
12. Мультимедиасистемы. Компьютер и видео.
13. Обзор компьютерных игр.
14. Системы управления базами данных под MS DOS и WINDOWS.
15. Системы управления распределенными базами данных. ORACLE и другие.
16. Обучающие системы. Средства создания электронных учебников.
17. Обучающие системы. Средства создания систем диагностики и контроля знаний.
18. Сетевые и телекоммуникационные сервисные программы.
19. О программах-поисковиках в Интернете.
20. О программах-броузерах в Интернете.
21. Системы компьютерной алгебры.
22. Пакет MathCad.
23. Развитие программных средств математических вычислений — от Eureka до Mathematica.

Темы семинарских занятий

1. Классификация и назначение прикладных программных средств общего назначения.
2. Организация «меню» в программных системах.

Дополнительная литература

1. *Богумирский Б. С.* Эффективная работа на IBM PC в среде Windows'95. — СПб.: Питер-Пресс, 1997.

2. *Брукс Ф. П.* Как создаются и проектируются программные комплексы: Пер. с англ. — М.: Наука, 1979.
3. *Брябрин В. М.* Программное обеспечение персональных ЭВМ. — М.: Наука, 1990.
4. *Кирмайер М.* Мультимедиа: Пер. с нем. — СПб.: BVH — Санкт-Петербург, 1994.
5. *Климов Д. М., Руденко В. М.* Методы компьютерной алгебры в задачах механики. — М.: Наука, 1989.
6. *Липаев В. В.* Проектирование программных средств. — М.: Высш. шк., 1990.
7. *Майерс Г.* Надежность программного обеспечения: Пер. с англ. — М.: Мир, 1980.
8. Проектирование пользовательского интерфейса на персональных компьютерах. — Вильнюс: DBS Ltd., 1992.
9. *Скляров В. А.* Программное и лингвистическое обеспечение персональных ЭВМ. Системы общего назначения. — Минск: Вышэйш. шк., 1992.
10. *Смирнов Н. Н.* Программные средства персональных ЭВМ. — Л.: Машиностроение. Ленинград. отд-ние, 1990.
11. *Фокс Дж.* Программное обеспечение и его разработка: Пер. с англ. — М.: Мир, 1985.
12. Электронные вычислительные машины: В 8 кн. / Под ред. А. Я. Савельева. — М.: Высш. шк., 1993. Кн. 1. Введение в ЭВМ.

§ 4. СИСТЕМЫ ОБРАБОТКИ ТЕКСТОВ

Рекомендации по проведению занятий

Тема предусматривает выработку значительных практических навыков.

На первом занятии студентам предлагается освоить основные принципы работы с Word: знакомство с рабочим полем, меню, командами; набор простого текста; сохранение и загрузка существующего текстового файла.

На последующих занятиях каждый студент выполняет упражнения и в зависимости от приобретенных умений и навыков работы с редактором приступает к выполнению лабораторных работ. Выбор номера задания лабораторных работ каждому студенту можно определить по списку студенческой группы.

Каждый студент сохраняет результаты (файлы) выполненных лабораторных работ в отдельной папке и представляет их преподавателю для защиты.

Представляется полезным осуществлять обмен текстовыми файлами между студентами.

Краткие сведения

Некоторые понятия технического редактирования

В издательских технологиях обработки текстов выделяют художественное и техническое редактирование. В данном практикуме мы ограничимся только техническим.

Объясним некоторые понятия технического редактирования.

Шрифты — основное изобразительное средство издательских систем. Шрифты различают по *гарнитуре* (рисунку), *начертанию*, *кеглю* (размеру).

Гарнитурой называется совокупность шрифтов одного рисунка во всех начертаниях и кеглях. Полный комплект гарнитуры содержит шрифты всех начертаний и кеглей, а в каждом кегле — русский и латинский (и, если надо, другие) алфавиты прописных и строчных букв, а также относящиеся к ним знаки.

Буквы располагаются по *базовой линии*. Расстояние между строками называют *интерлиньяжем*. Размер шрифта, определенный как расстояние между нижним и верхним выносными элементами, называют *кеглем* и измеряют в пунктах. Один пункт равен 0,376 мм. Шрифты могут быть прямыми и наклонными. Наклонный вариант шрифтов часто называют *курсивом*.

Шрифт на компьютере — это файл или группа файлов, обеспечивающих вывод текста на печать со стилевыми особенностями шрифта.

Выбор кегля шрифта осуществляется в соответствии:

- с квалификацией читателя;
- с учетом типа, характера и назначения издания;
- с учетом формата издания и длины строки.

Таблица представляет собой сложную наборную форму, в которую могут входить тексты, цифры, графический материал, формулы или любое сочетание этих элементов. При разметке таблиц учитывают следующее:

- таблицы набирают шрифтом, кегль которого на два пункта ниже кегля шрифта основного текста;
- кегль шрифта головки таблицы должен быть на два пункта ниже кегля шрифта основного текста;
- минимальный формат для набора сплошных текстов выбирают в соответствии с кеглем шрифта;
- если тексты в колонках представляют собой отдельные слова или короткие фразы (не более трех-четырех слов), формат для них выбирают, ориентируясь на самое длинное слово;
- толщина линеек в таблицах должна быть равной 2 п.;
- тексты в головках таблиц при большой насыщенности могут располагаться вертикально;
- в одном издании однородные элементы таблиц оформляются единообразно по шрифту и размещению;
- тематические заголовки разбивают на логически законченные строки, умещающиеся в формат полосы набора.

Для набора *формул* применяют гарнитуру основного шрифта. Однострочные формулы и однострочные элементы многострочных формул набирают шрифтом того же кегля, что и основной текст.

Математические символы набирают светлым курсивом, греческим и латинским светлым прямым, физические и химические символы — светлым прямым, математические сокращения — светлым прямым, сокращенные обозначения физических величин — светлым прямым без точек. Числа и дроби в формулах всегда набирают прямым шрифтом.

Основные правила верстки:

1. Не используйте на одной странице слишком много гарнитур: максимум три разные гарнитуры и не более трех размеров (кеглей).

2. При работе над многостраничным (периодическим) изданием выберите один раз набор гарнитур и строго придерживайтесь его в будущем.

3. При работе над многостраничным (периодическим) изданием строго придерживайтесь сетки-схемы размещения текста и иллюстраций на странице.

4. Не стремитесь максимально заполнить текстом всю площадь страницы.

Настольная издательская система WORD

В последнее время все большую популярность среди широкого круга пользователей завоевывает текстовый процессор Word для Windows.

Word — многофункциональная программа обработки текстов, настольная издательская система. Ее предназначение:

а) набор, редактирование, верстка текста и таблиц;

б) управление всеми пунктами меню, опциями и командами с помощью мыши;

в) просмотр на дисплее готового к печати документа без затраты бумаги на дополнительные распечатки;

г) вставка рисунков и слайдов;

д) заготовка бланков, писем и других документов;

е) обмен информацией с другими программами;

ж) проверка орфографии и поиск синонимов.

Первоначальные сведения и правила работы с WORD

Клавиша **Delete** нужна для стирания символа, расположенного правее положения курсора. Клавиша **Backspase** нужна для стирания символа, расположенного левее положения курсора.

Одно их основных понятий Word — *абзац*. Чтобы начать новый абзац, надо нажать клавишу **Enter**. Основные параметры абзаца (красная строка, межстрочный интервал, выравнивание, абзацный отступ) устанавливаются через меню ***Формат\Абзац***. Примеры абзацев:

Это абзац, состоящий из одной строки. Это абзац, состоящий из одной строки.

Это абзац, состоящий из двух строк. Это абзац, состоящий из двух строк. Это абзац, состоящий из двух строк. Это абзац, состоящий из двух строк. Это абзац, состоящий из двух строк.

Это абзац, состоящий из трех строк. Это абзац, состоящий из трех строк. Это абзац, состоящий из трех строк. Это абзац, состоящий из трех строк. Это абзац, состоящий из трех строк. Это абзац, состоящий из трех строк.

Чтобы работать с текстом, надо научиться его выделять. Это можно делать с помощью мыши или с помощью клавиатуры.

Чтобы выделить	Действие
Слово	Дважды щелкните слово
Строку текста	Переместите указатель к левому краю строки так, чтобы он превратился в стрелку, направленную вправо, после чего щелкните кнопкой мыши
Несколько строк текста	Переместите указатель к левому краю одной из строк так, чтобы он превратился в стрелку, направленную вправо, а затем перетащите указатель вверх или вниз
Часть строки	Переместите указатель к левому краю строки так, чтобы он принял вид I; удерживая нажатой кнопку мыши, проведите вправо до нужного символа
Блок текста	Щелкните начало фрагмента, удерживая нажатой клавишу Shift, щелкните конец фрагмента

Чтобы снять отметку, щелкните по любому месту левой кнопкой мыши.

Чтобы скопировать или переместить отмеченный блок, надо:

1) скопировать или вырезать его в буфер (меню ***Правка\Копировать (Вырезать***) или использовать соответствующие пиктограммы на панели инструментов);

2) щелкнуть в то место, куда его надо вставить;

3) выгрузить из буфера (меню ***Правка\Вставить*** или *пиктограмма* ***Вставить_из_буфера***), выгружать можно столько раз, сколько раз надо вставить.

Задание. Скопировать эту строку 4 раза.

Каждый абзац может иметь свой абзацный отступ и форматирование. Для задания отступа или форматирования надо:

1) выделить абзац;

2) перетащить верхний «бегунок» линейки на нужную цифру.

Например:

Этот абзац имеет отступ 0 см.

Этот абзац имеет отступ 1 см.

Этот абзац имеет отступ 2 см.

Этот абзац имеет отступ 7 см.

Для задания выравнивания надо:

1) выделить абзац;

2) нажать на одну из кнопок выравнивания 

Например:

Этот абзац выровнен по левому краю.

Этот абзац выровнен по правому краю.

Этот абзац выровнен по центру.

Интервал между строками абзаца может быть различен.

Например:

Интервал между строками этого абзаца — одинарный. Интервал между строками этого абзаца — одинарный. Интервал между строками этого абзаца — одинарный.

Интервал между строками этого абзаца — полуторный. Интервал между строками этого абзаца — полуторный. Интервал между строками этого абзаца — полуторный.

Интервал между строками этого абзаца — двойной. Интервал между строками этого абзаца — двойной. Интервал между строками этого абзаца — двойной.

Чтобы изменить интервал, надо:

1) выделить абзац;

2) выбрать пункт меню ***Формат\Абзац***;

3) выбрать нужный межстрочный интервал.

Размер шрифта, его стиль и цвет можно изменять. Например:

Это 16-й номер шрифта Arial Cyr.

Это 13-й номер шрифта Times New Roman Cyr, стиль — курсив.

Для этого надо:

1) выделить текст;

2) войти в пункт меню ***Формат\Шрифт***;
3) выбрать нужные параметры.

Во время работы в Word, чтобы узнать размер и кегль (начертание или вид) шрифта, надо установить курсор на текст в образце — вид шрифта и размер высветятся на панели инструментов, или войти в меню *Формат\шрифт*. Также обратите внимание на начертание (жирный, обычный, курсив).

Для набора верхних или нижних индексов необходимо:
1) набрать символ;
2) выделить его;
3) войти в пункт меню ***Формат\Шрифт***;
4) активизировать нужный индекс.

Например, для набора выражения «$3^2 + 5^4$» надо сначала набрать 32 + 54. Затем выделить 2, установить для нее атрибут — верхний индекс. Далее аналогично для цифры 4 выполнить те же действия.

Для любого абзаца можно заказывать обрамление и заливку.

Примеры:

Обрамление снизу. Обрамление снизу. Обрамление снизу.

Обрамление слева. Обрамление слева. Обрамление слева.

Обрамление снаружи. Обрамление снаружи. Обрамление снаружи. Обрамление снаружи.

Заливка +Обрамление

Для этого надо:

1) выделить абзац;

2) нажать одну из кнопок  для обрамления.

Или воспользоваться пунктом меню ***Формат\Обрамление_и_заливка***.

Чтобы отказаться от заливки и обрамления, надо:
1) выделить абзац;
2) воспользоваться пунктом меню ***Формат\Обрамление_и_заливка***;
3) выбрать **нет**.

Чтобы сохранить результаты работы в новом файле, надо:
1) щелкнуть пункт меню ***Файл\Сохранить_как***;
2) ввести имя файла с клавиатуры;
3) щелкнуть по кнопке ***Сохранить***.

Все установленные параметры абзаца и текста можно сохранить под определенным стилем. Для этого надо:
1) выделить абзац;
2) установить для него атрибуты шрифта и абзаца;
3) выбрать пункт меню ***Формат\Стиль***, кнопка ***Создать***;
4) заполнить диалоговое окно.

Далее этот стиль можно применить к другому абзацу: выделить абзац, через панель инструментов ***Стиль*** выбрать нужный стиль.

При нажатии на клавишу **Enter** создается новый абзац. Он наследует все параметры предыдущего абзаца.

Внедрение и связывание объектов

Для объединения в одном документе объектов разного происхождения, например, чтобы вставить в текст графику, музыку, фрагмент электронной таблицы и т.д., в приложениях Windows широко применяется технология OLE (Object Link and Embedding) — связь и внедрение объектов. Это означает, что помещаемый в текст объект может включаться в него в двух вариантах:

- как внедренный, т.е. он становится частью документа Word; все изменения, которые производятся в источнике, не будут отражаться в документе; например, если внедрить ранее созданный в Pbrush рисунок в Word, затем загрузить Pbrush и произвести изменения над рисунком, то эти изменения не отразятся на рисунке в текстовом документе;
- как связанный; здесь, наоборот, если производятся изменения над рисунком в источнике, то эти изменения отражаются и на рисунке, помещенном в текстовый документ.

Различают также термины технологии OLE:

- клиент — приложение, принимающее объект;
- сервер — приложение, средствами которого создается объект.

Внедренный или связанный объект можно впоследствии редактировать средствами приложения-сервера. Для этого надо выполнить двойной щелчок мыши на данном объекте (Word загрузит приложение-сервер), произвести изменения над объектом и вернуться в приложение-клиент.

Технология OLE осуществляется двумя способами:

1) через буфер обмена — командой приложения ***Правка\Специальная_вставка...***;
2) командой приложения ***Вставка\объект...***.

Первым способом можно внедрить (или связать) фрагмент документа или внедрить (или связать) весь документ, вторым способом — только целый документ. Причем во втором случае можно внедрять или связывать уже готовый объект, а также создать новый объект в процессе внедрения (связывания).

Через буфер обмена	Командой ***Правка\Объект...***
Внедрение:	Внедрение готового объекта:
1) в приложении-сервере выделить объект	1) в приложении-клиенте выполнить команду ***Вставка\Объект...***
2) занести его в буфер обмена (***Правка\Копировать***)	2) в появившемся диалоговом окне активизировать вкладку ***Создание*** из файла
3) перейти в приложение-клиент	3) через кнопку ***Обзор*** выбрать внедряемый объект
4) выполнить команду ***Вставка\Специальная вставка...***	Внедрение объекта, создаваемого в процессе внедрения:
5) в зависимости от того, какое приложение-сервер (например, MS Excel, CorelDraw, MS Word и т.д.), Word предложит диалоговое окно для выбора различных типов внедрения	1) в приложении-клиенте выполнить команду ***Вставка\Объект...***
	2) в появившемся диалоговом окне активизировать вкладку ***Создание***
	3) выбрать приложение-сервер, создать в нем объект
	4) вернуться в приложение-клиент (щелкнуть вне границы размеров выделенной области)

Окончание таблицы

Через буфер обмена:	Командой ***Правка\Объект...***
Связывание: выполнить те же самые действия, что и при внедрении, только в пункте 5 в диалоговом окне необходимо активизировать переключатель ***Связать***; при этом можно активизировать еще переключатель ***Значок***, который выводится вместо связанного объекта	Связывание: выполняется аналогично внедрению готового объекта, только в пункте 2 в диалоговом окне активизировать переключатель ***Связь_с_файлом***; при этом можно активизировать еще переключатель ***Значок***, который выводится вместо связанного объекта

Примечание. Во многих случаях команда вставки содержимого буфера обмена ***Правка\Вставить*** по умолчанию внедряет объект, т.е. выполняет роль одного из вариантов внедрения.

Графические изображения в документах Word

В текстовом процессоре Word используются следующие режимы работы с графикой:

1) вставка (или связь) объекта, созданного в некотором графическом редакторе (например, PaintBrush, PhotoShop и т.д.); наиболее часто используется способ внедрения или связи через буфер обмена;

2) рисование в самом документе, используя панель инструментов ***Рисование***;

3) использование готовых рисунков из коллекции Clipart, а также его редактирование; графический файл либо внедряется в документ Word, либо с ним устанавливается связь.

Контрольные вопросы

1. Назовите основные атрибуты шрифта.
2. Назовите основные атрибуты абзаца.
3. Как скопировать блок текста?
4. Какие основные пункты меню используются для форматирования текста и абзаца?
5. Как создать новый стиль?
6. Чем различаются внедренный и связанный объекты?
7. Что называется приложением-сервером и приложением-клиентом?

Темы для рефератов

1. Системы обработки текстов в MS DOS.
2. Текстовый редактор Лексикон.
3. Текстовый процессор Word.
4. Настольная издательская система PageMarker.
5. Настольная издательская система TeX.

Темы семинарских занятий

Семинар 1

Издательские системы. Основы технического редактирования текста. Принципы работы с текстовыми редакторами. Виды изданий, шаблоны текстов.

Семинар 2

Технология работы в текстовом процессоре Word. Ввод и редактирование текста. Форматирование текста. Стили и шаблоны. Работа с таблицами. Графика в Word. Внедрение и связывание объектов.

Рекомендации по программному обеспечению

1. Текстовый процессор Word (от версии Word-6 и старше).
2. Пакет Microsoft Office.

Задачи и упражнения

Ввод и редактирование текста, работа с блоками, форматирование абзацев, списки

Упражнение №1. Работа со шрифтами, копирование блока текста

1. Наберите следующее предложение «**Я изучаю Microsoft Word успешно**», скопируйте его пять раз и оформите различными шрифтами, как предложено ниже.

Я изучаю Microsoft Word успешно.

Я изучаю Microsoft Word успешно.

Я изучаю Microsoft Word успешно.

Я изучаю Microsoft Word успешно.

Я изучаю Microsoft Word успешно.

Упражнение № 2. Форматирование абзацев

Форматирование абзацев осуществляется при помощи меню ***Формат\Абзац***, а также при помощи кнопок форматирования (по левому, по правому краю, по центру, по ширине листа). Предварительно необходимо выделить форматируемый абзац.

Оформите текст следующим образом, используя кнопки форматирования:

Я изучаю Microsoft Word успешно.

Я изучаю Microsoft Word успешно.

Я изучаю Microsoft Word успешно.

Я изучаю Microsoft Word успешно.

Я изучаю Microsoft Word успешно.

Красная строка оформляется одним из следующих способов (предварительно надо выделить абзац(ы)):

- при помощи меню ***Формат\Абзац*** установить параметры «Первая строка»;
- схватить верхний бегунок и, не отпуская, перетащить на нужное место.

Оформите следующий текст, используя оба способа. Красную строку установите 1,5 см.

Я изучаю Microsoft Word успешно. *Я изучаю Microsoft Word успешно.* Я изучаю Microsoft Word успешно. *Я изучаю Microsoft Word успешно.* Я изучаю Microsoft Word успешно.

Упражнение № 3. Использование списков

Список форматируется как до ввода элементов, так и для набранных в виде отдельных абзацев элементов.

Существует несколько различных способов форматирования списков:

- ***Формат\Список***;
- ***Контекстное_меню\Список***;
- с помощью кнопок панели *Форматирование*: *Нумерация и список.*

Сформируйте список вашей подгруппы и оформите его в виде списка. Для оформления многоуровневого списка используйте один из следующих вариантов:

- комбинацию клавиш Shift + Alt + стрелки влево или вправо (клавиши управления курсором), предварительно установив курсор в нужную строку;
- предварительно выделив нужную строку, вызовите контекстное меню и выберите нужную команду (понизить уровень или повысить уровень).

Нумерованный список	Маркированный список	Многоуровневый список
1. Иванов 2. Петров 3. Сидоров 4. и т.д.	• Иванов • Петров • Сидоров • и т.д.	1. Мужской пол 1.1. Иванов 1.2. Петров 2. Женский пол 2.1. Петрова 2.2. Калинина

Работа с таблицами. Вычисления в таблицах. Встроенные функции в Word

Упражнение № 4. Создание таблиц

Создайте следующую таблицу (если имеются навыки работы с клавиатурой, необязательно вводить текст во все ячейки).

Сведения об обеспечении образовательного процесса учебной литературой в учебном центре ТОО «МаксСофт»

№ пп	Наименование предмета (курса, дисциплины, учебного плана) по годам обучения	Число обучающихся, воспитанников, изучающих предмет (курс, дисциплину)	Обеспечение обучающихся, воспитанников, литературой, указанной в учебной программе предмета (курса) в качестве обязательной	
			Перечень литературы (автор, название, год, место издания, издательство, год издания)	Число экземпляров
1	2	3	4	5
1	Курсы для пользователей	1000	Алексей Бабий. Компьютер куплен. Что дальше?...	1000
2	Курсы для программистов	10	Фирменная документация	1

Технология выполнения:

1. Выберите в меню пункт ***Таблица\Вставить_таблицу***. Укажите число строк и столбцов (по 5).

Или создайте таблицу, используя пиктограмму ***Таблица*** на панели инструментов: щелкните на ней мышью и, не отпуская кнопку мыши, протащите ее вниз и вправо до нужного числа столбцов и строк.

2. Для изменения ширины ячейки используйте меню ***Таблица\Высота и ширина ячейки*** (предварительно выделив нужные ячейки) или изменяйте ширину мышью: установите курсор мыши на линию границы между ячейками, появится двойная стрелка, ухватите мышью и тащите в нужном направлении.

3. Текст в верхней правой ячейке («Обеспечение обучающихся...») идет по длине двух ячеек. Для оформления этих ячеек выполните следующее: выделите две ячейки (четвертую и пятую). Выполните пункт меню ***Таблица\Объединить_ячейки***.

Примечание. При создании таблицы число столбцов и строк рекомендуется брать по максимуму.

4. Для оформления верхних трех левых ячеек выделите сначала в первом столбце две строки и выполните команду ***Таблица\Объединить_ячейки***. Далее выполните те же действия для второго и третьего столбцов.

5. Проведите разделительные линии: а) для этого надо сначала выделить таблицу или ее участки (если это необходимо); б) далее использовать меню ***Формат\Границы*** и ***Заливка*** или использовать пиктограмму ***Границы***.

6. Установите размер шрифта 10 в таблице: для этого первоначально выделите ее (можно выделить, используя меню ***Таблица\Выделить_таблицу***. Курсор перед выделением должен находиться внутри таблицы). Далее установите размер шрифта.

7. Введите текст.

Упражнение № 5. Вычисления в таблицах

Таблица может содержать максимум 31 столбец и произвольное число строк. Ячейки таблицы имеют адреса, образованные именем столбца (A, B, C...) и номером строки (1, 2, 3...), например, A1, C4 и т.д.

Word позволяет выполнять вычисления, записывая в отдельные ячейки таблицы формулы с помощью команды ***Таблица\Формула***... Формула задается как выражение, в котором использованы:

- *абсолютные* ссылки на ячейки таблицы в виде списка (разделяемые знаком «;» — A1; B5; E10 и т.д.) или блока (начало и конец блока ячеек — **A1**:**A10**);
- *ключевые* слова для ссылки на блок ячеек:
  LEFT — ячейки, расположенные в строке левее ячейки с формулой;
  RIGHT — ячейки, расположенные в строке правее ячейки с формулой;
  ABOVE — ячейки, расположенные в столбце выше ячейки с формулой;
  BELOW — ячейки, расположенные в столбце ниже ячейки с формулой;
- константы — числа, текст в двойных кавычках;
- закладки, которым соответствует определенный текст документа (например, числа), созданный с помощью команды ***Правка\Закладка***;
- встроенные функции Word, например, SUM(), AVERAGE().
- знаки операции (+, −, *, /, %, ^, =, <, >, <=, >=).

В данной таблице произвести вычисления успеваемости студентов.

Сведения об успеваемости студентов общеэкономического факультета КГУ за 2000/01 учебный год

№ пп	Учебная дисциплина	Группа	Средний балл	Всего сдавало	Отл.	Хор.	Удовл.	Неудовл.	Неявки
	Высшая математика								
1		51		ячейка E4	12	10	6	3	1
2		52			7	9	6	3	2
3		53			9	8	3	5	3
4		54			8	8	8	3	2
	Итого:								

Технология выполнения:

1. Введите формулы для расчета числа студентов каждой группы, сдавших экзамен по дисциплине «Высшая математика». Для этого установите курсор в ячейку E4 и введите формулу: SUM(RIGHT), предварительно убрав имеющуюся в ней запись. Ввести формулу надо, используя меню ***Таблица\Формула...***.
2. Произведите те же действия для ячеек E5 — E7.
3. Введите формулу для расчетов среднего балла по дисциплине «Высшая математика» для группы 51. Для этого установите курсор в ячейку D4 и введите формулу: =(F4*5+G4*4+H4*3+I4*2)/E4. Выберите формат числа 0,00.
4. Введите аналогичные формулы в ячейки D5 — D7.
5. Введите формулу для расчета общего числа студентов, сдавших экзамен по каждой дисциплине на отлично, хорошо и т.д. Для этого установите курсор в ячейку F8 и введите формулу: SUM(ABOVE), затем аналогично в ячейку G8 и т.д.

Упражнение № 6. Использование табуляторов

Создайте следующую ведомость, используя табуляторы. В конце каждой строки для перехода на новую строку нажимать клавишу **Enter**.

Стипендиальная ведомость

№ группы Фамилия, и., о. Стипендия, руб. Подпись

56 ..... Козлова И.М. ..... 100 _____
56 ..... Петрова И.И. ..... 120 _____
57 ..... Сидорова Н.Н. ..... 100 _____
57 ..... Черникова Л.П. ..... 130 _____
58 ..... Марков О.Л. ..... 100 _____

1. С помощью горизонтальной линейки задайте форматы табуляторов для абзаца с заголовками колонок. Для этого выберите тип табуляторов на линейке: 2,5 см, 7 см, 11 см, 14 см.
2. Введите заголовки колонок.
3. С помощью команды ***Формат\Табуляция*** для абзацев списка задайте форматы табуляторов:
   - 2,5 см — выравнивание влево, заполнитель — линия (2);
   - 7 см — выравнивание влево, заполнитель — линия (2);
   - 11 см — выравнивание влево, заполнитель — линия (2);
   - 14 см — выравнивание вправо, заполнитель — линия (4).
4. Введите текст, используя клавишу **Tab** для перемещения к следующей позиции.
5. Преобразуйте данные колонки в таблицу. Для этого выделите их. Выполните команду ***Таблица\Преобразовать_в_таблицу***.

№ группы	Фамилия, и., о.	Стипендия, руб.	Подпись
56	Козлова И.М.	100	
56	Петрова И.И.	120	
57	Сидорова Н.Н.	100	
57	Черникова Л.П.	130	
58	Марков О.Л.	100	

Примечание. Аналогично можно преобразовать в таблицу текст, оформленный в виде колонок.

Внедрение и связывание объектов

Упражнение № 7. Внедрение объекта через буфер обмена

1. Введите заголовок «Внедрение объекта через буфер обмена».
2. Откройте какой-либо рисунок в графическом редакторе Paint или создайте свой рисунок. Запуск Paint — ***Пуск\Программы\Стандартные\Графический редактор Paint***.
3. Выделите любой фрагмент рисунка, скопируйте его в буфер (***Правка\Копировать***).

4. Перейдите в свой документ и выполните команду ***Правка\Специальная_вставка***, выберите формат вставляемого объекта — Точечный рисунок BMP.

5. Закройте приложение, где создавался рисунок.

Примечание. В качестве сервера внедряемого объекта можно взять любое другое приложение, например, PhotoShop, CorelDraw, Excel и т.д.

Упражнение № 8. Внедрение объекта целиком в виде файла

1. Введите заголовок «Внедрение файла».
2. Выполните команду ***Вставка\Объект***.
3. В появившемся диалоговом окне выберите вкладку ***Создать из файла***.
4. Выберите, используя кнопку ***Обзор***, файл приложения Paint (с расширением BMP).

Упражнение № 9. Объединение нескольких текстовых документов

Для объединения нескольких документов Word или других текстовых файлов можно использовать буфер обмена либо команду ***Вставка\Файл***.

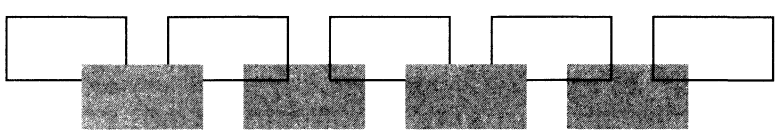
1. Введите заголовок «Объединение нескольких документов».
2. Выполните команду ***Вставка\Файл***. Выбрать какой-либо файл Word.
3. Если нужно осуществить связь, необходимо выбрать переключатель ***Связь с файлом***.

Упражнение № 10. Связывание объектов — непосредственная ссылка на файл

1. Введите заголовок «Непосредственная ссылка на файл».
2. Остальные пункты выполните аналогично пунктам 2 — 4 из упражнения 8. Только в пункте 4 надо обязательно указать переключатель ***Связь с файлом***.

Работа с графикой. Создание графических объектов с использованием панели инструментов «Рисование»

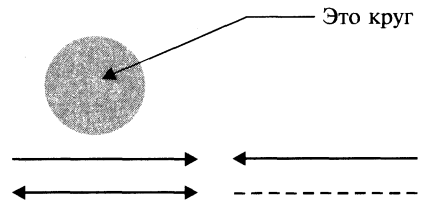
Упражнение № 11. Создание рисунка с использованием операции копирования и переноса



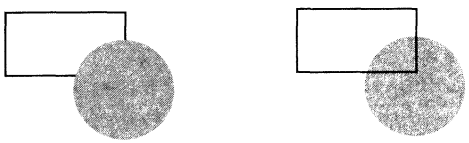
1. Активизируйте панель **Рисование**, если ее нет на экране. Для этого выберите пункт меню **Вид\Панель_инструментов**...
2. Сначала нарисуйте прямоугольник, скопируйте его несколько раз, затем редактируйте цвет и заливку, используя соответствующие инструменты на панели инструментов ***Рисование*** (прямоугольник, цвет линии, цвет заливки).
3. Для нужного расположения объектов используйте инструмент ***Действия\ Порядок.***

4. Сгруппируйте все объекты. Для этого выделите их все: щелкайте мышью на линии незакрашенных прямоугольников и на самих закрашенных прямоугольниках, удерживая нажатой клавишу Shift, затем используйте инструмент ***Действия\ Группировать***.

Упражнение № 12. Выполнение надписи с использованием инструментов Автофигуры \ Выноски



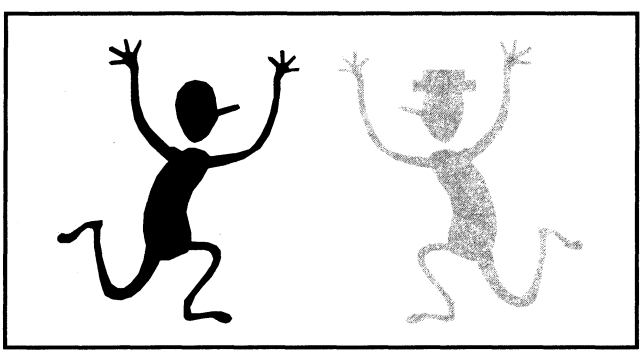
Упражнение № 13. Создание рисунка, копирование его, редактирование расположения объектов



Использование готовых рисунков из коллекции Clipart, а также их редактирование

Упражнение № 14. Вставка рисунка из коллекции Clipart и редактирование его

1. Выполните команду ***Вставка\Рисунок\Из_файла...*** .
2. Выберите подходящий рисунок.
3. Для редактирования рисунка необходимо выполнить двойной щелчок на рисунке.

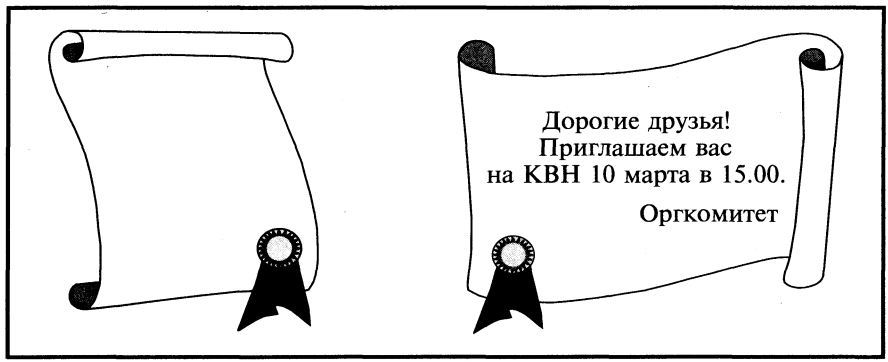


Например, рисунок слева — это тот рисунок, который загружается, рисунок справа — изменен цвет заливки с черного на серый, дорисована шляпа, сгруппированы объекты и выполнено зеркальное отражение через кнопку ***Действия\Повернуть\Отразить***

Клавиша Shift используется для одновременного выделения нескольких объектов.

Ниже приведен еще один пример измененного рисунка из Clipart (Diploma).

Для создания надписи используйте кнопку ***Надпись***, расположенную на панели инструментов ***Рисование:***, активизируйте ее, затем протяните на рисунке (как рисуем прямоугольник), затем вводите текст.



Упражнение № 15. Создание рекламы (самостоятельная работа)

Создайте текст рекламы «Цветное фото KODAK». Для написания текста «Цветное фото» используйте приложение WordArt. WordArt находится в меню ***Вставка\Рисунок\Объект_WordArt ...*** .

Работа с приложением MS Equation

Упражнение № 16. Использование *Редактора формул*

Наберите математические выражения, входящие в тексты заданий.

А. Решите неравенства:

$$\begin{cases} 3(x+1) - \dfrac{x-2}{4} < 5x - 7 \cdot \dfrac{x+3}{2}, \\ 2x - \dfrac{x}{3} + 6 < 4x - 3. \end{cases}$$

1. Запустите редактор формул (***Вставка\Объект\Microsoft_Equation***).
2. Для набора формул используйте панель инструментов (скобки, дроби, умножение).
3. Если текст задания вы хотите набрать, находясь в редакторе формул, то выберите команду ***Стиль\Текст***.
4. Для рисования фигурной скобки выделите обе строки, затем на панели инструментов «скобки» выберите левую фигурную скобку.
5. Возвращение в Документ происходит щелчком мышью вне окна Microsoft Equation.
6. Для редактирования набранной формулы (если необходимо после вставки внести изменения) используется двойной щелчок мыши.

Б. Корни уравнения $ax^2 + bx + c = 0$ находятся по формуле

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

В. Формула косинуса суммы (для набора греческих букв воспользуйтесь кнопкой **λωβ**)

$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta.$$

Динамический обмен данными

Упражнение № 17. Создание серийных писем

Создайте приглашение на вечер выпускников.

Word предоставляет эффективный механизм работы с серийными (шаблонными) письмами, т.е. с письмами, имеющими одинаковое содержание, рассылаемыми различным адресатам.

Принцип создания серийных писем заключается в выполнении следующей последовательности действий:

1) создания основного текста;

2) создания данных для подстановки или открытия готового файла в формате базы данных с данными для подстановки;

3) вставки в основной текст полей, которые будут заменяться в процессе слияния на нужные слова, например, на имена и фамилии;

4) проверки, правильно ли работает подстановка, т.е. замена полей на реальное содержание;

5) осуществления слияния документов.

I. Создание основного содержания письма

1. Выберите пункт меню ***Сервис\Слияние***.

2. В появившемся окне *Слияние* активизируйте кнопку ***Создать*** под заголовком *Основной документ*.

3. Выберите из открывшегося списка строку *Документы на бланке...* После этого появится диалоговое окно, в котором спрашивается: использовать активное окно для создания документа или создать новое.

Примечание. Если перед выбором окна *Слияние* был открыт заготовленный для письма документ, например, со вставленным графическим оформлением, то следует нажать кнопку ***Активное_окно***.

4. Нажмите кнопку ***Создать*** основной документ.

5. Нажать кнопку ***Правка\Выбрать*** свой документ.

6. После этого введите основное содержание письма, оставив пустые абзацы или пропуски в тех местах, где будут стоять имена, фамилии, адреса — все то, что будет подставляться (при этом на экране появится новая панель инструментов).

7. Сохраните свое письмо.

II. Создание структуры источника данных

1. Активизируйте кнопку на появившейся панели ***Окно диалога_Слияние*** (либо вызовите его через меню *Сервис\Слияние*).

2. В появившемся окне нажмите на кнопку ***Получить_данные***.

3. В открывшемся списке выберите строку ***Создать_источник_данных***. Появится окно *Создание_источника_данных*. В этом окне слева можно видеть список уже готовых полей. Существует возможность добавлять и удалять поля из списка, формируя необходимую структуру источника данных.

4. Оставьте пункты: имя, фамилия, адрес 1 (переименуйте в адрес), город. Добавьте поле «Пол». Поменяйте местами поля «Фамилия» и «Имя».

Примечание. Имя поля не должно быть длиннее 40 символов и должно начинаться с цифры. В имени поля можно использовать буквы, цифры, знаки подчеркивания «_». Пробелы в имени поля не допускаются.

5. Нажмите на кнопку подтверждения «ОК». Появится окно ***Сохранить_источник_данных***. Сохраните его. Оно будет иметь расширение .doc.

6. Word выводит сообщение о том, что источник данных не содержит самих данных. Нажмите кнопку ***Правка_источника_данных***. Появится диалоговое окно для ввода информации о выпускниках.

7. Введите 3—4 записи. После заполнения каждой записи нажимайте на кнопку ***Добавить***.

8. После ввода информации о всех приглашаемых на вечер сохраните источник информации на диске.

III. Вставка полей слияния

1. В окне ***Слияние*** включите режим редактирования основного текста письма.

2. Установите текстовый курсор в том месте письма, где надо расположить определенное поле (пока кроме имени).

3. Активизируйте кнопку ***Поля_слияния*** и выберите из списка нужное поле.

4. Для обращения используйте перед именем приглашаемого какое-либо прилагательное, например, «Уважаемый», «Дорогой» и т.д. Для этого нажмите кнопку

Поле и выберите из списка строку IF...THEN...ELSE (вставка условий в текст письма). Появится диалоговое окно.

5. Щелкнув мышью в окошечке ***Поле***, выберите «Пол».
6. В списке сравнить установите ***Равно***.
7. В поле ***Сравнить с чем*** введите мужской пол (точно так, как вы писали в полях документа источника данных).
8. В поле ***Вставить текст*** введите «Уважаемый», в поле ***В противном случае***... — «Уважаемая».
9. Вставьте поле слияния ***Имя*** после данного обращения.
10. Сохраните текст письма.

IV. Осуществление слияния

Для слияния можно использовать несколько вариантов.

1. Использовать кнопки на панели инструментов:
   А. Слияние с проверкой ошибок — кнопка ***Поиск_ошибок***.
   Б. Слияние в новый документ.
   В. Слияние при печати, если необходимо сразу выводить на печать.
   Г. Слияние (варианты слияния).
2. Использовать меню ***Сервис\Слияние\Объединить***.

Примечание. В окне ***Слияние*** можно выбрать перед слиянием кнопку ***Отбор_записей***, если не нужны все записи, т.е. осуществить выбор записей. Например, выбрать список только жителей Красноярска или выбрать список записей только мужского пола и т.д.

Лабораторные работы

Лабораторная работа № 1
Ввод и редактирование текста, работа с блоками, форматирование абзацев, списки

Время выполнения 8 часов.

Задание. Составьте краткий реферат на тему «Мой любимый писатель». В реферате отразите:

- биографические данные писателя;
- список его нескольких произведений;
- отрывки из его произведений.

Оформите реферат, используя 2 — 3 своих стиля.

Варианты заданий

Вариант 1. «Мой любимый писатель А.Дюма».
Вариант 2. «Мой любимый писатель А.С.Пушкин».
Вариант 3 — 25. «Мой любимый писатель...» (выберите самостоятельно).

Лабораторная работа № 2
Работа с таблицами. Вычисления в таблицах. Встроенные функции в Word

Время выполнения 6 — 8 часов.

Задание. Рассмотрим темы «Бизнес-план» и «Рекламный прайс-лист».

По первой теме разработайте годовой бизнес-план выполнения заданной научно-технической или производственной программы. В таблице отразите вид деятельности, исполнителей, финансирование по месяцам, затраты по кварталам и за год.

По второй теме составьте рекламный листок по издаваемой (продаваемой) продукции из не менее 10 наименований. В таблице отразите наименование товара, категорию (сорт, комплектность и т.п.), стоимость в у.е., пересчет в рублях на текущий валютный курс, суммарную стоимость всех обозначенных продуктов.

Варианты заданий

Вариант 1. Бизнес-план ремонта школы.
Вариант 2. Бизнес-план постройки гаража.
Вариант 3. Прайс-лист фирмы по продаже компьютеров.
Вариант 4. Бизнес-план деятельности фирмы.
Вариант 5. Бизнес-план выполнения проектной работы.
Вариант 6. Прайс-лист деревообрабатывающей фирмы.
Вариант 7. Бизнес-план ремонта квартиры.
Вариант 8. Бизнес-план постройки дачного домика.
Вариант 9. Прайс-лист фирмы по продаже расходных материалов.
Вариант 10. Бизнес-план по выполнению научного проекта.
Вариант 11. Бизнес-план выполнения экспериментальной работы в школе.
Вариант 12. Прайс-лист фирмы по продаже телевизоров.
Вариант 13. Бизнес-план ремонта дороги.
Вариант 14. Бизнес-план постройки бани или сауны.
Вариант 15. Прайс-лист фирмы по продаже музыкальных инструментов.
Вариант 16. Бизнес-план по созданию программного продукта.
Вариант 17. Бизнес-план организации платных курсов по информатике.
Вариант 18. Прайс-лист фирмы по продаже аудио- и видеотехники.
Вариант 19. Бизнес-план реконструкции помещения.
Вариант 20. Бизнес-план постройки погреба.
Вариант 21. Прайс-лист фирмы по продаже фототоваров.
Вариант 22. Бизнес-план постройки овощехранилища.
Вариант 23. Бизнес-план реконструкции здания.
Вариант 24. Прайс-лист фирмы по продаже спортинвентаря.
Вариант 25. Прайс-лист фирмы по продаже канцтоваров.

Лабораторная работа № 3
Работа с графикой. Внедрение и связывание графических объектов

Время выполнения 4 часа.

Задание «Гербы, эмблемы, флаги». Выберите символику вуза, города, региона или страны. Представьте выбранную символику в нескольких видах: нарисуйте эскиз «вручную», внедрите через буфер обмена рисунок, полученный простейшим графическим редактором, свяжите с графическим объектом и др.

Варианты заданий

Вариант 1. Символика Российского государства.
Вариант 2. Символика Вашего города, региона.

Вариант 3—25. Придумайте самостоятельно собственную символику Вашего вуза, факультета, группы.

Лабораторная работа № 4
Работа с приложением MS Equation (математические формулы)

Время выполнения 4—6 часов.

Задание. Наберите фрагмент из учебника математики по указанной в варианте теме, содержащий математические формулы. Фрагмент должен представлять собой связный текст; его выбор и объем предварительно согласуйте с преподавателем, ведущим лабораторную работу.

Варианты заданий

Вариант 1. Теория пределов.
Вариант 2. Непрерывность функции.
Вариант 3. Производная.
Вариант 4. Дифференциал функции.
Вариант 5. Правила дифференцирования.
Вариант 6. Производные элементарных функций.
Вариант 7. Неявные функции.
Вариант 8. Неопределенный интеграл.
Вариант 9. Элементы таблицы интегрирования элементарных функций.
Вариант 10. Определенный интеграл.
Вариант 11. Основная теорема интегрального исчисления.
Вариант 12. Двойной интеграл.
Вариант 13. Ряды Тейлора.
Вариант 14. Интерполирование по Лагранжу.
Вариант 15. Интерполирование по Ньютону.
Вариант 16. Метод наименьших квадратов.
Вариант 17. Дифференциальные уравнения первого порядка.
Вариант 18. Метод Рунге — Кутта.
Вариант 19. Метод простой итерации для решения нелинейных уравнений.
Вариант 20. Метод Ньютона для решения нелинейных уравнений.
Вариант 21. Метод Гаусса для решения систем линейных алгебраических уравнений.
Вариант 22. Определители и их свойства.
Вариант 23. Операции над матрицами.
Вариант 24. Скалярное и векторное произведение векторов.
Вариант 25. Уравнение прямой линии на плоскости.

Лабораторная работа № 5
Работа с формами

Время выполнения 4 часа.

Задание. Составьте формы одновременно рассылаемых многим адресатам текстовых материалов единого содержания (письма, приглашения, оповещения и т.п.).

Варианты заданий

Вариант 1. Приглашение на свадьбу.
Вариант 2. Оповещение о встрече выпускников педагогического института.
Вариант 3. Совещание ректората, Ученого совета.
Вариант 4. Информационное сообщение о проведении конференции.
Вариант 5. Приглашение участнику конференции.
Вариант 6. Рассылка участникам заочной олимпиады по информатике.
Вариант 7. Приглашение на День рождения.
Вариант 8. Оповещение о встрече ветеранов войны.
Вариант 9. Оповещение о семинаре для учителей информатики.
Вариант 10. Информационное сообщение о сроке и повестке заседания Госдумы.
Вариант 11. Приглашение участнику соревнований.
Вариант 12. Рассылка заданий участникам дистанционной олимпиады по информатике.
Вариант 13. Приглашение на юбилей.
Вариант 14. Оповещение о приезде комиссии.
Вариант 15. Совещание методического объединения учителей.
Вариант 16. Информационное сообщение о проведении Симпозиума.
Вариант 17. Приглашение участнику Симпозиума.
Вариант 18. Рассылка информации победителям конкурса.
Вариант 19. Приглашение на празднование Нового года.
Вариант 20. Оповещение о собрании акционеров предприятия.
Вариант 21. Извещение фирмам-партнерам о прекращении деятельности предприятия.
Вариант 22. Информационное сообщение о сроке и повестке заседания Законодательного собрания.
Вариант 23. Повестка офицерам запаса о явке в военкомат.
Вариант 24. Рассылка информации о кандидате в президенты доверенным лицам.
Вариант 25. Рассылка автореферата диссертации.

Дополнительная литература

1. *Вемпен Ф.* Microsoft Office 97 Professional: 6 книг в одной: Пер. с англ. — М.: Бином, 1997.
2. *Власенко С., Маленкова А.* Word 97 в вопросах и ответах. — СПб.: BHV — Санкт-Петербург, 1996.
3. *Джонс Э., Саттон Д.* Библия пользователя Office 97: Пер. с англ. — Киев: Диалектика, 1997.
4. *Ермолович Е.А., Макарова С.В., Хегай Л.Б.* Операционные системы и информационные технологии. — Красноярск, 2000.
5. *Ефимова О., Морозов В., Шафрин Ю.* Курс компьютерной технологии. — М.: АБФ, 1998.
6. *Ефимова О., Морозов В., Шафрин Ю.* Практикум по компьютерной технологии. — М.: АБФ, 1998.
7. *Лоу Д.* Секреты Word для Windows'95. — Киев: Диалектика, 1996.
8. *Львовский С.М.* Набор и верстка в пакете LaTeX. — М.: Космосинформ, 1994.
9. *Макарова Н.В. и др.* Информатика. Практикум по технологии работы на компьютере. — М.: Финансы и статистика, 1998.

10. *Наймершайм Д.* Word 6.0 для Windows: Пер. с англ. — М.: Междунар. отношения, 1995.
11. *Хаслер Р., Франенштих К.* Word 6.0 для Windows: Пер. с нем. — М.: Экон, 1994.
12. *Шкаев А. В.* Настольные издательские системы. — М.: Радио и связь, 1994.

§ 5. СИСТЕМЫ КОМПЬЮТЕРНОЙ ГРАФИКИ

Рекомендации по проведению занятий

Тема предусматривает выработку значительных практических навыков.

В начале занятий преподаватель знакомит студентов с основными понятиями компьютерной графики (1—2 семинарских занятия), знакомит с демонстрационными примерами и объясняет суть выполняемых упражнений.

На последующих занятиях студенты выполняют лабораторные работы и сохраняют свои результаты (в отдельных папках) для последующей их защиты перед преподавателем.

Рекомендуется проводить обмен рисунков между студентами, а также устраивать компьютерный вернисаж и конкурсы.

Краткие сведения

Растровый способ изображения графической информации

Существует два способа реализации построения изображений на экране дисплея — *векторный* и *растровый*. На современных персональных компьютерах чаще используется растровый способ изображения графической информации.

Растровое изображение — это совокупность светящихся (различными цветами) точек, координаты которых определяются декартовой (прямоугольной) системой с началом координат (как правило) в левом верхнем углу экрана. Абсцисса x точки увеличивается слева направо, ордината y — сверху вниз. Таким образом, любая графическая операция сводится к работе с отдельными точками экрана монитора — пикселами.

Большинство языков программирования имеют свои стандартные *графические библиотеки*. Так, у Бейсика графические команды являются встроенными; системы программирования Турбо-Паскаль содержат графическую библиотеку (модуль Graph.tpu), имеющую в своем составе процедуры и функции обработки простейших графических образов.

В последние годы возрос интерес со стороны пользователей к специальным инструментальным программам машинной графики: графическим редакторам, издательским системам и т. п. В них предоставляется удобный интерфейс для пользователей, автоматизируется большое число разнообразных действий с графической информацией — от построения простейших рисунков до создания мультипликационных (анимационных) роликов.

Графический редактор PaintBrush фирмы Microsoft

Как правило, ***PaintBrush*** является встроенным в современные программные среды (Windows, MS Office и т. п.). Позволяет, используя манипулятор «Мышь», выполнять рисунки — черно-белые и цветные, оформлять их текстом, выводить на пе-

чать. В PaintBrush можно работать с фрагментами графических изображений — копировать, перемещать, поворачивать, изменять размеры, записывать на диск и считывать с диска.

После загрузки программы появляется рабочий экран редактора. Большую часть экрана занимает рабочее поле, окрашенное в фоновый цвет. Над рабочим полем находится меню, позволяющее выполнять команды редактирования. Слева от рабочего поля есть панель инструментов, на которой подсветкой показан инструмент, являющийся в данный момент рабочим. Под рабочим полем расположена палитра. На левом краю палитры выводятся два вложенных квадрата, внутренний из которых окрашен в рабочий цвет, а внешний — в фоновый. В левом нижнем углу экрана выводится калибровочная шкала, которая позволяет устанавливать ширину рабочего инструмента (кисти, резинки и т.д.). Вдоль нижнего и правого края рабочего поля выводятся поля движения для перемещения рабочего поля по картинке, если размеры картинки больше размеров рабочего поля.

Графический редактор CorelDraw

По своей идеологии графический пакет ***CorelDraw*** близок к ***PaintBrush***, рассмотренному выше, но имеет гораздо больше возможностей и удобств. Выбор команды в меню осуществляется стандартными способами. Для вызова диалоговой панели выбора (установки) параметров необходимо установить указатель мыши на пункт меню или команду и щелкнуть кнопкой или нажать клавишу **Alt** и клавишу, соответствующую выделенной букве. Выполнение команд происходит после подтверждения правильности установки всех параметров (выбора значений) активизацией экранной кнопки OK или нажатием клавиши **Enter**.

Рисунок перед началом редактирования необходимо выбрать. Для этого следует активизировать пиктограмму с помощью мыши или нажатием клавиши **Пробел**, переместить указатель на любую точку контура рисунка и щелкнуть кнопкой. Выбранный рисунок будет окружен восемью квадратами черного цвета. Можно выделить одновременно несколько объектов, последовательно выбирая их с помощью мыши при нажатой клавише **Shift** либо отмечая на экране прямоугольную область, в которой они расположены. Для одновременного выбора всех рисунков на экране необходимо активизировать пиктограмму, переместить указатель мыши в один из углов выбираемого прямоугольного контура, нажать кнопку и, не отпуская ее, переместить указатель в противоположный угол и отпустить кнопку. Контур будет изображен штриховой линией. Для отмены выбора — переместить указатель мыши за контур и щелкнуть кнопкой.

Для изменения масштаба выводимого на экран рисунка необходимо активизировать соответствующую пиктограмму.

Для работы с текстом необходимо активизировать пиктограмму текста *TEXT*. Для выделения фрагмента текста необходимо активизировать пиктограмму с помощью мыши или клавиши **Пробел**. Переместите указатель мыши в один из углов выбираемого прямоугольного контура, нажмите кнопку **K**, не отпуская ее, переместите указатель в противоположный угол, отпустите кнопку. Весь контур будет окружен штриховой линией. Для выделения отдельного символа необходимо активизировать пиктограмму, переместить указатель мыши в один из углов выбираемого прямоугольного контура, нажать кнопку и, не отпуская ее, переместить указатель в противоположный угол. Выбранный объект не будет окружен рамкой из квадратов, однако на нем будут выделены все узловые точки. Выбор нескольких символов осуществляется аналогично при нажатой клавише **Shift**.

После запуска AutoCAD'а на текстовом экране появляется главное меню:

0 — выход;
1 — создание нового чертежа;
2 — редактирование существующего чертежа;
3 — вывод на плоттер;
4 — вывод на принтер;
5 — конфигурация;
6 — файловые утилиты;
7 — шрифты;
8 — стыковка со старыми версиями.

Режимы экранного меню:

AUTOCAD	—	выход в головное меню;
* * * *	—	режим объектного захвата;
BLOCKS	—	работа с блоками;
DISPLAY	—	работа с изображением без его изменения;
SETTINGS	—	настройка;
DIM	—	обезразмеривание;
EDIT	—	редактирование;
DRAN	—	рисование;
LAYER	—	работа со слоями;
INQUIRY	—	справки о примитивах;
UTILTTS	—	выход в DOS, запись чертежей в разных форматах;
PLOT	—	получение твердой копии;

и т.д.

В режиме DRAW (рисуй) имеется возможность строить графические примитивы и проводить с их помощью синтез изображений. Например, здесь существует восемь способов рисования дуг:

- по трем точкам на дуге /3 points/;
- по начальной точке, центру и длине хорды /S,C,L/;
- по начальной точке, центру и заключенному углу /S,C,A/;
- по начальной точке, конечной точке и радиусу /S,E,R/;
- по начальной точке, конечной точке и заключенному углу /S,E,A/;
- по начальной точке, конечной точке и исходному направлению /S,E,D/;
- по продолжению предыдущей линии или дуги /CONTIN:/.

Контрольные вопросы

1. Чем отличается растровый от векторного способа представления изображения на экране дисплея?
2. Что такое «пиксел»?
3. В чем заключается принцип создания изображения на экране дисплея?
4. Как формируется цветное изображение?
5. В чем преимущества компьютерной графики от традиционной?
6. Назовите основные возможности графических редакторов?
7. В чем заключаются отличия инженерной графики от иллюстративной?

Темы для рефератов

1. Возможности CorelDraw.
2. Что может Adobe Photoshop.
3. Обзор графических редакторов для IBM PC.
4. Компьютерная анимация.
5. Сканирование и распознавание изображений.
6. Возможности и перспективы развития компьютерной графики.
7. Форматы графических файлов.

Темы семинарских занятий

1. Знакомство с элементами компьютерной графики. Художественная графика. Инженерная графика.
2. Иллюстрационная графика и дизайн. Динамическая графика. Организация мультипликации и анимации.

Рекомендации по программному обеспечению

Для проведения практических занятий по компьютерной графике рекомендуются следующие программы:

1. PaintBrush.
2. CorelDraw.
3. AutoCAD.
4. Photoshop.
5. 3Dmax.

Задачи и упражнения

Упражнение № 1. Графический редактор PaintBrush. Рисование на экране. Текст. Сохранение в формате PCX

1. Перед началом рисования необходимо установить *цвет*. Для установки основного цвета поместите указатель мыши на нужный цвет и щелкните левой кнопкой. Для выбора второго цвета поместите указатель мыши на нужный цвет и щелкните правой кнопкой.

2. Выберите кисть для рисования на панели инструментов. След кисти (мазок) возникает на рабочем поле при фиксации левой кнопки мыши. Нарисуйте какой-нибудь рисунок.

3. Панель инструментов позволяет выбирать и изображать на «холсте» простейшие графические примитивы: линии, окружности, прямоугольники и т.д. Нарисуйте несколько геометрических фигур.

4. Перед началом *редактирования рисунка* его необходимо выделить (выбрать). Для этого активизируйте пиктограмму, переместите указатель мыши в один из углов выбираемого прямоугольного контура, нажмите кнопку и, не отпуская ее, переместите указатель в противоположный угол, после чего отпустите кнопку. Весь контур будет изображен штриховой линией. Если вы хотите вырезать произволь-

ную область на экране, активизируйте пиктограмму, переместите указатель мыши в начальную точку вырезания и щелкните левой кнопкой, переместите указатель в следующую точку и снова щелкните левой кнопкой и т.д. Для отмены выбора переместите указатель мыши за контур и щелкните левой кнопкой.

5. *Окраска контура объекта* осуществляется выбором *цвета заливки* в *палитре* с помощью указателя мыши. Окрасьте свои геометрические фигуры разными цветами.

6. Перед началом работы с текстом необходимо выбрать гарнитуру командой ***Style\Font***. Придумайте заголовок к рисунку. Подпишите рисунок своим именем.

7. Сохраните рисунок с помощью программы ***Frieze***. Frieze предназначена для сохранения выводимого на экран изображения в графическом PCX-файле с одновременным сохранением оформления экрана, возможностью последующего редактирования данного изображения и вставки его в качестве иллюстрации в текстовые редакторы и настольные издательские системы.

Упражнение № 2. Графический редактор PaintBrush. Построение узоров

Узор на плоскости получается из элементарного мотива с помощью образующих элементов его группы симметрий. Для бордюров существует 7 различных наборов образующих его групп симметрий, для орнаментов — 17 (см. учебник «Информатика»). Чтобы построить узор на компьютере, необходимо сделать следующее:

1) создать элементарный мотив узора (в графическом редакторе, на языке программирования, полученного со сканера);

2) выбрать и задать образующие или использовать метод калейдоскопа;

3) строить образы каждой точки элементарного мотива по его образующим, т.е. получить из элементарного мотива изображение конечной повторяющейся фигуры;

4) средствами имеющегося компьютера и программного обеспечения размножить получившийся «шаблон» (фигурку), циклически меняя координаты в одном или двух направлениях.

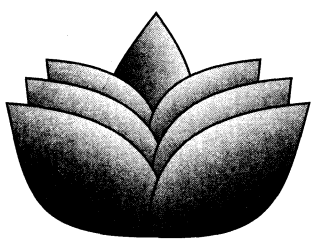
Упражнение № 3. Графический редактор CorelDraw. Рисование лотоса.

Ознакомиться с панелью инструментов ***Графика***. Можно использовать встроенную справку CorelDraw — знак **?** в верхнем меню.

1. Создать прямую: выбрать инструмент ***Кривая*** на панели инструментов ***Графика***, перевести курсор на рабочий лист, нажать кнопку мыши, отпустить, протянуть курсор мыши в нужном направлении и нажать еще раз курсор мыши.

2. Выбрать инструмент ***Форма***. На концах созданной прямой появятся узлы. Выделить их. Для этого можно воспользоваться одним из способов:

- нажать курсор мыши левее и выше выделяемого объекта, протянуть вниз и вправо до тех пор, пока весь объект не окажется внутри пунктирного прямоугольника, узлы на объекте приобретут черный цвет;
- щелкать поочередно на всех узлах, удерживая нажатой клавишу **Shift**.



Такой же принцип выделения и для объектов, только здесь используется инструмент ***Выделение***.

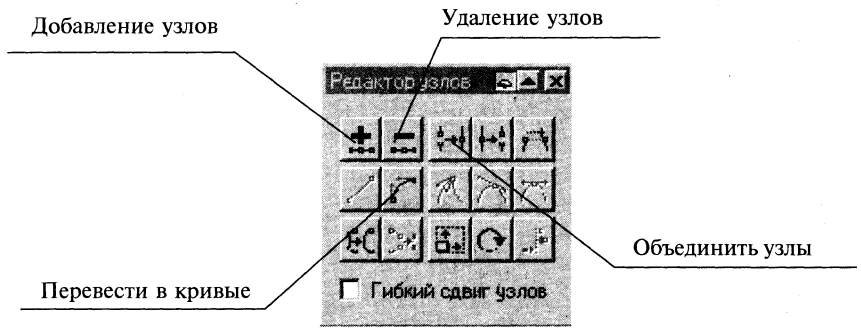
3. Выполнить двойной щелчок на инструменте ***Форма*** для вызова окна ***Редактор узлов***.

4. Добавить еще три узла на прямой. При выделенных узлах нажать на кнопку ***Добавление_узлов***.

5. Снять выделение узлов: щелкнуть кнопкой мыши в любом другом месте.

6. Хватая и перетаскивая поочередно узлы, сформировать четырехугольник.

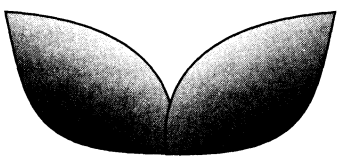
7. Выделить два замыкающих узла и выполнить команду ***Объединить_узлы***.



8. Выделить все узлы и выполнить команду ***Перевести_в_кривые***.

9. Удалить у прямоугольника два узла:

10. Выделить оставшиеся узлы. После перевода в кривые на каждом узле должны появиться рычаги. Двигая их в разные стороны, можно придавать форму линии. Создайте лепесток при помощи рычагов.

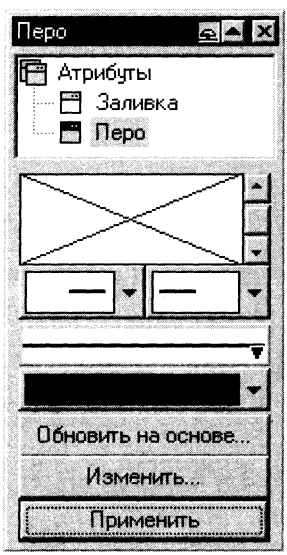


11. Выполнить градиентную заливку: выполнить двойной щелчок на инструменте ***Заливка***. Из появившейся дополнительной панели инструментов выбрать ***Градиентная заливка***, например, от красного к белому, радиальная со сдвигом по горизонтали и вертикали.

12. Сделать дубликат лепестка:

- выделить объект: выбрать инструмент ***Выделение*** и просто щелкнуть лепесток;
- выполнить ***Правка-Копировать***;
- выполнить ***Правка-Вставить***.

Также для создания копии можно нажать комбинацию клавиш **Ctrl+D**.



13. Зеркально отразить копию лепестка: выделить объект, ухватить мышью правый средний квадратик выделения и при нажатой клавише **Ctrl** протянуть влево. Можно также для создания зеркального отражения использовать соответствующую команду верхнего меню.

14. Сделать еще одну копию лепестка и оставить ее временно на свободном месте. Из него впоследствии будем формировать верхний лепесток лотоса.

15. Соединить лепестки и сгруппировать их: выделить оба объекта, выполнить команду ***Монтаж\ Сгруппировать***.

16. Сделать два раза дубликат полученного объекта, уменьшить их и оформить цветок. Для уменьше-

ния объекта выделить его. Используя значки выделения (квадратики), уменьшить объект. Для пропорционального уменьшения со всех сторон при изменении размеров удерживать нажатой клавишу **Shift**.

17. Для правильной последовательности лепестков использовать пиктограммы ***На_задний план***, ***На_передний план*** или соответствующую команду в верхнем меню.

18. Оформить верхний лепесток при помощи рычагов инструмента ***Форма***. Для того чтобы при оформлении не мешала заливка, использовать пиктограмму ***Каркас***: выделить лепесток и нажать на пиктограмму. Затем отключить его таким же образом.

19. Выделить все объекты, сгруппировать. Оставить объект выделенным.

20. Толщину линий у лепестков сделать невидимой: выполнить двойной щелчок на инструменте ***Перо***, в появившейся панели инструментов выбрать ***свиток_Перо***. Установить нулевую толщину и нажать кнопку ***Применить***.

Упражнение № 4. AutoCAD. Формирование параметров нового чертежа. Построение графических примитивов

1. Выберите ветвь 1 главного меню.
2. Создайте файл нового чертежа, используя в качестве имени *labor1*.
3. Войдите в режим **SETTINGS** в экранном меню режимов.
4. Командой **LIMITS** установите границы чертежа, характеризующиеся угловыми точками (0,0) и (12,9).
5. Наберите команду **STATUS** в командной строке и изучите параметры чертежа, описываемые этой командой.
6. Для возвращения в графический экран нажмите **F1**.
7. Наберите команду **UNITS**. Изучите формат линейных величин, угловой формат.
8. Используя команду **SNAP**, задайте сетку фиксации с интервалом 1.0. Убедитесь в перемещении курсора строго по углам сетки.
9. Выберите команду **AXIS**. Включите отображение осей.
10. Выберите команду **GRID**. Установите параметры сетки экрана, равные 0.5; 2.5; 5.0.
11. Вернитесь в меню режимов **AutoCAD** и войдите в режим **DRAW**.
12. Выберите команду **LINE** из экранного меню. Попробуйте вводить координаты точек линии из командной строки, например, 3.3, 5.5, 8.3 и т.д., затем продолжайте построение линий с помощью перекрестия экрана. Примените опцию CLOSE для построения замкнутого многоугольника.
13. Вызовите команду **CIRCLE**. Нарисуйте окружность, указав ее центр и радиус. Используйте режим отслеживания **DRAG**.

Упражнение № 5. AutoCAD. Синтез изображения из графических примитивов и редактирование

1. Начните новый чертеж.
2. Постройте графические примитивы: дуги, линии, полилинии, кольца. Используйте команды **ARC**, **LINE**, **PLINE**, **CIRCLE** в режиме **DRAW**.
3. Напишите какой-нибудь текст командами **TEXT** или **DTEXT**.
4. Для отработки команд редактирования рисунка вернитесь в главное экранное меню и задайте режим **EDIT**. Вызовите команду **MOVE**. На первый запрос укажите центр окружности, а на второй (second point) укажите точку в другой части экрана.

5. Вызовите команду **COPY**. Выберите изображение (укажите на объект и нажмите **Enter**) и скопируйте его в другую часть экрана. Попробуйте режим мультиплицирования копий, набрав **M** на вопрос о базовой точке и указав несколько точек для вставки копий. Для окончания режима дайте пустой ответ.

6. Вызовите команду **ROTATE** и поверните выбранный объект на 45°, 21°, 27°.

7. С помощью команды **ERASE** удалите все объекты, кроме двух.

8. Зеркально отобразите один из выбранных объектов относительно оси симметрии, заданной под любым углом к осям координат, с помощью команды **MIRROR**.

9. Постройте квадрат. Вызовите команду **ARRAY** и, пользуясь опциями «**P**» и «**R**», создайте круглую и прямоугольную (3×2) матрицы.

10. Сотрите изображения.

Упражнение № 6. AutoCAD. Уровни чертежа

Построение дорожного знака, укрепленного на столбе.

1. Выберите ветвь 1 главного меню. Создайте новый файл чертежа. Войдите в графический редактор системы.

2. Задайте значение шага сетки фиксации 0,5 (команда **SNAP**).

3. Создайте два новых слоя ZNAK и STOLB разных цветов и разных типов линий (команда **LAYER**).

4. Запросите список заданных слоев и сделайте слой ZNAK текущим.

5. С помощью команды **POLYGON** задайте два четырехугольника, вписанных в окружности диаметром 4 и 3 соответственно, с центром в точке (5, 6).

6. Измените тип линии для слоя STOLB на пунктирную (**DASHED**) и сделайте его текущим слоем.

7. С помощью команды **LINE** начертите невидимую часть столба. Обновите изображение на экране. В случае необходимости используйте команду **LTSCALE**.

8. Сделайте слой 0 текущим слоем.

9. Начертите видимую часть столба (**LINE**).

Упражнение № 7. AutoCAD. Размеры и штриховка

1. Начните новый чертеж.

2. Задайте командой **SNAP** разрешающую способность 0,01.

3. С помощью команды **UNITS** установите формат представления числовых величин с двумя значащими разрядами после десятичной точки.

4. Установите требующиеся значения системных переменных «размеривания» командой **DIM**.

5. Начертите прямоугольник, треугольник, ломаную, две окружности.

6. Нанесите вертикальные, горизонтальные, параллельные, угловые размеры, а также размеры диаметра и радиуса окружностей (**DIM**).

7. Начните новый чертеж.

8. Изобразите правильный треугольник, вписанный в квадрат, который, в свою очередь, вписан в окружность.

9. Размножьте изображение до трех с помощью команды **COPY**.

10. Запишите одно из изображений как блок (команды **BLOCK**, **KONTUR**, восстановление **OOPS**).

11. С помощью команды **HATCH** заштрихуйте 1-е изображение штриховкой обычного типа (**N**), 2-е изображение — «внешней» штриховкой, 3-е — «игнорирующей». Примените команду **COLOR** для получения цветной штриховки.

Лабораторные работы

Лабораторная работа № 1
Графический редактор PaintBrush

Время выполнения 8 часов.

Задание. Создайте с помощью PaintBrush компьютерный вернисаж.

Тематика выбирается каждым студентом самостоятельно и согласовывается с преподавателем.

Лабораторная работа № 2
Графический редактор CorelDraw

Задание. Нарисуйте на экране с помощью **CorelDraw** картинку «Мой город, моя улица, мой дом».

Тематика выбирается каждым студентом самостоятельно и согласовывается с преподавателем.

Лабораторная работа № 3
Система автоматизированного проектирования *AutoCAD*

Время выполнения 6 часов.

Задание. Изобразите чертеж объекта с размерами.

Варианты заданий

№ пп.	Объект	№ пп.	Объект	№ пп.	Объект
1	Стол	9	Петля	17	Пружина
2	Стул	10	Тумба	18	Полка
3	Болт	11	Гайка	19	Замок
4	Оконная рама	12	Шуруп	20	Лестница
5	Гантели	13	Гвоздь	21	Компьютер
6	Штанга	14	Скоба	22	Принтер
7	Колесо	15	Коленвал	23	«Мышь»
8	Дверь	16	Шарнир	24	Дискета

Дополнительная литература

1. *Ермолович Е.А., Макарова С.В., Хегай Л.Б.* Операционные системы и информационные технологии. — Красноярск, 2000.

2. *Ефимова О., Морозов В., Шафрин Ю.* Курс компьютерной технологии. — М.: ФБФ, 1998.

3. *Ефимова О., Морозов В., Шафрин Ю.* Практикум по компьютерной технологии. — М.: ФБФ, 1998.

4. *Зенкин А.А.* Когнитивная компьютерная графика. — М.: Наука, 1991.
5. *Котов Ю.В., Павлова А.А.* Основы машинной графики. — М.: Просвещение, 1993.
6. *Корриган Д.* Компьютерная графика: Секреты и решения: Пер. с англ.. — М.: Энтроп, 1995.
7. *Макарова Н.В. и др.* Информатика. Практикум по технологии работы на компьютере. — М.: Финансы и статистика, 1998.
8. *Олтман Р.* CorelDRAW! Для профессионалов: Пер. с англ. — М.: Энтроп, Март; Киев: Век, 1996.
9. Форматы графических файлов / Сост. А.С.Климов. — СПб.: НИПФ «ДиаСофт Лтд.», 1995.
10. *Шикин Е.В.* Начала компьютерной графики. — М.: Диалог-МИФИ, 1994.
11. *Шульте М.* CorelDRAW! 5.0. Все программы пакета: Пер. с англ. — СПб.: BVH — Санкт-Петербург, 1995.

§ 6. БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Рекомендации по проведению занятий

В начале занятий преподаватель знакомит студентов с основными видами баз данных, функциями и возможностями СУБД (1—2 семинарских занятия); на примере демонстрационной базы данных «Борей», встроенной в СУБД Access, объясняет суть выполняемых упражнений.

На последующих занятиях студенты выполняют лабораторные работы и сохраняют свои результаты в отдельных папках для последующей защиты перед преподавателем.

Краткие сведения

Для выполнения практической работы можно использовать любую систему управления базами данных (СУБД), имеющуюся в наличии на ваших компьютерах. В силу большой популярности и распространенности в вузах пакета MS Office ниже рассматривается СУБД *Microsoft Access.*

MS Access — это функционально полная реляционная СУБД. Запустить систему можно несколькими способами:

- запуск с помощью главного меню в WINDOWS'95;
- запуск с помощью Проводника;
- запуск с помощью ярлыка;
- и др.

Основные элементы главного окна Access, о которых необходимо иметь представление, таковы.

Строка заголовка. В строке заголовка отображается имя активной в данный момент программы.

Пиктограмма системного меню. Такая кнопка имеется в верхнем левом углу главного окна практически любого приложения. После щелчка на этой пиктограмме появляется меню, которое позволяет перемещать, разворачивать, сворачивать или

закрывать окно текущего приложения и изменять его размеры. При двойном щелчке на пиктограмме системного меню работа приложения завершается.

Полоса меню. Полоса меню содержит названия нескольких подменю. Когда активизируется любое из этих названий, на экране появляется соответствующее подменю. Перечень подменю на полосе Access и их содержание изменяются в зависимости от режима работы системы.

Панель инструментов. Панель инструментов — это группа пиктограмм, расположенная непосредственно под полосой меню. Главное ее назначение — ускоренный вызов команд меню. Кнопки панели инструментов тоже могут изменяться в зависимости от выполняемых операций. Также можно отобразить, спрятать, создать новую панель инструментов или настроить любую панель инструментов.

Строка состояния. В левой части строки состояния отображается информация о том, что вы делаете в настоящее время.

Окно базы данных. Это окно появляется при открытой базе данных. В поле окна сосредоточены все «рычаги управления» базой данных. Окно базы данных используется для открытия объектов, содержащихся в базе данных, таких, как таблицы, запросы, отчеты, формы, макросы и модули. Кроме того, в строке заголовка окна базы данных всегда отображается имя открытой базы данных.

Вкладки объектов. С помощью вкладок можно выбрать тип нужного объекта (таблицу, запрос, отчет, форму, макрос и модуль). Необходимо сказать, что при открытии окна базы данных всегда активизируется вкладка-таблица и выводится список доступных таблиц базы данных. Для выбора вкладки других объектов базы данных надо щелкнуть по ней мышью.

Кнопки. Кнопки, расположенные вдоль правого края окна базы данных, используются для работы с текущим объектом базы данных. Они позволяют создавать, открывать или изменять объекты базы данных.

Таблицы. Объект, который определяется и используется для хранения данных. Каждая таблица включает информацию в виде записей. Таблица содержит поля (столбцы) и записи (строки). Работать с таблицей можно в двух основных режимах:

1. *Режим конструктора.* Задание структуры таблицы, т.е. определяются типы, свойства полей, их число и названия. В этом режиме каждая строка верхней панели окна соответствует одному из полей определяемой таблицы.

2. *Режим таблицы.* Используется для просмотра, добавления, изменения, простейшей сортировки или удаления данных.

Форма. Объект, в основном предназначенный для удобного ввода данных. Форма — это формат (бланк) показа данных на экране компьютера. В форму могут быть внедрены рисунки, диаграммы, аудио- (звук) и видеоизображения.

Режимы работы с формой:

1. *Режим форм.* Используется для просмотра и редактирования данных. Режим форм предоставляет дружественную среду для работы с данными и удобный дизайн их представления на экране.

2. *Режим конструктора форм.* Используется для изменения структуры или шаблона формы.

3. *Режим таблицы.* Режим позволяет увидеть таблицу, включающую все поля форм; чтобы переключиться в этот режим при работе с формой, надо нажать кнопку таблицы на панели инструментов.

Отчет. Объект, предназначенный для создания документа, который впоследствии может быть распечатан или включен в документ другого приложения. Отчеты, как и формы, могут создаваться на основе запросов и таблиц.

Режимы работы с отчетом:

1. *Режим предварительного просмотра.* Этот режим позволяет увидеть отчет в таком виде, в каком он будет воплощен при печати.

2. *Режим конструктора.* Данный режим предназначен для изменения шаблона (структуры) отчета.

Макрос. Объект, представляющий структурированное описание одного или нескольких действий, которые должен выполнить ACCESS в ответ на определенное событие. Например, можно определить макрос, который в ответ на выбор некоторого элемента в основной форме открывает другую форму. В макросы включаются *макрокоманды.* В MS Access имеется свыше 40 макрокоманд. Макрокоманды выполняют такие действия, как открытие таблиц и форм, выполнение запросов, запуск других макросов, выбор опций из меню, изменение размеров открытых окон и т. п.

Запрос. Это объект, который позволяет пользователю получить нужные данные из одной или нескольких таблиц. Можно создать запросы на выбор, обновление, удаление или на добавление данных. С помощью запросов можно создавать новые таблицы, используя данные одной или нескольких таблиц, которые уже существуют.

Таблицы в базе данных могут быть связаны. В Access используется три типа межтабличных связей:

- один к одному — каждой записи первой таблицы соответствует одна запись из второй связанной таблицы (например, есть две таблицы: *Студент* с полями *Номер, Фамилия, Имя, Дата рождения, Место рождения, Год_поступления_в_вуз* и таблица *Сессия* с полями *Номер, Оценка1, Оценка2, Оценка3*; каждому номеру таблицы *Студент* соответствует один номер в таблице *Сессия*);
- один ко многим — любая запись в первой таблице может быть связана с несколькими записями во второй таблице (например, есть две таблицы: таблица *Студент* с полями *Номер, Фамилия, Имя, Факультет, Группа* и таблица *Библиотека* с полями *Код, Название_книги, Автор, Год_издания*; каждому номеру из таблицы *Студент* может соответствовать несколько кодов из второй таблицы);
- многие ко многим — любая запись в первой таблице может быть связана с несколькими записями во второй таблице и обратно — каждая запись второй таблицы связана с несколькими записями первой таблицы (например, есть две таблицы: таблица *Студент* с полями *Номер, Фамилия, Имя, Факультет, Группа* и таблица *Спортивные_секции* с полями *Код, Вид спорта*, каждый студент может посещать несколько секций, и каждую секцию могут посещать несколько студентов);

Завершив работу с Access (или с ее приложением), надо корректно закончить сеанс. Безопасно выйти из Access можно несколькими способами:

- двойным щелчком мыши на пиктограмме системного меню в строке заголовка главного окна Access;
- из меню Access выбором пункта ***Файл\Выход***;
- нажатием комбинации клавиш **<ALT +F1>**.

Контрольные вопросы

1. Назовите основные элементы окна СУБД Access.
2. Перечислите основные *объекты* окна базы данных.
3. Какие *режимы работы* используются для работы с таблицей, формой, отчетом?
4. Для чего нужен *запрос*?

Темы для рефератов

1. Информационная система (база данных) «Борей».
2. Информационные справочные системы в человеческом обществе.
3. Информационные поисковые системы в человеческом обществе.
4. Базы данных и Интернет.
5. Геоинформационные системы.
6. Проектирование и программирование баз данных.
7. СУБД Oracle.
8. Информационная система «Галактика».
9. Информационная система «Консультант плюс».
10. Информационная система «Гарант плюс».

Темы семинарских занятий

1. Назначение и функции СУБД. Объекты СУБД Access, их назначение. Режимы работы основных *объектов*.
2. Создание БД, установление *связей* в БД.
3. Составление *форм, запросов* и *отчетов* в *режиме* конструктора, при помощи *мастера*.

Рекомендации по программному обеспечению

Для выполнения лабораторно-практических занятий рекомендуется одна из следующих программ:

1. Dbase – 3, 4, 5;
2. FoxPro;
3. Paradox;
4. Clliper;
5. Access;
6. Oracle.

Задачи и упражнения

Создание БД, ввод и редактирование данных

Упражнение № 1. Формирование структуры таблицы

Создать новую базу данных — сведения о студентах вашего потока. Для этого выполнить следующие действия:

1. Запустить Access.
2. При запуске появится диалоговое окно, в котором надо выбрать строку *Новая база данных.*

Примечание. Если Вы в пункте 2 отказались от диалогового окна, то далее выполните следующие действия для создания новой базы данных:

• нажмите кнопку ***Создать*** на панели инструментов или воспользуйтесь пунктом меню ***Файл\Создать***;

• в появившемся диалоговом окне *Создание* выберите вкладку *Общие*;
• щелкните по пиктограмме ***Новая_база_данных*** и подтвердите выбор.

3. В окне *Файл новой базы данных* указать имя новой БД (например, «Деканат») в поле ввода *Имя файла* и сохранить в нужной папке. Нажать кнопку ***Создать***.

4. В появившемся окне *База данных* активизировать вкладку ***Таблицы*** и щелкнуть по кнопке ***Создать***.

5. Создать таблицу, воспользовавшись *Конструктором*. В окне *Новая таблица* выбрать пункт ***Конструктор*** и подтвердить выбор.

6. Определить поля таблицы

Поле	Тип поля	Размер поля
Номер	Текстовое	5
Фамилия	Текстовое	15
Имя	Текстовое	10
Отчество	Текстовое	15
Дата рождения	Дата	Краткий формат
Группа	Текстовое	3
Дом. адрес	Текстовое	20

В появившемся окне создать поля базы данных, согласно следующей таблице. Для ввода типа поля использовать значок всплывающего меню, который появляется при установке курсора в столбец *Тип данных*.

7. Определить первичный ключ для таблицы. В данной таблице ключевым является поле *Номер*. Чтобы сделать поле ключевым, выделить его и выбрать меню ***Правка\Ключевое_поле*** или нажать кнопку ***Ключевое_поле*** на панели инструментов. При этом слева от имени ключевого поля таблицы появится изображение ключа.

8. Закрыть заполненную таблицу.

9. Сохранить ее под именем *ФИЗИКИ*.

Упражнение № 2. Ввод и редактирование данных

1. В окне *База данных* появилось имя сохраненной таблицы. Для того чтобы вводить данные, надо открыть ее в режиме таблицы. Щелкнуть на кнопку ***Открыть***. (Если Вам необходимо внести изменения в структуру созданной таблицы, необходимо нажать на кнопку ***Конструктор***.)

2. Занести в таблицу 6 — 7 записей. Для поля *Группа* использовать номера 56, 57, 58.

3. Отредактировать введенные в таблицу данные: заменить во второй записи фамилию.

4. В поле *Дата рождения* изменить в первой записи год рождения.

5. Удалить последнюю запись в таблице. Для этого нужно выделить ее: установить курсор мыши к левой границе таблицы до изменения его в виде стрелки, направленной вправо, щелкнуть мышью и нажать клавишу **Delete**.

6. Добавить еще две записи.

7. Сохранить таблицу и закрыть ее.

Упражнение № 3. Разработка однотабличных пользовательских форм

Данные в таблицу БД удобнее вводить, если воспользоваться экраном в виде некоторого бланка (формы). Такой способ позволяет видеть на экране все данные одной записи.

Создать однотабличную пользовательскую форму для ввода и редактирования данных в ранее созданную таблицу. Для этого выполнить следующее:

1. В окне ***База_данных*** активизировать вкладку ***Форма***.
2. В том же окне нажать кнопку ***Создать***.
3. В диалоговом окне ***Новая_форма*** выбрать строку ***Мастер*** и выбрать в качестве источника данных имя таблицы *ФИЗИКИ*, подтвердить выбор.
4. В появившемся окне выбрать поля для создаваемой формы (выберем все имеющиеся).
5. Дальнейшие действия выполнить самостоятельно (стиль формы выбрать — *Обычная*).
6. Добавить в таблицу 1—2 записи в режиме формы.
7. Познакомиться с возможностями перемещения в таблице, представленной в виде формы (переместиться на следующую запись и обратно, к первой записи, к последней записи, новая запись).
8. Закрыть окно формы.
9. Открыть таблицу *ФИЗИКИ* и просмотреть добавленные записи в таблице.
10. Закрыть таблицу, выйти в окно ***База_данных***.

Упражнение № 4. Разработка отчета

1. В окне ***База_данных*** активизировать вкладку ***Отчеты*** и щелкнуть кнопку ***Создать***.
2. С помощью ***Мастера отчетов*** создать отчет для вывода сведений о студентах группы, выбрать для отчета следующие поля: *Номер*, *Фамилия*, *Имя*, *Дата рождения*. В качестве источника данных использовать таблицу *ФИЗИКИ*. При создании отчета использовать сортировку по полю *Фамилия*, вид отчета *Табличный*, стиль *Строгий*. Ввести имя отчета (по умолчанию Access вводит имя таблицы-источника).
3. Закрыть отчет и выйти в окно ***База_данных***.

Упражнение № 5. Поиск, сортировка и отбор данных

Поиск:

1. Открыть таблицу *ФИЗИКИ* в режиме ***Формы***.
2. Осуществить поиск какого-либо студента по полю *Фамилия*. Для этого выполнить следующее:

- установить курсор в строку поля, по которому будет осуществляться поиск (в нашем случае это поле *Фамилия*);
- выполнить команду ***Правка\Найти*** или нажать на пиктограмму ***Найти*** для вывода диалогового окна ***Поиск*** (если такая фамилия встречается несколько раз, использовать кнопку ***Найти_далее***).

3. Закрыть окно формы.

Сортировка:

4. Открыть таблицу *ФИЗИКИ*.
5. Отсортировать записи таблицы в алфавитном порядке по полю *Фамилия*. Для этого выполнить действия:

• установить курсор в поле ***Фамилия***;

• выполнить команду ***Записи\Сортировка\По_возрастанию*** или воспользоваться пиктограммой ***Сортировка_по_возрастанию***.

Фильтр:

6. Используя фильтр, вывести на экран список студентов группы 57. Для этого выполнить действия:

• выполнить команду ***Записи\Изменить_фильтр*** или использовать пиктограмму ***Изменить_фильтр***;

• установить курсор в поле *Группа*, нажать на значок всплывающего меню и выбрать номер нужной группы;

• выполнить команду ***Фильтр\Применить_фильтр*** или использовать соответствующую пиктограмму;

• чтобы убрать фильтр, воспользоваться командой ***Записи\Удалить_фильтр*** или применить ту же пиктограмму, только теперь ее назначение — ***Убрать_фильтр***;

• закрыть таблицу.

7. Открыть таблицу *ФИЗИКИ* в режиме ***Формы*** и выполнить тот же фильтр: отобрать студентов группы 57 мужского пола. Для этого выполнить действия:

• в поле *Формы* найти запись, которая содержит номер нужной группы (57);

• установить курсор в поле, по которому будет осуществляться фильтр (*Группа*);

• выполнить команду ***Записи\Фильтр\Фильтр_по_выделенному*** или воспользоваться соответствующей пиктограммой на панели инструментов;

• таким же образом выполнить выборку студентов мужского пола;

• удалить фильтр.

Упражнение № 6. Запросы

Создать запрос-выборку из таблицы *ФИЗИКИ*, содержащую сведения о студентах женского пола:

1. В окне ***База_данных*** активизировать вкладку ***Запрос*** и нажать кнопку ***Создать***.

2. Выбрать режим ***Конструктор*** и подтвердить выбор.

3. В появившемся окне ***Добавление_таблицы*** выделить имя таблицы, из которой будет производиться запрос (*ФИЗИКИ*) и выполнить команду ***Добавить***. Список полей этой таблицы должен появиться в окне ***Запрос_на_выборку***.

4. Закрыть окно ***Добавление_таблицы***.

5. В оставшемся окне ***Запрос_на_выборку*** щелкнуть верхнюю левую ячейку, относящуюся к заголовку ***Поле***.

6. В данной ячейке должен появиться значок всплывающего меню. Используя его, ввести в ячейку имя первого поля создаваемого запроса (например, *Фамилия*).

7. Аналогичным образом заполнить остальные ячейки первой строки (*Имя, Пол, Дата рождения, Группа*).

8. Установить сортировку по полю ***Фамилия***.

9. В строке *Условия отбора* внести в нужные поля критерии отбора: в поле ***Пол*** установить букву **ж**, в поле *Группа* — **57** (данные для условия отбора вносить в таком же виде, как они внесены в таблицу).

10. В строке ***Вывод_на_ экран*** значок «галочка» означает, что в результате выполнения запроса данное поле будет выводиться на экран.

11. Выполнить команду ***Запрос\Запуск*** или использовать соответствующую пиктограмму.

Многотабличная БД, установление связей между таблицами

Упражнение № 7. Создание многотабличной БД

1. Создать таблицы *СЕССИЯ* и *СТИПЕНДИЯ*, используя ту же технологию, что и при создании таблицы *ФИЗИКИ* в Упражнении 1. Атрибуты поля *Номер* таблицы *СЕССИЯ* должны быть такими же, как атрибуты этого же поля таблицы *ФИЗИКИ*. Состав полей и их свойства следующие:

СЕССИЯ

Признак ключа	Поле	Тип поля	Размер поля
Ключ	Номер	Текстовое	5
	Оценка 1	Числовое	Фиксированный
	Оценка 2	Числовое	Фиксированный
	Оценка 3	Числовое	Фиксированный
	Оценка 4	Числовое	Фиксированный
	Результат	Текстовое	3

СТИПЕНДИЯ

Признак ключа	Поле	Тип поля	Размер поля
Ключ	Результат	Текстовое	3
	Процент	Числовое	Процентный

2. Заполнить таблицы данными; оценки в записи ввести на свое усмотрение так, чтобы в записях присутствовали разные комбинации оценок из четырех групп:

Неуд.	Хор.	Хор.1	Отл.
За удовл. и неудовл.	За две 4 и более	5 5 5 4	5 5 5 5

3. В поле *Результат* данные заносить в соответствии с представленной таблицей, например, если в записи три оценки 5 и одна оценка 4, то в результат занести хор. 1.

4. Поле *Процент* заполнить в соответствии со следующей таблицей:

Результат	Процент
Неуд.	0,00%
Хор.	100,00%
Хор.1	200,00%
Отл.	300,00%

5. Сохранить обе таблицы и закрыть их.

Упражнение № 8. Установление связей между таблицами

1. В окне ***База_данных*** *Деканат* должны быть имена трех таблиц: *ФИЗИКИ, СЕССИЯ, СТИПЕНДИЯ*. Для установления связей выполнить команду ***Сервис\ Схема_данных***.

2. В появившемся окне ***Схема_данных*** выполнить добавление всех трех таблиц в схему.

3. Установить связи между таблицами *ФИЗИКИ* и *СЕССИЯ*. Для этого протащить указатель мыши от поля *Номер таблицы ФИЗИКИ* к полю *Номер таблицы СЕССИЯ* при нажатой клавише мыши.

4. В появившемся диалоговом окне ***Связи*** активизировать значок ***Обеспечение_ целостности_данных***, отношение «Один к одному», активизировать значки ***Каскадное_обновление_связанных_полей*** и ***Каскадное удаление связанных полей***. Прочитать встроенную справку об этих значках (щелкнуть на знак **?** в заголовке окна ***Связи***, подвести к нужному значку и нажать кнопку мыши). Нажать кнопку ***Создать***.

5. Установить связь между таблицами *СТИПЕНДИЯ* и *СЕССИЯ*. Для этого протащить указатель мыши от поля *Результат* таблицы *СТИПЕНДИЯ* к полю *Результат* таблицы *СЕССИЯ*. Здесь отношение «Один ко многим».

6. Закрыть окно ***Схема_данных***, при выходе сохранить связи.

Упражнение № 9. Разработка многотабличной пользовательской формы ввода данных

1. Создать форму на основе таблицы *СЕССИЯ* с использованием *Мастера*, включив в форму все поля таблицы. При выборе внешнего вида формы использовать расположение *В_один_столбец*. Дать имя форме *СЕССИЯ*.

2. Создать форму на основе таблицы *ФИЗИКИ* с использованием *Мастера*, включив в нее все поля, кроме поля *Номер*. При выборе внешнего вида формы использовать расположение *Табличный_вид*. Дать имя форме *СТУДЕНТ*.

3. Закрыть форму *СТУДЕНТ*. Форму *СЕССИЯ* открыть в режиме *Конструктора*. Для переключения между *режимом просмотра формы* и *режимом конструктора формы* можно использовать меню ***Вид*** или пиктограмму ***Вид*** на панели инструментов.

Для оформления атрибутов текста подчиненной таблицы или любого другого объекта формы (например, изменение цвета и начертания шрифта, оформление подчиненной формы — утопленное, приподнятое и т.д.) необходимо в *режиме конструктора* выполнить одно из действий:

- выделить этот объект и выбрать меню ***Вид\Свойства***, использовать пиктограмму на панели инструментов;
- выполнить двойной щелчок на этом объекте для открытия окна свойств.

4. Перенести из окна ***База_данных*** пиктограмму формы *СТУДЕНТ* в нижнюю часть поля формы *СЕССИЯ* и перейти в режим формы.

5. Просмотреть полученную составную форму. Пролистать записи до конца.

6. Добавить 2—3 записи, используя полученную составную форму: сначала заполнять поля из таблицы *ФИЗИКИ*, затем — поля из таблицы *СЕССИЯ*.

7. Закрыть форму.

Упражнение № 10. Формирование запросов для многотабличной базы данных

Построить *запрос*, позволяющий выводить фамилию, имя, отчество и номер группы студентов, которым может быть назначена стипендия, и размер стипендии в процентах. Информация для получения таких данных находится в трех таблицах *ФИЗИКИ, СЕССИЯ, СТИПЕНДИЯ.*

В данном случае создается новая таблица, содержащая сведения из разных взаимосвязанных таблиц.

1. В окне ***База_данных*** создать новый запрос на основе связанных таблиц. Для этого активизировать вкладку ***Запрос*** и нажать кнопку ***Создать***.

2. В появившемся окне ***Новый_запрос*** выбрать ***Простой_запрос*** (с использованием *Мастера запросов*) и подтвердить выбор.

3. В окне ***Создание_простых_запросов*** выбрать из таблицы *ФИЗИКИ* поля: *Фамилия, Имя, Отчество, Группа*; из таблицы *СТИПЕНДИЯ* — поле *Процент*. Закончить работу с *Мастером запросов* самостоятельно.

4. В полученной таблице в строке ***Условие отбора*** установить по полю *Процент* выражение >0, т.е. вывод тех студентов, у которых сессия сдана на положительные оценки. Для этого выполнить расширенный фильтр, упорядочить фамилии студентов в алфавитном порядке.

5. Дать имя запросу *Приказ* и закрыть его.

Задание:

1. Подготовить список студентов, сдавших сессию на «отлично».

2. Создать запрос, выводящий список студентов, имеющих хотя бы одну «тройку».

3. Создать запрос, выводящий список студентов, фамилия которых начинается на букву **А**.

Для выполнения задания предварительно прочитать справку Access: ввести в Предметном указателе в строке поиска запись ***like***. Вспомнить, как в Excel формируются условия, содержащие логические операторы *И, ИЛИ.*

Упражнение № 11. Разработка многотабличной формы отчета вывода данных

Создание отчета, в котором используется информация из различных таблиц базы данных.

Для создания отчета, включающего информацию из различных таблиц, используют предварительно сформированный запрос.

Построить отчет, сформированный на основе созданного запроса *Приказ.*

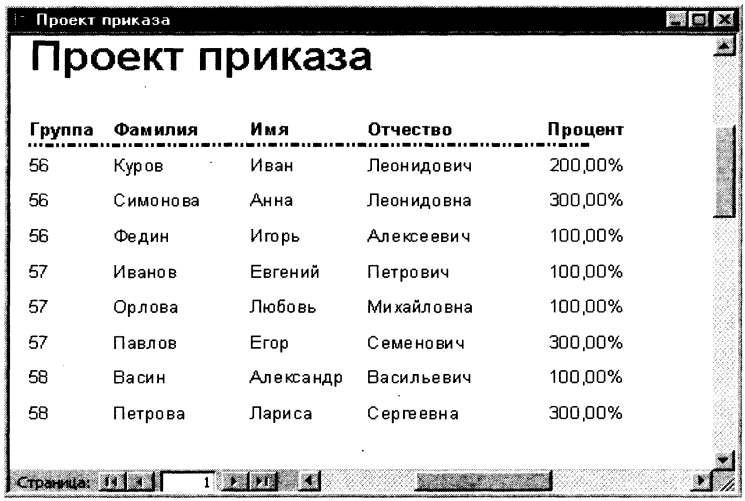
1. В окне ***База_данных*** активизировать вкладку ***Отчет*** и нажать кнопку ***Создать***.

2. В окне ***Новый_отчет*** выбрать *Мастер отчетов* и источник данных — запрос *Приказ.*

3. Дальнейшие действия работы с *Мастером* выполните самостоятельно: Тип представления данных — по таблице *ФИЗИКИ*, уровни группировки не задавать, осуществить сортировку по группам, внутри каждой группы упорядочить фамилии в алфавитном порядке, вид макета табличный.

4. Если необходимо сделать какие-либо изменения макета отчета, представьте его в режиме *Конструктора.* Для этого используйте меню ***Вид\Конструктор*** или пиктограмму ***Вид***.

5. В поле заголовков отчета вставьте дату. Меню ***Вставка\Дата***. Отчет должен иметь следующий вид:



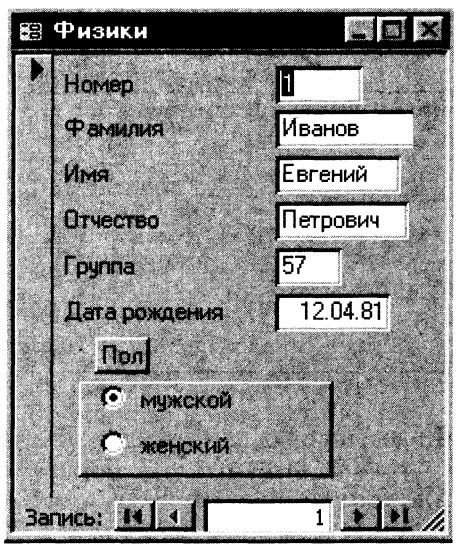
6. Просмотреть полученный отчет. Сохранить его.

Упражнение № 12. Создание элемента управления

Элемент управления — это графический объект в форме или отчете для представления данных или для выполнения определенных действий.

Создать элемент управления для ввода пола студентов в форме *ФИЗИКИ*, созданной в Упражнении № 3.

1. Создать в таблице *ФИЗИКИ*, созданной в Упражнении, новое поле под именем *Пол* в режиме *Конструктора*. В *Описании* данного поля ввести следующую запись: 1 — мужской, 2 — женский.



2. В окне ***База_данных*** перейти в режим *Формы*, открыть в режиме *Конструктора* форму *ФИЗИКИ*, созданную в Упражнении № 3.

3. Увеличить в высоту рабочее поле формы.

Элемент управления можно создать двумя способами:

- создание группы с помощью *Мастера*;
- создание группы без помощи *Мастера*.

Создадим переключатель пола вторым способом.

4. В *режиме конструктора* формы в дополнительной панели инструментов *Панель элементов* отключить кнопку ***Мастер_элементов*** (должна быть не выделена). Это приведет к отключению *Мастеров*.

5. Нажать кнопку ***Группа*** на панели элементов.

6. Открыть ***Список_полей***, используя данную кнопку на верхней панели инструментов, а затем перетащить поле *Пол* из списка полей в рабочее поле формы при нажатой кнопке мыши. В рабочем поле формы должна появиться рамка с заголовком *Пол* в верхней ее части.

7. Нажать на ***Панели_элементов*** кнопку ***Переключатель***, а затем выбрать внутри группы место, в которое помещается левый верхний угол флажка или выключателя. Подтвердить установку элемента щелчком мыши.

8. Установить курсор на появившийся текст (Переключатель) и ввести подпись *Мужской*.

9. Аналогичным образом добавить элемент для женского пола.

10. Выровнять значки: сначала выделить, далее использовать значки «палец» и «ладонь» для их перетаскивания. Самостоятельно разберитесь в различии назначения этих значков.

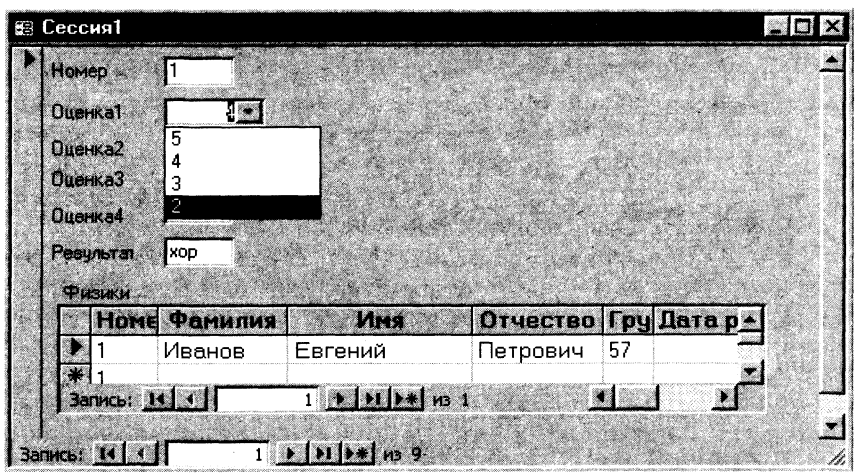
11. Отредактировать внешний вид созданной рамки, вызвав окно свойств (см. упражнение № 9 п. 3).

12. Перейти в *режим просмотра формы* и, используя элемент управления, установить пол в записях.

13. Перейти в режим таблицы и просмотреть поле *Пол*. В данном поле должны появиться цифры 1 или 2. При установлении курсора в данное поле в нижней области экрана в строке подсказки появится текст описания: 1 — мужской, 2 — женский.

14. Самостоятельно создайте элемент управления — поле со списком для ввода оценок, используя *Мастер_элементов*:

На рисунке представлен пример поля со списком для поля *Оценка1*.



При работе с мастером выбрать ***Фиксированный набор значений***, сохранить в поле *Оценка1*, задать подпись *Оценка1*.

Упражнение № 13. Создание вычисляемых полей в *Отчете*

Создать *Отчет* на основе таблицы *СЕССИЯ*. В отчете, используя *Построитель выражений*, создать новое поле — средний балл сдачи сессии каждым студентом.

1. Создать запрос, включив в него из таблицы *ФИЗИКИ* поля *Номер, Фамилия, Имя*, из таблицы *СЕССИЯ* — поля *Оценка 1, Оценка 2, Оценка 3, Оценка 4*. Дать имя отчету *Результаты*.

2. В окне диалога ***Создание_отчета*** выбрать таблицу *Результаты* и нажать кнопку ***Конструктор***.

3. Перетащить мышью все поля из окна списка полей таблицы *Результаты* в область данных отчета. Можно перенести сразу несколько полей. Для этого в списке полей при выборе полей удерживать нажатой клавишу **Shift**.

4. Для отчета нужно создать новое поле, в котором будет размещаться результат вычисления среднего балла для каждого студента. Для начала создать новое пустое поле. Для этого на ***Панели_элементов*** выбрать инструмент ***Поле*** и щелкнуть в области данных отчета, где будет размещаться поле *Средний балл* студента.

Появится элемент, состоящий из имени поля (Поле...) и содержимого.

5. Установить текстовый курсор на имя поля и ввести новое имя *Средний балл.*

6. Отредактировать размеры зоны имени поля (размеры по вертикали и горизонтали), используя значки квадратиков, которые появляются при выделении данного поля.

7. Для размещения формулы, вычисляющей средний балл, выделить область *Содержимого* и выполнить команду ***Вид\Свойства*** или использовать контекстное меню.

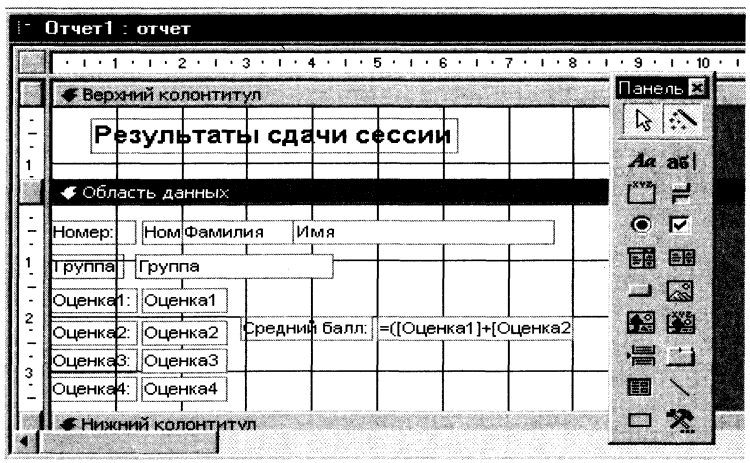
8. Установить курсор в поле ***Данное*** и нажать значок с тремя точками.

9. В поле ввода построителя выражений при помощи имеющихся кнопок и перечня полей составить следующее выражение:

$$=([\text{Оценка1}] + [\text{Оценка2}] + [\text{Оценка3}] + [\text{Оценка4}])/4$$

Для ввода имени имеющегося поля нужно выделить имя поля в списке и нажать кнопку **Добавить** или дважды щелкнуть по имени поля в списке.

10. Подтвердить введенное выражение: нажать клавишу ***OK***.



11. Перейти в режим просмотра отчета: ***Файл\Предварительный_просмотр*** или нажать кнопку на панели инструментов.

Пример отчета:

Номер:	1	*Иванов Петр*	
Группа:	56		
Оценка1:	5		
Оценка2:	5	Средний балл:	4,5
Оценка3:	4		
Оценка4:	4		

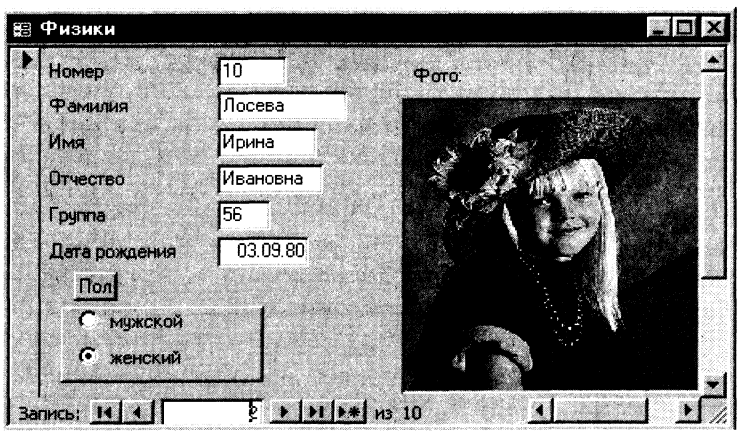
12. Закрыть отчет.

Упражнение № 14. Вставка графических объектов в БД

Графический объект можно вставлять (или осуществлять с ним связь) в форму или отчет как свободный рисунок. Также графический объект может быть вставлен в форму как объект типа OLE.

Вставить картинки людей из коллекции Clipart студентов в таблицу *Студент* БД *Деканат*. Если есть какие-либо фотоизображения людей, можно вставлять их, открыв в соответствующем приложении.

1. Добавить в таблицу *ФИЗИКИ* поле *Фото*, тип *поле объекта OLE*.
2. Запустить Графический редактор ***Paint*** через ***Главное_меню\Стандартные***.
3. Открыть любую картинку, выделить весь рисунок при помощи команды ***Правка\Выделить_все*** и занести в буфер при помощи команды ***Правка\Копировать***.
4. Перейти в Access и установить курсор на первую запись поля *Фото*.
5. Выполнить команду ***Правка\Специальная_вставка***.
6. В появившемся окне изучить при помощи значка **?** варианты команды ***Как***.
7. Выбрать тип Picture (Рисунок). В поле *Фотография* появится запись (Рисунок или Картинка...).
8. Аналогичным образом вставить еще 2—3 рисунка.
9. Закрыть таблицу.
10. Открыть форму *ФИЗИКИ Конструктора*.
11. Из пиктограммы ***Список_полей*** добавить поле *Фото*.
12. Отредактировать его местоположение.
13. Перейти в режим формы и просмотреть записи. В них должны присутствовать изображения.
14. В окне свойств поля *Фото* в строке *Установка размеров* выбрать *Вписать_в_рамку* или *По_размеру_рамки*.



Лабораторные работы

Лабораторная работа № 1
Создание БД, ввод и редактирование данных

Время выполнения 6 часов.

Задание. Создайте базу данных «Студенческая группа».

Задание общее для всех студентов. Создайте структуру базы данных с полями *ФИО*, *курс*, *номер группы*, *номер зачетной книжки*, *возраст*, *адрес жительства*. Введите не менее 10 записей. Удалите выборочно две записи, а затем добавьте 4 новые записи.

Лабораторная работа № 2

Многотабличная БД, установление связей между таблицами

Время выполнения 6 часов.

Задание. Создайте базу данных «Музыкальный альбом» из двух таблиц. Установите связи между ними и проведите операции, подобные упражнению 1.

Задание общее для всех студентов. Создайте структуру двух таблиц *КОМПОЗИТОР*, *ПЕВЦЫ*. Самостоятельно придумайте поля этих двух таблиц, выделите ключевые поля, установите связи между таблицами. Заполните данными и проведите редактирование записей таблиц.

Лабораторная работа № 3
Управление, вычисляемые поля, запросы по образцу, графика

Время выполнения 8 часов.

Задание. Разработайте информационно-справочную систему, содержащую несколько таблиц, входные формы, запросы и отчеты.

Варианты заданий

Вариант 1. Городской телефонный справочник.
Вариант 2. Каталог программного обеспечения персонального компьютера.
Вариант 3. Электронный алфавитный каталог библиотеки.
Вариант 4. Электронный систематический каталог библиотеки.
Вариант 5. Электронный алфавитно-систематический каталог домашней библиотеки.
Вариант 6. Система «Деканат».
Вариант 7. Система «Учебный план факультета информатики».
Вариант 8. Система «Научные труды и методические разработки кафедры».
Вариант 9. Система «Расписание занятий» (модель).
Вариант 10. Система «Выпускники факультета математики и информатики».
Вариант 11. Система «Биржа труда».
Вариант 12. Система «Тестовые задания по школьному курсу информатики».
Вариант 13. Система «Участники конференции».
Вариант 14. Система «Кто есть кто: выдающиеся информатики России».
Вариант 15. Система «Календарь проводимых мероприятий по информатике».
Вариант 16. Система «Телеконференции по информатике и информационным технологиям».
Вариант 17. Система «Белые страницы Интернет».
Вариант 18. Система «Желтые страницы Интернет».
Вариант 19. Система «Музыкальный альбом».
Вариант 20. Система «Инвентарная книга факультета».
Вариант 21. Система «Компьютерный салон».

Вариант 22. Система «Художественная галерея».

Вариант 23. Система «Склад товаров магазина "Детский мир"».

Вариант 24. Система «Склад продовольственных товаров мелкооптового магазина».

Вариант 25. Система «Учет товаров оптовой базы промышленных товаров».

Дополнительная литература

1. ACCESS 7.0 для WINDOWS'95. — Киев: Торгово-издательское бюро BHV, 1996.
2. *Бекаревич Ю. Б., Пушкина Н. В.* СУБД ACCESS для WINDOWS'95 в примерах. — СПб.: BHV — Санкт-Петербург, 1997.
3. *Богумирский Б.С.* Эффективная работа на IBM PC в среде Windows'95. — СПб.: Питер-Пресс, 1997.
4. *Вемпен Ф.* Microsoft Office Professional: 6 книг в одной: Пер. с англ. — М.: Бином, 1977.
5. *Джонс Э., Саттон Д.* Библия пользователя Office'97: Пер. с англ. — Киев: Диалектика, 1997.
6. *Ефимова О., Морозов В., Шафрин Ю.* Курс компьютерной технологии. — М.: ФБФ, 1998.
7. *Ефимова О., Морозов В., Шафрин Ю.* Практикум по компьютерной технологии. — М.: ФБФ, 1998.
8. *Каратыгин С., Тихонов А., Долголаптев В.* Базы данных: Простейшие средства обработки информации. Т. 1, 2. Серия «Компьютер для носорога». — М.: ФИА, 1995.
9. *Крамм Р.* Системы управления базами данных dBASE II и dBASE III для персональных компьютеров. — М.: Финансы и статистика, 1988.
10. *Лядова Л. Н.* Персональный компьютер: От начинающего пользователя до профессионала: В 2 т. — Пермь: ПГУ, 1998.
11. *Макарова Н.В. и др.* Информатика. Практикум по технологии работы на компьютере. — М.: Финансы и статистика, 1998.
12. *Назаров С. В., Першиков В. И. и др.* Компьютерные технологии обработки информации. — М.: Финансы и статистика, 1995.
13. *Смирнов С. С.* Программные средства персональных ЭВМ. — Л.: Машиностроение. Ленингр. отд-ние, 1990.

§ 7. ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

Рекомендации по проведению занятий

В начале занятий по данной теме рекомендуется ознакомить студентов с технологиями обработки числовых данных (вычисления, сводные ведомости, графическое представление числовых данных, функций). Познакомить с основными компонентами и возможностями электронных таблиц (семинарское занятие 1). На практических занятиях после выполнения набора упражнений распределить задания для самостоятельного выполнения.

Все результаты выполнения лабораторных работ студенты сохраняют в отдельной папке и сдают преподавателю индивидуально.

Краткие сведения

Электронные таблицы Excel

Электронная таблица — это прямоугольная матрица, состоящая из ячеек, каждая из которых имеет свой номер.

	A	B	C	D	E	F
1						
2						
3		Ячейка B3				
4						

Адрес ячейки определяется обычным координатным способом, например, ячейка B3, ячейка C5 и т.д.

В каждую из ячеек можно занести *число, формулу* (арифметическое выражение) или *текст*. Операндами формулы могут быть математические функции, константы, адреса ячеек; в последнем случае реальный операнд — содержимое ячейки с указанным адресом.

Тексты и числовые константы, занесенные в ЭТ, сами по себе в ходе работы таблицы никогда не изменятся. Что же касается формул, то программа по специальной команде производит вычисление их значений и отображает его на экране (в той же ячейке, куда занесена формула).

Данные, входящие в таблицы, можно автоматически обрабатывать в графическом виде. Их также можно подготовить и распечатать в удобном для пользователя виде, сохранять и использовать многократно.

Существует несколько программ такого класса. Ниже обсуждается одна из них, популярная в настоящее время программа Excel из пакета Microsoft Office, в которой реализованы многочисленные возможности и удобный и комфортный интерфейс.

Задаваемый обычным образом адрес ячейки называется *относительным адресом* или *относительной ссылкой*.

При некоторых операциях копирования, удаления, вставки электронная таблица автоматически изменяет адреса ячеек. Иногда возникает необходимость не менять адрес ячейки. В таких случаях используют *абсолютный адрес*. Абсолютный адрес ячейки создается с помощью знака доллара $.

Группа ячеек (*диапазон*) задается через двоеточие, например, B3:D4 (или B3..D4), и образует прямоугольник, включающий ячейки B3, C3, D3, B4, C4, D4.

Ячейки могут содержать следующие типы данных:

1. *Символьные (текстовые) данные.* Могут включать в себя алфавитные, числовые и специальные символы. По умолчанию символьные данные при вводе выравниваются по правому краю ячейки. При необходимости их можно выровнять по левому краю или по центру.

2. *Числовые данные.* Могут содержать только числовые данные. Исключением являются десятичная точка (или запятая) и знак числа, стоящий перед ним. Числовые данные по умолчанию выравниваются при вводе по левому краю ячейки.

3. *Формулы.* Формула может включать ряд арифметических, логических и других действий, производимых с данными из других ячеек. На экране после ввода формулы отображается результат вычислений, а сама формула отображается в строке ввода над таблицей. Ввод формул начинается со знака равенства.

4. *Функции.* Функции могут самостоятельно присутствовать в ячейках, а также могут входить в состав формул. Различают математические, статистические, логические, финансовые и другие функции.

5. *Даты.* В этом типе данных возможны такие функции, как добавление к дате числа (пересчет даты вперед и назад) или вычитание разности двух дат (длительность периода).

Названия ссылок	Обозначения	Результат при копировании или переносе формул
Частичная абсолютная ссылка	$A5	Не меняется номер столбца
	A$5	Не меняется номер строки
Полная абсолютная ссылка	$A$5	Не меняется ни номер строки, ни номер столбца

Excel имеет два окна — программное (внешнее) и рабочее (внутреннее). Внутреннее окно Sheet # содержит рабочую страницу (таких страниц несколько, они образуют книгу), представляющую двумерную прямоугольную таблицу. Справа и внизу на рабочей странице расположены линейки со стрелками прокрутки, позволяющие с помощью мыши быстро перемещаться по странице.

В окне Excel под зоной заголовка находится область заголовков меню. Чуть ниже находится основная линейка инструментов. Кнопки линейки инструментов позволяют быстро и легко вызывать различные функции Excel. Их можно вызывать также через меню.

Чтобы выполнить какое-либо действие с данными, помещенными в ячейки (ввод, копирование, удаление, форматирование и т.п.), необходимо их выделить. Чтобы выделить ячейку, укажите на нее и нажмите кнопку мыши. При нажатой кнопке можно выделить диапазон ячеек. После выделения необходимой области нажмите правую кнопку мыши, вызывая контекстное меню, которое позволяет выполнить ряд команд: ***Вырезать***, ***Копировать***, ***Вставить*** и т.п.

Изменение данных проводят прямо в ячейке. Перемещение или копирование содержимого ячеек можно осуществить перетаскиванием его с помощью мыши. Чтобы скопировать (а не переместить), держите нажатой клавишу **CTRL**.

Создание формулы начинается с ввода знака равенства (=). Формула содержит встроенные функции, адреса ячеек, константы. В случае затруднений с формированием формулы используйте *Мастер функций.*

Подобный сервис есть и при оформлении дизайна таблицы. Перед печатью таблиц (кнопка ***Печать***) удобно осуществить предварительный просмотр (кнопка ***Предварительный_просмотр*** в меню ***Сохранить***).

Excel работает с несколькими листами книги. Например, на одном листе можно разместить итоговые оценки студенческой группы за пять лет обучения, а на пяти следующих — данные за каждый год обучения. Листы книги могут служить местом для размещения графических иллюстраций, диаграмм.

Для отображения числовых данных в графической форме используют линии, полосы, столбцы, сектора и другие маркеры, а также их объединенные варианты. Для размещения диаграммы рядом с данными создают *внедренную диаграмму* на том же

листе. Можно создать диаграмму на отдельном *листе диаграммы*. Если нет времени, желания и возможности для относительно сложных построений, используют автоматическое оформление диаграмм с помощью команды ***Автоформат*** в меню ***Формат***.

Помимо того, что имеется большая встроенная библиотека построения графических образов (графиков, диаграмм, гистограмм), Excel содержит мощный встроенный графический редактор.

Excel не только «дружен» с текстовыми и графическими системами, но и поддерживает основные действия, характерные для систем управления базами данных (СУБД). Более того, у него развит аппарат импортирования и экспортирования данных из других программных систем.

В Excel, кроме адресов ячеек, предусмотрен также удобный способ ссылки на ячейку с помощью присвоения этой ячейке произвольного имени. Это делается при выделенной ячейке с помощи меню ***Вставка\Имя\Присвоить*** Ввести любое имя, например, *Сумма*. Введенное имя можно использовать в дальнейшем вместо адреса.

Для форматирования данных в ячейках электронной таблицы используется меню ***Формат\Ячейки***, для настройки ширины и высоты ячеек — меню ***Формат\Строка*** и ***Формат\Столбец***.

Контрольные вопросы

1. Что представляет собой электронная таблица?
2. Как формируется адрес ячейки?
3. Что называется диапазоном ячеек? Как он задается?
4. Чем различаются относительная и абсолютная ссылки?
5. Что означают частичная и полная относительная ссылки?
6. Какие типы данных встречаются в электронных таблицах?

Темы для рефератов

1. Работаем с QuattroPro.
2. Что мы знаем о Lotus 1, 2, 3.
3. Компьютерная графика в электронных таблицах.
4. Могут ли электронные таблицы заменить СУБД?
5. Программируем в электронных таблицах.

Тема семинарских занятий

Знакомство с технологией обработки числовых данных с помощью электронных таблиц. Основные сведения по работе с Excel.

Рекомендации по программному обеспечению

Могут быть использованы следующие программы:
1. SuperCalc-4, 5.
2. Excel-95, 97, 98, 2000.
3. QuattroPro.
4. Lotus 1, 2, 3.

Задачи и упражнения

Ввод данных (числа, формулы) в ячейки, копирование данных, форматирование числовых данных

Упражнение № 1. Основные приемы работы с ЭТ: ввод данных в ячейку, форматирование шрифта, автозаполнение, ввод формул, обрамление таблицы

Выполнить это упражнение на примере таблицы, вычисляющей n-й член и сумму арифметической прогрессии.

Формула n-го члена арифметической прогрессии: $a_n = a_1 + d(n-1)$, где a_1 — первый член прогрессии, d — разность арифметической прогрессии. Формула суммы n первых членов арифметической прогрессии: $S = (a_1 + a_n) \cdot n / 2$.

1. Создать новый документ (новую книгу), используя кнопку ***Создать*** на стандартной панели инструментов или использовать меню ***Файл\Создать***, выбрать вкладку ***Общие***.

Составить таблицу:

	A	B	C	D
1	Вычисление n-го члена и суммы арифметической прогрессии			
2				
3	d	n	a_n	S_n
4	0,725	1	−2	−2
5	0,725	2	−1,275	−3,275
6	0,725	3	−0,55	−3,825
7	0,725	4	0,175	−3,65
8	0,725	5	0,9	−2,75
9	0,725	6	1,625	−1,125
10	0,725	7	2,35	2,225
11	0,725	8	3,075	4,3
12	0,725	9	3,8	8,1
13	0,725	10	4,525	12,625

2. Выделить ячейку A1 (щелкнуть курсором мыши в нее) и ввести заголовок таблицы «Вычисление n-го члена и суммы арифметической прогрессии». Заголовок займет несколько ячеек правее A1. Введенные в ячейку данные фиксировать нажатием на клавишу выполнения (**Enter**).

3. Оформить строку заголовков столбцов. В ячейку A3 введите d, в B3 — n, в C3 — a_n, в D3 — S_n. Размер шрифта 11 п., выравнивание по центру, применить полужирный стиль начертания символов. Для набора нижних индексов воспользоваться командой ***Формат\Ячейки***, выбрать вкладку ***Шрифт\Нижний_индекс***.

4. Оформить заголовок таблицы. Выделить ячейки A1:D2, применить полужирное начертание символов и выполнить команду ***Формат\Ячейки***, вкладка ***Выравнивание*** — установить выравнивание по горизонтали «По центру», активизировать переключатели «Переносить по словам» и «Объединение ячеек».

5. В ячейку A4 ввести величину разности арифметической прогрессии.

6. Далее надо заполнить ряд нижних ячеек. Для этого в выделенной ячейке A4 установить курсор мыши в правый нижний угол ячейки. Когда курсор примет форму черного крестика (маркер заполнения), нажать на левую кнопку мыши и протянуть маркер заполнения на несколько ячеек вниз. Весь ряд заполнится данными, повторяющими данные ячейки A4.

7. В следующем столбце необходимо ввести последовательность чисел от 1 до 10. Для этого ввести в ячейку B4 число 1, в ячейку B5 число 2. Выделите обе ячейки и, взявшись за маркер заполнения, протяните его вниз.

8. Или: ввести в ячейку B4 число 1. Выделить блок, который надо заполнить. Выбрать команду ***Правка\Заполнить\Прогрессия***. Выбрать необходимые параметры.

9. В ячейку C4 ввести значение первого члена арифметической прогрессии.

10. В остальные нижние ячейки надо ввести формулу *n*-го члена арифметической прогрессии. Выделите ячейку C5 и введите в нее формулу =C4+A4 (все формулы начинаются со знака равенства!). Можно не набирать адреса ячеек при вводе формул, а просто в тот момент, когда надо в формуле набрать адрес ячейки, щелкнуть именно в эту ячейку.

11. Скопировать данную формулу в нижние строки данного столбца.

12. В данном примере разность арифметической прогрессии необязательно писать во всех строках, так как она везде одинакова. Достаточно ввести ее в ячейку A4. Тогда при вводе формулы в ячейку C5 необходимо применить абсолютную ссылку на ячейку A4. Внесите эти изменения.

13. Занести в ячейку D4 формулу суммы *n* первых членов арифметической прогрессии с учетом адресов ячеек =(C$4+C4)*B4/2. Заполните формулами нижние столбцы.

14. Выделить блок — ячейки с данными (кроме заголовков таблицы и столбцов) и установить необходимые параметры для шрифта, например, размер шрифта 11 п., выровнять вправо, используя меню ***Формат\Ячейки***, вкладки ***Выравнивание*** и ***Шрифт***. Если какие-то данные в ячейках не помещаются, необходимо выделить блок ячеек с данными и выполнить автоподбор ширины ячеек через меню ***Формат\Столбец... \Автоподбор_ширины***.

15. Если выбрать команду ***Файл\Просмотр***, то наша таблица окажется необрамленной. Для обрамления выделить таблицу (без заголовка), выбрать стиль линии (***Формат\Ячейки...*** вкладка ***Рамка***) и активизировать переключатели *Сверху*, *Снизу*, *Слева*, *Справа*.

16. Выполнить те же действия для обрамления заголовка, предварительно выделив блок ячеек заголовка.

Диаграммы, графики, условия, функции

Упражнение № 2. Нахождение наибольшего и наименьшего элементов в числовой таблице

Например, имеется таблица:

	A	B	C	D
1	12	−14,5	23	42
2	36	17	9	−3,76
3	64	39	25	−1
4				

1. Создать новую рабочую книгу.

2. Внести элементы данной таблицы в ячейки.

3. Установить курсор в ячейку С4, ввести запись «максимальное».

4. Перейти в ячейку D4, выполнить команду ***Вставка\Функция*** или щелкнуть на кнопку ***Вставка_функции*** на панели инструментов.

5. В появившемся диалоговом окне выбрать функции ***Статистические\МАКС***.

6. В следующем диалоговом окне необходимо в строке ***Число1*** ввести диапазон A1:D3. Для этого выделить его в таблице.

7. Аналогично пунктам 3 и 4 выполнить действия по нахождению минимального в строке 5.

8. Переименовать лист. Для этого выполнить команду ***Формат\Лист\Переименовать*** или выполнить двойной щелчок внизу на вкладке Лист1. Ввести имя *МаксМин*.

Упражнение № 3. Построение диаграммы

1. Перейти на Лист 2.

2. Протабулировать функцию $y = 2 \cdot x^2 - 4 \cdot x - 6$ на отрезке [−5, 5] с шагом 1. Найти промежутки перемены знака значений функции. Определить корни уравнения. Должна получиться следующая таблица:

	A	B
1	X	*Y*
2	−5	64
3	−4	42
4	−3	24
5	−2	10
6	−1	0
7	0	−6
8	1	−8
9	2	−6
10	3	0
11	4	10
12	5	24

3. Построить график функции на данном интервале. Для этого выполнить команду ***Вставка\Диаграмма\На_этом_листе*** или щелкнуть на панели инструментов кнопку ***Мастер_диаграмм***. Далее выполнить шаги 1—4.

4. Шаг 1: во вкладке ***Стандартные*** выбрать тип ***График*** и левый верхний вид.

5. Шаг 2: при активной вкладке ***Диапазон_данных*** выделить диапазон значений функции в таблице вместе с заголовком C1:C12. При активной вкладке ***Ряд*** в строке ***Подписи_оси_X*** активизировать курсор, затем выделить диапазон данных в таблице B2:B12.

6. Шаг 3: самостоятельно изучить все вкладки на этом этапе.

7. Шаг 4: указать местоположение полученной диаграммы (на новом листе или на этом).

8. Измените цвет и толщину линии графика. Для этого подведите курсор мыши к линии графика и выполните двойной щелчок мышью. В появившемся окне ***Форматирование_ряда_данных*** выберите другой цвет и другую толщину линии, активизировав вкладку ***Вид***.

9. Выполнив двойной щелчок мыши на линиях осей, измените цвет в появившемся диалоговом окне *Форматирование осей.*

10. Дайте имя Листу 2 «Функция» или «График».

Упражнение № 4. Условия в электронных таблицах

Вычислить значения функции в зависимости от значений аргумента на интервале [−5, 5] с шагом 1.

$$y = \begin{cases} x \cdot x - 4 & \text{для } x < 0, \\ x + 5 & \text{для } x > 0 \text{ или } x = 0. \end{cases}$$

Должна получиться следующая таблица:

	A	B
1	*X*	*Y*
2	−5	=ЕСЛИ(A2<0;A2*A2−4;A2+5)
3	−4	
4	−3	
5	−2	
6	−1	
7	0	
8	1	
9	2	
10	3	
11	4	
12	5	

1. Перейти на Лист 2.

2. В ячейки столбца, озаглавленного буквой *X*, внести значения от −5 до 5 с шагом 1.

3. Прочитать справку в Excel о логических функциях. Для этого выбрать в меню Excel **?*Вызов_справки***. Во вкладке ***Предметный_указатель*** в строке поиска ввести текст *логические функции.*

4. В первую строку значений *Y* ввести логическую функцию ***Если***, используя *Мастер функций.*

5. Скопировать формулу в нижние ячейки.

6. Построить график. Определить, при каких значениях *X* функция *Y* принимает значение нуль.

Упражнение № 5. Нахождение корня уравнения методом последовательных приближений

Напомним, что метод последовательных приближений при решении уравнения $x = \varphi(x)$ заключается в построении итерационной последовательности $x_{n+1} = \varphi(x_n)$. Если функция $\varphi(x)$ является сжимающей, то последовательность $\{x_n\}$ сходится к корню уравнения.

	A	B	C	D
1	Метод последовательных приближений			
2	Номер приближения	Приближение	Значение *X*	Отклонение
3	1	1	1,393	0,393
4	2	1,393	1,474	0,081
5	3	1,474	1,487	0,013
6	4	1,487	1,489	0,002
7	5	1,489	1,490	0,001
8	6	1,490	1,490	0,000

Продемонстрируем процесс решения на примере уравнения $x = 1 + 0{,}5\ \mathrm{arctg}(x)$.

1. Перейти на Лист 3
2. Ввести данные (константы) в ячейки A3:A7.
3. В ячейку B3 ввести значение 1.
4. В ячейку C3 ввести формулу 1 + 0,5*ATAN($B3), для ввода функции arctg использовать *Мастер_функций.*
5. В ячейку D3 ввести абсолютную разность ячеек B3 и C3, применив для них абсолютную адресацию столбцов. Для ввода функции использовать *Мастер_функций.*
6. В ячейку B4 ввести содержимое ячейки C3, используя абсолютный адрес столбца: =$C3.
7. Скопировать все формулы вниз до тех пор, пока в колонке *Отклонение* не появится нуль.
8. Дать имя листу «Приближение».

Упражнение № 6. Решение квадратного уравнения

Решить квадратное уравнение $y = ax^2 + bx + c$, используя ЭТ. Оформить заголовок. Выполнить решение для нескольких наборов коэффициентов:

1) $a = 2$, $в = -3$, $c = -2$; 2) $a = 2$, $в = -4$, $c = -6$; 3) $a = -3$, $в = 1$, $c = 1$.

Упражнение № 7. Использовать ЭТ для решения математических, физических, экономических и других прикладных задач

Каждый студент предлагает свою задачу.

Структурирование и отбор данных в ЭТ

Упражнение № 8. Сортировка (упорядочение) записей списка

Рассмотрим заданную таблицу *УЧЕТ ТОВАРОВ НА СКЛАДЕ,* представленную ниже.

Прокомментируем эту таблицу. Таблица имеет вид базы данных, состоящей из записей продажи товаров со склада. Запись указывает, какой организации продан товар, когда проведена продажа, товар, единицу измерения товара, его стоимость и количество. В столбцах *Дебет* и *Кредит* заносится стоимость покупки и долг перед организацией, т. е. *Цена*Кол-во.* В последнем столбце указывается форма оплаты: безналичный расчет (б/р), бартер (бар), наличный расчет (н/р).

Для дальнейшей работы создать эту таблицу в Excel на *Листе1* и сохранить ее в виде отдельного файла.

Учет товаров на складе

	A	B	C	D	E	F	G	H	I	J
1	**Организация**	**Дата**	**Товар**	**Ед. изм.**	**Цена**	**Кол-во 1**	**Дебет**	**Кол-во 2**	**Кредит**	**Ф. опл.**
2	АО «Альянс»	1 Янв	соль	кг	15 000	550	8 250 000			б/р
3	АОЗТ «Белокуриха»	1 Янв	сахар	кг	16 000	200	3 200 000			б/р
4	АОЗТ «Белокуриха»	3 Янв	хлеб	бул	700		0	900	630 000	бар
5	Бийск.маслосырзавод	3 Июн	сода	пач	5500	300	1 650 000			б/р
6	АОЗТ «Белокуриха»	4 Янв	сок	бан	56	26 000	1 456 000			б/р
7	к/з «ЗАРЯ»	4 Янв	пиломт	метр			0			б/р
8	АО «Альянс»	13 Янв	лимоны	кг	4000	50	200 000			б/р
9	АО «Альянс»	3 Фев	компьют	шт	250 000	2	5 000 000			б/р
10	АОЗТ «Белокуриха»	12 Фев	хлеб	бул	700		0	500	350 000	б/р
11	Бийск.маслосырзавод	12 Фев	бензин	л	450		0	6048	2 721 600	н/р
12	АОЗТ «Белокуриха»	2 Мар	хлеб	бул	3000	215	645 000			б/р
13	к/з «ВОСТОК»	2 Мар	апельсин	кг	4000	100	400 000			б/р
14	к/з «ЗАРЯ»	5 Мар	апельсин	кг	2300	124	285 200			н/р
15	к/з «ЛУЧ»	4 Апр	апельсин	кг	5000		0	50	250 000	б/р
16	к/з «ЗАРЯ»	6 Апр	мука	кг	20 000	1000	2 000 000			н/р
17	к/з «ВОСТОК»	6 Май	сахар	кг	16 000	50	800 000			б/р
18	к/з «ВОСТОК»	13 Июн	лимоны	кг	6000		0	50	300 000	б/р
19	к/з «ВОСТОК»	13 Июн	хлеб	бул	700	300	210 000			б/р

1. Для упорядочения записей необходимо определить, по каким полям вы хотите отсортировать таблицу. Например, необходимо отсортировать наименования организаций в алфавитном порядке, внутри каждой организации наименование товара в алфавитном порядке и внутри каждого наименования товара отсортировать по возрастанию количество проданного товара.

2. Курсор установить в область таблицы, выполнить команду ***Данные\Сортировка***. В первом уровне сортировки выбрать поле *Организация*, во втором — *Товар*, в третьем — *Кол-во1*.

3. Просмотреть результаты сортировки.

Упражнение № 9. Фильтрация (выборка) записей списка

Автофильтр:

1. Скопируйте таблицу с Листа1 на Лист2 и назовите новый лист *Автофильтр*.

2. Допустим, нам необходимо выбрать из заданного списка только те строки, где есть запись АОЗТ «Белокуриха». Выполнить команду ***Данные\Фильтр\Автофильтр***.

3. В строке заголовка таблицы появились значки падающего меню. (Чтобы их убрать, необходимо выполнить ту же команду, по которой их вызывали.)

4. Щелкнуть на значок в столбце *Организация* и выбрать АОЗТ «Белокуриха». Появились только те записи, где присутствует указанная организация. Чтобы вернуть все записи, надо опять щелкнуть на значок и выбрать строку *Все*.

5. Вывести на экран записи, содержащие организацию АОЗТ «Белокуриха», где в столбце «Товар» присутствует «хлеб», т.е. осуществить выборку по двум полям. Вернуть все записи.

6. Вывести на экран записи, содержащие организацию АОЗТ «Белокуриха», в которых цена товара не превышает 16 000. Для выборки по столбцу «Цена» при открытии меню выбрать строку *Условие*. В появившемся окне *Пользовательский_автофильтр* при помощи значков открывающегося меню установить условие <16 000 в верхней строке. Вернуть все записи.

7. Вывести на экран записи, содержащие колхоз «Восток» и дату покупки товара в промежутке после 2 марта до 13 июня. В данном случае в окне *Пользовательский_автофильтр* заполнить обе строки. Правильно выбрать соединение условий И или ИЛИ. Вернуть все записи.

8. Вывести на экран записи, содержащие колхоз «Восток», а в поле *Цена* установить условие: больше 700, но меньше 16 000. Вернуть все записи.

Расширенный фильтр:

Прочитать встроенную справку Excel ***Фильтры\Расширенные:_Примеры_условий_отбора_ расширенного_ фильтра***.

1. Скопируйте таблицу с Листа 1 на Лист 3 и дайте имя листу *Расширенный фильтр*.

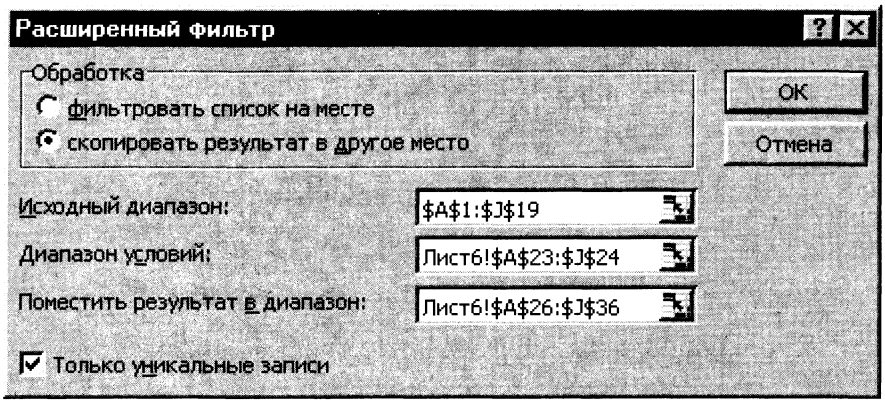
2. Выполнить задание по автофильтру, воспользовавшись командой ***Расширенный_фильтр***.

• Ниже таблицы, оставив пустые 2—3 строки, скопировать строку заголовка таблицы, например, в строку 23. В строке 24 сформировать критерий отбора записей. В столбец *Организация* ввести АОЗТ «Белокуриха», в столбец *Товар* — «хлеб». Установить курсор в область таблицы, в которой будет производиться выборка данных.

• Далее выполнить команду ***Данные\Фильтр\Расширенный_фильтр***. В появившемся диалоговом окне в строке *Исходный диапазон* появится запись A1:J19.

• В диалоговом окне установить курсор в строку *Диапазон* условий, перейти в таблицу и выделить диапазон A23:J24.

• Для того чтобы новые данные печатались в другом месте, необходимо активизировать кнопку ***Скопировать_результат_в_другое_место*** и также указать диапазон, куда будут выводиться отсортированные данные, например, $A$26:$J$36.



Начиная со строки 26 и ниже расположен результат выполнения расширенного фильтра.

	A	B	C	D	E	F	G	H	I	J
1	**Организация**	**Дата**	**Товар**	**Ед. изм.**	**Цена**	**Кол-во1**	**Дебет**	**Кол-во2**	**Кредит**	**Форма оплаты**
2	АО «Альянс»	01.01.98	соль	кг	15 000,00	550	8 250 000,00			б/р
3	АОЗТ «Белокуриха»	01.01.98	сахар	кг	16 000,00	200	3 200 000,00			б/р
4	АОЗТ «Белокуриха»	03.01.98	хлеб	бул	700,00			900	630 000,00	бар
5	Бийск.маслосырзавод	03.06.98	сода	пач	5500,00	300	1 650 000,00			б/р
6	АОЗТ «Белокуриха»	04.01.98	сок	бан	56,00	26 000	1 456 000,00			б/р
7	к/з «ЗАРЯ»	04.01.98	пиломт	метр						б/р
8	АО «Альянс»	13.01.98	лимоны	кг	4000,00	50	200 000,00			б/р
9	АО «Альянс»	03.02.98	компьют	шт	250 000,00	2	500 000,00			б/р
10	АОЗТ «Белокуриха»	12.02.98	хлеб	бул	700,00			5 000	3 500 000,00	б/р
11	Бийск.маслосырзавод	12.02.98	бензин	л	450,00			6 048	2 721 600,00	н/р
12	АОЗТ «Белокуриха»	02.03.98	хлеб	бул	3000,00	215	645 000,00			б/р
13	к/з «ВОСТОК»	02.03.98	апельсин	кг	4000,00	100	400 000,00			б/р
14	к/з «ЗАРЯ»	05.03.98	апельсин	кг	2300,00	124	285 200,00			н/р
15	к/з «ЛУЧ»	04.04.98	апельсин	кг	5000,00			50	250 000,00	б/р
16	к/з «ЗАРЯ»	06.04.98	мука	кг	20 000,00	1 000	20 000 000,00			н/р
17	к/з «ВОСТОК»	06.05.98	сахар	кг	16 000,00	50	800 000,00			б/р
18	к/з «ВОСТОК»	13.06.98	лимоны	кг	6000,00			50	300 000,00	б/р
19	к/з «ВОСТОК»	13.06.98	хлеб	бул	700,00	300	210 000,00			б/р
20										
21										
22	**Организация**	**Дата**	**Товар**	**Ед. изм.**	**Цена**	**Кол-во1**	**Д-76**	**Кол-во2**	**К-76**	**Форма оплаты**
23	АОЗТ «Белокуриха»		хлеб							
24										
25	**Организация**	**Дата**	**Товар**	**Ед. изм.**	**Цена**	**Кол-во1**	**Дебет**	**Кол-во2**	**Кредит**	**Форма оплаты**
26	АОЗТ «Белокуриха»	03.01.98	хлеб	бул	700,00			900	630 000,00	бар
27	АОЗТ «Белокуриха»	12.02.98	хлеб	бул	700,00			5000	3 500 000,00	б/р
28	АОЗТ «Белокуриха»	02.03.98	хлеб	бул	3000,00	215	645 000,00			б/р

Лист1 / Лист2 / Лист3 / Лист4 / Лист6

3. Выполнить задание п. 6 из упражнения. В данном случае для выборки товаров, стоимость которых не превышает 16 000, в диапазоне критериев введите «<16 000».

Примечание. Алгоритм выполнения расширенного списка можно прочитать в справке Excel ***Фильтры\Расширенные:_Фильтрация_списка_с_помощью_расширенного_фильтра***.

Упражнение № 10. Автоматическое подведение итогов

1. Скопируйте таблицу с Листа 1 на Лист 4.

2. Предположим, необходимо подвести итоги о продаже товаров каждой организации, затем еще итоги в каждой организации по датам.

3. Сначала необходимо упорядочить таблицу по полю *Организация*, второй уровень сортировки — *Дата*.

4. Затем выполнить команду ***Данные\Итоги***. В появившемся окне в первой строке выбрать *Организация*, в строке *Операция* из списка выбрать *Сумма*, в третьей строке выбрать поля, по которым подводить итоги: *Дебет и Кредит*.

5. Просмотреть результаты. В левой половине экрана появились символы структуры (значки «плюс» и «минус»). Пощелкать на них и ознакомиться с их назначением.

6. Далее вновь выполнить команду подведения итогов. Вводим поле *Дата*. Чтобы предыдущие итоги не стерлись, значок *Заменять текущие итоги* должен быть выключен. В результате получим таблицу итогов о продаже товара каждой организации и вдобавок еще итоговые данные по датам продажи.

7. Отмените все итоги. Чтобы убрать итоги, вызвать вновь окно *Итоги* и выбрать команду ***Убрать_все***.

Упражнение № 11. Консолидация данных (способ получения итоговой информации из разных листов, одинаковых по структуре)

1. Предположим, есть три таблицы одинаковой структуры *УЧЕТ ПРОДАЖИ ТОВАРА* одной фирмы, имеющей три склада в разных точках города. Создадим эти три таблицы. Для этого надо скопировать таблицу с Листа 1 на Листы 4, 5 и 6. Дать имена Листам 4, 5, 6 Склад 1, Склад 2, Склад 3 соответственно.

2. Изменить некоторые данные на Листах 5 и 6 для организации АОЗТ «Белокуриха» для товара *хлеб*, например, *Количество1, Количество 2* или *Цену*.

3. Допустим, необходимо подвести итоги о продаже хлеба организации АОЗТ «Белокуриха» в сумме в этих трех точках.

4. Для этого все три таблицы должны быть упорядочены по полю *Организация*, внутри каждой организации упорядочены по полю *Товар*. Подвести итоги по полю *Товар*, суммирующие значения по полям *Дебет* и *Кредит*.

5. Дать имя Листу 7 *Консолидация*. Находясь в этом листе, выполнить команду ***Данные\Консолидация***.

6. В появившемся диалоговом окне выбрать функцию *Сумма*.

7. Щелкнуть мышью в поле *Ссылки*, перейти на лист *Склад 1* и выделить итоговую сумму продажи сахара АОЗТ «Белокуриха». Данные появятся в поле ссылки. Нажать кнопку ***Добавить***.

8. То же самое выполнить для листов «Февраль» и «Март».

9. Указать флажок *Создавать_связи_с_исходными_данными*. Тогда, если будут меняться исходные таблицы, автоматически будет пересчитываться и суммирующая таблица.

10. Просмотреть полученную таблицу.

Упражнение № 12. Сводные таблицы

Команда ***Данные\Сводная_таблица*** вызывает *Мастера_сводных_таблиц* для построения сводов, т.е. итогов определенных видов на основании данных списков,

других сводных таблиц, внешних баз данных. Сводная таблица обеспечивает различные способы агрегирования информации.

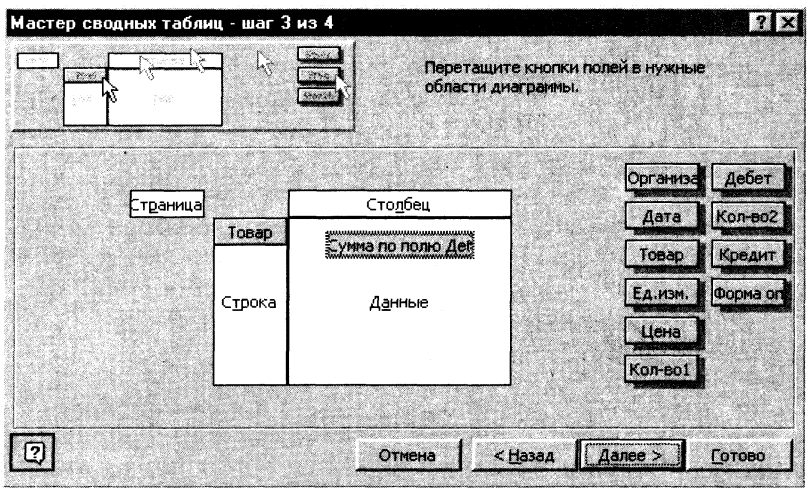
Предположим, нам необходимо узнать, на какую сумму закупила товар какая-нибудь организация.

1. Скопировать таблицу с Листа 1 на Лист 8.
2. Выполнить команду ***Данные\Сводная_таблица***.
3. Выполнить первые два шага работы с *Мастером* самостоятельно.
4. На третьем шаге перетащить значки с названиями столбцов нашей таблицы следующим образом:

Информация, которую мы хотим разместить в строках, например:

- *Организация*. Схватить мышью значок и перетащить в область *Строка*.
- Перенести значок *Дебет* в область *Данные*.

5. Если выполнить двойной щелчок на перенесенных значках, можно редактировать их назначение. На значках, помещенных в окне *Строка*, активизировано состояние строки, в окне *Столбец* — столбца. Двойной щелчок на значке в области данных позволяет выполнить операцию.



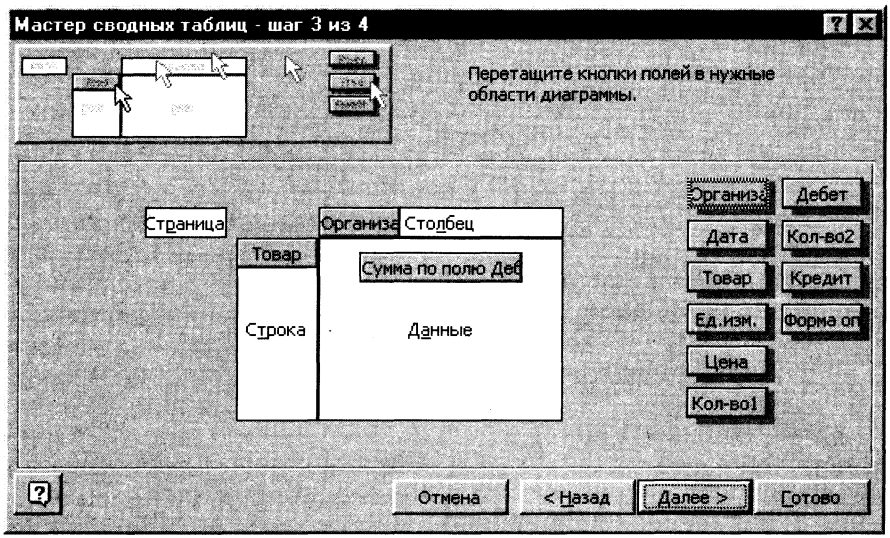
6. Просмотреть полученную таблицу:

	A	B
1	Сумма по полю Дебет	
2	Товар	Всего
3	апельсин	685200
4	бензин	0
5	компьют	500000
6	лимоны	200000
7	мука	20000000
8	пиломт	0
9	сахар	4000000
10	сода	1650000
11	сок	1456000
12	соль	8250000
13	хлеб	855000
14	Общий итог	37596200
15		

7. Программа предлагает создать на новом листе сводную таблицу или создать на нем существующую таблицу.

8. Установить курсор в область сводной таблицы и выполнить команду ***Данные\ Сводная_таблица***.

9. Перенести поле *Организация* на область *Столбец*, а поле *Товар* на область *Строка*.



10. Просмотреть полученную таблицу:

	A	B	C	D	E	F	G	H
1	Сумма по полю Деб	Организация						
2	Товар	АО «Альянс	АОЗТ «Бело	Бийск.маслс	к/з «ВОСТО	к/з «ЗАРЯ»	к/з «ЛУЧ»	Общий итог
3	апельсин				400000	285200	0	685200
4	бензин			0				0
5	компьют	500000						500000
6	лимоны	200000			0			200000
7	мука					20000000		20000000
8	пиломт					0		0
9	сахар		3200000		800000			4000000
10	сода			1650000				1650000
11	сок		1456000					1456000
12	соль	8250000						8250000
13	хлеб		645000		210000			855000
14	Общий итог	8950000	5301000	1650000	1410000	20285200	0	37596200
15								
16								

11. Можно менять группировку строк и столбцов следующим образом: в полученной таблице поле *Организация* перенести влево. Ваша таблица примет другой вид. В конце таблицы — общая сумма.

	A	B	C
2	Товар	Организация	Всего
3	апельсин	к/з «ВОСТОК»	400000
4		к/з «ЗАРЯ»	285200
5		к/з «ЛУЧ»	0
6	апельсин Всего		685200
7	бензин	Бийск.маслосырзавод	0
8	бензин Всего		0
9	компьют	АО «Альянс»	500000
10	компьют Всего		500000
11	лимоны	АО «Альянс»	200000
12		к/з «ВОСТОК»	0
13	лимоны Всего		200000
14	мука	к/з «ЗАРЯ»	20000000
15	мука Всего		20000000
16	пиломт	к/з «ЗАРЯ»	0
17	пиломт Всего		0
18	сахар	АОЗТ «Белокуриха»	3200000
19		к/з «ВОСТОК»	800000
20	сахар Всего		4000000
21	сода	Бийск.маслосырзавод	1650000
22	сода Всего		1650000
23	сок	АОЗТ «Белокуриха»	1456000
24	сок Всего		1456000
25	соль	АО «Альянс»	8250000
26	соль Всего		8250000
27	хлеб	АОЗТ «Белокуриха»	645000
28		к/з «ВОСТОК»	210000
29	хлеб Всего		855000
30	Общий итог		37596200
31			

12. Если нужна группировка не по товарам, а по организациям, поле *Товар* перетащить вправо.

Фильтрация сведенных данных:

13. Перетащить в полученной таблице поле *Организация* в левый верхний угол. Тащить до тех пор, пока не появятся три прямоугольничка, расположенных друг на друге.

14. Щелкнуть по язычку ***Все*** и выбрать интересующую вас организацию.

	A	B
1	Организация	(Все)
2		
3	Сумма по полю Дебет	
4	Товар	Всего
5	апельсин	685200
6	бензин	0
7	компьют	500000
8	лимоны	200000
9	мука	20000000
10	пиломт	0
11	сахар	4000000
12	сода	1650000
13	сок	1456000
14	соль	8250000
15	хлеб	855000
16	Общий итог	37596200
17		

	A	B
1	Организация	к/з «ЗАРЯ»
2		
3	Сумма по полю Дебет	
4	Товар	Всего
5	апельсин	285200
6	мука	20000000
7	пиломт	0
8	Общий итог	20285200
9		
10		

Предположим, нам необходима сводная таблица о продажах по всем организациям по месяцам.

15. Изменим сводную таблицу. Курсор должен находиться в области таблицы. Выполнить команду ***Данные\Сводная_таблица***. Установить значок *Организация* в область строк, а значок *Дата* в область *Столбец*. В поле *Данные* оставить значок *Сумма по полю Дебет*.

16. Просмотреть полученную таблицу.

17. Преобразуем полученную таблицу по месяцам. Для этого установить курсор на ячейку с записью *Дата* и выбрать пункт меню ***Данные\ Группа и структура\ Группировать***.

Упражнение № 13. Структурирование таблиц

Применяется для работы с большими таблицами, если есть необходимость закрывать и открывать отдельные строки таблицы.

1. Отсортировать строки списка по нужной вам классификации, например, по организациям или по наименованию товара.

2. Вставить пустые строки для разделенных групп.

3. Выделить первую группу, выполнить команду ***Данные\Группа и структура\ Группировать***.

4. Аналогичные действия выполнить для последующих групп.

5. Слева на экране появится значок «-». При щелчке на этот значок данные скрываются.

6. Для отмены группировки необходимо выделить группы и выполнить команду ***Данные\Группа и структура\Разгруппировать***.

7. Отменить все группировки.

Лабораторные работы

Лабораторная работа № 1
Ввод данных (числа, формулы) в ячейки, копирование данных, форматирование числовых данных

Время выполнения 2 часа.

Задание. Составьте таблицу значений функции двух переменных $F(x, y)$, заданной в прямоугольной области $[a, b] \times [c, d]$, для аргументов $x_i = a + ih_x$, $y_j = c + jh_y$, где $i = 0..N_x$, $j = 0..N_y$, $(h_x = (b - a)/N_x$, $h_y = (d - c)/N_y)$.

Варианты заданий

№	$F(x,y)$	a	b	c	d	h_x	h_y
1	$xy + 5{,}6(x + y)$	0	1	0	1	0,1	0,1
2	$\ln(x + y)$	1	3	0	2	0,2	0,2
3	$\cos(x) + \sin(y)$	0	1	0	1	0,1	0,1
4	$\sin(x) + \cos(y)$	0	1	0	1	0,1	0,1
5	$\mathrm{tg}(x + y)$	1	3	0	2	0,2	0,2
6	$\sin(x) + xy$	0	1	0	1	0,1	0,1
7	$\cos(x) + 5xy$	0	1	0	1	0,1	0,1
8	$y + \mathrm{tg}\,(x + y)$	1	3	0	2	0,2	0,2
9	$\sin(xy) + \cos(xy)$	0	1	0	1	0,1	0,1

Окончание таблицы

№	$F(x,y)$	a	b	c	d	h_x	h_y
10	$5\sin(\cos(x + y) + 3{,}78)$	0	1	0	1	0,1	0,1
11	$6{,}4\cos(5{,}8 + \ln(xy))$	1	3	0	2	0,2	0,2
12	$xy + x^2 + y^2$	0	1	0	1	0,1	0,1
13	$\sin^2(x + y) + \cos^2(x + y)$	0	1	0	1	0,1	0,1
14	$x\sin(y) + y\cos(x)$	1	3	0	2	0,2	0,2
15	$10xy(\sin(xy) + \cos(y))$	0	1	0	1	0,1	0,1
16	$xy - 5{,}6(x - y)$	0	1	0	1	0,1	0,1
17	$xy\ln(x + y)$	1	3	0	2	0,2	0,2
18	$xy(\cos(x) + \sin(y))$	0	1	0	1	0,1	0,1
19	$x(\sin(x) + \cos(y))$	0	1	0	1	0,1	0,1
20	$xy\,\mathrm{tg}(x + y)$	1	3	0	2	0,2	0,2
21	$\sin(x) + xy - 2$	0	1	0	1	0,1	0,1
22	$\cos(x) + 5xy - \sin(y)$	0	1	0	1	0,1	0,1
23	$yx + \mathrm{tg}(x+y) - 4$	1	3	0	2	0,2	0,2
24	$xy(\sin(xy) + \cos(xy))$	0	1	0	1	0,1	0,1
25	$xy\sin(\cos(x + y) + 3{,}78)$	0	1	0	1	0,1	0,1

Лабораторная работа № 2
Диаграммы, графики, условия, функции

Время выполнения 4 часа.

Задание. С помощью ЭТ найдите приближенное значение одного из корней уравнения $F(x) = 0$ заданным методом:

а) деления отрезка пополам; б) хорд;

в) касательных; г) простой итерации.

Для каждого из методов сделать 10 итераций. Составить таблицу приближений для каждого шага итераций. Проверить точность каждого приближения подстановкой в заданное уравнение.

Построить график функции вблизи корня.

Варианты заданий

№	$F(x)$	Метод	№	$F(x)$	Метод
1	$x\cdot x - 5{,}6\cdot x + \ln(x)$	а)	13	$\cos(x) + \sin(x\cdot x) - 0{,}5$	а)
2	$x\cdot x - 5{,}6\cdot x + \ln(x)$	б)	14	$\cos(x) + \sin(x\cdot x) - 0{,}5$	б)
3	$x\cdot x - 5{,}6\cdot x + \ln(x)$	в)	15	$\cos(x) + \sin(x\cdot x) - 0{,}5$	в)
4	$x\cdot x - 5{,}6\cdot x + \ln(x)$	г)	16	$\cos(x) + \sin(x\cdot x) - 0{,}5$	г)
5	$\ln(x + 1{,}13) - x$	а)	17	$\sin(x) + x\cdot x$	а)
6	$\ln(x + 1{,}13) - x$	б)	18	$\sin(x) + x\cdot x$	б)
7	$\ln(x + 1{,}13) - x$	в)	19	$\sin(x) + x\cdot x$	в)
8	$\ln(x + 1{,}13) - x$	г)	20	$\sin(x) + x\cdot x$	г)
9	$1/x + x\cdot x - \sin(x)$	а)	21	$\log(x) - 1/x + 2{,}7$	а)
10	$1/x + x\cdot x - \sin(x)$	б)	22	$\log(x) - 1/x + 2{,}7$	б)
11	$1/x + x\cdot x - \sin(x)$	в)	23	$\log(x) - 1/x + 2{,}7$	в)
12	$1/x + x\cdot x - \sin(x)$	г)	24	$\log(x) - 1/x + 2{,}7$	г)

Лабораторная работа № 3
Структурирование и отбор данных в ЭТ

Время выполнения 8 часов.

Задание. Создайте ЭТ *СТИПЕНДИАЛЬНАЯ ВЕДОМОСТЬ ФАКУЛЬТЕТА*. Представьте, что на факультете — 5 курсов, на каждом курсе — 2 группы, в группах — по 25 человек. В таблице используйте данные: ФИО студента, успеваемость (средний балл за сессию), сумма, надбавки за отличную и хорошую учебу. Стипендия студентам, имеющим балл ниже 3,5, не начисляется (в соответствующей графе указать 0).

Подготовьте отчеты по указанным в вариантах заданиям.

Варианты заданий

Вариант 1

Сформируйте сводную ведомость студентов с отличной учебой. Выдайте диаграмму с долей таких учащихся. Создайте отчеты по каждому курсу с графическим отображением. Оформите диаграммы распределения отличников по группам курса и по курсам.

Вариант 2

Сформируйте сводную ведомость студентов со средней успеваемостью. Выдайте диаграмму с долей таких учащихся. Создайте отчеты по каждому курсу с графическим отображением. Оформите диаграммы распределения студентов со средней успеваемостью по группам курса и по курсам.

Вариант 3

Сформируйте сводную ведомость неуспевающих студентов. Выдайте диаграмму с долей таких учащихся. Создайте отчеты по каждому курсу с графическим отображением. Оформите диаграммы распределения неуспевающих студентов по группам курса и по курсам.

Вариант 4

Пусть первоначально составленная ведомость определяет фиксированный стипендиальный фонд факультета. Отмените выдачу стипендии для студентов, имеющих средний балл успеваемости ниже 4. Перераспределите экономию стипендиального фонда для каждой группы и для каждого курса в зависимости от доли отличников. Выдайте соответствующие отчеты.

Вариант 5

Пусть первоначально составленная ведомость определяет фиксированный стипендиальный фонд факультета. Отмените надбавки за отличную учебу. Перераспределите экономию стипендиального фонда для каждой группы и для каждого курса в зависимости от доли студентов, получающих стипендию. Выдайте соответствующие отчеты.

Вариант 6

Пусть первоначально составленная ведомость определяет фиксированный стипендиальный фонд факультета. Отмените выдачу стипендии для студентов, имеющих средний балл успеваемости ниже 4,5. Экономию стипендиального фонда пе-

перераспределите всем студентам пропорционально их успеваемости. Составьте диаграммы роста размера стипендии для успевающих студентов (балл выше 4,5).

Вариант 7

Пусть первоначально составленная ведомость определяет фиксированный стипендиальный фонд факультета. Перераспределите заданный фонд всем (без исключения) студентам. Составьте диаграммы изменения размера стипендии для трех категорий студентов (отличники, успевающие, неуспевающие).

Вариант 8

Отчислите из каждой группы произвольным образом по три студента (не только неуспевающих!). Выполните задание варианта 1.

Вариант 9

Отчислите из каждой группы произвольным образом по несколько студентов, не менее 5 в каждой группе (не только неуспевающих!). Выполните задание варианта 2.

Вариант 10

Пусть первоначально составленная ведомость определяет фиксированный стипендиальный фонд факультета. Внесите в каждую группу дополнительно до трех студентов с разной успеваемостью. Перераспределите стипендиальный фонд в основном за счет лишения стипендии студентов с низким баллом успеваемости. Составьте отчет по изменению числа студентов, получающих стипендию, по курсам.

Вариант 11

Дополнительно факультету выделено 50 % стипендиального фонда. Проведите перерасчет размера стипендии. Составьте отчеты изменения размера стипендии по трем категориям студентов (по успеваемости).

Вариант 12

Дополнительно факультету выделено 50 % стипендиального фонда. Назначьте стипендию всем студентам. Составьте отчеты изменения размера стипендии по трем категориям студентов (по успеваемости).

Вариант 13

Дополнительно факультету выделено 50 % стипендиального фонда. Распределите надбавку среди отличников. Составьте диаграмму стипендиальных фондов каждой группы и курса.

Вариант 14

Дополнительно факультету выделено 50 % стипендиального фонда. Распределите надбавку среди успевающих. Составьте диаграмму стипендиальных фондов каждой группы и курса.

Вариант 15

Дополнительно факультету выделено 50 % стипендиального фонда. Распределите надбавку всем студентам групп. Составьте диаграмму стипендиальных фондов каждой группы и курса.

Дополнительная литература

1. *Вемпен Ф.* Microsoft Office Professional: 6 книг в одной: Пер. с англ. — М.: Бином, 1977.
2. *Джонс Э., Саттон Д.* Библия пользователя Office 97: Пер. с англ. — Киев: Диалектика, 1997.
3. *Ермолович Е.А., Макарова С.В., Хегай Л.Б.* Операционные системы и информационные технологии. — Красноярск, 2000.
4. *Ефимова О., Морозов В., Шафрин Ю.* Курс компьютерной технологии. — М.: ФБФ, 1998.
5. *Ефимова О., Морозов В., Шафрин Ю.* Практикум по компьютерной технологии. — М.: ФБФ, 1998.
6. *Лядова Л.Н.* Персональный компьютер: От начинающего пользователя до профессионала: В 2 т. — Пермь: ПГУ, 1998.
7. *Макарова Н.В. и др.* Информатика. Практикум по технологии работы на компьютере. — М.: Финансы и статистика, 1998.
8. *Персон Р.* Microsoft Excel 97 в подлиннике: В 2 т. — СПб.: BHV—Санкт-Петербург, 1997.
9. *Семенщиков А.* Microsoft Excel. Приемы и методы практического программирования. — Брянск, 1998.

Тесты к главе 2

Операционные системы

1. В состав программного обеспечения ЭВМ **не входят:**
   1) системы программирования; 2) операционные системы;
   3) аппаратные средства; 4) прикладные программы.
2. Операционная система представляет из себя:
   1) комплекс программ специального назначения;
   2) комплекс аппаратных средств;
   3) совокупность ресурсов компьютера;
   4) комплекс инструментальных программ.
3. ОС MS DOS является:
   1) однопользовательской, однозадачной;
   2) однопользовательской, многозадачной;
   3) многопользовательской, однозадачной;
   4) многопользовательской, многозадачной.
4. Директорий в ОС MS DOS может содержать символов в своем полном имени:
   1) 11; 2) 8; 3) 7; 4) 12.
5. Назначение оболочек операционных систем:
   1) защита операционной системы;
   2) предоставление возможности написания программ;
   3) облегчение взаимодействия пользователя с компьютером;
   4) перечислены в пунктах 1—3.
6. Поименованная совокупность данных, хранимая во внешней памяти, — это:
   1) файловая система; 2) директорий; 3) файл; 4) запись.
7. Принципиальным отличием ОС Windows от ОС MS DOS является:
   1) многозадачность;

2) возможность обмена данными между работающими программами;
3) графический интерфейс;
4) перечислены в п. 1—3.

8. Командный процессор — это:
1) ресурс; 2) устройство; 3) программа; 4) часть центрального процессора.

9. Интерпретатором команд MS DOS является файл с именем:
1) AUTOEXEC.bat; 2) MS DOS.sys;
3) CONFIG.sys; 4) COMMAND.com.

10. Основными компонентами в составе ОС являются:
1) утилиты, командный процессор, ядро;
2) резидентные программы, утилиты;
3) утилиты, командный процессор, центральный процессор;
4) резидентные программы, ядро, командный процессор.

11. В ОС Unix реализован механизм несмежного распределения блоков файлов. Какому элементу списка, находящегося в дескрипторе файла, принадлежит первый уровень косвенной адресации:
1) 10-му; 2) 11-му; 3) 12-му; 4) 13-му.

12. Системной причиной прерываний первого рода является:
1) необходимость синхронизации между параллельными процессами;
2) потребность активного процесса в некотором ресурсе;
3) получение запроса на прерывание от пользователя;
4) окончание интервала мультиплексирования.

13. Системной причиной прерываний второго рода является:
1) необходимость синхронизации между параллельными процессами;
2) потребность активного процесса в некотором ресурсе;
3) получение запроса на прерывание от пользователя;
4) окончание интервала мультиплексирования.

14. Если слева от раскрытой папки в ОС Windows изображен знак «+», то это означает, что:
1) в папке есть файлы;
2) в папке есть папки;
3) в папке есть непустые файлы;
4) в папку можно добавлять файлы.

15. Символ «?», используемый при написании имени файла в ОС MS DOS:
1) заменяет один произвольный символ;
2) заменяет произвольное число произвольных символов;
3) заменяет расширение файла;
4) указывает на то, что путь к файлу не известен.

16. В Norton Commander при нажатии функциональной клавиши F4:
1) происходит вызов контекстной подсказки;
2) вызывается простейший редактор;
3) архивируется указанный файл;
4) создается директорий на активной панели.

17. Исполняемыми в ОС MS DOS являются файлы с расширениями:
1) com, pas, exe; 2) bat, exe, doc;
3) pas, bat, com; 4) bat, exe, com.

18. Скобки [ ] в описании формата команда ОС MS DOS:
1) указывают на возможность отсутствия фрагмента;
2) предназначены для обозначения ключей;
3) предназначены для обозначения атрибутов;

4) как правило, содержат имя и путь файла.

19. Чтобы отключить панели в Norton Commander, можно нажать клавиши:
1) <Alt>+<F1>; 2) <Alt>+<F2>; 3) <Ctrl>+<O>; 4) <Ctrl>+<S>.

20. Для того чтобы команда COPY в ОС MS DOS не запрашивала подтверждения при замене существующих файлов, нужно набрать ключ:
1) [/S]; 2) [/Y]; 3) [/Q]; 4) [/X].

21. Команда «DIR» с ключом [/P] в ОС MS DOS:
1) выводит список файлов и каталогов, пока экран не заполнится, для получения следующих экранов нужно нажимать любую клавишу;
2) выводит информацию в сокращенном виде — только имена файлов и директориев;
3) выводит только скрытые файлы;
4) выводит все файлы, кроме системных.

22. Если необходимо вызвать имя файла в командную строку для формирования параметров команды в Norton Commander, нужно, выделив имя файла указателем, нажать одновременно:
1) <Alt> и <F1>; 2) <Ctrl> и <Enter>;
3) <Ctrl> и <S>; 4) <Ctrl> и <Tab>.

23. Для того чтобы команда «DIR» в ОС MS DOS выводила только скрытые файлы, нужно набрать ключ:
1) /P[[:] H]; 2) /A[[:] S]; 3) /A[[:] H]; 4) /A[[:]-H].

24. Для вызова окна установки атрибутов файла в Norton Commander следует нажать:
1) Alt-E; 2) Ctrl-F1; 3) Alt-Q; 4) Shift-F3.

Системы программирования

1. Все существующие языки программирования делятся на:
1) функциональные и логические; 2) русско- и нерусскоязычные;
3) процедурные и непроцедурные; 4) языки низкого и высокого уровня.

2. Выберите верное утверждение:
1) компиляторы делятся на трансляторы и интерпретаторы;
2) трансляторы делятся на компиляторы и интерпретаторы;
3) интерпретаторы делятся на трансляторы и компиляторы;
4) перевод текста программы в машинный код осуществляется либо компилятором, либо транслятором.

3. В главном меню систем программирования Турбо установку параметров отладки программы позволяет делать команда:
1) tools; 2) run; 3) window; 4) debug.

4. Язык программирования Си является:
1) непроцедурным; 2) процедурным;
3) функциональным; 4) логическим.

5. Транслятор — это программа, которая:
1) переводит текст программы в машинный код;
2) предоставляет средства для просмотра и изменения значений переменных;
3) подключает к исходному объектному модулю объектные модули соответствующих подпрограмм;
4) распознает и выполняет команды программы.

6. Процедура очистки экрана в системе Турбо-Паскаль входит в модуль:
1) Graph; 2) Crt; 3) String; 4) Turbo3.

7. Непроцедурным языком **не является**:
1) Лисп; 2) Кобол; 3) Оккам; 4) Смолтолк.
8. При вычислении какого выражения транслятор обратится к подпрограмме:
1) $(x + y)(x + y)$; 2) sqr$(x + y)$;
3) $(x + y)/N$; 4) $xy - x/y$.
9. Компоновкой называется:
1) процесс описания переменных в программе;
2) проверка, не нарушены ли формальные правила, содержащиеся в данном языке программирования;
3) просмотр и изменение значений переменных в ходе отладки программы;
4) подключение к исходному объектному модулю объектных модулей соответствующих подпрограмм.
10. **Неверным** является утверждение:
1) системный диск может не содержать файл config.sys;
2) файл autoexec.bat не может не содержать ни одного байта;
3) система программирования может не содержать транслятора;
4) файл с расширением txt может быть не текстовым.
11. Минимальный состав системы программирования, необходимый для работы программы, включает:
1) транслятор, отладчик, макроассемблер, средства редактирования, компоновки, загрузки;
2) транслятор, отладчик, макроассемблер;
3) транслятор, отладчик, макроассемблер, командный процессор;
4) транслятор, отладчик.
12. Какой из представленных процессов трансляции имеет верный порядок:
1) синтаксический анализ, семантический анализ, компиляция, компоновка;
2) синтаксический анализ, лексический анализ, интерпретация, компоновка;
3) лексический анализ, семантический анализ, компоновка, загрузка;
4) синтаксический анализ, трансляция, компоновка, загрузка.

Текстовые редакторы и издательские системы

1. Текстовый редактор Word — это:
1) прикладная программа; 2) базовое программное обеспечение;
3) сервисная программа; 4) редактор шрифтов.
2. Издательская система представляет собой:
1) систему управления базой данных; 2) операционную оболочку;
3) комплекс аппаратных и программных средств; 4) графический редактор.
3. Под термином «кегль» понимают:
1) размер полосы набора; 2) размер шрифта;
3) расстояние между строками; 4) начертание шрифта.
4. Гарнитурой называется:
1) оптимальная пропорция издания;
2) совокупность шрифтов одного рисунка во всех начертаниях и кеглях;
3) совокупность элементов, из которых строится буква;
4) расстояние между нижним и верхним выносными элементами.
5. Шрифт, которым набираются формулы в издании, всегда совпадает со шрифтом основного текста:
1) по наклону очка; 2) по размеру;
3) по гарнитуре; 4) по насыщенности.

6. Фронтиспис — это:
1) иллюстрация, помещаемая в начале издания перед титулом;
2) начальная страница издания, предшествующая титулу;
3) страница издания, на которой размещаются основные библиографические данные;
4) добавочный титул, который помещается перед отдельными частями издания и содержит номер и название части.

7. Выберите верное утверждение:
1) аннотации набирают шрифтом той же гарнитуры и кегля, что и основной текст;
2) знак охраны авторского права помещают в нижнем правом углу полосы набора;
3) на одной странице рекомендуется использовать не более одной гарнитуры и не более четырех кеглей;
4) шрифты кегля 9 рекомендуются для малоформатных справочников и словарей.

8. Выберите верное утверждение:
1) при наборе титула желательно отдавать предпочтение шрифтам жирного начертания;
2) библиотечный индекс помещают в нижнем левом углу полосы набора;
3) комплексный книготорговый индекс помещают в нижнем правом углу полосы набора;
4) на одной странице рекомендуется использовать не более трех гарнитур и не более трех кеглей.

9. В текстовом редакторе Лексикон выход в меню осуществляется нажатием функциональной клавиши:
1) F1; 2) F4; 3) F9; 4) F10.

10. В текстовых редакторах и настольных издательских системах, как правило, с помощью клавиш Alt + F4 происходит:
1) переход в окно с предыдущей программой;
2) переход в окно со следующей программой;
3) открытие файла;
4) выход из программы.

11. В текстовых редакторах и настольных издательских системах, как правило, с помощью клавиш Ctrl + S происходит:
1) создание нового документа; 2) открытие файла;
3) сохранение файла; 4) печать файла.

12. Следующая последовательность действий:
установить указатель мыши на полосе выделения рядом с текстом;
нажать левую клавишу мыши и, удерживая ее, передвигать мышь в нужном направлении
в Word приведет:
1) к выделению текста; 2) к удалению текста;
3) к перемещению текста; 4) к копированию текста в буфер.

13. Следующая последовательность действий:
выделить нужный участок текста;
нажать на нем левую клавишу мыши и, удерживая ее, передвигать мышь до нужного места
в Word приведет:
1) к копированию выделенного участка текста;

2) к переносу выделенного участка текста;
3) к замене текущего текста на выделенный;
4) к удалению выделенного участка текста в буфер.

14. Абзацные отступы и ширина колонок могут изменяться в Word с помощью:
1) линейки прокрутки; 2) координатной линейки;
3) строки состояния; 4) поля пиктограмм.

15. При нажатии на кнопку с изображением изогнутой влево стрелки на панели пиктографического меню в Word:
1) появляется диалоговое окно для добавления гиперссылки;
2) отменяется последняя команда;
3) происходит разрыв страницы;
4) повторяется последняя команда.

16. При нажатии на кнопку с изображением дискеты на панели пиктографического меню в Word происходит:
1) считывание информации с дискеты; 2) запись документа на дискету;
3) сохранение документа; 4) печать документа.

17. При нажатии на кнопку с изображением ножниц на панели пиктографического меню в Word:
1) происходит разрыв страницы;
2) вставляется вырезанный ранее текст;
3) удаляется выделенный текст;
4) появляется схема документа, разбитого на страницы.

18. Какую комбинацию «горячих клавиш» нужно нажать в Word, чтобы вставить скопированный блок текста без использования пиктограмм:
1) Ctrl + C; 2) Shift + Enter; 3) Ctrl + E; 4) Ctrl + V?

19. Какую комбинацию «горячих клавиш» нужно нажать в Word, чтобы выделить весь файл без использования пиктограмм:
1) Ctrl + S; 2) Ctrl + B; 3) Shift + Insert; 4) Ctrl + A?

20. Какую комбинацию «горячих клавиш» нужно нажать в Word, чтобы вставить в текст гиперссылку без использования пиктограмм:
1) Alt + G; 2) Ctrl + K; 3) Shift + V; 4) Ctrl + C?

Графические системы

1. Способ реализации построения изображений на экране дисплея, при котором электронный луч поочередно рисует на экране различные знаки — элементы изображения, называется:
1) растровым; 2) векторным; 3) лучевым; 4) графическим.

2. Редактор PaintBrush используется:
1) для работы базы данных; 2) для создания звуковых сигналов;
3) для создания текстовых документов; 4) для создания рисунков.

3. Способ реализации построения изображений на экране дисплея, при котором изображение представлено прямоугольной матрицей точек, имеющих свой цвет из заданной палитры, называется:
1) растровым; 2) мозаичным; 3) пиксельным; 4) графическим.

4. Бесконечный ряд равных плоских фигур, расположенных друг за другом таким образом, что элементарная конечная фигура переносится вдоль одного измерения бесконечно, называется:
1) орнаментом; 2) бордюром;
3) паркетом; 4) огранкой.

5. Какое количество типов симметрии плоских орнаментов существует:
1) 7; 2) 17; 3) 24; 4) 12?

6. Представление относительных величин объектов, которым на изображении сопоставляют размеры и расположение кругов в прямоугольной системе координат, называется:
1) гистограммой; 2) структурной схемой;
3) круговой гистограммой; 4) круговой диаграммой.

7. Отображение исходных величин в виде точек, соединенных отрезками прямых линий, называется:
1) структурной схемой; 2) временной диаграммой;
3) гистограммой; 4) линейным графиком.

8. Автокад — это:
1) АСНИ; 2) САПР; 3) АСУ; 4) АСУ ТП.

9. Что характеризует общие принципы строения, целостность предмета, лежит в основе ритма, гармонии, ансамбля в архитектуре:
1) мера; 2) гармония; 3) симметрия; 4) композиция?

10. Для того чтобы изобразить дугу по начальной точке, центру и длине хорды в программе АВТОКАД, необходимо выбрать режим:
1) AUTOCAD; 2) BLOCKS; 3) DRAW; 4) DISPLAY.

11. Набор чисел, логических параметров, играющих роль коэффициентов в уравнениях, задающих графический объект заданной формы в научной графике, называют:
1) координатными моделями; 2) аналитическими моделями;
3) приближенными моделями; 4) демонстрационными моделями.

Системы управления базами данных

1. Структура данных, для которой характерна подчиненность объектов нижнего уровня объектам верхнего уровня, называется:
1) табличной; 2) реляционной; 3) иерархической; 4) сетевой.

2. Реализованная с помощью компьютера информационная структура, отражающая состояние объектов и их отношения, — это:
1) база данных; 2) информационная структура;
3) СУБД; 4) электронная таблица.

3. В общий набор рабочих характеристик БД **не входит**:
1) полнота; 2) правильная организация;
3) актуальность; 4) единичность отношений.

4. СУБД состоит из:
1) ЯОД, ЯМД; 2) ЯОД, ЯМД, СПО;
3) ЯОД, СПО, СВД; 4) ЯОД, ЯМД, СВД.

5. Отличительной чертой реляционной базы данных является:
1) подчиненность объектов нижнего уровня объектам верхнего уровня;
2) то, что отношения между объектами определяются как «многие ко многим»;
3) то, что каждая запись в таблице содержит информацию, относящуюся только к одному конкретному объекту;
4) возможность поиска данных по ключу.

6. ЯОД является языком:
1) низкого уровня; 2) высокого уровня;
3) функциональным; 4) логическим.

7. ЯОД предназначен:
   1) для формализованного описания типов данных, их структур и взаимосвязей;
   2) для выполнения операций с БД (наполнения, удаления, поиска, обновления);
   3) для защиты данных БД от постороннего вмешательства;
   4) для поддержания дружественности интерфейса.
8. ЯМД **не предназначен**:
   1) для наполнения БД, удаления данных из БД;
   2) для обновления БД;
   3) для выборки информации из БД;
   4) для описания типов данных БД.
9. Макрос — это:
   1) объект, представляющий собой структурированное описание одного или нескольких действий;
   2) часть командного процессора;
   3) язык программирования;
   4) текстовый редактор.
10. К основным функциям СУБД **не относится**:
   1) определение данных; 2) хранение данных;
   3) обработка данных; 4) управление данными.
11. СПО в составе СУБД нужны:
   1) для создания и распечатки сводок по заданным формам на основе информации БД;
   2) для формализованного описания типов данных, их структур и взаимосвязей;
   3) для защиты данных БД от постороннего вмешательства;
   4) для организации связей между таблицами через общие атрибуты.
12. Тип Memo поля Type в СУБД DBASE существует:
   1) для поля дат; 2) для поля памяти;
   3) для числовых полей; 4) для логических полей.
13. С помощью какой из перечисленных команд СУБД DBASE невозможно редактировать базу данных:
   1) Change; 2) Edit; 3) Display; 4) Delete?
14. Команда Join to в СУБД DBASE позволяет:
   1) соединять целые БД; 2) передавать данные из одной БД в другую;
   3) осуществлять поиск в базе; 4) индексировать БД.
15. Ввод данных СУБД DBASE осуществляется командами:
   1) Input; 2) Read; 3) Update; 4) команды 1) и 2).
16. Поле Field name в СУБД DBASE **не должно содержать**:
   1) знака подчеркивания; 2) пробела; 3) цифр; 4) скобок и кавычек.
17. Пусть имеется отношение «Успеваемость» со следующей схемой:

```
Успеваемость (ФИО_студента, Дисциплина, Оценка, Дата, Преподаватель).
```

Результатом выполнения следующей команды

```
select unique ФИО_студента from Успеваемость
```

будет:
   1) все отношение «Успеваемость»;
   2) ФИО, Дисциплина, Оценка, Дата, Преподаватель тех студентов, которые не имеют однофамильцев;
   3) все отношение «Успеваемость», отсортированное по полю ФИО_студента;
   4) список всех студентов из отношения «Успеваемость».

18. Файлы отчетов в СУБД DBASE имеют расширение:
1) PRG; 2) DBF; 3) FRM; 4) RDX.
19. Команды, обеспечивающие поиск записей в СУБД DBASE:
1) Find, locate, seek; 2) Find, locate, quit;
3) Locate, seek, index; 4) Use, locate, index.
20. Чтобы изменить структуру или шаблон формы в СУБД Access, нужно открыть форму в режиме:
1) таблицы; 2) конструктора; 3) формы; 4) предварительного просмотра.
21. В СУБД Access допустимы типы полей записей:
1) числовой, символьный, графический, музыкальный;
2) логический, дата, числовой, денежный, OLE;
3) числовой, текстовый, гипертекстовый, логический;
4) числовой, символьный, Memo, дата, логический, массив.
22. В поле OLE (СУБД Access) можно разместить:
1) файл; 2) число; 3) ссылку на другую таблицу; 4) калькулятор.
23. Группа пиктограмм, главное назначение которой — ускоренный вызов команд меню в СУБД Access, — это:
1) полоса меню; 2) строка состояния;
3) панель инструментов; 4) вкладки объектов.
24. Служебные слова order by <атрибут> asc определяют в SQL:
1) сортировку результата выборки в порядке возрастания;
2) сортировку результата выборки в порядке убывания;
3) группировку данных по значениям;
4) подсчет количества записей в таблице.
25. Команда выборки в SQL может содержать слова:
1) select, from, where; 2) order by, group by;
3) having set, minus; 4) все перечисленные в 1—3.

Электронные таблицы

1. Основное отличие электронных таблиц от реляционных БД:
1) приспособленность к расчетам; 2) структуризация данных;
3) табличное представление данных; 4) свойства, перечисленные в 1, 2.
2. SuperCalc, QuattroPro, Excel — это:
1) графические редакторы; 2) СУБД;
3) текстовые редакторы; 4) электронные таблицы.
3. В ячейку электронной таблицы **нельзя** ввести:
1) текст; 2) формулу; 3) иллюстрацию; 4) число.
4. Операндами формулы в электронных таблицах могут быть:
1) математические функции; 2) константы;
3) номера ячеек; 4) все перечисленное в 1—3.
5. Табличный процессор SuperCalc написан для среды:
1) DOS; 2) Windows;
3) Unix; 4) операционные системы, перечисленные в 1—3.
6. Пользователь может сортировать в электронной таблице:
1) клетки; 2) строки клеток;
3) столбцы клеток; 4) все перечисленное в 1—3.
7. Электронные таблицы SuperCalc имеют по умолчанию стандартное расширение:
1) Dbf; 2) Cal;
3) Htm; 4) Elt.

8. Электронные таблицы SuperCalc:
1) могут создавать базы данных;
2) имеют простейшие средства, характерные для БД;
3) располагают средствами для перевода информации к виду, доступному из текстовых редакторов;
4) располагают возможностями 1–3.

9. Абсолютный адрес в электронных таблицах SuperCalc — это:
1) расстояние от клетки, содержащей формулу, до клетки, на которую в ней имеется ссылка;
2) диапазон клеток, содержащих макрос;
3) адрес, в котором не перенастраиваются номера строк и столбцов;
4) полный адрес, указывающий номера строки и столбца клетки.

10. DENTRY, LEARN и DIRECT в электронных таблицах SuperCalc—это:
1) команды; 2) режимы создания макросов;
3) операнды; 4) служебные слова.

11. Рабочая страница в электронной таблице Excel содержит:
1) программное окно; 2) внутреннее окно;
3) нижнее окно; 4) правое окно.

12. Graph-Type, Time-Labs, Var-Labs в электронной таблице SuperCalc — это:
1) команды графического меню; 2) режимы создания макросов;
3) операнды; 4) типы переменных.

13. Ввод повторяющегося текста в электронной таблице SuperCalc начинается со знака:
1) «:»; 2) «'»; 3) «—»; 4) «/».

14. Сортировка таблицы в SuperCalc осуществляется с помощью команды:
1) Arrange; 2) Sort; 3) Format; 4) Protect.

15. Команда Alt + A в электронных таблицах SuperCalc:
1) фиксирует заголовок таблицы;
2) завершает сеанс работы с программой;
3) запускает на выполнение макрос \a;
4) записывает макрос \a.

16. Создание формулы в электронной таблице Excel начинается с ввода знака:
1) «:»; 2) «=»; 3) «/»; 4) «\».

Инструментальные ПС и интегрированные пакеты

1. MS Works — это:
1) ПС специального назначения; 2) экспертная система;
3) интегрированный пакет; 4) авторская система.

2. MATHCAD — это:
1) прикладная программа; 2) экспертная система;
3) ПС общего назначения; 4) интегрированная система.

3. REDUCE — это:
1) ПС общего назначения; 2) ПС специального назначения;
3) ПС профессионального уровня; 4) интегрированный пакет.

4. Отличием интегрированных пакетов от специализированных инструментальных программных средств является:
1) единый интерфейс всех ПС; 2) наличие табличного процессора;
3) ограниченность команд обработки БД; 4) свойства 1 и 3.

5. Система REDUCE позволяет:
1) проводить вычисления в аналитическом виде;
2) производить численные операции;
3) решать обе вышеперечисленные задачи;
4) работать с электронными таблицами.

6. Если переменной не присвоено какое-либо значение в системе REDUCE, то:
1) ее значением становится ее имя; 2) ей присваивается ноль;
3) ей присваивается пустое множество; 4) такая ошибка недопустима.

7. Документ пакета MATHCAD может совмещать:
1) текст и формулы;
2) графики и формулы;
3) текст, графики и формулы;
4) либо текст, либо графики и формулы.

8. Чтобы получить символьный вывод значения переменной, функции, выражения в среде MATHCAD, нужно закончить выражение знаком:
1) «=»; 2) «стрелка вправо»; 3) «:»; 4) «<=>».

9. Для перехода в главное меню системы MS Works нужно:
1) нажать клавишу Alt; 2) набрать команду Menu;
3) нажать клавишу F10; 4) применить один из способов — 1 или 3.

10. Для выхода из текстового редактора системы MS Works нужно:
1) нажать клавишу Esc; 2) нажать клавишу F10;
3) нажать комбинацию клавиш Alt + X; 4) нажать клавишу X.

11. В электронных таблицах системы MS Works формулы начинаются со знака:
1) «!»; 2) «=»;
3) «#»; 4) «/».

12. Запуск программы на выполнение в системе REDUCE осуществляется:
1) командой Run; 2) клавишей «#»;
3) комбинацией символов «(» и «)»; 4) клавишами Ctrl + F9.

13. Кнопка m..n 1-й палитры пакета MATHCAD служит для:
1) задания размерности матриц;
2) задания границ интегрирования;
3) табуляции функций и выражений;
4) задания количества строк и столбцов электронной таблицы.

14. Для того чтобы результат выражения вывести на экран в системе REDUCE, нужно закончить его символом:
1) «:»; 2) «;»; 3) «=»; 4) «#».

15. Чтобы обратить матрицу A в системе MATHCAD, нужно набрать:
1) A^−1=; 2) inv A; 3) A−1=; 4) matr−1(A).

16. Для перехода к следующей записи в СУБД системы MS Works нужно нажать:
1) Shift + Tab; 2) Ctrl + PgDn; 3) Tab; 4) способами 2 и 3.

17. Выберите листинг следующей программы в системе REDUCE:

```
A:=S# A; A:=X*Y# A; Q:=X:=Y# Q; X; (и)
```

1) `A:=S A=X2*Y Q=X=Y u=0`; 2) `S X*Y Y Y`;
3) `S X*Y X=Y Y`; 4) `u=o`.

18. Стандартная форма отчета СУБД системы MS Works позволяет вычислять для полей следующие значения:
1) сумма;
2) сумма, минимальное значение;
3) сумма, минимальное значение, максимальное значение;
4) сумма, минимальное значение, максимальное значение, среднее значение.

19. Следующий оператор в системе MATHCAD root(t — cosh(t), t)= выдаст на выходе:

1) производную выражения t — cosh(t) по переменной t;
2) корень степени t из выражения t — cosh(t);
3) численное решение уравнения t — cosh(t) = 0;
4) корни уравнения t — cosh(t) = t.

Компьютерное тестирование

1. Компьютерное тестирование — это:
   1) процесс выставления оценки знаний учащегося;
   2) процесс оценки соответствия личностной модели знаний и экспертной модели знаний;
   3) процесс оценки компетентности преподавателя;
   4) процесс выяснения соотношения объемов усвоенного и неусвоенного материала.
2. Набор взаимосвязанных тестовых заданий, позволяющих оценить уровень соответствия знаний ученика экспертной модели знаний предметной области, — это:
   1) тест; 2) тестовое пространство;
   3) тестовая программа; 4) база данных тестовых заданий.
3. Класс эквивалентности — это:
   1) множество тестовых заданий по всем модулям экспертной модели знаний;
   2) множество тестовых заданий, таких, что выполнение одного из них учеником гарантирует выполнение других;
   3) множество тестовых заданий, имеющих одинаковое время выполнения;
   4) множество тестовых заданий из одного раздела темы.
4. Какая из перечисленных форм представления тестовых заданий наиболее неудобна для компьютерного тестирования:
   1) закрытая; 2) открытая; 3) с фасетом; 4) задание на соответствие?
5. Тестовая оболочка — это:
   1) внешний вид тестовой программы, служащий для обеспечения диалога с тестируемым;
   2) информационная структура, хранящая всю базу тестовых заданий;
   3) программа, создающая компьютерные тесты, формируя базу данных из набора тестовых заданий;
   4) файл, в котором сохраняются ответы тестируемого.
6. Выбор типов тестов определяется:
   1) особенностями инструментальных тестовых программ;
   2) особенностями предметной области;
   3) опытом и мастерством экспертов;
   4) факторами, перечисленными в пп. 1—3.

Компьютерные игры

1. Компьютерные игры, основанные на управлении игровыми объектами, называются:
   1) играми на мастерство; 2) азартными играми;
   3) логическими играми; 4) обучающими играми.
2. Компьютерные игры, содержащие стратегию поведения игрока, называются:
   1) играми на мастерство; 2) азартными играми;

3) логическими играми; 4) обучающими играми.

3. Компьютерные игры, в которых исход в большей степени зависит от случайности, называются:
1) играми на мастерство; 2) азартными играми;
3) логическими играми; 4) обучающими играми.

4. К компьютерным играм на мастерство **не относятся**:
1) футбол; 2) звездные войны; 3) тетрис; 4) шахматы.

5. К логическим компьютерным играм **не относятся**:
1) шахматы; 2) крестики-нолики; 3) сапер; 4) покер.

6. К логическим компьютерным играм на мастерство **не относятся**:
1) покер; 2) «поле чудес»; 3) карате; 4) шахматы.

7. Отображение всех перемещений и изменений на экране дисплея в компьютерной игре является результатом действия:
1) оперативного уровня игры; 2) тактического уровня игры;
3) стратегического уровня игры; 4) технического уровня игры.

8. Изменение сложности компьютерной игры, темпа, уровня происходят на этапе:
1) оперативного уровня игры; 2) тактического уровня игры;
3) стратегического уровня игры; 4) технического уровня игры.

Компьютерные вирусы

1. Антивирусные средства предназначены:
1) для тестирования системы;
2) для защиты программ от вируса;
3) для проверки программ на наличие вируса и их лечения;
4) для мониторинга системы.

2. Какое из следующих качеств необязательно присуще программе-вирусу:
1) самостоятельно запускается;
2) присоединяет свой код к кодам других программ;
3) занимает малый объем памяти;
4) приводит к потере информации.

3. Не существует следующего понятия:
1) антивирусное средство «сторож»; 2) антивирусное средство «фаг»;
3) сетевой вирус; 4) загрузочно-файловый вирус.

4. В классификации компьютерных вирусов нет разновидности:
1) драйверные вирусы; 2) файловые вирусы;
3) загрузочно-драйверные вирусы; 4) загрузочно-файловые вирусы.

5. Какая из ниже перечисленных программ не является антивирусным средством:
1) Aidstest; 2) Doctor Web;
3) VSAFE; 4) Vsearch.

6. Вирусы, которые в простейшем случае заражают пополняемые файлы, но могут распространяться и через файлы документов, — это:
1) файловые вирусы; 2) загрузочно-файловые вирусы;
3) это качество вирусов и 1, и 2; 4) драйверные вирусы.

7. Вирусы, запускающие себя путем включения в файл конфигурации дополнительной строки, называются:
1) файловые вирусы;
2) загрузочно-файловые вирусы;
3) сетевые вирусы;
4) драйверные вирусы.

8. Вирусы, заражающие программу начальной загрузки компьютера, хранящуюся в загрузочном секторе дискеты или винчестера и запускающиеся при загрузке компьютера, — это:

1) загрузочные вирусы; 2) загрузочно-файловые вирусы;
3) это качество вирусов и 1, и 2; 4) драйверные вирусы.

9. Антивирусная программа, контролирующая возможные пути распространения программ-вирусов и заражения компьютеров, называется:

1) детектором; 2) фагом;
3) сторожем; 4) ревизором.

10. Антивирусное средство, способное только обнаруживать вирус, называется:

1) детектором; 2) фагом;
3) сторожем; 4) ревизором.

11. Резидентная программа, постоянно находящаяся в памяти компьютера и контролирующая операции, связанные с изменением информации на магнитных дисках, называется:

1) детектором; 2) фагом; 3) сторожем; 4) ревизором.

Правильные ответы

Операционные системы

№	1	2	3	4	№	1	2	3	4
1			X		13	X			
2	X				14		X		
3	X				15	X			
4		X			16		X		
5			X		17				X
6			X		18	X			
7				X	19			X	
8			X		20		X		
9				X	21	X			
10	X				22		X		
11		X			23			X	
12		X			24	X			

Системы программирования

№	1	2	3	4	№	1	2	3	4
1			X		7		X		
2		X			8		X		
3				X	9				X
4		X			10			X	
5	X				11	X			
6		X			12	X			

Текстовые редакторы и издательские системы

№	1	2	3	4	№	1	2	3	4
1	X				11			X	
2			X		12	X			
3		X			13		X		
4		X			14		X		
5			X		15		X		
6	X				16			X	
7		X			17			X	
8				X	18				X
9				X	19				X
10				X	20		X		

Графические системы

№	1	2	3	4
1		X		
2				X
3	X			
4		X		
5		X		
6			X	
7				X
8		X		
9	X			
10			X	
11		X		

Системы управления базами данных

№	1	2	3	4	№	1	2	3	4
1			X		14	X			
2	X				15				X
3				X	16		X		
4		X			17				X
5			X		18			X	
6		X			19	X			
7	X				20		X		
8				X	21		X		
9	X				22	X			
10		X			23			X	
11	X				24		X		
12		X			25				X
13			X						

Электронные таблицы

№	1	2	3	4	№	1	2	3	4
1				X	9			X	
2				X	10		X		
3			X		11		X		
4				X	12	X			
5	X				13		X		
6				X	14	X			
7		X			15			X	
8				X	16		X		

Инструментальные ПС и интегрированные пакеты

№	1	2	3	4	№	1	2	3	4
1			X		11		X		
2	X				12			X	
3			X		13			X	
4				X	14		X		
5			X		15	X			
6	X				16				X
7			X		17		X		
8		X			18				X
9	X				19			X	
10				X					

Компьютерное тестирование

№	1	2	3	4	№	1	2	3	4
1		X			4		X		
2	X				5			X	
3		X			6				X

Компьютерные игры

№	1	2	3	4
1	X			
2			X	
3		X		
4				X
5				X
6			X	
7	X			
8			X	

Компьютерные вирусы

№	1	2	3	4	№	1	2	3	4
1			X		7				X
2				X	8			X	
3		X			9				X
4			X		10	X			
5				X	11			X	
6			X						

Глава 3

ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

В данном разделе практикума приведены материалы для проведения занятий по программированию на различных языках высокого уровня. Не исключено, что, наряду с данным практикумом, в учебном процессе по этому разделу могут использоваться сборники задач по программированию, которых (по крайней мере, по традиционным методологиям программирования) имеется большое число.

Хотя основу практикума по этому разделу составляют лабораторные работы, на которых студенты самостоятельно разрабатывают и реализуют (пользуясь консультациями преподавателей) программы на компьютере, это не исключает проведения семинарских занятий, на которых отрабатываются навыки решения типовых задач. Опыт показывает, что для многих студентов занятия по принципам алгоритмизации и программирования в аудитории, без компьютера, полезны и даже необходимы. Заметим в связи с этим, что в название раздела включено слово «методы» (а не просто «языки программирования»). Этим подчеркивается важность освоения именно методов разработки алгоритмов и программ, а не только кодирования на том или ином языке.

Основу практикума составляет программирование на трех языках: Паскаль, Си и Пролог. Наиболее полно отражен Паскаль, что соответствует его реальной роли в подготовке учителей информатики и ряду других специальностей. На базе Паскаля предполагается отработка навыков по объектно-ориентированному программированию. На уровне получения начальных практических навыков реализован практикум по Си и Прологу. В то же время, в практикум не включены разделы по программированию на языках Бейсик и ЛИСП, описанных в базовом учебном пособии.

Вопрос о целесообразности проведения практикума по языку Бейсик, формам и объему этого практикума следует рассматривать в контексте постановки изучения программирования в конкретном вузе. Роль Бейсика в профессиональном программировании в настоящее время невелика (достаточно популярный объектно-ориентированный Visual Basic, по существу, есть иной язык). Чаще всего на Бейсик отпускается лишь немного времени для ознакомления (либо эта тема полностью исключается или переносится на самостоятельное изучение). Учитывая, что языки Бейсик (в его современных версиях типа QBasic) и Паскаль нацелены на решение достаточно схожего круга прикладных задач, весьма объемный практикум по Бейсику с решением задач и выполнением лабораторных работ может быть при желании организован с помощью заданий, приведенных в разделе по Паскалю.

Язык ЛИСП, реализующий принципиально иную, нежели Паскаль, парадигму программирования, в принципе заслуживает ознакомления, что и имели в виду авторы базового учебного пособия, включив в него соответствующую главу. Однако на практическое освоение этого языка при подготовке студентов той катего-

рии, которой адресован практикум, обычно не остается времени. Соответствующих требований нет и в государственных образовательных стандартах.

Важную роль в данном разделе практикума играет тема «Методы и искусство программирования». Именно на классических задачах поиска и сортировки и построения рекурсивных алгоритмов традиционно оттачиваются практические навыки будущего программиста в сфере алгоритмизации и программирования. Вопрос о том, на каких языках реализовывать соответствующие программы, вторичен. В данном случае это могут быть и Паскаль, и Си, и Бейсик.

§ 1. ПАСКАЛЬ КАК ЯЗЫК СТРУКТУРНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Рекомендации по проведению занятий

Паскаль чаще всего является основным языком, изучаемым в ходе подготовки учителей информатики. Это обусловливает особую роль в проведении лабораторно-практических занятий по программированию на Паскале. Если в отношении других языков ставятся в основном ознакомительные цели, то при освоении Паскаля требуется выработка устойчивых практических навыков программирования.

Это обусловливает и организацию занятий. Они включают практические (групповые) семинарские занятия, на которых отрабатываются навыки реализации типовых алгоритмов; для этой деятельности и для домашних заданий необходимо некоторое число традиционных задач. Не все эти задачи становятся объектом отладки на компьютере, поскольку это требует немалого времени.

По отдельным темам обычно проводятся контрольные работы, для которых также нужны задания. Следует отметить, что указанные оценки продолжительности выполнения контрольных работ, приведенные ниже, весьма условны. Они исходят из полного выполнения всех приведенных заданий и отладки программ на ЭВМ; и то, и другое необязательно, и реальный состав задания определяется преподавателем. Часть этих заданий может быть использована на лабораторных работах.

Практически полностью самостоятельная работа студентов, во время которой преподавателю отводится роль консультанта, реализуется в ходе выполнения лабораторных работ. Для этих работ необходимы наборы индивидуальных заданий, предусматривающих разработку типовых алгоритмов и программирование, с отладкой (и, в случае целесообразности, тестированием) программ, проведением по ним расчетов. Такие задания даются ниже по основным темам.

Практикум тесно привязан к базовому учебнику. В силу этого, иногда при разборе упражнений практикуются ссылки на программы, приведенные в учебнике.

Краткие сведения

Паскаль-программа является текстовым файлом с собственным именем и с расширением .pas. Схематически программа представляется в виде последовательности восьми разделов:

1. Заголовок программы.
2. Описание внешних модулей, процедур и функций.
3. Описание меток.

4. Описание констант.
5. Описание типов переменных.
6. Описание переменных.
7. Описание функций и процедур.
8. Раздел операторов.

Основные операторы языка

Реализация последовательности действий (структуры следования) выполняется с помощью *составного оператора*:

```
begin <последовательность операторов> end;
```

Для реализации развилки в Паскале предусмотрены два оператора: *условный оператор и оператор варианта* (выбора). Первый выглядит так:

```
if <логическое выражение> then <оператор1> else <оператор2>;
```

или

```
if <логическое выражение> then <оператор>;.
```

Оператор варианта имеет следующую форму:

```
case <выражение> of
   <список констант 1>: <оператор 1>;
   <список констант 2>: <оператор 2>;
   ..............................
   <список констант N>: <оператор N>
end;.
```

Для реализации *циклов* в Паскале имеются три оператора.

Цикл с предусловием:

```
while <логическое выражение> do <оператор>;.
```

Действие: вычисляется значение логического выражения. Если оно равно ***true***, то выполняется оператор, после чего снова вычисляется значение логического выражения, в противном случае действие заканчивается.

Цикл с постусловием:

```
repeat
<последовательность операторов>
until <логическое выражение>;
```

Действие: выполняется последовательность операторов. Далее вычисляется значение логического выражения. Если оно равно ***true***, то действие заканчивается, иначе снова выполняется последовательность операторов и т.д.

Цикл с параметром:

```
for <параметр>:= <выражение 1> to <выражение 2> do <оператор>
```

Параметр, выражение 1, выражение 2 должны быть одного ординального типа. Параметр в этом цикле возрастает.

```
for <параметр>:=<выражение 1> downto <выражение 2> do <оператор>.
```

Параметр в этом цикле убывает.

Простые типы

К ним относятся:
real — вещественный;
integer — целый;
boolean — логический;
byte — байтовый;
char — символьный.

Перечисляемые и интервальные типы данных задаются с помощью простых типов. Описание перечисляемого типа выполняется в разделе типов по схеме:

```
type <имя типа> = <список имен>
```

Примеры:

```
type operator=(plus,minus,multi,divide);
color=(white,red,blue,yellow,purple,green);
```

Интервальный тип — это подмножество другого уже определенного простого типа, называемого базовым. Пример:

```
type days = (mon,tue,wed,thu,fri,sat,sun);
workdays = mon..fri;
index = 1..30;
letter = 'a'..'z';
```

Можно задать интервал и в разделе переменных:

```
var a:1..100; b: -25..25;.
```

Составные типы

Массив — это последовательность, состоящая из фиксированного числа однотипных элементов:

```
type <имя типа> = array <список типов индексов> of <тип элементов>.
```

Число типов индексов называется размерностью массива. После описания типа массива конкретные массивы можно задать в разделе описания переменных, например:

```
type vector = array [1..10] of real;
table = array ['A'..'Z',1..5] of integer;
var a,b: vector;
c: table;
```

Описание массива—типа можно совместить с описанием соответствующих переменных:

```
var a,b: array [1..10] of real;
d: array [byte] of char; .
```

Строковый тип определяется в разделе описания типов, переменные этого типа – в разделе описания переменных:

```
type word: string[20];
var a,b,c: word;
```

или

```
var a,b,c: string[20];
d: string[30];
```

Для строковых величин определены четыре *стандартные функции*:

1. Функция соединения — **concat**(s1,s2,...,sk).
2. Функция выделения — **copy**(s,i,k). Из строки *s* выделяется *k* символов, начиная с *i*-го символа.
3. Функция определения длины строки — **length**(s).
4. Функция определения позиции — **pos**(s,t). Вычисляется номер позиции, начиная с которой строка *s* входит первый раз в строку *t*; результат равен 0, если строка *s* не входит в *t*.

Также определены четыре *стандартные процедуры* для обработки строковых величин:

1. Процедура удаления — **delete**(s,i,k). Из строки *s* удаляется *k* символов, начиная с *i*-го символа.
2. Процедура вставки — **insert**(s,t,i). Строка *s* вставляется в строку *t*, начиная с позиции *i*.
3. Процедура преобразования числа в строку символов — **str**(k,s).
4. Процедура преобразования строки из цифр в число — **val**(s,k,i).

Множественный тип можно определить в разделе описания типов по схеме:

```
type <имя> = set of <тип элементов>. Например:
type t = set of byte;
var a: t;
```

или

```
var code: set of 0..7;
digits: set of '0'..'9';
```

Для данных множественного типа определены операции *объединения, пересечения и дополнения* множеств, обозначаемые соответственно знаками +, * и −, а также отношения равенства множеств ($A = B$), неравенства ($A <> B$), включения ($A <= B$, $A >= B$). Логическая операция принадлежности: x **in** A принимает значение *true*, если элемент x принадлежит множеству A и *false* в противном случае.

Запись — это последовательность, состоящая из фиксированного числа величин разных типов, называемых *полями* или *компонентами записи*:

```
type имя типа записи = record
имя поля 1: тип;
имя поля 2: тип;
.................
имя поля N: тип
end;
```

Например, адресные данные (индекс, город, улица, номер дома, квартиры) можно представить как запись:

```
type address = record
index: string[6];
city: string[20];
street: string[20];
haus,flat: integer
end;
```

Подпрограммы

В Паскале подпрограммы называются *процедурами* и *функциями*.

Процедура имеет такую же структуру, как и программа, но с двумя отличиями:

1) заголовок процедуры имеет другой синтаксис:

procedure <имя> (<список описаний формальных параметров>)

2) описание процедуры заканчивается точкой с запятой (а не точкой).

Оператор вызова процедуры имеет вид

<имя процедуры> (<список выражений>);

Функция — это подпрограмма, определяющая единственное скалярное, вещественное или строковое значение. Отличия подпрограммы-функции от процедуры:

1) заголовок функции начинается со служебного слова **function** и заканчивается указанием типа значения функции:

function <имя> (<список описаний формальных параметров>): <тип>;

2) раздел операторов функции должен содержать хотя бы один оператор присваивания имени функции;

3) обращение к функции — не оператор, а выражение.

Функции и процедуры могут быть *рекурсивными*.

В Паскале имеются две встроенные *процедуры ввода*:

1) **read** <список переменных>;

2) **readln** <список переменных>.

Имеются две *процедуры вывода* на экран дисплея:

1) **write** <список выражений>;

2) **writeln** <список выражений>.

Модули

Модуль — это набор констант, типов данных, переменных, процедур и функций. Используется для создания библиотек и разделения больших программ на логически связанные не зависимые друг от друга составные части. В состав модуля входят следующие разделы: *заголовок, интерфейс, реализация, инициализация*. Заголовок необходим для ссылок на модуль. Интерфейс содержит объявления, включая процедуры и функции, представленные списком заголовков и доступные пользователям в теле основной программы. Раздел <реализация> содержит тела процедур и функций, перечисленных в интерфейсной части модуля. Раздел <инициализация> содержит операторы, необходимые для инициализации модуля.

Каждый модуль компилируется отдельно; результат компиляции — файл с расширением *.tpu* (TurboPascalUnit). Каждый элемент модуля можно использовать в программе пользователя без дополнительного объявления, для чего достаточно записать имя используемого модуля в директиве *Uses* в начале программы после его заголовка: **uses** <имя модуля>

Файлы

Для связи Паскаль-программы с внешними устройствами используют *файловые переменные*. Связь осуществляется оператором языка Паскаль:

assign (<имя файловой переменной>,'<имя устройства>').

Например:

```
assign (f, 'primer.dat')
```

Здесь *f* — имя файловой переменной, `primer.dat` — имя файла данных на внешнем носителе.

Если внешним устройством является принтер, то связь осуществляется оператором **assign**`(f,'lst:')`. Здесь `lst` — логическое имя печатающего устройства.

Ниже приведены логические имена внешних устройств ввода-вывода:

`con:`	консоль;	`trm:`	терминал;
`kbd:`	клавиатура;	`lst:`	принтер;
`aux:`	буфер сети;	`usr:`	драйвер пользователя.

После осуществления связи файловая переменная *f* отождествляется с соответствующим файлом.

Для работы с файлом его необходимо открыть, по окончании работы — закрыть. Файл открывается для чтения оператором **reset**`(f)`, для записи — оператором **rewrite**`(f)`. Чтение и запись данных осуществляется известными процедурами **read/write**, только в начале списка помещается имя файловой переменной:

```
read (f, <список ввода>); readln (f, <список ввода>);
write(f, <список вывода>); writeln(f, <список вывода>).
```

Закрытие файла осуществляется командой **close**`(f)`. Команда **reset**`(f)` устанавливает указатель маркера файла на нулевое состояние.

В системе Турбо-Паскаль предусмотрены встроенные функции по работе с файлами:

filesize`(f)` — возвращает текущее число компонент открытого файла;
filepos`(f)` — возвращает номер текущей позиции маркера;
rename`(f,имя)` — переименование файла, связанного с *f*;
erase`(f)` — уничтожение файла;
execute`(f)` — выполнение COM-файла;
chain`(f)` — выполнение CHN-файла;
seek`(f,N)` — устанавливает маркер на позицию N;
eof`(f)` — возвращает TRUE, если найден конец файла;
ealn`(f)` — возвращает TRUE, если найден конец строки.

Файловый тип данных в программе задается по следующему правилу:

```
type <имя файлового типа> = file of <тип компонентов>.
```

В качестве типа компонент файла разрешается использовать любой тип данных, кроме файлового. Например:

```
type intfile = file of integer;
```

Динамические переменные и указатели

Переменные *ссылочного типа* (указатели) вводятся в употребление обычным путем с помощью их описания в разделе переменных.

Значением указателя является адрес ячейки, начиная с которой будет размещена в памяти соответствующая динамическая величина.

Задание ссылочного типа выполняется по схеме:

```
type <имя ссылочного типа> = ^ <имя типа динамической величины>
```

(значок ^ указывает на то, что величина является динамической).

Например:

```
type p = ^integer;
   q = ^record
   x: integer;
   y: string [20]
end;
```

В некоторых случаях возникает необходимость в качестве значения указателя принять «пустую» ссылку **nil**, которая не связывает с указателем никакого объекта и принадлежит любому ссылочному типу.

Для порождения динамического объекта, на который указывает ссылочная переменная i, служит стандартная процедура **new**(i). Имя ссылочной переменной с последующим символом ^ называют *«переменной с указателем»*. Именно она синтаксически выполняет роль динамической переменной и может быть использована в любых конструкциях языка, где допустимо использование переменных того типа, что и тип динамической переменной.

Процедура **dispose**(i) уничтожает порожденные динамические объекты.

Графика

Графические возможности реализованы с помощью стандартного модуля **Graph.tpu.** Подключение модуля к программе выполняется директивой **uses graph.**

Процедура инициализации графического режима имеет три аргумента:

```
Initgraph(<драйвер>, <режим>, '<путь к драйверу>')
```

и может быть выполнена так:

```
uses graph;
   var gd, gm: integer;
   {переменные gd и gm определяют драйвер и режим}
begin
   gd:=vga; gm:=vgahi;
   initgraph(gd,gm,'d:\tp55');
   .........
```

Первые две команды можно заменить одной: **gd:=detect** с целью автоматического распознавания драйвера и установления режима максимального разрешения для данной машины.

Процедура **closegraph** освобождает память от драйвера и устанавливает режим работы экрана, который был до инициализации графики.

Для обнаружения ошибок в графике применяются функции **graphresult** и **grapherrormsg** (код ошибки). Инициализация графического режима с проверкой ошибок может быть выполнена в программе следующим образом:

```
uses graph; var gd, gm, errorcod: integer;
begin
   gd:=detect; initgraph(gd,gm,''); errorcod:=graphresult;
   if errorcod <> grok then
   begin
      writeln('ошибка графики');
      writeln(grapherrormsg(errorcod));
   halt
end;
```

Процедура **halt** останавливает выполнение программы и возвращает управление операционной системе.

Параметр «цвет» в процедурах работы с палитрой является выражением целого типа со значением из интервала 0..15, в частности, может быть константой из приведенного списка:

Black	=	0 (черный)	Darkgray	=	8 (темно-серый)
Blue	=	1 (синий)	Lightblue	=	9 (светло-синий)
Green	=	2 (зеленый)	Lightgreen	=	10 (светло-зеленый)
Cyan	=	3 (голубой)	Lightcyan	=	11 (светло-голубой)
Red	=	4 (красный)	Lightred	=	12 (светло-красный)
Magenta	=	5 (малиновый)	Lightmagenta	=	13 (светло-малиновый)
Brown	=	6 (коричневый)	Yellow	=	14 (желтый)
Lightgray	=	7 (светло-серый)	White	=	15 (белый).

Процедуры **setcolor**(<цвет>) и **setbkcolor**(<цвет>) устанавливают текущий цвет рисунка и цвет фона. При инициализации графики по умолчанию устанавливается черный фон и белый цвет рисунка.

Таблица 3.1

Основные процедуры модуля Graph

Заголовок процедуры	Геометрический смысл
`putpixel(x,y,c)`	Построить точку (*x*,*y*) цветом *c*
`setlinestyle(a,b,t)`	Установить стиль, образец и толщину линий
`line(x1,y1,x2,y2)`	Соединить две точки отрезком
`rectangle(x1,y1,x2,y2)`	Прямоугольник с заданными концами диагонали и сторонами, параллельными осям координат
`circle(x,y,r)`	Построить окружность с центром (*x*,*y*) и радиусом *R*
`arc(x,y,a,b,r)`	Построить дугу окружности: *a*, *b* — начальный и конечный углы в градусах
`ellipse(x,y,a,b,rx,ry)`	Построить эллиптическую дугу: *rx*, *ry*— полуоси эллипса
`setfillstyle(t,c)`	Установить стиль закраски и ее цвет
`fillellipse(x,y,rx,ry)`	Построить закрашенный эллипс, используя цвет рисунка
`floodfill(x,y,cg)`	Закрасить фигуру до границы с цветом *cg*; (*x*,*y*) — внутренняя точка фигуры
`bar(x1,y1,x2,y2)`	Построить столбец, используя тип и цвет закраски
`pieslice(x,y,a,b,r)`	Построить и закрасить сектор круга: *a*, *b* — начальный и конечный углы дуги в градусах
`sector(x,y,a,b,rx,ry)`	Построить и закрасить эллиптический сектор
`settextstyle(f,n,d)`	Установить шрифт, направление вывода, размер символов текста
`outtextxy(x,y,st)`	Вывести строку *st*, начиная с точки (*x*,*y*)
`outtext(st)`	Вывести строку, начиная с точки расположения текущего указателя
`setlinestyle(a,b,t)`	Стиль линии

Таблица 3.2

Значения первого аргумента процедуры setlinestyle(a,b,t)

Значение стиля	Смысл
`0 solidln`	Непрерывная линия
`1 dotteln`	Пунктирная линия
`2 centerln`	Штрихпунктирная линия
`3 dashedln`	Штриховая линия
`4 userbitln`	Определенная пользователем

Второй параметр *b*, «образец», имеет значение 4, если $a = 4$, в остальных случаях $b = 0$.

Третий параметр *t*, толщина линии, может иметь значение 1 (нормальная толщина) или 3 (жирная линия).

Первый аргумент процедуры `setfillstyle(t,c)` — тип закраски *t* — принимает значения из интервала 0..12. Наиболее употребителен тип $t = 1$ — заполнение фигуры текущим цветом.

Для вывода текста на графический экран сначала выполняется процедура `settextstyle(f,n,d)`, устанавливающая шрифт *f*, направление вывода *n* и размер символов (параметр *d*). При $f = 0$ используется стандартный точечный шрифт, встроенный в систему Турбо-Паскаль.

Контрольные вопросы

1. Из каких разделов состоит программа на Паскале?
2. В каких случаях используют циклы с параметром, с предусловием и с постусловием? Приведите примеры.
3. В чем различия и сходство данных типа *char* и *string*?
4. В чем различия и сходство данных типа *массив* и *множество*?
5. В чем различия и сходство данных типа *массив* и *запись*?
6. В чем принципиальные отличия *подпрограммы-процедуры* от *подпрограммы-функции*?
7. Для каких целей служат *модули*?
8. Какую роль в программе играет файловая переменная?
9. Для чего вводят *динамические структуры*? В чем их принципиальное отличие от *статических типов данных*?
10. Каким образом в Паскаль-программе инициализируется графический режим?

Темы для рефератов

1. Программирование баз данных на Паскале.
2. Программирование экспертных систем на Паскале.
3. Программирование игр на Паскале.
4. Программирование с использованием TurboVizion.

5. Объектно-ориентированное программирование на Паскале.
6. Отладка программ в среде Турбо.

Темы семинарских занятий

1. Основные конструкции языка Паскаль. Структура программы.
2. Циклы. Типовые задачи реализации циклических вычислительных процессов.
3. Простые типы данных. Символьный тип. Перечисляемые и интервальные типы.
4. Процедуры и функции.
5. Программы с рекурсивными процедурами и функциями.
6. Файлы.
7. Структуры данных. Массивы. Типовые задачи обработки массивов.
8. Множества.
9. Записи.
10. Модули.
11. Динамические информационные структуры.
12. Графические возможности Турбо-Паскаля.

Рекомендации по программному обеспечению

Для проведения всех занятий в пределах данного практикума достаточно стандартной системы программирования Турбо-Паскаль начиная с версии 5.5 и старше.

Задачи и упражнения

Основные конструкции языка

Упражнение № 1. Первое знакомство с системой программирования Турбо-Паскаль

1. Включить компьютер, запустить систему Турбо-Паскаль, познакомиться с текстовым редактором Edit.

Команды редактора Edit:

⇒	— перемещение курсора на символ вправо;
⇐	— перемещение курсора на символ влево;
⇑	— перемещение курсора на строку вверх;
⇓	— перемещение курсора на строку вниз;
Home	— перемещение курсора в начало текущей строки;
End	— перемещение курсора в конец текущей строки;
Page Up	— перемещение курсора на страницу вверх;
Page Down	— перемещение курсора на страницу вниз;

Ctrl + *Home*	— перемещение курсора в левый верхний угол;
Ctrl + *End*	— перемещение курсора в левый нижний угол;
Insert	— включение и выключение режима вставки;
Delete	— удаление символа, стоящего в позиции курсора;
Backspace	— удаление символа, стоящего перед курсором;
Ctrl + *N*	— вставка пустой строки над строкой, где находится курсор;
Ctrl + *Y*	— удаление строки, где находится курсор;

2. Набрать свою фамилию, имя и отчество. В следующей строке — свой домашний адрес, номер телефона (если он есть).

3. Набрать предложение: «Шла собака по роялю и сложила песню.» В этом предложении заменить все буквы «о» на «е», а после каждой буквы «а» вставить букву «с».

4. Очистить рабочее поле и набрать текст первой программы на Паскале.

Пример 1.

```
Program Example_1;
Var a,b,rez: Integer;
Begin Writeln('Введите два числа через пробел');
   Readln(a,b);
   rez:=a*b;
   Writeln('Их произведение равно', rez);
   Writeln('Нажмите <Enter>');
   Readln;
End.
```

Пояснения к программе

Имя этой программы `Example_1`. Из разделов описаний имеется лишь один – раздел переменных. Он начинается со служебного слова **`Var`**, после которого идет последовательность объявления переменных. После описательной части идет раздел операторов, начинающийся со служебного слова **`Begin`**, после которого идут операторы языка. Первый встречающийся оператор — это `Writeln('текст');` – записать (вывести) на экран текст, заключенный между апострофами, *ln* добавляется в конце этого оператора для того, чтобы курсор автоматически переходил на следующую строку при выводе на экран текстов или результатов выполнения программы. Следующий оператор — это `Readln(a,b);` — читать данные с клавиатуры. В данном случае необходимо ввести два целых числа через пробел, тогда переменной *a* присваивается значение, равное первому введенному числу, а переменной *b* присваивается значение, равное второму введенному числу. Например, вы ввели числа 12 и 45, тогда $a = 12$, а $b = 45$. В конце этого оператора также можно ставить `ln`.

После этих двух операторов стоит оператор присваивания: `rez:=a*b;` (`:=` – это знак присваивания в языке Паскаль). При выполнении этого оператора переменная `rez` получит значение, равное произведению числа *a* на число *b*.

Следующий оператор — это снова оператор `writeln('текст',rez)` — он выведет на экран текст, заключенный между апострофами, а за ним значение переменной `rez`. Затем следующий оператор `writeln` выведет на экран сообщение: «Нажмите <Enter>», а оператор `readln` будет ожидать этого нажатия в окне

выполнения. В конце раздела операторов стоит служебное слово **End,** после которого стоит точка.

5. Провести компиляцию программы командой из меню COMPILE. Добиться успешной компиляции, исправляя указанные системой ошибки.

6. Запустить программу на выполнение командой из меню RUN.

На экране появляется сообщение:

Введите два целых числа через пробел.

Наберите на клавиатуре два целых числа через пробел и нажмите <Enter>. После этого появляется сообщение:

Произведение равно...

Нажмите <Enter>.

Вместо точек будет написано значение переменной *rez*, то есть число, равное произведению первого введенного числа на второе.

А теперь попробуйте выполнить следующие задания:

а) измените программу для нахождения суммы двух чисел;

б) измените программу для нахождения суммы четырех чисел;

в) найдите значение выражения $(a + (d - 12)3)(c - 5k)$, где значения переменных a, d, c и k вводятся с клавиатуры.

7. Сохраните текст программы в виде файла с помощью команд **File/Save as.**

Например: *c*:\prim1_1.pas; здесь *c*:\ — это название диска, на котором будем сохранять файл.

Задания для самостоятельной работы

1. Найдите периметр:
   а) прямоугольника (ширину и длину вводить с клавиатуры);
   б) треугольника (длины всех сторон вводить с клавиатуры);
   в) произвольного четырехугольника.
2. Вычислите значение выражения:
   а) $y = 15x^2 + 8x - 9$;
   б) $a = (b + c)d - k$.
3. Вычислите рациональным способом, то есть за минимальное число операций:
   а) $y = x^5$ $(y = (x^2)^2x$, то есть за три операции);
   б) $y = x^6$ $(y = (x^3)^2 = (x^2x)^2$, то есть за три операции);
   в) $y = x^8$ $(y = ((x^2)^2)^2$, тоже за три операции).
4. Найдите значение выражения:
   а) $y = |x| + x^4$, при $x = -3$; $x = 3$;
   б) $a = |x| + 4x^3 - 7x^2$, при $x = 2$; $x = -2$;
   в) $z = |x - 2| + 3x^8$, при $x = -2$; $x = 1$;
   г) $a = 6b^2 + |b - 3|^3 - 15$, при $b = 9$; $b = -3$.
5. Напишите программу вычисления значения выражения:
   а) $y = (3x^3 + 18x^2)x + 12x^2 - 5$;
   б) $a = (d + c + b)e - 5k - 1$;
   в) $d = 3c^3 + |c^2 - 4c + 7|^3 - 5c$;

г) $c = |x + 4| - |x^2 - 3x + 6|$.

6. Поменяйте местами значения переменных x и y:

а) y с использованием промежуточной переменной (`t:=x; x:=y; y:=t;`);

б) без использования промежуточной переменной (`x:=x-y;y:=x+y;x:=y-x;`).

Упражнение № 2. Целый и логический типы данных. Условный оператор

Разберите несколько примеров на использование целых и логических типов данных, условного оператора.

Пример 2. *Вывести на экран большее из двух данных чисел.*

```
Program Example_2;
Var x,y: Integer;
Begin
   Writeln('введите 2 числа');
   Readln(x,y); {вводим два целых числа через пробел}
   If x>y Then Writeln(x) {если (If)x больше y, то (Then) выводим x}
   Else Writeln(y); {иначе (Else) выводим y}
   Readln;
End.
```

Пример 3. *Даны целые числа a, b, c. Если $a \le b \le c$, то все числа заменить их квадратами, если $a > b > c$, то каждое число заменить наибольшим из них, в противном случае сменить знак каждого числа.*

Решение.

Условие задачи перепишем следующим образом:

$a = a^2$, $b = b^2$, $c = c^2$, если $a <= b <= c$,

$a = c$, $b = c$, если $a > b > c$,

$a = -a$, $b = -b$, $c = -c$, в остальных случаях.

```
Program Example_3;
Var a,b,c: Integer;
Begin
   Writeln('Введите числа a, b, c');
   Readln(a,b,c);
   If (a<=b) And (b<=c) Then
   Begin a:=sqr(a); b:=sqr(b); c:=sqr(c) End
   Else If (a>b) And (b>c) Then Begin a:=c; b:=c End
   Else Begin a:=-a; b:=-b; c:=-c End;
   Writeln(a:3,b:3,c:3);
   Readln;
End.
```

Задания для самостоятельной работы

1. Какими будут значения переменных j, k после выполнения условного оператора: `If j>k Then j:=k-2 Else dec(k,2);` если исходные значения переменных равны а) $j = 3$, $k = 5$; б) $j = 3$, $k = 3$; в) $j = 3$, $k = 2$.

2. Запишите условный оператор, в котором значение переменной вычисляется по формуле: $a + b$, если a — нечетное и ab, если a — четное.

3. Вычислите значение функции:

$$\begin{cases} x^2 + 5 & \text{при} \quad x > 3, \\ x - 8 & \text{при} \quad x \le 3. \end{cases}$$

4. Найдите наибольшее из трех данных чисел.

5. Выведите на экран номер четверти, которой принадлежит точка с координатами (x,y), при условии, что x и y отличны от 0.

6. Вычислите значение функции:

$$\begin{cases} x - 12 & \text{при } x > 0, \\ 5 & \text{при } x = 0, \\ x^2 & \text{при } x < 0. \end{cases}$$

7. Даны три целых числа, найдите среднее из них. Средним назовем число, которое больше наименьшего из данных чисел, но меньше наибольшего.

8. Напишите фрагмент программы, подсчитывающий сумму только положительных из трех данных чисел.

9. Даны три числа. Напишите фрагмент программы, подсчитывающий количество чисел, равных нулю.

10. После выполнения операторов

```
a:=0;
If a<>0 Then; a:=2
```

значение переменной равно двум. Объясните почему.

11. Используя составной оператор, упростите следующий фрагмент программы:

```
If a>b Then c:=1;
If a>b Then d:=2;
If a<=b Then c:=3;
If a<=b Then d:=4.
```

12.Составьте программу нахождения произведения двух наибольших из трех введенных с клавиатуры чисел.

13. Если целое число M делится нацело на целое число N, то вывести на экран частное от деления, в противном случае — сообщение «M на N нацело не делится».

14. Найдите количество положительных (отрицательных) чисел среди четырех целых чисел A, B, C и D.

15. Чему равны значения переменных a и b после выполнения последовательности действий:

а) `a:= 15 Div (16 Mod 7); b:= 34 Mod a*5 - 29 Mod 5*2;`

б) `a:= 4*5 Div 3 Mod 2; b:= 4*5 Div (3 Mod 2);`

в) `a:= a*b; b:= b*b.`

16. Составьте программу, которая определяла бы вид треугольника (если данные отрезки позволяют его построить).

17. Составьте программу, которая уменьшает первое число в пять раз, если оно больше второго по абсолютной величине.

18. Составьте программу вычисления выражения:

а) $\max(x + y + z,\ xyz) + 3$;

б) $\min(x^2 + y^2,\ y^2 + z^2) - 4$,

если x, y, z введены с клавиатуры.

19. Составьте программу, которая из трех введенных с клавиатуры чисел возводит в квадрат положительные, а отрицательные оставляет без изменения.

Упражнение № 3. Целый тип данных. Цикл с параметром

Пример 4. *Составить программу вычисления значения выражения*

$y = ((...(20^2 - 19^2)^2 - 18^2)^2 - ... - 1^2)^2$.

Решение. В данном случае целесообразно организовать цикл с параметром, изменяющимся от 20 до 1, то есть шаг изменения параметра равен −1.

Обозначим: y — очередное значение квадрата числа; n — параметр цикла.

Учитывая это, составим программу:

```
Program Example_4;
Var y, n: Integer;
Begin
   y:=sqr(20);
   For n:=19 Downto 1 Do y:=sqr(y-sqr(n));
   Writeln('Значение выражения равно');
   Writeln(y);
End.
```

Пример 5. *Из чисел от 10 до 99 вывести те, сумма цифр которых равна n (0 < n ≤18).*

Решение. Обозначим: k — это просматриваемое число, $p1$ — это первая цифра числа k, $p2$ — это вторая цифра числа k, s — это сумма цифр данного числа k. Число k будем выписывать только в том случае, когда сумма $p1$ и $p2$ будет равна s.

```
Program Example_5;
Var k,n,p1,p2,s:Integer;
Begin
   Writeln('введите целое число');
   Readln(n); {вводим целое число}
   For k:=10 To 99 Do {для(For) k от 10 до(To) 99 делать(Do)}
   Begin
      p1:=k Div 10; {выделяем первую цифру}
      p2:=k Mod 10; {выделяем вторую цифру}
      s:=p1+p2; {находим сумму цифр}
      If s=n Then Writeln(k); {если сумма равна n, то выводим K}
   End;
   Readln;
End.
```

Задания для самостоятельной работы

1. Сколько раз будут выполнены операторы из тела циклов в следующих фрагментах программ:

а) `For k:=-1 To 1 Do ...`

б) `For k:=10 To 20 Do ...`

в) `For k:=20 To 10 Do ...`

г) `k:=5; r:=15;`

д) `For i:=k+1 To r-1 Do ...`

е) `k:=5;r:=15;`

ж) `For i:=0 To k*r Do ...`

з) `k:=r;`

и) `For i:=k To r Do ...`

2. Определите значение переменной s после выполнения следующих операторов:

```
s:=0; n:=10;
For i:=2 To n Do s:=s+100 Div i;
```

3. Составьте программу возведения натурального числа в квадрат, используя следующую закономерность:

$1^2 = 1$

$2^2 = 1 + 3$

$3^2 = 1 + 3 + 5$

$4^2 = 1 + 3 + 5 + 7$

.....

$n^2 = 1 + 3 + 5 + 7 + 9 + ... + (2n-1)$

4. Определите количество трехзначных натуральных чисел, сумма цифр которых равна заданному числу *N*.

5. Составьте программу вычисления суммы кубов чисел от 25 до 125.

6. Среди двузначных чисел найдите те, сумма квадратов цифр которых делится на 13.

7. Напишите программу поиска двузначных чисел, таких, что если к сумме цифр этого числа прибавить квадрат этой суммы, получится это число.

8. Квадрат трехзначного числа оканчивается тремя цифрами, которые как раз и составляют это число. Напишите программу поиска таких чисел.

9. Напишите программу поиска четырехзначного числа, которое при делении на 133 дает в остатке 125, а при делении на 134 дает в остатке 111.

10. Найдите сумму положительных нечетных чисел, меньших 100.

11. Найдите сумму целых положительных чисел из промежутка от *A* до *B*, кратных 4 (значения переменных *A* и *B* вводятся с клавиатуры).

12. Найдите сумму целых положительных чисел, больших 20, меньших 100, кратных 3 и заканчивающихся на 2, 4 или 8.

Упражнение № 4. Отладка. Пошаговая детализация

1. Загрузите файл *c:\Example_5.pas*.

2. Откройте окно *Watches* и введите переменные *p*1, *p*2, *k*, s.

3. Проследите работу программы в пошаговом режиме и составьте следующую таблицу для значений *k* от 10 до 15:

k	*p*1	*p*2	*s*

Примечание. Для удобства следует уменьшить размеры окна Watch и поместить его в удобное для Вас место экрана.

Пример 6. *Дано число n. Каким образом можно построить «перевертыш» данного числа?*

Решение. Обозначим: *n* — вводимое число, *m* — дубликат числа *n*, *a* — перевертыш числа *n*, *i* — переменная цикла для создания перевертыша.

```
Program Example_6;
Var n,m,a,i:Integer;
Begin
   Writeln('введите целое число, не большее 9999');
   Readln(n); {вводим целое число}
   m:=n; a:=0;
   {создание перевертыша}
   For i:=1 To 4 Do {так как число четырехзначное}
   Begin
      a:=a*10+m Mod 10; m:=m Div 10;
   End;
   If a=n Then Writeln('ДА!') Else Writeln('НЕТ!!!');
   {если перевертыш равен данному числу,
   то выводим «ДА», иначе — «НЕТ»}
   Readln;
End.
```

Трассировка примера

Рассмотрим выполнение этой программы в пошаговом режиме для числа 3994. Так как значение переменной *a* не равно значению переменной *n*, то на экране появится слово «НЕТ!!!».

i	*n*	*m*	*a* — это перевертыш
—	3994	3994	0
1	3994	399	$0 \cdot 10 + 3994 \bmod 10 = 0 + 4 = 4$
2	3994	39	$4 \cdot 10 + 399 \bmod 10 = 40 + 9 = 49$
3	3994	3	$49 \cdot 10 + 39 \bmod 10 = 490 + 9 = 499$
4	3994	0	$499 \cdot 10 + 3 \bmod 0 = 4990 + 3 = 4993$

Пример 7. *Даны натуральные числа n, k (n, k≤9999). Из чисел от n до k выбрать те, запись которых содержит ровно три одинаковые цифры.* Например, числа 0006, 0060, 6766, 5444, содержат ровно три одинаковые цифры.

Решение. Если данное число содержит ровно три одинаковых цифры, то только одна из цифр отличается от остальных, т.е. возможны четыре случая.

В первом случае отличается последняя цифра, во втором — третья, в третьем — вторая, а в четвертом — первая. Для каждого числа выполняется только одно из условий.

Фрагмент решения

	1	2	3	4	
1	x	x	x		— первое условие
2	x	x		x	— второе условие
3	x		x	x	— третье условие
4		x	x	x	— четвертое условие

```
Begin {Example_7};
   Writeln('Введите два числа, не больших 9999');
   Readln(n, k);
   For i:=n To k Do
   Begin
      m:=i;
      {выделение цифр: a1 – первая, a2 – вторая,a3 – третья, a4 –
четвертая}
      a4:=m Mod 10; m:=m Div 10;
      a3:=m Mod 10; m:=m Div 10;
      a2:=m Mod 10; a1:=m Div 10;
      {проверка условий}
      If ((a1=a2) And (a1=a3) And (a1<>a4)) Or
      {первое условие}
      ((a1=a2) And (a1=a4) And (a1<>a3)) Or {второе условие}
      ((a1=a3) And (a1=a4) And (a1<>a2)) Or {третье условие}
      ((a2=a3) And (a2=a4) And (a2<>a1)) {четвертое условие}
      Then Writeln(i:5);
   End;
   Readln;
End.
```

Трассировка примера

Рассмотрим выполнение программы для числа 3733.

n	m	$a1$	$a2$	$a3$	$a4$
3733	3733	—	—	—	3
3733	373	—	—	3	3
3733	37	3	7	3	3

Для данного числа выполняется третье условие, поэтому на экране появится число 3733.

Задания для самостоятельной работы

При решении задач следует использовать метод пошаговой отладки программы.

1. Составьте программу возведения данного натурального числа a в степень n. Исследовать для различных a максимальное значение n.

2. Даны натуральные числа a, b. Вычислите произведение $a\,b$, используя в программе лишь операции «+», «−», «=».

3. Пусть n — натуральное число и пусть $n!!$ означает $1 \cdot 3 \cdot 5 \cdot \ldots \cdot n$ для нечетного n и $2 \cdot 4 \cdot \ldots \cdot n$ для четного n. Для заданного натурального n вычислите $n!!$ и $(-1)^{n+1} \cdot n!!$

4. Даны натуральные числа n, a_1, a_2, ..., a_n:

а) определите число членов a_k последовательности a_1, a_2, ..., a_n, имеющих четные порядковые номера и являющихся нечетными числами;

б) получите сумму тех чисел данной последовательности, которые удовлетворяют условию $|a_i| < i^2$;

в) верно ли, что в последовательности больше отрицательных членов, чем положительных;

г) $\min(a_2,a_4,...) + \max(a_1,a_3,...)$.

5. Даны натуральные n, b_0, b_1, ..., b_n. Вычислите $f(b_0) + f(b_1) + \ldots + f(b_n)$, где

$$f(x) = \begin{cases} x^2, & \text{если } x \text{ кратно 3,} \\ x, & \text{если } x \text{ при делении на 3 дает остаток 1,} \\ \left[\frac{x}{3}\right], & \text{в остальных случаях.} \end{cases}$$

6. Дано натуральное число n. Получите все его натуральные делители.

7. Даны натуральные числа m, n. Получите все кратные им числа, меньшие $m \cdot n$.

8. Среди четырехзначных чисел выбрать те, у которых все четыре цифры различны.

9. Дано четырехзначное число n. Выбросите из записи числа n цифры 0 и 5, оставив прежним порядок остальных цифр. Например, из числа 1509 должно получиться 19.

10. Натуральное число из n цифр является числом Армстронга, если сумма его цифр, возведенных в n-ю степень, равна самому числу (например, $153 = 1^3 + 5^3 + 3^3$). Получите все числа Армстронга, состоящие из трех и четырех цифр.

11. Дана последовательность из 20 целых чисел. Определите количество чисел в наиболее длинной подпоследовательности из подряд идущих нулей.

Упражнение № 5. Циклы с условиями

Пример 8. *Цикл с предусловием. Дано натуральное число n. Подсчитайте количество цифр данного числа.*

Решение. Подсчет количества цифр начнем с последней цифры числа. Увеличим счетчик цифр на единицу. Число уменьшим в 10 раз (тем самым мы избавляемся от последней цифры числа). Далее с получившимся числом проделаем ту же последовательность действий и т.д., пока число не станет равным нулю.

```
Program Example_8;
Var m, n: Longint;
   k: Integer; {счетчик цифр}
Begin
   Writeln('Введите целое число'); {вводим целое число n}
   Readln(n);m:=n;
   k:=0;
   While m<>0 Do {пока число m<>0 делать(Do)}
   Begin
      Inc(k); {или k:=k+1;}
      m:=m Div 10; {«уменьшаем» число на последнюю цифру}
   End;
   Writeln('В числе',n,' - ',k,' цифр !'); {вывод количества цифр}
   Readln;
End.
```

Трассировка программы

Рассмотрим выполнение этой программы в пошаговом режиме для числа 65387.

В результате работы программы на экране появится предложение:

```
В числе 65387 - 5 цифр !
```

n	*m*	*k*
65387	65387	0
65387	6538	1
65387	653	2
65387	65	3
65387	6	4
65387	0	5

Пример 9. *Дана непустая последовательность натуральных чисел, за которой следует 0. Составить программу поиска в данной непустой последовательности порядкового номера наименьшего элемента.*

Решение. Обозначим через *x*, *i* — очередной член последовательности и его номер; min, *k* - минимальный член последовательности и его номер. Считывание членов последовательности производится до тех пор, пока не будет введен 0, то есть пока *x*<>0. Начальное значение минимума определяется значением первого члена последовательности. Очередное вводимое число требуется сравнивать с текущим значением минимума, и если текущее значение min окажется больше очередного члена последовательности, то его надо изменить.

```
Program Example_9;
Var x, i, min, k: Integer;
Begin
   Writeln('Введите первый член последовательности');
   Read(x); k:=1;
   min:=x; i:=2;
   While x<>0 Do
   Begin
      If x<min Then Begin min:=x; k:=i End;
      Writeln('Введите ',i,' элемент последовательности');
      Read(x);
      Inc(i);
   End;
   Writeln('Номер минимального элемента - ', k);
End.
```

Пример 10. *Цикл с постусловием. Составить программу планирования закупки товара в магазине на сумму, не превышающую заданной величины.*

Решение. Обозначим через *x*, *k* соответственно цену и количество товара, через *p* — заданную предельную сумму, через *s* — общую стоимость покупки. Начальное значение общей стоимости покупки (*s*) равно нулю. Значение предельной суммы считывается с клавиатуры. Необходимо повторять запрос цены и количества выбранного товара, вычислять его стоимость, суммировать ее с общей стоимостью и выводить результат на экран до тех пор, пока она не превысит предельную сумму *p*. В этом случае на экран надо вывести сообщение о превышении.

```
Program Example_10;
Var x, k, p, s: Integer;
Begin
   Writeln('Предельная сумма - '); Readln(p);
```

```
s:=0;
Repeat
Writeln('Введите цену товара и его количество');
Readln(x,k);
s:=s+x*k;
Writeln('Стоимость покупки равна ',s);
Until s>p;
Writeln('Суммарная стоимость покупки превысила предельную сумму');
End.
```

При описании циклов с постусловием необходимо принимать во внимание следующее:

- перед первым выполнением цикла условие его окончания (или продолжения) должно быть определено;
- тело цикла должно содержать хотя бы один оператор, влияющий на условие окончания (продолжения), иначе цикл будет бесконечным;
- условие окончания цикла должно быть в результате выполнено.

Пример 11. *Написать программу нахождения наибольшего общего делителя (НОД) двух неотрицательных чисел.*

Решение. Для решения данной задачи воспользуемся алгоритмом Евклида. Пусть x и y одновременно не равные нулю целые неотрицательные числа и пусть $x \geq y$, тогда если $y = 0$, то НОД$(x,y) = x$, а если $y \neq 0$, то для чисел x, y и r, где r – остаток от деления x на y выполняется равенство НОД(x,y) = НОД(y,r).

```
Program Example_11;
Var x, y: Integer;
Begin
   Writeln('Введите два числа');
   Readln(x,y); {вводим два целых числа }
   Repeat {выполнять}
   If x>y Then x:=x Mod y Else y:=y Mod x;
   Until (x=0) Or (y=0);
   {до тех пор, пока одно из чисел не станет равно нулю}
   Writeln('НОД=', x+y); {вывод НОД – без условного оператора,
   так как одно из чисел обязательно равно нулю}
   Readln;
End.
```

Пример 12. *Даны натуральные числа x и y, не равные нулю одновременно. Найти d = НОД(x,y) и такие целые q и w, что $d = qx + wy$.*

Решение. Добавим в алгоритм Евклида переменные p, q, r, s, m и n, такие, что $m = pa + qb$, $n = ra + sb$, где первоначально $m = a = x$, $n = b = y$.

Рассмотрим решение задачи для чисел 48 и 18.

M	N	P	Q	R	S		Результаты
48	18	1	0	0	1		$48 = 48 \cdot 1 + 18 \cdot 0$ $18 = 48 \cdot 0 + 18 \cdot 1$
48 mod 18 = 12	18	1	−2			$m > n$	$12 = 48 \cdot 1 + 18 (-2)$
12	18 mod 12 = 6			−1	3	$m < n$	$6 = 18 \cdot 1 + 12 (-1) =$ $= 48 (-1) + 18 \cdot 3$
12 mod 6 = 0	6	3	−8			$m > n$	$0 = 18 \cdot 1 + 6 (-2) =$ $= 48 \cdot 3 + 18 (-8)$
0	6					$m = 0$	$d = n \quad q = r \quad w = s$

Итак, d = НОД(48,18) = 6 и 6 = 48(−1) + 18·3.

Значения переменных p, q, r, s изменяются следующим образом:

• как только значение переменной m уменьшается на $k \cdot n$, значение p уменьшается на $k \cdot r$, а q уменьшается на $k \cdot s$;

• аналогично, как только значение n уменьшается на $k \cdot m$, значения переменных r и s уменьшаются соответственно на $k \cdot p$ и на $k \cdot q$.

Учитывая все, что сказано выше, составим программу:

```
Program Example_12;
Var x,y: Integer; {исходные данные}
   p,q,r,s,m,n: Integer; {введенные вспомогательные переменные}
   k: Integer; {для изменения значений p,q,r,s}
   d: Integer; {значение наибольшего общего делителя}
Begin
   Read(x,y);
   m:=x; n:=y; p:=1; q:=0; r:=0; s:=1;
   Repeat
      If m>n Then
   Begin
      k:=m Div n; m:=m Mod n;
      p:=p-k*r; q:=q-k*s
End
Else
Begin k:=n Div m; n:=n Mod m; r:=r-k*p; s:=s-k*q End;
   Until (m=0) Or (n=0);
   If m=0 Then Begin d:=n; q:=r; w:=s; End
   Else Begin d:=m; q:=p; w:=q; End;
   Writeln(d,'=',q,'*',x,'+',w,'*',y);
End.
```

X	Y		Результаты
48	18		
48 mod 18 = 12	18	$x>y$	НОД(48,18) = НОД(12,18)
12	18 mod 12 = 6	$x<y$	НОД(12,18) = НОД(12,6)
12 mod 6 = 0	6	$x>y$	НОД(12,6) = НОД(0,6)
0	6	$x=0$	НОД(0,6) = 6

Пример 13. *Вложенные циклы. Даны натуральные числа n и k. Составить программу вычисления выражения* $1^k + 2^k + \ldots + n^k$.

Решение. Для вычисления указанной суммы целесообразно организовать цикл с параметром i, в котором, во-первых, вычислялось бы очередное значение $y = i^k$ и, во-вторых, осуществлялось бы накопление суммы прибавлением полученного слагаемого к сумме всех предшествующих ($s = s + y$).

```
Program Example_13;
Var n, k, y, i, s, m: Integer;
Begin
   Writeln('Введите исходные данные n и k');
   Readln(n,k);
```

```
    s:=0;
    For i:=1 To n Do
    Begin
        y:=1;
        For m:=1 To k Do y:=y*i; {нахождение степени k числа i}
        s:=s+y;
    End;
    Writeln('Ответ: ',s);
End.
```

Таким образом, для решения задачи потребовалось организовать два цикла, один из которых пришлось поместить внутрь другого. Такие конструкции называют вложенными циклами.

Пример 14. *Модифицировать предыдущую программу так, чтобы она вычисляла сумму* $1^1 + 2^2 + ... + n^n$.

Решение. Данная задача отличается от предыдущей тем, что показатель степени очередного слагаемого совпадает со значением ее основания, следовательно, параметры внутреннего цикла (цикла, в котором вычисляется очередное слагаемое) совпадают с параметрами внешнего цикла.

```
Program Example_14;
Var n, y, i, s, m: Integer;
Begin
    Writeln('Введите начальное значение n');
    Readln(n);
    s:=0;
    For i:=1 To n Do
    Begin y:=1;
        For m:=1 To i Do y:=y*i; {нахождение степени k числа i}
        s:=s+y;
    End;
    Writeln('Ответ: ',s);
End.
```

Внутренний и внешний циклы могут быть любыми из трех рассмотренных ранее видов: циклами с параметром, циклами с предусловием или циклами с постусловием. Правила организации как внешнего, так и внутреннего циклов такие же, как и для простого цикла каждого из этих видов. Но при использовании вложенных циклов необходимо соблюдать следующее условие: внутренний цикл должен полностью укладываться в циклическую часть внешнего цикла.

Пример 15. *Старинная задача. Сколько можно купить быков, коров и телят, если плата за быка 10 руб., за корову — 5 руб., за теленка — полтинник (0,5 руб.), если на 100 руб. надо купить 100 голов скота.*

Решение. Обозначим b — число быков; k — число коров; t — число телят. После этого можно записать два уравнения: $10b + 5k + 0{,}5t = 100$ и $b + k + t = 100$. Преобразуем их: $20b + 10k + t = 200$ и $b + k + t = 100$.

На 100 рублей можно купить:
- не более 10 быков, т.е. $0 \le b \le 10$;
- не более 20 коров, т.е. $0 \le k \le 20$;
- не более 200 телят, т.е. $0 \le t \le 200$.

```
Program Example_15;
Var b, k, t: Integer;
Begin
   For b:=0 To 10 Do
      For k:=0 To 20 Do
         For t:=0 To 200 Do
            If (20*b+10*k+t=200) And (b+k+t=100) Then
         Writeln('быков ',b,'коров',k,'телят',t);
End.
```

Пример 16. *Сколько раз будет проверяться условие в данной программе?*

Решение. Значение переменной b изменяется 11 раз (от 0 до 10), для каждого ее значения переменная k изменяется 21 раз, а для каждого значения переменной k переменная t изменяется 201 раз. Таким образом, условие будет проверяться $11 \cdot 21 \cdot 201$ раз. Но если известно число быков и коров, то число телят можно вычислить по формуле $t = 100 - (b + k)$ и цикл по переменной t исключается.

```
Program Example_16;
Var b, k, t: Integer;
Begin
   For b:=0 To 10 Do
   For k:=0 To 20 Do
   Begin
      t:=100-(b+k);
      If (20*b+10*k+t=200) Then
      Writeln('быков ',b,' коров ',k,' телят ',t);
   End;
End.
```

При этом решении условие проверяется $11 \cdot 21$ раз.

Задания для самостоятельной работы

1. Дана последовательность операторов:

```
a:=1; b:=1;
while a+b<8 do Begin a:=a+1; b:=b+2 End;
s:=a+b;
```

Сколько раз будет повторен цикл и какими будут значения переменных a, b, и s после завершения этой последовательности операторов?

2. Какими будут значения переменных a и b после выполнения операторов:

```
a:=1; b:=1;
While a<=3 Do a:=a+1; b:=b+1;
```

3. Определите значение переменной s после выполнения следующих операторов:

а) `s:=0; i:=0; While i<5 Do Inc(i); s:=s+100 Div i;`

б) `s:=0; i:=1; While i>1 Do Begin s:=s+100 Div i; dec(i) End;`

4. Дана последовательность операторов, вычисляющих факториал f числа n:

```
k:=1; f:=0;
While k<n Do f=f*k
k:=k+1,
```

которая содержит пять ошибок. Найдите эти ошибки.

5. Найдите и исправьте ошибки в следующем фрагменте программы, определяющей для заданного натурального числа n число, записанное цифрами числа n в обратном порядке:

```
p:=n;
While p>=0 Do
Begin
   a:=a+p Mod 10;
   p:=p Div 10
End;
```

6. Найдите сумму цифр числа.

7. Найдите первую цифру числа.

8. Припишите по 1 в начало и в конец записи числа n. Например, было $n = 3456$, стало $n = 134561$.

9. Поменяйте местами первую и последнюю цифры числа.

10. Поменяйте порядок цифр числа на обратный. Например, было 12345, стало 54321.

11. Найдите количество четных цифр целого положительного числа.

12. Найдите самую большую цифру целого числа.

13. Найдите сумму цифр целого числа, больших 5.

14. Сколько раз данная цифра встречается в целом числе?

15. Составьте программу, проверяющую, является ли последовательность из 10 целых чисел, вводимых с клавиатуры, возрастающей.

16. Составьте программу, проверяющую, является ли заданное натуральное число палиндромом, то есть таким, десятичная запись которого читается одинаково слева направо и справа налево.

17. Определите значение переменной s после выполнения следующих операторов:

```
s:=0; i:=1;
Repeat s:=s+5 Div i; i:=1-1; Until i<=1;
```

18. Произведение N первых нечетных чисел равно p. Сколько сомножителей взято?

19. Числа Фибоначчи (f_n) определяются формулами: $f_0 = f_1 = 1$; $f_n = f_{n-1} + f_{n-2}$ при $n = 2, 3, \ldots$ Составьте программу:

а) определения f — 40-е число Фибоначчи;

б) поиска f — первого числа Фибоначчи, большего m ($m > 1$);

в) вычисления s — суммы всех чисел Фибоначчи, которые не превосходят 1000.

20. Составьте программу, проверяющую, является ли заданное натуральное число совершенным, то есть равным сумме своих положительных делителей, кроме самого этого числа.

21. Покажите, что любой оператор цикла с предусловием можно записать с помощью условного оператора и оператора цикла с постусловием.

22. Покажите, что любой оператор цикла с постусловием можно записать с помощью условного оператора и оператора цикла с предусловием.

23. Дана непустая последовательность натуральных чисел, за которой следует 0. Вычислите сумму положительных элементов последовательности, порядковые номера которых нечетны.

24. Найдите НОД трех чисел. *Примечание.* НОД (a,b,c) = НОД(НОД(a,b),c).

25. Проверьте, являются ли два данных числа взаимно простыми. Два числа называются взаимно простыми, если их наибольший общий делитель равен 1.

26. Найдите наименьшее общее кратное (НОК) чисел n и m, если

$$\text{НОК}(n, m) = \frac{nm}{\text{НОД}(n, m)}.$$

27. Даны натуральные взаимно простые числа n, p. Найдите m такое, что, во-первых, $m < p$, во-вторых, произведение чисел $m \cdot n$ при делении на p дает остаток 1. *Примечание.* Воспользуйтесь алгоритмом, описанным в примере 2.

28. От прямоугольника 324×141 отрезают квадраты со сторонами 141, пока это возможно. Затем вновь отрезают квадраты со стороной, равной $324 - 2 \times 141 = 42$ и т.д. На какие квадраты и на сколько квадратов будет разрезан прямоугольник?

29. Напишите вариант алгоритма Евклида, основанный на соотношениях НОД($2a$, $2b$) = 2НОД(a, b);

НОД($2a$, b) = НОД(a, b), при нечетном b, не включающий деления с остатком, использующий лишь деление на 2 и проверку четности.

30. Даны натуральные числа m и n. Найдите такие натуральные p и q, не имеющие общих делителей, что $\frac{p}{q} = \frac{m}{n}$.

31. Что будет выведено на экране монитора после выполнения следующего фрагмента программы:

```
a:=1; b:=1;
For i:=0 To n Do
Begin
   For j:=1 To b Do Write('*');
   Writeln;
   c:=a+b; a:=b; b:=c;
End;
```

Если $n = 6$, решение какой задачи выражает этот фрагмент программы?

32. Что будет выведено на экране монитора после выполнения следующего фрагмента программы:

```
b:=0;
While a<>0 Do
Begin
   b:=b*10+a Mod 10;
   a:=a Div 10;
End;
Write(b);
```

Если $a = 13305$, решение какой задачи выражает этот фрагмент программы?

33. Исходное данное — натуральное число q, выражающее площадь. Напишите программу для нахождения всех таких прямоугольников, площадь которых равна q и стороны выражены натуральными числами.

34. Составьте программу получения всех совершенных чисел, меньших заданного числа n. Число называется совершенным, если равно сумме всех своих положительных делителей, кроме самого этого числа. Например, 28 — совершенно, так как $28 = 1 + 2 + 4 + 7 + 14$.

35. Из истории. Грекам были известны первые четыре совершенных числа: 6, 28, 496, 8128. Эти числа высоко ценились. Даже в XII веке церковь утверждала, что для спасения души необходимо найти пятое совершенное число. Это число было найдено

только в XV веке. До сих пор совершенные числа полностью не исследованы — не известно, имеется ли конечное число совершенных чисел или их число бесконечно, кроме того, не известно ни одного нечетного совершенного числа, но и не доказано, что таких чисел нет.

36. Дано натуральное число n. Можно его представить в виде суммы трех квадратов натуральных чисел? Если можно, то:

- укажите тройку x, y, z таких натуральных чисел, что $x^2 + y^2 + z^2 = n$;
- укажите все тройки x, y, z таких натуральных чисел, что $x^2 + y^2 + z^2 = n$.

37. Найдите натуральное число от 1 до 10000 с максимальной суммой делителей.

38. Даны натуральные числа a, b ($a < b$). Получите все простые числа p, удовлетворяющие неравенствам $a \le p \le b$.

39. Даны натуральные числа n, m. Получите все меньшие n натуральные числа, квадрат суммы цифр которых равен m.

40. Даны натуральные числа n и m. Найдите все пары дружественных чисел, лежащих в диапазоне от n до m. Два числа называются дружественными, если каждое из них равно сумме всех делителей другого (само число в качестве делителя не рассматривается).

41. В данном натуральном числе переставьте цифры таким образом, чтобы образовалось наименьшее число, записанное этими же цифрами.

42. Составьте программу, печатающую для данного натурального числа k-ю цифру последовательности

12345678910..., в которой выписаны подряд все натуральные числа;

14916253649..., в которой выписаны подряд квадраты всех натуральных чисел;

1123581321..., в которой выписаны подряд все числа Фибоначчи.

43. Составьте программу возведения заданного числа в третью степень, используя следующую закономерность:

$1^3 = 1$
$2^3 = 3 + 5$
$3^3 = 7 + 9 + 11$
$4^3 = 13 + 15 + 17 + 19$
$5^3 = 21 + 23 + 25 + 27 + 29$

44. Составьте программу для нахождения всех натуральных решений уравнения $n^2 + m^2 = k^2$ в интервале (1,10).

Примечание. Решения, которые получаются перестановкой n и m, считать совпадающими.

45. Найдите все трехзначные числа, удовлетворяющие каждому из условий:

- любые две цифры различны;
- число равно среднему арифметическому всех трехзначных чисел (включая данное), имеющих тот же цифровой состав.

46. Стороны прямоугольника заданы натуральными числами M и N. Составьте программу, которая будет находить, на сколько квадратов, стороны которых выражены натуральными числами, можно разрезать данный прямоугольник, если от него каждый раз отрезается квадрат максимально большой площади.

47. Дано натуральное число $n \le 2$. Составьте программу разложения этого числа на простые множители: простой множитель p должен быть выведен k раз, где k — натуральное число, такое, что n делится на p^k и не делится на p^{k+1}; каждый простой множитель должен быть выведен ровно один раз.

48. Даны натуральные числа N и p. Получите все натуральные числа, меньшие N и взаимно простые с p.

49. Даны целые числа p и q. Получите все делители числа q, взаимно простые с p.

50. Сумма квадратов длин катетов a и b прямоугольного треугольника равна квадрату длины гипотенузы c: $a^2 + b^2 = c^2$. Тройка натуральных чисел, удовлетворяющих этому равенству, называется пифагоровыми числами. Составьте программу нахождения основных троек пифагоровых чисел, используя следующие формулы:

$$a = u*v;\ b = \frac{u^2 - v^2}{2};\ c = \frac{u^2 + v^2}{2},$$

где u и v — взаимно простые нечетные натуральные числа и $u > v$.

51. Найдите наименьшее натуральное число N, представимое двумя различными способами в виде суммы кубов двух натуральных чисел x^3 и y^3 ($x \geq y$).

52. Даны натуральные числа m, n_1, n_2, ..., n_m ($m \geq 2$). Вычислите НОД(n_1, n_2, ..., n_m), воспользовавшись соотношением НОД(n_1, n_2, ..., n_m) = НОД(НОД(n_1, n_2, ..., n_{m-1}), n_m) и алгоритмом Евклида.

53. Найдите все простые несократимые дроби, заключенные между 0 и 1, знаменатели которых не превышают 7 (дробь задается двумя натуральными числами — числителем и знаменателем).

Простые типы данных. Символьный тип. Перечисляемые и интервальные типы

Упражнение № 6. Символьный тип

Пример 17. *Напишите программу вывода последовательности символов: A AB ABC ... AB..YZ на экран.*

Решение. Последовательность символов строится по следующему правилу: последовательно выводятся начальные отрезки латинского алфавита, состоящие из 1 символа, потом из 2 символов и так далее, до тех пор, пока не будет выведен весь алфавит. Число таких отрезков равно количеству букв в алфавите. Так как символьный тип данных является порядковым типом, то можем использовать цикл с параметром.

```
Program Example_17;
Var i,j:Char;
Begin
   For i:='a' To 'z' Do {число начальных отрезков алфавита}
   For j:='a' To i Do {число символов в данном начальном отрезке}
   Write(j);
   Readln;
End.
```

Результат работы программы:

```
a a b a b c a b c d . . . x y z
```

Модифицируйте программу для вывода последовательности символов в виде пирамиды:

```
      a
      a b
      a b c
      ...
      a b c ... x y z
```

Пример 18. *Напишите программу, которая подсчитывает число цифр, входящих в исходный текст. Текст — это последовательность символов, ввод которой заканчивается нажатием клавиши <Enter>.*

Решение. Так как окончанием ввода последовательности служит нажатие клавиши <Enter> (ее обозначение — #10), будем вводить символы до тех пор, пока значение очередного символа не совпадет со значением #10. Анализируя каждый символ, будем увеличивать счетчик, если символ является цифрой.

```
Program Example_18;
Var ch:Char;
   k:Integer;
Begin
   Read(ch);
   k:=0;
   While ch<>#10 Do {пока не нажата клавиша <Enter>}
   Begin
      If (ch>='0') And (ch<='9') Then Inc(k);
      Read(ch);
   End;
   Writeln(^G,'Число цифр равно ',k);
End.
```

Модифицируйте программу так, чтобы она решала следующие задачи:

- определить, является ли введенная строка правильной записью целого числа;
- вычислить сумму цифр введеного числа.

Задания для самостоятельной работы

1. Напишите программу вывода последовательности символов на экран:
а) ZYYXXX...AA...AA;
б) ABC...ZZBC...ZZZC...ZZ...ZZ.
2. Составьте программу, которая печатает *true*, если в заданном тексте буква *A* встречается чаще, чем *B*, и печатает *false* в противном случае.
3. Проверьте, правильно ли в заданном тексте расставлены круглые скобки (т.е. находится ли справа от каждой открывающей скобки соответствующая ей закрывающая скобка, а слева от каждой закрывающей — соответствующая ей открывающая).
4. Дана последовательность литер, имеющая следующий вид: $d_1 \pm d_2 \pm ... \pm d_n$ (d_i — цифры, $n > 1$). Вычислите значение этой алгебраической суммы.
5. Составьте программу, запрашивающую координаты ферзя на шахматном поле и показывающую поля доски, находящиеся «под боем».
6. Используя символьный тип данных, введите заданное вещественное число, записанное по правилам языка Паскаль, и присвойте его вещественной переменной *x*.

Упражнение № 7. Вещественный тип

Пример 19. *Напечатайте таблицу значений функции y = sin(x) на отрезке [0, 1] с шагом 0,1 (считайте, что при печати на каждое вещественное число отводится по 4 позиции строки).*

Решение. Постановка задачи наталкивает нас на использование цикла с вещественным параметром, но цикл с параметром предполагает использование переменной порядкового типа, а тип `Real` таковым не является.

```
Program Example_19_1; {вариант 1}
Var i: Real;
Begin
   i:=0;
   While i<=1 Do
   Begin
      Writeln(i:2:1,'',sin(i):4:3);
      i:=i+0.1;
   End;
   Readln;
End.
```

```
Program Example_19_2; {вариант 2}
Var i: Integer;
Begin i:=0;
   While i<=10 Do
   Begin
      Writeln(i,'',sin(i/10):4:3);
      Inc(i);
   End;
   Readln;
End.
```

На первый взгляд работа этих программ должна быть одинаковой, но, запустив программы, мы обнаружим, что первая программа выдает значения функции sin(x) для всех значений x от 0 до 0,9, а вторая программа — для всех значений x от 0 до 1. Почему это происходит?

Пример 20. *Дано x, принадлежащее интервалу от -1 до 1. Составьте программу вычисления суммы ряда* $x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots$ *с заданной точностью E. Нужная точность считается полученной, если очередное слагаемое оказалось по модулю меньше, чем данное малое положительное число E (это и все последующие слагаемые учитывать не надо).*

Решение. На первый взгляд программа должна вычислять очередное слагаемое $\frac{x^n}{n}$ и прибавлять его к сумме, полученной на предыдущем этапе. При нахождении значения x^n можно воспользоваться циклом с параметром, но, с другой стороны, чтобы вычислить значение x^n, достаточно значение x^{n-1} (найдено на предыдущем шаге) умножить на x.

```
Program Example_20;
Var x,st,sl,y,e:Real;
   n,z:Integer;
Begin
   Write('Введите x, принадлежащее (-1,1)'); Readln(x);
   Write('Введите погрешность вычисления'); Readln(e);
   y:=0; n:=1; z:=1; st:=x; sl:=x;
   Repeat
      Inc(y,z*sl); Inc(n); z:=-z; st:=st*x; sl:=st/n;
   Until sl<e;
   Writeln(y);
   Readln;
End.
```

Задания для самостоятельной работы

1. Даны действительные положительные числа a, b, c, x, y. Выяснить, пройдет ли кирпич с ребрами a, b, c в прямоугольное отверстие со сторонами x, y. Просовывать кирпич разрешается только так, чтобы каждое из его ребер было перпендикулярно или параллельно каждой из сторон отверстия.

2. Выписать фрагмент программы для решения указанной ниже задачи и обосновать, почему был выбран тот или иной вариант оператора цикла:

а) вычислить c — наибольший общий делитель натуральных чисел a и b;

б) найти u — первый отрицательный член последовательности $\cos(\operatorname{ctg} n)$, где $n = 1, 2, 3, \ldots$;

в) вычислить $p = (1 - \frac{1}{2^2})(1 - \frac{1}{3^2})\ldots(1 - \frac{1}{n^2}), \quad n > 2$;

г) вычислить $y = \cos(1 + \cos(2 + \ldots + \cos(39 + \cos 40)\ldots))$.

3. Квадратное уравнение $Ax^2 + Bx + C = 0$ задается его коэффициентами. Составить диалоговую программу нахождения корней квадратного уравнения.

4. Вычислить значение выражений:

а) $\sin x + \sin\sin x + \ldots + \underbrace{\sin\sin\ldots\sin x}_{n \text{ раз}}$;

б) $\sin x + \sin x^2 + \ldots + \sin x^n$;

в) $\sin x + \sin^2 x + \ldots + \sin^n x$;

г) $\frac{\cos 1}{\sin 1}\frac{\cos 1 + \cos 2}{\sin 1 + \sin 2}\cdot\ldots\cdot\frac{\cos 1 + \ldots + \cos n}{\sin 1 + \cdots + \sin n}$.

5. Представить обыкновенную дробь $\frac{m}{n}$ $(m < n)$ в виде цепной дроби $\frac{m}{n} = \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots \cfrac{1}{a_k}}}$. Например, $\frac{5}{7} = \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{2}}}$.

6. Вычислить $\cfrac{1}{1 + \cfrac{1}{3 + \cfrac{1}{5 + \cfrac{1}{\ldots + \cfrac{1}{101 + \cfrac{1}{103}}}}}}$.

7. Рассмотрим бесконечную последовательность $y_1, y_2, y_3, \ldots$, образованную по следующему закону:

$$y_1 = \frac{x + m - 1}{2};$$

$$y_i = \frac{1}{m}\left((m-1)y_{i-1} + \frac{x}{y_{i-1}^{m-1}}\right) \quad (i = 2, 3, \ldots),$$

где x — данное действительное число, m — натуральное число. Эта последовательность позволяет получить сколько угодно точные приближения числа $\sqrt[m]{x}(x \geq 0)$. Составить программу для вычисления значения $\sqrt[m]{x}$ с заданной точностью E ($E = |x - y_i^m|$).

8. Вычислить $1 - \frac{1}{2} + \frac{1}{3} - \ldots + \frac{1}{9999} - \frac{1}{10000}$ следующими четырьмя способами:

а) последовательно слева направо;

б) последовательно слева направо вычисляются $1 + \frac{1}{3} + \ldots + \frac{1}{9999}$ и $\frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{10000}$, затем второе значение вычитается из первого;

в) последовательно справа налево;

г) последовательно справа налево вычисляются суммы, выписанные в б), затем — вычитание.

Почему при вычислениях на ЭВМ каждым из этих способов получаются разные результаты?

9. Вычислить приближенное значение бесконечной суммы:

а) $\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} + \ldots$;

б) $\frac{1}{1 \cdot 3} + \frac{1}{2 \cdot 4} + \frac{1}{3 \cdot 5} + \ldots$;

в) $\frac{1}{1 \cdot 2 \cdot 3} + \frac{1}{2 \cdot 3 \cdot 4} + \frac{1}{3 \cdot 4 \cdot 5} + \ldots$.

10. Дано действительное число x, вычислить:

а) $\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} - \ldots (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots$;

б) $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + (-1)^n \frac{x^{2n}}{(2n)!} + \ldots$.

Нужное приближение считается полученным, если очередное слагаемое оказалось по модулю меньше данного положительного числа E.

11. Составить программу перевода действительного числа a ($0 < a < 1$) в двоичную систему счисления.

12. Дано 100 вещественных чисел. Вычислить разность между максимальным и минимальным из них.

13. Дано 200 вещественных чисел. Определить, сколько из них больше своих «соседей», то есть предыдущего и последующего чисел.

Упражнение № 8. Перечисляемые и интервальные типы

Пример 21. *Напишите программу, которая переменной t присваивает значение true, если первая дата предшествует (в рамках года) второй дате, и значение false в противном случае.*

Решение. Так как в условии задачи оговаривается, что обе даты должны находиться в рамках года, то дата должна задаваться днем и месяцем. Число дней любого месяца года не может быть более 31, число месяцев в году равно 12. Значение переменной *t* равно *true*, если номер первого месяца меньше второго, значение первого дня меньше второго при условии, что номера месяцев совпали.

```
Program Example_21;
Var d1,d2:1..31;
   m1,m2:1..12;
   t:Boolean;
Begin
   Write('Введите первую дату (день, месяц)');
   Readln(d1,m1);
   Write('Введите вторую дату (день, месяц)');
   Readln(d2,m2);
   t:=(m1<m2) Or ((m1=m2)And(d1<d2));
   Writeln(t);
End.
```

Модифицируйте программу так, чтобы осуществлялась проверка корректности введенных дат.

Пример 22. *Составьте программу, которая определяет, является ли введенный символ гласной или согласной буквой английского алфавита.*

Решение. По условию задачи все символы делятся на следующие группы:

- гласные буквы английского алфавита;
- согласные буквы английского алфавита;
- символы, не являющиеся буквами английского алфавита.

```
Program Example_22;
Var ch: Char;
Begin
   Write('Введите символ');
   Readln(ch);
   Case Upcase(ch) Of'
   'A','E','I','O','U': Writeln('Это гласная буква английского
            алфавита');
   'A'..'Z': Writeln('Это буква английского алфавита');
   Else Writeln('Этот символ не является буквой английского
          алфавита');
End.
```

Обратите внимание на то, что константы здесь в первом случае перечисляются через запятую, а во втором используется интервал значений.

Пример 23. *«Вечный календарь». Установлено: если исследуемая дата лежит в диапазоне от 1582 до 4902 г., в этом случае номер дня недели (воскресенье имеет номер 0, понедельник — 1, ..., суббота — 6) равен остатку от деления на 7 значения выражения* $[2{,}6m - 0{,}2] + d + y + [y/4] + [c/4] - 2c$, *где d — номер дня в месяце (1, 2, ...); m — номер месяца в году, нумерация начинается с марта (март имеет номер 1, апрель — номер 2, ..., декабрь — номер 10, январь и февраль считаются месяцами с номером 11 и 12 предыдущего года); y — две младшие цифры года; c — две старшие цифры года;* $[x]$ *означает целую часть числа x.*

Вычислить число пятниц, приходящихся на 13 число XX столетия.

```
Program Example_23;
Type month=(march, april, may, june, july,
            august, september, october, november,
            december, january, february);
   day=1..31;
   year=1582..4902;
   week=(sunday,monday,tuesday,wednesday,thursday,friday,saturday);
Const h=20;
   d: day=13;
   d_w: week= friday;
Var k:Integer; {для подсчета количества пятниц}
    y: year; Mod_y:0..99; int_y:15..49;
    m: month;
    n:-50..1000;
Begin
   k:=0;
   For y:=(h-1)*100 To h*100-1 Do {просмотрим все годы столетия}
   For m:= marth To february Do {просмотрим все месяцы года}
   Begin
      Mod_y:=y Mod 100; {найдем две последние цифры года}
      int_y:=y Div 100; {найдем две первые цифры года}
      n:=trunc(2.6*(Ord(m)+1)-0.2)+d+Mod_y+trunc(Mod_y/4)+
      trunc(int_y/4)-2*int_y;
      If n Mod 7=Ord(d_w) Then Inc(k);
   End;
   Writeln('число пятниц, приходящихся на ',d,' число в ',h,' столетии равно',k);
End.
```

Пример 24. *Найти k-е простое число в арифметической прогрессии 11, 21, 31, 41, 51, 61, ...*

Решение. Для решения поставленной задачи необходимо просматривать числа последовательности и проверять каждое из них на свойство простоты. Поскольку нам не известно, сколько членов последовательности необходимо просмотреть, мы должны просматривать этот ряд до тех пор, пока не найдем *k*-е простое число; для этого воспользуемся циклом с условием.

```
Program Example_24;
Var k: Integer;
   n, p,d: Longint;
Begin
   Writeln('Введите номер числа');
   Readln(k);
   n:=0; p:=1;
   While n<k Do
   Begin
      Inc(p,10); d:=2;
      While (p Mod d<>0) And (d<sqrt(p)) Do Inc(d);
      If d>=sqrt(p) Then Inc(n);
   End;
   Writeln(p);
```

```
    Readln;
End.
```

В этом решении мы смогли записать условие `d<sqrt(p)`, так как типы `Integer` и `Real` совместимы.

Задания для самостоятельной работы

1. Имеются описания

```
Var x,y: (winter, spring, summer, autumn);
   t: (cold, warm);
```

а) Допустимы ли присваивания: `x:=spring;t:=warm;t:=hot;y:=x;y:=t`?

б) Вычислить значения выражений:

```
spring<summer;
autumn<winter;
Succ(spring);
Pred(autumn);
Ord(spring);
winter<=summer;
spring<>warm;
Pred(spring);
Pred(cold);
Pred(autumn)+Ord(cold);
```

в) Допустим ли оператор цикла с заголовком `For x:=spring To autumn Do`?

2. Напишите программу, которая по заданной дате определяет время года. Программа должна проверять корректность введенной даты.

3. Даны описания следующих переменных:

```
Var m, m1: (january, february, march, april, may, june, july,
august, september, october, november, december);
k: 1..maxint; n: 1..12;
```

Присвоить переменной *m*1:

а) название месяца, следующего за месяцем *m*;

б) название *k*-го месяца после месяца *m*.

4. В старояпонском календаре был принят 60-летний цикл, состоящий из пяти 12-летних подциклов. Подциклы обозначались названием цвета: зеленый, красный, желтый, белый и черный. Внутри каждого подцикла годы носили название животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. Напишите программу, которая по номеру года определяет его название по старояпонскому календарю (1984 — год зеленой крысы).

5. Дано неотрицательное число *k*, не превышающее десяти тысяч. Напечатать фразу «*k* ворон» русскими словами. (Пример: если $k = 23$, то должно быть напечатано «двадцать три вороны»; если $k = 3651$, то «три тысячи шестьсот пятьдесят одна ворона»).

6. Имеются описания `Var d:'0'..'9'; k:0..9; n:Integer;` Ответьте на следующие вопросы:

а) Какие значения может принимать переменная *d*? Каков ее базовый тип? Допустимы ли присваивания: `d:='7'; d:='a'; d:=7`?

б) Какие значения может принимать переменная *k*? Каков ее базовый тип? Допустимы ли присваивания: `k:=5; k:=10; k:=-0; k:='5'`?

в) Верно ли, что к значениям ограниченного типа можно применять те же операции, что и к значениям базового типа?

г) Есть ли ошибки в операторе:

```
If k+n>7*k Then k:=abs(n) Mod 10 Else d:=chr(k+Ord('0')?
```

7. Укажите ошибки в следующем разделе типов:

```
Const n=180; pi=3.1415;
Type sign=('a','b','c','d');
   gl=(a, 'e, i, o, u);
   sgl=(b..d, f, g);
   log=Boolean;
   sign='0'..'9';
```

8. Найдите ошибки в следующей программе:

```
Program mistake_1;
Type month=(january, february, march, april, may, june, july,
   august, september, october, november, december);
   autumn=september..november;
Var m:autumn; d:'0'..'9'; k:0..9;
Begin
   Readln(m,d,k)
   If m>september Then d:=k Else k:=Ord(m)- 8;
   Writeln(k,d+k);
End.
```

9. Найдите ошибки в следующей программе и объясните, какие правила языка Паскаль здесь нарушены:

```
Program mistake_2;
Type month=(january, february, march, april, may, june, july
   august, september, october, november, december);
   winter=december..february;
   spring=marth..may;
Var m:month; k:1..12;
Begin
   Write('Введите месяц'); Readln(m);
   If m>spring Then m:=june;
   For k:=Ord(january) To Ord(m) Do m:=succ(m);
   Writeln(m);
End.
```

10. День учителя ежегодно отмечается в первое воскресенье октября. Дано натуральное число n, обозначающее номер года. Определите число, на которое приходится День учителя.

11. Рассмотрим некоторое натуральное число. Если это не палиндром, то изменим порядок его цифр на обратный и сложим исходное число с получившимся. Если сумма не палиндром, то над ней повторяется то же действие и т.д., пока не получится палиндром. Даны натуральные числа k, m, l (k<l). Проверить, верно ли, что для любого натурального числа из диапазона от k до l процесс завершается не позднее, чем после m таких действий.

12. Найдите 100 первых простых чисел.

13. Дано натуральное число n, целые числа a_1, a_2, ..., a_n. Рассмотрите отрезки последовательности a_1, a_2, ..., a_n (подпоследовательности идущих подряд членов), состоящих из:

а) полных квадратов;

б) степеней пятерки;

в) простых чисел.

В каждом случае получите наибольшую из длин рассматриваемых отрезков.

Контрольные работы

Простые типы данных. Символьный тип. Перечисляемые и интервальные типы

Контрольная работа № 1

Время выполнения 4 — 6 часов.

Вариант 1

1. Дано натуральное число:
   - найти сумму цифр этого числа;
   - верно ли, что число начинается и заканчивается одной и той же цифрой.
2. Найти все трехзначные числа, такие, что сумма цифр равна A, а само число делится на B (A и B вводятся с клавиатуры).
3. Дано натуральное число. Приписать к нему такое же число.

Вариант 2

1. Дано натуральное число:
   - найти произведение цифр числа;
   - верно ли, что в данном числе нет данной цифры A (цифру A вводить с клавиатуры).
2. Найти все трехзначные числа, которые при увеличении на 1 делятся на 2, при увеличении на 2 делятся на 3, при увеличении на 3 делятся на 4, а при увеличении на 4 делятся на 5.
3. Из данного натурального числа удалить все цифры A (A вводится с клавиатуры).

Вариант 3

1. Дано натуральное число:
   - найти количество цифр данного числа;
   - верно ли, что данное число заканчивается на нечетную цифру.
2. Найти количество трехзначных чисел, сумма цифр которых равна A, а само число заканчивается цифрой B (A и B вводятся с клавиатуры).
3. Найти все симметричные натуральные числа из промежутка от A до B (A и B вводятся с клавиатуры).

Вариант 4

1. Дано натуральное число:
   - найти количество четных цифр числа;
   - верно ли, что данная цифра A встречается в числе более двух раз (A вводить с клавиатуры).

2. Найти все четырехзначные числа, у которых сумма крайних цифр равна сумме средних цифр, а само число делится на 6 и 27.

3. Найти количество различных цифр данного натурального числа.

Вариант 5

1. Дано натуральное число:
   - найти первую и последнюю цифры числа;
   - верно ли, что сумма цифр данного числа равна A (A вводится с клавиатуры).

2. Найти все трехзначные числа, которые при делении на 2 дают остаток 1, при делении на 3 — остаток 2, при делении на 4 — остаток 3, а само число делится на 5.

3. Дано натуральное число. Приписать к нему такое же.

Вариант 6

1. Дано натуральное число:
   - сколько раз данная цифра A встречается в данном числе (A вводится с клавиатуры);
   - верно ли, что в данном числе сумма цифр больше B, а само число делится на B (B вводиться с клавиатуры).

2. Найти все четырехзначные числа, в которых есть две одинаковые цифры.

3. Из данного натурального числа удалить все цифры A (A вводится с клавиатуры).

Вариант 7

1. Дано натуральное число:
   - найти вторую (сначала) цифру данного числа;
   - верно ли, что данное число делится на A, B и C (A, B и C вводятся с клавиатуры).

2. Найти все двузначные числа, которые при умножении на 2 заканчиваются на 8, а при умножении на 3 — на 4.

3. Найти количество различных цифр данного натурального числа.

Вариант 8

1. Дано натуральное число:
   - найти количество цифр данного числа, больших A (A вводится с клавиатуры);
   - верно ли, что данное число принадлежит промежутку от A до B и кратно 3, 4 и 5 (A и B вводятся с клавиатуры).

2. Найти сумму всех чисел из промежутка от A до B, кратных 13 и 5 (A и B вводятся с клавиатуры).

3. Найти все симметричные натуральные числа из промежутка от A до B (A и B вводятся с клавиатуры).

Вариант 9

1. Дано натуральное число:
   - сколько четных цифр в данном целом числе;
   - верно ли, что в данном числе встречаются цифры A и B (A и B вводятся с клавиатуры).

2. Найти все симметричные четырехзначные числа. Пример: 7667, 1331.

3. Дано натуральное число. Приписать к нему такое же.

Вариант 10

1. Дано натуральное число:
   - сколько раз первая цифра встречается в данном числе;
   - верно ли, что данное число начинается на A, а заканчивается на B (цифры A и B вводятся с клавиатуры).
2. Найти все четырехзначные числа, в которых ровно две одинаковых цифры.
3. Найти количество различных цифр данного натурального числа.

Вариант 11

1. Дано натуральное число:
   - найти две первые цифры числа;
   - верно ли, что первая цифра данного числа — четная.
2. Найти все трехзначные числа, которые состоят из разных цифр, а их сумма равна A (A вводится с клавиатуры).
3. Найти все симметричные натуральные числа из промежутка от A до B (A и B вводятся с клавиатуры).

Контрольная работа № 2

Время выполнения 4—6 часов.

Вариант 1

1. Найти количество делителей натурального числа. Сколько из них четных?
2. Найти все натуральные числа a, b и c из интервала от 1 до 20, для которых выполняется равенство $a^2 + b^2 = c^2$.

Вариант 2

1. Найти сумму нечетных делителей натурального числа.
2. Найти все равновеликие прямоугольники, стороны которых выражены целыми числами a и b, а площадь равна S (a и b принадлежат интервалу от 1 до 20, а S вводится с клавиатуры).

Вариант 3

1. Найти все натуральные числа из промежутка от 1 до 200, у которых количество делителей равно N (N вводить с клавиатуры).
2. Найти все натуральные числа a, b и c из интервала от 1 до 20, для которых выполняется равенство $a + b^2 = c^2$.

Вариант 4

1. Найти все натуральные числа из промежутка от 1 до 200, у которых сумма делителей равна S (S вводить с клавиатуры).
2. Найти все такие тройки натуральных чисел x, y и z из интервала от 1 до 20, для которых выполняется равенство $x^2 - y = z^2$.

Вариант 5

1. Найти количество делителей натурального числа, больших K (K вводить с клавиатуры).

2. Найти все натуральные числа a, b и c из интервала от 1 до 20, для которых выполняется равенство $a^2b = c^2$.

Вариант 6

1. Найти сумму целых чисел из промежутка от 1 до 200, у которых ровно 5 делителей.
2. Найти все такие тройки натуральных чисел x, y и z из интервала от 1 до 20, для которых выполняется равенство $x^2 + y^2 = z^2$.

Вариант 7

1. Найти все целые числа из промежутка от 100 до 300, у которых сумма делителей равна K (K вводить с клавиатуры).
2. Найти все такие тройки натуральных чисел x, y и z из интервала от 1 до 20, для которых выполняется равенство $x^2 + y^2 - z^2 = 0$.

Вариант 8

1. Найти все натуральные числа из промежутка от a до b, у которых количество делителей превышает заданное число K.
2. Найти все натуральные числа a, b и c из интервала от 1 до 20, для которых выполняется равенство $a + b = c^2$.

Вариант 9

1. Найти сумму четных делителей натурального числа.
2. Найти все равновеликие прямоугольные треугольники, катеты которых выражены целыми числами a и b, а площадь равна S (a и b принадлежат интервалу от 1 до 20, а S вводится с клавиатуры).

Вариант 10

1. Найти количество нечетных делителей натурального числа, больших K (K вводить с клавиатуры).
2. Найти все натуральные числа x, y и z из интервала от 1 до 20, для которых выполняется равенство $xy^2 = z^2$.

Процедуры и функции. Рекурсия

Упражнение № 9. Процедуры

Пример 25. *Составить программу, которая будет находить a^n, то есть n-ю степень числа a, где a и n — это целые числа и $n > 0$, вводимые с клавиатуры.*

Решение. Составим процедуру, которая вычисляет степень целого числа.

```
Program Example_25;
Var a, n: Integer;
   s: Longint;
Procedure Degree(x,y: Integer; Var st: Longint);
Var i:Integer;
Begin
   st:=1;
   For i:=1 To y Do st:=st*x;
```

```
End;
Begin
   Writeln(' введите два числа'); {ввод значений}
   Readln(a,n);
   Degree(a,n,s); {обращение к процедуре}
   Writeln('Результат',s); {вывод значения a^n}
   Readln;
End.
```

Пример 26. *Даны два целых числа. Поменять местами их значения.*

Решение. Менять местами значения двух чисел можно двумя способами — через промежуточную переменную или без нее. Напишем процедуру, воспользовавшись первым способом.

```
Program Example_26;
Var a, b: Integer;
Procedure Swap (Var x, y:Integer);
Var z: Integer;
Begin
   z:=x; x:=y; y:=z;
End;
Begin
   Writeln('введите два числа'); {ввод значений}
   Readln(a, b);
   Swap(a, b) {обращение к процедуре}
   Writeln('a=',a,' b=',b); {вывод новых значений}
   Readln;
End.
```

Задания для самостоятельной работы

1. Используя процедуру из примера 25, вычислить значение выражения

$$y = a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5,$$

где коэффициенты a_1, a_2, a_3, a_4, a_5 и x — это числа, вводимые с клавиатуры.

2. Используя процедуру из примера 26, упорядочить значения трех переменных a, b и c в порядке их возрастания.

3. Даны координаты трех вершин треугольника. Найти длины всех его сторон.

4. Дано натуральное число. Найти все его делители. Подсчитать их число.

5. Даны два натуральных числа. Определить, является ли первое число перевертышем второго?

6. Даны координаты трех вершин треугольника ABC и даны координаты четвертой точки D. Определить, является ли эта точка внутренней точкой треугольника.

Упражнение № 10. Функции

Пример 27. *Составить программу, подсчитывающую число сочетаний без повторения из N элементов по K элементов.*

Решение. Число сочетаний без повторения считается по формуле

$$C_n^k = \frac{n!}{k!(n-k)!}.$$

Обозначим n, k — переменные для хранения введенных чисел; C — переменная для хранения результата.

Чтобы подсчитать число сочетаний без повторения, необходимо вычислить $n!$, $(n-k)!$, $k!$.

Опишем функцию, вычисляющую факториал числа n ($n! = 1 \cdot 2 \cdot \ldots \cdot n$).

```
Program Example_27;
Var n,k: Integer;
a1, a2, a3, c: Longint;
Function factorial(n:Integer):Longint;
Var i: Integer;
     rez: Longint;
Begin
   rez:=1;
   For i:=1 To n Do rez:=rez*i;
   factorial:=rez;
End;
Begin
   Writeln('Введите n и k для подсчета числа сочетаний из n по k ');
   Readln(n,k);
   a1:= factorial(n); { вычисление n!}
   a2:= factorial(k); {вычисление k!}
   a3:= factorial(n-k); {вычисление (n-k)!}
   c:=a1 Div (a2*a3); {результат}
   Writeln(c);
   Readln;
End.
```

Пример 28. *Написать функцию, подсчитывающую количество цифр числа. Используя ее, определить, в каком из двух данных чисел больше цифр.*

Решение. Для решения задачи вспомним, как подсчитать количество цифр.

Для этого можно выделять последнюю цифру до тех пор, пока число не станет равным нулю. При этом каждый раз надо увеличивать счетчик на 1 (начальное значение счетчика — 0).

```
Program Example_28;
Var n1, n2: Longint;
    k1, k2: Byte;
Function Quantity(x: Longint): Byte;
Var k: Byte;
Begin
   k:=0;
   While x <> 0 Do
   Begin
      Inc(k);
      x:=x Div 10;
   End;
   Quantity:=k;
End;
Begin
   Writeln('Введите два числа'); {ввод чисел}
   Readln(n1, n2);
```

```
    k1:= Quantity(n1); {количество цифр первого числа}
    k2:= Quantity(n2); {количество цифр второго числа}
    {Сравнение количества цифр в числах}
    If k1=k2 Then Writeln('Одинаковое количество цифр')
    Else If k1>k2 Then Writeln('В первом числе цифр больше')
    Else Writeln('Во втором числе цифр больше');
    Readln;
End.
```

Задания для самостоятельной работы

1. Найти сумму цифр числа.
2. Найти первую цифру числа.
3. Найти количество делителей числа.
4. Найти числа из промежутка от A до B, у которых больше всего делителей.
5. Найти сумму всех делителей числа.
6. Определить, является ли число совершенным, то есть равно ли оно сумме своих делителей, кроме самого себя (используя преобразованную функцию из предыдущего примера).
7. Определить, является ли число простым.
8. Среди чисел из интервала от A до B найти все простые.
9. Составить программу, проверяющую, является ли число палиндромом (например, число 12721 — палиндром).
10. Определить, является ли число автоморфным, то есть квадрат этого числа заканчивается этим же числом, например число 6, так как его квадрат 36 заканчивается на 6 или число 25 — его квадрат 625.
11. Используя функцию из предыдущей задачи, найти все автоморфные числа из промежутка от A до B.

Примечание. Задачи 8 и 11 можно модифицировать — найти N первых таких чисел или найти первые N таких чисел, начиная с числа M.

12. Составить программу нахождения наибольшего общего делителя нескольких чисел, используя функцию нахождения НОД двух чисел.
13. Составить программу, вычисляющую наименьшее общее кратное четырех заданных с клавиатуры чисел (использовать функцию из предыдущего примера).
14. Дано четыре числа. Вывести на экран наибольшую из первых цифр заданных чисел. Например, если $a = 25$, $b = 730$, $c = 127$, $d = 1995$, то должна напечататься цифра 7.
15. Дано натуральное число n. Выяснить, имеются ли среди чисел n, $n + 1$, ..., $2n$ близнецы, т.е. простые числа, разность между которыми равна двум.

Упражнение № 11. Примеры рекурсивного программирования

Пример 29. *Вычисление факториала натурального числа.*

Решение. Для того чтобы вычислить $N!$, надо знать значение $(N-1)!$ и умножить его на N, при этом $1!=1$. В общем виде это можно записать так:

$$N! = \begin{cases} 1, & \text{при } N = 1, \\ N(N-1)!, & \text{при } N > 1. \end{cases}$$

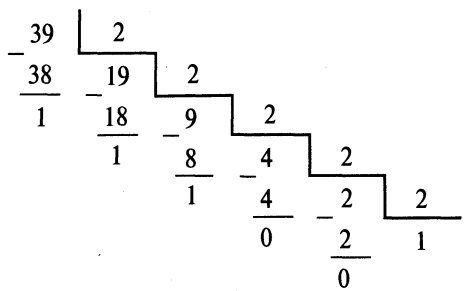
```
Function factorial(n: Integer): Longint;
Begin
   If n=1 Then factorial:=1
   Else factorial:=n*factorial(n-1);
End;
```

Найдем 5!. Как же будет вычисляться факториал этого числа? Первый вызов этой функции будет из основной программы (Например, `a:= factorial(5)`, где переменной *a* присваиваем значение 5!).

Так как $N \neq 1$, то пойдем по ветке `Else` и функции `Factorial` присваиваем значение `n*Factorial(n-1)`, то есть надо умножить 5 на значение функции `Factorial(4)`. Поэтому обращаемся второй раз к этой же функции, но передаем ее новое значение параметра — 4. Так делаем до тех пор, пока не передадим значение, равное 1. Тогда $N = 1$, а поэтому значение функции `Factorial:=1`. Таким образом, $N = 1$ — это условие, по которому процесс входа в следующую рекурсию заканчивается. Идет возвращение в точку вызова и подстановка в оператор присвоения значения вычисленной функции. То есть возвращаемся в предыдущую функцию для $n=2$: `Factorial:=n*Factorial(n-1)`, значит, `Factorial:=2*1`, следовательно, `Factorial(2)=2`. И возвращаемся дальше. Таким образом, получаем значение `Factorial(5)=120`, это значение и присвоим переменной *a*.

Пример 30. *Перевод натурального числа из десятичной системы счисления в двоичную.*

Решение. Для решения этой задачи рассмотрим сначала, как перевести число из десятичной системы счисления в двоичную. Пусть есть число 39, которое и надо представить в двоичной системе. Для этого разделим его на 2, получим целую часть и остаток от деления. Целую часть снова делим на 2 и получаем целую часть и остаток. Так делаем до тех пор, пока целую часть можно делить на 2 (то есть пока она не станет равной 1).



Теперь, начиная с этой единицы, выписываем в обратном порядке все остатки от деления, это и будет запись числа 39 в двоичной системе счисления:

$39_{10} = 100111_2$.

Таким образом можно переводить любое натуральное число из десятичной системы счисления в двоичную, а также и в другие системы (например, восьмеричную или шестеричную).

Опишем процедуру:

```
Procedure Rec(n: Integer);
Begin
   If n>1 Then Rec(n Div 2);
```

```
    Write(n Mod 2);
End;
```

Первая цифра (1) выводится на экран из последнего вызова, следующая цифра (0) из предпоследнего и так далее, последняя (1)— из первого. Таким образом, вывод очередной цифры происходит перед тем, как выйти из данной функции.

Задания для самостоятельной работы

1. Составить рекурсивную программу ввода с клавиатуры последовательности чисел (окончание ввода — 0) и вывода ее на экран в обратном порядке.
2. Составить программу вычисления значений функции Аккермана для неотрицательных чисел n и m, вводимых с клавиатуры.

$$A(n,m) = \begin{cases} m+1, & \text{если } n = 0, \\ A(n-1,1), & \text{если } n \neq 0,\ m = 0, \\ A(n-1, A(n, m-1)), & \text{если } n > 0,\ m \geq 0. \end{cases}$$

3. Найти сумму первых N членов арифметической (геометрической) прогрессии.
4. Найти первые N чисел Фибоначчи. Каждое число Фибоначчи равно сумме двух предыдущих чисел при условии, что первые два равны 1 (1, 1, 2, 3, 5, 8, 13, 21,...), поэтому в общем виде n-е число можно определить так:

$$\Phi(n) = \begin{cases} 1, & \text{если } k = 1,\ n = 2, \\ \Phi(n-1) + \Phi(n-2), & \text{если } n > 2. \end{cases}$$

5. Определить, является ли заданное натуральное число простым.
6. Для заданного натурального числа $N \geq 1$ определить единственное натуральное число a, для которого выполняется неравенство: $2^{a-1} \leq N < 2^a$.
7. Функция $F(n)$ определена для целых положительных чисел следующим образом:

$$F(n) = \begin{cases} 1, & \text{если } n = 1, \\ \sum_{i=2}^{n} F(n \text{ div } i), & \text{если } n \geq 2. \end{cases}$$

Вычислить значения этой функции для $n = 5, 6, 7, \ldots, 20$. Прорисовать вызовы (обратить внимание, что из одной функции может быть несколько вызовов этой же функции).

Файлы

Упражнение № 12. Файловый тип данных. Открытие файла. Чтение и запись

Пример 31. *Прочитаем файл целых чисел и выведем их на экран:*

```
Assign(F1,'a:int.dat'); {связываем с внешним файлом}
Reset(F1); {открываем его для чтения}
While Not EOF (F1) Do {пока не достигнут конец файла F1}
```

```
Begin
   Read(F1,n); {считываем очередное число}
   Write(n,''); {выводим его на экран}
End;
Close(F1); {закрываем файл}
```

Пример 32. *Создадим файл целых чисел с именем Dan1.dat, причем ни одно из чисел не равно 0.*

Решение. Первоначально «свяжем» файловую переменную с конкретным внешним файлом при помощи процедуры `Assign`. Откроем файл для записи — процедура `Rewrite`. Конец ввода чисел— ввод числа ноль.

```
Program Example_32;
Var F: File Of Integer;
     n: Integer;
Begin
   Assign(F, 'a:dan1.dat'); {связываем с внешним файлом}
   Rewrite(F); {открываем его для записи}
   Writeln('конец ввода чисел - 0');
   Repeat {пока не будет введен 0}
      Writeln('введите число');
      Readln(n); {ввод числа с клавиатуры}
             {если введено число, отличное от 0,
                то дописываем его в данную строку файла F1}
      If n<>0 Then Write(F, n);
   Until n=0; {если ввели 0, то заканчиваем запись данного файла}
   Close(F); {закрываем файл}
End.
```

Пример 33. *В файле Dan1.dat записаны целые числа (см. предыдущую задачу). Вычислить сумму элементов и результат вместе с исходными данными записать в файл Dan2.dat.*

```
Program Example_33;
Var F1, F2: File Of Integer;     {файловые переменные }
    S, N: Integer;
Begin
      {с именем файла F1 связывается внешний файл на дискете}
   Assign(F1,'Dan1.dat');
   Reset(F1);
      {открытие файла F1 для чтения }
      {с именем файла F2 связывается внешний файл на дискете}
   Assign(F2,'Dan2. dat');
   Rewrite(F2); {открытие файла F2 для записи}
   S:=0;
   While Not EOf(F1) Do {проверка на конец файла F1}
   Begin
      Read(F1,N); {чтение элемента из файла F1}
      Write(F2,N); {запись элемента в файл F2}
      S:=S+N; {вычисление суммы}
   End;
   {запись суммы элементов в конец файла F2}
```

```
    Write(F2,S);
    Write('Результат находится в файле Dan2.dat')
    Close(F1); {закрытие файла F1 для чтения}
    Close(F2); {закрытие файла F2 для записи}
    Readln;
End.
```

Задания для самостоятельной работы

1. Дан файл *F*, компоненты которого являются целыми числами. Найти:
   а) число элементов;
   б) наибольшее из значений; если их несколько, то подсчитать число таких элементов;
   в) среднее арифметическое элементов.
2. Даны символьные файлы *F* и *G*. Записать в файл *H*:
   а) все компоненты файлов *F* и *G*;
   б) все латинские буквы (большие и маленькие) файла *F*.
3. Дан целочисленный файл *A*. Записать в файл *B* все четные числа, а в файл *C* все нечетные.
4. Даны два файла *A* и *B* (тип элементов одинаковый). Поменять местами содержимое этих файлов.

   Примечание. Можно решить эту задачу двумя способами. Первый — использовать процедуру `Rename`, второй — переписать все содержимое первого файла в промежуточный, затем все содержимое второго файла в первый, а теперь вернуть все из промежуточного во второй. Для лучшей реализации второго способа можно написать процедуру, которая будет переписывать все из одного файла в другой. В этом случае в разделе типов надо описать свой тип данных для файловых переменных.
5. Дан символьный файл *F*. Записать в перевернутом виде элементы файла *F* в файл *G*.
6. Даны два файла, причем первый *A* — целочисленный, то есть его элементы целые числа, а второй *B* — символьный. Вывести на экран все числа первого, а рядом с ними элементы второго с этим номером, если элемента второго файла с данным номером нет, то сообщить об этом.

Упражнение № 13. Текстовые файлы

Пример 34. *Дан текстовый файл, содержащий только целые числа, в каждой строке может быть несколько чисел, которые разделяются пробелами. Вывести на экран все числа с учетом разбиения на строки и подсчитать число элементов в каждой строке.*

Решение. Пусть в файле содержится следующая информация:

```
-32   16    0    8    7
  4    5    9   13   11   -5   -8
  6   -8    0  -12
  5    4    3    2    1   12
  1    2
 -1   -2   -4
 -1   -2    4
```

Этот файл можно создать в среде Turbo Pascal таким образом:
- создать новый файл (команда *New* меню *File*);
- записать все числа в строках через пробелы;
- сохранить его, например «*a: int1.dat*».

Теперь этот файл будем использовать в программе.

```
Program Example_34;
Var F: Text;
   x, k: Integer;
Begin
   Assign(F, 'a: int1.dan'); {связываем с внешним файлом}
   Reset(F); {открываем для чтения}
   While Not Eof(F) Do {пока не конец файла}
   Begin
      k:=0; {начальное число элементов строки}
      While Not Eoln(F) Do {пока не конец строки}
      Begin
         Read(F, x); {считываем очередное число}
         Write(x,''); {вывод его на экран}
         Inc(k); {увеличиваем счетчик}
      End;
      Writeln(' в строке', k,'элементов');
      Readln(F); {переходим к следующей строке файла}
   End;
   Close(F); {закрываем файл}
   Readln;
End.
```

Пример 35. *Дан текстовый файл, содержащий программу на языке Паскаль. Проверить эту программу на несоответствие числа открывающих и закрывающих круглых скобок. Считать, что каждый оператор программы занимает не более одной строки файла.*

Решение. Так как по условию задачи каждый оператор занимает не более одной строки, то будем подсчитывать число открывающих и закрывающих скобок в одной строке. Надо заметить, что при проверке правильности расстановки скобок в строке число рассмотренных закрывающих скобок не должно превышать числа уже рассмотренных открывающих скобок. Кроме того, такой файл должен быть создан заранее.

```
Program Example_35;
Var F: Text;
   k1, k2, n: Integer;
   Ch: Char;
   Logic,Pp: Boolean;
Begin
   {с именем файла F связывается внешний файл}
   Assign(F, 'a:...');
   Reset(F); {открытие файла F для чтения}
   n:=0; {счетчик количества строк}
   Logic:=True; {пока ошибки не определены, то значение True}
   While Not Eof(F) Do {пока не конец файла}
   Begin
```

```
        Inc(n); {увеличиваем счетчик количества строк}
        k1:=0; {счетчики количества открывающих скобок}
        k2:=0; {счетчики количества закрывающих скобок}
        Pp:=False;
            {Pp предназначена для определения ошибки расстановки скобок
            в строке, начальное значение False, так как пока ошибки
            расстановки не было}
        While Not Eoln (F) Do
        {пока не конец текущей строки файла}
        Begin
            Read(F, Ch); {очередной символ строки}
            {если встретили открывающую скобку, то увеличиваем  их
            счетчик}
            If Ch = '(' Then Inc(K1);
            {если встречена закрывающая скобка, то если она стоит не
            раньше открывающей, значение K1<K2, поэтому просто уве-
            личиваем счетчик этих скобок, иначе помечаем Pp значени-
            ем True}
            If (Ch = ')') Then
            If (K1<K2) Then Inc(K2)
            Else Pp:=True;
        End;
        {если не все закрывающие скобки расставлены (K1<>K2) или
        одна из закрывающих скобок стоит раньше открывающей (Pp=True),
        то была ошибка расстановки}
        If (K1<>K2) Or Pp Then
        Begin
            Writeln('Ошибка в ',N,'строке');
            {помечаем, что в одной из строк была ошибка}
            Logic:=False;
        End;
        Readln(F);
        {переходим на следующую строку файла}
    End;
    {если значение остается истинным, то ошибок расстановки не было}
    If Logic Then Writeln('Скобки расставлены правильно');
    Close(F); {закрываем файл}
    Readln;
End.
```

Задания для самостоятельной работы

1. Дан текстовый файл, содержащий целые числа. Найти:
   а) максимальный элемент в каждой строке;
   б) номер данного числа; если такого нет в данной строке, то сообщить об этом.
2. Дан текстовый файл, содержащий строки. Найти:
   а) число строк;
   б) число строк, начинающихся и заканчивающихся одинаковыми символами;
   в) самые короткие строки;

г) симметричные строки.

3. Дан текстовый файл. Вставить в начало каждой строки ее номер и записать преобразованные строки в новый файл.

4. Даны два текстовых файла. Записать в третий только те строки, которые есть и в первом, и во втором файлах.

5. Дан текстовый файл. Дописать в его конце следующие данные: число строк, число символов в каждой строке, число элементов в каждой строке.

Массивы

Упражнение № 14. Одномерные массивы. Заполнение массива. Простейшие операции с массивами

Пример 36. *Найти сумму пяти целых чисел.*

Решение. Для решения этой задачи необходимо описать пять переменных для целых чисел и еще одну — для их суммы. Обозначим первые как *a*1, *a*2, *a*3, *a*4 и *a*5, а их сумму — *s*. Тогда можно составить такую программу, используя функцию нахождения суммы пяти чисел:

```
Program Example_36;
Var a1,a2,a3,a4,a5,s: Integer;
Function Sum(x1,x2,x3,x4,x5: Integer): Integer;
Begin
   Sum:=x1+x2+x3+x4+x5;
End;
Begin
   Writeln(' введите пять целых чисел');
   Readln(a1,a2,a3,a4,a5); {вводим пять целых чисел}
   s:=Sum(a1,a2,a3,a4,a5); {находим их сумму}
   Writeln('их сумма равна ',s); {вывод результата на экран}
   Readln;
End.
```

А как найти сумму, например, 100 целых чисел? Здесь необходимо использовать массив.

Пример 37. *Составить программу нахождения суммы элементов массива.*

Решение. Опишем две процедуры (формирования и вывода массива) и функцию нахождения суммы элементов, которые будем использовать в основной части. Заметим, что *заполнение и вывод массива можно осуществить только поэлементно*, то есть можно сначала присвоить значение первому элементу, затем второму и так далее, то же самое и с выводом на экран — выводим первый, второй, третий... и так до последнего. Будем вводить значения элементов массива с клавиатуры.

```
Program Example_37;
Const n=30; {n - это число элементов массива}
Type myarray=Array[1..n] Of Integer;
Var A: myarray;
   s: Integer;
   {s - значение этой переменной будет
   равно сумме всех элементов массива}
Procedure Init1(Var m: myarray);
```

```
Var i: Integer;
{i - это переменная для работы с элементами массива}
Begin
   Writeln('введите ',n, ' чисел');
   For i:=1 To n Do {ввод массива с клавиатуры}
   Readln(m[i]); {чтение i-го элемента}
End;
Procedure Print(m: myarray);
Var i: Integer;
Begin
   For i:=1 To n Do {вывод массива}
      Write(m[i]: 3); {вывод i-го элемента}
   Writeln;
End;
Function Sum(m: myarray): Integer;
Var i, sum: Integer;
Begin
   sum:=0; {начальное значение суммы}
   For i:=1 To n Do sum:=sum+m[i]; {к уже найденной сумме
   первых (i-1) элементов прибавляем i-й элемент}
End;
Begin
   Init1(A); {обращение к процедуре формирования}
   Print(A); {вывод массива}
   s:=Sum(A); {нахождение суммы элементов}
   Writeln('их сумма равна ',s); {вывод результата на экран}
   Readln;
End.
```

Первый способ задания одномерного массива — это задание с клавиатуры (он был рассмотрен в примере выше — процедура *Init1*).

Второй способ задания — это задание с помощью генератора случайных чисел; этот способ более удобен, когда много элементов в массиве.

Пример 38. *Составим программу заполнения и распечатки одномерного массива с помощью генератора случайных чисел. Процедура вывода уже составлена ранее, а процедуру формирования напишем новую.*

```
Program Example_38;
Const n=30; dd=51;
{n - это число элементов массива, dd - для генератора случайных чисел}
Type myarray = Array [1..n] Of Integer;
Var A: myarray;
Procedure Init2(Var m:myarray);
{процедура заполнения (инициализации) массива случайными числами}
Var i:Integer;
Begin
   For i:=1 To n Do m[i]:=-25+Random(dd);
   {Random выбирает случайное число из отрезка от 0 до dd-1, тогда
   i-му элементу массива будет присвоена сумма выбранного случай-
   ного числа и -25, таким образом, массив будет заполняться слу-
   чайными числами от -25 до -25+(dd-1), то есть до -26+dd}
```

```
End;
Procedure Print(m:myarray); {процедура вывода (распечатки) массива}
...
Begin
   Randomize; {включение генератора случайных чисел}
   Init2(A); {обращение к процедуре заполнения массива}
   Print(A); {обращение к процедуре вывода заполненного массива}
   Readln;
End.
```

Третий способ задания — это чтение чисел из файла.

Пример 39. *Пусть в файле записано несколько строк, а в каждой из них по 30 целых чисел. Составить программу с использованием процедуры заполнения массива из файла.*

```
Program Example_39;
Const n=30; {n - это число элементов массива,}
Type myarray = Array [1..n] Of Integer;
Var A: myarray;
   F: text;
Procedure Init3(Var m:myarray);
{процедура заполнения (инициализации) массива}
Var i:Integer;
Begin
   For i:=1 To n Do Read(f, m[i])
   {чтение из файла очередного числа}
End;
Procedure Print(m:myarray);
{процедура вывода (распечатки) массива}
...
Begin {связываем файловую переменную с конкретным внешним файлом}
   Assign(F, '...');
   Reset(F); {открываем его для чтения}
   While Not Eof(F) Do
   Begin
      {считываем очередную строку}
      Init3(A); {обращение к процедуре заполнения массива}
      Print(A); {обращение к процедуре вывода}
      Readln(F);
   End;
   Readln;
End.
```

Задания для самостоятельной работы

1. Найти сумму положительных элементов массива.
2. Найти сумму всех четных элементов массива (или сумму элементов, кратных заданному числу).
3. Найти сумму всех четных элементов массива, стоящих на четных местах, то есть имеющих четные номера.
4. Найти сумму первых пяти элементов массива.

5. Найти сумму элементов с $k1$-го по $k2$-й, где $k1$ и $k2$ вводятся с клавиатуры. Сделать проверку корректности их ввода.

6. Найти сумму элементов, больших данного числа A (A вводить с клавиатуры).

7. Найти сумму элементов, принадлежащих промежутку от A до B (A и B вводить с клавиатуры).

Примечание. В задачах 1 — 7 можно находить не только сумму, но и произведение, или проверять, что сумма первых двух цифр равна сумме последних двух цифр.

8. Найти максимальный элемент массива и его номер при условии, что все элементы различные.

9. Найти номера всех отрицательных элементов (вывести их на экран); если таких нет, то сообщить об этом.

10. Найти номера всех элементов с максимальным значением.

11. Найти минимальный элемент.

12. Найти число нечетных элементов.

13. Найти число отрицательных элементов.

14. Сколько элементов массива превосходят по модулю заданное число A?

15. Найти все элементы, кратные 3 или 5. Сколько их?

16. Есть ли в данном массиве два соседних положительных элемента? Найти номера первой (последней) пары.

17. Есть ли в данном массиве элемент, равный заданному числу? Если есть, то вывести номер одного из них.

18. Найти число четных чисел среди пяти целых чисел (без использования массива).

19. Если существует треугольник со сторонами a, b и c, то напечатать «ДА», в других случаях напечатать «НЕТ» (значении a, b и c вводить с клавиатуры).

Упражнение № 15. Методы работы с элементами одномерного массива

Пример. Заменить отрицательные элементы на противоположные по знаку.

Решение. Для этого опишем процедуру. Ей будем передавать один параметр — массив, который будет результатом ее выполнения, так как некоторые элементы могут быть заменены.

```
Procedure Substitation1(Var m: myarray);
Var i: Integer;
Begin
   For i:=1 To n Do If m[i]<0 Then m[i]:=-m[i];
End;
```

Пример. Прибавить к каждому элементу число 25.

Решение. Преобразуем предыдущую процедуру.

```
Procedure Substitation2(Var m: myarray);
Var i: Integer;
Begin
   For i:=1 To n Do m[i]:=m[i] + 25;
End;
```

Пример. Если элемент четный, то прибавить к нему первый, если нечетный — прибавить последний. Первый и последний элементы не изменять.

Решение. Будем просматривать каждый элемент, кроме первого и последнего, и если он четный, то есть делится на 2 без остатка, то увеличим его на значение первого элемента, иначе — на значение последнего элемента.

```
Procedure Substitation3 (Var m: myarray);
Var i: Integer;
Begin
   For i:=2 To n-1 Do
   If m[i] Mod 2=0 Then m[i]:=m[i] + m[1]
   Else m[i]:=m[i]+m[n];
End;
```

Пример. Даны два одномерных массива одинаковой размерности. Получить третий массив такой же размерности, каждый элемент которого равен сумме соответствующих элементов данных массивов.

Решение. Пусть даны два массива A и B, состоящие из пяти элементов. Получим из них массив C, состоящий тоже из пяти элементов, первый элемент этого массива равен сумме первых элементов массивов A и B, второй — сумме вторых элементов данных массивов и так далее.

i	1	2	3	4	5
A	14	2	7	8	9
B	3	6	5	12	4
C	17	8	12	20	13

Для решения этой задачи опишем процедуру, которой передаются три параметра. Первые два — это два исходных одномерных массива, третий — это одномерный массив той же размерности, который является результатом выполнения этой процедуры и заполняется по указанному правилу.

```
Procedure Sum_Ar(a, b: myarray; Var c: myarray);
Var i: Integer;
Begin
   For i:=1 To n Do c[i]:=a[i]+b[i];
End;
```

Пример. Дан первый член арифметической прогрессии и ее разность. Найти первые n членов.

Решение. Пусть $a1$ — это первый член прогрессии, а k — это ее разность, тогда i-й член можно найти по правилу — $a[i] = a[i-1]+k$, или $a[i] = a1 + k(i-1)$, если $i = 1$, то $a[i] = a1$. Опишем процедуру, которой передаем эти два параметра и результатом выполнения является одномерный массив. Первый элемент его равен первому члену прогрессии, второй — второму и так далее. Возьмем в качестве основной первую формулу.

```
Procedure Progress(a1, k: Integer; Var a: myarray);
Var i: Integer;
Begin
   a[1]:=a1;
   For i:=2 To n Do a[i]:=a[i-1]+k
End;
```

Пример. Даны два одномерных массива A и B. Найти их скалярное произведение.

Решение. Скалярным произведением двух массивов одинаковой размерности называется сумма произведений соответствующих элементов. Это можно записать так:

$a[1]b[1] + a[2]b[2] + ... + a[n-1]b[n-1] + a[n]b[n]$, где n — это число элементов в массивах.

Тогда можно составить следующую функцию:

```
Function Sp(a, b: myarray): longint;
Var i: Integer;
   s: Longint;
Begin
   s:=0;
   For i:=1 To n Do s:=s+a[i]*b[i];
   Sp:=s;
End.
```

Задания для самостоятельной работы

1. Изменить знак у максимального по модулю элемента массива.
2. Заменить все четные элементы на их квадраты, а нечетные удвоить.
3. Вычесть из положительных элементов элемент с номером $k1$, а к отрицательным прибавить элемент с номером $k2$, нулевые элементы оставить без изменения.
4. К четным элементам прибавить A, а из элементов с четными номерами вычесть B.
5. Отрицательные элементы возвести в квадрат.
6. Даны два целочисленных массива, состоящие из одинакового числа элементов. Получить третий массив той же размерности, каждый элемент которого равен большему из соответствующих элементов данного массива.

Например, даны два массива A и B, состоящие из пяти элементов. Получим из них массив C, состоящий тоже из пяти элементов, первый элемент этого массива равен большему из первых элементов массивов A и B, второй — большему из вторых элементов данных массивов и так далее.

Таким образом, получим массив C:

i	1	2	3	4	5
A	14	2	7	8	9
B	3	6	5	12	4
C	14	6	7	12	9

7. Дан одномерный массив $A(a_1, a_2, ..., a_n)$. Найти массив B, той же размерности $B(b_1, b_2, ..., b_n)$, где:

а) $b_i = a_i^2 + 2a_i - 1$, для всех $i = 1, ..., n$;

б) $b_i = \begin{cases} 1\ (\text{True}), & \text{если } a_i \text{ делится на } k, \\ 0\ (\text{False}), & \text{если } a_i \text{ не делится на } k; \end{cases}$

в) $b_i = a_1 + a_2 + ... + a_i$, для всех $i = 1, ..., n$.

8. Дан первый член геометрической прогрессии и ее знаменатель. Найти первые n членов этой прогрессии.
9. Получить первые n чисел Фибоначчи — первые два числа равны 1, а каждое следующее равно сумме двух предыдущих.
10. Даны два массива. Найти среднее арифметическое элементов каждого и сравнить эти значения.
11. При выводе на экран:

а) после каждого элемента, кроме последнего, поставить точку;
б) пропустить все отрицательные.

Упражнение № 16. Удаление элементов из одномерного массива

Пример 40. *Удалить из массива максимальный элемент, если все элементы разные.*

Решение. Для того, чтобы решить данную задачу, необходимо:
- найти номер максимального элемента — *k*;
- сдвинуть все элементы, начиная с *k*-го, на один элемент влево;
- последнему элементу присвоить значение 0.

```
Program Example_40;
Const n=30; dd=51;
Type myarray = Array [1..n] Of Integer;
Var A: myarray;
   k:Integer; {k - номер максимального элемента}
Procedure Init2(Var m:myarray);
{процедура заполнения (инициализации) массива случайными числами}
...
Procedure Print1(n1: Integer; m: myarray);
{процедура вывода (распечатки) массива}
Var i: Integer;
Begin
   For i:=1 To n1 Do Write(m[i]:5);
   Writeln;
End;
Function Maximum(m: myarray): Integer;
Var i, max, maxi: Integer;
Begin
   max:=-maxint;
   { -maxint - целая константа, имеющая минимальное среди целых
   чисел значение, равное - 32 768 }
   For i:=1 To n Do {просмотр всех элементов массива}
   If m[i]>max Then
   {если данный элемент больше максимального элемента, найденного
   среди первых i-1 элементов, то}
   Begin
      max:=A[i]; {новое значение максимального элемента }
      maxi:=i; {номер максимального элемента в массиве}
   End;
   Maximum:=maxi;
End;
Procedure Delete(k1: Integer; Var m: myarray);
Var i: Integer;
Begin {сдвиг элементов на один влево}
   For i:=k1 To n-1 Do
   m[i]:=m[i+1]; {i-му элементу присваиваем значение (i+1)-го}
   m[n]:=0; {последний элемент равен 0}
End;
Begin
```

```
    Randomize; {включение генератора случайных чисел}
    Init2(A); {заполнения массива A}
    Print1(n,A); {вывод заполненного массива A}
    k:=Maximum(A); {поиск номера максимального элемента}
    Delete(k, A); {удаление элемента с номером k}
    Print1(n-1,A); {вывод нового массива A}
    Readln;
End.
```

Таким образом, на экране появятся следующие строки:

6 3 4 7 11 2 13 8 1 5 — начальный массив,
6 3 4 7 11 2 8 1 5 — массив после удаления максимального элемента.

Пример 41. *Предположим, что максимальный элемент встречается несколько раз.*

Решение. Когда необходимо удалять несколько элементов, то это лучше всего делать с конца массива, так как иначе надо будет снова возвращаться к элементу с номером, которым только что удаляли (это возникает тогда, когда подряд идут два максимальных элемента, если первый удалим, то на его месте будет стоять снова максимальный элемент). Это можно сделать при помощи цикла с параметром, который имеет следующий вид:

```
For i:=B Downto A Do <тело цикла>,
```

где значения переменной *i* будут уменьшаться на единицу, начиная от *B* до *A* (значение *B* должно быть меньше значения *A*).

Кроме того, номер максимального элемента запоминать не будем, а просмотрим массив с конца и если элемент имеет максимальное значение, то удалим его, при этом значение счетчика *k* будем увеличивать на 1. Для решения этой задачи надо изменить функцию *Maximum*, сейчас нам нужен не номер, а значение максимального элемента.

```
Program Example_41;
Const n=30; dd=51;
Type myarray = Array [1..n] Of Integer;
Var A: myarray;
   m, k, i:Integer; {m - значение максимального элемента,
   k - число удаленных элементов}
Procedure Init2(Var m:myarray);
{процедура заполнения (инициализации) массива случайными числами}
...
Procedure Print1(n1: Integer; m: myarray);
{процедура вывода (распечатки) массива}
...
Function Maximum(m: myarray): Integer;
Var i, max: Integer;
Begin
   max:=-maxint;
   For i:=1 To n Do {просмотр всех элементов массива}
   If m[i]>max Then max:=A[i];
   {новое значение максимального элемента}
   Maximum:=max;
End;
```

```
Procedure Delete(k1: Integer; Var m: myarray);
{процедура удаления элемента с данным номером}
...
Begin
   Randomize; {включение генератора случайных чисел}
   Init2(A); {заполнение массива A}
   Print1(n,A); {вывод заполненного массива A}
   {поиск значения максимального элемента}
   m:=Maximum(A);k:=0;
   {просмотр всех элементов, начиная с последнего}
   For i:=n Downto 1 Do
   If A[i]=m Then
   {если данный элемент имеет максимальное значение, то}
   {удаляем элемент с номером i}
   Begin
      Delete(i,A); Inc(k);
   End;
   Print1(n-k,A); {вывод нового массива A}
   Readln;
End.
```

Задания для самостоятельной работы

1. Удалить первый отрицательный элемент, если такой элемент есть.
2. Удалить все отрицательные элементы.
3. Удалить все элементы, большие данного числа A (A вводить с клавиатуры).
4. Удалить все четные элементы, стоящие на нечетных местах.
5. Удалить все повторяющиеся элементы, оставив только их первые вхождения, то есть получить массив различных элементов.
6. Удалить последний четный элемент.
7. Удалить все элементы, кратные 3 или 5.
8. Удалить все элементы, начиная с $k1$-го по $k2$-й ($k1$ и $k2$ вводить с клавиатуры). Сделать проверку корректности ввода значений $k1$ и $k2$ ($k1 \le k2$), если ввод некорректный, то вывести сообщение об ошибке и закончить работу.

Упражнение № 17. Вставка элементов в одномерный массив

Пример 42. *Вставка элемента после элемента с заданным номером. Вставить число 100 после пятого элемента массива.*

Решение. Пусть k — это номер элемента, после которого мы должны вставить элемент x (k и x будем вводить с клавиатуры). Тогда вставка осуществляется следующим образом:

- первые k элементов массива остаются без изменений;
- все элементы, начиная с $(k+1)$-го, необходимо сдвинуть на один назад;
- на место $(k+1)$-го элемента записываем значение x, то есть после k-го элемента массива.

```
Program Example_42;
Const n=10; dd=51;
Type myarray=Array[1.. n+1] Of Integer;
```

```
Var A: myarray;
   x, k:Integer;
   {x - вставляемое число, k - номер элемента, после которого
вставляем}
Procedure Init2(Var m: myarray); {процедура заполнения (инициа-
лизации) массива случайными числами}
...
Procedure Print1(n1: Integer; m: myarray); {процедура вывода (рас-
печатки) массива}
...
Procedure Insert1(k1, x1: Integer; Var m: myarray);
Var i: Integer;
Begin {сдвиг элементов на одну позицию назад}
   For i:=n Downto k1+1 Do m[i+1]:=m[i];
   m[k1+1]:=x1; {вставка элемента на место после k1-го}
End;
Begin
   Init2(A);
   Print1(n,A); { первый вывод начального массива из n элементов }
   Writeln('Номер элемента, после которого вставлять, ');
   Writeln('и вставляемое число');
   Readln(k,x); {ввод номера и вставляемого элемента}
   Insert1(k,x,A);
   Print1(n+1,A); {вывод массива после вставки в него}
   Readln;
End.
```

Пример 43. *Вставка элемента перед данным. Вставить число 100 перед пятым элементом массива.*

Решение. Эта вставка немногим отличается от предыдущей: в первой сдвигали назад все элементы, стоящие после k-го, то есть с $(k+1)$-го, а на его место записывали новый элемент, в этой — сдвигаем все элементы с k-го, а затем на его место записываем новый.

```
Program Example_43;
Const n=10; dd=51;
Type myarray= Array[1.. n+1] Of Integer;
Var A: myarray;
   x,k:Integer; {x - вставляемое число
   k - номер элемента, после которого вставляем}
Procedure Init2(Var m:myarray);
{процедура заполнения (инициализации) массива случайными числами}
...
Procedure Print1(n1: Integer; m: myarray);
{процедура вывода (распечатки) массива}
...
Procedure Insert2(k1, x1: Integer; Var m: myarray);
Var i: Integer;
Begin
   {сдвиг на одну позицию назад}
   For i:=n Downto k1 Do m[i+1]:=m[i];
```

```
   m[k1]:=x1; {вставка x1 на место k1-го}
End;
Begin
   Init2(A);
   Print1(n,A); {первый вывод начального массива}
   Writeln('Номер элемента, перед которым вставлять, ');
   Writeln('и вставляемое число');
   Readln(k,x); {ввод номера и вставляемого элемента}
   Insert2(k,x,A);
   Print1(n+1,A); {вывод массива после вставки в него}
   Readln;
End.
```

Пример 44. *Вставить число после всех элементов массива, кратных 3.*

Решение. Первое, на что необходимо обратить внимание — это описание массива: на сколько элементов может увеличиться массив? Максимальное число элементов, после которых будет вставлен новый элемент, совпадает с числом элементов массива, так как может случиться, что все элементы массива отвечают заданному свойству. Поэтому массив может увеличиться максимум в два раза .

Второе. Если мы будем просматривать элементы массива с начала и вставлять новый после элемента с заданным свойством, то номер последнего элемента каждый раз может меняться, кроме того, будет просматриваться и новый (вставленный) элемент, который необходимо будет пропускать («перепрыгивать»), поэтому решение будет не очень эффективным.

Лучше всего просматривать массив, начиная с конца, тогда вставляемый элемент мешать не будет. Кроме того, номер последнего элемента можно будет знать (если знать, сколько элементов вставлено на данный момент), при этом просмотр будет последовательным от *N*-го до 1-го.

```
Program Example_44;
Const n=10; dd=51;
Type myarray=Array[1.. 2*n] Of Integer;
Var A: myarray;
x, k, i:Integer;
{x - вставляемое число, k - число вставленных элементов}
Procedure Init2(Var m:myarray);
{процедура заполнения (инициализации) массива случайными числами}
...
Procedure Print1(n1: Integer; m: myarray);
{процедура вывода (распечатки) массива}
...
Procedure Insert3(k1, x1: Integer; Var m: myarray);
Var i: Integer;
Begin
   {сдвиг элементов на одну позицию назад,
   n+k - номер последнего элемента в данный момент}
   For i:=n+k Downto k1+1 Do m[i+1]:=m[i];
   m[k1+1]:=x1; {вставка элемента на место - после k1-го}
   Inc(k); {увеличение счетчика вставленных элементов}
End;
Begin
```

```
    Init2(A);
    Print1(n,A);
    Writeln('Введите вставляемое число');
    Readln(x);
    k:=0;
    For i:= n Downto 1 Do
    If A[i] Mod 3=0 Then Insert3(i,x,A);
    Print1(n+k,A); {вывод массива после вставки в него всех элементов}
    Readln;
End.
```

Задания для самостоятельной работы

1. Вставить элемент после первого отрицательного элемента.
2. Вставить элемент перед последним отрицательным элементом.
3. Вставить два элемента: первый — после максимального элемента, второй — перед максимальным элементом (удобнее всего применить именно такой порядок вставки).
4. Вставить по одному элементу перед всеми элементами, кратными заданному числу.
5. Вставить по одному элементу перед всеми отрицательными элементами.
6. Вставить два элемента: первый — после всех элементов, больших данного числа P, а второй — перед всеми элементами, большими данного числа P (P вводить с клавиатуры).
7. Вставить число A перед всеми элементами, большими A, а число B — после всех элементов, меньших его.

Упражнение № 18. Перестановки элементов массива

Пример. Поменять местами два элемента с номерами $k1$ и $k2$, где $k1$ и $k2$ вводятся с клавиатуры.

Решение. Опишем процедуру, которой будем передавать номера переставляемых элементов и массив.

```
Procedure Swap(k1, k2: Integer; Var m: myarray);
Var x: Integer;
Begin
   x:=m[k1]; m[k1]:=m[k2]; m[k2]:=x;
End;
```

Пример. Дан одномерный массив A, состоящий из $2n$ элементов. Поменять местами первую и вторую его половины.

Решение. Пусть массив A состоит из 10 элементов, то есть $n = 5$: 1, 12, 23, 3, 7, 13, 27, 6, 9, 11. Тогда, если мы поменяем местами первую и вторую его половины, то получим такой массив A: 13, 27, 6, 9, 11, 1, 12, 23, 3, 7. Заметим, что мы меняем местами элементы с номерами 1 и $n+1$, 2 и $n+2$ и так далее, последняя пара — n и $2n$. Поэтому можно вывести правило перестановки — элемент с номером i меняется местами с элементом с номером $n+i$. Поэтому, используя процедуру *Swap* из предыдущего примера, можно в основной программе сделать так:

```
For i:=1 To n Do Swap(i,i+n,A);
```

Задания для самостоятельной работы

1. Поменять местами:
   а) первый и максимальный элементы;
   б) второй и минимальный;
   в) первый и последний отрицательный.
2. Дан одномерный массив A, состоящий из $2n$ элементов. Поменять его половины следующим образом: первый элемент поменять с последним, второй с предпоследним и так далее.
3. Дан одномерный массив B, состоящий из $2n$ элементов. Переставить его элементы по следующему правилу:
   а) $b[n+1]$, $b[n+2]$, ..., $b[2n]$, $b[1]$, $b[2]$, ..., $b[n]$;
   б) $b[n+1]$, $b[n+2]$, ..., $b[2n]$, $b[n]$, $b[n-1]$, ..., $b[1]$;
   в) $b[1]$, $b[n+1]$, $b[2]$, $b[n+2]$, ..., $b[n]$, $b[2n]$;
   г) $b[2n]$, $b[2n-1]$, ..., $b[n+1]$, $b[1]$, $b[2]$, ..., $b[n]$.
4. Дан одномерный массив. Переставить в обратном порядке элементы массива, расположенные между минимальным и максимальным элементами.

Упражнение № 19. Двумерные массивы. Ввод и вывод двумерного массива. Простейшие операции.

Пример 45. *Составить программу ввода и вывода двумерного массива A размерностью 10×15.*

Решение. Формирование и вывод массива описаны в виде двух процедур, которые вызываются последовательно из основной программы. Надо заметить, что формирование двумерного массива можно осуществлять всеми тремя способами, описанными для одномерных массивов. Пусть в нашем примере элементы задаются генератором случайных чисел.

```
Program Example_45;
Const n=10; m=15;
Type dmyarray = Array[1..n,1..m] Of Integer;
Var A: dmyarray;
Procedure Init(Var x: dmyarray); {процедура формирования массива}
Var i, j: Integer;
Begin
   For i:=1 To n Do
   For j:= 1 To m Do x[i,j]:=-25+Random (51);
End;
Procedure Print(x: dmyarray); {процедура вывода массива на экран}
Var i, j: Integer;
Begin
   For i:=1 To n Do
   Begin {вывод i-й строки массива}
      For j:=1 To m Do Write(x[i,j]:5);
      Writeln; {переход на начало следующей строки}
   End;
End;
Begin {основная программа}
   Init(A); {вызов процедуры формирования массива}
```

```
   Writeln('Массив A:');
   Print(A); {вызов процедуры вывода}
   Readln;
End.
```

Пример. Сформировать одномерный массив, каждый элемент которого равен сумме отрицательных элементов соответствующей строки заданной целочисленной матрицы.

Решение. Опишем одномерный массив, размерность которого равна количеству строк в двумерном массиве.

```
Const n=10;m=15;
Type omyarray=Array [1..n] Of Integer;
   dmyarray=Array[1..n,1..m] Of Integer;
Var B: omyarray;
   A: dmyarray;
Procedure Sum(x: dmyarray; Var y: omyarray);
Var i, j: Integer;
Begin
   For i:=1 To n Do
   Begin
      y[i]:=0;
      {задание начальных значений элементов массива суммы}
      For j:=1 To m Do {накопление суммы отрицательных}
      If x[i,j]<0 Then y[i]:=y[i]+x[i,j];
   End;
End.
```

В основной программе обращаемся к процедуре *Sum*(*A*,*B*) и остается только вывести на экран одномерный массив *B*, в котором записаны суммы отрицательных элементов каждой строки.

Пример. Найти максимальный элемент массива и его индексы.

Решение. Так как элементы могут повторяться, то договоримся, что будем запоминать только индексы первого максимального элемента. Опишем процедуру, которой передается массив, и ее результатом являются значение максимального элемента и индексы первой встречи такого значения.

```
Procedure Maximum(x: dmyarray; Var max, maxi, maxj: Integer);
Var i, j: Integer;
Begin
   max:=x[1,1]; maxi:=1; maxj:=1; {начальные значения}
   For i:=1 To n Do
   For j:=1 To m Do If x[i,j]>max Then
   Begin {присвоение новых значений}
      max:=x[i,j];
      maxi:=i; maxj:=j;
   End;
End.
```

Пример 46. *Составить программу вычисления произведения двух квадратных целочисленных матриц A и B размером 5× 5 соответственно. Элементы результирующей, также целочисленной, матрицы C (размером 5× 5) определяются по формуле*

$$c[i, j]=\sum_{k=1}^{n} a[i, k]b[k, j],$$

где n — размерность матриц A и B.

Решение. Формирование матриц будем производить с помощью генератора случайных чисел, вычисление элементов результирующей матрицы C — с помощью вложенных циклов, где во внутреннем цикле (по параметру k) будет накапливаться сумма, определяющая элемент $c[i, j]$.

```
Program Example_46;
Const n=5;
Type dmyarray=Array[1..n,1..n] Of Integer;
Var A, B, C: dmyarray;
Procedure Init(Var x: dmyarray);
...
Procedure Print(x: dmyarray);
...
Procedure Mult(x,y: dmyarray; Var z: dmyarray);
Var k, i, j: Integer;
Begin
   For i:=1 To n Do
   For j:=1 To n Do
Begin
      z[i,j]:=0;
      For k:=1 To n Do z[i,j]:=z[i,j]+x[i,k]*y[k,j];
   End;
End;
Begin {основная программа}
   Writeln('массив A:'); Init(A); Print(A);
   Writeln('массив B:'); Init(B); Print(B);
   Mult(A,B,C);
   Writeln('массив C:');Print(c);
   Readln;
End.
```

Задания для самостоятельной работы

1. Найти сумму и число элементов каждого столбца с заданным условием (хранить эти значения в массивах):
   а) элементы, кратные $k1$ или $k2$;
   б) элементы, попадающие в промежуток от A до B;
   в) элементы, которые являются простыми числами;
   г) данные элементы положительны и лежат выше главной диагонали.
2. Найти сумму элементов в строках с $k1$-й по $k2$-ю.
3. Найти номера:
   а) всех максимальных элементов;
   б) первых отрицательных элементов каждой строки (столбца);
   в) последних отрицательных элементов каждой строки (столбца).
4. Найти число элементов в каждой строке, больших (меньших) среднего арифметического элементов данной строки.

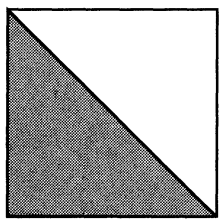
5. Найти произведение двух двумерных массивов A и B, если массив A имеет размерность $n \times m$, а B — $m \times n$. Определить размерность результирующего массива и правило нахождения элемента с номерами i и j.

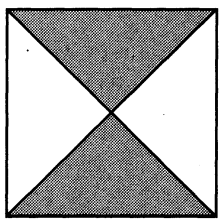
6. Даны два квадратных массива A и B. Вывести на экран тот из них, у которого след меньше (сумма элементов главной диагонали).

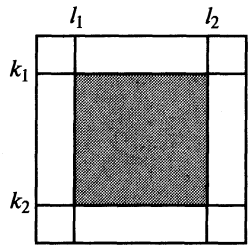
7. Изменить функцию в задании 6 так, чтобы просмотр элементов заканчивался в случае, когда найден нулевой элемент.

8. Изменить функцию в задании 7 так, чтобы просмотр пар прекращался тогда, когда найдена пара неравных элементов.

9. Определить:
   а) есть ли в данном массиве отрицательный элемент;
   б) есть ли два одинаковых элемента;
   в) есть ли данное число A среди элементов массива;
   г) есть ли в заштрихованной области массива элемент, равный A (массив имеет размерность $n \times n$):







10. Определить:
   а) является ли массив логическим квадратом, то есть суммы по всем горизонталям, вертикалям и двум диагоналям должны быть равны;
   б) добавить к предыдущему условию требование, чтобы сумма была равна данному числу A.

11. Определить, есть ли в данном массиве строка (столбец):
   а) состоящая только из положительных элементов;
   б) состоящая только из положительных или нулевых элементов;
   в) состоящая только из элементов, больших числа A;
   г) состоящая только из элементов, принадлежащих промежутку от A до B.

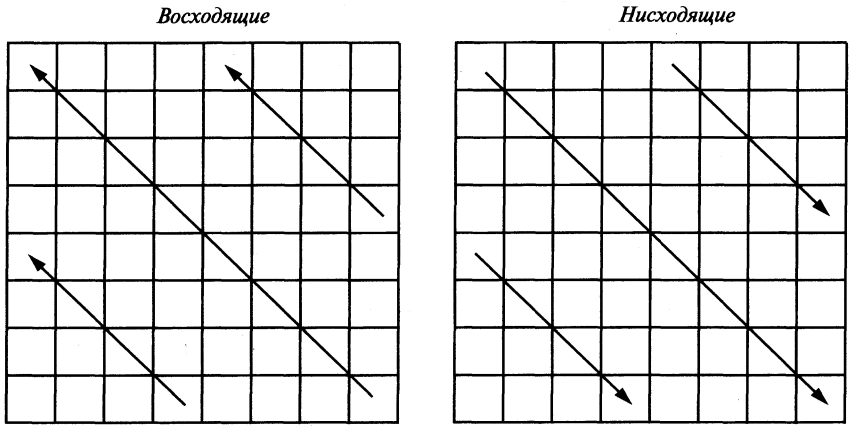
Упражнение № 20. Двумерный массив. Работа с элементами

Пример 47. *Составить программу, запрашивающую координаты ферзя на шахматной доске и показывающую поля доски, находящиеся «под боем».*

Решение.

1. Заметим, что шахматную доску удобно представить в виде двумерного массива размером 8×8. Координаты ферзя можно вводить двумя числами (номер строки и номер столбца), но в шахматах принято вводить букву и число. Буква отвечает за номер строки, а число — за номер столбца. Поэтому не будем отступать от традиций и введем координаты именно таким образом. В программе сделаем проверку правильности ввода и, если все правильно, переведем букву в соответствующее ей число («*a*» — 1, «*b*» — 2, «*c*» — 3, «*d*» — 4, «*e*» — 5, «*f*» — 6, «*g*» — 7, «*h*» — 8), тогда будет удобнее работать.

2. Для решения надо еще знать следующие свойства шахматной доски. Все диагонали делятся на восходящие и нисходящие:



Свойство каждой диагонали:

• для любой восходящей диагонали сумма номера строки и номера столбца постоянна и для разных диагоналей — различна, то есть $i + j = \text{const1}$;

• для нисходящих — разность номера строки и номера столбца тоже постоянна и для разных диагоналей — различна, то есть $i - j = \text{const2}$.

Это свойство необходимо учитывать, чтобы определить номера диагоналей, на которых поставлен ферзь.

```
Program Example_47;
Const n=8;
Type dmyarray=Array[1..n,1..n] Of Integer;
Var A: dmyarray;
   c: Char;
   str, stl: Integer; {str - номер строки, stl - номер столбца}
Function Place(ch: Char): Integer;
Var k: Integer;
Begin
   Case ch Of
   'a': k:=1; 'b': k:=2; 'c': k:=3; 'd': k:=4;
   'e': k:=5; 'f': k:=6; 'g': k:=7; 'h': k:=8;
   End;
   Place:=k;
End;
Procedure Init(k, l: Integer; Var x: dmyarray); {k - номер строки,
l - номер столбца, где поставлен ферзь}
Var i, j: Integer;
Begin
   For i:=1 To n Do
   For j:=1 To n Do
   {если мы стоим в той же горизонтали или вертикали, или восходя-
   щей диагонали, или нисходящей диагонали, то данная клетка «под
   боем», поэтому помечаем ее 1, в других случаях - 0}
   If (i=k) Or (j=l) Or (i+j=k+l) Or (i-j=k-l)
```

```
    Then x[i,j]:=1 Else x[i,j]:=0;
    x[k,l]:=2; {здесь стоит ферзь}
End;
Procedure Print (x: dmyarray);
Var i, j: Integer;
Begin
    For i:=1 To n Do
    Begin
        For j:=1 To n Do
        Case x[i,j] Of
        0: Write('':3);
        1: Write('*':3);
        2: Write('F':3);
        End;
        Writeln;
    End;
End;
Begin
    Writeln('Введи координаты ферзя');
    Readln(c, stl);
    If (c<'a') Or (c>'h') Or (stl<1) Or (stl>n)
    Then Writeln ('некорректный ввод')
    Else
    Begin
        str:= Place(c);
        Init(str,stl,A);
        Print(A);
    End;
    Readln;
End.
```

Задания для самостоятельной работы

1. В каждой строке сменить знак максимального по модулю элемента на противоположный.
2. Последний отрицательный элемент каждого столбца заменить нулем.
3. Положительные элементы умножить на первый элемент соответствующей строки, а отрицательные — на последний, то есть положительные элементы первой строки умножаем на первый элемент первой строки, а отрицательные — на последний элемент тоже первой строки, то же самое сделаем и с остальными строками.
4. Заменить все элементы строки с номером k и столбца с номером l на противоположные по знаку (элемент, стоящий на пересечении, не изменять).
5. К элементам столбца $k1$ прибавить элементы столбца $k2$.
6. Написать программу, запрашивающую координаты коня и определяющую поля, находящиеся «под боем».
7. Ввести координаты ферзя и коня и определить:
   а) если конь ходит первым, то бьет ли он ферзя;
   б) бьет ли ферзь коня, если первый ход ферзя.
8. Составить программу заполнения и вывода на экран таблицы Пифагора (умножения).

9. Даны два двумерных массива одинаковой размерности. Создать третий массив той же размерности, каждый элемент которого равен сумме соответствующих элементов первых двух.

10. Даны два двумерных массива *A* и *B* одинаковой размерности. Создать массив *C*, где каждый элемент равен 1 (*True*), если соответствующие элементы *A* и B имеют одинаковый знак, иначе элемент равен 0 (*False*).

11. Составить программу вывода на экран арифметического квадрата, в нем первый столбец и первая строка заполнены 1 (единицами), а каждый из остальных элементов равен сумме своих соседей сверху и слева.

Упражнение № 21. Вставка и удаление

Пример. Вставить строку из нулей после строки с номером *k*.

Решение. Для решения этой задачи необходимо:

1) первые *k* строк оставить без изменения;

2) все строки после *k*-й сдвинуть на одну назад; лучше начать с последней строки и идти до $(k+1)$-й;

3) элементам строки $k+1$ присвоить заданное значение.

Кроме того, необходимо изменить размерность массива. Так как мы вставляем строку, то число строк будет на одну больше:

```
Const n=5; m=7;
Type dmyarray=Array[1..n+1,1..m] Of Integer;
Var A: dmyarray;
```

Теперь опишем процедуру вставки:

```
Procedure Insert(k1: Integer; Var x: dmyarray);
Var i, j: Integer;
Begin
   For i:=n Downto k1+1 Do
   For j:=1 To m Do x[i+1,j]:=x[i,j];
   {элементу столбца j присваиваем элемент этого же столбца,
   но из предыдущей строки}
   For j:=1 To m Do x[k1+1,j]:=0;
End;
```

Так как число строк меняется, то процедуру `Print` надо изменить, она должна выводить данное число строк, начиная с первой:

```
Procedure Print1(n1: Integer; x: dmyarray);
Var i, j: Integer;
Begin
   For i:=1 To n1 Do
   Begin
      For j:=1 To m Do Write(x[i,j]:4);
      Writeln;
   End;
End;
```

Часть основной программы будет такой:

```
Begin
   Init(A);
```

```
    Print(n,A);
    Writeln('Введите номер строки, после которой вставляем');
    Readln(k);
    Insert(k,A);
    Print(n+1,A);
    Readln;
End.
```

Пример. Удалить строку с номером k.

Решение. Для того чтобы удалить строку с номером k, необходимо:

- сдвинуть все строки, начиная с данной, на одну вверх;
- последнюю строку «обнулить», то есть всем элементам последней строки присвоить значение 0.

Описание массивов оставим прежним (для размерности $n \times m$). Также в программе будем использовать процедуру вывода `Print1`. Будем выводить на экран сначала все строки, затем, после удаления, на одну меньше. Теперь опишем процедуру удаления строки с данным номером:

```
Procedure Delete(k1: Integer;Var x: dmyarray);
Var i, j: Integer;
Begin
   For i:=k1 To n-1 Do
   For j:=1 To m Do x[i,j]:=x[i+1,j];
   For j:=1 To m Do x[n,j]:=0;
End;
```

Задания для самостоятельной работы

1. Вставить первую строку после строки, в которой находится первый встреченный максимальный элемент.
2. Вставить второй столбец после первого столбца, в котором все элементы положительны. Если такого столбца нет, то сообщить об этом.
3. Вставить нулевую строку и нулевой столбец перед строкой и столбцом, где находится первый минимальный элемент.
4. Вставить после всех строк, в которых есть заданное число A, последнюю строку.
5. Вставить перед всеми столбцами, в которых нет отрицательных элементов, второй столбец.
6. Вставить перед всеми строками, в которых есть 0, первую строку, а после всех столбцов, в которых есть отрицательные элементы — первый столбец.
7. Удалить столбец, в котором находится минимальный элемент. Если такой элемент встречается несколько раз, то удалить все столбцы.
8. Удалить строку с номером k и столбец с номером l.
9. Удалить все столбцы, в которых нет нулевого элемента.
10. Удалить все строки и столбцы, на пересечении которых стоят отрицательные элементы.

Упражнение № 22. Перестановка элементов массива

Пример. Поменять местами два элемента массива A с заданными координатами (номерами строки и столбца).

Решение. Можно эту задачу решить несколькими способами.

Первый способ — по аналогии с перестановкой в одномерном массиве, когда в процедуру передаются индексы элементов и массив, в котором надо их поменять. Тогда процедура может быть такой:

```
Procedure Swap1(k1, l1, k2, l2: Integer; Var x: dmyarray);
Var c: Integer;
Begin
c:=x[k1,l1]; x[k1,l1]:=x[k2,l2];x[k2,l2]:=x;
End;
```

Второй способ. Вспомним процедуру *Swap*, которая меняет местами значения двух целых переменных.

```
Procedure Swap(Var x, y: Integer);
Var z: Integer;
Begin
z:=x; x:=y; y:=z;
End;
```

Пример. Поменять местами столбцы с номерами *l*1 и *l*2.

```
Procedure Swap2(l1, l2: Integer; Var x: dmyarray);
Var i: Integer;
Begin
   If ((l1<1) Or (l1>m)) Or ((l2<1) Or (l2>m))
   Then Writeln('Ввод неправильный')
   Else For i:=1 To m Do Swap(x[i,l1], x[i,l2]);
End;
```

Задания для самостоятельной работы

1. Поменять местами первый максимальный и последний минимальный элементы.
2. В каждой строке поменять местами первый элемент и максимальный по модулю.
3. В каждой строке переставить первый отрицательный и последний положительный, если таких нет, то сообщить об этом.
4. Переставить вторую и предпоследнюю строки.
5. Поменять местами первую строку и строчку, в которой находится первый нулевой элемент.
6. В двумерном массиве переставить строки следующим образом: первую с последней, вторую — с предпоследней и так далее. Если строк нечетное число, то средняя останется неизменной, иначе средние строки тоже меняем местами.
7. Дан двумерный массив *A*. Расставить его столбцы в следующем порядке:
   а) последний, предпоследний, ..., второй, первый;
   б) первый, последний, второй, предпоследний, третий, ...
8. Дан двумерный массив. Начиная с первой строки, сдвинуть все строки на две вниз, а последние две перенести на место первых двух строк.
9. Начиная с *k*-го столбца, сдвинуть их назад, а последние *k* поставить на место первых.
10. Начиная с *k*-го столбца, сдвинуть их вперед, а первые *k* поставить на место последних.

Контрольные работы

Контрольная работа № 3. Одномерные массивы. Работа с элементами

Время выполнения 4 — 6 часов.

Вариант 1

1. Правильно ли описан массив *A*? Если нет, то что надо изменить?

```
Type myarray=Array[-10..n] Of Integer;
Var A: myarray;
```

2. Что получится в результате выполнения программы?

```
Program Variant1;
Const n=7;
Type myarray=Array[1..n] Of Integer;
Var C: myarray;
   i: Byte; p: Integer;
Begin
   p:=0;
   For i:=1 To n Do
   Begin
      C[i]:=-50+Random(151);
      If C[i]>50 Then p:=p+C[i];
   End;
   Writeln(p);
   Readln;
End.
```

3. Дан массив целых чисел, состоящий из 20 элементов. Заполнить его с клавиатуры. Найти:
   - сумму элементов, имеющих нечетное значение;
   - и вывести индексы тех элементов, значения которых больше заданного числа *A*.

4. Определить, есть ли в данном массиве положительные элементы, кратные *k* (*k* вводить с клавиатуры).

Вариант 2

1. Правильно ли описан массив *C*? Если нет, то что надо изменить?

```
Const n1=25;
Type m=Array[15..-n1] Of Integer;
Var C: m;
```

2. Что получится в результате выполнения программы?

```
Program Variant2;
Const n=10;
Type myarray=Array[1..n] Of Integer;
Var D: myarray;
   i: Byte; p: Integer;
Begin
   p:=0;
```

```
    For i:=1 To n Do
    Begin
        D[i]:=-25+Random(51);
        If D[i]<0 Then p:=p+D[i];
    End;
    Writeln(p);
    Readln;
End.
```

3. Дан массив целых чисел, состоящий из 25 элементов. Заполнить его с клавиатуры. Найти:

- сумму элементов, имеющих нечетные индексы;
- число элементов массива, значения которых больше заданного числа A и кратны 5.

4. Найти номер первого отрицательного элемента, делящегося на 5 с остатком 2.

Вариант 3

1. Правильно ли описан массив *A*? Если нет, то что надо изменить?

```
Type odmyarray=Array[1..n+20] Of Integer;
Var A: odmyarray;
```

2. Что получится в результате выполнения программы?

```
Program Variant3;
Const n=17;
Type myarray=Array[1..n] Of Integer;
Var B: myarray;
   i: Byte; p: Integer;
Begin
   p:=0;
   For i:=1 To n Do
   Begin
      B[i]:=-35+Random(121);
      If C[i] Mod 10=0 Then p:=p+1;
   End;
   Writeln(p);
   Readln;
End.
```

3. Дан массив целых чисел, состоящий из 15 элементов. Заполнить его с клавиатуры. Найти:

- сумму положительных элементов, значения которых меньше 10;
- и вывести индексы тех элементов, значения которых кратны 3 и 5.

4. Определить, есть ли пара соседних элементов с суммой, равной заданному числу.

Вариант 4

1. Правильно ли описан массив *D*? Если нет, то что надо изменить?

```
Type odm=Array[-n..n] Of Integer;
Var D: odm;
```

2. Что получится в результате выполнения программы?

```
Program Variant4;
Const n=25;
Type myarray=Array[1..n] Of Integer;
Var A: myarray;
   i: Byte; p: Integer;
Begin
   p:=0;
   For i:=1 To n Do
   Begin
      A[i]:=-50+Random(151);
      If A[i]<=10 Then p:=p+A[i];
   End;
   Writeln(p);
   Readln;
End.
```

3. Дан массив целых чисел, состоящий из 10 элементов. Заполнить его с клавиатуры. Найти:
   - удвоенную сумму положительных элементов;
   - и вывести индексы тех элементов, значения которых больше значения предыдущего элемента (начиная со второго).
4. Определить, есть ли две пары соседних элементов с одинаковыми знаками.

Вариант 5

1. Правильно ли описан массив *A*? Если нет, то что надо дописать?

```
Type myarray=Array[0..-n] Of Integer;
Var A: myarray;
```

2. Что получится в результате выполнения программы?

```
Program Variant5;
Const n=12;
Type myarray=Array[1..n] Of Integer;
Var C: myarray;
   i: Byte; p: Integer;
Begin
   For i:=1 To n Do
   Begin
      C[i]:=-25+Random(71);
      If C[i] Mod 3=0 Then p:=p+1];
   End;
   Writeln(p);
   Readln;
End.
```

3. Дан массив целых чисел, состоящий из 30 элементов. Заполнить его с клавиатуры. Найти:
   - сумму отрицательных элементов;
   - число тех элементов, значения которых положительны и не превосходят заданного числа *A*.
4. Найти номер последней пары соседних элементов с разными знаками.

Контрольная работа № 4. Одномерные массивы. Работа с элементами

Время выполнения 4 — 6 часов.

Вариант 1

1. Заменить максимальный по модулю отрицательный элемент нулем.
2. Заменить первые k элементов на противоположные по знаку.
3. Из элементов массива C сформировать массив A той же размерности по правилу: если номер четный, то $A_i = C_i^2$, если нечетный, то $A_i = 2C_i$.

Вариант 2

1. Заменить минимальный по модулю положительный элемент нулем.
2. Заменить элементы с $k1$-го по $k2$-й на обратные.
3. Из элементов массива A сформировать массив D той же размерности по правилу: первые 10 элементов — $D_i = A_i + i$, остальные — $D_i = A_i - i$.

Вариант 3

1. Заменить первый отрицательный элемент нулем.
2. Умножить все элементы, кратные 3, на третий элемент массива.
3. Из элементов массива P сформировать массив M той же размерности по правилу: если номер четный, то $P_i = iM_i$, если нечетный, то $P_i = -M_i$.

Вариант 4

1. Заменить максимальный элемент на противоположный по знаку.
2. Заменить нулями элементы между минимальным и максимальным, кроме них самих.
3. Из элементов массива C сформировать массив A той же размерности по правилу: элементы с 3-го по 12-й $A_i = -C_i^2$, все остальные $A_i = C_i - 1$.

Вариант 5

1. Заменить первый элемент, кратный 5, нулем.
2. Заменить элементы с нечетными номерами на квадрат их номера.
3. Из элементов массива D сформировать массив A той же размерности по правилу: если номер четный, то $A_i = D_i^2$, если нечетный, то $A_i = D_i/i$.

Вариант 6

1. Заменить последний положительный элемент на второй элемент массива.
2. Разделить все элементы с четными номерами на первый элемент.
3. Из элементов массива C сформировать массив A той же размерности по правилу: если номер четный, то $A_i = C_i^2$, если нечетный, то $A_i = 2C_i$.

Контрольная работа № 5. Одномерные массивы. Удаление, вставка и перестановка элементов

Вариант 1

Дан массив целых чисел ($n = 15$), заполненный случайным образом числами из промежутка [−20, 50].

1. Удалить из него все элементы, в которых есть цифра 5.

2. Вставить число k после всех элементов, кратных своему номеру (k вводить с клавиатуры).

3. Поменять местами первый положительный и последний отрицательный элементы.

Вариант 2

Дан массив целых чисел ($n = 10$), заполненный случайным образом числами из промежутка $[-40, 30]$.

1. Удалить из него все элементы, которые состоят из одинаковых цифр (включая однозначные числа).

2. Вставить число k перед всеми элементами, в которых есть цифра 1 (k вводить с клавиатуры).

3. Переставить первые три и последние три элемента местами, сохраняя их следование.

Вариант 3

Дан массив целых чисел ($n = 12$), заполненный случайным образом числами из промежутка $[-10, 60]$.

1. Удалить из него все элементы, в которых последняя цифра четная, а само число делится на нее.

2. Вставить число k перед и после всех элементов, заканчивающихся на данную цифру (k вводить с клавиатуры).

3. Переставить элементы следующим образом: $a[1]$, $a[12]$, $a[2]$, $a[11]$, ..., $a[5]$, $a[8]$, $a[6]$, $a[7]$.

Вариант 4

Дан массив целых чисел ($n = 25$), заполненный случайным образом числами из промежутка $[-35, 75]$.

1. Удалить из него все элементы, первая цифра которых четная.

2. Вставить число $k1$ после всех элементов, больших заданного числа, а число $k2$ — перед всеми элементами, кратными 3 ($k1$ и $k2$ вводить с клавиатуры).

3. Перенести первые k элементов в конец массива, то есть: $a[k+1]$, $a[k+2]$, ..., $a[n]$, $a[1]$, $a[2]$, ..., $a[k]$.

Вариант 5

Дан массив целых чисел ($n = 20$), заполненный случайным образом числами из промежутка $[-45, 95]$.

1. Удалить из него все элементы, кратные 7 и принадлежащие промежутку $[a, b]$ (a и b вводить с клавиатуры).

2. Вставить число k между всеми соседними элементами, которые образуют пару элементов с одинаковыми знаками (k вводить с клавиатуры).

3. Переставить в обратном порядке часть массива между элементами с номерами $k1$ и $k2$, включая их. Сделать проверку корректности ввода $k1$ и $k2$; если ввод неправильный, то ничего не делать.

Контрольная работа № 6. Двумерные массивы. Работа с элементами

Время выполнения 6 — 8 часов.

Вариант 1

1. Дан двумерный массив размерностью 5×6, заполненный целыми числами с клавиатуры. Сформировать одномерный массив, каждый элемент которого равен произведению четных положительных элементов соответствующего столбца.

2. Дан двумерный массив размером $n \times m$, заполненный случайным образом. Определить, есть ли в данном массиве строка, в которой ровно два отрицательных элемента.

3. Заполнить массив размерностью 7×7 по правилу

1	0	0	0	0	0	1
0	1	0	0	0	1	0
0	0	1	0	1	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
1	0	0	0	0	0	1

Вариант 2

1. Дан двумерный массив размерностью 4×6, заполненный целыми числами с клавиатуры. Сформировать одномерный массив, каждый элемент которого равен количеству элементов соответствующей строки, больших данного числа.

2. Дан двумерный массив размером $n \times m$, заполненный случайными числами. Определить, есть ли в данном массиве столбец, в котором имеются одинаковые элементы.

3. Заполнить массив размерностью 7×7 по правилу

1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1

Вариант 3

1. Дан двумерный массив размерностью 5×6, заполненный целыми числами с клавиатуры. Сформировать одномерный массив, каждый элемент которого равен наибольшему по модулю элементу соответствующего столбца.

2. Дан двумерный массив размером $n \times m$, заполненный случайными числами. Определить, есть ли в данном массиве строка, в которой имеется два максимальных элемента всего массива.

3. Заполнить массив размерностью 6×6 по правилу

1	2	3	4	5	6
2	3	4	5	6	1
3	4	5	6	1	2
4	5	6	1	2	3
5	6	1	2	3	4
6	1	2	3	4	5

Вариант 4

1. Дан двумерный массив размерностью 4×5, заполненный целыми числами с клавиатуры. Сформировать одномерный массив, каждый элемент которого равен количеству отрицательных элементов, кратных 3 или 5, соответствующей строки.
2. Дан двумерный массив размером $n \times m$, заполненный случайными числами. Определить, есть ли в данном массиве столбец, в котором равное число положительных и отрицательных элементов.
3. Заполнить массив размерностью 6×6 по правилу

1	1	1	1	1	1
1	2	3	4	5	6
1	3	6	10	15	21
1	4	10	20	35	56
1	5	15	35	70	126
1	6	21	56	126	252

Вариант 5

1. Дан двумерный массив размерностью 6×5, заполненный целыми числами с клавиатуры. Сформировать одномерный массив, каждый элемент которого равен первому четному элементу соответствующего столбца, если такого нет, то равен нулю.
2. Дан двумерный массив размером $n \times m$, заполненный случайными числами. Определить, есть ли в данном массиве строка, содержащая больше положительных элементов, чем отрицательных.
3. Заполнить массив размерностью 7×7 по правилу

1	0	0	1	0	0	1
0	1	0	1	0	1	0
0	0	1	1	1	0	0
1	1	1	1	1	1	1
0	0	1	1	1	0	0
0	1	0	1	0	1	0
1	0	0	1	0	0	1

Контрольная работа № 7. Двумерные массивы

Время выполнения 4 — 6 часов.

Вариант 1

Дан двумерный массив размером 5×6, заполненный случайным образом.
1. Заменить максимальный элемент каждой строки на противоположный.
2. Вставить после столбцов с максимальными элементами столбец из нулей.
3. Удалить среднюю строку.
4. Поменять местами средние столбцы.

Вариант 2

Дан двумерный массив размером 8×7, заполненный случайным образом.
1. Заменить все элементы первых трех столбцов на их квадраты.
2. Вставить между средними строками первую строку.

3. Удалить все столбцы, в которых первый элемент больше последнего.
4. Поменять местами средние строки с первой и последней.

Вариант 3

Дан двумерный массив размером 5×8, заполненный случайным образом.
1. Заменить все симметричные элементы на нули.
2. Вставить перед всеми строками, первый элемент которых делится на 3, строку из нулей.
3. Удалить столбец, в котором находится первый четный отрицательный элемент.
4. Поменять местами средние столбцы со вторым и предпоследним.

Вариант 4

Дан двумерный массив размером 6×7, заполненный случайным образом.
1. Заменить максимальный элемент каждой строки на противоположный.
2. Вставить после столбцов с максимальными элементами столбец из нулей.
3. Удалить все столбцы, в которых первый элемент больше заданного числа A.
4. Поменять местами средние строки.

Строковый тип. Множества и записи

Упражнение № 23. Строковый тип данных

Пример. Определить, сколько раз в данной строке встречается символ «*a*».
Решение. Опишем функцию, которой будем передавать строку. Результат выполнения — целое число.

```
Function Q_Ch(st: String): Byte;
Var i, k: Byte;
Begin
   k:=0; {просматриваем все символы строки, их число равно длине
   строки, если очередной символ равен 'a', увеличиваем счетчик}
   For i:=1 To Length(st) Do If st[i]='a' Then Inc(k);
   Q_Ch:=k;
End;
```

Пример. Если длина строки нечетное число, то удалить среднюю букву.
Решение. Пусть *k* — это длина строки, если оно нечетное, то надо удалить средний символ, а его номер равен `k Div 2+1`.

```
Procedure Del(Var st: String);
Var k: Byte;
Begin
   k:=Length(st);
   If k Mod 2=1 Then Delete(st, k Div 2+1,1);
End;
```

Пример. Заменить все вхождения подстроки `'del'` на `'Insert'`.
Решение. Пока такая подстрока встречается, необходимо находить номер первого символа первой встречи, удалять в этом месте `'del'` и сюда вставлять `'Insert'`.

```
Procedure Ins(Var st: String);
Var k: Byte;
```

```
Begin
   While Pos('del',st)<>0 Do
   Begin
      k:=Pos('del',st);
      Delete(st,k,Length('del'));
      Insert('Insert',st,k);
   End;
End;
```

Пример 48. *Дана строка, состоящая из нескольких слов, между словами стоит один пробел, в конце предложения — точка. Подсчитать число слов и вывести на экран только те из них, которые начинаются с буквы «а» (слов не больше 30).*

Решение. Разобьем предложение на отдельные слова и каждое будем хранить как элемент массива.

```
Program Example_48;
Const n=30;
Type Myarray_Str=Array[1..n] Of String;
Var A: Myarray_Str;
str: String[255];
k: Byte;
Procedure Init(Var b: Myarray_Str);
Var i: Integer;
Begin
   k:=1; {пока не встретится пробел, формируем очередное слово k,
   прибавляя по одной букве}
   For i:=1 To Length(str)-1 Do
   If str[i]<>'' Then b[k]:=b[k]+str[i]
   Else
   {если это не последний символ, то увеличиваем счетчик слов
   и начинаем формировать соответствующий элемент массива}
   If i<>Length(str)-1 Then
   Begin Inc(k); b[k]:='' End;
End;
Begin
   Writeln('Введите предложение');
   Readln(str);
   Init(A);
   Writeln('Всего слов: ',k); {просматриваем все слова,
   если первый символ очередного слова есть 'a',
   то выводим его}
   For i:=1 To k Do If A[i][1]='a' Then Write(A[i],'');
   Readln;
End.
```

Задания для самостоятельной работы

1. Подсчитать, сколько раз в данной строке встречается некоторая буква, вводимая с клавиатуры.

2. Дан текстовый файл, в котором записано одно из стихотворений А. С. Пушкина. Сколько раз в каждой строке встречаются гласные буквы?

3. Из строки удалить среднюю букву, если длина строки нечетная, если четная — удалить две средние буквы. Заменить все вхождения в текст некоторой буквы на другую букву (их значения вводить с клавиатуры).

4. Заменить все вхождения подстроки *Str*1 на подстроку *Str*2 (подстроки вводятся с клавиатуры).

5. Даны две строки. Если они начинаются с одинаковых символов, то напечатать «ДА», в противном случае — «НЕТ».

6. Дана последовательность слов. Напечатать все слова, отличные от слова «hello».

7. Дана последовательность слов. Напечатать все слова в алфавитном порядке.

8. Дана последовательность слов. Напечатать все слова последовательности, которые встречаются в ней по одному разу.

9. Дано предложение. Напечатать все различные слова.

10. Дана последовательность слов. Напечатать все слова, предварительно преобразовав каждое из них по следующему правилу:

а) удалить из слова все предыдущие вхождения последней буквы;

б) оставить в слове только первые вхождения каждой буквы.

11. Дана последовательность слов. Напечатать те слова последовательности, которые отличны от последнего слова и удовлетворяют следующему свойству:

а) в слове нет повторяющихся букв;

б) буквы слова упорядочены по алфавиту;

в) слово совпадает с начальным отрезком латинского алфавита (*a*, *ab*, *abc*, *abcd*, ...);

г) слово симметрично.

12. Составить программу вывода самой большой цифры в записи заданного числа.

13. Найти сумму всех чисел строки.

Упражнение № 24. Множественный тип данных

Пример 49. *Составить программу выделения из множества целых чисел от 1 до 30 следующих множеств:*

- множества чисел, кратных 2;
- множества чисел, кратных 3;
- множества чисел, кратных 6;
- множества чисел, кратных 2 или 3.

```
Program Example_49;
Const n = 30;
Type mn=Set Of 1..n;
Var n2, n3, n6, n23: mn;
{n2 - множество чисел, кратных 2, n3 - кратных 3, n6 - кратных 6,
   n23 - кратных 2 или 3}
k: Integer;

Procedure Print(m: mn);
Var i: Integer;
Begin
   For i:=1 To n Do If i In m Then Write(i:3);
   Writeln;
End;
Begin
   n2=[ ]; n3=[ ]; {начальное значение множеств}
```

```
For k:=1 To n Do {формирование n2 и n3}
Begin {если число делится на 2, то заносим его в n2}
   If k Mod 2=0 Then n2:=n2+[k];
   If k Mod 3=0 Then n3:=n3+[k]; {если число делится на 3,
   то добавляем его в n3}
End;
{числа, кратные 6, - это те, которые кратны и 2, и 3, поэтому
это - пересечение двух первых множеств, а числа, кратные 2 или
3, - это объединение этих же множеств}
n6:=n2*n3; n23:=n2+n3;
Writeln('числа, кратные 2'); {вывод множеств}
Print(n2);
Writeln('числа, кратные 3');
Print(n3);
Writeln('числа, кратные 6');
Print(n6);
Writeln('числа, кратные 2 или 3');
Print(n23);
Readln;
End.
```

Пример 50. *«Мешанина». Если взять то общее, что есть у боба с ложкой, добавить кота и поместить в тепло, то получится муравей. Так ли это? Состоит ли муравей из кота?*

```
Program Example_50;
Var y1, y2, y3, y4, x: Set Of Char;
s: Char;
Begin
   y1:=['б', 'о', 'б']; y2:=['л', 'о', 'ж', 'к', 'а'];
   y3:=['к', 'о', 'т']; y4:=['т', 'е', 'п', 'л', 'о'];
   x:=(y1*y2)+y3-y4;
   Writeln('множество x'); {вывод множества x}
   For s:='a' To 'я' Do If s In x Then Write(s); Writeln;
   If y3<=x Then Write('муравей состоит из кота')
{проверка: состоит ли муравей из кота}
   Else Write('муравей не состоит из кота');
End.
```

Пример 51. *Дано натуральное число n. Составить программу, печатающую все цифры, не входящие в десятичную запись данного натурального числа в порядке возрастания.*

Программа для решения этой задачи такова:

```
Program Example_51;
Type mn = Set Of 0..9;
Var s: mn;
n: Longint;
l,k: Integer;
Begin
   Writeln ('введите число n');
   Readln(n);
```

```
    s:=[ ];
    While n<>0 Do
    {формирование множества цифр десятичной записи натурального числа}
    Begin
        k:=n Mod 10;
        n:=n Div 10;
        If Not (k In s) Then s:=s+[k];
    End;
    For k:=0 To 9 Do If Not (k In s) Then Write(k:2);
    {вывод цифр в порядке возрастания}
    Writeln;
    Readln;
End.
```

Пример 52. *«Решето Эратосфена». Составить программу поиска простых чисел в числовом промежутке [1...n]. Число n вводится с клавиатуры.*

Решение. Простым числом называется число, которое не имеет других делителей, кроме единицы и самого этого числа. Для решения этой задачи воспользуемся методом «решета Эратосфена», идея которого заключается в следующем: сформируем множество *M*, в которое поместим все числа заданного промежутка. Затем последовательно будем удалять из него элементы, кратные 2, 3, 4 и так далее, до [*n*/2] (целая часть числа), кроме самих этих чисел. После такого «просеивания» в множестве *M* останутся только простые числа.

```
Program Example_52;
Var m: Set Of Byte;
i,k,n: Integer;
Begin
   Writeln(' введите размер промежутка (до 255) ');
   Readln(n);
   m:=[2..n]; {начальное значение}
   For k:=2 To n Div 2 Do {перебираем все делители }
   For i:=2 To n Do
   If (i Mod k=0) And (i<>k) Then m:=m-[i];
   {если число кратно делителю и отлично от него, то удаляем его}
   For i:=1 To n Do If i In m Then Write(i:3);
   {распечатаем оставшиеся элементы }
   Readln;
End.
```

Пример 53. *Ребус.*

Решение. Каждая буква — это цифра, разным буквам соответствуют разные цифры. Необходимо заменить буквы цифрами так, чтобы получилось верное равенство.

$$\begin{array}{r} \text{МУХА} \\ + \quad \text{МУХА} \\ \hline \text{СЛОН} \end{array}$$

Для решения этой задачи используется метод перебора с возвратом. Используем множество *S*1 для хранения цифр слова МУХА, причем будем вносить в него цифры последовательно, учитывая уже внесенные цифры. Начальное значе-

ние *S*1 — пусто. После выбора всех цифр первого слова создаем его числовой эквивалент и числовой образ слова СЛОН. Выделяем цифры СЛОНа (множество *S*2) и если слова состоят из разных цифр (то есть пересечение *S*1 и *S*2 пустое) и все цифры СЛОНа разные (то есть пересечение множеств цифр тоже пустое), то выводим решение на экран. А сейчас идет возврат — удаляем из множества *S*1 последнюю внесенную цифру и пытаемся выбрать еще одно ее значение. Таким образом, мы перебираем все возможные варианты и выводим на экран только те, которые удовлетворяют равенству.

Заметим, что значение буквы «М» в слове МУХА может иметь значения от 1 до 4, а буква «А» в этом же слове не может быть равна 0.

```
Program Example_53;
Type mn = Set Of 0..9;
Var m,y,x,a: 0..9; {цифры числа МУХА}
n1,n2: Integer; {числа МУХА и СЛОН}
a1,a2,a3,a4: 0..9; {цифры числа СЛОН}
s1, s2:mn; {для хранения цифр каждого из чисел}
Procedure Print(x,y:Integer); {вывод решения в виде ребуса}
Begin
   Writeln(x:5);
   Writeln('+');
   Writeln(x:5);
   Writeln(' ____')
   Writeln(y:5);
End;
Begin
   s1:=[ ];s2:=[ ];
   For m:=1 To 4 Do
   Begin
      s1:=s1 + [m]; {заносим первую использованную цифру}
      For y:=0 To 9 Do
      If Not (y In s1) Then {если эта цифра не была еще взята, то
      добавляем ее во множество цифр числа МУХА
      и выбираем цифру для следующей буквы}
      Begin
         s1:=s1+[y];
         For x:=0 To 9 Do
         If Not (x In s1) Then
         Begin
            s1:= s1 + [x];
            For a:=1 To 9 Do
            If Not (a In s1) Then
            Begin
               s1:=s1+[a];
               n1:=1000*m+100*y+10*x+a; {число для слова МУХА}
               n2:=2*n1; {число для слова СЛОН}
               a1:=n2 Div 1000; {выделяем цифры СЛОНа}
               a2:=n2 Div 100 Mod 10;
               a3:=n2 Div 10 Mod 10;
               a4:=n2 Mod 10;
```

```
                    s2:=[a1,a2,a3,a4]; {множество цифр СЛОНа}
                    {если слова состоят из разных цифр и в слове
                    СЛОН нет одинаковых, то выводим решение ребуса на
                    экран}
                    If (s1*s2=[ ]) And ([a1]*[a2]*[a3]*[a4]=[ ])
                    Then Print(n1,n2);
                    s1:=s1-[a]; {удаляем занесенную цифру}
                End;
                s1:=s1-[x];
            End;
            s1:=s1-[y];
        End;
        s1:=s1-[m];
    End;
    Readln;
End.
```

Задания для самостоятельной работы

1. Дана непустая последовательность символов. Построить и напечатать множества, элементами которых являются встречающиеся в последовательности:
   а) цифры от «0» до «9» и знаки арифметических операций;
   б) буквы от «*A*» до «*F*» и от «*X*» до «*Z*»;
   в) знаки препинания и буквы от «*E*» до «*N*».
2. Составить программу подсчета общего количества цифр и знаков «+», «−», «*» в строке *s*, введенной с клавиатуры.
3. Составить программу печати элементов данного множества в алфавитном порядке.
4. Составить программу формирования множества строчных латинских букв, входящих в строку, введенную с клавиатуры, и подсчета количества знаков препинания в ней.
5. Составить программу подсчета количества цифр в заданной строке и печати их.
6. Составить программу печати по одному разу в алфавитном порядке всех строчных русских гласных букв, входящих в заданный текст.
7. Составить программу печати в алфавитном порядке всех букв текста (текст оканчивается точкой), входящих в него:
   а) не менее двух раз;
   б) не более двух раз;
   в) более двух раз.
8. Составить программу печати в возрастающем порядке всех цифр, входящих в десятичную запись данного десятичного числа.
9. Составить программу печати всех символов заданного текста, входящих в него по одному разу.
10. Составить программу, подсчитывающую число гласных и согласных букв в заданном тексте и определяющую, каких букв больше (гласных или согласных), учесть, что в строке могут быть и другие символы, кроме букв.
11. Составить программу печати всех первых вхождений в данный текст строчных латинских букв, сохраняя их взаимный порядок.
12. Составить программу поиска и печати в порядке убывания всх простых чисел из промежутка [2 ... 201], используя метод «решета Эратосфена».

13. Задано множество вычислительных машин. Известен набор машин, имеющихся в каждом из 10 техникумов города. Построить и распечатать множества, включающие в себя вычислительные машины:

а) которыми обеспечены все техникумы;

б) которые имеет хотя бы один техникум;

в) которых нет ни в одном техникуме.

Упражнение № 25. Комбинированный тип данных (записи)

Пример 54. *Для каждого из двадцати пяти учеников класса известны фамилия и оценки (в баллах) по пяти дисциплинам. Требуется вычислить среднюю оценку каждого из учеников и выбрать человека, имеющего максимальный средний балл.*

```
Program Example_54;
Type
   pupil = Record
fam:String[15];
b1,b2,b3,b4,b5:2..5;
sb:Real;
End;
Var class: Array[1..25]Of pupil;
   p: pupil;
i,m: Integer;
sbmax: Real;
Begin
   For i:=1 To 25 Do {ввод исходных данных}
   With class[i] Do
   Begin
      Writeln('Введите фамилию и пять оценок');
      Readln(fam);
      ReadLn(b1,b2,b3,b4,b5);
   End;
   For i:=1 To m Do {вычисление среднего балла}
   With class[i] Do sb:=(b1+b2+b3+b4+b5)/5;
   sbmax:=0; {поиск максимального среднего балла}
   For i:=1 To m Do
   If class[i].sb>=sbmax Then sbmax:=class[i].sb;
   For i:=1 To m Do {печать результатов}
   If class[i].sb=sbmax Then
   With class[i] Do Writeln(fam:20,' - ',sb:6:3);
   Readln;
End.
```

Пример 55. *Определить дату завтрашнего дня.*

Решение. Чтобы определить дату завтрашнего дня, надо знать не только дату сегодняшнего дня, но и число дней данного месяца (так как если это последний день месяца, то завтра будет первый день следующего), кроме того, надо знать, какой год — високосный или нет (от этого зависит число дней февраля).

Пусть дата вводится следующим образом:

1 2 1997

Первая цифра — это число, вторая — месяц, третья — год. Тогда можно описать запись даты таким образом:

```
Type year=1500..2000;
month=1..12;
day=1..31;
data=Record
   y: year;
   m: month;
   d: day;
End;
```

Заметим, что:

- если это не последний день месяца, то завтра будет этот же год, этот же месяц, а число увеличится на 1;
- если это последний день месяца, то:
  а) если это не декабрь, то завтра будет тот же год, но первое число следующего месяца;
  б) если это декабрь, то завтра наступит следующий год, первый месяц и первое число.

```
Program Example_55;
Type year=1500..2000;
month=1..12;
day=1..31;
data=Record
      y: year;
      m: month;
      d: day;
   End;
Var dat, next: data; {dat - переменная для сегодняшней даты, next -
переменная для определения даты завтрашнего дня}
Function Leap(yy: year): Boolean; {функция, определяющая,
високосный год или нет}
Begin {год называется високосным, если его номер делится на 4, но
   если это год столетия, то номер столетия не делится на 4, то
   есть не делится на 400}
   Leap:=(yy Mod 4=0) And (yy Mod 400 <> 0);
End;
Function Dmonth(mm: month; yy: year): day; {функция определения
количества дней данного месяца в данном году}
Begin
   Case mm Of
   1,3,5,7,8,10,12: Dmonth:=31;
   4,6,9,11: Dmonth:=30;
   2: If Leap(yy) Then Dmonth:=29 Else Dmonth:=28;
   End;
End;
Procedure Tomorrow(td: data; Var nd: data); {процедура определения
завтрашней даты}
Begin {если это не последний день месяца}
   If td.d<> Dmonth(td.m,td.y) Then
```

```
    With nd Do
    Begin
        d:=td.d+1;
        m:=td.m;
        y:=td.y;
    End
    Else {если это последний день месяца}
    If td.m=12 Then {если это декабрь}
    With nd Do
    Begin
        d:=1; m:=1;
        y:=td.y+1;
    End
    Else {если это не декабрь}
    With nd Do
    Begin
        d:=1;
        m:=td.m+1;
        y:=td.y;
    End;
End;
Begin
    Writeln('Введите сегодняшнее число, месяц и год');
    Readln(dat.d,dat.m,dat.y);
    Tomorrow(dat,next);
    Writeln('Завтра будет ');
    Writeln(next.d,'.',next.m,'.',next.y);
    Readln;
End.
```

Задания для самостоятельной работы

1. Написать программу, определяющую:
   а) дату следующего (предыдущего) дня;
   б) дату, которая наступит через *m* дней;
   в) дату, которая была за *m* дней до сегодня;
   г) число суток, прошедших от даты *t*1 до *t*2;
   д) день недели, выпадающий на дату *t*1, если известно, что в первый день нашей эры был понедельник.
2. Дано время, описанное следующим образом:

```
Type time = Record
      h: 0..23;
      m, s: 0..59
End;
```

Описать:
   а) логическую функцию для проверки, предшествует ли время *t*1 времени *t*2 (в рамках суток);
   б) процедуру, присваивающую параметру *t*1 время, на 1 секунду большее времени *t* (учесть смену суток).

3.
```
Const n = 300;
Type Myrecord = Record
   Key: Integer;
   Name: String;
End;
Table=Array[1..n]Of Myrecord.
```

Считая, что в таблице записи имеют различные ключи, описать:

а) процедуру, упорядочивающую записи таблицы по убыванию значений поля *Key*;

б) логическую функцию поиск(T,K,H), определяющую, есть ли в таблице T (все записи которой уже упорядочены по возрастанию значений поля Key) запись со значением поля *Key*, равным K, и, если есть, присваивающую ее номер параметру H.

4. Дан массив, содержащий информацию об учениках некоторой школы:

а) заполнить второй массив данными об учениках только девятых классов;

б) выяснить, на сколько человек в восьмых классах больше, чем в девятых.

5. Багаж пассажира характеризуется количеством вещей и общим весом вещей. Дан массив, содержащий сведения о багаже нескольких пассажиров. Сведения о багаже каждого пассажира представляют собой запись с двумя полями: одно поле целого типа (количество вещей) и одно — действительное (вес в килограммах):

а) найти багаж, средний вес одной вещи в котором отличается не более чем на 0,3 кг от общего среднего веса одной вещи;

б) найти число пассажиров, имеющих более двух вещей и число пассажиров, количество вещей которых превосходит среднее число вещей;

в) выяснить, имеется ли пассажир, багаж которого состоит из одной вещи весом менее 30 кг.

Контрольные работы

Контрольная работа № 8. Строковый тип. Множественный тип

Время выполнения 6 — 8 часов.

Вариант 1

1. Дана последовательность слов. Напечатать все слова, предварительно выполнив преобразования их по правилу: заменить во всех словах первую букву заглавной.

2. Вывести общие русские буквы трех предложений.

3. Решить ребус:

```
  VOLVO
- FIAT
-------
 MOTOR
```

Вариант 2

1. Дана последовательность слов. Напечатать все слова, предварительно выполнив преобразования их по правилу: в словах наибольшей длины удалить среднюю (средние) буквы.

2. Вывести различные русские буквы трех предложений (то есть такие, какие есть только в одном из них).

3. Решить ребус:

```
   ОДИН
   ОДИН
+  ОДИН
   ОДИН
   ОДИН
  -----
   ПЯТЬ
```

Вариант 3

1. Дана последовательность слов. Напечатать все слова, предварительно выполнив преобразования их по правилу: заменить в каждом слове первую встреченную букву «а» на «о», удалив все остальные (если в слове нет такой буквы, то ничего не делать).

2. Вывести наибольшие цифры трех чисел (целых или действительных).

3. Решить ребус: КУБ=$(К+У+Б)^3$.

Вариант 4

1. Дана последовательность слов. Проверить правильность написания сочетаний «жи», «ши», «ча», «ща», «чу» и «щу». Если надо, то исправить ошибки их написания.

2. Даны три строки. Определить, можно ли из символов первых двух строк получить третью строку.

3. Решить ребус:

```
   ТРИ
+
   ДВА
  -----
  ПЯТЬ
```

Контрольная работа № 9. Комбинированный тип данных (записи)

Время выполнения 6 — 8 часов.

Вариант 1

1. Дан текстовый файл, в котором хранятся данные об учениках класса: фамилия, имя, отчество, адрес (улица, дом, квартира) и домашний телефон (если есть). Вывести на экран фамилию, имя и адрес тех учеников, до кого нельзя дозвониться.

2. Дан массив данных о работающих в фирме: фамилия, имя, отчество, адрес (улица, дом, квартира) и дата поступления на работу (месяц, год). Во второй массив записать данные только тех из них, кто на сегодняшний день проработал не менее 5 лет.

Вариант 2

1. Дан текстовый файл, в котором хранятся данные об учениках нескольких школ: фамилия, имя, отчество, адрес (улица, дом, квартира), школа и класс.

Вывести на экран фамилию, имя и адрес тех учеников, кто учится в данной школе в старших классах (номер школы вводить с клавиатуры).

2. Дан массив данных о клиентах пункта проката: фамилия, имя, отчество, адрес (улица, дом, квартира) и что взял (только один предмет). Во второй массив записать данные только тех из них, кто взял телевизор.

Вариант 3

1. Дан текстовый файл, в котором хранятся данные об учениках класса: фамилия, имя, отчество, дата рождения (число, месяц и год). Вывести на экран фамилию и имя тех учеников, у кого сегодня день рождения (сегодняшнюю дату вводить с клавиатуры).

2. Дан массив данных о работающих на фабрике: фамилия, имя, отчество, адрес (улица, дом, квартира) и дата поступления на работу (месяц, год). Определить, есть ли в списке Ивановы (Иванов, Иванова), если есть, то вывести их адрес (адреса).

Вариант 4

1. Дан массив данных об учениках школы: фамилия, имя, адрес (улица, дом, квартира), класс. Записать все данные об учениках данного класса во второй массив. Распечатать его, выделяя тех из них, кто живет на улице Ленина.

2. Дан текстовый файл, в котором хранятся данные о расписании поездов: номер поезда, название (откуда — куда, например Киров — Москва), время прибытия на станцию, время отправления (часы, минуты). Будем считать, что все поезда приходят каждый день. По данному времени определить, какие из поездов стоят сейчас на станции (время вводить с клавиатуры).

Динамические информационные структуры

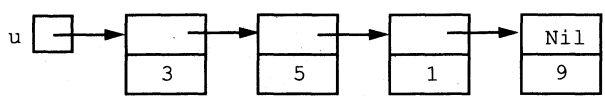
Упражнение № 26. Линейные списки (основные операции). Создание и просмотр списка

Пример. Сформировать список, содержащий целые числа 3, 5, 1, 9.

Решение. Определим запись типа *s* с полями, содержащими характеристики данных — значения очередного элемента и адреса следующего за ним элемента.

```
Type EXS = ^S;
   S=Record
   Data: Integer;
   Next: EXS;
End;
```

Таким образом, мы описали ссылочный тип, с помощью которого можно создать наш связанный однонаправленный список:

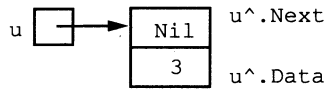


Чтобы список существовал, надо определить указатель на его начало. Опишем переменные:

```
Var u, x: EXS;
```

Создадим первый элемент:

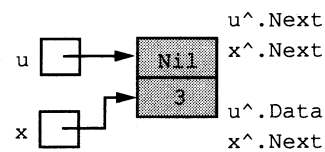
```
New(u); {выделим место в памяти для переменной типа S}
u^. Next:=nil; {указатель пуст}
u^. Data:=3; {информационное поле первого элемента}
```



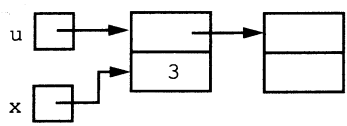
Продолжим формирование списка, для этого надо добавить элемент в конец списка.

```
x:=u; {введем вспомогательную переменную указательного типа,
которая будет хранить адрес последнего элемента списка.
Сейчас последний элемент списка совпадает с его началом}
```

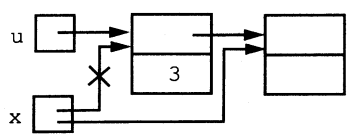
Таким образом, к области памяти (закрашена на рисунке) можно обратиться через два указателя



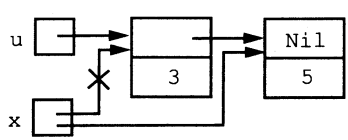

```
New(x^.Next);
{выделим область памяти для следующего элемента списка}
```




```
x:=x^.Next; {переменная x принимает значение
адреса выделенной области памяти}
```




```
x^.Data:=5; {определим значение этого элемента списка}
```

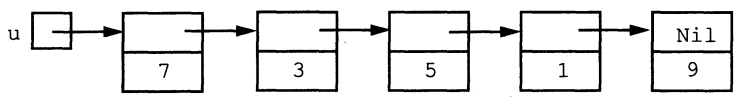



```
x^.Next:=Nil;
```

Оформим создание списка в виде процедуры, в которой его элементы вводятся с клавиатуры.

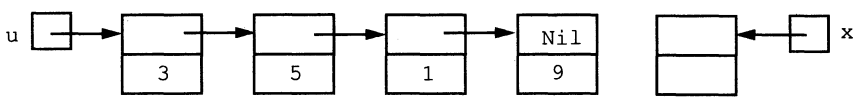
```
Procedure Init(Var u: Exs); {создание списка u}
Var x, y: Exs;
Digit: Integer; {значение информационной части элемента списка}
Begin
   Writeln('Введите список ');
   u:=Nil; {список пуст}
   WriteLn('Введите элементы списка. Конец ввода 0');
   Read(Digit);
   While Digit<>0 Do
   Begin
      New(y); {формирование элемента списка}
      y^.Next:=Nil; y^.Data:=Digit;
      If u=nil Then u:=y {вставляем первый элемент списка}
      Else x^.Next:=y; {вставляем элемент в конец списка}
      x:=y;
      {переносим значение указателя на последний элемент списка}
      Read(Digit);
   End;
   Writeln;
End.
```

Нам необходимо получить список, изображенный на рисунке:

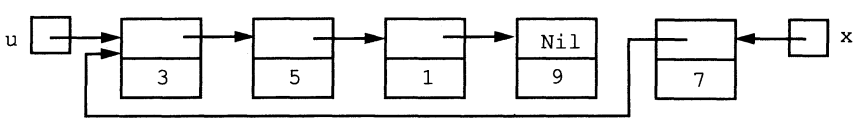


Выполним следующие действия:

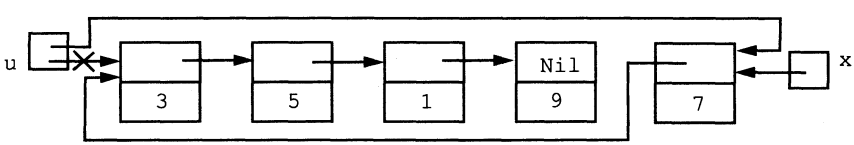
```
New(x); {создание новой динамической переменной}
```




```
x^. data:=7; {информационное поле созданного элемента}
x^. next:=u; {присоединим элементы списка u к созданному элементу}
```




```
u:=x; {изменим значение указателя начала списка}
```



Просмотр списка. Просмотр элементов списка осуществляется последовательно, начиная с его начала. Указатель *p* последовательно ссылается на первый, второй и т.д. элементы списка до тех пор, пока весь список не будет пройден. При этом с каждым элементом списка выполняется операция *Op*. Начальное значение *p* — адрес первого элемента списка *p^*.

```
While p указывает не на конец списка Do
Begin
   выполнить с элементом p^ операцию Op;
   перейти к следующему элементу
End;
```

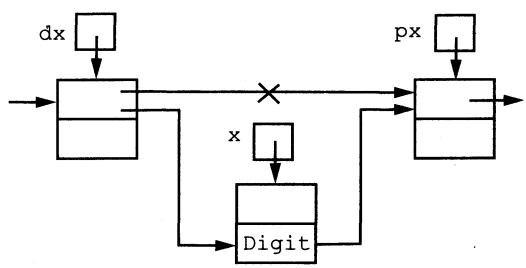
Пусть *Op(p^)* — это вывод элемента *p^* на экран.
Если *p* указывает на конец списка, то его значение равно NIL, то есть:

```
While p<>Nil Do
Begin
   Write(p^,'');
   p:=p^. Next;
End;
```

Пример. Сформировать список целых чисел, упорядоченный по неубыванию.
Решение. После ввода очередного числа с клавиатуры определяем его место в списке. Заметим, что при этом элемент может быть вставлен либо в начало списка, либо в конец его, либо во внутрь. Первый и второй случаи рассмотрены выше. Остановимся на третьем случае.

Для того чтобы вставить в список элемент со значением `Digit` между двумя элементами, надо найти эти элементы и запомнить их адреса (первый адрес — в переменной `dx`, второй — в `px`), после чего установить новые связи с переменной, в которой хранится значение `Digit` .

Графически это можно представить так:



Приведем процедуру `Insinto`, вставляющую в список элемент, переданный ей как параметр. (Адрес первого элемента списка хранится в глобальной переменной *u*.)

```
Procedure Insinto(Digit:Integer; Var u: Exs); {вставка заданного
элемента в список}
Var dx, px, x: Exs;
Begin
   New(x); x^.data:=Digit; x^.next:=NIL;
   {если список пуст, то вставляется первый элемент}
   If (u=Nil) Then u:=x
   Else
   {если список не пуст, то просматриваем его до тех пор, пока
```

```
    не отыщется подходящее место для x^ или не закончится список}
    Begin
        dx:=u; px:=u;
        While (px<>Nil) And (px^.data<=Digit) Do
        Begin dx:=px; px:=px^.next; End;
        If px=Nil {пройден весь список}
        Then dx^.next:=x {элемент добавляется в конец списка}
        Else {пройден не весь список}
        Begin
            x^.next:=px;
            If px=u Then u:=x {вставляем в начало списка}
            Else da^.next:=x; {вставляем внутрь списка}
        End;
    End;
End;
Begin {основная программа}
    u:=NIL; {список пуст}
    Writeln('Введите элемент. Окончание ввода 0');
    Read(Digit);
    While Digit<>0 Do Begin
    InsinTo(Digit, u); Read(Digit); End;
    Writeln;
    Writeln('список: ');
    Print(u); {вывод списка}
End.
```

Пример. Удалить из заданного списка все вхождения элемента с заданным значением информационной части.

Решение. Обозначим `u` — исходный список, `Digit` — значение информационной части удаляемого элемента. Удалить элемент можно из любого места списка.

Опишем процедуру удаления элементов из списка:

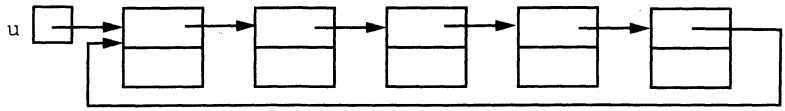
```
Procedure Del(Digit: Integer; Var u: Exs);
Var x, dx: Exs;
Begin
    x:=u;
    While x<>Nil Do
    If x^.Data=digit Then
    Begin
        If x=u Then Begin u:=u^.Next; Dispose(x); x:=u End
        Else
        Begin dx^.Next:=x^.Next; Dispose(x); x:=dx^.Next; End;
    End;
    Else Begin dx:=x; x:=x^.Next; End;
End;
```

Основная программа может иметь следующий вид:

```
Begin
    Init(u); {формирование списка}
    Print(u); {вывод списка}
    Write('Введи число: '); Readln(Digit); Del(Digit,u); Print(u)
End.
```

Пример 56. *N детей располагаются по кругу. Начав отсчет от первого, удаляют каждого k-го, смыкая при этом круг. Определить порядок удаления детей из круга.*

Решение. Для хранения данных об участниках игры используется список.




```
Program Example_56;
Type Children=^Child;
Child=Record
   Data:Integer;
   Next:Children;
   End;
Var circl: Children;
n, k: Integer;

Procedure Init(Var u: Children; Var k: Integer);
Var x, y: Children;
i: Integer;
Begin
   Write('Введите число детей: ');
   Readln(n);
   Write('Введите число слов в считалочке: ');
   ReadLn(k);
   For i:=1 To N do
   Begin
      New(x); x^.data:=i;
      If u=Nil Then u:=x Else y^.next:=x;
      y:=x;
   End;
   x^.next:=u;
End;

Procedure Print(u: Children);
Var x: Children;
Begin
   x:=u;
   Repeat
   Write(x^.Data,'');
   x:=x^.Next;
   Until x=u;
Readln;
End;
Procedure Game(u: Children);
Var x: Children;
   i: Integer;
Begin
   x:=u;
   Repeat
   For i:=1 to k-1 do Begin x:=u; u:=u^.next; End;
```

```
    Write(u^.Data,': '); x^.next:=u^.next; dispose(u); u:=x; Print(u);
    Until u=u^.next;
End;
Begin
    Init(circl); Print(circl); Game(circl); Readln;
End.
```

Упражнение № 27. Стек

Стек — упорядоченный набор элементов, в котором добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека.

```
Type EXST=^ST;
   ST=Record
       Data: Integer;
       Next: EXST;
End;
Var Stack: EXST;
```

Занесение в стек производится аналогично вставке нового элемента в начало списка:

```
Procedure WriteStack(Var u:EXST; digit:Integer); {запись в стек}
Var x: EXST;
Begin
   New(x); {создание нового элемента стека}
   x^.Data:=digit;
   x^.Next:=u {созданный элемент определить как вершину стека}
   u:=x;
End;
```

Основная программа формирования стека будет иметь вид:

```
Var Stack: EXST; {текущая переменная}
   digit: Integer;
   Begin
       Stack:=Nil;
       WriteLn('Введите элементы стека. Окончание ввода - 0');
       Read(Digit);
       While Digit<>0 Do
           Begin
               WriteStack(Stack, digit); Read(digit);
           End;
   End.
```

Извлечение элемента из стека. В результате выполнения этой операции некоторой переменной *i* должно быть присвоено значение первого элемента стека и изменено значение указателя на начало списка.

```
Procedure ReadStack(Var u:EXST; Var i:Integer);
Var x: EXST;
Begin
   i:=u^.Data; x:=u; u:=u^.Next; Dispose(x);
End;
```

Пример 57. *Написать программу, проверяющую своевременность закрытия скобок в строке символов.*

Для решения задачи определим стек, элементами которого являются символы:

```
Type EXST: ^ST;
   ST=Record
       Data: Char;
       Next: EXST;
End;
```

Будем двигаться по строке *a: String* до ее конца. Если в процессе просмотра встретится одна из открывающих скобок — {, (, [— занесем ее в стек. При обнаружении закрывающейся скобки, соответствующей скобке, находящейся в вершине стека, последняя удаляется. При несоответствии скобок выдается сообщение об ошибке.

```
Program Example_57;
Type Exst=^st;
   St=Record
       Next:exst;
       Data:Char;
   End;
Var a:String;
   f:Boolean;
   i:Integer;
Procedure Writestack(Var x1:exst; c:Char); {Процедура занесения
элемента в стек}
Var u:exst;
Begin
   New(u); u^.Data:=c; u^.next:=x1; x1:=u
End;
Procedure delstack(Var x1:exst);
{Процедура удаления верхнего элемента стека}
Var u:exst;
Begin
   u:=x1; x1:=x1^.next; Dispose(u);
End;
Procedure Solve(a:String); {Процедура проверки правильности рас-
становки скобок}
Var stack: Exst;
Begin
   stack:=Nil; i:=1;
   While (i<=length(a)) And f Do
   Begin
       If (a[i]='(')Or(a[i]='{')Or(a[i]='[')
       Then Writestack(stack,a[i])
       Else If (a[i]=')') Or (a[i]='}') Or (a[i]=']')
       Then If Ord(stack^.data)-Ord(a[i])<=2 Then delstack(stack)
       Else f:=False;
       Inc(i);
   End;
End;
```

```
Begin {Основная программа}
   Writeln('введите строку'); Readln(a); f:=True;
   If a<>'' Then
   Begin
      Solve(a);
      If f Then Writeln('все скобки расставлены верно')
      Else Writeln('скобка ',a[i-1],' закрыта преждевременно');
   End;
   Else Writeln('строка пуста');
   Readln;
End.
```

Пример 58. *Написать программу вычисления значения выражения, представленного в обратной польской записи.*

Обычная запись:	Обратная польская запись:
$(b+c)*d$	$bc+d*$
$a+(b+c)*d$	$abc+d*+$

Решение. Логика решения этой задачи не вызывает трудностей. Просматривая строку, анализируем очередной символ, если это:

- число, то записываем его в стек;
- знак, то достаем два элемента из стека, выполняем математическую операцию, определяемую этим знаком, и заносим результат в стек.

```
Program Example_58;
Type Exst=^st;
   St=Record
      Data:Real;
      Next:exst;
   End;
Var aa:String;
   stack:exst;
   i,k:Integer;
   x,y:Real;
Procedure Writestack(Var x1:exst; c:Real);
{запись нового элемента в стек}
Var u:exst;
Begin
   New(u); u^.Data:=c; u^.next:=x1; x1:=u
End;
Procedure Readstack(Var x1:exst;Var d:Real);
{извлечение элемента из стека}
Var u:exst;
Begin
   d:=x1^.data; u:=x1; x1:=x1^.next; Dispose(x1);
End;
Procedure Operation(a,b:Real;c:Char);
{выполнение действия и занесение результата в стек}
Var d:Real;
Begin
   Case c Of
   '+': d:=a+b; '-': d:=b-a; '*': d:=a*b; '/': d:=b/a;
```

```
    Else Writeln('ошибка в записи');
    End;
    Writestack(stack,d);
End;
Begin {основная программа}
    Writeln('введите выражение в обратной польской записи');
    Readln(aa); Stack:=Nil;
    For i:=1 To length(aa) Do
    Begin
        Val(aa[i],x,k);
        If k=0 Then Writestack(stack,x);
        Else
        Begin
            Readstack(stack,x);
            Readstack(stack,y); Operation(x,y,aa[i]);
        End;
    End;
    Write('ответ:'); Writeln(stack^.data); Readln;
End.
```

Упражнение № 28. Очереди

Очередь —упорядоченный набор элементов, в котором извлечение элементов происходит с одного его конца, а добавление новых элементов — с другого.

```
Type EXO=^O;
    O=Record
        Data: Integer;
        Next: EXO;
    End;
```

Над очередью определены две операции: занесение элемента в очередь и извлечение элемента из очереди. Поскольку очередь представлена списком, занесение элемента в очередь соответствует занесению элемента в конец списка.

```
Procedure WriteO (Var x1,x2: EXO; c: Integer);
Var u: EXO;
Begin
    New(u); u^.data:=c; u^.next:=NIL;
    If x1=NIL Then x1:=u {если очередь пуста}
    Else x2^.next:=u; {заносим элемент в конец списка}
    x2:=u;
End;
```

Основная программа, создающая очередь, может быть такой:

```
Begin
    x1:=NIL; x2:=NIL;
    Writeln('Введите элементы очереди. Конец ввода - 0'); Read(Digit);
    While Digit<>0 Do
    Begin
        WriteO(x1,x2,Digit); Read(Digit);
    End;
End.
```

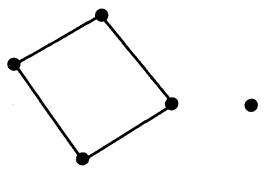
Процедура извлечения элемента из очереди аналогична удалению элемента из начала списка. Поскольку извлечение элемента из пустой очереди осуществить нельзя, опишем логическую функцию, проверяющую, есть ли элементы в очереди.

```
Procedure Reado(Var x1,x2:EXO;Var c:Integer);
Var u: EXO;
Function Nul (x1:EXO):Boolean;
Begin
   Nul:=(x1=Nil);
End;
Begin
   If Nul(x1) Then Writeln('Очередь пуста')
   Else
   Begin
      c:=x1^.Data; u:=x1; x1:=x1^.Next; Dispose(u);
   End;
End;
```

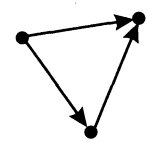
Упражнение № 29. Деревья

Граф — непустое множество точек (вершин) и множество отрезков (ребер), концы которых принадлежат заданному множеству точек.

Граф называется *связным*, если любые две его вершины соединены путем.



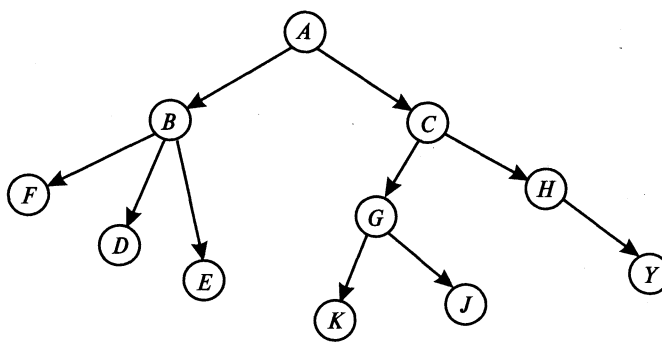
Несвязный граф



Связный граф

Связный граф без циклов называется *деревом*.

Пример.



Для дальнейшей работы с деревьями необходимо определить ряд понятий.

• Вершина *y*, находящаяся непосредственно ниже вершины *x*, называется непосредственным потомком *x*, а вершина *x* называется предком *y*.

• Если вершина не имеет потомков, то она называется терминальной вершиной или *листом*, если имеет, то называется *внутренней вершиной*.

• Число непосредственных потомков внутренней вершины называется ее *степенью*.

• *Степенью дерева* называется максимальная степень всех вершин.

Например:

• вершины *F*, *D*, *E* являются непосредственными потомками вершины *B*;

• вершины *F*, *D*, *E* являются *листьями*;

• вершины *C*, *G*, *H* — внутренние;

• степень вершины *B* — 3, а вершины *H* — 1;

• степень дерева равна 3.

Двоичное дерево — дерево, в котором из каждой вершины исходит не более двух ребер.

Дерево — это сложная динамическая структура данных, применяющаяся для эффективного хранения информации.

Очевидно, что для описания дерева требуются ссылки. Опишем как переменные с фиксированной структурой сами вершины, тогда степень дерева будет определять число ссылочных компонент, указывающих на вершины поддеревьев. В бинарном дереве их два — левое и правое.

```
Type TreeLink =^Tree;
   Tree=Record
       Data: <тип данных>;
       Left, Right: TreeLink;
   End;
```

Корень дерева опишем в разделе описания переменных:

```
Var kd: TreeLink;
```

К основным операциям над деревьями относятся

• занесение элемента в дерево;

• обход дерева;

• удаление элемента из дерева.

Рассмотрим вставку и обход дерева на примере следующей задачи.

Пример. Создать и вывести на экран дерево, элементы которого вводятся с клавиатуры и имеют целый тип. Причем для каждой вершины дерева во всех левых вершинах должны находиться числа меньшие, а в правой — бóльшие, чем числа, хранящиеся в этой вершине. Такое дерево называется *деревом поиска*.

Опишем процедуру вставки в дерево новой вершины. При вставке в дерево вершина вставляется либо как поддерево уже существующей вершины, либо как единственная вершина дерева. Поэтому и левая, и правая связи новой вершины должны быть равны *Nil*. Когда дерево пусто, значение передаваемой в виде параметра ссылки равно *Nil*. В этом случае надо изменить ее так, чтобы она указывала на новую вершину, которая была вставлена как корневая. При вставке второго элемента переданный из основной программы параметр *t* уже не будет равен *Nil*, и надо принимать решение о том, в какое поддерево необходимо вставить новую вершину.

```
Procedure InsTree(n:Integer; Var t:treelink);
Begin
   If t=NIL Then
   Begin
       new(t);
       With t^ Do
```

```
        Begin
            left:=NIL; right:=NIL; data:=n
        End;
    End
    Else If n<=t^.data Then InsTree(n,t^.left)
    Else InsTree(n,t^.right);
End;
```

Опишем процедуру вывода значений элементов двоичного дерева на экран. Для этого необходимо выполнить полный обход дерева. При обходе дерева его отдельные вершины посещаются в определенном порядке. Вывод двоичного дерева можно производить рекурсивно, выполняя для каждой вершины три действия:

- вывод числа, хранящегося в узле;
- обход левого поддерева;
- обход правого поддерева.

Порядок выполнения этих действий определяет способ обхода дерева. Способы вывода:

- прямой вывод (сверху вниз);
- обратный вывод (слева направо);
- концевой вывод (снизу вверх).

Процедура обратного вывода дерева имеет следующий вид:

```
Procedure PrintTree(t:treelink);
Begin
    If t<>NIL Then
    Begin
        PrintTree(t^.left); Write(t^.data:3); PrintTree(t^.right)
    End;
End;
```

Основная программа осуществляет ввод чисел с клавиатуры. Используются переменная *nd* типа *treelink* — значение указателя на корень дерева; переменная *Digit* типа *integer* для хранения очередного введенного числа.

```
Begin {Основная программа}
    Writeln('окончание ввода - 0'); kd:=Nil; Read(Digit);
    While Digit<>0 Do
    Begin
        InsTree(Digit,kd);
        Writeln('введите очередное число');
        Read(Digit);
    End;
    PrintTree(kd);
End.
```

В дереве поиска можно найти место каждого элемента, двигаясь от корня и переходя на левое или правое поддерево в зависимости от значений встречающихся данных.

Использование деревьев поиска значительно сокращает время решения задачи. В среднем для нахождения элемента в списке надо просмотреть половину списка, то есть если в списке N элементов, то надо выполнить $N/2$ сравнений. Для поиска же заданного элемента в дереве при его правильной организации может потребоваться не более $\log N$ сравнений.

Правильно организованным деревом считается *идеально сбалансированное дерево*, то есть такое, что для каждой его вершины число вершин в левом и правом поддереве различается не более чем на 1.

Пример 59. *Сформировать идеально сбалансированное дерево, элементами которого являются N чисел, вводимых с клавиатуры.*

Решение. Поскольку требуется построить идеально сбалансированное дерево, то его узлы в процессе построения должны распределяться равномерно. Сформулируем *правило равномерного распределения узлов* при известном их числе:

- взять один узел в качестве корня;
- построить левое поддерево с числом узлов $n1 = N\text{div}2$ тем же способом;
- построить правое поддерево с числом узлов $n2 = N - n1 - 1$ тем же способом.

```
Program Example_59;
Uses Crt;
Type Pt=^Node;
      Node=Record
          Data:Integer;
          Left, Right: Pt;
      End;
Var n:Integer;
   kd:Pt;
   f:text;
Function Tree(n:Integer):pt;
Var newnode:pt;
   x,n1,n2:Integer;
Begin
   If n=0 Then Tree:=nil
   Else
   Begin
      n1:=n Div 2; n2:=n-n1-1; Read(f,x); New(newnode);
      With newnode^ Do
      Begin
          Data:=x; Left:=Tree(n1); Right:=Tree(n2);
      End;
      Tree:=newnode;
   End;
End;
Procedure PrintTree(t:pt; h:Integer);
Var i:Integer;
Begin
   If t<> Nil Then
   With t^ Do
   Begin
      PrintTree(left,h+1);
      For i:= 1 To h Do Write(' ');
      Writeln(Data:6); PrintTree(Right,h+1);
   End;
End;
Begin
      Clrscr; Assign(f,'c:\f.pas');
```

```
        Reset(f); Write('n=');
        Readln(n); kd:=tree(n); printtree(kd,0); Readln;
End.
```

Пример 60. *Задана последовательность слов. Определить частоту вхождения каждого из слов в последовательность.*

Решение. Для решения задачи любое слово ищется в дереве, которое на начальном этапе пусто. Если слово найдено, то счетчик его вхождений увеличивается на единицу, если нет, то слово включается в дерево с единичным значением счетчика.

```
Program Example_60;
Uses Crt;
Type Words=^Wordtree;
   Wordtree=Record
       data: String;
       k:Integer;
       left,right:Words;
   End;
Var nInteger;kd:Words;
   x: String; f: text;
Procedure Tree(x:String; Var p:Words);
Begin
   If p=Nil Then
   Begin
       New(p);
       With p^ Do
       Begin
           k:=1; Data:=x; Left:=Nil; Right:=Nil;
       End;
   End
   Else If x>p^.Data Then Tree(x,p^.Left)
       Else If x<p^.data Then Tree(x,p^.Right)
           Else Inc(p^.k);
End;
Procedure PrintTree(t:Words; h:Integer);
Var i:Integer;
Begin
   If t <> nil Then
       With t^ Do
       Begin
           PrintTree(Left,h+1);
           For i:=1 To h Do Write(' ');
           Writeln(data,',(',k,')'); PrintTree(Right,h+1);
       End;
End;
Begin
   Clrscr; Assign(f,'c:\f.dan');
   Reset(f); Write('n='); Readln(n); kd:=Nil;
   While n>0 Do Begin ReadLn(f,x); Tree(x,kd); dec(n); End;
   Close(f); PrintTree(kd,0); Readln;
End.
```

Задания для самостоятельной работы

Указатели и ссылки

1. Пусть переменные *x* и *y* имеют значения, показанные на рисунке.

Каковы типы переменных *x* и *x*^? Что будет выдано на печать в результате выполнения следующих операторов:

```
Var x, y: ^Integer;
x^:=y^;
If x=y Then x:=Nil Else If x^=y^ Then y:=x;
If x=y Then q^:=4;
WriteLn(q^);
```

2. Имеется программа:

```
Program Dynamik;
Var x: ^Integer; y: Integer;
Begin {a}
   New(x); {b}
   x^:=10; {c}
   y:=x^+6;
   Dispose(x); {d}
   WriteLn(y) {e}
End.
```

а) Какие переменные существуют в каждой из точек *a*, *b*, *c*, *d*, *e* и каковы их значения в эти моменты?

б) Можно ли переменной *x* присвоить ссылку на переменную *y*? Можно ли с помощью процедуры *Dispose* уничтожить переменные *x* и *y*?

3.
```
Type Chain=^Elem;
   Elem=Record
   Data: Integer;
   Next: Chain;
   End;
Var p,q: Chain;
```

Привести схемы памяти после выполнения следующих операторов:

а) `New(p); p^.Data:=4; p^.Next:=Nil;`
б) `New(p); p^.Data:=7; p^.Next:=p;`
в) `New(q); q^.Data:=2; q^.Next:=Nil;`
г) `New(p); p^.Data:=1; p^.Next:=q;`
д) `New(p); p^.Data:=5; New(p^.Next); p^.Next^:=p^`

Линейные списки

1. Какие операции требуется выполнить для вставки элемента списка?
2. Можно ли для построения списка обойтись одной переменной?

3. Сколько элементов может содержать список? Когда прекращать ввод элементов списка?

4. Описать функцию, которая вычисляет среднее арифметическое элементов непустого списка.

5. Описать рекурсивную и нерекурсивную процедуры или функции проверки наличия в списке заданного числа.

6. Описать процедуру, которая меняет местами первый и последний элементы непустого списка.

7. Описать процедуру, которая вставляет новый элемент:
   а) перед каждым вхождением заданного элемента;
   б) за каждым вхождением заданного элемента.

8. Описать процедуру или функцию, которая:
   а) проверяет на равенство списки $L1$ и $L2$;
   б) определяет, входит ли список $L1$ в список $L2$;
   в) переносит в конец непустого списка L его первый элемент;
   г) переносит в начало непустого списка его последний элемент;
   д) копирует в список L за каждым вхождением заданного элемента все элементы списка $L1$.

9. Описать процедуру, которая объединяет два упорядоченных по неубыванию списка $L1$ и $L2$ в один упорядоченный по неубыванию список:
   а) построив новый список L;
   б) сменив соответствующим образом ссылки в $L1$ и $L2$.

10. Описать функцию, которая проверяет, упорядочены ли элементы списка по алфавиту.

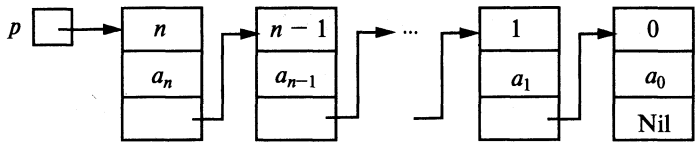
11. Описать функцию, подсчитывающую число слов списка, которые:
   а) начинаются и оканчиваются одной и той же литерой;
   б) начинаются с той же литеры, что и следующее слово.

12. Описать процедуру, которая удаляет:
   а) из списка L второй элемент, если такой есть;
   б) из списка L за каждым вхождением элемента E один элемент, если такой есть и он отличен от E;
   в) из списка L первый отрицательный элемент, если такой есть;
   г) из списка L все отрицательные элементы.

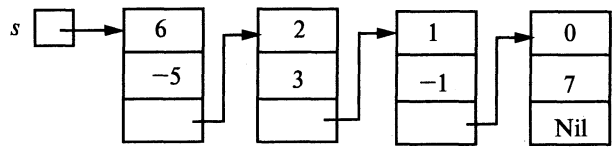
13. Описать процедуру, которая формирует список L, включив в него по одному разу элементы, которые:
   а) входят хотя бы в один из списков $L1$ и $L2$;
   б) входят одновременно в оба списка $L1$ и $L2$;
   в) входят в список $L1$, но не входят в список $L2$;
   г) входят в один из списков $L1$ и $L2$, но в то же время не входят в другой из них.

14. Многочлен $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ с целыми коэффициентами можно представить в виде списка (рис. *a*), причем если $a_i=0$, то соответствующий элемент не включается в список (на рис. *b* показано представление многочлена $s(x) = -5x^6 + 3x^2 - x + 7$).

Описать на Паскале тип данных, соответствующий такому представлению многочленов, определить следующие функции и процедуры для работы с этими списками-многочленами:
   а) логическую функцию `equality(p,q)`, проверяющую на равенство многочлены p и q;
   б) функцию `Meaning(p,x)`, вычисляющую значение многочлена в целочисленной точке x;



a



b

в) процедуру `add(p,q,r)`, которая строит многочлен *p* — сумму многочленов *q* и *r*;

г) процедуру `print(p,v)`, печатающую многочлен *p* как многочлен от переменной, однобуквенное имя которой является значением литерного параметра *v*, например, для указанного выше многочлена *s* процедура `print(s,'x')` должна напечатать

```
-5y^6 + 3y^2 - y+7.
```

15. Используя стек, разработайте программу, которая преобразует обычную форму записи арифметического выражения в обратную польскую запись.

Очереди

16. За один просмотр файла *f*, элементами которого являются действительные числа, и без использования дополнительных файлов напечатать его элементы так: сначала все числа, меньшие заданного *a*, затем все числа из отрезка [*a*, *b*] и все остальные, сохраняя их взаимный порядок в каждой из групп чисел.

Идея решения. Для решения задачи будем последовательно считывать из файла числа. Если очередное число меньше *a*, то выведем его на экран, если оно принадлежит отрезку [*a*, *b*], то занесем его в первую очередь, иначе — занесем его во вторую очередь. После того, как все данные будут рассмотрены, выведем на экран обе очереди.

17. Содержимое текстового файла *f*, разделенного на строки, переписать в текстовый файл *g*, перенося при этом в конец каждой строки все входящие в него цифры, с сохранением взаимного исходного порядка.

Деревья

18. Описать рекурсивную логическую функцию, проверяющую наличие заданного числа в сформированном дереве.

19. Описать рекурсивную числовую функцию, подсчитывающую сумму элементов дерева.

20. Описать функцию, которая находит наибольший элемент непустого дерева.

21. Описать рекурсивно и нерекурсивно логическую функцию, входными параметрами которой являются два дерева, проверяющую на равенство эти деревья.

22. Описать логическую функцию, проверяющую, есть ли в непустом дереве хотя бы два одинаковых элемента.

23. Описать процедуру, которая:
- каждый элемент дерева возводит в квадрат;
- каждый отрицательный элемент заменяет на его абсолютную величину;
- печатает все элементы дерева по уровням.

24. Описать функцию, которая:
- находит максимальный элемент в дереве T;
- находит сумму всех элементов дерева;
- подсчитывает число элементов в дереве;
- для заданного числа x находит число его вхождений в дерево.

Графика Турбо-Паскаля

Упражнение № 30. Рисование узоров

Пример 61. *Муаровый узор. Решение.* Рассмотрим пересечение двух семейств расходящихся отрезков, в которых можно выделить следующие параметры:

$k1$, $k2$ — расстояния между отрезками слева и справа;

h — смещение-наклон вниз (вверх) всего семейства.

Этапы построения семейств линий:

1) построение горизонтальной линии осуществляется графической процедурой `line (0,k,640,k);`

2) построение семейства из n линий задается в цикле

```
for i:=1 to n do line(0,i*k,640, i*k);
```

3) наклонное семейство получается добавлением шага h к одной из координат по y:

```
for i:=1 to n do line(0,i*k,640, i*k + h); (h — величина смещения вниз или вверх);
```

4) расширяющиеся или сужающиеся прямые возникают при различных $k1$ и $k2$:

```
for i:=1 to n do line(0,i*k1,640, i*k2 + h);
```

5) второе семейство линий можно построить аналогично, но с зеркальной симметрией:

```
for i:=1 to n do line(0,i*k2+h,640, i*k1);.
```

```
Program Example_61;
uses crt, graph;
var gD,gM,i,k1,k2,h: integer;
begin
   k1:=8; k2:=3; h:=110; gD:=Detect;
   InitGraph(gD,gM,''); setcolor(green);
   for i:=1 to (480 div k1) do
   begin
      line(0,i*k1,640,i*k2+h); line(0,i*k2+h,640,i*k1);
   end;
   repeat until keypressed;
   CloseGraph;
end.
```

Пример 62. *Звезда.*

Решение. Узоры можно моделировать с помощью простых мотивов, в частности, штрихованного угла. Штрихованный угол задается координатами трех точек и изображается семейством отрезков, соединяющих точки разбиения сторон угла с одинаковыми номерами. Оформим отдельный угол процедурой. Зададим n углов, опирающихся на окружность. Пусть p — число вершин, r, $r2$ — диаметры внешней и внутренней окружностей.

```
Program Example_62;
uses crt, graph;
const p=5;
Var gd,gm,i: integer;
    r,r2,al,a2,a3,t: real;

Procedure Ugol(x1,y1,x2,y2,x3,y3:real;n:integer);
Var k: integer;
a,hx1,hx2,hy1,hy2: real;
Begin
   hx1:=(x2-1)/n; hx2:=(x3-x2)/n;
   hy1:=(y2-y1)/n; hy2:=(y3-y2)/n;
   for k:=0 to n do
   line(round(x1+k*hx1),round(y1+k*hy1),
   round(x2+k*hx2),round(y2+k*hy2));
End;
Begin
   gd:=Detect; InitGraph(gd,gm,'');
   r:=180; r2:=60; t:= 200 ; setcolor(green);
   for i:=0 to p-1 do
   Begin
      al:=2*pi*i/p; a2:=al+pi/p; a3:=al-pi/p;
      Ugol(t,t,t+r*sin(al),t+r*cos(al),t+r2*sin(a2),
      t+r2*cos(a2),25);
      Ugol(t,t,t+r*sin(al),t+r*cos(al),t+r2*sin(a3),
      t+r2*cos(a3),25);
   End;
   repeat until keypressed; CloseGraph;
End.
```

Упражнение № 31. Графики функций

Пример 63. *Разработать программу построения графика функции* $y = x^2\sin(1/x)$ *на произвольном интервале* $[a, b]$.

Решение. Процесс построения графика оформим в виде процедуры.

```
Program Example_63;
uses crt, graph;
Var gd,gm, n: integer;
   a,b: real;
function f(x:real):real;
Begin
   if x<>0 then f:=x*x*sin(1/x);
End;
```

```
Procedure grafun(x0,x1,y0,y1,n:word; a,b:real);
Var h,m,x, t1,t2: real;
   i, u,v,xv,yv: word;
Begin
   h:=(b-a)/n;
   m:=abs(f(a)); {поиск максимума Sf(x)S}
   for i:=1 to n do if m<abs(f(a+i*h)) then m:=abs(f(a+i*h));
   t1:=(x1-x0)/(b-a); t2:=(y1-y0)/(2*m);
   {построение координатных осей}
   setfillstyle(1,15);  bar(x0-5,y0-5,x1+5,y1+5);
   xv:=round(x0-a*t1);  yv:=round((y0+y1)/2);
   setcolor(1); line(xv,y0,xv,y1);
   line(x0,yv,x1,yv); {установка курсора в начало графика}
   Moveto(x0,yv-round(f(a)*t2)); {построение графика}
   setcolor(3);
   for i:=1 to n do
   Begin
      x:=a+i*h; u:=x0+round((x-a)*t1);
      v:=yv-round(f(x)*t2); lineto(u,v);
   End;
End; {конец процедуры}
Begin
   clrscr; write('введи a,b и n: ');
   readln(a,b,n); gd:=Detect;
   InitGraph(gd,gm,'');
   grafun(100,500,50,300,n,a,b);
   grafun(550,620,10,100,200,-0.1,0.1);
   repeat until keypressed; CloseGraph;
End.
```

График рассматриваемой функции представлен на двух отрезках [*a*, *b*] и [−0,1, 0,1]. Чтобы построить график другой функции, достаточно задать ее аналитический вид в описании функции (*function f*).

Упражнение № 32. Динамические рисунки

Пример 64. Динамическая модель Солнечной системы

Решение. Организуем движение точки (Земли)по окружности, в центре которой размещается круг Солнце. Установку точки на орбите осуществим по параметрическим формулам окружности:

```
x0:= 320 + r1*sin(A1);
y0:= 240 + r1*cos(A1),
```

где *r*1 — радиус орбиты Земли, *A*1— параметрический угол, меняющийся от 0 до 360 градусов. Чтобы организовать движение, достаточно в цикле устанавливать точку с координатами (*x*0, *y*0) для всех углов *A*1, принимающих значения от 0 до 360 с шагом *h*. Аналогичная процедура справедлива и для второй точки (Луны), которая изображается по подобным формулам, в которых центр орбиты (Земля) является подвижным:

```
x:= x0 + r*sin(A);
y:= y0 + r*cos(A),
```

где r — радиус орбиты Луны, A — угол вращения.

```
Program Example_64;
uses crt, graph;
   const pi=3.1415;
   Var gD,gM: integer;
   fi,h,fi1,h1: real;
   x0,y0,x,y,r,r1: integer;
Begin
   gD:=Detect; InitGraph(gD,gM,' ');
   h:=5; h1:=1;fi:=0; fi1:=0; r:=20;
   r1:=100; circle(320,240,10);
   repeat
   x0:=round(r1*sin(fi1))+320;
   y0:=round(r1*cos(fi1))+240;
   x:=x0+round(r*sin(fi));
   y:=y0+round(r*cos(fi));
   fi:=fi+2*pi*h/360;
   fi1:=fi1+2*pi*h1/360;
   putpixel(x0,y0,15);
   putpixel(x,y,15);
   delay(50);
   putpixel(x,y,0);
   putpixel(x0,y0,0);
   until keypressed;
   CloseGraph;
End.
```

Задания для самостоятельной работы

1. Разработайте программу моделирования паркетов из прямоугольных треугольников.
2. Разработайте программу моделирования паркетов из равносторонних треугольников.
3. Разработайте программу моделирования паркетов из произвольных остроугольных треугольников.
4. Разработайте программу моделирования паркетов из произвольных треугольников.
5. Разработайте программу моделирования паркетов из параллелограммов.
6. Разработайте программу моделирования паркетов из ромбов
7. Разработайте программу моделирования паркетов из трапеций.
8. Разработайте программу моделирования паркетов из правильных многоугольников.
9. Разработайте программу моделирования паркетов из неправильных многоугольников.
10. Разработайте программу моделирования муаровых узоров из семейств парабол.
11. Разработайте программу моделирования муаровых узоров из семейств полиномов третьего порядка.
12. Разработайте программу построения пяти графиков заданных функций.
13. Постройте линии уровня для функции $z = f(x,y)$.
14. Разработайте программу построения гистограмм и диаграмм.

15. Разработайте программу изображения поверхности как функции двух переменных.

16. Постройте динамическую модель Солнечной системы для двух планет.

Лабораторные работы

Лабораторные работы по языку Паскаль проводятся в компьютерном классе и состоят в реализации и отладке программ, написанных при решении задач. Каждому студенту предварительно предлагается для каждой работы 2—3 задачи из приведенных выше списков, которые он должен выполнить в системе программирования Турбо-Паскаль.

Целесообразно проведение следующих лабораторных работ:

1. Основные конструкции языка Паскаль. Составление простейших программ.
2. Ветвления. Развилка и выбор.
3. Циклы. Типовые задачи реализации циклических вычислительных процессов.
4. Символьный тип.
5. Перечисляемые и интервальные типы.
6. Процедуры и функции.
7. Рекурсивные процедуры и функции.
8. Файлы.
9. Массивы. Типовые задачи обработки массивов.
10. Строковый тип.
11. Множества.
12. Записи.
13. Динамические информационные структуры.
14. Графика Турбо-Паскаля.

Дополнительная литература

1. *Абрамов В. Г., Трифонов Н. П., Трифонова Г. Н.* Введение в язык Паскаль. — М.: Наука, 1988.

2. *Абрамов С. А.* и др. Задачи по программированию.— М.: Наука, 1988.

3. *Дайтибегов Д. М., Черноусов Е. А.* Основы алгоритмизации и алгоритмические языки. — М.: Финансы и статистика, 1992.

4. *Джонс Ж., Харроу К.* Решение задач в системе Турбо-Паскаль: Пер. с англ. — М.: Финансы и статистика, 1991.

5. *Довгаль С. И., Литвинов Б. Ю., Сбитнев А. И.* Персональные ЭВМ: Турбо-Паскаль 7.0, Объектное программирование, локальные сети. — Киев: Информсистема сервис, 1993.

6. *Зуев Е. А.* Практическое программирование на языке Турбо-Паскаль 6.0—7.0. — М.: Радио и связь, 1994.

7. *Зуев Е. А.* Программирование на языке TURBO-PASCAL 6.0—7.0. — М.: Радио и связь, 1993.

8. Информатика. Задачник-практикум: В 2 т. Т. 1 / Под ред. И. Г. Семакина и Е. К. Хеннера. — М.: Лаборатория базовых знаний, 1999.

9. *Немнюгин С. А.* Турбо-Паскаль. — СПб.: Питер, 2000.

10. *Пильщиков В. Н.* Сборник упражнений по языку Паскаль. — М.: Наука, 1989.

11. *Рубекинг И.* Турбо-Паскаль для Windows. — М.: Мир-СК Ферлаг Интернешнл, 1994.

12. *Семакин И. Г.* Лекции по программированию. — Пермь: Изд-во ПГУ, 1996.
13. *Сергиевский М. В., Шалашов А. В.* Турбо-Паскаль 7.0. — М.: Машиностроение, 1994.
14. Языки программирования Ада, Си, Паскаль. Сравнение и оценка: Пер. с англ. / Под ред. А. Фьюэра и Н. Джехани. — М.: Радио и связь, 1989.

§ 2. МЕТОДЫ И ИСКУССТВО ПРОГРАММИРОВАНИЯ

Рекомендации по проведению занятий

Искусство программирования — это умения и навыки составления алгоритмов и программ по различным методикам. Содержание настоящего раздела является практически неисчерпаемым. Рекомендуется ограничиться классическими темами: «рекурсивные алгоритмы» и «сортировка и поиск».

На семинарских занятиях полезно разобрать известные алгоритмы решения ставших традиционными в курсах информатики задач, провести анализ их сложности и эффективности. Практические занятия разумно посвятить разбору примеров и задач, выполнению практических упражнений с пошаговой трассировкой предлагаемых образцов программ, с последующей модификацией их. Рекомендуется каждому студенту выполнить самостоятельно все задания, приведенные в конце каждой темы.

Темы семинарских занятий

1. Проектирование алгоритмов и программ. Нисходящее и восходящее проектирование.
2. Рекурсия. Рекурсивные алгоритмы.
3. Алгоритмы с возвратом.
4. Сортировка. Методы сортировки массивов.
5. Улучшенные методы сортировки массивов.
6. Методы сортировки файлов.
7. Поиск в массиве данных. Бинарный поиск.

Рекомендации по программному обеспечению

Все алгоритмы и программы, рассматриваемые ниже, приведены в нотации Турбо-Паскаля. Однако для организации обучения по данному разделу можно использовать любую систему программирования, электронные таблицы.

Краткие сведения

Рекурсивные алгоритмы

Ситуация, когда алгоритм вызывает себя в качестве вспомогательного, называется *рекурсией* (это слово происходит от латинского *recursio* — возвращение). Рекурсивными бывают подпрограммы-процедуры и подпрограммы-функции. В рекурсивных подпрограммах в теле процедуры или функции имеется вызов этих же

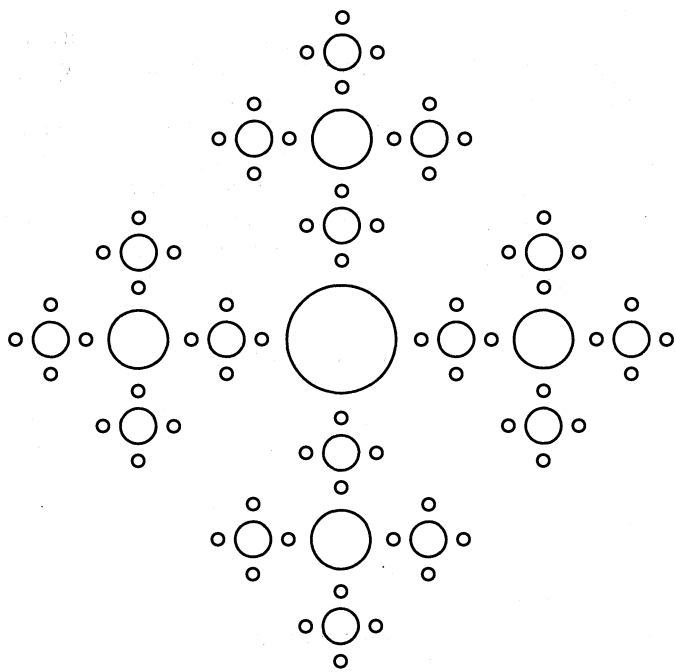
подпрограмм. Процесс обращения к себе самому может длиться бесконечно. Для обеспечения остановки устанавливают барьер в виде некоторого условия.

Задачи и упражнения

Рекурсивные алгоритмы

Упражнение № 1. Знакомство с рекурсивными алгоритмами

Пример 65. *Написать программу построения изображения, представленного на следующем рисунке:*



Анализ изображения: на рисунке большая окружность окружена четырьмя окружностями поменьше, каждая из которых в свою очередь окружена четырьмя окружностями с еще меньшим радиусом. Всего изображено четыре уровня окружностей.

Для того, чтобы составить программу построения этого изображения, можно:

- описать процедуру изображения одной окружности с четырьмя окружностями поменьше;
- для изображения каждой окружности следующего уровня использовать эту же процедуру, только с другими значениями параметров — координат центров, величин радиусов и т. д.

Опишем алгоритм рисования окружности радиуса r с центром в точке (x,y) с четырьмя окружностями вокруг. При этом необходимо знать расстояния $(r1)$ от точки (x,y) до центров окружностей окружения (радиусы орбиты), которые, очевидно, равны. Пусть $\frac{r'}{r} = k1$, где r' — радиус окружности окружения, $\frac{r1}{r} = k2$.

```
Procedure Picture1(x,y,r,r1:Integer);
Begin
{изобразить окружность с центром (x,y) и радиусом r} ;
{вычислить r1};
   For i:=1 To 4 Do
   Begin {вычислить координаты центра (x1,y1) i-й окружности};
      Picture1(x1,y1,<новое значение r>, r1);
   End;
End.
```

Таким образом, в описанной процедуре мы осуществляем вызов ее же самой в качестве вспомогательной.

Как вычислить значения $x1$, $y1$? Они зависят от x и y следующим образом: $x1 = x + dx$, $y1 = y + dy$. Необходимо выразить значения приращений координат dx, dy. Воспользуемся определениями тригонометрических функций sin и cos:

$dx = r1 \cdot \cos(\alpha)$
$dy = r1 \cdot \sin(\alpha)$, где $\alpha = \pi/2, \pi, 3\pi/2, 2\pi$.

Если осуществить вызов этой же процедуры из основной программы с начальными параметрами, то рекурсивные вызовы никогда не закончатся и программа будет работать бесконечно. Для того, чтобы этого не случилось, можно ввести в качестве аргумента процедуры некоторую величину n, которая при каждом новом вызове процедуры будет уменьшаться на 1, а в тело процедуры включить условие, что его операторы должны выполняться только при $n > 0$, то есть это условие будет играть роль своеобразной «заглушки», ограничивающей число вызовов процедуры.

Ниже приведена программа, полностью решающая поставленную задачу.

В основной программе запрашиваются радиус r самой большой окружности и число уровней n, а также задаются значения коэффициентов $k1$ и $k2$. Центр самой большой окружности располагается в центре экрана.

```
Program Example_65;
Uses Graph;
Var x,y,n,r,r1,cd,gm:Integer;
k1,k2:Real;
Procedure Picture2(x,y,r,r1,n:Integer);
Var x1,y1:Integer;
i:Integer;
Begin
   If n>0 Then {"заглушка"}
   Begin
      circle(x,y,r);{рисование окружности}
      r1:=trunc(r*k2);  {вычисление радиуса орбиты}
      For i:=1 To 4 Do
      Begin
         x1:=trunc(x+r1*cos(pi/2*i)); y1:=trunc(y+r1*sin(pi/2*i));
         {координаты центра i-й окружности}
         picture2(x1,y1,trunc(r*k1),r1,n-1);
         {вызов процедуры с новыми параметрами}
      End;
   End;
End;
```

```
Begin {задание начальных значений}
   Writeln('Введите число уровней n.');
   Readln(n); x:=600 Div 2; y:=400 Div 2;
   Writeln('Введите радиус первой окружности r'); Readln(r);
   k1:=0.3; k2:=2;
   cd:=detect; gm:=1;    {инициализация графики}
   initgraph(cd, gm, 'c:\tp7_0\bin'); picture2(x,y,r,r1,n);
   Readln;    {задержка на экране}
   closegraph;
End.
```

Примечания:

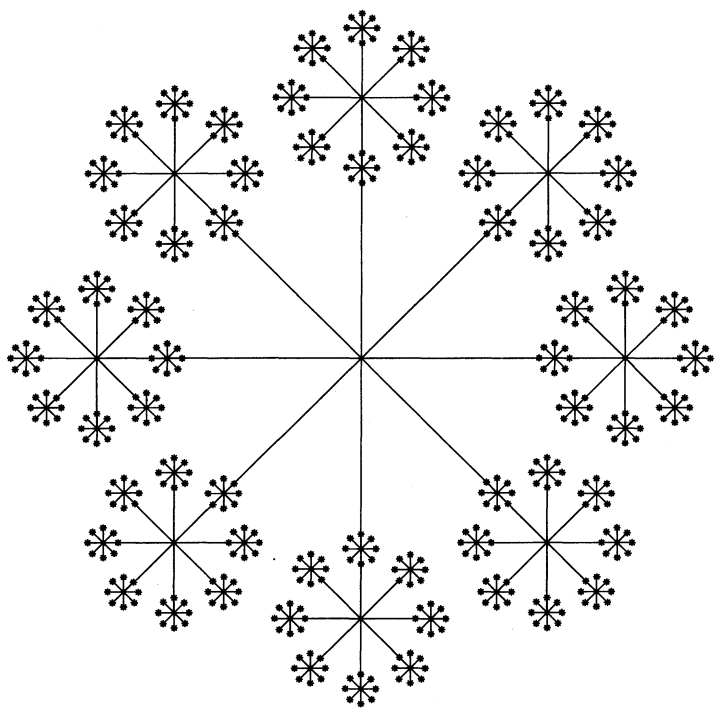
1. Для того чтобы проследить последовательность рисования окружностей при рекурсивных вызовах, можно дополнительно включить в тело процедуры простой цикл, обеспечивающий некоторую паузу перед выводом на экран очередной окружности.

2. Изменяя в процессе демонстрации работы программы значения $k1$, $k2$, n, можно получать множество привлекательных рисунков.

Пример 66. *Снежинка. Разработать программу, которая обеспечивает рисование снежинки, число звеньев и число ветвей которой задаются пользователем.*

На рисунке ниже изображена снежинка, состоящая из трех звеньев и восьми ветвей.

Сделаем расчет длины каждого звена по известному значению n — количеству звеньев и величине экрана. Будем считать, что снежинка строится в квадрате 400×400 точек. Центр снежинки совпадает с центром квадрата, а длина отрезка каждого



очередного звена в четыре раза меньше предыдущего. Если через l обозначить длину первого звена, то справедливо следующее равенство:

$$l + \frac{l}{4} + \frac{l}{4^2} = 200$$

или

$$l \times \frac{1 - \frac{1}{4^n}}{1 - \frac{1}{4}} = 200 \text{ (сумма членов геометрической прогрессии),}$$

то есть

$$l = 200 \times 3 \times \frac{4^{n-1}}{4^n - 1}.$$

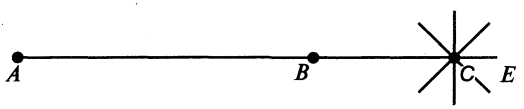
Пусть снежинка состоит из одного звена и p ветвей, тогда соответствующая программа проста. Основная ее часть имеет вид

```
For i:=1 To p Do
Begin
   x1:=trunc(x+l*cos(2*pi*(i-1)/p));
   y1:=trunc(y+l*sin(2*pi*(i-1)/p));
   {координаты конца очередного звена}
   line(x,y,x1,y1);  {рисование звена}
End;
```

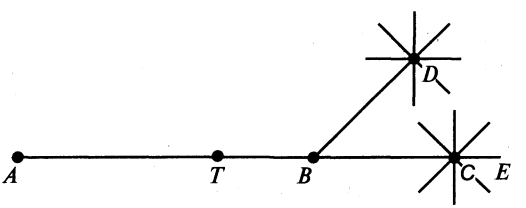
Здесь x, y — координаты точки центра снежинки. Наша снежинка рисуется как бы много раз, при этом длины звеньев изменяются, поэтому их надо просчитать один раз и запомнить в массиве L из n элементов. Длина каждого звена уменьшается в четыре раза по отношению к предыдущему. Длина первого звена определяется из того, что в квадрате из 400×400 точек необходимо построить снежинку из n звеньев. Если мы дошли до этапа рисования самого маленького звена (самой маленькой снежинки), то наша программа должна работать как приведенный набросок.

Итак, логика решения задачи.

Начинаем рисовать из центра, точки A, нарисовали первый отрезок AB (*а*), если это не последнее звено, то будем рисовать отрезок следующего звена BC, звено не последнее, поэтому продолжим. Предположим, что нарисовали CE, это первая ветвь самой маленькой снежинки, наша программа должна работать как набросок, то есть рисовать все ветви этой снежинки. После этого мы должны вер-



а



б

нуться в точку *B*. Так как это не последняя ветвь этой снежинки, то мы снова должны нарисовать следующую ветвь — отрезок *BD* (*б*), а затем снова полностью самую маленькую снежинку. Что необходимо для реализации этой логики? Пусть значение переменной *k* будет равно текущему номеру звена снежинки, в начальный момент оно равно *n*. Тогда при переходе от точек *C, D* к точке *B* мы должны «вспомнить» координаты точки *B* и номер ветви снежинки, рисуемой из точки *B*. При переходе от точки *B* к точке *A* мы должны «вспомнить» координаты точки *A* и номер ветви снежинки, рисуемой из точки *A*. Эта логика легко реализуется с помощью рекурсии.

Примечание. Для вычисления значения показательной функции a^n используется тождество $a^n = e^{n \ln a}$.

```
Program Example_66;
Uses Graph;
Const p=8; n=4;
Var x, y, i,cd, gm: Integer;
   L: Array[1..n] Of Integer;
Procedure Picture3(x, y, k: Integer);
Var x1, y1: Integer;
   i: Integer;
Begin
   If k>0 Then   {«заглушка»}
   Begin
      For i:=1 To p Do
      Begin  {координаты конца очередного звена}
         x1:=trunc(x+L[k]*cos(2*pi*(i-1)/p));
         y1:=trunc(y+L[k]*sin(2*pi*(i-1)/p));
         line(x,y,x1,y1);  {рисование звена}
         picture3(x1,y1,k-1);
      End;
   End;
End;
Begin
   L[n]:= trunc(200*3*exp((n-1)*ln(4))/(exp(n*ln(4))-1));{***}
   For i:=2 To n Do L[n-i+1]:=trunc(L[n]/exp((i-1)*ln(4)));
   x:=300; y:=200;   {координаты центра снежинки}
   cd:=detect; gm:=1;    {инициализация графики}
   Initgraph(cd,gm,'c:\tp7_0\bin');
   Picture3(x,y,n);
   Readln;    {задержка на экране}
   Closegraph;
End.
```

Задания

1. Модифицировать программу так, чтобы пересекающиеся части отрезков (*AB*, *BT*) рисовались один раз.

2. Если строку программы, помеченную {***}, изменить на

```
L[1]:=trunc(200*3*exp((n-1)*ln(4))/(exp(n*ln(4))-1));
```

то какие дальнейшие изменения необходимо внести в программу?

Упражнение № 2. Простые задачи

Пример 67. *Написать рекурсивную программу поиска минимального элемента массива.*

Решение. Опишем функцию p_min, которая определяет минимум среди первых n элементов массива a. Параметрами этой функции являются число элементов в рассматриваемой части массива n и значение последнего элемента этой части $a[n]$. При этом если $n > 2$, то результатом является минимальное из двух чисел — $a[n]$ и минимального числа из первых $(n - 1)$ элементов таблицы. В этом заключается рекурсивный вызов. Если же $n = 2$, то результатом является минимальное из первых двух элементов массива. Чтобы найти минимум всех элементов массива, надо обратиться к функции p_min, указав в качестве параметров значение размерности массива и значение последнего его элемента. Минимальное из двух чисел определяется с помощью функции min, параметрами которой являются эти числа.

```
Program Example_67;
   Const n=10;
   Type MyArray=Array[1..n] Of Integer;
   Const a:MyArray=(4,2,-1,5,2,9,4,8,5,3);
Function min (a,b:Integer):Integer;
Begin
   If a>b Then min:=b Else min:=a;
End;
Function p_min(n,b:Integer):Integer;
Begin
   If n=2 Then p_min:=min(n,a[1])
   Else p_min:=min(a[n],p_min(n-1,a[n]));
End;
Begin
   Writeln('Минимальный элемент массива -',p_min(n,a[n]));
End.
```

Пример 68. *Ханойские башни. Имеются три стержня A, B, C. На стержень A нанизано n дисков радиуса 1, 2, ..., n таким образом, что диск радиуса i является i-м сверху. Требуется переместить все диски на стержень B, сохраняя их порядок расположения (диск с большим радиусом находится ниже). За один раз можно перемещать только один диск с любого стержня на любой другой стержень. При этом должно выполняться следующее условие: на каждом стержне ни в какой момент времени никакой диск не может находиться выше диска с меньшим радиусом.*

Решение. Когда кольцо одно, никаких проблем нет, все действия очевидны.

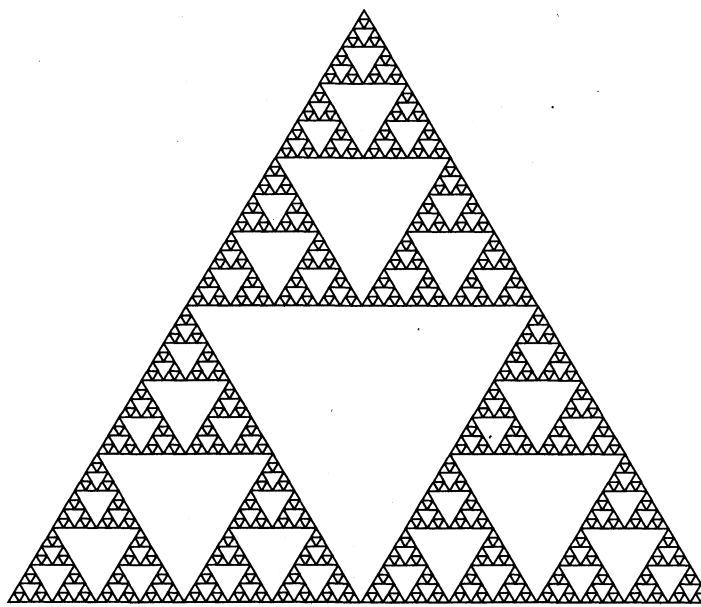
Предположим, что мы умеем перекладывать пирамиду из $(n-1)$ кольца. Рассмотрим пирамиду из n колец. Переместим первые $(n-1)$ колец на стержень C (это мы умеем). Затем перенесем последнее n-е кольцо со стержня A на стержень B. Далее перенесем пирамиду из $(n-1)$ кольца со стержня C на стержень B. Так как n-е кольцо самое большое, то условие задачи не будет нарушено. Таким образом, вся пирамида будет на стержне B. Действуя так, можно переложить любое число колец. При этом для решения задачи будет достаточно $2^n - 1$ перекладываний.

```
Program Example_68;
Const k=3;
Var a,b,c: Char;
Procedure Ring(n: Integer; a, b, c: Char);
```

```
Begin
   If n>0 Then
   Begin
      Ring(n-1,a,c,b); Writeln('кольцо',n,'c',a,'-->',b,'');
      Ring(n-1,c,b,a);
   End;
End;
Begin
   a:='A'; b:='B'; c:='C'; Ring(k,a,b,c); Readln
End.
```

Упражнение № 3. Фрактальные кривые

Пример 69. *Написать программу получения следующего изображения:*



В треугольнике проводятся все три средние линии. В результате он разбивается на 4 новых треугольника. К трем из них, примыкающим к вершинам первоначального треугольника, применяется та же процедура.

```
Program Example_69;
Uses Graph;
Var xc,xa,xb,ya,yb,yc:Integer;  {координаты вершин треугольника}
   n,cd,gm:Integer;
Procedure Triangle(xa, ya, xb, yb, xc, yc, n:Integer);
{координаты середин сторон треугольника}
Var xp, xq, xr, yp, yq, yr: Integer;
Begin
   If n>0 Then   {«заглушка»}
   Begin  {вычисление координат середин сторон треугольника}
      xp:=(xb+xc) Div 2;
```

```
        yp:=(yb+yc) Div 2; xq:=(xa+xc) Div 2; yq:=(ya+yc) Div 2;
        xr:=(xb+xa) Div 2; yr:=(yb+ya) Div 2;
        line(xp,yp,xq,yq);
        {изображение средних линий треугольника}
        line(xq,yq,xr,yr);
        line(xp,yp,xr,yr);
        Triangle(xa,ya,xr,yr,xq,yq,n-1);
        Ttriangle(xb,yb,xp,yp,xr,yr,n-1);
        Triangle(xc,yc,xq,yq,xp,yp,n-1);
    End;
End;
Begin {задание начальных значений}
    {координаты вершин самого большого треугольника}
    xc:=300; yc:=0; xb:=600; yb:=400; xa:=0; ya:=400;
    cd:=detect; gm:=1;
    initgraph(cd,gm,'c:\tp7_0\Bin');
    {изображение первого самого большого треугольника}
    line(xa,ya,xb,yb); line(xb,yb,xc,yc);
    line(xa,ya,xc,yc);
    Triangle(xa,ya,xb,yb,xc,yc,6);
    Readln;    {задержка на экране}
    Closegraph;
End.
```

Задания для самостоятельного выполнения

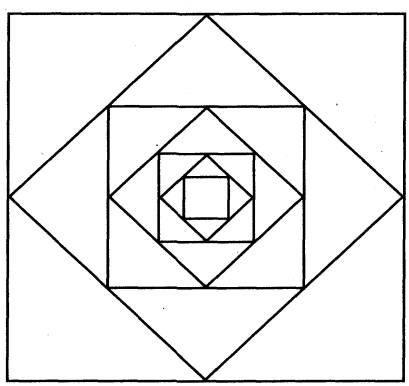
1. Написать рекурсивную программу вычисления *n*-го члена геометрической прогрессии, суммы ее *n* первых членов и суммы ее членов, начиная с *i*-го по *k*-й.

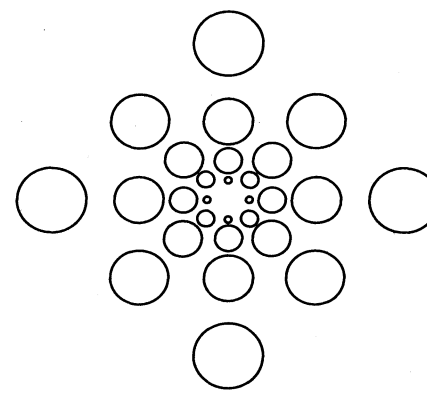
2. Описать рекурсивную функцию вычисления максимального числа Фибоначи, ближайшего к заданному *n* по недостатку.

3. Описать рекурсивную функцию поиска индекса минимального элемента массива.

4. Составить рекурсивную программу, проверяющую, является ли палиндромом фрагмент строки с *i*-го по *j*-й символ.

5. Составить программы для построения на экране следующих изображений:

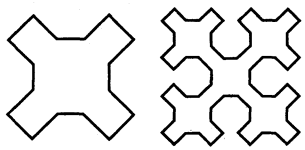




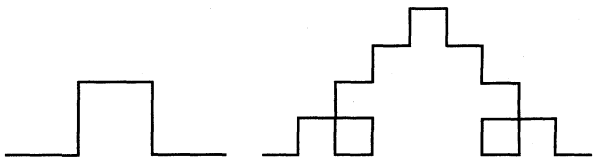
6. Изменить программу рисования кривых Гильберта так, чтобы обеспечить рисование кривых $C[1]$, $C[2]$, $C[3]$, $C[4]$, $C[5]$ (см. базовый учебник «Информатика»).

7. Изменить программу рисования кривых Гильберта так, чтобы обеспечить рисование кривых $A[3]$, $B[3]$, $C[3]$, $D[3]$.

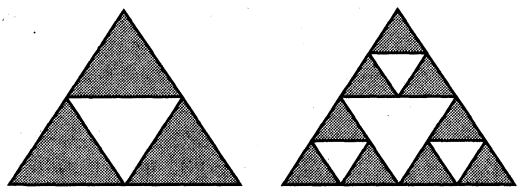
8. На рисунке изображены кривые Серпинского 1 и 2-го порядков. Составить программу построения кривых 1, 2, 3, 4 и 5-го порядков, так чтобы центры этих кривых совпадали.



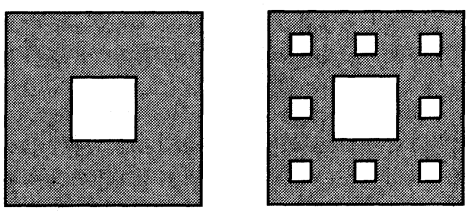
9. На рисунке изображены кривые Коха 1 и 2-го порядка. Составить программу построения кривой N-го порядка.



10. На рисунке изображены прокладки Серпинского 1 и 2-го уровня. Разработать программу построения прокладки Серпинского N-го порядка.



11. На рисунке изображен ковер Серпинского 1 и 2-го уровня. Разработать программу построения ковра Серпинского N-го порядка.



Упражнение № 4. Перебор с возвратом

Многие комбинаторные задачи можно представить как поиск элементов конечного множества, удовлетворяющих определенным условиям. Очевидно, что всегда есть «лобовое» решение, состоящее в переборе всех элементов этого множества с

проверкой соответствующих условий. Однако даже для простых задач такой перебор потребует огромных временных затрат.

Общая схема

Любая комбинаторная задача, к которой применим алгоритм перебора с возвратом, может быть описана в общем случае следующим образом: даны n линейно упорядоченных множеств $U_1, U_2, ..., U_n$ и требуется построить вектор $A = (a_1, a_2, ..., a_n)$, где $a_1 \in U_1$, $a_2 \in U_2, ..., a_n \in U_n$, удовлетворяющий заданному множеству условий и ограничений.

В алгоритме перебора с возвратом вектор A строится покомпонентно слева направо. Предположим, что уже найдены значения первых $k-1$ компонент,

$$A = (a_1, a_2, ..., a_{k-1}, ?, ..., ?),$$

тогда заданное множество условий ограничивает выбор следующей компоненты a_k некоторым множеством $S_k \subset U_k$. Если S_k оказалось непустым, мы вправе выбрать в качестве a_k наименьший элемент S_k и перейти к рассмотрению S_{k+1} и так далее. Однако, если условия таковы, что S_k оказалось пустым, мы возвращаемся к предыдущему этапу, выбрасываем $(k-1)$-й элемент a_{k-1} и выбираем в качестве нового a_{k-1} тот элемент S_{k-1}, который непосредственно следует за только что отброшенным. Может оказаться, что для нового a_{k-1} условия задачи допускают непустое S_k, и тогда мы попытаемся снова выбрать элемент a_k. Если невозможно выбрать a_{k-1}, то возвращаемся еще на шаг назад и выбираем новый элемент a_{k-2} и так далее.

Алгоритм

Шаг 1. [Начало] Положить $k = 1$ и $S_1 = U_1$.

Шаг 2. [Следующий элемент S_k] Если S_k пусто, то перейти к шагу 5, иначе — положить a_k равным наименьшему из элементов S_k.

Шаг 3. [Закончено ли решение?] Если $k < n$, то перейти к шагу 4, иначе — записать $(a_1, a_2, ..., a_n)$ как решение. Если необходимо найти все возможные решения, то положить $k = k + 1$ и перейти к шагу 5, иначе — остановиться.

Шаг 4. [Увеличение k] Увеличить k на 1, вычислить S_k и вернуться к шагу 2.

Шаг 5. [Возврат] Если $k = 1$, то дальнейшее продвижение назад невозможно; поэтому — остановиться; при этом либо все решения найдены, либо их не существует. Если $k > 1$, то уменьшить k на 1, положить $S_k = S_k \backslash \{a_k\}$ и вернуться к шагу 2.

Пример 70. *Задача о шахматном коне. Существуют способы обойти шахматным конем шахматную доску, побывав на каждом поле по одному разу. Составить программу нахождения всех возможных способов обхода доски.*

Решение

Структуры данных. Для хранения «истории» прохождения конем доски необходимо знать, был на конкретном поле[i, j] конь или нет.

Из поля [i, j] конь может попасть (в максимальном варианте) на восемь полей. В массивах di[1..8], dj[1..8] будем хранить приращения к значениям координат, в сумме с которыми i, j дают возможные координаты нового хода коня.

Примечание. На рисунке символом ** обозначено положение коня, а цифрами 1, 2, ..., 8 — номера ходов, которые можно сделать из поля [3, 5].

Выполнив очередной ход, можно:

- попасть за пределы доски;
- попасть в поле, которое уже занято;
- попасть в поле, которое пока еще свободно.

Учитывая это, для доски из m горизонталей и n вертикалей вводится массив

	1	2	3	4	5	6	7	8	ri[k]	rj[k]	k
1									−2	1	1
2									−1	2	2
3					**				1	2	3
4									2	1	4
5									2	−1	5
6									1	−2	6
7									−1	−2	7
8									−2	−1	8

```
r: Array[-1.. m+2,-1.. n+2] Of Integer;,
```

где

$$r[i,j] = \begin{cases} 0, & \text{если поле } [i,j] \text{ конем не посещалось;} \\ -1, & \text{если поле } [i,j] \text{ находится за пределами доски;} \\ t, & \text{если поле } [i,j] \text{ посещалось конем в результате хода с номером } t. \end{cases}$$

Основная логика работы достаточно проста:

```
Begin
   If <клетка занята> Or <клетка за пределами доски>
   Then exit
   Else
   Begin
      <сделать ход>; <выбрать очередной ход>;
   End;
End;
```

Поиск варианта обхода начинается с левой верхней клетки доски, то есть с *r*[1, 1]. Выполнить ход можно только в том случае, если клетка свободна или если не осуществляется выход за пределы доски (*r*[*i*, *j*] = 0). Сделать ход конем означает изменить значение соответствующего элемента массива *r* на очередное значение *t* — *r*[*i*, *j*]:=*t*. Если очередная клетка занята или если ход делается за пределы доски, надо попытаться сделать из той же позиции другой ход— взять следующие значения из массивов *ri* и *rj*. Если все 8 вариантов ходов из текущего положения будут проверены и ни одно из них не приведет к выполнению хода, то есть это ситуация, когда конь не может продолжать обход доски. В этом случае мы должны вернуть один ход или серию ходов и попытаться найти новый вариант обхода. Для этого надо взять следующие значения приращений координат из массивов *ri* и *rj*, попытаться сделать новый ход и так далее. Но для того, чтобы осуществить «возврат», необходимо знать «историю» обхода. Завершается поиск обхода, если все клетки пройдены, то есть значение *t* равно $m \times n$ (тогда надо вывести массив на экран), или если конь не может выполнить очередной ход (на экран надо вывести сообщение о невозможности выполнения обхода).

Реализовать такую логику поиска обхода выгоднее всего с помощью рекурсии.

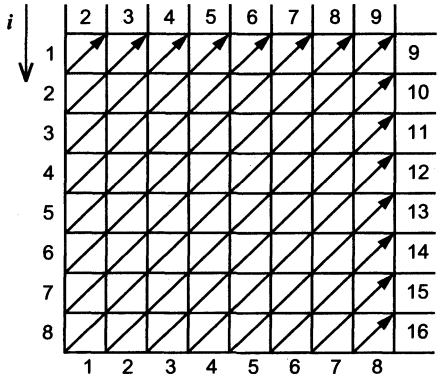
```
Program Example_70;
Const n=4; m=5;  {размеры доски}
q: Boolean=false;    {возможно ли сделать обход}
```

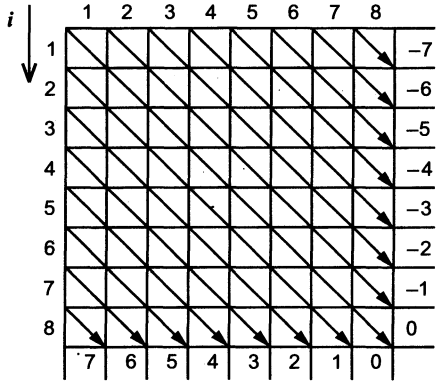
```
di: Array[1..8] Of Integer=(-2,-1,1,2,2,1,-1,-2);
dj: Array[1..8] Of Integer=(1,2,2,1,-1,-2,-2,-1);
Var a: Array[-1..n+2,-1..m+2] Of Integer;
i, j, l: Integer;
Procedure Print; {вывод двумерного массива на экран}
Begin
   For i:=1 To n Do
   Begin
      For j:=1 To m Do Write(a[i,j]:3);
      Writeln
   End;
End;

Procedure rec(i,j,t:Integer);    {процедура поиска обхода}
Var k:Integer;
Begin
   If (a[i,j]<>0) Then exit{выход за пределы доски или клетка
                            занята}
   Else
   Begin
      a[i,j]:=t; {сделать ход}
      If t=n*m Then {выполнен полный обход}
      Begin print; q:=true End
      Else For k:=1 To 8 Do rec(i+di[k],j+dj[k],t+1);
      {выбор нового хода}
      a[i,j]:=0; {возврат на один шаг назад}
   End;
End;
Begin  {заполнение массива}
   For i:=-1 To n+2 Do
   For j:=-1 To m+2 Do
   If (i<1) Or (i>n) Or (j<1) Or (j>m) Then a[i,j]:=-1;
   For i:=1 To n Do
   For j:=1 To m Do rec(i,j,1);
{выбор местоположения первого коня}
   If not q Then Writeln('Невозможно выполнить обход');
   Readln;
End.
```

Пример 71. *Задача о ферзях. Существуют способы расстановки 8 ферзей на шахматной доске 8× 8 так, чтобы они не били друг друга. Составить программу нахождения всех возможных способов.*

Решение. Структуры данных. Из шахмат известно, что ферзь, установленный на поле $[i,j]$, бьет все фигуры, находящиеся на той же самой вертикали j, горизонтали i и диагоналях, для которых сумма $(i + j)$ и разность $(i - j)$ индексов постоянны. На рисунке показаны значения сумм и разностей индексов для восходящих от 2 до 16 и нисходящих от −7 до 7 диагоналей.

Рассмотрим нашу задачу для доски 4×4. Начинаем просмотр горизонтали 1 и пытаемся найти поле $[i,j]$ для постановки ферзя. Если мы установили ферзя на поле [1,1], то поля, отмеченные на рисунке «*», считаются занятыми («под боем»). Переходим на вторую горизонталь. Ферзя можно установить на поле [2,3], тогда





поля, отмеченные «#», также выбывают из рассмотрения. Переходим на третью горизонталь, на ней все поля «под боем», поэтому необходимо вернуться на вторую горизонталь и попытаться установить на ней ферзя по-другому, рис. *б*. Следующее поле [2,4]. Возвращаемся на третью горизонталь, поле [3,2] свободно. Устанавливаем ферзя и переходим на четвертую горизонталь. Установить на ней ферзя нельзя. Возвращаемся на третью горизонталь — больше допустимых полей нет, на вторую — на ней также нет и, наконец, на первую.

	1	2	3	4
1	1	*	*	*
2	*	*	2	#
3	*	#	*	#
4	*		#	*

а

1	2	3	4
1	*	*	*
*	*	#	2
*	3	*	#
*	#	@	*

б

1	2	3	4
*	1	*	*
*	*	*	2
3	*	#	*
@	*	4	#

в

Устанавливаем ферзя на следующее поле [1,2]. Поля, отмеченные «*», — «под боем», рис. *в*, второго ферзя — на поле [2,4], поля «#» — «под боем», третьего ферзя на поле [3,1], поля «@» — «под боем». И наконец, четвертого ферзя можно поставить на поле [4,3].

Итак, логика основной части программы, текст которой приведен ниже, имеет следующий вид:

```
Program Example_71;
Const n=8;
k:Integer=0; {счетчик проверок безопасности полей}
Var x:Array [1..n] Of Integer;          {решение}
   a:Array [1..n] Of Boolean;    {вертикаль}
   b:Array [2..2*n] Of Boolean; {восходящая диагональ(/)}
   c:Array [-n+1..n-1] Of Boolean;    {нисходящая диагональ(\)}
Procedure Init;  {начальные значения}
Var i:Integer;
Begin
   FillChar(a,SizeOf(a),True);
   FillChar(b,SizeOf(b),True); FillChar(c,SizeOf(c),True);
```

```
End;
Procedure Move(i,j:Integer);     {процедура «фиксируй ход»}
Begin
   x[i]:=j; a[j]:=False; b[i+j]:=False; c[i-j]:=False;
End;
Procedure Back_Move(i,j:Integer);     {процедура «отмена хода»}
Begin
   a[j]:=true; b[i+j]:=true; c[i-j]:=true;
End;
Function D_Hod(i,j:Integer):Boolean; {функция «возможен ли ход»}
Begin
   D_Hod:=a[j] And b[i+j] And c[i-j]
End;
Procedure Print; {вывод решения}
Var i,j :Integer;
Begin
   For i:=1 to n Do
   Begin
      For j:=1 To N Do
      If j=x[i] Then Write('i':2) Else Write('*':2);
      Writeln(x[i]:3);
   End;
   WriteLn(k);
End;
Procedure Solve(i:Integer);{нахождение решения}
Var j:Integer;
Begin
   If i=n+1 Then Begin Inc(k); Print End
   Else
   For j:=1 To n do
   If D_Hod(i,j) Then
   Begin
      Move(i,j); Solve(i+1); Back_Move(i,j);
   End;
End;
Begin {основная программа}
   Init; Solve(1); ReadLn
End.
```

Пример 72. *Задача о лабиринте. Прямоугольное клеточное поле ограничено препятствиями. Кроме того, на поле задается произвольная система препятствий, начальная клетка, на которой находится Черепашка, и конечная клетка. Найти маршрут выхода из лабиринта, если он существует, и пронумеровать клетки маршрута в том порядке, в котором проходит их Черепашка. Черепашка может делать шаг на одну клетку в любом из четырех направлений: влево, вправо, вверх, вниз.*

Решение. Структуры данных. Клеточное поле размером $N \times M$ опишем двумерным массивом A размерности $N + 2$, $M + 2$. При этом будем считать, что

$A[i, j] = 0$, если клетка $[i, j]$ свободна и не посещалась Черепашкой;

$A[i, j] = -1$, если клетка $[i, j]$ является клеткой, на которой находится препятствие, или находится за пределами лабиринта.

	0	1	2	3	4	5	6
0	−1	−1	−1	−1	−1	−1	−1
1	−1	0	−1	0	−1	0	−1
2	−1	0	−1	0	−1	−1	−1
3	−1	0	−1	0	0	−1	−1
4	−1	0	−1	0	−1	−1	−1
5	−1	0	−1	0	0	−1	−1
6	−1	0	−1	−1	0	−1	−1

Координаты входа — $i1, j1$, координаты выхода — $i2, j2$. Они задаются пользователем. Блок задания исходных данных в программе и вывод массива Z достаточно просты, поэтому остановимся на разборе основной логики. Черепашка, перемещаясь по свободным клеткам ($A[i, j] = 0$), должна найти клетку выхода ($A[i2, j2] = 0$), запоминая при этом свой путь, и после нахождения отметить свой путь числами, то есть в каждую клетку пути записать номер шага, на котором она проходит эту клетку.

Предположим, что Черепашка находится в клетке $a[i, j]$. Из этой клетки она может сделать либо шаг на север, либо шаг на восток, либо шаг на запад, либо шаг на юг, если соответствующее направление не закрыто препятствием. Пусть Черепашка осуществляет поиск свободного поля для выполнения хода в следующем порядке:

↑ — соответствует уменьшению координаты i (Dec(i));
← — соответствует уменьшению координаты j (Dec(j));
→ — соответствует увеличению координаты j (Inc(j));
↓ — соответствует увеличению координаты i (Inc(i)).

Рассмотрим процесс поиска прохода на примере. Пусть в представленном выше лабиринте Черепашка должна найти проход из клетки с координатами $i1 = 1, j1 = 3$ в клетку с координатами $i2 = 6, j2 = 1$. Первый шаг делается в начальную клетку, то есть $a[1, 3]$:=1. Из начальной клетки она может сделать шаг только в направлении 4 (inc(i)), так как в направлении 1, 2 произойдет выход за пределы лабиринта, в направлении 3 Черепашка наткнется на препятствие. Следовательно, надо присвоить $a[3, 2]$ значение 2 (шаг сделан). Далее снова производится анализ направлений хода в том же порядке: направление 1 отбрасывается, так как приводит к возврату, направления 2, 3 тоже отбрасываются, так как там препятствия. Черепашка должна сделать ход вновь в четвертом направлении: $a[3, 3] := 3$. Четвертый шаг можно сделать в направлении 3 и 4. Пусть Черепашка всегда выбирает первый из всех допустимых вариантов, в данном случае 3 (→): $a[3, 4] := 4$. Это тупик — с трех сторон клетка с координатами 4, 5 окружена препятствиями. Следовательно, надо вернуться на один шаг назад и попытаться по-другому сделать четвертый шаг. Из предыдущей клетки можно шагнуть еще и в четвертом направлении, то есть $a[4, 3]$:=4. Затем вновь делается шаг в четвертом направлении, единственно возможном ($a[5, 3]$:=5). И так далее до тех пор, пока Черепашка не доберется до конечной клетки.

Таким образом, при анализе вариантов очередного хода Черепашке необходимо проверить, свободна ли рассматриваемая клетка. В этом случае $a[i, j] = 0$. Ход делать нельзя в клетку, для которой $a[i, j] = -1$ (выход за границы лабиринта, в

клетке препятствие), или $a[i, j] > 0$ (ход назад). При выполнении l-го хода в клетку с координатами i, j надо $a[i, j]$ присвоить значение l.

Если Черепашка зашла в тупик (в трех направлениях препятствия или границы лабиринта), то она должна попытаться сделать новый вариант прохода. Для этого ей надо вернуться на один шаг назад и проверить оставшиеся варианты из предыдущей позиции. При этом необходимо «занулить» значение $a[i, j]$.

Выбор варианта хода осуществляется из четырех возможных вариантов. В массивах b и c, содержащих по 4 элемента, будем хранить приращения к значениям текущих координат, в сумме с которыми они дают возможные координаты нового хода Черепашки:

```
b:Array[1..4] Of Integer=(-1,0,0,1);
c:Array[1..4] Of Integer=(0,-1,1,0);
```

Тогда выбор хода и проверку его возможности можно организовать с помощью цикла

```
For k:=1 To 4 Do
If a[i+b[k],j+c[k]]=0 Then...
```

Опишем этот процесс в виде рекурсивной процедуры *Move(i, j, l: Integer)*, входными параметрами которой являются значения текущих координат Черепашки i, j и l — номер ее очередного шага. Если текущие значения координат совпадут с заданными конечными значениями $i2, j2$, то необходимо вывести лабиринт с отмеченным проходом Черепашки и закончить работу. Вывод осуществляется процедурой *Print(a:myarray)*.

Если прохода из заданной начальной клетки в заданную конечную клетку не существует, то необходимо вывести информацию об этом на экран. В этом случае, после того как Черепашка проверит все возможные способы прохода, и необходимый результат не будет получен, значение $a[i2, j2]$ по-прежнему будет равно 0.

```
Program Example_72;
Uses Crt;
Const n=5; m=6;  {размеры поля}
   b:Array[1..4] Of Integer=(-1,0,0,1);
   c:Array[1..4] Of Integer=(0,-1,1,0);
Type myarray=Array [0..n+1,0..m+1] Of Integer;
Var a: myarray;  {исходный лабиринт}
i,j,i1,i2,j1,j2: Integer;
Procedure Init;  {формирование массива a из файла}
Var i, j, t: Integer;
f: text;
Begin
   Assign(f,'c:\1.dat'); Reset(f);
   For i:=0 To n+1 Do For j:=0 To m+1 Do a[i,j]:=-1;
   For i:=1 To n Do For j:=1 To m Do Read(f,a[i,j]);
   Close(f);
End;
Procedure Print(a: myarray);     {печать лабиринта}
Var i, j: Integer;
Begin
   For i:=1 To n Do
      Begin
```

```
            For j:=1 To m Do
            Case a[i,j] Of
            -1: Write('*');
            0: Write('');
            Else Write(a[i,j]:3);
        End;
        Writeln;
    End;
End;
Procedure Move(i, j, l: Integer);
Var k:Integer;
Begin
    If (i=i2) And (j=j2) Then     {достигнута конечная клетка}
    Begin print(a); exit End
    Else
    Begin
    For k:=1 To 4 Do  {выбор направления хода}
    If a[i+b[k],j+c[k]]=0 Then    {шаг сделать можно}
        Begin
            a[i+b[k],j+c[k]]:=l;   {сделать шаг}
            hod(i+b[k],j+c[k],l+1);
            a[i+b[k],j+c[k]]:=0;   {возврат на 1 шаг назад}
        End;
    End;
End;
Begin
    Clrscr; Writeln('ДАННЫЙ ЛАБИРИНТ:');
    Writeln('_________________'); Init; Print(a);
    Writeln('ВВЕДИТЕ КООРДИНАТЫ ВХОДА'); Readln(i1,j1);
    Writeln('ВВЕДИТЕ КООРДИНАТЫ ВЫХОДА'); Readln(i2,j2);
    a[i1,j1]:=1;
    {первый шаг в начальную клетку с координатами i1, j1}
    Move(i1,j1,2);
    If a[i2,j2]=0 Then Writeln('ПРОХОД НЕВОЗМОЖЕН');
    Readln;
End.
```

Задания для самостоятельной работы

1. Какое наименьшее число ферзей можно расставить на доске так, чтобы держали под боем все ее свободные поля? Модификация задачи. Найти расстановку ферзей, которая одновременно решает задачу для досок 9×9, 10×10 и 11×11.

2. Расставить на доске $N\times N$ ($N\le12$) N ферзей так, чтобы наибольшее число ее полей оказалось вне боя ферзей.

3. Расставить на доске как можно больше ферзей так, чтобы при снятии любого из них появлялось ровно одно не атакованное поле.

4. За какое наименьшее число ходов ферзь может обойти все поля доски 8×8?

5. Расставить на доске 8×8 максимальное число ферзей так, чтобы каждый из них нападал ровно на p ($p\le2$) ферзей. (Ответ. При $p = 1$ десять, а при $p = 2$ четырнадцать.)

6. Задача о коне Аттилы («Трава не растет там, где ступил мой конь!»). На шахматной доске стоят белый конь и черный король. Некоторые поля доски считаются «горящими». Конь должен дойти до неприятельского короля, повергнуть его и вернуться на исходное место. При этом ему запрещено становиться как на горящие поля, так и на поля, которые уже пройдены.

7. Магараджа — это фигура, которая объединяет в себе ходы коня и ферзя. Для доски 10×10 найти способ расстановки 10 мирных магараджей.

8. (Задача с международной олимпиады школьников по программированию 1991 г.). Пронумеровать позиции в матрице (таблице) размером 5×5 таким образом, чтобы номер i ($1 \le i \le 25$) соответствовал позиции с координатами (x, y), вычисляемыми по одному из следующих правил:

- $(z, w) = (x \pm 3, y)$;
- $(z, w) = (x, y \pm 3)$;
- $(z, w) = (x \pm 2, y \pm 2)$.

Требуется:

а) написать программу, которая последовательно нумерует позиции матрицы 5×5 при заданных координатах позиции, в которой поставлен номер 1 (результаты должны быть представлены в виде заполненной матрицы);

б) вычислить число всех возможных расстановок номеров для всех начальных позиций, расположенных в правом верхнем треугольнике матрицы, включая ее главную диагональ.

Краткие сведения

Сортировка и поиск

Для решения многих задач удобно сначала упорядочить данные по определенному признаку. Процесс упорядочения заданного множества объектов по заданному признаку называется ***сортировкой.***

Сортировка методом простого выбора

Эта сортировка обычно применяется для массивов, не содержащих повторяющихся элементов. Для достижения поставленной цели можно действовать следующим образом:

1) выбрать максимальный элемент массива;

2) поменять его местами с последним элементом (после этого самый большой элемент будет стоять на своем месте);

3) повторить пп.1—2 с оставшимися $n-1$ элементами, то есть рассмотреть часть массива, начиная с первого элемента до предпоследнего, найти в ней максимальный элемент и поменять его местами с предпоследним $(n-1)$-м элементом, затем с оставшимися $n-2$ элементами и так далее, пока не останется один (наименьший) элемент, уже стоящий на своем месте.

Сортировка методом простого обмена

Сортировка методом простого обмена может быть применена для любого массива. Этот метод заключается в последовательных просмотрах массива сверху вниз (от начала к концу) и обмене местами **соседних** элементов, расположенных «не-

правильно», то есть таких, что $i<j$, а $a[i]>a[j]$. Опишем его подробнее. Просмотрим весь массив.

Начинаем с первой пары элементов ($a[1]$ и $a[2]$). Если первый элемент этой пары больше второго, то меняем их местами, иначе — оставляем без изменения. Затем берем вторую пару элементов ($a[2]$ и $a[3]$), если $a[1]>a[2]$, то меняем их местами и так далее. На первом шаге будут просмотрены все пары элементов массива $a[i]$ и $a[i+1]$ для i от 1 до $n-1$. В результате максимальный элемент массива переместится в конец массива. Поскольку самый большой элемент находится на своем месте, рассмотрим часть массива без него, то есть с первого до $(n-1)$-го элемента.

Повторим предыдущие действия для этой части массива, в результате чего второй по величине элемент массива переместится на последнее место рассматриваемой части массива, то есть на $(n-1)$-е место во всем массиве. Эти действия продолжают до тех пор, пока число элементов в текущей части массива не уменьшится до двух. В этом случае необходимо выполнить последнее сравнение и упорядочить последние два элемента. При сортировке выполняется $n-1$ просмотр массива. После этого массив отсортирован.

Сортировка методом прямого включения

Сортировка этим методом производится последовательно шаг за шагом. На k-м шаге считается, что часть массива, содержащая первые $k-1$ элемент, уже упорядочена, то есть $a[1] \le a[2] \le \ldots \le a[k-1]$. Далее необходимо взять k-й элемент и подобрать для него место в отсортированной части массива такое, чтобы после его вставки упорядоченность не нарушилась, то есть надо найти такое j $(1 \le j \le k-1)$, что $a[j] \le a[k] < a[j+1]$. Затем надо вставить элемент $a[k]$ на найденное место.

С каждым шагом отсортированная часть массива увеличивается. Для выполнения полной сортировки потребуется выполнить $n-1$ шаг.

Сортировка методом слияний

Этот метод состоит в разбиении данного массива на несколько частей, которые сортируются по отдельности, и в последующем составлении из нескольких уже упорядоченных массивов искомого упорядоченного массива.

Пусть массив $a[1..n]$ разбивается на части длиной k, тогда первая часть — $a[1]$, $a[2]$, ..., $a[k]$, вторая — $a[k+1]$, $a[k+2]$, ..., $a[2k]$ и так далее. Если n не делится на k, то в последней части будет менее k элементов.

После того как массивы-части упорядочены, можно объединить их в упорядоченные массивы-части, состоящие не более, чем из $2k$ элементов, которые далее объединить в упорядоченные массивы длиной не более $4k$, и так далее, пока не получится один искомый массив.

Таким образом, чтобы получить отсортированный массив этим методом, надо многократно «сливать» два упорядоченных отрезка массива в один упорядоченный отрезок. При этом другие части массива не затрагиваются.

Обменная сортировка с разделением (сортировка Хоара)

Сортировка методом простого обмена (методом «пузырька») часто является самой неэффективной. Это обусловлено самой идеей метода, которая требует в

процессе сортировки сравнивать и обменивать между собой только соседние элементы. Можно существенно улучшить метод сортировки, основанный на обмене. Это улучшение приводит к самому лучшему на сегодняшний день методу сортировки массивов, который можно назвать обменной сортировкой с разделением. Он основан на сравнениях и обменах элементов, стоящих на возможно больших расстояниях друг от друга. Предложил этот метод Ч. Э. Р. Хоар в 1962 г. Поскольку производительность этого метода просто впечатляюща, автор назвал его «быстрой сортировкой».

Идея метода:

1. В исходном неотсортированном массиве выбрать некоторый элемент $x = a[k]$ (барьерный элемент).

2. Переставить элементы массива таким образом, чтобы слева от x оказались элементы массива, меньшие или равные x, а справа — элементы массива, большие x.

Пусть при этом элемент x попадет в позицию с номером k, тогда массив будет иметь вид

$(a[1], a[2], ..., a[k-1]), a[k], (a[k+1], ..., a[n])$.

Каждый из элементов $a[1], a[2], ..., a[k-1]$ меньше либо равен $a[k]$, каждый из элементов $a[k+1],..., a[n]$ больше $a[k]$. Отсюда можно сделать вывод, что элемент $a[k]$ стоит на своем месте. А исходный массив при этом разделится на две неотсортированные части, барьером между которыми является элемент $a[k]$.

Для дальнейшей сортировки необходимо применить пп.1, 2 для каждой из этих частей. И так до тех пор, пока не останутся подмассивы, состоящие из одного элемента, то есть пока не будет отсортирован весь массив.

Одной из целей сортировки является облегчение последующего ***поиска*** элементов в отсортированном множестве. Результатом поиска служит элемент множества, равный эталону, или отсутствие такового.

Линейный поиск

Для решения задачи разумно применить очевидный метод — последовательный просмотр массива и сравнение значения очередного рассматриваемого элемента с эталоном x. При их совпадении запоминаем номер элемента. Этот линейный способ решения поставленной задачи приводит к ответу на поставленный вопрос, но обладает рядом существенных недостатков:

- если значение x встречается в массиве несколько раз, то найдено будет последнее из них;
- после того, как нужное значение уже найдено, массив просматривается до конца, то есть всегда выполняется n сравнений.

Линейный поиск с использованием барьера

В массив на $n + 1$ место запишем искомый элемент x, который будет являться барьерным. Тогда если в процессе работы программы обнаружится такой индекс i, что $a[i] = x$, то элемент будет найден, а если $a[i] = x$ будет только при $i = n + 1$, то, значит, интересующего нас элемента в массиве нет.

При таком способе поиска в случае наличия в массиве нескольких элементов, удовлетворяющих заданному свойству, будет также найден элемент с наименьшим номером.

Поиск делением пополам

Сравнение эталона x осуществляется с элементом, расположенным в середине массива и в зависимости от результата сравнения (больше или меньше) дальнейший поиск проводится в левой или правой половине массива.

Контрольные вопросы

1. С какой целью сортируют массивы данных?
2. Возьмите игральные карты и произвольно выберите 10 карт. Отсортируйте карты в руках по старшинству (тузы, короли и т.д., старшинство по мастям — «червы», «бубны», «трефы», «пики») методом простого выбора.
3. Отсортируйте карты в руках по старшинству методом простого обмена.
4. Отсортируйте карты в руках по старшинству методом прямого включения.
5. Отсортируйте карты в руках по старшинству методом слияний.
6. Отсортируйте карты в руках по старшинству методом быстрой сортировки.
7. Подсчитайте число сравнений в методе поиска делением пополам.

Задачи и упражнения

Упражнение № 5. Сортировка методом простого выбора

Пусть исходный массив a состоит из 10 элементов и имеет вид

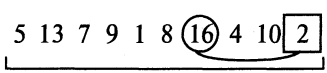
5 13 7 9 1 8 16 4 10 2

После сортировки массив должен выглядеть так:

1 2 4 5 7 8 9 10 13 16

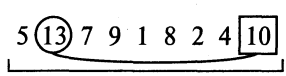
Процесс сортировки представлен ниже. Максимальный элемент текущей части массива заключен в кружок, а элемент, с которым происходит обмен, в квадратик. Скобкой помечена рассматриваемая часть массива.

1-й шаг: рассмотрим весь массив и найдем в нем максимальный элемент — 16 (стоит на седьмом месте), поменяем его местами с последним элементом — с 2.



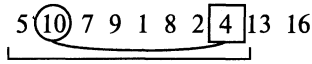
Максимальный элемент записан на свое место.

2-й шаг: рассмотрим часть массива — с первого до девятого элемента. Максимальный элемент этой части — 13, стоящий на втором месте. Поменяем его местами с последним элементом этой части — с 10.

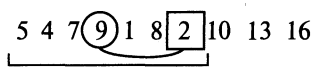


Отсортированная часть массива состоит теперь уже из двух элементов.

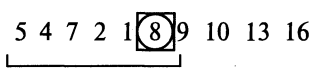
3-й шаг: снова уменьшим рассматриваемую часть массива на один элемент. Здесь надо поменять местами второй элемент (его значение — 10) и последний элемент этой части — 4.



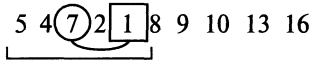
В отсортированной части массива — 3 элемента.
4-й шаг: аналогично.



5-й шаг: максимальный элемент этой части массива является последним в ней, поэтому его надо оставить на старом месте.



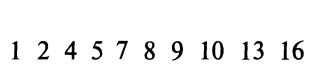
Далее действуем аналогично.
6-й шаг:



7-й шаг:

8-й шаг:

9-й шаг:



Итог:

1 2 4 5 7 8 9 10 13 16

Для программной реализации этого процесса необходимо организовать цикл по *i* — длине рассматриваемой части массива, которая изменяется от *n* до 2. В качестве начального значения максимума разумно взять значение последнего элемента рассматриваемой части.

Теперь можем записать алгоритм сортировки:

```
For i:=n Downto 2 Do
Begin
   найти максимальный элемент из a[1],..., a[i];
   запомнить его индекс в переменной k;
   если i<>k поменять местами a[i] и a[k]
End;
```

Опишем этот алгоритм подробно в виде процедуры:

```
Procedure sorting1(Var a:ar);
{поскольку в процессе работы процедуры массив изменится, формальный параметр a описывается как параметр-переменная}
Var i, j, k: Integer;
m: Integer;
{значение максимального элемента рассматриваемой части массива}
Begin
   For i:=10 Downto 2 Do
   {цикл по длине рассматриваемой части массива}
   Begin {поиск максимального элемента и его номера
      в текущей части массива}
      k:=i; m:=a[i]; {начальные значения макс. элемента
      и его индекса в рассматриваемой части массива}
      For j:=2 To i-1 Do
      If a[j]>m Then Begin k:=j; m:=a[k] End;
      If k<>i Then
      Begin  {перестановка элементов}
         a[k]:=a[i]; a[i]:=m
      End
   End
End.
```

Упражнение № 6. Сортировка методом простого обмена

Отсортируем по возрастанию методом простого обмена массив из пяти элементов: 5 1 8 4 9.

Длина текущей части массива — $n-k+1$, где k — номер просмотра, i — номер проверяемой пары, номер последней пары — $n-k$. За вертикальной чертой располагаются отсортированные элементы.

Первый просмотр: рассматривается весь массив.

$i=1$ 5 4 8 2 9
> меняем

$i=2$ 4 5 8 2 9
< не меняем

$i=3$ 4 5 8 2 9
> меняем

$i=4$ 4 5 2 8 9
< не меняем

Число 9 стоит на своем месте.

Второй просмотр: рассматриваем часть массива с первого до четвертого элемента.

$i=1$ 4 5 2 8 9
< не меняем

$i=2$ 4 5 2 8 9
> меняем

$i=3$ 4 2 5 8 9
< не меняем

Число 8 стоит на своем месте.

Третий просмотр: рассматриваемая часть массива содержит три первых элемента.

$i = 1$ 4 2 5 8 9
> меняем

$i = 2$ 2 4 5 8 9
< не меняем

Число 5 стоит на своем месте.

Четвертый просмотр: рассматриваем последнюю пару.

$i = 1$ 2 4 5 8 9
< не меняем

Число 4 стоит на своем месте. Для самого маленького элемента (2) остается только одно место — первое.

Итак, наш массив отсортирован по возрастанию элементов методом простого обмена. Этот метод называют также методом «пузырька».

Опишем процедуру «пузырьковой» сортировки.

```
Procedure sorting2(Var a:ar);
Var k,i,t:Integer;
{k - номер просмотра (изменяется от 1 до n-1),
i - номер рассматриваемой пары,
t - промежуточная переменная для перестановки местами элементов}
Begin
   For k:=1 To n-1 Do    {цикл по номеру просмотра}
   For i:=1 To n-k Do
   If a[i]>a[i+1] Then   {перестановка элементов}
   Begin
      t:=a[i]; a[i]:=a[i+1]; a[i+1]:=t
   End
End.
```

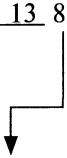
Упражнение № 7. Сортировка методом прямого включения

Рассмотрим этот процесс на примере. Пусть требуется отсортировать массив из 10 элементов по возрастанию методом прямого включения:

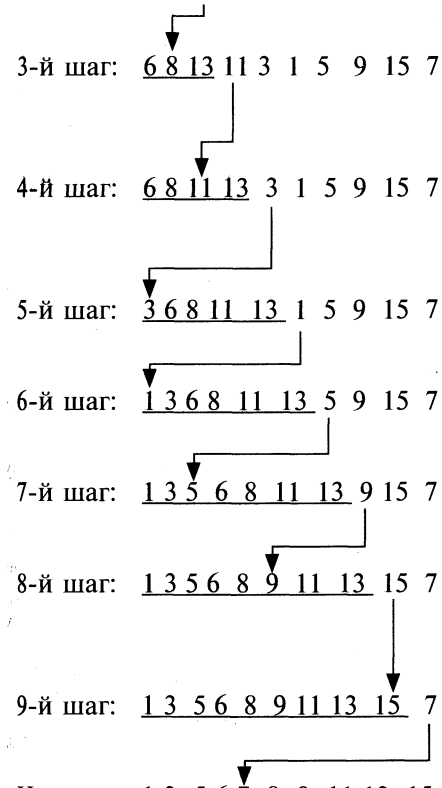
1-й шаг: 13 6 8 11 3 1 5 9 15 7

Рассматриваем часть массива из одного элемента (13). Надо вставить в нее второй элемент массива (6) так, чтобы упорядоченность сохранилась. Так как 6<13, вставляем 6 на первое место. Отсортированная часть массива содержит два элемента (6 13).

2-й шаг: 6 13 8 11 3 1 5 9 15 7



Возьмем третий элемент массива (8) и подберем для него место в упорядоченной части массива. 8>6 и 8<13, следовательно, его надо вставить на второе место.



Следующий элемент — 11. Он записывается в упорядоченную часть массива на третье место, так как 11>8, но 11<13.

Далее, действуя аналогичным образом, определяем, что 3 необходимо записать на первое место.

По той же причине 1 записываем на первое место.

Так как 5>3, но 5<6, то место 5 в упорядоченной части — третье.

Место числа 9 — шестое.

Определяем место для предпоследнего элемента 15. Оказывается, что этот элемент массива уже находится на своем месте.

Осталось подобрать подходящее место для последнего элемента (7).

Массив отсортирован полностью.

Сейчас можно коротко описать фрагмент алгоритма сортировки с помощью простого включения:

```
For k:=2 To n Do
{так как начинаем сортировку с поиска подходящего места для a[2],
i изменяется от 2 до n}
Begin
   x:=a[k];
   "вставить x на подходящее место в a[1],...,a[k]"
End;
```

Осталось ответить на вопрос, как осуществить поиск подходящего места для элемента x. Поступим следующим образом: будем просматривать элементы, расположенные левее x (то есть те, которые уже упорядочены), двигаясь к началу массива. Надо просматривать элементы $a[j]$, j изменяется от $k-1$ до 1.

Такой просмотр закончится при выполнении одного из следующих условий:

- найден элемент $a[j]<x$, что говорит о необходимости вставки x между $a[j-1]$ и $a[j]$;
- достигнут левый конец упорядоченной части массива, следовательно, надо вставить x на первое место.

До тех пор, пока одно из этих условий не выполнится, будем смещать просматриваемые элементы на 1 позицию вправо, в результате чего в отсортированной части будет освобождено место под x.

Учитывая это, опишем процедуру сортировки:

```
Procedure Sorting3(Var a:ar);
Var k,j,x:Integer;
Begin
   For k:=2 To n Do
   Begin
      x:=a[k]; j:=k-1;
      While (j>0) And (x>=a[j]) Do
      Begin
         a[j+1]:=a[j]; Dec(j)
      End;
      a[j+1]:=x
   End
End.
```

Упражнение № 8. Сортировка методом слияний

Пусть первая часть массива (часть *a*) состоит из пяти элементов:
... 3 5 8 11 16...,
а вторая часть (часть *b*) — из восьми элементов:
... 1 6 7 9 12 13 18 20...

По условию оба массива-части упорядочены. Надо сформировать массив *c*, который будет содержать 13 элементов.

Введем обозначения:

- i — номер обрабатываемого элемента части *a*;
- j — номер обрабатываемого элемента части *b*;
- l — номер заполняемого элемента массива *c*.

Поскольку заполнение массива *c* ведется с его начала, на первом шаге значение $l = 1$.

1-й шаг: на первое место в массиве *c* претендуют $a[k+1] = 3$ и $b[m+1] = 1$ ($i = k+1$, $j = m+1$). Так как $1 < 3$, в массив *c* надо занести 1 и перейти к следующему элементу части *b*, то есть увеличить j на 1 (значение j станет равно $m+2$), а также увеличить на 1 значение l (значением l будет 2).

2-й шаг: теперь надо сравнить $a[k+1] = 3$ и $b[m+2] = 5$. $3 < 5$, следовательно, на второе место в *c* надо занести $a[k+1] = 3$. Затем надо увеличить на 1 значения i и l.

3-й шаг: на третье место массива *c* претендуют два одинаковых элемента — $a[k+2] = 5$ и $b[m+2] = 5$. В этом и подобных случаях договоримся заносить в *c* первым элемент из части *a*. Таким образом, $c[3] = a[k+2]$, а значение $l = 4$, $i = k+3$, j остается равным $m+2$.

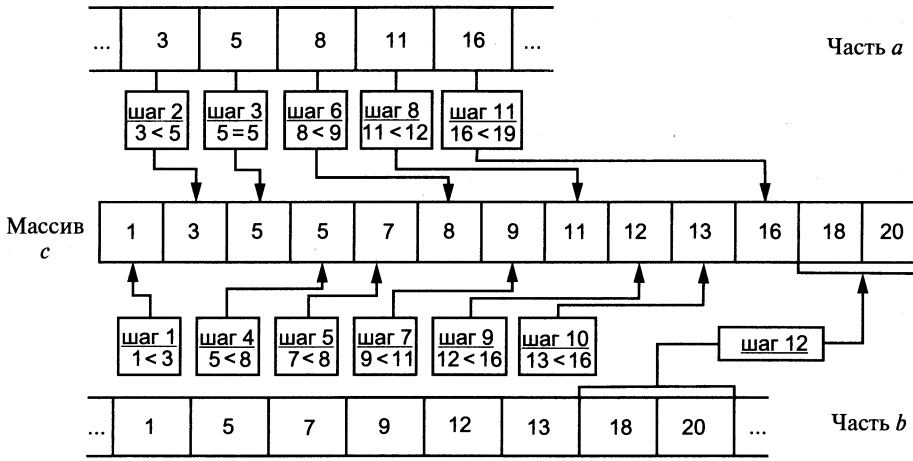
4—11-й шаги: рассуждая аналогичным образом, надо занести в *c* элементы 6, 7, 8, 9, 11, 12, 13, 16. После выполнения 11-го шага первая часть будет занесена в *c* полностью, значение j равно $m+7$, значение l равно 12.

12-й шаг: так как часть *a* закончилась, а «хвост» части *b* упорядочен по условию, то двенадцатым элементом массива *c* будет первый элемент «хвоста» *b*, то есть $b[m+7] = 18$.

13-й шаг: последним элементом массива *c* будет последний элемент *b* — 20.

На рисунке схематично изображен процесс заполнения массива *c*.

Сейчас можем описать процедуру слияния двух упорядоченных массивов размерностей m и n в третий упорядоченный массив, размерность которого p ($p = n + m$).




```
Procedure sorting4(k,m,n:Integer);
Begin
   i:=k+1;
   j:=m+1;
   k:=1;
   While (i<=m) And (j<=n) Do
   {пока не закончилась хотя бы одна часть}
   Begin
      If a[i]<=b[j] Then Begin c[k]:=a[i]; Inc(i); End
      Else Begin c[k]:=b[j]; Inc(j); End;
      Inc(k);
   End; {один из массивов-частей обработан полностью, осталось
перенести в 'c' остаток другого массива-части}
   While i<=m Do Begin c[k]:=a[i]; Inc(i); Inc(k) End;
   While j<=n Do Begin c[k]:=b[j]; Inc(j); Inc(k) End
End.
```

Далее остается лишь переписать результат слияния рассматриваемых частей — массив *c* — обратно в массив *a*.

Упражнение № 9. Обменная сортировка с разделением (сортировка Хоара)

Пример 73. *Составить программу «быстрой сортировки».*
Решение. Пусть исходный массив состоит из восьми элементов:

8 12 3 7 19 11 4 16

В качестве барьерного элемента будем брать средний элемент массива. Барьерный элемент — 7. Произведя необходимые перестановки для разделения, получим:

(4 3) 7 (12 19 11 8 16)

Число 7 стоит на своем месте. Далее сортируем подмассивы, элементы которых заключены в скобки. Этот процесс будем повторять до тех пор, пока не получим полностью отсортированный массив.

Левый подмассив:

```
(3)  4   7  (12  19  11   8  16)
 3   4   7  (12  19  11   8  16)
```

Правый подмассив:

```
 3   4   7  (8)  11  (19  12  16)
 3   4   7   8   11  (19  12  16)
 3   4   7   8   11   12  (19  16)
 3   4   7   8   11   12  (16)  19
 3   4   7   8   11   12   16   19
```

Массив отсортирован полностью.

Алгоритм «быстрой» сортировки можно определить как рекурсивную процедуру, параметрами которой являются нижняя и верхняя границы изменения индексов сортируемой части исходного массива.

```
Program Example_73;
Var a: Array[1..10] Of Integer;
Procedure Init;  {формирование массива из файла}
Var f: text;
i: Integer;
Begin
   Assign(f,'c:\s.dat'); Reset(f);
   For i:=1 To 10 Do Read(f,a[i]);
End;

Procedure Print; {печать массива}
Var i: Integer;
Begin
   For i:=1 To 10 Do Write(a[i]:5);
   Writeln
End;

Procedure Quick_sorting(m,l: Integer);
Var i,j,x,w: Integer;
Begin
   i:=m; j:=l; x:=a[(m+l) Div 2];
   Repeat
   While a[i]<x Do Inc(i);
   While a[j]>x Do dec(j);
   If i<j Then
   Begin
      w:=a[i]; a[i]:=a[j]; a[j]:=w; Inc(i); dec(j);
   End
   Until i>j;
   If m<j Then quick_sorting(m,j);
   If i<l Then quick_sorting(i,l);
End;
Begin {основная программа}
   Writeln('массив:'); init; print; quick_sorting(1,10);
   Writeln('отсортированный массив'); print;
End.
```

Упражнение № 10. Линейный поиск заданного элемента в массиве

Пример 74. *Написать программу поиска элемента x в массиве из n элементов. Значение элемента x вводится с клавиатуры.*

Решение. Структуры данных:

```
Const n=10;   {размерность массива}
Var a: Array[1..n] of Integer;   {массив из n элементов целого типа}
x:Integer;    {искомый элемент целого типа}
```

Для решения задачи можно использовать цикл

```
For i:=1 To n Do If a[i]=x Then k:=i;.
```

Однако в случае нескольких равных эталону элементов будет найден лишь последний.

Чтобы прервать просмотр сразу после обнаружения заданного элемента, необходимо использовать цикл с предусловием. В результате:

- либо будет найден искомый элемент, то есть найдется такой индекс i, что $a[i] = x$;
- либо будет просмотрен весь массив, и искомый элемент не будет обнаружен.

```
Program Example_74;
const n=10;
Var a: Array[1..n] Of Integer;
x, i : Integer;
Procedure Init;  {формирование массива из файла}
Var f: text;
i: Integer;
Begin
   Assign(f,'c:\s.dat'); Reset(f);
   For i:=1 To n Do Read(f,a[i]);
End;
Procedure Print; {печать массива}
Var i: Integer;
Begin
   For i:=1 To n Do Write(a[i]:5);
   Writeln
End;
Begin {основная программа}
   Writeln('массив:'); init; print; Writeln;
   Write('введи эталон x='); Readln(x);i:=0;
   While (i<=n) and (a[i]<>x) Do
   Begin
      i:=i+1;
   End;
   If i<=n then Write ('найден элемент на ', i,'месте')
   Else Write('такого элемента нет');
End.
```

Задания

1. Оформить программу и проследить ее работу в режиме пошагового просмотра при различных значениях x.

2. Модифицировать программу для поиска элемента массива, равного x, с максимально возможным индексом.

3. Подсчитать число производимых сравнений:

а) в лучшем случае;

б) в худшем случае;

в) в среднем.

Упражнение № 11. Бинарный поиск (поиск делением пополам)

Пример. Дано целое число x и массив a[1..*n*], *отсортированных в порядке неубывания чисел, то есть для любого k:* $1 \le k < n$: $a[k-1] \le a[k]$. *Найти такое i, что* $a[i]=x$ *или сообщить, что элемента x в массиве нет.*

Решение. Идея бинарного метода состоит в том, чтобы проверить, является ли x средним элементом массива. Если да, то ответ получен. Если нет, то возможны два случая:

• *x меньше среднего элемента*, следовательно, в силу упорядоченности массива a, можно исключить из рассмотрения все элементы массива, расположенные в нем правее среднего (так как они больше среднего элемента, который больше x) и применить этот метод к левой половине массива;

• *x больше среднего элемента*, следовательно, рассуждая аналогично, можно исключить из рассмотрения левую половину массива и применить этот метод к его правой части.

Средний элемент и в том, и в другом случае в дальнейшем не рассматривается. Таким образом, на каждом шаге отсекается та часть массива, где не может быть обнаружен элемент x.

Рассмотрим пример:

Пусть $x = 6$, а массив a состоит из 10 элементов:

3 5 6 8 12 15 17 18 20 25.

1-й шаг: найдем номер среднего элемента: $m = \left[\frac{1+10}{2}\right] = 5$. Так как $6 < a[5]$, далее можем рассматривать только элементы, индексы которых меньше 5. Об остальных элементах можно сразу сказать, что они больше x вследствие упорядоченности массива, и среди них искомого элемента нет:

3 5 6 8 ~~12 15 17 18 20 25~~;

2-й шаг: рассматриваем лишь первые четыре элемента массива; находим индекс среднего элемента этой части $m = \left[\frac{1+4}{2}\right] = 2$; $6 > a[2]$, следовательно, первый и второй элементы из рассмотрения исключаем:

~~3 5~~ 6 8 ~~12 15 17 18 20 25~~;

3-й шаг: рассматриваем два элемента; значение $m = \left[\frac{3+4}{2}\right] = 3$;

~~3 5~~ **6** 8 ~~12 15 17 18 20 25~~;

a[3]=6! Элемент найден, его номер — 3.

Ниже приведен фрагмент программной реализации бинарного поиска.

```
Begin
   l:=1; r:=n;   {на первом шаге рассматривается весь массив}
   f:=False;  {признак того, что x не найден}
   While (l<=r) And Not f Do
   Begin
      m:=(l+r) Div 2;
      If a[m]=x Then f:=True
      {элемент найден! Поиск надо прекратить}
      Else If a[m]<x Then l:=m+1     {отбрасывается левая часть}
      Else r:=m-1   {отбрасывается правая часть}
   End
End.
```

Упражнение № 12. Поиск подстроки в строке

Пример 75. *Заданы две строки s и x. Длина первой строки n, длина второй — m, причем $0 \le m \le n$. Требуется ответить на вопрос, является ли строка x подстрокой строки s, при этом поиск подстроки должен обнаруживать первое вхождение x в s.*

Решение. Самым простым методом поиска является метод прямого поиска. Рассмотрим его на примере. Пусть s = «воротник», а x = «рот». Длина первой строки $n = 8$, длина второй — $m = 3$. В данном случае x является подстрокой s, следовательно, надо найти такое значение индекса i, что для любого значения индекса k (1„k„3) будет выполняться равенство $s[i+k] = x[k]$.

Начальное значение i равно 0.

1-й шаг: $i = 0$, $k = 1$ — сравниваем $s[1]$ и $x[1]$: «в» ≠ «р», значит, с первой позиции вхождения нет, надо увеличить на 1 значение i.

2-й шаг: $i = 1$, $k = 1$ — сравниваем $s[2]$ и $x[1]$: «о» ≠ «р», снова надо перейти к следующему i.

3-й шаг: $i = 2$, $k = 1$ — сравниваем $s[3]$ и $x[1]$: «р» = «р», следовательно, возможно совпадение, надо увеличить k.

4-й шаг: $i = 2$, $k = 2$ — $s[4] = x[2]$ («о» = «о») — снова надо увеличить k.

5-й шаг: $i = 2$, $k = 3$ — $s[5] = x[3]$ («т» = «т») — полное совпадение! Далее поиск можно не продолжать, так как требовалось обнаружить лишь первое вхождение x в s.

Таким образом, прямой поиск подстроки в строке сводится к последовательным сравнениям отдельных символов. Поиск продолжается до тех пор, пока не обнаружится вхождение (факт вхождения будем хранить в логической переменной f) или пока не будет пройдена вся строка s. При этом можно закончить просмотр, когда i будет равно $n - m$, так как при следующих значениях i длина любого фрагмента строки s с позиции i меньше m.

```
Program Example_75;
Var s,x: String;
i,j,n,m: Integer;
f: Boolean;
Begin
   Writeln('введите s, x'); Readln(s); Readln(x);
   n:=length(s); m:=length(x);   {определение длин строк}
   i:=0;
   f:=False;  {признак того, что подстрока найдена}
   Repeat
```

```
    j:=1;
    While (j<=m) And (s[i+j]=x[j]) Do Inc(j);
    If j=m+1 Then f:=True Else Inc(i);
    Until f Or (i>n-m);
    If f Then Writeln(x,'является подстрокой',s,'с позиции - ',i)
    Else Writeln(x,'не является подстрокой',s);
    Readln;
End.
```

Этот алгоритм требует достаточно больших временных затрат, поскольку, когда n значительно больше m, число выполняемых сравнений — $(n - m)m \sim nm$.

Пример 76. *Даны строка и подстрока. Составить программу, которая определяет вхождение подстроки в строку по алгоритму Бойера — Мура.*

Решение. Поиск ведется от начала строки s, но с конца искомой подстроки x, для которой формируется таблица, размерность которой равна 256 (число всех возможных символов). Элементами таблицы являются расстояния от последнего символа искомой подстроки x до ее каждого символа. (Если в x встречаются одинаковые символы, то в таблицу заносится расстояние до ближайшего из них.) Если символ в x не входит, то в соответствующую ячейку таблицы заносится m — длина подстроки x. Когда очередной символ подстроки не совпадает с очередным символом строки s, для последнего из таблицы расстояний определяется соответствующее расстояние, после чего x сдвигается вправо на соответствующее число позиций. Тем самым ряд позиций пропускается, сокращая время поиска.

```
Program Example_76;
Var s,x: String;
sd: Array[0..255] Of Integer;
Procedure search(s,x:String);
Var i,j,n,m: Integer;
   f: Boolean;
   h: Char;
Begin
   n:=Length(s); m:=Length(x);{определение длин строки
   и подстроки; начальное заполнение массива расстояний}
   For i:=0 To 255 Do sd[i]:=m;
   For i:=1 To m-1 Do    {заполнение массива расстояний}
   Begin
      h:=x[i]; sd[Ord(h)]:=m-i;
   End;
   i:=0; f:=False;   {признак того, что подстрока найдена}
   While (i<n-m+1) And (Not f) Do
   Begin
      j:=m;
      While (j>0) And (s[i+j]=x[j]) Do j:=j-1;
      If j=0 Then f:=True  {полное совпадение!}
      Else i:=i+sd[Ord(s[i+j])];
   End;
   If f Then Writeln(x,'является подстрокой',s,'с позиции',i+1)
   Else Writeln('нет вхождения.');
End;
```

```
Begin {Основная программа}
   Writeln('Введите строку s.'); Readln(s);
   Writeln('Введите подстроку x.');
   Readln(x); search(s,x); Readln
End.
```

Пример 77. *Ниже представлена программа поиска подстроки в строке по алгоритму Кнута, Мориса, Пратта. Изучите идею метода и работу алгоритма.*

```
Program Example_77;
Const NMax=50; MMax=250;
Type MString=String[MMax];
NString=String[Nmax];
Var s: MString;  {строка}
   p: NString;   {подстрока}
   m: 1..Mmax;   {длина строки}
   n: 1..Nmax;   {длина подстроки}
   d: array[1..Nmax] Of Integer;
Procedure Table(x:NString); {вычисление значений массива d}
Var j: 0..Nmax;  {текущий символ подстроки}
   k:0..Nmax; {длина максимальной последовательности символов,
   совпадающих с началом подстроки}
Begin
   j:=1; k:=0; d[1]:=0;
   While j<n Do
   Begin
      While (k>0) and (p[j]<>p[k]) Do k:=d[k];
      Inc(j); Inc(k);
      If p[j]=p[k] Then d[j]:=d[k] Else d[j]:=k;
   End;
End;
Procedure Search;
Const j: 0..Nmax=0;  {текущий символ строки}
      i: 0..MMax=0; {текущий символ подстроки}
Begin
   j:=0; i:=0;
   Repeat
   Inc(i); Inc(j);
   While (j>0) and (p[j]<>s[i]) Do j:=d[j];
   Until (j=n) Or (i=m);
   If j=n Then
   Writeln('Подстрока входит в строку с позиции ',i-j+1)
   Else WriteLn('Подстрока не найдена');
End;
Begin
   WriteLn('Введите строку');
   ReadLn(s); m:=Length(s);
   WriteLn('Введите подстроку для поиска');
   ReadLn(p); n:=Length(p);
   Table(p); Search; ReadLn;
End.
```

Задания для самостоятельной работы

1. Написать программу сортировки методом простого выбора элементов массива, имеющих нечетные индексы.
2. Написать программу сортировки методом простого выбора нечетных элементов массива.
3. Написать программу сортировки методом простых вставок элементов массива, имеющих четные индексы.
4. Написать программу сортировки методом простых вставок четных элементов массива.
5. Использование идеи двоичного поиска позволяет значительно улучшить алгоритм сортировки массива методом простого включения. Учитывая, что готовая последовательность, в которую надо вставить элемент, является упорядоченной, можно методом деления пополам определять позицию включения нового элемента в нее. Такой модифицированный алгоритм сортировки называется методом двоичного включения. Написать программу, реализующую этот метод.

Дополнительная литература

1. *Баррон Д.* Рекурсивные методы в программировании: Пер. с англ. — М.: Мир, 1974.
2. *Бердж В.* Методы рекурсивного программирования: Пер. с англ. — М.: Машиностроение, 1983.
3. *Брукс Ф. П.* Как проектируются и создаются программные комплексы / Очерки по системному программированию: Пер. с англ. — М.: Наука, 1979.
4. *Вирт Н.* Алгоритмы и структуры данных: Пер. с англ. — М.: Мир, 1989.
5. *Вирт Н.* Алгоритм + структура данных = программа: Пер. с англ. — М.: Мир, 1985.
6. *Кнут Д.* Искусство программирования: Пер. с англ. Т. 1, 2, 3. — М.: Мир, 1976 — 1978.
7. *Кристиан К.* Руководство по программированию на языке МОДУЛА-2: Пер. с англ. — М.: Мир, 1989.
8. *Лорин Г.* Сортировка и системы сортировки: Пер. с англ. — М.: Наука, 1983.
9. *Миков А. И.* Информатика. Введение в компьютерные науки. — Пермь: ПГУ, 1998.
10. *Рейнгольд Э., Нивергельт Ю., Део Н.* Комбинаторные алгоритмы. Теория и практика: Пер. с англ. — М.: Мир, 1980.
11. *Холстед М.Х.* Начала науки о программах: Пер. с англ. — М.: Финансы и статистика, 1981.

§ 3. ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ СИ

Рекомендации по проведению занятий

Практикум по программированию на языке Си направлен на общее знакомство с особенностями этого языка программирования и пропедевтику дальнейшего его изучения в специальных курсах. Его целесообразно проводить в форме лабораторных работ, а также на вычислительной практике, в форме коллективной работы

над программным проектом объемом 4000—10000 строк. Полезно проведение и семинарских занятий, написание рефератов, посвященных истории языка Си, практическим примерам применения, а также сопоставлению с другими языками программирования (Basic, Pascal).

При выполнении лабораторных работ полезно ориентироваться на тот же набор задач, который решался в практикуме по языку Pascal, и некоторый набор новых задач, ярко демонстрирующих особенности языка Си (такие, как сокращенное присваивание и арифметические операции, развитый ввод и вывод, функции с косвенным эффектом). Для выполнения работ необходимо иметь большое число вариантов однотипных задач, дифференцированных по сложности с тем, чтобы обеспечить их индивидуальное решение группой студентов.

Краткие сведения

Программа на языке Си

Си-программа является текстовым файлом с собственным именем и с расширением `.c`.

Программа на языке Си представляет собой набор последовательно описанных функций (процедуры и подпрограммы в языке Си считаются частным случаем функций). Каждая функция — самостоятельная единица программы, предназначенная для решения определенной задачи (или подзадачи).

При описании она имеет следующий вид:

```
Тип_функции Имя (<список аргументов>)
<описания аргументов>
{
<описания>
<операторы>
}
```

Список аргументов может быть пустым (однако скобки после ее имени сохраняются).

Имеется одна главная функция (с именем `main`), с которой начинается выполнение программы. Функции могут обращаться к другим функциям посредством конструкций вызова. Вызов функции используется при вычислении значения выражения. В результате вызова функция возвращает вычисленное значение, которое и является значением вызова функции. Попутно функция может преобразовывать значения своих аргументов. Такой результат вызова функции называется побочным эффектом.

В модуле, вызывающем данную функцию, тип возвращаемого ею значения должен быть описан (даже если это — неопределенное значение) вместе с описанием переменных.

Аргументы (которые бывают формальными и фактическими) передаются по значению путем копирования в соответствующие (по порядку) параметры, указанные в определении функции. Соответствие аргументов и параметров по количеству и типу не контролируется в языке Си.

Основные операторы языка

Операторы препроцессора могут появляться в любом месте программы и их действие распространяется на весь исходный файл.

Весьма часто используются следующие операторы препроцессора:

```
#include
#define
```

Важная возможность препроцессора — включение в исходный текст содержимого других файлов. Эта возможность в основном используется для того, чтобы снабжать программы какими-то общими для всех данными определениями. Например, чрезвычайно часто в начале программы на языке Си встречается препроцессорная конструкция

```
#include <stdio.h>
```

Когда исходный текст программы обрабатывается препроцессором, на место этой инструкции ставится содержимое расположенного в некоем стандартном месте (каталоге INCLUDE) файла `stdio.h`, содержащего макроопределения и объявления данных, необходимых для работы функций из стандартной библиотеки ввода-вывода.

Директива #define позволяет дать в программе ***макроопределения*** (или задать ***макросы***). Оператор макроопределения имеет вид:

```
#define <макроимя> <строка лексем>
```

или

```
#define <макроимя (<список параметров>)> <строка лексем>
```

Макроимя — идентификатор. *Строка лексем* — последовательность лексем от Макроимени до конца строки. Точка с запятой в конце макроопределения не ставится.

Препроцессорная обработка макроопределения сводится к тому, что любое появление Макроимени (макровызов) в качестве отдельной лексемы в тексте программы, расположенном после макроопределения, ведет к замене этого Макроимени на указанную Строку лексем.

Оператор присваивания имеет общий вид

```
<Имя переменной> = <Выражение>;
```

В языке Си разрешается включать присваивания в выражения, то есть присваивание может рассматриваться как операция с учетом старшинства и влияния скобок.

В языке Си имеются специальные операции

```
+= -= *= /= %=
```

для компактной записи операторов присваивания. Так, следующие две записи на языке Си эквивалентны: `i=i+2` и `i+=2`.

Операция присваивания сама по себе имеет значение (равное значению выражения, стоящего справа от знака «=») и может входить в выражения.

Оператор **`if/else`** имеет вид

```
if (<выражение>)
<оператор1>
else
<оператор2>;
```

Здесь часть `else <оператор2>` является необязательной, можно применять и одиночный оператор

```
if(<выражение>) <оператор1>;
```

Вначале вычисляется значение выражения. `Оператор1` выполняется, если значение выражения истинно. Если выражение ложно (его значение равно нулю) и если есть часть с `else`, то выполняется `оператор2`.

При программировании требуется аккуратно различать знаки = и ==, потому что в ряде случаев компилятор не сможет обнаружить ошибки, связанной с неправильным использованием знаков этих операций, что приведет к неверным результатам.

В языке Си имеется компактный способ записи оператора `if/else`. Он называется «условным выражением» или «тернарной операцией». Такое выражение выглядит так:

```
B1 ? B2 : B3
```

Сначала вычисляется значение выражения `B1`. Если оно отлично от нуля (истинно), то вычисляется значение выражения `B2`, которое и становится значением условного выражения. В противном случае вычисляется значение выражения `B3`, и оно становится значением условного выражения. Условное выражение удобно использовать в тех случаях, когда имеется некоторая переменная, которой можно присвоить одно из двух возможных значений. Типичным примером являются присваивание переменной значения большей из двух величин: `max=(a>b)?a:b;`

***Оператор-переключатель* `switch`** удобен в тех случаях, когда в программе необходимо произвести выбор одного из нескольких вариантов. Его синтаксис:

```
switch (<выражение>)
{
case <константа1>: <список операторов1>;
case <константа2>: <список операторов2>;
. . .
case <константаN>: <список операторовN>;
default : <список операторов>;
}
```

Оператор-переключатель выполняется следующим образом. Вычисляется значение выражения в скобках, приведенного после ключевого слова `switch`, затем программа просматривает список меток, указанных после слов case, до тех пор, пока не находит ту, которая соответствует данному значению. Далее программа переходит к выполнению оператора, расположенного в этой строке. Если подходящей метки не найдется и если существует строка с меткой `default`:, то будет выполняться оператор, помеченный этой меткой. В противном случае произойдет переход к оператору, расположенному за оператором `switch`.

Оператор цикла

```
for (<оператор1>;<выражение>;<оператор2>)
<оператор3>;
```

позволяет организовать повторяющийся вычислительный процесс. Как правило, `оператор1` и `оператор2` являются операторами присваивания или обращения к функции, а выражение — условным выражением.

Цикл `for` удобно использовать в тех случаях, когда заранее известно число повторений тела цикла или имеется явно выраженная переменная, управляющая циклом. В этом случае `выражение1` вычисляется один раз и задает инициализацию управляющей переменной. Выражение является условием завершения цикла, а оператор2 задает приращение управляющей переменной.

Любой из операторов и выражений в цикле `for` может быть опущен, хотя точка с запятой при этом должна оставаться. Если отсутствует `оператор1` или `оператор2`, то он просто выпадает из вычислений. Если же отсутствует `выражение`, то считается, что оно всегда истинно.

Оператор цикла с предусловием **`while`** в общем виде записывается так:

```
while (<выражение>)
<оператор>;
```

Цикл `while` является «условным» циклом, использующим условие на входе. Если выражение «истинно» (или в общем случае не равно нулю), то оператор, входящий в цикл `while`, выполняется один раз, а затем выражение проверяется снова. Эта последовательность действий, состоящая из проверки и выполнения оператора, периодически повторяется до тех пор, пока выражение не станет ложным (или в общем случае равным нулю). После этого управление передается оператору, следующему за оператором цикла `while`.

При построении цикла `while` необходимо включить в него какие-то конструкции, изменяющие величину проверяемого выражения так, чтобы в конце концов оно стало ложным. В противном случае выполнение цикла никогда не завершится.

Оператор цикла с постусловием — в нем истинность условия проверяется после выполнения каждой итерации цикла. Этот подход реализуется с помощью цикла `do/while`.

Тело цикла `do/while` всегда выполняется, по крайней мере один раз, поскольку проверка условия осуществляется только после его завершения.

Форма записи:

```
do <оператор>
while (<выражение>);
```

Оператор **`break`** дает возможность выйти из операторов цикла `for`, `while`, `do/while`, а также из переключателя `switch` без проверки условия. Оператор `break` приводит к немедленному выходу из самого внутреннего охватывающего его цикла или из переключателя.

Оператор **`continue`** вызывает преждевременное завершение выполнения тела цикла и переход к следующему шагу цикла. Оператор `continue` действует только на самый внутренний цикл, частью которого он является.

Оператор безусловного перехода **`goto`** предназначен для безусловной передачи управления в заданную точку программы. Его выполнение заключается в передаче управления оператору, помеченному заданной меткой.

В качестве метки используется идентификатор. Метка отделяется от оператора, к которому она относится, двоеточием.

Синтаксис оператора:

```
goto <метка>;
. . .
<метка>: <оператор>
```

Символы «{» и «}» используются для объединения описаний и операторов в составной оператор или блок, так что все конструкции, заключенные в фигурные скобки, оказываются синтаксически эквивалентными одному оператору. Точка с запятой никогда не ставится после первой фигурной скобки, которая завершает блок.

Оператор **`return`** завершает выполнение данной функции и передает управление вызывающей функции. Оператор `return`, в главной функции `main`, вызывает завершение выполнения всей программы.

Оператор return может содержать любое выражение:

```
return (<выражение>);
```

Если выражение не пусто, то вычисляется его значение, которое и становится значением данного вызова функции. Достижение «конца» функции (правой закрывающей фигурной скобки) эквивалентно выполнению оператора `return` без возвращаемого значения (т.е. оператор `return` в конце функции может быть опущен). С помощью оператора `return` функции можно вернуть и несколько значений.

Средства ввода-вывода. Имеется ряд библиотечных функций Си, обеспечивающих стандартную систему ввода-вывода для программ на Си. Макроопределения, описания переменных и определения этих функций содержатся в файле стандартных заголовков `stdio.h`. Поэтому каждая пользовательская программа должна содержать в начале ссылку

```
#include <stdio.h>.
```

Наиболее часто используется функция форматного вывода printf. В общем случае обращение к этой функции имеет вид:

```
printf(<формат>,<выражение1>,<выражение2>,...,<выражениеN>);
```

где `<выражение1>`, `<выражение2>`, ..., `<выражениеN>` — произвольные выражения, результаты которых надо вывести. Управляющая строка «формат» содержит объекты двух типов: обычные символы, которые просто копируются в выходной поток (печатаются), и спецификации преобразования значений из внутреннего машинного представления в текстовое для вывода на печатающем устройстве.

Функцией ввода, аналогичной функции вывода `printf()`, является `scanf()` — стандартная функция форматного ввода.

Обращение к этой функции имеет вид:

```
scanf(<формат>,<&имя1>,<&имя2>,...,<&имяN>);
```

где `<имя1>`, `<имя2>`, ..., `<имяN>` — имена переменных, значения которых надо ввести. Наличие символа «`&`» перед каждым именем обязательно (кроме переменных строкового типа), его смысл будет пояснен ниже.

При обращении к функции `scanf` выполнение программы приостанавливается, ожидается ввод значений указанных переменных, после чего работа программы продолжается.

В качестве спецификаций в формате можно использовать те же символы, что и в функции `printf()`. Спецификации формата должны соответствовать количеству и типу вводимых переменных.

Набор стандартных функций ввода и вывода включает большое число функции для работы с данными различного типа, различными устройствами, буферизованного и небуферизованного, форматного и бесформатного ввода и вывода.

Лексемы и литералы

Лексемами называют последовательности символов языка (идентификаторы, служебные слова, константы, строки, составные знаки операций, разделители). Лексемы разделяются пробелами и другими неграфическими символами языка.

Идентификатор — это последовательность латинских букв, цифр и символа «_», начинающаяся с буквы или символа «_».

Большие и маленькие латинские буквы считаются различными! Например, *y* и *Y* — это разные имена. Рекомендуется в именах переменных использовать только строчные буквы.

Примеры правильных идентификаторов:

```
schetchik get_line a12 Param1 _ab
```

Примеры неправильных идентификаторов:

```
%ab 12abc -x вася
```

Литералы — неизменяемые объекты языка (константы). Литерал может быть либо числовым, либо символьным, либо строковым.

Числовые литералы могут быть десятичными (целыми и вещественными, простыми и длинными), восьмеричными, шестнадцатеричными.

Примеры:

```
/* Целые десятичные литералы */
57 32000001 /* длинный */ 2e3 5E3
/* Вещественные десятичные литералы */
0.00 5.3 7.1e-3 6.34E-2 .21e+56
```

Лидирующий нуль (0) указывает на числовой восьмеричный литерал:

```
030 /* Десятичное 24 */
040 /* Десятичное 32 — символ пробел */
```

Лидирующий 0 указывает на числовой шестнадцатеричный литерал:

```
0x22 /* Десятичное 34 — символ « */
0x6C /* Десятичное 108 — символ i */
```

Символьный литерал — это один символ, заключенный в одинарные кавычки:

```
'c' '*' 'q' '\007'   /* Звонок, восьмеричный код после \ */
'\x0a' /* Перевод на новую строку, шестнадцатеричный код после \x */
```

Последовательность символов, заключенных в двойные кавычки, называется ***строковым литералом***.

Примеры:

```
"STRING\n"
"" /* Строчный литерал состоит из одного символа "\0" */
"Очень,"\
"очень,"\
"очень длинный строковый литерал!"
```

Программа на процедурных языках, к которым относится Си, представляет собой описание операций над величинами различных типов. Тип определяет множество значений, которые может принимать величина, и множество операций, в которых она может участвовать. В языке Си типы связаны с именами (идентификаторами) величин, то есть с переменными. Все переменные должны быть описаны до их использования. Каждая переменная должна быть описана только один раз.

Типы величин и их описание

Описание состоит из спецификатора типа и следующего за ним списка переменных. Переменные в списке разделяются запятыми. В конце описания ставится точка с запятой.

Переменным могут быть присвоены начальные значения внутри их описаний. Если за именем переменной следует знак равенства и константа, то эта константа служит в качестве инициализатора.

Основные типы в языке Си:

int — целый (*«integer»*); имеются служебные слова, которые можно использовать с типом int: `short int` (*«short integer»* — *«короткое целое»*), `unsigned int` (*«unsigned integer»* — *«целое без знака»*), `long int` (*«длинное целое»*), которые сокращают или, наоборот, расширяют диапазон представления чисел.

char — символьный (*«character»*); допустимое значение для этого типа — один символ (не путать с текстом!); символ записывается в апострофах.

float — вещественный (с плавающей точкой). Значения этого типа — числа, но, в отличии от `char` и `int`, не обязательно целые.

double — вещественные числа двойной точности; этот тип аналогичен типу float, но имеет значительно больший диапазон значений (например, для системы программирования Borland-C от 1.7E-308 до 1.7E+308 вместо диапазона от 3.4E-38 до 3.4E+38 для типа float).

В языке Си нет специального типа, который можно было бы использовать для описания строк. Вместо этого строки представляются в виде «массива» элементов типа `char`. Это означает, что символы в строке будут располагаться в соседних ячейках памяти.

Указатель — некоторое символическое представление адреса ячейки памяти, отведенной для переменной. Например, `&name` — указатель на переменную *name*. Здесь `&` — операция получения адреса. Фактический адрес — это число, а символическое представление адреса `&name` является константой типа «указатель».

Спецификация типа задает тип переменной, на которую ссылается указатель, а символ * определяет саму переменную как указатель. Описание вида `int *pi;` говорит, что `pi` — это указатель и что `*pi` — величина типа `int`.

Для ***описания массива*** (одномерного и многомерного) после идентификатора переменной при описании ее типа в квадратных скобках указывается максимальное значение индекса элементов массива. Минимальное значение индекса 0.

В языке Си предусмотрена возможность ***определения имен типов данных***. Любому типу данных с помощью определения typedef можно присвоить имя и использовать это имя в дальнейшем при описании объектов.

Формат: `typedef <старый тип> <новый тип>`

Имя производного типа рекомендуется записывать прописными буквами, чтобы они выделялись в тексте программы. Определение typedef не вводит каких-либо новых типов, а только добавляет новое имя для уже существующего типа.

Основные группы операций

Арифметические операции

Сложение (+)	*Вычитание (бинарное)* (−)	*Умножение* (*)
Деление (/)	*Остаток от деления нацело* (%)	*Вычитание (унарное)* (−)

В языке Си принято правило: если делимое и делитель имеют тип `int`, то деление производится нацело, то есть дробная часть результата отбрасывается.

Специальные операции

Увеличение (++) *Уменьшение* (--).

Следующие записи на языке Си являются эквивалентными:

```
i=i+1; и i++;
j=j-1; и j--; .
```

Символ «++» или «--» записывается после имени переменной или перед ним.

Как и обычные присваивания, увеличение и уменьшение можно использовать в выражениях. При этом существенно, с какой стороны от имени стоит знак «++» или «--». Если знак стоит перед переменной (в этом случае говорят о префиксной форме операции), то сначала выполняется увеличение (уменьшение) значения переменной, а лишь затем полученный результат используется в выражении. Если же знак стоит после переменной (постфиксная форма операции), то в выражении используется старое значение переменной, которое затем изменяется.

Операции отношения и логические операции

Больше или равно >=	*Больше* >	*Меньше или равно* <=
Меньше <	*Равно* ==	*Не равно* !=
Логическое «и» &&	*Логическое «или»* ||	*Отрицание «не»* !

Логическое значение «ложь» представляется целым нулевым значением, а значение «истина» представляется любым ненулевым значением.

Выражения, связанные логическими операциями `&&` и `||`, вычисляются слева направо, причем вычисление значения выражения прекращается сразу же, как только становится ясно, будет ли результат истинен или ложен.

Старшинство операции `&&` выше, чем у операции `||`.

Битовые операции

Битовое «и» &	*Сдвиг вправо* >>
Битовое «или» |	*Сдвиг влево* <<
Битовое исключающее «или» ~	*Инверсия битов* (унарная операция) \~

Выражения

Выражения — это конструкции, включающие константы (литералы), переменные, знаки операций, скобки для управления порядком выполнения операций, обращения к функциям. Если в выражениях встречаются операнды различных типов, то они преобразуются к общему типу в соответствии в определенными правилами:

1) переменные типа `char` интерпретируются как целые без знака (`unsigned`);

2) переменные типа `short` автоматически преобразуются в `int`; если один из операндов имеет тип `unsigned`, то другой (другие) также преобразуется к типу `unsigned` и результат имеет тип `unsigned`;

3) если один из операндов имеет тип `int`, то другой (другие) также преобразуется к типу `int` и результат имеет тип `int`;

4) если один из операндов имеет тип `char`, то другой (другие) также преобразуется к типу `char` и результат имеет тип `char`;

5) во время операции присваивания значение правой части преобразуется к типу левой части, который и становится типом результата;

6) в процессе преобразования `int` в `char` лишние старшие 8 бит просто отбрасываются.

Кроме того, существует возможность точно указывать требуемый тип данных, к которому необходимо привести некоторую величину (в скобках перед этой величиной). Скобки и имя типа вместе образуют операцию, называемую *приведением типов.*

Контрольные вопросы

1. Какова область применения языка программирования Си?
2. Охарактеризуйте этапы обработки программы на Си при получении исполняемого модуля.
3. Что такое директивы препроцессора?
4. Какова структура программы на Си?
5. Какие типы данных используются в Си?
6. Перечислите виды присваиваний в Си?
7. Чем отличается запись основных операторов на Си от их записи в Pascal?
8. Охарактеризуйте встроенные функции форматного ввода и вывода в Си.
9. Охарактеризуйте встроенные функции неформатного ввода и вывода в Си.
10. Как организуются модули программы Си – функции, определяемые программистом?

Темы для рефератов

1. История возникновения и перспективы развития языка Си.
2. Сравнительный анализ языков Си и Basic.
3. Сравнительный анализ языков Си и Pascal.
4. Сфера применения языка Си при создании промышленных систем.
5. Особенности реализации языка Си в системах программирования различных видов.
6. Библиотеки и надстройки языка Си.

Темы семинарских занятий

1. История возникновения и перспективы развития языка Си. Область его применения.
2. Типы данных в Си и особенности типизации данных в Си.
3. Сравнение записи основных операторов в Си и Pascal. Особенности организации модульной структуры программы в Си.
4. Стандартные библиотеки, ввод-вывод в Си.
5. Система программирования Borland C++.3.1

Рекомендации по программному обеспечению

Для выполнения лабораторного практикума можно рекомендовать использовать систему программирования Borland C++ версии 3.1, работающую на компьютерах IBM PC 386 и старше под управлением MS DOS, а также Windows NT и 9X. Данная

система также устойчиво работает в бездисковых классах компьютеров, загружаясь с сервера Novell. Настройки по умолчанию вполне удовлетворяют в большинстве случаев. Следует, однако, следить за соответствием описанных директориев INCLUDE и LIB реально существующим (и доступным на сетевых дисках).

Задачи и упражнения

Задачи, сформулированные в форме вопроса или задания «найти...» (вычислить, определить), предполагают написание соответствующей программы. По уровню сложности задачи разделены на 3 категории: самые простые (без звездочек), посложнее (*) и относительно сложные (**). Сложность в данном случае носит идейный характер и не приводит к увеличению размеров программы.

Выражения, ветвление

1. Вычислить значения производной функции x^x в заданной точке a ($a > 0$).

2. Для заданного a вычислить принадлежащий интервалу $(J, 2J)$ корень уравнения $\ln(\mathrm{ctg}x - 1) = a$.

3.* Вычислить дробную часть среднего геометрического трех заданных положительных чисел.

4. По заданным коэффициентам и правым частям уравнений системы

$$a_1x + b_1y = c_1;$$
$$a_2x + b_2y = c_2$$

найти ее решения в предположении, что определитель системы не равен 0.

5.* Определить, является ли треугольник, заданный длинами его сторон, остроугольным.

6. Вычислить периметр и площадь прямоугольного треугольника по длинам двух катетов.

7. По координатам трех вершин некоторого треугольника найти его площадь и периметр.

8. По длинам двух сторон некоторого треугольника и углу (в градусах) между ними найти длину третьей стороны и площадь этого треугольника.

9.* Найти произведение цифр заданного четырехзначного числа.

10.* Определить число, полученное выписыванием в обратном порядке цифр заданного трехзначного числа.

11.**Определить, равна ли сумма двух первых цифр заданного четырехзначного числа сумме двух его последних цифр.

12.**Определить, есть ли среди первых трех цифр из дробной части заданного положительного вещественного числа цифра 0.

13.**Определить, есть ли среди цифр заданного трехзначного числа одинаковые.

14. Даны три произвольных числа. Определить, можно ли построить треугольник с такими длинами сторон.

15. Даны координаты (как целые от 1 до 8) двух полей шахматной доски. Определить, может ли конь за один ход перейти с одного из полей на другое.

16. Перераспределить значения переменных x и y так, чтобы в x оказалось большее из этих значений, а в y — меньшее.

17. Переменной k присвоить номер четверти плоскости, в которой находится точка с координатами x, y ($xy <> 0$).

18. Вычислить

$$y = \frac{\max(x, y, z) - 2x \min(x, y, z)}{\sin 2 + \max(x, y, z) / \min(x, y, z)}.$$

19.* Даны числа a_1, b_1, c_1, a_2, b_2, c_2. Напечатать координаты точки пересечения прямых, описываемых уравнениями $a_1x + b_1y = c_1$, $a_2x + b_2y = c_2$, или сообщить, что прямые совпадают, не пересекаются или вообще не существуют.

20.**Дано целое k от 1 до 180. Определить, какая цифра находится в k-й позиции последовательности 10111213...9899, в которой выписаны подряд все двузначные числа.

21.** Определить, равен ли квадрат заданного трехзначного числа кубу суммы цифр этого числа.

22.* Даны действительные числа x_1, x_2, x_3, y_1, y_2, y_3. Определить, принадлежит ли начало координат треугольнику с вершинами (x_1, y_1), (x_2, y_2), (x_3, y_3).

23.* Дано натуральное число $n<9999$. Является ли это число палиндромом?

24.* На поле (k, l) шахматной доски расположен ферзь. Угрожает ли он полю (m, n)?

25. Выяснить, являются ли поля (k, l) и (m, n) шахматной доски полями одного цвета.

Циклы и ветвления

1.** Подсчитать k — число цифр в десятичной записи целого неотрицательного числа n.

2.* Проверить, является натуральное число k степенью 3 или нет.

3. Дано 10 вещественных чисел. Вычислить разность между максимальным и минимальным из них.

4.* Даны целое $n>0$ и последовательность из n вещественных чисел, среди которых есть хотя бы одно отрицательное число. Найти величину наибольшего среди отрицательных чисел этой последовательности.

5.* Вычислить по алгоритму Горнера $y = x^{10} + 2x^9 + 3x^8 + \ldots + 10x + 11$; $y = 11x^{10} + 10x^9 + 9x^8 + \ldots + 2x + 1$.

6. Вычислить $y = n!!$, $n>0$.

7. Дана непустая последовательность различных натуральных чисел, за которой следует 0. Определить порядковый номер наименьшего из них.

8.* Даны натуральное число n и вещественные числа t, a_0, a_1, ..., a_n. Вычислить значение многочлена $a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n$ и его производной в точке $t^2 + 0{,}5$.

9. Пусть задано значение x. Найти первое из чисел $\sin x$, $\sin\sin x$, $\sin\sin\sin x$, ..., меньшее по модулю 10^{-4}.

10.* Вычислить s — сумму всех чисел Фибоначчи, которые не превосходят 1000.

11. Вычислить для заданного n

$$\sum_{k=1}^{n} \sin(kn).$$

12. Напечатать в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр (деления не использовать!).

13. Выяснить, можно или нет натуральное число n представить в виде суммы трех полных квадратов.

14.* Даны n — натуральное и вещественные пары $x_1, y_1, \ldots, x_n, y_n$. Определить радиус наименьшего круга с центром в начале координат, внутрь которого попадают эти точки.

15.* Дано 10 вещественных чисел. Найти порядковый номер того из них, которое наиболее близко к целому числу.

16.* Дано 10 целых чисел. Определить, сколько из них принимает наибольшее значение.

17.** Дано не менее трех различных натуральных чисел, за которыми следует нуль. Определить три наибольших числа среди них.

18.* Не используя стандартные функции (за исключением abs), вычислить с точностью $eps > 0$: $y = \cos x = 1 - x^2/2! + x^4/4! - \ldots + (-1)^n x^{2n}/(2n)! + \ldots$.

19. Вычислить k — число точек с целочисленными координатами, попадающих в круг радиуса R.

20.* Дано 10 вещественных чисел. Определить, образуют ли они возрастающую последовательность.

21.** Не используя стандартных функций вычислить tg x.

22.* Дана последовательность положительных целых чисел, за которой следует 0 (признак конца последовательности). Вычислить среднее геометрическое этих чисел.

23. Подобрать всевозможные варианты размена произвольной > 6 руб. суммы с помощью 2- и 5-рублевых монет.

24. Получить сумму нечетных отрицательных элементов последовательности целых чисел $a_1, a_2, \ldots, a_n$.

25. Дано натуральное число n. Вычислить $1 \times 2 + 2 \times 3 \times 4 + \ldots + n \times (n+1) \times \ldots \times 2n$.

Одномерные массивы

1.* Преобразовать массив X по следующему правилу (x'_k — значение k-го элемента массива после преобразования):

$$x'_k = \max(x_i) \text{ при } 1 < i < k.$$

2. Преобразовать массив X, расположив его элементы в обратном порядке.

3.** В целочисленном масиве A содержится десять чисел от 0 до 9 включительно, а в целочисленном массиве B — два целых числа от 0 до 9. Переменной x присвоить вещественное число $0.a_0 a_1 \ldots a_9 \times 10^{b1b2}$.

4.** По массиву t, где указана температура каждого дня некоторого невисокосного года, определить m — название месяца с наибольшей среднемесячной температурой.

5. Преобразовать массив x по следующему правилу (воспользовавшись вспомогательным массивом): все отрицательные элементы массива x перенести в его начало, а все остальные — в конец, сохраняя исходное взаимное расположение как среди отрицательных, так и среди остальных элементов.

6.* Дано натуральное n. Сколько различных цифр встречается в его записи?

7. Переменной k присвоить либо номер первого вхождения y в массив x, либо число $n+1$, если y не входит в x.

8.* Вычислить для массива из n элементов

$y = x_1 + x_1 \times x_2 + x_1 \times x_2 \times x_3 + \ldots + x_1 \times x_2 \times x_3 \ldots x_m$, где m — либо номер первого отрицательного элемента массива x, либо число n, если в массиве x нет отрицательных элементов.

9. Элементы целочисленного массива x упорядочены по возрастанию. Требуется присвоить переменной k номер элемента массива x, равного числу y, или 0, если такого элемента нет.

10. Даны действительные числа $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$.

Вычислить $(a_1 + b_n) \times (a_2 + b_{n-1}) \times \ldots \times (a_n + b_1)$.

11.* Проверить, имеется ли среди элементов массива x хотя бы одно число Фибоначчи.

12.** Упорядочить массив x по возрастанию, используя следующий алгоритм сортировки (метод пузырька): последовательно сравнивая пары соседних элементов x_k и x_{k+1} ($k = 1,2, \ldots, n-1$) и, если $x_k > x_{k+1}$, то они переставляются; тем самым наибольший элемент окажется на своем месте в конце массива; затем этот метод применяется ко всем элементам, кроме последнего, и т. д.

13.** Упорядочить массив x по убыванию, используя следующий алгоритм сортировки (сортировка вставками): пусть первые k элементов массива уже упорядочены по убыванию; берется $(k+1)$-й элемент и размещается среди первых k элементов так, чтобы упорядоченными оказались уже $k+1$ первых элементов; этот метод применяется при k от 1 до $n-1$.

14.* Даны две последовательности по пять целых чисел в каждой. Найти наименьшее среди тех чисел первой последовательности, которые не входят во вторую.

15.** Рассматривая массивы x, y и z как представление некоторых множеств из объектов типа индекс ($x[k] = 1$, если элемент k принадлежит множеству x, и $x[k]=0$ иначе), реализовать следующие операции над этими массивами-множествами:

1) z — объединение множеств x и y;

2) z — пересечение множеств x и y;

3) z — разность множеств x и y.

16. Дана последовательность из десяти целых чисел. Определить число инверсий в этой последовательности.

17. Дана последовательность из десяти целых чисел. Найти сумму чисел этой последовательности, расположенных между максимальным и минимальным числами (в сумму включить и оба этих числа).

18.** Рассматривая массивы x и y как представление некоторых множеств из объектов типа индекс ($x[k] = 1$, если элемент k принадлежит множеству x, и $x[k] = 0$ иначе), проверить, что множество x является подмножеством множества y.

19. Даны координаты n точек на плоскости: $(x_1, y_1), \ldots, (x_n, y_n)$ ($n = 20$). Найти номера двух точек, расстояние между которыми наибольшее (считать, что такая пара точек единственная).

20.** Даны вещественные числа $a_0, a_1, \ldots, a_{15}$. Найти коэффициенты многочлена $(x - a_0)(x - a_1) \ldots (x - a_{15})$.

21. Даны целые числа $a_0, a_1, \ldots, a_n$. Получить новую последовательность, выбросив из исходной все члены, равные $\max(a_0, a_1, \ldots, a_n)$.

22. В последовательности $a_0, a_1, \ldots, a_n$ поменять местами наибольший и наименьший члены.

23.* Даны целые числа $a_0, a_1, \ldots, a_n$. Получить новую последовательность, заменяя a_i нулями, если a_i не равно $\max(a_0, a_1, \ldots, a_n)$ и единицами — в противном случае.

24. Даны целые числа $a_0, a_1, \ldots, a_{2n}$. Получить $\max(a_1+a_{2n}, a_2+a_{2n-1}, \ldots, a_n+a_{n+1})$.

25.** Определить число различных элементов массива X.

Двумерные массивы

1.* Даны натуральное число n и матрица $A(m \times m)$. Вычислить n-ю степень этой матрицы.

2. Заполнить массив A следующим образом:

```
1    2    ...   10
11   12   ...   20
21   22   ...   30
.....
91   92   ...   100
```

3.* Заполнить массив A по правилу: $A_{ij} = X_j^i$, где массив X задан.

4.* Определить k — число «особых» элементов массива C, считая элемент «особым», если он больше суммы остальных элементов своего столбца.

5.** Определить k — число различных элементов матрицы C (т. е. повторяющиеся элементы считать один раз)

6.* Дана вещественная матрица порядка $n \times m$. Упорядочить ее строки по неубыванию их наибольших элементов.

7.* Определить, симметрична ли заданная целая квадратная матрица n-го порядка (относительно главной диагонали)

8. Дана вещественная матрица $n \times n$, все ее элементы различны. Найти скалярное произведение строки с наибольшим элементом матрицы на столбец с наименьшим элементом.

9.** Определить, магический ли квадрат заданная целая квадратная матрица n-го порядка, т.е. такой, где суммы элементов во всех строках и столбцах одинаковы.

10.** По заданным коэффициентам A_{ij} и правым частям B_i решить систему линейных уравнений.

11.* Дана матрица $n \times m$. Переставляя ее строки и столбцы, переместить наибольший элемент в верхний левый угол.

12. Заполнить массив A следующим образом:

```
1   2   3   ...   10
0   1   2   ...   9
0   0   1   ...   8
......
0   0   0   ...   1
```

13. Получить массив B из массива A удалением n-й строки и k-го столбца.

14.** В заданной квадратной целочисленной матрице указать индексы всех элементов с наибольшим значением.

15.* Дана вещественная матрица $n \times m$. Упорядочить ее строки по неубыванию суммы их элементов.

16.* Преобразовать массив S, поворачивая его вокруг центра на 90 градусов против часовой стрелки.

17.** Седловая точка — элемент матрицы, наименьший в своей строке и одновременно наибольший в своем столбце или наоборот. Для заданной матрицы $n \times m$ найти индексы всех ее седловых точек.

18. Определить, является ли заданная целая квадратная матрица 10-го порядка ортонормированной, т.е. такой, что скалярное произведение каждой пары различных строк равно нулю, а каждой строки на себя равно единице.

19.* По заданным коэффициентам A_{11}, A_{12}, ..., A_{1n}, A_{22}, A_{23}, ..., A_{2n}, ... A_{nn} ($A_{ii}<>0$) и правым частям B_1, B_2, ..., B_n найти решение «треугольной» системы линейных уравнений:

$$A_{11}X_1 + A_{12}X_2 + A_{13}X_3 + \ldots + A_{1n}X_n = B_1$$
$$A_{22}X_2 + A_{23}X_3 + \ldots + A_{2n}X_n = B_2$$
$$A_{33}X_3 + \ldots + A_{3n}X_n = B_3$$
$$\ldots\ldots\ldots$$
$$A_{nn}X_n = B_n$$

20.* Преобразовать систему n линейных уравнений с n неизвестными к треугольному виду (см. задачу 19).

21.** Все элементы с наибольшим значением в данной целочисленной матрице заменить нулями

22.* Найти номера строк квадратной целочисленной матрицы, элементы в каждой из которых одинаковы.

23.** Найти номера строк квадратной целочисленной матрицы, элементы каждой из которых образуют монотонную последовательность.

24.* Дана целочисленная матрица $[a_{ij}]$ n-го порядка. Получить b_1, ..., b_n, где $b_k = \max\{a_{1k}, a_{2k}, \ldots, a_{nk}\}$.

25.* Дана целочисленная матрица $[a_{ij}]$ n-го порядка. Получить b_1, ..., b_n, где $b_k = \min\{a_{1k}, a_{2k}, \ldots, a_{nk}\}$

Литеры и строки

1. Напечатать в одну строку все буквы между «*A*» и «*Z*», включая и эти буквы.

2. Проверить, правильно ли в текст входят круглые скобки. Ответ ДА или НЕТ.

3. Удалить из текста все буквы «*b*».

4.* Удалить из текста все буквы «*k*», идущие сразу за буквой «*n*».

5.* Напечатайте текст, удалив из него лишние пробелы, т.е. чтобы пробелы встречались по одному.

6.** Подсчитать число слов в тексте, начинающихся и заканчивающихся одной и той же буквой.

7.* Подсчитать число слов в тексте, содержащих букву «*d*».

8.* Подсчитать число слов в тексте, содержащих одну букву «*a*» и две буквы «*b*».

9.* Напечатать текст, подчеркивая в нем заглавные буквы (строкой ниже).

10.** Удалить из слова повторяющиеся буквы.

11.* Если в заданный текст входит каждая из букв слова «*key*», напечатать *yes*, иначе *no*.

12. Напечатать букву, которая идет в тексте непосредственно за буквой «*a*».

13. Удалить из текста все пары букв «*oo*».

14.** Напечатать все слова в тексте задом наперед.

15.* Подсчитать число слов в тексте, оканчивающихся буквой «*w*».

16.** Проверить, является ли данное слово перевертышем.

17. Подсчитать число слов в тексте, содержащих ровно три буквы «*e*».

18.* Значениями литерных переменных являются цифры. Присвоить составленное из них число целой переменной.

19.* Удалить из слов в тексте все гласные буквы.

20. Если слово нечетной длины, удалить из него среднюю букву.

21.** Рассортировать английские слова в алфавитном порядке.

22.** Подсчитать частоты вхождения букв в текст.

23.** Подсчитать частоты длин слов в тексте.

24. Заменить в тексте строчные буквы прописными, а прописные строчными.

25*. Найти в тексте самое большое целое число.

Процедуры и функции

1. Определить наибольший общий делитель трех натуральных чисел.

2.** Получить число 2^{500}, выводя его по одной цифре.

3.** Получить число 2^{500}, выводя по n цифр.

4.* Вывести таблицу зависимости от n количества целых чисел взаимно простых с n.

5.* Даны коэффициенты многочленов $P(x)$, $Q(x)$ и число a. Вычислить $P(a + Q(a)P(a + 1))$.

6. Напечатать пары простых чисел-«близнецов» из отрезка $[n, 2n]$.

7.* Три прямые заданы уравнениями. Если они попарно пересекаются и образуют треугольник, то найти его площадь.

8. Даны координаты вершин двух треугольников. Определить, какой из них имеет наибольшую площадь.

9.* Написать функцию, приводящую дробь к несократимому виду.

10.** Получить число $1! + 2! + ... + 100!$, выводя его по одной цифре.

11.** Получить число $1! + 2! + ... 100!$, выводя по n цифр.

12. Вывести таблицу зависимости от n количества положительных делителей числа n.

13.* Напечатать пары дружественных чисел, не превосходящих n.

14. Найти наименьшее общее кратное четырех заданных натуральных чисел.

15.* Даны координаты вершин треугольника и точки внутри него. Найти расстояние от этой точки до ближайшей стороны.

16. Дано m n-мерных матриц. Выбрать из них матрицу с наименьшим следом.

17. Заменить члены последовательности a_i величинами a_i^i.

18.** Даны координаты вершин треугольника и точка. Определить, принадлежит ли эта точка треугольнику.

19.* Заменить полные квадраты в матрице их квадратными корнями, остальные элементы — нулями.

20. Найти периметр десятиугольника, заданного координатами вершин.

21.* Для четного числа $n > 2$ проверить гипотезу Гольдбаха.

22.* Составить функцию сжатия исходной последовательности символов (для повторяющихся символов ввести число повторений: *аааааа* → *а*(6)).

23.* Вычислить значение полинома над C по схеме Горнера.

24.* Составить функцию, заменяющую в тексте буквы их номерами по порядку в алфавите.

25.* Вычислить $n!$, используя рекурсию.

Лабораторные работы

Лабораторные работы по языку Си проводятся в компьютерном классе и состоят в реализации и отладке программ, написанных при решении задач. Каждому студенту предварительно предлагается 1—3 задачи из приведенного выше списка, которые он должен выполнить в системе программирования Borland C++.

Целесообразно проведение следующих лабораторных работ.

1. Знакомство с системой программирования Borland C++ 3.1. Ввод, сохранение, компиляция, отладка и запуск программ (4 часа).

2. Простейшие программы на Си. Выражения, ветвления. Форматный ввод-вывод (4 часа).

3. Циклы в программах на Си (4 часа).

4. Использование одномерных и многомерных массивов в программах на Си (4 часа).

5. Обработка строк (4 часа).

6. Определение функций программиста. Модульное программирование на Си (6 — 8 часов).

Дополнительная литература

1. *Берри Р.*, *Микинз Б.* Язык Си. Введение для программистов: Пер. с англ. — М.: Финансы и статистика, 1988.

2. *Болски М. И.* Язык программирования Си: Справочник: Пер. с англ. — М.: Радио и связь, 1988.

3. *Дансмур М.*, *Дейвис Г.* Операционная система UNIX и программирование на языке Си: Пер. с англ. — М.: Радио и связь, 1989.

4. *Джехани Н.* Программирование на языке Си: Пер. с англ. — М.: Радио и связь, 1988.

5. *Кернган Б.*, *Ритчи Д.*, *Фьюэр А.* Язык программирования Си. Задачи по языку Си: Пер. с англ. — М.: Финансы и статистика, 1985.

6. *Уэйт М.*, *Прата С.*, *Мартин Д.* Язык Си: Пер. с англ. — М.: Мир, 1988.

7. *Хенкок Л.*, *Кригер М.* Введение в программирование на языке Си: Пер. с англ. — М.: Радио и связь, 1986.

8. Языки программирования Ада, Си, Паскаль. Сравнение и оценка: Пер. с англ. / Под ред. А. Фьюэра и Н. Джехани. — М.: Радио и связь, 1989.

§ 4. ОСНОВЫ ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ ПРОЛОГ

Рекомендации по проведению занятий

Практикум по основам программирования на языке Пролог нацелен на освоение идейного арсенала дескриптивного и логического программирования, а также иллюстрирование понятий искусственного интеллекта, инженерии знаний.

Практикум предусматривает написание рефератов по Прологу, решение задач, семинарские и лабораторные занятия.

Краткие сведения

Пролог, родившийся, как и многие языки программирования, в начале 70-х годов, к настоящему времени так и не стал промышленным языком программирования: для написания прикладных систем Пролог используется крайне редко. В своем большинстве программисты-практики Пролога не знают или оценивают его невысоко. Еще в конце 80-х годов такие крупные фирмы программного обеспечения, выпускающие системы программирования, как Borland International,

MicroSoft прекратили развитие систем программирования на Прологе. Тем не менее, Пролог сохраняет свое значение в структуре подготовки специалиста по информатике, реализуя парадигму дескриптивного программирования, альтернативную процедурной и функциональной парадигмам. Пролог является прекрасной иллюстрацией метода представления знаний на основе продукций в интеллектуальных системах, с ним связаны очень большие ожидания в исследованиях по искусственному интеллекту. Есть уверенность в том, что идеи, реализованные в Прологе, будут в дальнейшем развиваться в промышленных интеллектуальных информационных системах, т. е. будущее этого языка программирования — впереди.

Программирование на Прологе включает в себя следующие этапы:

1) объявление фактов об объектах и отношениях между ними;

2) определение правил взаимосвязи объектов и отношений между ними;

3) формулировка вопроса об объектах и отношениях между ними.

Основная операция, выполняемая в языке Пролог, — это ***операция сопоставления*** (называемая также унификацией или согласованием). Операция сопоставления может быть успешной, а может закончиться неудачно, и определяется так:

1) константа сопоставляется только с равной ей константой;

2) идентичные структуры сопоставляются друг с другом;

3) переменная сопоставляется с константой или с ранее связанной переменной (и становится связанной с соответствующим значением);

4) две свободные переменные могут сопоставляться (и связываться) друг с другом; с момента связывания они трактуются как одна переменная: если одна из них принимает какое-либо значение, то вторая немедленно принимает то же значение.

Факты — это предикаты с аргументами-константами, обозначающие отношения между объектами или свойства объектов, именованные этими константами. Факты в программе считаются всегда и безусловно истинными и таким образом служат основой доказательства, происходящего при выполнении программы.

Правила — это хорновские фразы с заголовком и одной или несколькими подцелями-предикатами. Правила имеют форму

```
<голова правила> :- <список подцелей>
```

и позволяют определить новые отношения между объектами на основе уже объявленных с помощью фактов. В качестве аргументов в предикатах правила могут использоваться не только константы, но и переменные. На переменные в правилах действуют кванторы общности, поэтому правила концентрированно и лаконично выражают конструкции логического вывода.

Переменные в Прологе получают свои значения в результате сопоставления с константами в фактах и правилах.

Вопрос — отправная точка логического вывода, происходящего при выполнении программы. На любой вопрос компьютер будет пытаться дать ответ «Да» или «Нет» в зависимости от того, согласуется или нет утверждение, стоящее в вопросе, с фактами и правилами базы знаний. Вопрос, не содержащий переменных, является общим: «имеет ли место факт... ?».

Вопрос, в котором имеются переменные, является частным: «для каких значений переменных факт ... имеет место ?». В процессе сопоставлений при выполнении программы переменные конкретизируются — получат значения тех констант, для которых сопоставление запроса в целом успешно, и будут выведены на экран. Для интерпретатора Пролога существенны только совпадения и различия имен, а также связи между предикатами, устанавливаемые с помощью конъюнкций и импликаций. Однако в Прологе существуют предопределенные имена

(встроенные предикаты), которые позволяют выполнить арифметические операции, сравнения, графические построения, ввод-вывод и другие полезные операции как побочный продукт выполнения программы. Встроенные предикаты Arity-Prolog описаны в справке по системе программирования, вызываемой нажатием клавиши F1.

Аналогичный набор встроенных предикатов имеется в других версиях языка Пролог.

Существует целый класс задач, в которых отношения между объектами можно определить только пользуясь самими определяемыми соотношениями. Получающиеся при этом правила называются *рекурсивными*.

В системах логического программирования рекурсия служит также для описания циклов, повторений и является важнейшим методом программирования.

В общем виде рекурсия на Прологе выглядит так:

```
P(1,...).
P(n,...) :- Q1,..., Qn, P(n-1,...),R1,...Rm.
```

Правило *P* обращается само к себе, при этом происходит углубление рекурсии. Предикаты *Q*1, ..., *Qn* выполняются на прямом ходе рекурсии, а *R*1, ..., *Rm* — на обратном; *n* — это некоторый условный параметр, входящий в условие продолжения рекурсии, а *P*(1, ...) — факт, завершающий процесс рекурсии.

Особенно простым случаем рекурсии является простое циклическое повторение. Один из способов организации повторения связан с наличием в базе знаний процедуры вида

```
repeat.
repeat :- repeat.
```

Использование *repeat* в качестве подцели некоторого правила приводит к многократному повторению остальных подцелей этого правила.

Управление процессом просмотра предложений является важным аспектом программирования на Прологе. Это осуществляется с помощью специальной встроенной функции «*резать*», обозначаемой символом «!».

Данная встроенная функция может быть использована для достижения следующих трех целей:

1) исключения бесконечной петли при выполнении программы;
2) программирования взаимоисключающих утверждений;
3) блокирования просмотра целей.

На практике часто встречаются задачи, связанные с перечислением объектов. В некоторых случаях при решении задач важно сохранять информацию об уже сделанных шагах решения, чтобы их не повторять. Для решения таких задач в языке Пролог предусмотрены *списки*.

Список можно задать перечислением элементов. Например, имена учеников класса:

```
[саша,петя,дима,ксюша,лена].
```

Элементами списка могут быть не только атомы, но и функции и вообще любые элементы, даже списки. Заранее длина списка не задается, и в ходе выполнения программы она может меняться.

Альтернативный способ задания списка использует понятия *головы* и *хвоста списка*. Например, в списке `[X|Y]` `X` — это голова списка, а `Y` — его хвост. Хвост списка по определению также является списком.

Теперь список может быть определен рекурсивно:

1) пустой список `[]` — список;

2) `[X|Y]` — список, если `Y` — список.

Определение списка через его голову и хвост в сочетании с рекурсией лежит в основе большого количества программ, оперирующих списками. Эти программы состоят:

1) из факта, ограничивающего рекурсию и описывающего операцию для пустого списка;

2) из рекурсивного правила, определяющего операцию над списком, состоящим из головы и хвоста (в голове правила), через операцию над хвостом (в подцели).

Арифметические операторы. Арифметическое выражение состоит из функтора и его аргументов. *Функтор* — арифметическая операция; *аргументы* — целые или переменные, они являются операндами для функтора. Например: `+(X,1)`.

Пролог позволяет использовать инфиксную нотацию. Например: `X+1`.

Можно использовать арифметические выражения вида: `X+Y`, `X-Y`, `X*Y`, `X/Y` (обычное деление, результат – действительное число), `X//Y` (целое деление, результат — целое число), `X^Y` (возведение в степень); `- X`;

`X/\Y` (конъюнкция; только int) (`AND`),
`X\/Y` (дизъюнкция; int) (`OR`),
`\(X) - NOT` (только int),
`X<<Y` — смещение `X` влево на `Y` позиций (`int`),
`X>>Y` — смещение `X` вправо (`int`),
`[X]` — оценить (приблизительное вычисление) `X`,
`X mod Y` — выдает остаток от деления `X` на `Y`,
`abs(X)` — абсолютное значение (`X`),
`acos(X) – arccos (X) asin (X) – arcsin(X) atan(X) – arctg(X),`
`cos(X), exp(X), ln(X), log(X), sin(X), sqrt(X), tg(X),`
`round(X, N)` — округление `X` до `N` символов после запятой,
(`N<=15`).

Предикаты сравнения (`E1` и `E2` — арифметические выражения). `E1>E2` проверяет, больше ли значение `E1` значения `E2`; аналогично: `E1<E2`, `E1>=E2`, `E1<=E2`;

`X is E1` — `E1` вычисляется и сопоставляется с `X`;
`E1=:=E2` — вычисляются `E1` и `E2` и проверяются на эквивалентность;
`E1=\=E2` — вычисляются `E1` и `E2` и проверяются на неравенство;
`randomise (+Seed)` — переустановка генератора случайных чисел (аргумент `Seed` — целый);
`inc (+X,-Y)` — увеличение целого `X` и возврат в `Y`;
`dec (+X,-Y)` — уменьшение целого `X` и возврат в `Y`.

Контрольные вопросы

1. Каковы отличия программы на Прологе от программ на процедурных языках программирования?
2. Какова сфера применения языка Пролог?
3. Каковы основные элементы программы на Прологе?
4. Какие типы данных используются в Прологе?
5. Что такое *имена*? *переменные*?
6. Что такое *факт*, что он выражает? Приведите примеры.

7. Что такое *правило*, что оно выражает? Приведите примеры.
8. Какова роль *вопроса* в программе на Прологе?
9. Что такое *сопоставление*?
10. Что такое *конкретизация переменных*?
11. Какая стратегия логического вывода реализована в Прологе?
12. Какова роль рекурсии в программах на Прологе?
13. Для чего служат *списки* в программах на Прологе?
14. Каким образом можно управлять процессом логического вывода при решении задачи на Прологе?

Темы для рефератов

1. Парадигма логического программирования — альтернатива процедурному и функциональному программированию.
2. Пролог и продукционное представление знаний.
3. Прикладная логика предикатов 1-го порядка и фразы Хорна.
4. Логическая природа решения задач на Прологе.
5. Примеры баз знаний на Прологе.
6. Моделирование интеллектуальной деятельности человека и Пролог.
7. Рекурсия — основной метод программирования на Прологе.
8. Управление логическим выводом. Отсечение.
9. Списки и структуры в решении логических задач.
10. Эволюция систем программирования на Прологе.

Темы семинарских занятий

1. Моделирование интеллектуальной деятельности человека и представление знаний с помощью продукций.
2. Основы логики предикатов 1-го порядка и фразы Хорна.
3. Предложения программы на Прологе: *факты*, *правила*, *вопрос*.
4. Стратегия логического вывода Пролога. Согласование. Унификация переменных.
5. Примеры баз знаний на Прологе. Встроенные предикаты.
6. Рекурсия в поиске решения логических задач. Использование *списков*.
7. Перспективы развития и применения Пролога.

Рекомендации по программному обеспечению

Арсенал систем программирования на Прологе небогат. Наибольшую известность получили системы TurboProlog 1.0 и 2.0 для IBM PC в среде MS-DOS, выпущенные компанией Borland International в конце 80-х годов. В целях повышения эффективности объектного кода входной язык этих систем значительно отличается от стандартного Пролога (DEC-10), в частности, необходимостью описания предикатов, так что этот язык следует считать самостоятельным языком логического программирования, поэтому использование этих систем при проведении практикума нежелательно.

Полностью соответствует стандарту Пролога входной язык системы программирования ArityProlog версий 5.0, 6.0. Для выполнения практикума наиболее подходит интерпретатор ArityProlog api.exe. Имеется русифицированная версия интерпретато-

ра ArityProlog 5.0. Прекрасным вариантом программного обеспечения для обучения Прологу остается Пролог-Д (разработанный под руководством С. Г. Григорьева) для КУВТ Yamaha-2 или учебная система Prologus для IBM PC (разработанная под руководством Д. П. Федюшина). Установка этих систем трудностей не вызывает.

Запуск **интерпретатора ArityProlog** производится следующим образом:

```
C> api.exe <Ввод>.
```

На экране дисплея будет открыто главное окно интерпретатора и строка меню, содержащая опции

```
File   Edit   BuffersInfo   Debug     Swich     Help
```

Курсор помещается в главном окне. Приглашение интерпретатора (?—) также появится в нем.

Укажем основные способы перемещения между различными элементами интерпретатора Arity/Prolog:

• из главного окна в меню — **F10**. Каждый элемент в строке меню также можно выбрать с помощью горячих клавиш (**Alt** + в имени элемента меню выделена цветом, отличным от остальной части имени), при этом откроется спускающееся меню, если оно есть;

• из меню в главное окно — **Esc**;

• из главного окна в редактор или из редактора в главное окно можно перейти нажатием клавиши **F8**;

• выход из интерпретатора: войдите в меню, выберите с помощью курсора элемент меню **File**. Нажмите клавишу **Enter** (появится выпадающее меню для **File**). Выберите с помощью клавиш управления курсором опцию **Halt** и нажмите клавишу **Enter**. Если вы редактировали какие-либо файлы, то система спросит вас, хотите ли вы их сохранить перед выходом;

• текст программы на Прологе можно вводить и редактировать, выбрав опцию **Open**... из подменю **File**. Процедура переходов между редактором и меню такая же, как между главным окном и меню. Нажимая **Esc**, вы переместите курсор в окно (главное или редактора), в котором вы находились перед этим;

• в главном меню вы можете переходить от одного элемента меню к другому, нажимая клавиши управления курсором «влево» (←) и «вправо» (→). Многие элементы меню включают выпадающее меню. Выпадающее меню предлагает выбор опций, соответствующих элементу меню;

• вы можете двигаться по элементам выпадающего меню, нажимая клавиши со стрелками вверх и вниз. Вы можете также передвинуться в другое спускающееся меню нажатием клавиш → (←). В этом случае вы можете видеть спускающееся меню, соответствующее каждому элементу строки меню.

Некоторые элементы меню, такие как опция **Merge File** в выпадающем меню **File**, имеют цвет, отличный от других элементов меню. Серый цвет этих элементов указывает, что они не могут быть выбраны, так как эти элементы предназначены для использования их в редакторе ArityProlog.

Если за элементом меню следует многоточие (...), это указывает, что при выборе этого элемента появляется окно диалога. Окно диалога содержит дополнительную информацию или операции, которые следует задать.

Функциональные клавиши. Определенные действия можно выполнить простым нажатием одной из десяти функциональных клавиш на клавиатуре. Каждая функциональная клавиша соответствует некоторой опции меню. Функциональные клавиши и соответствующие им операции представлены в табл. 3.3.

Таблица 3.3

Функциональная клавиша	Операция
F1	Помощь
F2	Копирует отмеченный текст
F3	Повторяет последний поиск при нахождении строки в тексте
F4	Нахождение строки в файле, в окне диалога надо указать строку поиска
F5	Осуществляет замену найденной строки на новую строку (замена осуществляется от текущей позиции курсора до конца файла)
F6	Выбор буфера
F7	Переход в предыдущий активный буфер из текущего буфера
F8	Переход из главного окна в окно редактора и наоборот
F9	Не определено
F10	Активизирует главное меню

Использование диалогового окна дает дополнительную информацию системе для того, чтобы произвести действие, соответствующее выбранной опции меню. Диалоговое окно состоит из одного или более элемента управления. Элемент управления в диалоговом окне используется для ввода информации. Перемещаться по диалоговому окну можно с помощью клавиши **Tab**.

Например, рассмотрим диалоговое окно при выборе опции **Open File**...:

1. Нажатием **Alt+F** перейдем в меню **File**. Появится выпадающее меню.
2. Нажатием клавиши **O** выберем опцию **Open File**...

В следующем диалоговом окне запрашивается имя файла, который надо открыть. В окне списка диалогового окна появятся имена всех файлов с расширением **ari**. В ArityProlog расширение **ari** обычно используется для наименования файлов, содержащих тексты программ. Если вы хотите, чтобы в окне списка появился файл из другой директории или с другим расширением, напечатайте в редактируемом поле имя файла с путем к нему или явно укажите расширение. Например, если вы хотите, чтобы в окне списка появились все файлы из директории **Prolog** с расширением **pgm**, вы должны напечатать в редактируемом поле **\prolog*.pgm**.

Диалоговое окно имеет кнопки управления ***OK*** и ***Cancel***. Эти управления используются для выхода из диалогового окна, но ***OK*** подтверждает все действия, а ***Cancel***, наоборот, отменяет их. То же самое можно сделать, нажав клавишу **Esc**, находясь в диалоговом окне. Одна из кнопок ***OK*** или ***Cancel*** окружена двойной рамкой. Это действие будет выполнено, если выйти из диалогового окна, нажав клавишу **Enter**. Например, если ***OK*** окружено двойной рамкой, то нажатие <**Enter**> после выбора имени файла приведет к выходу из диалогового окна и выбору файла.

Опции меню File. Меню File содержит следующие опции в выпадающем меню.

New — размещает новый пустой файл в открытом файловом буфере редактора Arity. Если файловый буфер не открыт, то система спрашивает, хотите ли вы загрузить новый файл и очистить содержимое текущего буфера.

Open File... — открывает файл, имя которого можно задать, выбрав эту опцию в появившемся окне.

Merge File... — добавляет содержимое файла после текущей позиции курсора. Эта опция вызывает окно, в котором можно указать, с каким файлом будет слияние.

Save File — запись файла из текущего буфера под текущим именем. Если имя не задано, то опция предлагает окно, в котором можно задать имя.

Save File as... — запись файла с новым именем из текущего буфера. Эта опция вызывает диалоговое окно, в котором можно указать имя файла, под которым файл будет сохранен.

Consult File... — Consult или reconsult добавляет файлы во внутреннюю базу данных (БД) Пролога. Эта опция вызывает диалоговое окно, в котором можно указать имя файла, который надо добавить в БД Пролога.

Restore Db — восстановление (перезапись) БД, которая была записана в последний раз с помощью опции Save Db. Таким образом, используя эту опцию, можно уничтожить все изменения базы, которые были сделаны при последней записи БД.

Restore Db from... — восстановление предыдущей версии БД из файла. Эта опция отображает диалоговое окно, в котором можно указать имя файла с БД, которая будет восстановлена.

Save Db — записывает текущее состояние БД Пролога.

Save Db as... — запись состояния БД Пролога с новым именем. Эта опция вызывает диалоговое окно, в котором можно выбрать имя для состояния программы, которое надо записать. Файл внутренней базы с расширением idb будет создан.

Shell — временное переключение Arity в операционную систему MS DOS или OS/2. Чтобы вернуться в интерпретатор ArityProlog, надо напечатать Exit в командной строке после приглашения.

Halt... — выход из Arity. При этом появляется окно со списком файлов, которые были отредактированы, но не сохранены.

Опции меню Edit. Меню Edit включает опции для операций редактора. Выпадающее меню Edit включает следующие опции.

Find... — нахождение строки в файле. Опция вызывает диалоговое окно, в котором можно задать строку для поиска. Опция может быть выбрана нажатием функциональной клавиши F4.

Find selected — эта опция также ищет строку в файле, однако, в отличие от опции Find, при ее выборе не появляется диалоговое окно. Чтобы использовать эту опцию, сначала надо выбрать текст в главном окне, используя Shift и клавиши со стрелками. После выделения текста и выбора опции система будет просматривать текст и находить следующее вхождение выбранного текста. Эту опцию также можно выбрать нажатием Ctrl + \.

Repeat Last Find — эта опция повторяет последнюю выполнявшуюся опцию Find... или Find Selected. Эту опцию также можно выбрать нажатием функциональной клавиши F3.

Change... — используется для поиска и замены отмеченного текста на новый. В диалоговом окне задаются текст для поиска и текст для замены. В диалоговом окне также можно указать, надо ли, чтобы система запрашивала подтверждение о замене, когда найден текст, или надо выполнять замены автоматически, без подтверждения. Эту опцию также можно выбрать нажатием функциональной клавиши F5.

Undo — восстановление строки текста в состояние, в котором она находилась, когда курсор впервые попал на эту строку. Например, если удалить слово из файла и впечатать новое слово, с помощью опции Undo можно восстановить строку в виде, в котором она была до удаления и добавления. Эту опцию также можно выбрать нажатием Ctrl + r.

Cut — вырезает отмеченный текст. Текст помещается в Clipboard (буфер вырезанного изображения). Эту опцию также можно выбрать нажатием Shift + Del, когда текст выбран.

Copy — копирует выбранную часть текста. Копия размещается в Clipboard. Эту опцию можно выбрать нажатием функциональной клавиши F2, если текст отмечен.

Paste — вставка содержимого Clipboard после текущей позиции курсора. Эту опцию также можно выбрать нажатием Shift + Ins.

Clear — удаляет отмеченную часть текста. Текст, который удаляется с помощью опции Clear, не сохраняется в Clipboard и поэтому не может быть восстановлен. Эту опцию можно выбрать нажатием Del, если текст выделен.

Опции меню Buffers. Редактор ArityProlog содержит 9 различных буферов (рабочих областей). Дополнительно в буфере редактора имеется Clipboard (буфер вырезанного изображения). Текущим может быть только один буфер, и все опции меню относятся к текущему буферу. Опции этого меню позволяют выбрать определенный буфер и назначать различные атрибуты для текущего буфера. Спускающееся меню Buffer включает следующие опции.

Go To — позволяет сделать активным определенный буфер. Редактировать текст можно только в активном буфере. При выборе этой опции появляется диалоговое окно, в котором можно выбрать буфер, сделав его активным. Эту опцию также можно выбрать нажатием функционального ключа F6.

Go To Last — позволяет вернуться в предыдущий активный буфер. Таким образом, выбрав эту опцию, можно быстро переключаться между двумя буферами. Эту опцию также можно выбрать нажатием функциональной клавиши F7.

Erase Buffer — очищает текущий буфер.

Save All Buffers — если выбрать этот элемент меню, все текущие буферы редактора сохраняются.

Reconsult Buffer — запись текущего буфера во внутреннюю базу данных Пролога.

Save On Exit — указывает, что необходимо записать содержимое буфера при выходе из него. При выборе этой опции в меню появится маркер. Выбор осуществляется нажатием клавиши Enter.

Reconsult On Exit — указывает (маркером), что содержимое текущего буфера Пролога надо поместить во внутреннюю базу данных при выходе из редактора. Выбор/отмена данной опции — Enter.

Indent — указывает, что при наборе текста в текущем буфере надо делать смещение вправо. Если выбрана эта опция, то курсор при переходе на следующую строку будет переходить в позицию, с которой начиналась предыдущая строка. Выбор/отмена — Enter внутри подменю Buffers.

Read Only — назначает текущий буфер только для чтения. Можно только копировать текст из данного буфера. Выбор/отмена — Enter.

Редактор. Можно использовать Arity-редактор для редактирования программ и других файлов. Редактор имеет 9 рабочих буферов и Clipboard. Таким образом, можно одновременно редактировать 9 различных файлов и обмениваться между ними, используя для связи Clipboard.

Активизировать редактор можно следующими способами:

- используя команду **Switch** вы автоматически попадаете в 1-й буфер редактора;
- используя **New** в меню **File** вы также попадаете в 1-й буфер;
- **Open File...**

Существует три режима использования редактора:

1) **Insert** — вставка;

2) **Overtype** — замена;
3) **ReadOnly** — только чтение.

Для переключения между **Insert** и **Overtype** используется клавиша **INS**.

Для задания режима «только для чтения» **ReadOnly** надо:

- войти в нужный буфер;
- в меню **Buffers** выбрать **ReadOnly** (верно для текущего буфера).

Перемещение курсора. Клавиши со стрелками — перемещение на один символ влево/вправо или на одну строку вверх/вниз, в сочетании с клавишей Control (^) — на одно слово (из латинских букв) влево/вправо.

Home — в начало строки (физическое).

End — в конец строки (логический).

PgUp — на страницу вверх.

PgDn — на страницу вниз.

^Home — в начало текста.

^End — в конец текста.

^G — появляется окно, в котором можно задать число строк, на какое надо спуститься вниз.

Удаление, копирование и восстановление.

Backspace — удаление предыдущего символа.

Del — удаление текущего символа.

Shift + <клавиша со стрелкой> — выделение текста для удаления, копирования. Выделение производится от текущей позиции курсора.

После выделения можно выполнить следующие действия:

1) **Shift + Del** (=**Cut** в меню Edit) — уничтожение и перемещение в Clipboard;
2) **F2** (= **Copy** в меню Edit) — копирование и перемещение в Clipboard;
3) **Del** (= **Clear** в меню Edit) — удаление, в Clipboard текст не заносится.

Использование Clipboard. Заметим, что Clipboard содержит только последний «удаленный» текст из буфера.Таким образом, если надо восстановить из Clipboard текст, который был уничтожен в буфере, надо восстановить его до следующего удаления.

Shift + Ins (= **Paste** в меню Edit) — помещает содержимое Clipboard в текст после текущей позиции курсора.

По пустой строке перемещение осуществляется клавишей пробела.

^R (= **Undo** в меню Edit) — откатка. Действует в пределах строки, т.е., если произвели какие-либо изменения, а затем перешли на другую строку, даже вернувшись обратно, вы не восстановите текст. После откатки курсор переходит на начало строки.

Переключение буферов.

F6 (= **Go To...** в меню Buffers) — в диалоговом окне можно задать либо номер буфера, либо выбрать буфер, перемещаясь по списку, используя Tab и клавиши со стрелками. После выбора нажать Enter.

F7 (= **Go To Last** в меню Buffers) — переход в предыдущий активный буфер. Таким образом, между двумя буферами можно переключаться, используя эту опцию.

Существуют опции, которые действуют только в пределах одного буфера — это опции подменю Buffers. Они выбираются с помощью Enter (появится маркер). Таким образом, для каждого буфера можно задать свои опции.

Загрузка файла с примером программы. Для того чтобы запустить программу, необходимо загрузить ее во внутреннюю базу данных Пролога. Это делается выбором опции **Consult** в подменю **File...**

В появившемся диалоговом окне надо указать имя файла, который следует загрузить. В диалоговом окне существуют три опции:

1) **Consult**;
2) **Reconsult**;
3) **Cancel**.

Чтобы загрузить файл в базу Пролога, надо выбрать Reconsult.

Ошибки. Если попытаться загрузить в базу данных файл, в котором есть синтаксические ошибки, он не загрузится, и появится сообщение об ошибке. Угловые скобки << >> будут указывать место, где встретилась ошибка.

Для исправления ошибки надо вернуться в редактор.

Запуск Prolog-программы. Войти в интерпретатор после загрузки программы можно, выбрав опцию Switch в меню одним из способов:

1) **F8**;
2) **Alt + S**, **Enter**;
3) **Esc**, на Switch **Enter**, **Enter**.

Переключившись в главное окно, вы увидите приглашение (?–), которое означает, что интерпретатор запущен и готов к действию.

Типы запросов. Запросы могут быть двух видов:

1) цель достигнута (**succeed**);
2) цель не достигнута (**fail**).

Если надо продолжить поиск в базе по этому запросу, то используйте «;», после «;» нажимать Enter не надо. Если не надо — то без «;» нажмите Enter.

Каждая следующая цель удовлетворяется в зависимости от предыдущей цели.

Прерывание выполнения запроса. Если не надо, чтобы запрос, который уже набран, выполнялся (например, следует его изменить), используйте **^C**.

Задачи и упражнения

1. Опишите на Прологе:
   а) свою родословную, определите бабушек, дедушек, прабабушек, прадедушек и т.д.;
   б) телефонную книгу;
   в) районы вашего города, республики, области, укажите численность их населения, местные достопримечательности;
   г) европейские государства (население, площадь и т.д.);
   д) таблицу дат и событий русской истории;
   е) небольшой словарь для перевода с русского языка на иностранный язык, который вы изучаете;
   ж) ведомость зачета вашей группы;
   з) успеваемость вашей группы; дайте определение «отличник»;
   и) каталог книг в библиотеке.
2. Запишите на Прологе правила, являющиеся решением следующих заданий:
   а) даны два числа a и b, получите их сумму, разность, произведение;
   б) дана длина ребра куба, найдите объем куба и площадь его боковой поверхности;
   в) дан радиус основания r и высота цилиндра h, найдите его объем и площадь боковой поверхности;

г) даны стороны *a* и *b* параллелограмма, а также угол между ними, найдите диагонали параллелограмма и его площадь.

3. Вычислите значения выражений:
   а) $2x + 3y + 4$; б) $(2x + 8y + 4)/2$;
   в) $y - x^2$; г) $x^2 + xy + y^2$;
   д) $x/2 + 5y$; е) $x^2 + 3y^2$;
   ж) $5(34x - y)$.

4. Напишите программы, выполняющие следующие операции над списками:
   а) объединить два списка, найти максимальный элемент и удалить его;
   б) удалить из списка элемент, найти длину оставшегося списка;
   в) добавить к списку элемент, вычислить среднее арифметическое его элементов;
   г) обратить список, найти последний и предпоследний элементы;
   д) исключить из списка заданный элемент во всех вхождениях, кроме первого, найти длину оставшегося списка;
   е) проверить, имеются ли в списке повторяющиеся элементы, и все их удалить;
   ж) удалить из списка все элементы, равные последнему, найти длину оставшегося списка;
   з) объединить два списка, найти MIN и удалить его;
   и) обратить список, найти MAX и удалить его;
   к) к списку добавить обращенный 2-й список, найти длину результата;
   л) отсортировать список, используя метод пузырька.

5. Напишите программу, решающую задачу о волке, козе и капусте.

6. Напишите программу, решающую задачу о трех туземцах, трех миссионерах и двухместной лодке.

7. Напишите программу, решающую задачу об обходе препятствия.

8. Напишите программу, определяющую положение «шах королю».

9. Напишите программу, определяющую, как шахматному коню попасть с поля *A* на поле *B*.

10. Напишите программу, определяющую, как разлить 10 л молока по 5 л, пользуясь бидонами на 3, 7 и 10 л.

11. Напишите программу, аналитически дифференцирующую элементарную функцию.

12. Напишите программу, играющую в «крестики и нулики» на бесконечной плоскости.

13. Напишите программу, вычисляющую интервал между двумя датами одного года, например 7 марта и 9 сентября.

14. Напишите программу, составляющую частотный словарь вводимого текста.

15. Напишите программу, решающую задачу о четырех ферзях на поле размером 4×4 клетки.

16. Напишите программу, строящую латинский квадрат 3×3.

Лабораторные работы

При проведении практикума по логическому программированию на языке Пролог целесообразно проведение следующих лабораторных работ:

1. Знакомство с системой программирования на языке Пролог. Ввод, сохранение, отладка и исполнение простейшей программы (4 часа).

2. Разработка простых баз знаний на Прологе (4 часа).

Основой для проведения этой лабораторной работы может стать задача 1 из приведенных выше.

3. Встроенные предикаты Пролога, арифметические выражения и сравнения (4 часа).

Основой для проведения этой лабораторной работы могут стать задачи 2, 3 из приведенных выше.

4. Рекурсия в Прологе. Обработка списков (4 часа).

Основой для проведения этой лабораторной работы может стать задача 4 из приведенных выше.

5. Решение логических задач на Прологе (8 часов).

Примерами заданий, которые могут быть предложены для этой лабораторной работы, являются задачи 5 — 16 из приведенных выше. Отметим, что это наиболее сложная лабораторная работа, с которой самостоятельно могут справиться лишь немногие студенты.

Дополнительная литература

1. *Братко И.* Программирование на языке Пролог для искусственного интеллекта: Пер. с англ. — М.: Мир, 1990.

2. *Доорс Дж., Рейблейн А.Р., Вадера С.* Пролог — язык программирования будущего: Пер. с англ. — М.: Финансы и статистика, 1990.

3. *Иванова Г.С., Тихонов Ю.В.* Введение в Пролог. — М.: Издательство МГТУ, 1990.

4. *Клоксин У., Меллиш К.* Программирование на языке Пролог: Пер. с англ. — М.: Мир, 1987.

5. *Ковальски Р.* Логика в решении проблем: Пер. с англ. — М., 1990.

6. *Малпас Дж.* Реляционный язык Пролог и его применение: Пер. с англ. — М.: Наука, 1990.

7. *Стерлинг Л., Шапиро Э.* Искусство программирования на языке Пролог: Пер. с англ. — М.: 1990.

8. *Тей А., Гриболон П., Луи Ж. и др.* Логический подход к искусственному интеллекту: От классической логики к логическому программированию: Пер. с фр. — М.: Мир, 1990 .

9. *Хоггер К.* Введение в логическое программирование: Пер. с англ. — М., 1988.

§ 5. ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Рекомендации по проведению занятий

На семинарских занятиях полезно разобрать алгоритмы приведенных ниже упражнений, использующих тип «объект». Практические занятия разумно посвятить разбору примеров и задач, выполнению практических упражнений с пошаговой трассировкой предлагаемых образцов программ, с последующей их модификацией. Рекомендуется каждому студенту выполнить самостоятельно все задания, приведенные в конце каждой темы.

Темы семинарских занятий

1. Тип «объект» в Турбо-Паскале. Примеры создания объектов с наследованием.
2. Примеры работы с числовыми объектами.
3. Примеры работы с объектами графического типа.
4. Оболочка Turbo-Vision.
5. Приемы работы в среде Дельфи.
6. Проектирование программ в среде Дельфи.
7. Программирование в системе Си++.

Рекомендации по программному обеспечению

1. Турбо-Паскаль 6, 7.
2. Дельфи 3,5.
3. Си++.

Краткие сведения

Средства объектно-ориентированного программирования в Паскале. Объекты в Турбо-Паскале

Объект — это структура данных, содержащая поля данных (аналогично типу *record* — запись) различных типов и заголовки методов. *Методы* — это процедуры и/или функции, объявленные и действующие внутри объекта.

Пример:

```
type
   Point = object
      X,Y : integer;
      Visible : boolean;
      procedure Create (a,b: integer);
      procedure SwitchOn;
      procedure SwitchOff;
      procedure Move (dx,dy : integer);
      function GetX : integer;
      function GetY : integer;
   End;
```

Наследование — определение нового объекта с использованием свойств ранее объявленных объектов, дополняя или изменяя их.

Пример:

```
type
   Star = object(Point)
      procedure Create (a,b: integer);
      procedure Move (dx,dy : integer);
   End;
```

Связь между методами и вызывающими их процедурами может устанавливаться во время трансляции (раннее связывание) или во время выполнения программы (позднее связывание). Такие методы должны быть *виртуальными* (ключевое слово — «virtual»). Инициализация экземпляра объекта, имеющего виртуальный метод, должна выполняться с помощью специального метода — *конструктора*. Заголовок конструктора начинается словом `constructor` вместо `procedure`. Действия, обратные действиям конструктора, выполняет специальный метод — *деструктор* (заголовок — «destructor»).

Слово `privat` позволяет ограничить доступ к полям объекта, оставляя возможность использовать их лишь через методы этого объекта.

Переменные объектного типа могут быть *динамическими*, то есть размещаться в памяти только на время их использования. Для работы с динамическими объектами используется расширенный синтаксис процедур `New` и `Dispose`, содержащих в качестве второго параметра вызов конструктора или деструктора: `New (P, Conctructor), Dispose(P, Destructor)`. Эквивалентные операторы: `New (P); P^.Constructor;` и `P^.Dispose; Dispose(P);`.

Полиморфизм — следствие наследования, возможность использования одного имени для различных действий в зависимости от типа объекта.

Тип «объект» в Паскале тесно связан с модульным программированием. В этой связи вспомним основные понятия модуля. Модуль в Паскале — это самостоятельная программная единица, объединяющая совокупность программных ресурсов, предназначенных для использования другими модулями и программами. Структура модуля представлена ниже:

```
Unit <UnitName>;
Interface
```

Описания видимых объектов

```
Implementation
```

Описания скрытых объектов

```
Begin
```

Операторы инициализации объектов модуля

```
End.
```

Каждый модуль компилируется отдельно с созданием файла с расширением .tpu. Доступ к интерфейсным объектам модуля осуществляется спецификацией

```
uses UnitName;
```

Как правило, в интерфейсной части модуля определяют объектные типы. В разделе `implementation` осуществляется реализация методов объектов.

Пример.

```
Unit matrices;
interface
      uses crt;
```

```
    const
        max = 3;
    type
        str10 = string[10];
        matrix = array[1..max,1..max] of pointer;
        complex = record
            re, im : real;
        end;
    complex_ptr = ^complex;
    complex_matrix = array[1..max, 1..max] of complex;
    matrix_obj = object
        M : matrix;
        constructor init;
        procedure set_to_zero(var p:pointer);
            virtual;
        procedure scalar_sum(const p,q: pointer; var r:
                                    pointer);
            virtual;
        procedure scalar_product(const p,q: pointer;
                                    var r : pointer);
            virtual;
        procedure matrix_sum(const A,B: matrix);
        procedure print(name : str10);
        destructor done;
    end;
Implementation
    constructor matrix_obj.init;
        begin
        end;
    procedure matrix_obJ.matrix_product(const A,B: matrix);
        var i,j,k: word; prod, sum: pointer;
        begin
            getmem(prod, SizeOf(M[1,1]));
            for i:=1 to max do for k:=1 to max do
            begin
                sum:= M[i,k];
                set_to_zero(sum);
                for J:=1 to max do
                begin
                    scalar_product(A[i,J], B[J,k], prod);
                    scalar_sum(sum, prod, sum);
                end
            end;
                freemem(prod,SizeOf(M[1,1]));
        end;
...
destructor matrix_obJ.done;
        begin
        end;
End.
```

Задачи и упражнения

Объекты в Турбо-Паскале

Упражнение № 1. Знакомство с типом «объект». Работа с модулем, содержащим объект «звезда»

Рассмотрим модуль Unplan1. В нем определен объект Pozition, позволяющий инициализировать положение точки на экране дисплея, и объект-потомок Star, наследующий свойства объекта Pozition. Методы объекта Star позволяют устанавливать точку по заданным координатам, с заданным цветом, перемещать и гасить ее. Ниже приведен полный листинг модуля. Запишите этот модуль в виде файла с именем *Unplan1.pas*.

```
Unit Unplan1;
Interface
   uses Graph, Crt;
   type Pozition = object
      X,Y: Integer;
      procedure Init(CoordX, CoordY: Integer);
      function GetX: Integer;
      function GetY: Integer;
   end;
Star = object (Pozition)
      Visible: Boolean;
      Color: Word;
      constructor Init(CoordX, CoordY: Integer; InitColor:Word);
      destructor Done; virtual;
      function IsVisible: Boolean;
      procedure Show; virtual;
      procedure Blind; virtual;
      procedure Jump (NextX, NextY: Integer); virtual;
   end;
Implementation
{ ** ******** Методы объекта Pozition *************** }
procedure Pozition.Init(CoordX, CoordY: Integer);
   begin
      X:=CoordX; Y:=CoordY;
   end;
function Pozition.GetX: Integer;
   begin
      GetX:=X;
      end;
function Pozition.GetY: Integer;
   begin
      GetY:=Y;
   end;
{ ** ******** Методы объекта Star ****************** }
constructor Star.Init(CoordX, CoordY: Integer; InitColor: Word);
   begin
      Pozition.Init(CoordX, CoordY);
```

```
        Visible:= False; Color:= InitColor;
    end;
destructor Star.Done;
    begin
        Blind;
    end;
procedure Star.Show;
    begin
        Visible:= True; PutPixel(X,Y,Color);
    end;
function Star.IsVisible: Boolean;
    begin
        IsVisible:= Visible;
    end;
procedure Star.Blind;
    begin
        Visible:= False; PutPixel (X, Y, GetBkColor);
        {точка закрашивается фоновым цветом}
    end;
procedure Star.Jump ;{(NextX,NextY:integer);}
    begin
        Blind; X := NextX; Y := NextY; Show;
    end;
{секции инициализации нет}
End.
```

Пример 78. *Программа «блуждающая звезда».*

В нижеприведенной программе используется объект `Star` из подключаемого модуля `Unplan1.tpu`. Предполагается, что модуль был предварительно откомпилирован. После инициализации и установки точки (переменная `Astar`) в цикле происходит гашение и установка точки с новыми координатами, определяемыми случайным образом.

```
Program Example_78, Star1;
    Uses Crt, Graph, Unplan1;
    Var GraphDriver,GraphMode,ErrorCode: Integer;
        AStar: Star;
        i: integer;
begin
    Randomize; GraphDriver:=Detect;
    InitGraph(GraphDriver,GraphMode,'');
    if GraphResult <>GrOk then Halt(1);
        Astar.Init(200,100,15); {инициализация точки}
        Astar.show; readln; {установка точки до нажатия ввода}
for i:=1 to 10 do
    begin
        Astar.blind; {гашение точки}
        Astar.Jump(50+Random(400),50+Random(300));
        Astar.show; readln;
        {установка точки в новом случайном месте}
    end;
```

```
    AStar.Done;
    CloseGraph;
End.
```

Пример 79. *Программа «звездное небо с мигающими звездами».*

С использованием модуля `Unplan1.tpu` в программе задается массив экземпляров объекта `Star`.

```
Program Example_79, Star2;
Uses Crt, Graph, Unplan1;
Var GraphDriver,GraphMode,ErrorCode: Integer;
    AStar: array[1..200] of Star;
    i,s: integer;
begin
    GraphDriver := Detect; InitGraph(GraphDriver,GraphMode,' ');
    if GraphResult <>GrOk then Halt(1);
    Randomize;
    for i:=1 to 200 do {начальная установка 200 звезд}
        begin
            Astar[i].Init(50+random(350),50+random(250),15);
            Astar[i].show;
        end;
    readln;
    repeat
        s:=random(199)+1; Astar[s].blind; delay(100); Astar[s].show;
    Until Keypressed; readln;
    for i:=1 to 200 do AStar[i].Done;
    CloseGraph;
End.
```

Пример 80. *Программа «звездное небо с пролетающим метеоритом».*

В алгоритм предыдущей программы добавлен экземпляр `Bstar`, который двигается прямолинейно до границ рассматриваемой области.

```
Program Example_80, Star3;
Uses Crt, Graph, Unplan1;
Var GraphDriver, GraphMode, ErrorCode: Integer;
    AStar: array[1..200] of Star;
    BStar: Star;
    i,s: integer;
begin
    GraphDriver := Detect; InitGraph(GraphDriver,GraphMode,'');
    if GraphResult <>GrOk then Halt(1);
    Randomize;
    for i:=1 to 200 do {начальная установка 200 звезд}
        begin
            Astar[i].Init(50+random(350),50+random(250),green);
            Astar[i].show;
        end;
    BStar.Init(100,60,white);
    BStar.show; {начальная установка метеорита}
    readln;
```

```
    i:=1;
    repeat
       s:=random(199)+1; Astar[s].blind; delay(200); Astar[s].show;
       BStar.Jump(100+i,60+i); BStar.show;
       i:=i+1;
    until (Keypressed) or (BStar.GetX>400) or (BStar.GetY>300);
    {останов по нажатию любой клавиши, либо когда метеорит долетит
до границы области}
    readln;
    for i:=1 to 200 do AStar[i].Done; BStar.Done;
    CloseGraph;
End.
```

Упражнение № 2. Работа с модулем, содержащим объекты «звезда» и «планета»

В следующем модуле описаны объекты: «звезда» и «планета». С их помощью на экране дисплея можно изображать точку — «звезду» и закрашенный круг — «планету» заданного цвета, удалять, инициировать и перемещать их.

Объект `Pozition` инициирует положение точки на экране дисплея и запоминает ее координаты. Объект `Star` наследует свойства `Pozition`, устанавливает точку с заданными координатами и с заданным цветом. Его методы позволяют перемещать и гасить точку. Объект `Planet` наследует свойства `Star` и имеет аналогичные методы по инициализации, установке и перемещению закрашенного круга.

Рекомендуется записать текст модуля `jPlanet` в файл с именем `Jplanet.pas` и откомпилировать его с целью получения файла `Jplanet.tpu`.

```
Unit jPlanet;
Interface
uses Graph, Crt;
   type Pozition = object
      X,Y: Integer;
      procedure Init(CoordX, CoordY: Integer);
      function GetX: Integer;
      function GetY: Integer;
   end;
StarPtr=^Star;
Star = object (Pozition)
      Visible: Boolean;
      Color: Word;
      constructor Init(CoordX, CoordY: Integer; InitColor:Word);
      destructor Done; virtual;
      function IsVisible: Boolean;
      procedure Show; virtual;
      procedure Blind; virtual;
      procedure Jump (NextX, NextY: Integer); virtual;
   end;
PlanetPtr= ^Planet;
Planet=object(Star)
      Radius: Integer;
```

```
        P1: Pointer;
        Size: Word;
        constructor Init(CoordX, CoordY: Integer; InitColor: Word;
        InitRadius: Integer);
        destructor Done; virtual;
        procedure Show; virtual;
        procedure Blind; virtual;
        procedure Jump(NextX, NextY: Integer); virtual;
    end;
Implementation
{********** Методы объекта Pozition ***************}
Procedure Pozition.Init(CoordX, CoordY: Integer);
    begin
        X:=CoordX; Y:=CoordY;
    end;
Function Pozition.GetX: Integer;
    begin GetX:=X; end;
Function Pozition.GetY: Integer;
    begin GetY:=Y; end;
{********** Методы объекта Star ******************}
Constructor Star.Init(CoordX, CoordY: Integer; InitColor: Word);
    begin
        Pozition.Init(CoordX, CoordY);
        Visible:= False; Color:= InitColor;
    end;
Destructor Star.Done;
    begin Blind; end;
Procedure Star.Show;
    begin
        Visible:= True; PutPixel(X,Y,Color);
    end;
Function Star.IsVisible: Boolean;
    begin IsVisible:= Visible; end;
Procedure Star.Blind;
    begin
        Visible:= False; PutPixel (X, Y, GetBkColor);
        {точка закрашивается фоновым цветом}
    end;
Procedure Star.Jump;
    begin
Blind; X := NextX; Y:= NextY; Show;
    end;
{*** ******** методы объекта Planet ***************}
Constructor Planet.Init(CoordX, CoordY: Integer; InitColor: Word;
                        InitRadius: Integer);
    begin
        Star.Init(CoordX, CoordY, InitColor); Radius:= InitRadius;
        Size:= ImageSize(X, Y, X+2*Radius, Y+2*Radius);
        {вычисляем размер прямоугольной области, занимаемой планетой}
        GetMem(P1,Size);
```

```
        {выделяем память в куче для хранения областей}
    end;
Destructor Planet.Done;
    begin Blind; FreeMem(P1, Size); end;
Procedure Planet.Show;
Var
    i: Integer;
    PromColor: Word;
    begin
        Visible := True;
        GetImage(X-Radius, Y-Radius, X+Radius, Y+Radius, P1^);
        PromColor := Graph.GetColor;
        {запоминаем установленный ранее цвет рисования}
        Graph.SetColor(Color);
        {устанавливаем заданный для планеты цвет}
        for I := 1 to Radius do Graph.Circle(X, Y, I);
        Graph.SetColor(PromColor);
    end;
Procedure Planet.Blind;
    begin
        Visible := False;
        PutImage(X - Radius, Y - Radius, P1^, NormalPut);
        {выдаем старый фон}
    end;
Procedure Planet.Jump(NextX, NextY: Integer);
    begin
        Blind; X := NextX; Y := NextY; Show;
    end;
    {секции инициализации нет}
End.
```

Пример 81. *Программа установки звезды (экземпляр aa) и двух планет (экземпляры APlanet, BPlanet).*

Следующие программы используют модуль `jPlanet.tpu`.

```
Program Example_81, Planet1;
Uses Crt, Graph, jPlanet;
Var GraphDriver, GraphMode, ErrorCode: Integer;
    APlanet, BPlanet: Planet;
    aa: star;
Begin
    GraphDriver := Detect; DetectGraph (GraphDriver, GraphMode);
    InitGraph(GraphDriver,GraphMode,'');
    if GraphResult <>GrOk then Halt(1);
    aa.Init(290,150,15); aa.show; readln;
    APlanet.Init(151,82,3,50); BPlanet.Init(201,82,4,50);
    APlanet.Show; Readln; BPlanet.Show; Readln;
    BPlanet.Jump(55,82); Readln; BPlanet.Blind; Readln;
    APlanet.Blind; Readln;
    APlanet.Done; BPlanet.Done; CloseGraph;
End.
```

Пример 82. *Программа «планеты на фоне мигающих звезд».*

Программа составлена на основе алгоритма примера 2 с добавлением двух экземпляров типа объект Planet.

```
Program Example_82, Planet2;
Uses Crt, Graph, Jplanet;
Var GraphDriver,GraphMode,ErrorCode: Integer;
   AStar: array[1..20] of Star;
   i,s: integer; APlanet, BPlanet: Planet;
Begin
   GraphDriver := Detect; InitGraph(GraphDriver,GraphMode,' ");
   if GraphResult <>GrOk then Halt(1);
   Randomize;
   for i:=1 to 20 do
   begin
      Astar[i].Init(50+random(350),50+random(250),15);
      Astar[i].show;
   end;{начальная установка 20 звезд}
   APlanet.Init(85,82,3,20); BPlanet.Init(201,82,4,10); readln;
   APlanet.Show; {установка стационарной планеты}
   BPlanet.Show; {установка блуждающей планеты} readln;
   i:=1;
   repeat
      BPlanet.Jump(55+i,82+i); s:=random(19)+1; Astar[s].blind;
      delay(100); Astar[s].show; i:=i+1;
   until (Keypressed) or (BPlanet.GetX>400) or (BPlanet.GetY>300);
      readln;
      for i:=1 to 20 do AStar[i].Done;
      APlanet.Done; BPlanet.Done;
   CloseGraph;
End.
```

Пример 83. *Программа «Динамическая модель солнечной системы».*

Программа содержит блоки программ, рассмотренных выше, а также алгоритм перемещения двух экземпляров объектного типа по окружностям.

```
Program Example_83, Planet3;
   Uses Crt, Graph, Jplanet;
   const pi=3.1415;
Var GraphDriver,GraphMode,ErrorCode: Integer;
   AStar: array[1..20] of Star;
   i,s: integer;
   Sun, Zemla, Moon: Planet;
   fi,h,fi1,h1 : real;
   x0,y0,x,y,r,r1 : integer;
Begin
   GraphDriver := Detect; InitGraph(GraphDriver,GraphMode,'');
   if GraphResult <>GrOk then Halt(1);
Randomize;
   for i:=1 to 20 do
   begin
      Astar[i].Init(50+random(550),50+random(350),15);
```

```
        Astar[i].show;
    end; {начальная установка 20 звезд}
    Sun.Init(320,240,15,20);
    Zemla.Init(320,340,blue,10);
    Moon.Init(320,370,red,5); readln;
    Sun.Show; Zemla.Show; Moon.Show; readln; h:=5;
    h1:=1;fi:=0; fi1:=0;
    r:=Moon.GetY-Zemla.GetY; {радиус орбиты Луны}
    r1:=Zemla.GetY-Sun.GetY; {радиус орбиты Земли}
    repeat
        x0:=round(r1*sin(fi1))+Sun.GetX;
        y0:=round(r1*cos(fi1))+Sun.GetY;
        x:=x0+round(r*sin(fi)); y:=y0+round(r*cos(fi));
        fi:=fi+2*pi*h/360; fi1:=fi1+2*pi*h1/360;
        Zemla.Jump(x0,y0); Moon.Jump(x,y); s:=random(19)+1;
        Astar[s].blind; delay(200); Astar[s].show;
    until Keypressed;
    for i:=1 to 20 do AStar[i].Done;
    Sun.Done; Zemla.Done; Moon.Done;
    CloseGraph;
End.
```

Задания для самостоятельной работы

1. Используя модуль *Unplan1.tpu*, разработайте программу «*Броуновское движение точки*».
2. Используя модуль *Unplan1.tpu*, разработайте программу «*Хаотичное движение точек*».
3. Используя модуль *Unplan1.tpu*, разработайте программу «*Имитационная модель диффузии двух газов*».
4. Используя модуль *Unplan1.tpu*, разработайте программу «*Звездный дождь*».
5. Используя модуль *Unplan1.tpu*, разработайте программу *"Звездное небо с мигающими звездами созвездия «Большая медведица»"*.
6. Используя модуль *jPlanet*, разработайте программу *«Звездное небо с планетами»*.
7. Используя модуль *jPlanet*, разработайте программу *«Статическая картина Солнечной системы»*.
8. Используя модуль *jPlanet*, разработайте программу *«Динамическая картина Солнечной системы с двумя планетами со своими спутниками»*.
9. Используя модуль *jPlanet*, разработайте программу *«Динамическая модель атома с двумя электронами»*.
10. Используя модуль *jPlanet*, разработайте программу *«Движение бильярдного шара»*.
11. Используя модуль *jPlanet*, разработайте программу *«Хаотичное движение шаров с упругим столкновением»*.

Упражнение № 3. Работа с числовыми объектами

Ниже приведен листинг модуля, содержащий процедуры матричной алгебры [6]. Задан класс «комплексные матрицы». Методы этих объектов содержат арифметические действия над комплексными матрицами.

```
Unit matrices;
interface
   uses crt;
   const max = 3;
   type
      str10 = string[10];
      str30 = string[30];
      matrix = array[1..max,1..max] of pointer;
      complex = record
         re, im : real;
            end;
      complex_ptr = ^complex;
      complex_matrix = array[1..max, 1..max] of complex;
      matrix_obj = object
         M : matrix;
         constructor init;
         procedure set_to_zero(var p:pointer); virtual;
         procedure scalar_sum(const p,q: pointer;
                   var r : pointer);virtual;
         procedure scalar_product(const p,q: pointer;
                   var r : pointer); virtual;
         procedure matrix_sum(const A,B: matrix);
         procedure matrix_product(const A,B: matrix);
         procedure convert_to_string(p:pointer; var tt : str30);
         virtual;
         procedure print(name : str10);
         destructor done;
      end;
   complex_matrix_obj = object(matrix_obj)
         constructor init(M0: complex_matrix);
         procedure set_to_zero(var p: pointer); virtual;
         procedure scalar_sum(const p, q: pointer; var r: pointer);
                virtual;
         procedure scalar_product(const p,q: pointer;
                   var r: pointer); virtual;
         procedure convert_to_string(p: pointer; var tt: str30);
                virtual;
         procedure print(name: str10);
   destructor done;
   End;
   Implementation
   constructor matrix_obj.init;
   Begin End;
   Procedure matrix_obj.set_to_zero(var p:pointer);
   Begin End;
   Procedure matrix_obj.scalar_sum(const p,q: pointer; var r: pointer);
   Begin End;
   Procedure matrix_obJ.scalar_product(const p,q: pointer; var r:
pointer);
   Begin End;
```

```
Procedure matrix_obj.matrix_sum(const A,B: matrix);
var i,j: word;
begin
for i:=1 to max do for J:=1 to max do scalar_sum(A[i,J],B[i,J],M[i,J]);
end;
Procedure matrix_obJ.matrix_product(const A,B: matrix);
var i,j,k: word;
prod, sum: pointer;
begin
getmem(prod, SizeOf(M[1,1]));
for i:=1 to max do for k:=1 to max do
begin
sum:= M[i,k]; set_to_zero(sum);
for J:=1 to max do
begin
scalar_product(A[i,J], B[J,k], prod); scalar_sum(sum, prod, sum);
end
end;
freemem(prod,SizeOf(M[1,1]));
end;
Procedure matrix_obj.convert_to_string(p: pointer; var tt: str30);
begin end;
Procedure matrix_obJ.print(name: str10);
var i, J, k, hold_y: word;
temp: str30;
begin
hold_y := WhereY; gotoxy(1, hold_y + (max + 1) div 2);
write(name, '=');
k := length(name) + 4; gotoxy(k, hold_y); write('v');
for i:=1 to max do
begin
gotoxy(k, hold_y + i); write('|');
for J:=1 to max do
begin
convert_to_string(M[i,J], temp); write(temp);
end;
      write('|')
   end;
   gotoxy(k, max + hold_y +1); write('%');
   gotoxy(k + length(temp) * max + 1, hold_y); write('J');
   gotoxy(k + length(temp) * max + 1, max + hold_y + 1); writeln('^');
end;
Destructor matrix_obJ.done;
   begin end;
Constructor complex_matrix_obj.init(M0:complex_matrix);
   var i, j : word;
   begin
      inherited init;
      for i:= 1 to max do
         for j := 1 to max do
```

```
            begin
                new(complex_ptr(M[i,j]));
                complex_ptr(M[i,j])^ :=M0[i,j];
            end;
end;
Procedure complex_matrix_obj.set_to_zero(var p: pointer);
   begin
       complex_ptr(p)^.re := 0.0; complex_ptr(p)^.im := 0.0;
       end;
Procedure complex_matrix_obj.scalar_sum(const p,q: pointer;
                                                    var r : pointer);
   var x, y, z : complex;
   begin
       x:= complex_ptr(p)^; y:= complex_ptr(q)^;
       z.re:= x.re + y.re; z.im:= x.im + y.im;
       if r = nil then new(complex_ptr(r));
       complex_ptr(r)^ := z;
   end;
Procedure complex_matrix_obJ.scalar_product(const p, q : pointer;
                                        var r : pointer);
   var x, y, z : complex;
   begin
       x := complex_ptr(p)^; y := complex_ptr(q)^;
       z.re := x.re * y.re - x.im * y.im;
       z.im := x.re * y.im + x.im * y.re;
       if not Assigned(r) then new(complex_ptr(r));
       complex_ptr(r)^ := z;
   end;
Procedure complex_matrix_obJ.convert_to_string(p: pointer;
                                                    var tt: str30);
   var t1, t2: string[6];
   begin
       str(complex_ptr(p)^.re:6:4, t1);
       str(complex_ptr(p)^.im:6:4, t2);
       tt:=''+ t1 + '+' + t2 +'i';
   end;
Procedure complex_matrix_obJ.print(name: str10);
   var i, J, k, hold_y: word; temp: str30;
   begin
       hold_y := WhereY; gotoxy(1, hold_y + (max + 1) div 2);
       write(name, '=');
       k := length(name) + 4; gotoxy(k, hold_y); write('v');
       for i:=1 to max do
       begin
           gotoxy(k, hold_y + i); write('|');
           for J:=1 to max do
               begin
                   convert_to_string(M[i,J], temp); write(temp);
               end;
               write('|')
```

```
        end;
    gotoxy(k, max + hold_y +1);
    write('%');
    gotoxy(k + length(temp) * max + 1, hold_y);
    write("J");
    gotoxy(k + length(temp) * max + 1, max + hold_y + 1);
    writeln('^');
    end;
Destructor complex_matrix_obJ.done;
    var i, j: word;
    begin
        for i:=1 to max do
        for J:= 1 to max do
        begin
            dispose(complex_ptr(M[i,J]));
        end;
    inherited done;
    end;
End.
```

Пример 84. *Программа работы с комплексными матрицами.*
Изучите приведенную ниже программу, демонстрирующую использование модуля *Matrices.*

```
Program Example_84, obJ_demo;
    uses crt, matrices;
Procedure test_complex;
    var
        A0,B0,C0: complex_matrix;
        A, B, C : complex_matrix_obJ;
        i,j : word;
    begin
        for i:=1 to max do
        for J:=1 to max do
            begin
                A0[i,j].re:=1.0; A0[i,J].im:=0.0; B0[i,J].re:=i;
                 B0[i,J].im:=J; C0[i,J].re:=0.0; C0[i,J].im:=0.0;
            end;
        A.init(A0); B.init(B0); C.init(C0);
        A.print('A'); B.print('B');
        C.matrix_sum(A.M, B.M); C.print("A + B");
        C.matrix_product(A.M, B.M); C.print("A * B");
        A.done; B.done; C.done;
    end; {конец процедуры }
Begin
    clrscr;
    writeln("Комплексные матрицы");
    test_complex;
    write("Нажмите <Enter>");
    readln;
End.
```

Задания для самостоятельной работы

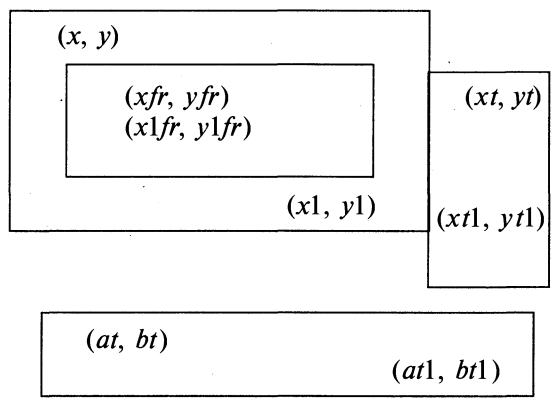
1. Используя модуль *Matrices*, разработайте программу определения суммы и произведения диагональных элементов комплексной матрицы 4×4, заданной случайным образом.
2. Используя модуль *Matrices*, разработайте программу сложения трех комплексных матриц 3×3 и в результирующей матрице определите сумму диагональных элементов.
3. Используя модуль *Matrices*, разработайте программу умножения трех комплексных матриц 3×3.

Упражнение № 4. Работа с объектами «окно»

Создадим модуль работы с окнами.

Объект «окно» содержит поля, задающие координаты окна (x, y) – ($x1$, $y1$), рамки (xfr, yfr) – ($x1fr$, $y1fr$), тени, состоящей из двух прямоугольников (xt, yt) – ($xt1$, $yt1$) и (at, bt) – ($at1$, $bt1$), рабочего поля (xz, yz) – ($xz1$, $yz1$).

Максимальные размеры окон:
без рамки (1, 1, 80, 25);
с рамкой (2, 2, 78, 24);
с тенью (1, 1, $x1$ + 2, $y1$ + 1).



Следующие атрибуты полей задают цвет фона, букв, рамку (одинарную или двойную) с заголовком или без, цвет рамки, тень, цвет тени.

Методы объекта:

Procedure `InitWn(Fx,Fy,Fx1,Fy1,Ffon,Fletter:byte)` – инициализация окна с координатами `(Fx,Fy)-(Fx1,Fy1)`, цветом фона `Ffon`, цветом букв `Fletter`.

Procedure `InitFr(Fframe,Fcolorframe:byte; Frubre:string)` – инициализация рамки: `Fframe=0` – нет рамки; `Fframe=1` – одинарная рамка; `Fframe=2` – двойная рамка; цвет рамки `Fcolorframe`, заголовок `Frubre`.

Procedure `InitSh(FcolorShade:byte)` – инициализация тени с цветом `FcolorShade`.

Procedure `Show` – открыть окно.

Procedure `Hide` – закрыть окно.

Собственные процедуры модуля:

Procedure `OknoBegin` – начало работы с окнами (обязательная процедура).

Procedure `OknoEnd` – завершение работы с окнами (обязательная процедура).

```
Unit Ok1;
interface
   Uses Crt;
   Type okno = Object
        x,y,x1,y1 : byte; {Координаты окна}
        xk,yk: byte; {Координаты курсора}
        xfr,yfr,x1fr,y1fr: byte;{Координаты рамки}
        xt,yt,xt1,yt1, at,bt,at1,bt1: byte; {Координаты тени}
        xz,xz1,yz, yz1: byte; {Координаты пред.области}
        frame: byte; {0 - нет рамки 1,2 - есть рамка}
        rubre:string[60]; {Заголовок}
        lrubre:byte;
        shade :byte; {0 - Нет тени 1 - Есть тень}
        fon,letter: byte; {Цвета фона, букв}
        colorframe,colorshade:byte;
        {Методы:}
        Procedure InitWn(Fx,Fy,Fx1,Fy1,Ffon,Fletter:byte);
        Procedure InitFr(Fframe,Fcolorframe:byte; Frubre:string);
        Procedure InitSh(FcolorShade:byte);
        Procedure Show; {Открыть окно}
        Procedure Hide; {Закрыть окно}
   end; {object}
Procedure OknoBegin; {Начало работы}
Procedure OknoEnd; {Конец работы}
{ ************************************************ }
Implementation
Procedure Errwn;
   begin
      Writeln("Для продолжения работы нажмите ENTER"); Readln;
   end;
Procedure Okno.InitWn;
   {Инициировать окно: установить координаты окна, цвет фона,
   букв, цвет внешнего окна для закраски — выполняется первой}
   begin
      x:=Fx; y:=Fy; x1:=Fx1; y1:=Fy1; fon:=Ffon;
      letter:=Fletter; frame:=0; lrubre:=0; shade:=0;
   end;
Procedure Okno.InitFr;
{Инициировать рамку: установить тип, цвет рамки и текст заголовка.
Если заголовка нет, то указывается «2»}
Var prz,i:integer;
   rb:string[60];
   lrb:byte absolute rubre;
Begin
   prz:=0; frame:=Fframe; colorframe:=Fcolorframe;
   rb:=Frubre; rubre:=rb; lrubre:=lrb;
   if x< 2 then begin x:=2; prz:=1 end;
   if y< 2 then begin y:=2; prz:=1 end;
   if x1>78 then begin x1:=78; prz:=1 end;
   if y1>24 then begin y1:=24; prz:=1 end;
```

```
    xfr:=x-1; yfr:=y-1; x1fr:=x1+1; y1fr:=y1+1;
    if prz <>0 then begin Writeln ("Изменены размеры окна из-за
                                    рамки=", x:3, y:3, x1:3, y1:3);
                    Errwn;
                    end;
    prz:=x1fr-xfr-lrubre-3;
    if prz<=0 then begin Writeln("Не поместился заголовок:");
    for i:=1 to lrubre do Write (rubre[i]);
    Writeln; Errwn; lrubre:=0;
                end;
  End; {InitFr}
  Procedure Okno.InitSh; {Инициировать тень: установить координаты и
цвет тени}
  Var xr,yr,x1r,y1r:byte;
     prz:integer;
  Begin
     shade:=1; {Смещение тени} colorshade:=Fcolorshade;
     repeat
     prz:=0;
     if frame <> 0 then
        begin xr:=xfr; x1r:=x1fr; yr:=yfr; y1r:=y1fr; end
        else begin xr:=x; x1r:=x1; yr:=y; y1r:=y1; end;
        {Координаты тени}
     xt:=x1r+1; xt1:=x1r+2; yt:=yr+1; yt1:=y1r;
     at:=xr+2; at1:=x1r+2; bt:=y1r; bt1:=y1r+1;
     if at1=79 then begin x1fr:=x1fr-1; x1:=x1-1; prz:=1 End;
     if bt1=25 then begin y1fr:=y1fr-1; y1:=y1-1; prz:=1 End;
     if prz <> 0 then begin
     Writeln ("Изменены размеры окна из-за тени: ",
                x:3,y:3,x1:3,y1:3,"смещение=",shade);
        errwn;
     end;
  until prz = 0; {До построения тени}
  End;
  Procedure Okno.Show; {Процедура Show открывает окно:рисует цвет,
рамку, заголовок, тень}
  Const
     UXY_2 =#201; {Двойная}
     UXY_1 =#218; {Одинарная}
     UX1Y_2=#187; {Рамка}
     UX1Y_1=#191; {Рамка}
     UXY1_2=#200; UXY1_1=#192; UX1Y1_2=#188; UX1Y1_1=#217;
     HRZ_2 =#205; HRZ_1 =#196; VER_2 =#186; VER_1 =#179;
  Var
     frameline:array[1..80] of char;
     xr,yr,xr1,yr1:byte;
     uxy,ux1y,uxy1,ux1y1,hrz,ver:char;
     lr,lh,lg:integer;
     {длина рамки, высота, количество горизонталий}
     num,num1,numr:integer; {длины без текста}
```

```
    k,i:integer;
    Begin
        xz:=x; yz:=y; xz1:=x1; yz1:=y1; {Запомнить экран}
        if frame<>0 then begin
        xz:=xfr; yz:=yfr; xz1:=x1fr; yz1:=y1fr;
    end;
if shade<>0 then begin xz1:=xt1;yz1:=bt1; end;
Window(xz,yz,xz1,yz1);
if shade<> 0 then begin
TextBackGround(colorshade); TextColor (white);
Window(xt,yt,xt1,yt1); Clrscr; Window(at,bt,at1,bt1); Clrscr;
                end;
{Построить рамку}
   if frame<>0 then begin
      if frame=1 then begin
         ver:=VER_1; hrz:=HRZ_1; uxy:=UXY_1;
         ux1y:=UX1Y_1; uxy1:=UXY1_1; ux1y1:=UX1Y1_1;
                      end
             else begin
                 ver:=VER_2; hrz:=HRZ_2; uxy:=UXY_2;
                 ux1y:=UX1Y_2;uxy1:=UXY1_2; ux1y1:=UX1Y1_2;
                     end;
            {Наружное окно}
            TextBackGround(fon); TextColor(colorframe);
            Window (xfr,yfr,x1fr, y1fr);
            Clrscr; {Размеры рамки}
            lr:=x1fr-xfr+1; lg:=x1fr-xfr-1; lh:=y1fr-yfr-1;
   {Верхняя горизонталь}
      if lrubre<>0 then
         begin
            num:=lg-(lrubre+2); num1:=num div 2;
            if (num mod 2) <>0 then numr:=num1+1 else numr:=num1;
            for i:=1 to 80 do frameline[i]:='';
            frameline[1]:=uxy;
            for i:=2 to num1 do frameline[i]:=hrz;
            frameline[i+1]:=''; k:=2+num1;
            for i:=1 to lrubre do frameline[k+i]:=rubre[i];
            k:=k+lrubre+1; frameline[k]:='';
            for i:=1 to numr do frameline [k+i]:=hrz;
            frameline[k+numr+1]:=ux1y;
         end else{ нет заголовка }
      begin
         frameline [1]:=uxy;
         for i:=2 to lg+1 do frameline[i]:=hrz; frameline[lr]:=ux1y;
      end;
for i:=1 to lr do write (frameline[i]);
for i:=1 to lh do begin
                    gotoxy(1,1+i); write(ver);
                    gotoxy(x1fr-xfr+1,1+i); write (ver);
                end;
```

```
    gotoxy(2,yr1);
    for i:=1 to 80 do frameline [i]:='';
    frameline[1]:=uxy1;
    for i:=2 to lg+1 do frameline[i]:=hrz;
    frameline[lr]:=ux1y1; window(xfr,y1fr,x1fr+1,y1fr);
    for i:=1 to lr do write (frameline[i]);
    textbackground(fon); textcolor(letter);
    {Внутреннее окно}
    window(x,y,x1,y1); clrscr;
end {есть рамка}
else {рамки нет}
    begin
        textbackground (fon); textcolor(letter);
        window(x,y,x1,y1); clrscr;
    end;
End;
Procedure Okno.Hide; {Закрыть окно}
Begin
    Window(xz,yz,xz1,yz1); clrscr; textbackground (blue);
End;
Procedure OknoBegin; {Установить атрибуты Input,Output}
Begin
    Assigncrt(input); Reset(input); Assigncrt (output);
    Rewrite (output);
End;
Procedure OknoEnd; {Восстановить атрибуты Input.Output}
    Begin
        Assign(input,''); Reset (input); Assign (output,'');
        Rewrite (output);
    End;
End. {unit}
```

Пример 85. *Программа, использующая модуль с объектом «окно».*

```
{Используются три окна a, b, c.}
Program Example_85 primer1;
    Uses crt,ok1;
var a,b,c : okno;
Begin
    OknoBegin;
    a.initwn(2,2,78,24,cyan,white); a.initfr(2,red,"ПРОВЕРКА ОКОН");
    b.initwn (6,6,24,21,white,blue);
    b.initfr(2,blue,"окно с тенью "); b.initsh(blue);
    c.initwn(4,4,45,12,lightred,yellow);
    c.initfr(1,white,'с одинарной рамкой'); c.initsh (green);
    a.show; writeln("ABCDE"); readln;
    b.show; writeln("asdfghj"); writeln("kkkkkk");
    writeln("yyyyyy"); writeln("hhhhhh"); readln;
    c.show; readln; c.hide; readln; b.hide; readln; a.hide; readln;
OknoEnd;
End.
```

Упражнение № 5. Работа с библиотекой Turbo-Vision

В состав интегрированных пакетов Турбо-Паскаль 6, 7 входит библиотека модулей *tvision.tph* (Turbo-Vision), содержащая объекты, процедуры и функции (методы объектов), позволяющие создавать с их помощью удобный интерфейс пользовательских программ. Примером использования этой библиотеки является интерфейс самой интегрированной среды Turbo-Pascal 6.

В папке *Example* найдите директории TV и TVDEMO. В них содержатся тексты модулей и тексты программ, использующих эти модули. Скомпилируйте модули и разместите соответствующие им откомпилированные файлы с расширением *.tpu* в эти же папки.

Теперь можно запускать демонстрационные программы TVDEMO.

Демонстрационные программы папки TV могут служит начальной заготовкой для создания новых программ. Создайте свой вариант простой обучающей программы с использованием библиотеки Turbo-Vision.

Задания для самостоятельной работы

1. Используя модуль *Ok1*, создайте интерфейс меню, аналогичное структуре инструментальной среды Турбо-Паскаль.
2. Используя модуль *Ok1*, создайте каскадный вид разных окон.
3. Изучите демонстрационные примеры с использованием библиотеки Turbo-Vision.
4. Придумайте пример построения интерфейса с помощью оболочки Turbo-Vision.

Краткие сведения

Система Дельфи

Delphi — среда визуального программирования, позволяет одновременно работать с несколькими раскрытыми окнами. Стандартный интерфейс Delphi 3 показан ниже.

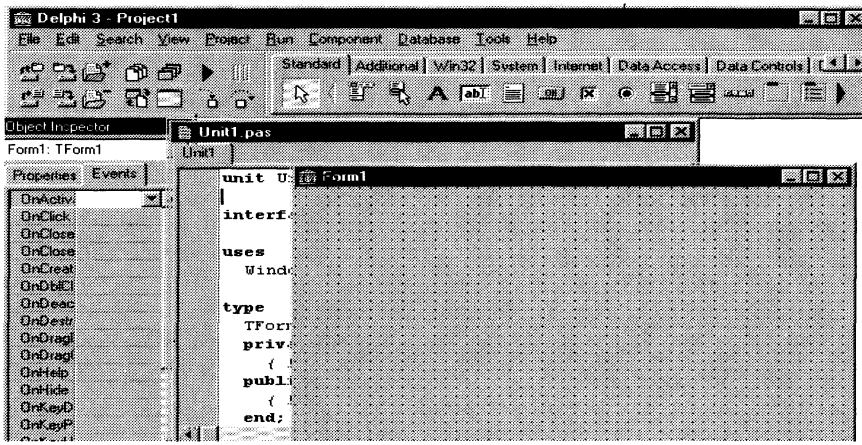
Главное меню (заголовок Delphi 3 — *Proect1*) осуществляет основные функции управления проектом создаваемой программы.

Окно формы (*Form1*) представляет проект интерфейсной части будущей программы в форме Windows-окон. В начале оно пусто. Имеется возможность размещать на нем необходимые компоненты интерфейса: кнопки, окна, меню и др.

Окно инспектора объекта (*Object Inspector*) содержит две страницы — *Properties* (Свойства) и *Events* (События). Первая страница служит для установки нужных свойств компонента, страница *Events* определяет реакцию компонента на то или иное событие.

Окно кода программ (*Unit1*) предназначено для создания и редактирования текста искомой программы на языке программирования Object-Pascal, практически совпадающего с версиями Turbo-Pascal 6,7 и имеющего некоторые расширения и усовершенствования. Например, введен специальный тип (вместо типа «объект») — класс, который содержит поля, методы и свойства. Термин позаимствован из Си++. Для совместимости с другими версиями Pascal сохранен тип *Object*. Аналогично «объекту» определяются основные свойства классов.

Инкапсуляция — объединение трех сущностей — полей, методов и свойств в единое целое (структура класс).



Наследование — порождение класса от другого класса с наследованием полей, методов и свойств своего родителя. Все классы порождены от единственного родительского класса *Tobject*.

Полиморфизм — свойство классов решать схожие задачи разными способами. Допускаются одноименные методы в классе-родителе и классе-потомке, имеющие разные алгоритмы обработки данных и придающие объектам разные свойства.

Первоначально окно кода содержит минимальный исходный текст (в виде модуля *Unit1*), жестко связанный с окном формы. Все изменения в окне форм отображаются в тексте программы.

Быстрые клавиши:

F9 — прогон программы;

F11 — открыть окно *Object Inspector*;

F12 — переключатель окна формы и окна кода программы.

Для начала работы в среде Delphi рекомендуется создать рабочую папку для хранения пользовательских проектов и модулей. Для каждого проекта создайте собственную папку.

Для удобства работы предлагается настроить среду следующим образом:

1. **Tools | Enviroument Optoins | Preferences | Autosave Options + Editor Files + Desktop | Compiling and Running + Show Compiler Progress.**

2. **Tools | Enviroument Optoins | Display + Editor Font + Courier New Cyr.**

Рекомендуется также использовать для новых программ архив заготовок, находящихся в репозитории.

Файл проекта имеет расширение *.dpr*, а текст модулей содержится в файлах с расширением *.pas*.

Задачи и упражнения

Упражнение № 6. Работа с формой. Изменение заголовка. Вставка компонентов. Первая программа

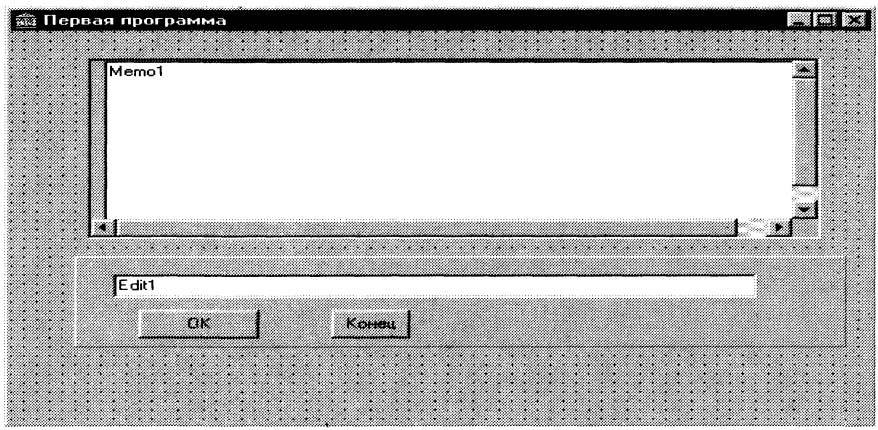
1. Изменение заголовка: в окне «*Инспектор Объектов*» выберем **Caption** и введем новый заголовок «Первая программа».

2. Выберем в инструментарии страницу **Additional** пиктограмму «*ScrollBox*» и разместим соответствующее ей окно в верхней части формы. Далее из палитры компо-

нентов **Standart** выберем пиктограмму «*Memo*» и установим ее в созданное окно скроллинга, перекрывая ее.

3. В нижней части формы разместим окно «*Панель*» (пиктограмма «*Panel*» в палитре **Standart**). Разместим в области окна-панели две кнопки (пиктограмма «*Ok*») и с помощью **Caption** из окна «*Инспектор Объектов*» проименуем их как «*OK*» и «*Конец*».

4. Запустим проект на исполнение командой *Run* или (F9), сохранив при этом текст модуля и проекта в соответствующей рабочей папке. На экране появится рисунок, показанный ниже.



Пример 86. *Программа умножения двух целых чисел.*

Задумаем программу, которая вводит два целых числа и выводит результат их умножения.

1. Создадим интерфейсную форму следующим образом. Разместим в верхней части формы окно многострочного редактирования «*Memo*». Уточним его свойства. С помощью Инспектора объектов установим значения: **Align** — alClient; **Lines** — *удалить;* **Name** — mmOut; **WordWrap** — False.

2. Расположим ниже окно «*Панель*». Установим основные свойства: **Caption** — *удалить*, **Name** — Panel.

3. На панели установим метку (компонента Label). Свойства: : **Align** — alBotton, **Caption** — *удалить,* **Name** — LbOut.

4. На панели, ниже метки, разместим окно «*Edit*». Свойства: **Text** — *удалить*, **Name** — edInput.

5. Установим кнопку «OK», по нажатию которой будем размещать первое введенное число в рабочее поле. Свойства: **Kind** — bkOk; **Name** — Bt1, **Caption** — OK.

6. Установим вторую кнопку «OK», по нажатию которой будем размещать второе введенное число в рабочее поле. Свойства: **Kind** — bkOk; **Name** — Bt2, **Caption** — ok.

7. Установим еще одну кнопку «OK» для выхода из программы. Свойства: **Kind** — bkOk; **Name** — Bt3, **Caption** — Выход.

8. Дважды щелкнем мышью по кнопке «OK» и в появившемся окне кода программ добавим в процедуру обработки события *OnClick* (одинарное нажатие клавиши) следующие команды (выделены жирным шрифтом):

```
procedure TForm1.Bt1Click(Sender: TObject);
```

```
begin
   X:=StrToInt(Trim(edInput.Text));
   mmOut.Lines.Add(edInput.Text);
   edInput.Text:=''; edInput.SetFocus;
   LbOut.Caption:='Введите 2-й операнд: ';
   Bt2.Show; Bt1.Hide;
end;
```

(*Комментарий*: позже удобно разместить две кнопки на одном месте наложением друг на друга, чтобы после использования одной кнопки она исчезала, а появлялась другая, и наоборот.)

9. Дважды щелкнем мышью по второй кнопке «OK» и в появившемся окне кода программ добавим команды (выделены жирным шрифтом):

```
procedure TForm1.Bt2Click(Sender: TObject);
begin
   Y:=StrToInt(Trim(edInput.Text));
   mmOut.Lines.Add(edInput.Text);
   mmOut.Lines.Add("Результат: "+IntToStr(X)+ " * "+
   IntToStr(Y)+" = "+IntToStr(X*Y));
   edInput.text:=''; edinput.SetFocus;
   LbOut.Caption:="Введите 1-й операнд: ";
   Bt2.Hide; Bt1.Show;
end;
```

10. Дважды шелкнем мышью по кнопке «Выход» и в появившемся окне кода программ добавим команду (выделена жирным шрифтом):

```
procedure TForm1.Bt3Click(Sender: TObject);
begin
   Close;
end;
```

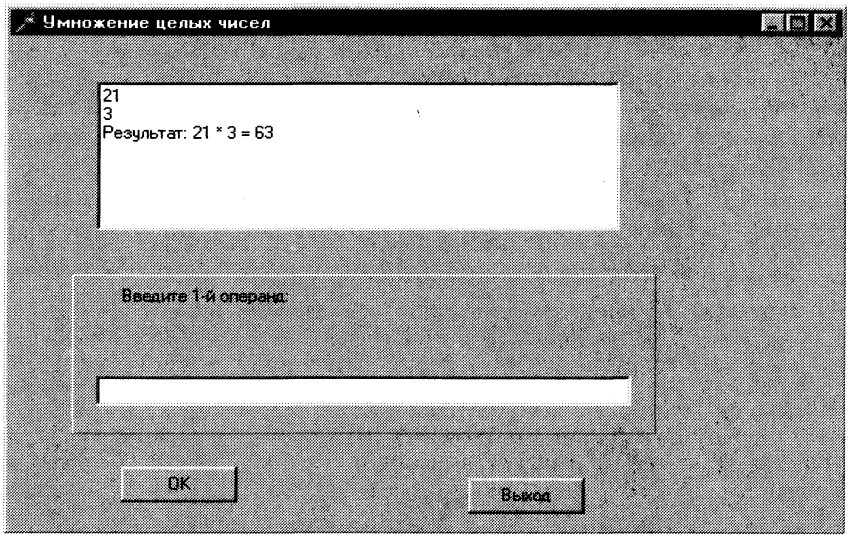
11. Осталось передать фокус ввода (очистить строку) окну «edInput» в момент старта программы и определить переменные `X` и `Y` для хранения операндов. В секции `private` в самом начале кода модуля формы разместим описание:

```
Private
{Private declarations}
X, Y : Integer;
```

12. В окне «*Инспектор Объектов*» выберем исходную форму *Form1* и после двойного щелчка на строке *OnActivate* на странице *Event* введем следующие понятные команды: (выделены жирным шрифтом):

```
procedure TForm1.FormActivate(Sender: TObject);
begin
edInput.SetFocus;
LbOut.Caption:=" Введите 1-й операнд: ";
Bt1.Show; Bt2.Hide;
end;
```

13. Сохраните все результаты работы в отдельной папке командой *Save All*, и запустите программу (*Run*). Типичная картина работы программы показана ниже:



Упражнение № 7. Работа со стандартными формами

В среде Дельфи имеются готовые формы-заготовки для создания прикладных программ. Шаблоны заготовок интерфейсов можно задать командами меню: **File | New | Projects.**

1. Выберите на странице *Projects* пиктограмму «**MDI Application**». Запустите программу (*Run*). Изучите интерфейсное окно и внесите свои пробные команды для работы с данной формой. Обратите внимание, что для каждой компоненты меню интерфейсного окна задана пустая процедура. Попытайтесь для каждой компоненты меню создать соответствующую форму.

2. Выберите на странице *Projects* пиктограмму «**SDI Application**». Запустите программу (*Run*). Изучите интерфейсное окно и внесите свои пробные команды для работы с данной формой. Для каждой компоненты меню создайте произвольную форму.

3. Вызовите «*Мастер форм*» и с его помощью создайте произвольный проект программы.

Задания для самостоятельной работы

1. Разработайте программу «Тренажер таблицы умножения».
2. Разработайте программу «Калькулятор».
3. Разработайте программу «Музыкальный редактор».
4. Разработайте программу «Графический редактор».
5. Разработайте программу «Текстовый редактор».
6. Используя «*Мастер форм*» и заготовки Дельфи, разработайте программу «Текстовый редактор» с развитым интерфейсом.
7. Используя «*Мастер форм*» и заготовки Дельфи, разработайте программу «Графический редактор» с развитым интерфейсом.
8. Используя «*Мастер форм*» и заготовки Дельфи, разработайте простейшую программу «Справочная система».

9. Используя «*Мастер форм*» и заготовки Дельфи, разработайте программу «Тест по заданной теме».

10. Используя «*Мастер форм*» и заготовки Дельфи, разработайте программу «Электронный учебник».

Дополнительная литература

1. *Буч Г.* Объектно-ориентированное программирование с примерами применения: Пер. с англ. — Киев: Диалектика, 1992.

2. *Дантеман Д., Мишел Д., Тейлор Д.* Программирование в среде Delphi: Пер. с англ. — Киев: НИПФ-ДиаСофт Лтд., 1995.

3. *Дарахвелидзе П., Марков Е.* Delphi — среда визуального программирования. — СПб.: BVH — Санкт-Петербург, 1996.

4. *Довгаль С.И., Литвинов Б.Ю., Сбитнев А.И.* Персональные ЭВМ: Турбо-Паскаль 7.0. Объектное программирование. Локальные сети. — Киев: Информсистема сервис, 1993.

5. *Зуев Е.А.* Язык программирования Турбо-Паскаль 6.0. — М.: Унитех, 1992.

6. *Немнюгин С.А.* Турбо-Паскаль. — СПб.: Питер, 2000.

7. Неформальное введение в C++ и Turbo-Vision. — СПб.: Галерея «Петрополь», 1992.

8. *Фаронов В.В.* DELPHI 5. Учебный курс. — М.: Нолидж, 2000.

Тесты к главе 3

Языки программирования высокого уровня

1. Алфавит языка программирования — это:
   1) фиксированный набор символов, однозначно трактуемых;
   2) *a..я*;
   3) *a..z*;
   4) набор слов, которые понимает компьютер.
2. Оператор — это:
   1) функция, которая оперирует с данными;
   2) законченная фраза языка, предписание, команда;
   3) алгоритм действия программы, написанной на данном языке;
   4) процедура обработки данных.
3. Переменная — это:
   1) объект, способный принимать различные значения; 2) значения чисел;
   3) меняющееся число; 4) динамический объект.
4. Модуль — это:
   1) отдельная программа, которая взаимодействует с другими программами;
   2) набор символов и идентификаторов;
   3) специальная программная единица для создания библиотек;
   4) вспомогательная процедура.
5. Язык программирования — это:
   1) набор слов для написания программы;
   2) определенная последовательность бит;
   3) специально созданная система обозначений слов, букв, чисел;

4) двоичные коды для компьютера.

6. Синтаксис языка программирования — это:
   1) набор правил расстановки знаков препинания;
   2) система правил, определяющая допустимые конструкции языка;
   3) интерпретация отдельных языковых конструкций языка;
   4) фиксированный набор основных символов, допускаемых для составления программы.
7. Семантика языка программирования — это:
   1) система правил, определяющая допустимые конструкции языка;
   2) система правил однозначного истолкования языковых конструкций языка;
   3) набор металингвистических формул бэкуса-наура;
   4) синтаксическая конструкция, определяющая свойства программных объектов.
8. Синтаксическая диаграмма — это:
   1) график функции;
   2) таблица понятий;
   3) семантический граф;
   4) графическое представление значения метапеременной.
9. Метаформула — это:
   1) специальная формула, для каждого языка принимающая свое значение;
   2) переменная, состоящая из нескольких семантик;
   3) суперпозиция формул;
   4) нормальная форма для понятия из языка программирования.
10. Функция в языке программирования — это:
   1) программный объект, принимающий значение с помощью оператора присваивания;
   2) программный объект, задающий вычислительную процедуру определения значения от аргумента;
   3) сегмент программы, хранящий некоторое значение, зависящее от аргумента;
   4) выражение, означающее зависимость левой части от правой.
11. Языки программирования высокого уровня являются:
   1) набором нулей и единиц;
   2) ограниченными по объему информации;
   3) машинно-зависимыми;
   4) машинно-независимыми.
12. Метаязык — это:
   1) язык для описания языков;
   2) суперязык, состоящий из комбинаций разных языков;
   3) язык, состоящий из метапеременных;
   4) язык, в котором используются метаформулы.
13. Язык программирования образуют три составляющие:
   1) алфавит, орфография, диалектика; 2) алфавит, синтаксис, семантика;
   3) переменные, процедуры, функции; 4) модули, описания, реализация.
14. Величины в языках программирования характеризуются:
   1) элементами, размером, значением;
   2) однородностью, предназначением, полезностью;
   3) местоположением, принадлежностью, значением;
   4) типом, именем, значением.
15. Метаформула выглядит как:

1) <метаформула>:=<формула(формула)>;
2) <переменная>:- <буква>|<цифра>;
3) <переменная>::=A|B <выражение>::=<переменная>|<переменная>+<переменная>|<переменная>-<переменная>;
4) <имя переменной>:=<буква>.

16. Простой величине соответствует:
1) одна ячейка памяти;
2) массив из простых чисел;
3) структура входных, выходных и промежуточных значений;
4) множество простых ее элементов.

17. Характеристики структурной величины:
1) упорядоченность, однородность, способ доступа, фиксированность числа элементов;
2) индивидуальность имен, порядок перечисления элементов;
3) однозначность, неизменность, множество элементов;
4) размер занимаемой памяти, многофункциональность, способ доступа к элементам.

18. Команда на машинном языке содержит:
1) строку из спецсимволов;
2) задание сделать ту или иную операцию;
3) служебное слово;
4) код и адреса ячеек, с содержимым которых выполняется закодированное действие.

Язык программирования Паскаль

1. Что называют операторными скобками?
1) (); 2) { }; 3) begin..end; 4) [ ].

2. Процедуры и функции — это:
1) операторы; 2) подпрограммы;
3) имена; 4) переменные.

3. Файл — это:
1) база данных; 2) поименованный участок на внешних носителях памяти;
3) список; 4) раздел на жестком диске.

4. Упорядоченный тип — это:
1) тип переменной, значения которой упорядочены в обычном смысле;
2) запись;
3) целые и вещественные;
4) значения переменных такого типа находятся в порядке, случайно выбранном.

5. Выражение — это:
1) конструкция языка, значение которой может меняться;
2) текст программы, заключенный в операторные скобки;
3) множество символов, которые являются упорядоченными;
4) конструкция, задающая правила вычисления значений переменных.

6. Рекурсия — это:
1) повторение выполнения функции или процедуры внутри себя;
2) оператор;
3) цикл;
4) метод определения функции или процедуры.

7. Массив — это:
   1) запись множества переменных разного типа;
   2) неупорядоченная совокупность отличных друг от друга однотипных элементов;
   3) последовательность, состоящая из фиксированного числа однотипных элементов;
   4) тип одномерных величин.
8. Множество — это:
   1) список элементов, заключенный в круглые скобки, вида:
   `<имя поля>:<значение>;`
   2) неупорядоченная совокупность отличных друг от друга однотипных элементов;
   3) совокупность с фиксированным числом однотипных элементов, отличных только индексами;
   4) совокупность отличных друг от друга элементов разных типов.
9. Запись — это:
   1) константное значение;
   2) последовательность, состоящая из фиксированного числа однотипных элементов;
   3) последовательность однотипных элементов, отличающихся индексами;
   4) последовательность, состоящая из фиксированного числа величин, называемых полями.
10. Константное значение — это:
   1) тип, вида: `к=record i:integer; r:real end;`
   2) тип вида: `const=array [1..3];`
   3) тип вида: `const=set of real;`
   4) список элементов, заключенный в круглые скобки, вида
   `<имя поля>:<значение>.`
11. Для обозначения величин используют имена. Именем будет являться:
   1) `Name-«Петя»`; 2) `1Name`; 3) `/Name`; 4) `.Name1`.
12. *N* будет константой в описании:
   1) `const N=5`; 2) `N:const=5`; 3) `N=5`; 4) `N:integer=5`.
13. Оператор присваивания выглядит следующим образом:
   1) `<имя переменной>:-<значение>;`
   2) `<имя переменной>:=<выражение>;`
   3) `<имя переменной >::= <выражение>;`
   4) `<значение>:= <имя переменной>.`
14. Внешние библиотеки создаются в виде:
   1) отдельного файла или нескольких; 2) другой программы;
   3) процедуры; 4) функции.
15. Над вещественными величинами определены операции:
   1) `not, and, or` и стандартные;
   2) `<, >, =, odd(), abs()`, и стандартные;
   3) *, +, -, / и стандартные;
   4) `odd(), eof(), abs(), sin(), cos()`.
16. Над логическими величинами определены операции:
   1) +, -, *, /;
   2) `not, and, or, odd()`;
   3) `sin(), cos(), tg(), abs()`;

4) `trunc(), round(), ord().`

17. Цикл с постусловием записывается в виде:
   1) `While <логическое выражение> do <оператор>;`
   2) `For i:=1 to n do <оператор>;`
   3) `Repeat <последовательность операторов> until <логическое выражение>;`
   4) `Case k of <последовательность операторов>.`

18. Цикл с предусловием запишется в виде:
   1) `While <логическое выражение> do <оператор>;`
   2) `For i:=1 to n do <оператор>;`
   3) `Repeat <последовательность операторов> until <логическое выражение>;`
   4) `Case k of <последовательность операторов>.`

19. Цикл с параметром запишется в виде:
   1) `While <логическое выражение> do <оператор>;`
   2) `For i:=1 to n do <оператор>;`
   3) `Repeat <последовательность операторов> until <логическое выражение>;`
   4) `Case k of <последовательность операторов>.`

20. В массиве индексы можно вычислить. Их тип должен быть:
   1) логическим; 2) перечисляемым; 3) ординальным; 4) массивом.

21. В типе String количество символов одной строки не должно превышать:
   1) 256; 2) 255;
   3) 1024; 4) 2400.

22. Глобальные переменные действуют:
   1) во всех процедурах; 2) во всех функциях;
   3) во всех модулях; 4) во всей программе.

23. Обращение к функции в программе имеет вид:
   1) `<имя функции>(<список фактических параметров>);`
   2) `<оператор функции>;`
   3) `<имя функции>:=<значение>;`
   4) `<имя функции>.`

24. Обращение к процедуре в программе имеет вид:
   1) `<имя процедуры>(<список глобальных параметров>);`
   2) `<оператор процедуры>;`
   3) `<имя процедуры>:тип значения;`
   4) `<имя процедуры>.`

25. Связь программы с принтером осуществляется процедурой:
   1) `assign(f,"aux");` 2) `assign(f,"usr");`
   3) `assign(f,"lst");` 4) `assign(f,"№ порта принтера").`

26. Для динамических переменных выделение и очистка памяти происходит:
   1) на этапе трансляции; 2) на этапе компиляции;
   3) на этапе отладки; 4) в ходе выполнения программы.

27. Значением указателя динамической переменной является:
   1) адрес сегмента носителя информации, в котором будет храниться соответствующая динамическая величина;
   2) адрес ячейки памяти, начиная с которой будет храниться соответствующая динамическая величина;
   3) № кластера жесткого диска, в котором будет храниться соответствующая динамическая величина;

4) значение динамической величины.

28. Что произойдет, если выполнить операторы:

```
New(i);
Writeln(i);
```

1) выдастся адрес динамической переменной;
2) выдастся значение динамической переменной с адресом *i*;
3) перезагрузка компьютера;
4) ничего.

29. Каков будет результат выполнения программы:

```
var s1,s2,s3:string;
begin
s1:="паро"; s2:="воз";
s3:=concat(s1,s2);
Writeln(s3);
end.
```

1) пар и воз; 2) парвз; 3) 7; 4) паровоз.

30. Каков будет результат выполнения программы:

```
var s1,s2:string;
begin
s1:="информатика";
delete(s1,3,4);
Writeln(s1);
end.
```

1) инатика; 2) форма; 3) инф; 4) инфо.

31. Каков будет результат выполнения программы:

```
var s1,s2:string;
begin
s1:=copy("крокодил",4,3);
Writeln(s1);
end.
```

5) крок; 6) одил; 7) код; 8) кродил.

32. Каков будет результат выполнения программы:

```
var s:string;
begin s:=length("каникулы") End.
```

1) s=0; 2) s=1; 3) s=8; 4) s=true.

33. Каков будет результат выполнения программы:

```
var r:real;
begin
r:=4. 869; T:=trunc(r)
End.
```

1) T=23. 07; 2) T=2. 207; 3) T= 5; 4) T=4.

34. Какая из данных программ записана без ошибок:

1) `var b:boolean; begin b:=7; writeln("результат: ", b); end.`
2) `var b:boolean; begin b:=false; if not b then writeln("Ура!"); end.`
3) `var b:boolean; begin b:="Hello, World"; writeln(b); end.`
4) `var b:boolean; c:real; begin c:=sqr(b); writeln("результат: ", c); end.`

35. Какая из данных программ на Паскале правильная:

1) `var r:string; begin r:=true; If r then halt; end.`

2) `var r:string; c:char; begin r:=4/c; end.`
3) `var r:string; begin r:="Hello World!"; Writeln(r); end.`
4) `var r:string; begin r:=Hi, friend; Write(r); end.`

36. Выберите правильный результат действия программы на Паскале:

```
const n=2;
var k:integer; m,l:real;
begin
l:=0;
For k:=1 to 6 do
m:=k/n;
l:=l+m
end.
```

1) `l=10.5`; 2) `l=20.5`; 3) `l=1.5`; 4) `l=10`.

37. Каков будет результат выполнения программы:

```
type digits=set of 0..9;
var d1,d2,d3:digits;
begin
d1:=[1,3,5];
d2:=[0,4,5];
d3:=d1*d2;
end.
```

1) `d3=[0,1,3,4,5]`; 2) `d3=[0,1,3,4]`;
3) `d3=[5]`; 4) `d3=[1,3,5,0,4,5]`.

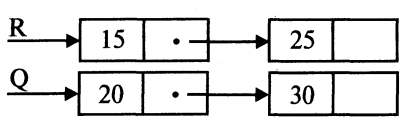
38. Каков будет результат выполнения программы:

```
var f:text; t:integer;
begin
assign(f,'<>');
reset(f);
Write(f,'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
t:=filesize(f)
end.
```

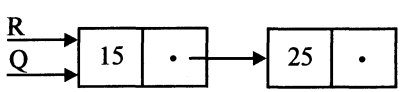
1) `t=true`; 2) `t="ABCDEFGHIJKLMNOPQRSTUVWXYZ"`;
3) `t=26`; 4) `t=1`.

39. Каков будет результат выполнения программы:

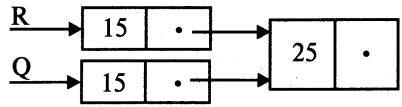
```
Type Point=^Ct;
Ct=Record I:integer; P:Point End;
Var Q,R:^Point;
Begin
Q:=R;
End.
```



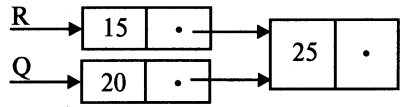
1) `Q` указывает на ту же переменную, что и `R`;



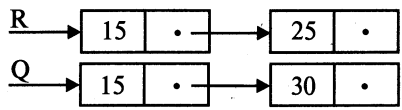
2) на место указанной переменной [20|], указывавшей на 30, заслана переменная [15|], указывающая на 25;



3) на место ссылки на компоненту [30|] заслана ссылка на компоненту [25|], поле целого значения не изменилось;

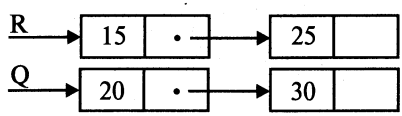


4) на место 20 заслано 15, поле указателя не изменилось.

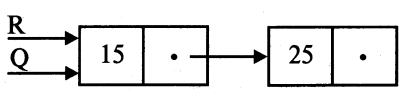


40. Каков будет результат выполнения программы

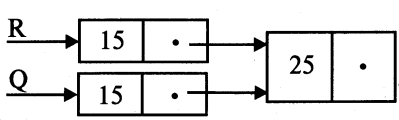
```
Type Point=^Ct;
Ct=Record
I:integer;
P:Point;
End;
Var Q,R:^Point;
Begin
Q^:=R^;
End.
```



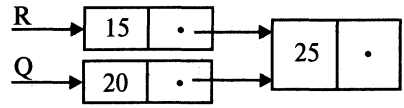
1) Q указывает на ту же переменную, что и R;



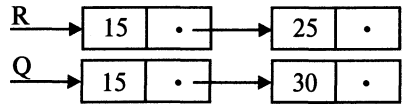
2) на место указанной переменной [20|], указывавшей на 30, заслана переменная [15|], указывающая на 25;



3) на место ссылки на компоненту [30|] заслана ссылка на компоненту [25|], поле целого значения не изменилось;

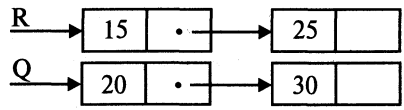


4) на место 20 заслано 15, поле указателя не изменилось.

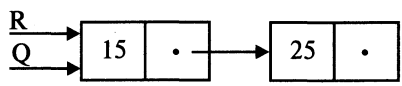


41. Каков будет результат выполнения программы

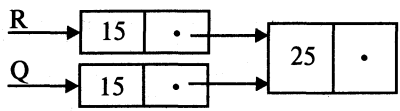
```
Type Point=^Ct;
Ct=Record I:integer; P:Point End;
Var Q,R:^Point;
Begin Q^.I:=R^.I End.
```



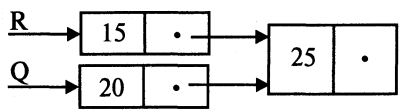
1) `Q` указывает на ту же переменную, что и `R`;



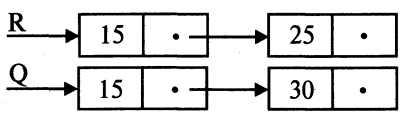
2) на место указанной переменной [20|], указывавшей на 30, заслана переменная [15|], указывающая на 25;



3) на место ссылки на компоненту [30|] заслана ссылка на компоненту [25|], поле целого значения не изменилось;

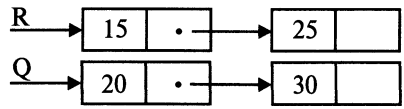


4) на место 20 заслано 15, поле указателя не изменилось.

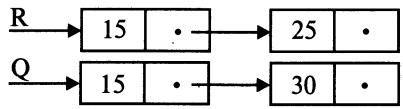


42. Каков будет результат выполнения программы:

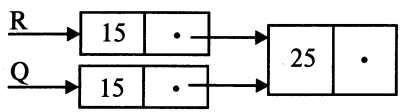
```
Type Point=^Ct;
Ct=Record I:integer; P:Point End;
Var Q,R:^Point;
Begin Q^.P:=R^.P End.
```



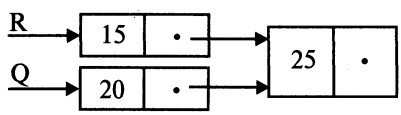
1) Q указывает на ту же переменную, что и R;



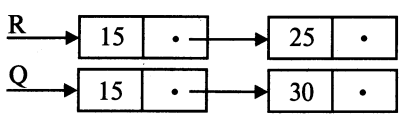
2) на место указанной переменной [20|], указывавшей на 30, заслана переменная [15|], указывающая на 25;



3) на место ссылки на компоненту [30|] заслана ссылка на компоненту [25|], поле целого значения не изменилось;



4) на место 20 заслано 15, поле указателя не изменилось.



43. Каков будет результат выполнения программы:

```
Var n, i: integer;
Begin
   n:=0;
   For i:=1 to 10 do n:=n+i
end.
```

1) 55; 2) 10; 3) 25; 4) 225.

44. Что означает описанная ниже процедура:

```
Type Stackp=^Stackcomp;
```

```
    Stackcomp=Record
    I:Integer; P:Stackp
End;
Var S:Stackp;
Begin
    S:=nil;
    Procedure IS(k:integer);
    Var IN:Stackp;
    Begin
        New(IN);
        With IN do begin
        I:=k; P:=S;
        End;
    S:=IN;
End;
```

1) помещение компоненты IN в стек;
2) извлечение компоненты IN из стека;
3) помещение компоненты IN в очередь;
4) извлечение компоненты IN из очереди.

45. Что делают операторы

```
Stackp=^Stackcomp;
Stackcomp=Record
    I:Integer; P:Stackp
End;
Var NP, L:Stackcomp;
Begin NP^.P:=L^.P; L^.P:=NP End.
```

1) запись новой компоненты в очередь;
2) запись новой компоненты в стек;
3) удаление компоненты из очереди;
4) удаление компоненты из стека.

46. В текстовом файле записаны целые числа из диапазона от 1000 до 10000 по 10 чисел в каждой строке. В следующей программе должно выводиться в качестве результата число, равное одному проценту от суммы всех чисел, хранящихся в файле. Определить результат работы программы, если в файле 1000 строк, содержащих только числа 1000.

```
Var f:text; i, s, a : integer; p: real;
begin
    assign (f, "in.txt"); reset (f);
    while not eof (f) do
    for i:=1 to 10 do
    begin
        read (f, a);
        s:= s+a;
    end;
    p:=s/100; writeln (p:9:2)
end.
```

1) правильный результат не может быть вычислен;
2) программа зациклится;

3) 0.00;

4) результат не может быть выведен в указанном формате.

47. Какая из приведенных ниже программ наполняет заданное множество целыми числами, принадлежащими диапазону от 0 до 50, содержащимися в текстовом файле, и считает их количество?

1.
```
Program inp_set;
var m : set of 1..50;
x, i, n : integer;
f : text;
begin
assign(f, "text_file");
reset(f);
m:=[];
n:=0;
for i:=1 to 1000 do
begin
read(f, x);
if (x in m) then
begin
m:=m+x;
n:=n+1
end
end
end.
```

2.
```
Program inp_set;
var m : set of 1..50;
x, i, n : integer;
f : text;
begin
assign(f,"text_file");
reset (f);
m:=[];
n:=0;
While not eof (f) do
begin
read(f, x);
if (0<=x<=50) then
begin
m:=m+[x];
n:=n+1
end
end
end.
```

3.
```
Program inp_set;
var m : set of
1..50;
x, i, n : integer;
f : text;
begin
assign(f, "text_file");
reset (f);
m:=[];
n:=0;
Whie not eof (f) do
begin
read(f, x);
if (0<=x) and (x<=50)
then
begin
m:=m+x;
n:=n+1
end
end
end.
```

1) ни одна из программ; 2) программа 1; 3) программа 2; 4) программа 3.

48. Какое из приведенных ниже выражений тождественно выражению «`a or b`»:

1) `not (not a and not b and (c or not c));`
2) `not (not a or not b);`
3) `(not a or b) and (b and not b);`
4) `(a>b) or (a<b).`

49. Результат вычисления какого из приведенных ниже выражений является решением следующей задачи: для целого *n*, где 100 < *n* < 1000, определить символьный вид третьей справа цифры в записи числа:

1) `chr(n/100);`
2) `chr((n/100)*10);`
3) `(n div 1000 mod 10);`
4) `chr((n mod 1000) div 100);`

50. Определить результат вычисления функции `akk (1, k)`; если задано ее описание:

```
Function akk (m,n : integer) : integer;
begin
   if m=0
   then akk:=n+1
   else
   if n=0
   then akk:=akk(m-1,1)
```

```
    else akk:=akk(m-1, akk(m,n-1));
end;
```

1) 2*k; 2) k+3; 3) 2*k=3; 4) k+2.

51. Вычислить значение выражения

```
(trunc (r) >= d) and (c > "a") and (b < (ord (c) =0))
```

при c = "c", d=0, r=0.5, если это возможно.

1) в записи выражения есть ошибка; 2) true; 3) false; 4) 4. 0.

52. В некоторой программе описаны переменные:

```
Var i:integer; r:real; c:char;
```

Какая из процедур или функций, заголовки которых приведены ниже, может быть вызвана из этой программы с помощью оператора `c := f(i, 5, r, 5*i, c, «r»);`

1) `function f(var a: integer; b: integer; var c: real; d: real; var e: char; g : char);`
2) `function f(var a: integer; b: integer; var c: real; d: real; var e: char; g : char): char;`
3) `Procedure f(var a: integer; b: integer; var c: real; d: real; var e: char; g : char);`
4) `function f(a: integer; var b: integer; var c: char; d: real; var e: char; g : char): char;`

53. Определить, какая задача решается с помощью данной последовательности операторов

```
m :=a[1];
for i:=1 to n do
if a[i] > m then a[i] := m else m := a[i];
```

1) каждому элементу массива *a* присваивается значение, равное минимальному из значений всех элементов исходного массива, предшествующих данному элементу, и значения самого этого элемента.
2) ищется максимальный элемент массива;
3) меняются местами минимальный и максимальный элементы массива;
4) выполняется сортировка элементов массива в порядке возрастания.

54. В программе описана матрица

```
a: array [1..n,1..m] of Integer;
```

где n, m — целые константы.

Во время работы программы формируется вектор, в который в порядке возрастания помещаются все элементы матрицы. Определить, какое из приведенных ниже описаний подходит для данного вектора:

1) `b: array [1..n+m] of Integer;`
2) `b: array [1..n*m] of Integer;`
3) `b: array [n+m] of Integer;`
4) `b: array [1..2*(n+m)] of Integer.`

Методы и искусство программирования

1. Альтернатива — это:
1) композиция разных действий; 2) вариант;
3) конструкция ветвления; 4) шаг выполнения программы.

2. Итерация — это:
   1) шаг выполнения программы; 2) циклическая конструкция алгоритма;
   3) язык программирования; 4) функция прерывания.
3. Дедуктивный принцип — это:
   1) когда определяется связь между входными, выходными данными и процессами обработки;
   2) принцип построения модели от частного к общему;
   3) упрятывание информации и абстрактных типов данных;
   4) принцип построения модели от общего к частному.
4. Индуктивный принцип — это:
   1) когда определяется связь между входными, выходными данными и процессами обработки;
   2) принцип построения модели от частного к общему;
   3) упрятывание информации и абстрактных типов данных;
   4) принцип построения модели от общего к частному.
5. Линейный связный список — это:
   1) конечный набор пар, состоящих из информативных и указующих частей;
   2) рекурсивная конструкция алгоритма;
   3) совокупность динамических переменных;
   4) массив указателей.
6. Сортировка — это:
   1) процесс нахождения в заданном множестве объекта;
   2) процесс перегруппировки заданного множества объектов в некотором порядке;
   3) установка индексов элементов в возрастающем порядке;
   4) обработка элементов в алфавитном порядке.
7. Композиция — это:
   1) циклическая конструкция алгоритма;
   2) линейная конструкция алгоритма, состоящая из последовательно следующих друг за другом функциональных вершин;
   3) конструкция ветвления, имеющая предикатную вершину;
   4) суперпозиция двух алгоритмов.
8. Блок-схема — это:
   1) ориентированная сеть, у которой могут быть вершины типов: функциональные, предикатные и объединяющие;
   2) рисунок с изображением алгоритма;
   3) семантический граф операторов алгоритма;
   4) семантическая диаграмма.
9. Тестирование программы — это:
   1) оценивание ресурсов компьютера, на котором будет работать программа;
   2) перевод проекта в форму программы для конкретного компьютера;
   3) системный подход к построению алгоритма с использованием декомпозиции и синтеза;
   4) процесс исполнения программы с целью выявления ошибок.
10. Инспекция при тестировании — это:
   1) надзор за изменением состояний переменных;
   2) отслеживание логических ошибок;
   3) набор процедур и приемов обнаружения ошибок;
   4) надзор за соответствием типов и атрибутов переменных.
11. Граничные условия в тестах — это:

1) ситуации, возникающие непосредственно на, выше или ниже границ входных и выходных классов эквивалентности;
2) тестовые задания, имеющие наивысшую вероятность обнаружения ошибок;
3) выход индексов заданий за пределы допустимых;
4) границы применимости теста.

12. Если данные размещены на внешнем носителе, то доступ к ним возможен:
1) моментальный; 2) прямой; 3) последовательный; 4) выборочный.

13. Если данные размещены в оперативной памяти, то доступ к ним возможен:
1) прямой; 2) параллельный; 3) последовательный; 4) перебором.

14. Процедура линейного поиска — это:
1) просмотр массива с конца; 2) просмотр массива с середины;
3) сравнение эталона осуществляется с элементом, расположенным в середине массива;
4) последовательный просмотр всех элементов массива и сравнение их с эталоном.

15. Процедура поиска делением пополам заключается:
1) в просмотре массива с конца до середины;
2) в просмотре массива с середины;
3) в сравнении эталона с элементом, расположенным в середине массива;
4) в последовательном просмотре всех элементов массива и сравнении их с эталоном.

16. Дан алгоритм сортировки: определяется минимальный элемент среди всех и меняется местами с первым и т.д., начиная со второго. Вид сортировки:
1) метод прямого включения; 2) метод прямого выбора;
3) пузырьковый метод; 4) с помощью «дерева».

17. Алгоритм сортировки: идет обмен местами двух элементов в массиве после их сравнения друг с другом:
1) прямого включения; 2) прямого выбора;
3) пузырьковый метод; 4) с помощью дерева.

18. Реализация алгоритма включает в себя:
1) гипотезу, инструкцию, умозаключение;
2) выбор задачи и цели, разработку, анализ;
3) кодирование, интеграцию, тестирование;
4) определение проблемы, формализацию стратегии, установку интерфейса.

19. Деструктивность процесса тестирования проявляется в следующем:
1) тест удачный, если обнаружена ошибка;
2) тест удачный, если проведен без ошибок;
3) тест неудачный, если обнаружена еще не выявленная ошибка;
4) тест неудачный, если все задания некорректны.

20. Тестирование программы как черного ящика заключается в следующем:
1) знаем, какие данные будут на выходе;
2) не знаем, какие данные подаем на вход;
3) анализ входных данных и результатов работы программы;
4) управляем логикой программы, используя ее внутреннюю структуру.

21. Тестирование программы как белого ящика заключается в следующем:
1) не знаем, какие данные будут на выходе;
2) не знаем, как получаются данные на выходе;
3) анализ входных данных и результатов работы программы;
4) управляем логикой программы, используя ее внутреннюю структуру.

22. Целью декомпозиции является:
1) определение связи между модулями;
2) процедурное описание программы;
3) создание модулей, которые взаимодействуют друг с другом по определенным правилам;
4) неформальное описание модуля: обзор действий.

23. В методологии Джексона предусматривается:
1) определение структуры программы структурой данных, подлежащих обработке;
2) определение связи между входными, выходными данными и процессом обработки с помощью иерархической декомпозиции;
3) упрятывание информации и абстрактных типов данных; рассматриваются данные, модули и системы в качестве объектов;
4) объектно-ориентированное программирование.

24. Метод иерархических диаграмм предусматривает:
1) определение структуры программы структурой данных, подлежащих обработке;
2) определение связи между входными, выходными данными и процессом обработки с помощью иерархической декомпозиции;
3) упрятывание информации и абстрактных типов данных; рассматриваются данные, модули и системы в качестве объектов;
4) объектно-ориентированное программирование.

Язык программирования Бейсик

1. Подпрограммная единица SUB — это:
1) процедура; 2) функция; 3) модуль; 4) подключение файла.

2. Подпрограммная единица FUNCTION — это:
1) подпрограмма-процедура; 2) подпрограмма-функция;
3) функциональная зависимость модуля; 4) подключение файла.

3. Функция TIMER — это:
1) функция возврата текущей даты системной среды;
2) функция возврата текущего времени системной среды;
3) показатель времени, заданного в программе;
4) счетчик времени в цикле.

4. Оператор LINE — это:
1) оператор для изображения очень тонкой линии;
2) оператор для изображения линий одной толщины;
3) оператор изображения окружности;
4) оператор для закрашивания.

5. Команда PRINT — это:
1) команда вывода на экран результатов вычислений или сообщений;
2) команда распечатки на принтере;
3) команда распечатки в файл;
4) команда распечатки всего текста программы.

6. Команда LIST — это:
1) команда вывода на экран результатов вычислений или сообщений;
2) команда распечатки на принтере;
3) команда распечатки в файл;
4) команда вывода на экран дисплея всего текста программы.

7. Команда RUN — это:
   1) команда распечатки всего текста программы;
   2) вывод значений всех переменных;
   3) команда запуска программы на выполнение;
   4) команда запуска компиляции программы .
8. Оператор NEXT — это:
   1) выполнение следующего оператора прежде, чем текущего;
   2) циклическая конструкция;
   3) переход к следующей процедуре;
   4) присвоение текущему номеру строки следующего номера.
9. Оператор SAVE — это:
   1) команда сохранения текста программы в оперативную память;
   2) команда сохранения текста программы в виде файла;
   3) компиляция в память компьютера;
   4) команда загрузки ранее сохраненной программы.
10. Оператор LOAD — это:
   1) команда сохранения текста программы в виде файла;
   2) команда сохранения текста программы в оперативную память;
   3) компиляция в память компьютера;
   4) команда загрузки ранее сохраненной программы.
11. Сформировать собственные функции можно с помощью:
   1) LET; 2) DEF; 3) RUN; 4) INPUT.
12. Ноты обозначают:
   1) A — до, B — ре, C— ми, D — фа, E — соль, F — ля, G — си;
   2) C — до, D — ре, E — ми, F — фа, G — соль, A — ля, B — си;
   3) D — до, R — ре, M — ми, F — фа, S — соль, L — ля, C — си;
   4) C — до, L — ре, S — ми, F — фа, M — соль, R — ля, D — си.
13. Бемоль — это:
   1) звучание на тон выше; 2) звучание на тон ниже;
   3) звучание на полтона выше; 4) звучание на полтона ниже
14. Диез — это:
   1) звучание на полтона ниже; 2) звучание на тон ниже;
   3) звучание на полтона выше; 4) звучание на тон выше.
15. Программировать музыку можно командой:
   1) `RUN`; 2) `SIGN`; 3) `PLAY`; 4) `PAINT`.
16. Октава O1 в соответствии с фортепианным рядом будет:
   1) контроктава; 2) первая октава;
   3) большая октава; 4) малая октава.
17. Пауза в программе музыки обозначается:
   1) `+, -`; 2) `C8, D2`; 3) `Rn` или `Pn`; 4) `Play "03 aaa"`.
18. Задание графического экрана производится с помощью:
   1) `PAINT`; 2) `SCREEN`; 3) `WINDOW`; 4) `COLOR`.
19. Задание масштаба или окна на экране производится:
   1) `PAINT`; 2) `SCREEN`; 3) `WINDOW`; 4) `COLOR`.
20. Для задания фона экрана используется команда `COLOR z1, z2`:
   1) `z1` — цвет фона, `z2` — цвет текста;
   2) `z1` — цвет текста, `z2` — цвет фона;
   3) `z1` — координата *x* вывода текста на экран, `z2` — координата *y* вывода текста на экран;
   4) `z1` — координата *y* вывода текста на экран, `z2` — координата *x* вывода текста на экран.

21. Изображение точки с координатами и заданным цветом осуществляется командой

1) `CIRCLE(X, Y), Z`; 2) `PSET(X, Y),Z`;
3) `PIX(X, Y),Z`; 4) `PAINT(X, Y), Z`.

22. Звуковой сигнал стандартной частоты и длительности звучания можно дать командой:

1) `SOUND`; 2) `PIK`; 3) `SIGNAL`; 4) `BEEP`.

23. PAINT — это оператор:

1) закрашивания областей, ограниченных линией одного цвета;
2) закрашивания всех областей с любыми границами;
3) рисования закрашенной окружности;
4) рисования закрашенного прямоугольника.

24. Пример имени строковой переменной в языке Бейсик:

1) `NAME$`; 2) `#NAME`; 3) `NAME_STR`; 4) `STR(NAME)`.

25. Для организации подпрограммы в языке Бейсик используют команды:

1) `INTERFACE, IMPLEMENTATION`; 2) `BEGIN, END`;
3) `INPUT, GOTO`; 4) `GOSUB, RETURN`.

26. В языке Бейсик для записи в файл используют команду:

1) `INPUT`; 2) `PRINT`; 3) `OPEN`; 4) `INKEY`.

27. Каков будет результат выполнения программы на языке Бейсик:

```
10 a=5
20 b=4
30 a*b
```

1) `5*4`; 2) `20`; 3) `a*b`; 4) `4*5`.

28. Каков будет результат выполнения программы на языке Бейсик:

```
10 def F(x, y, z)=x*x+y*y+z*z
20 x = 5
30 y = 4
40 z = 10
50 ? F(x, y, z)
```

1) 141; 2) 2000; 3) 19; 4) error.

29. Каков будет результат выполнения оператора `PLAY"cdefgab"`:

1) произношение букв cdefgab; 2) ничего;
3) исполнение гаммы нот: до-ре-ми-фа-соль-ля-си;
4) исполнение гаммы нот: си-ля-соль-фа-ми-ре-до.

30. Каков будет результат выполнения оператора `PLAY"C8 D2 L16 fbe"`:

1) звучание ноты «до» длительностью в одну восьмую ноты, «ре» — в одну вторую, а ноты фа-си-ми звучат с длительностью в одну шестнадцатую ноты;
2) звучание ноты «ми» длительностью в одну восьмую ноты, «фа» — в одну вторую, а ноты ля-ре-соль звучат с длительностью в одну восьмую ноты;
3) звучание ноты «фа» длительностью в одну восьмую ноты, «си» — в одну вторую, а ноты до-ре-ми звучат с длительностью в одну шестнадцатую ноты;
4) звучание ноты «до» длительностью в одну целую ноты, «ре» — в одну вторую, а ноты фа-си-ми звучат с длительностью в одну вторую ноты.

31. Каков будет результат выполнения программы:

```
10 c$ = "аеиоуыэюяАЕИОУЫЭЮЯ"
20 a$="Сердце"
30 n = 0
```

```
40 FOR k = 1 TO LEN(c$)
50 b$ = MID$(c$, k, 1)
60 FOR i = 1 TO LEN(a$)
70 IF b$ = MID$(a$, i, 1) THEN n = n + 1
80 NEXT i
90 NEXT k
100 PRINT n
110 END
```

1) $n = 6$; 2) $n = 4$; 3) $n = 2$; 4) $n = 3$.

Язык программирования Си

1. В языке Си лексема — это:
   1) набор специальных символов и директив;
   2) множество строк, определяющих состояние программы;
   3) процедура, выполняющая определенные задания;
   4) последовательности символов языка, разделяющиеся пробелами и другими неграфическими символами.
2. В языке Си указатель — это:
   1) специальный значок, показывающий, что это динамическая переменная;
   2) символическое представление адреса ячейки памяти;
   3) символ, указывающий на что-либо;
   4) метка.
3. В языке Си литерал — это:
   1) переменная зарезервированного типа;
   2) неизменяемый объект языка;
   3) строка;
   4) буква.
4. Комментарии заключаются в скобки:
   1) { }; 2) /* */; 3) [ ]; 4) /% %/.
5. Идентификатор — это:
   1) последовательность латинских букв, цифр и символа «_», начинающаяся с буквы или символа «_»;
   2) неизменяемые объекты языка (константы);
   3) последовательность латинских и русских букв;
   4) способ кодирования, допустимые преобразования над значением данной переменной.
6. Фактический адрес в указателях — это:
   1) строка; 2) указатель; 3) число; 4) буква.
7. Составной оператор — это:
   1) последовательность операторов, заключенная в фигурные скобки `{}`;
   2) последовательность операторов, заключенная квадратные скобки `[]`;
   3) последовательность операторов, заключенная в операторные скобки `begin ... end`;
   4) последовательность операторов, заключенная в круглые скобки `()`.
8. Спецификация типа — это:
   1) задание типа переменной;
   2) список переменных;
   3) перечисление всех переменных, которые использовались в программе;
   4) список типов переменных, которые использовались в программе.

9. Логическое «не равно» обозначается:
   1) <>; 2) ||; 3) !; 4) !=;
10. Логическое «и» обозначается:
   1) =; 2) ||; 3) &; 4) &&.
11. Логическое «не» обозначается:
   1) !; 2) !!; 3) ||; 4) not.
12. Битовая операция инверсии битов обозначается:
   1) \~; 2) ~; 3) >>; 4) << .
13. Битовая операция исключающего «или» обозначается:
   1) \~; 2) ~; 3) ||; 4) &&.
14. Операция битового «и» обозначается:
   1) \~; 2) ~; 3) ||; 4) &.
15. Операция битового «или» обозначается:
   1) \~; 2) ~; 3) |; 4) &.
16. Текстовый поток — это:
   1) логическое понятие, которое система может относить к чему угодно — от дисковых файлов до терминалов;
   2) последовательность символов, которая организуется в строки, завершающиеся символами новой строки;
   3) последовательность символов, которая организуется в списки слов, завершающиеся точкой с запятой;
   4) текст программы.
17. Выражения — это:
   1) конструкции, включающие константы (литералы), переменные, знаки операций, скобки для управления порядком выполнения операций, обращения к функциям;
   2) основные строительные блоки программы; в языке Си указанием на наличие выражения служит символ «точка с запятой», стоящий в конце него;
   3) набор символов и операций;
   4) операторы, выполняющие определенные действия с переменными.
18. Тернарное выражение — это:
   1) компактный способ записи оператора `WHILE/DO`;
   2) компактный способ записи оператора `IF/ELSE`;
   3) выбор одного из нескольких вариантов;
   4) выражение, описывающее действия логических связывающих операторов на переменные.
19. Оператор-переключатель — это:
   1) оператор для выбора одного из нескольких вариантов (`SWITCH`);
   2) строка с меткой `DEFAULT`; 3) `CASE`; 4) `BREAK`.
20. Оператор цикла **`DO/WHILE`** является:
   1) конструкцией цикла с предусловием;
   2) конструкцией цикла с постусловием;
   3) конструкцией цикла с выбором варианта;
   4) конструкцией цикла с перебором значений параметра.
21. Формальный аргумент — это:
   1) конкретное значение, присвоенное этой переменной вызывающей программой;
   2) переменная в вызываемой программе;
   3) строка, которая пишется в скобках функции;

4) строка, которая пишется в скобках процедуры.

22. Фактический аргумент — это:
1) конкретное значение, присвоенное этой переменной вызывающей программой;
2) переменная в вызываемой программе;
3) строка, которая пишется в скобках функции;
4) строка, которая пишется в скобках процедуры.

23. Писать `#include <stdio.h>` нужно для:
1) подключения файла, содержащего макроопределения и объявления данных, необходимых для работы функций из стандартной библиотеки ввода-вывода;
2) позволяет дать в программе макроопределения (или задать макросы);
3) переопределения не только константы, но и целых программных конструкций;
4) замены каждого параметра в строке лексем на соответствующий аргумент макровызова.

24. Точка с запятой является:
1) разделителем операторов; 2) частью оператора;
3) ключевым знаком языка Си; 4) спецсимвол.

25. Какой тип данных отсутствует в Си в отличие от большинства других языков:
1) `Real`; 2) `Integer`; 3) `String`; 4) `Char`.

26. Символ % сигнализирует программе:
1) о начале описания переменных;
2) о начале описания функции;
3) о присваивании переменной значения;
4) начиная с этой позиции, необходимо вывести значение переменной.

27. В языке Си тело функции ограничено операторными скобками:
1) `begin end`; 2) `start finish`;
3) `[]`; 4) `{}`.

28. В языке Си программа начинает выполняться с функции:
1) `Start`; 2) `Main`; 3) `Go`; 4) `Do`.

29. Обращение к функции форматного ввода имеет вид:
1) `scanf(<формат>,<&имя1>,<&имя2>,...,<&имяN>);`
2) `printf(<формат>,<&имя1>,<&имя2>,...,<&имяN>);`
3) `scanf(<формат>,<имя1>,<имя2>,...,<имяN>);`
4) `printf(<формат>,<имя1>,<имя2>,...,<имяN>);`

30. Идентификатором будет:
1) `schetchik get_line a12 Param1 _ab;`
2) `%ab 12abc -x schetchik;`
3) `\b ab 12abc -x schetchik;`
4) `* ab 12abc -x schetchik.`

31. Лидирующий нуль в литералах означает:
1) числовой шестнадцатеричный литерал;
2) вещественный десятичный литерал;
3) числовой восьмеричный литерал;
4) целый десятичный литерал.

32. Символьным литералом будет:
1) «q»; 2) %q; 3) «s»; 4) «sq».

33. Строковым литералом будет
1) «sq»; 2) %q; 3) «s»; 4) .«qsqs».

34. Оператор INT в Си применяется для:
1) переопределения диапазона целых чисел;
2) преобразования переменной к целому типу;
3) описания переменных целого типа;
4) прибавления единицы к коду символа.

35. Строки в Си представляются в виде:
1) множества символов, стоящих в один ряд;
2) одного идентификатора;
3) массива элементов типа CHAR;
4) символического представления ячейки памяти.

36. Наличие нуль-символа (\0) означает, что:
1) количество ячеек массива должно быть, по крайней мере, на одну больше, чем число символов, которые необходимо размещать в памяти;
2) логическим значением переменной является «ложь»;
3) количество ячеек массива должно быть на одну меньше, чем число символов, которые необходимо размещать в памяти;
4) логическим значением переменной является «истина».

37. Пример: `val = *ptr;` операция косвенной адресации * производит:
1) получение адреса;
2) перенаправление адреса переменной val к переменной ptr;
3) определение значения, на которое указывает ptr;
4) определение значения, на которое указывает valv.

38. Если в цикле задано два разных условия выхода, то используется оператор:
1) `CONTINUE`; 2) `BREAK`; 3) `GOTO`; 4) `NEXT`.

39. Если в выражениях встречаются операнды различных типов, то они преобразуются к общему типу в соответствии с определенными правилами. Если один из операндов имеет тип `Char`, то:
1) другие также преобразуются к типу `Char` и результат имеет тип `Char`;
2) другие преобразуются к типу `Int` и результат имеет тип `Int`;
3) во время операции присваивания значение правой части преобразуется к типу левой части, который и становится типом результата;
4) остается как есть и результат будет `Char`.

40. Метки в операторе `Switch` должны быть:
1) указателями; 2) переменной; 3) константой; 4) типа `Char`.

41. Используя форму обращения `Function1(x)`, получаем:
1) передачу в функцию значения переменной *x*;
2) передачу адреса переменной *x*;
3) использование глобальной переменной;
4) использование класса памяти *x*.

42. Используя форму обращения `Function1(&x)`, получаем:
1) передачу в функцию значения переменной *x*;
2) передачу адреса переменной *x*;
3) использование глобальной переменной;
4) использование класса памяти *x*.

43. Тип функции определяется:
1) типом ее аргументов; 2) использованием в программе;
3) типом ее описания; 4) типом возвращаемого ею значения.

44. Автоматические объекты:
1) существуют во время выполнения данного блока и теряют свои значения при выходе из него;

2) хранятся вне любой функции, входящей в состав программы, и существуют в течение выполнения всей программы;
3) являются объектами статического класса памяти;
4) можно инициализировать только выражениями с константами и с указателями на ранее описанные объекты.

45. Макровызов должен состоять:
1) из списка макросов; 2) из списка макропеременных;
3) из списка макроимен;
4) из макроимени и заключенного, в круглые скобки списка аргументов.

46. Каков будет результат выполнения операторов:

```
nrs = 22;
ptr = &nrs;
val = *ptr;
```

1) присваивание значения 22 переменной `ptr`;
2) `&nrs` дает адрес переменной `val`;
3) `&nrs` дает адрес переменной `ptr`;
4) присваивание значения 22 переменной `val`.

47. Каков будет результат выполнения операторов:

```
int i,j,s;
i=j=2; /* i и j получают значение 2 */
s=(i++)+(++j);
```

1) $i = 3, j = 2, s = 5$; 2) $i = 3, j = 3, s = 6$;
3) $i = 3, j = 3, s = 5$; 4) $i = 2, j = 3, s = 5$.

48. Каков будет результат выполнения операторов:

```
int x,y,a;
x=5;
y=x*2+7;
a=y/4;
```

1) $x = 5, y = 17, a = 4{,}25$; 2) $x = 5, y = 17, a = 4$;
3) $x = 5, y = 10, a = 2{,}25$; 4) $x = 5, y = 32, a = 8$.

49. Каков будет результат выполнения операторов:

```
a=(y=(x=5)*2+7)/4
```

1) $a = 4{,}25$; 2) $a = 4$; 3) $a = 2{,}25$; 4) error.

50. Каков будет результат выполнения операторов:

```
int x,y;
x=y=5;
x+=2;
y-=3;
x*=y;
x/=++y;
```

1) $y = 3, x = 4$; 2) $y = 4, x = 12$; 3) $y = 12, x = 12/3$; 4) $y = 3, x = 14$.

51. Каков будет результат выполнения операторов:

```
int a,b
a=4;
b=7;
m=(a>b)?a:b;
```

1) $m = 4$; 2) $m = 11$; 3) $m = 3$; 4) $m = 7$.

52. Каков будет результат выполнения операторов:

```
int x,y
y=-4;
x=(y<0)?-y:y;
```

1) $x = 4$; 2) $x = -4$; 3) $x = 0$; 4) $x = 8$.

53. Каков будет результат выполнения операторов:

```
char ch
for(ch='a';ch<='z';ch++)
printf("значение для %c равна %d.\n",ch,ch);
```

1) выдача величины кода ASCII;
2) подсчет арифметической прогрессии;
3) подсчет геометрической прогрессии;
4) выдача номера места символа в алфавите.

54. Каков будет результат выполнения операторов:

```
#define PI 3.14159 #define E 2.711828
```

1) препроцессор заменит в программе все имена PI и E на соответствующие числовые константы;
2) переопределение значения E на PI;
3) подключение математических понятий E и PI;
4) запись указаний компилятору.

55. В языке Си какой вывод будет после выполнения операции

```
num=-256;
Print("Это число %x",num)
```

1) 256; 2) 400; 3) 100; 4) −100.

56. В языке Си какой вывод будет после выполнения операции

```
num=-11;
Print("Это число %o",num)
```

1) 11; 2) −11; 3) 13; 4) −13.

57. В языке Си какой вывод будет после выполнения операции

```
num=-11;
Print("Это число %u",num)
```

1) 1011; 2) 13; 3) 11; 4) 2.

Язык программирования Пролог

1. В языке Пролог *факт* — это:
   1) неопровержимое доказательство;
   2) истинное происшествие;
   3) предикат с аргументами-константами;
   4) правило, которое выполняется всегда.
2. В языке Пролог *правило* — это:
   1) хорновские фразы с заголовком и одной или несколькими подцелями;
   2) предикаты, носящие приказывающий характер;
   3) факты, в которых содержится условие;
   4) алгоритм действия.
3. *Вопрос* — это:
   1) отправная точка логического ввода, происходящего при выполнении программы;

2) отправная точка логического вывода, происходящего при выполнении программы;
3) отправная точка логического вывода свободных переменных;
4) запрос программы на сопоставление переменных.

4. *Имя* — это:
1) последовательность букв и цифр, начинающаяся со строчной буквы;
2) последовательность букв и цифр, начинающаяся с заглавной буквы;
3) конструкция, состоящая из имени и заключенного в круглые скобки списка его аргументов, разделенных запятыми.
4) объединение элементов произвольных видов, разделенных запятыми и заключенных в квадратные скобки.

5. *Переменная* — это:
1) последовательность букв и цифр, начинающаяся со строчной буквы;
2) последовательность букв и цифр, начинающаяся с заглавной буквы;
3) конструкция, состоящая из имени и заключенного в круглые скобки списка его аргументов, разделенных запятыми;
4) объединение элементов произвольных видов, разделенных запятыми и заключенных в квадратные скобки.

6. Вопрос называется *общим*, если:
1) все переменные, которые он содержит, — свободные;
2) хотя бы одна переменная, которую он содержит, — свободная;
3) он не содержит переменных;
4) все переменные, которые он содержит, — связанные.

7. Вопрос называется *частным*, если:
1) все переменные, которые он содержит, — свободные;
2) хотя бы одна переменная, которую он содержит, — свободная;
3) он содержит переменные;
4) все переменные, которые он содержит, — связанные.

8. *Структура* — это:
1) объединение элементов произвольных видов, разделенных запятыми и заключенных в квадратные скобки;
2) конструкция, состоящая из имени структуры и ее свойств, разделенных запятыми;
3) последовательности букв и цифр, начинающиеся со строчной буквы;
4) конструкция, состоящая из имени структуры и заключенного в скобки списка ее аргументов, разделенных запятыми.

9. *Список* — это:
1) объединение элементов произвольных видов, разделенных запятыми и заключенных в квадратные скобки;
2) конструкция, состоящая из имени структуры и ее свойств, разделенных запятыми;
3) последовательности букв и цифр, начинающиеся со строчной буквы;
4) конструкция, состоящая из имени структуры и заключенного в скобки списка ее аргументов, разделенных запятыми.

10. *Программа* на Прологе является:
1) алгоритмом действия операторов на переменные;
2) записью условия задачи на языке формальной логики;
3) процедурным описанием алгоритма;
4) функциональным описанием алгоритма.

11. Чем в языке Пролог заканчивается строка программы?

1) : 2) :- 3) ; 4) .

12. Какая операция в языке Пролог является основной?
1) присваивание; 2) сопоставление;
3) отсекание; 4) редуцирование.

13. Как в языке Пролог выглядит запрос?
1) `?`; 2) `/Help`; 3) `zapros`; 4) `say`.

14. В языке Пролог выход из рекурсии обеспечивается:
1) `Halt`; 2) `Break`; 3) `Stop`; 4) `!`(отсечение).

15. В языке Пролог *списком* будет:
1) `[голова|хвост]`; 2) `p^next`; 3) `set of item`; 4) `rray`.

16. Что в языке Пролог будет являться *именем*:
1) `Name`; 2) `name`; 3) `$name`; 4) `#name`.

17. Что в языке Пролог будет являться переменной:
1) `Name`; 2) `name`; 3) `$name`; 4) `#name`.

18. Переменная, используемая в качестве аргумента предиката, когда конкретное значение переменной несущественно, — это переменная:
1) свободная; 2) связанная; 3) анонимная; 4) декларативная.

19. Переменная, которая еще не получила конкретного значения в результате сопоставления с константами в фактах, — это:
1) свободная; 2) связанная; 3) анонимная; 4) декларативная.

20. Переменная, которая приняла конкретное значение, называется:
1) свободной; 2) связанной; 3) анонимной; 4) декларативной.

21. Переменные служат:
1) хранилищем информации; 2) частью процесса сопоставления;
3) отправной точкой логического вывода; 4) заменой констант.

22. Стратегия согласования *«замкнутый мир»* — это когда:
1) поиск подходящих для согласования фактов и правил в базе знаний происходит последовательно сверху вниз, и если подходящих фактов не найдено, — ответ отрицательный;
2) поиск подходящих для согласования фактов и правил в базе знаний происходит последовательно снизу вверх, и если подходящих фактов не найдено, — ответ отрицательный;
3) поиск подходящих для согласования фактов и правил в базе знаний происходит последовательно сверху вниз, и если подходящих фактов не найдено, — ответ положительный;
4) поиск подходящих для согласования фактов и правил в базе знаний происходит последовательно снизу вверх, и если подходящих фактов не найдено, — ответ положительный.

23. *Декларативный* подход к программе — это когда:
1) последовательность сопоставлений, конкретизаций переменных и резолютивных выводов происходит при ее выполнении;
2) описанные отношения объектов некоторой предметной области и связи рассматриваются статически;
3) описанные отношения объектов некоторой предметной области и связи рассматриваются динамически;
4) последовательность сопоставлений, конкретизаций переменных и резолютивных выводов происходит при ее компиляции в оперативную память.

24. *Процедурный* подход к программе — это когда:
1) последовательность сопоставлений, конкретизаций переменных и резолютивных выводов происходит при ее выполнении;

2) описанные отношения объектов некоторой предметной области и связи рассматриваются статически;
3) описанные отношения объектов некоторой предметной области и связи рассматриваются динамически;
4) последовательность сопоставлений, конкретизаций переменных и резолютивных выводов происходит при ее компиляции в оперативную память.

25. Каков будет результат выполнения программы:

```
much ([], 0).
much ([A|B], N) :- much (B, M), N is M+1.
?- much ([саша, игорь, лена]), X).
```

1) $X = 3$; 2) $X =$ лена; 3) $X =$ саша; 4) $X =$ игорь.

26. Каков будет результат выполнения программы:

```
prin (X,[X|Y]).
prin (X,[A|Y]) :- prin (X,Y).
?prin (4,[1,3,4,9]).
```

1) Yes; 2) No; 3) True; 4) False.

27. Каков будет результат выполнения программы:

```
pris ([],P,P).
pris ([X|Y],P,[X|T]):- pris (Y,P,T).
? pris (L,[джим..R],[джек,бил,джим,тим,джим,боб]).
```

1) L=[джек,бил]. R=[тим,джим,боб]. L=[джек,бил,джим,тим]. R=[боб].
2) R=[джек,бил]. L=[тим,джим,боб]. R=[джек,бил,джим,тим]. L=[боб].
3) L=[джек,тим]. R=[джек, тим,джим,боб]. L=[джек,тим]. R=[джим].
4) L=[джек]. R=[боб]. L=[джек,бил,джим,тим]. R=[джек, боб].

28. Каков будет результат выполнения программы:

```
max([X],X).
max([X|Y],X):- max(Y,W), X>W, !.
max([X|Y],W):- max(Y,W).
?max([1,7,6,4,3],M)
```

1) $M = 1$; 2) $M = 7$; 3) $M = 3$; 4) $M = 6$.

29. Предикат вычисления факториала натурального числа *n* выглядит:

1) `faktorial(1,1). faktorial(N,X):- faktorial(N-1,Y), X is Y*N;`
2) `faktorial(1,1). faktorial(N,X):- faktorial(N,Y), Y is X*N;`
3) `faktorial(0,1). faktorial(N,X):- faktorial(N,X), X is N*(N-1);`
4) `faktorial(0,1). faktorial(N,Y):- faktorial(N-1,X), X is Y*(N-1).`

30. Каков будет результат выполнения программы:

```
clauses
Man ("Агамемнон"). Man ("Аид").
Man ("Атлант"). Man ("Гелиос").
Woman ("Автоноя"). Woman ("Агава").
Woman ("Антигона"). Woman ("Афродита"). Woman ("Галатея").
Parent ("Агамемнон", "Аид"). Parent ("Автоноя", "Аид").
Parent ("Гелиос", "Атлант"). Parent ("Галатея", "Атлант").
Parent ("Атлант", "Афродита"). Parent ("Антигона", "Афродита").
Mother(X,Y):-Parent(X,Y),Woman(X).
Father(X,Y):-Parent(X,Y),Man(X).
Daughter(X,Y):-Parent(X,Y),Woman(Y).
Sun(X,Y):-Parent(X,Y),Man(Y).
```

```
Predok(X,Y):-parent(X,Y).
Predok(X,Y):-Parent(Z,Y),Predok(X,Z).
?Father("Гелиос","Аид").
```

1) Yes; 2) No; 3) Гелиос; 4) Аид.

31. Каков будет результат выполнения программы:

```
clauses
Man ("Агамемнон"). Man ("Аид").
Man ("Атлант"). Man ("Гелиос").
Woman ("Автоноя"). Woman ("Агава").
Woman ("Антигона"). Woman ("Афродита"). Woman ("Галатея").
Parent ("Агамемнон", "Аид "). Parent ("Автоноя", "Аид").
Parent ("Гелиос", "Атлант"). Parent ("Галатея", "Атлант").
Parent ("Атлант", "Афродита"). Parent ("Антигона", "Афродита").
Mother(X,Y):-Parent(X,Y),Woman(X).
Father(X,Y):-Parent(X,Y),Man(X).
Daughter(X,Y):-Parent(X,Y),Woman(Y).
Sun(X,Y):-Parent(X,Y),Man(Y).
Predok(X,Y):-parent(X,Y).
Predok(X,Y):-Parent(Z,Y),Predok(X,Z).
?Mother(X,"Афродита")
```

1) X=Антигона; 2) X=Атлант; 3) X=Гелиос; 4) X=Галатея.

32. Каков будет результат выполнения программы:

```
clauses
Man ("Агамемнон"). Man ("Аид").
Man ("Атлант"). Man ("Гелиос").
Woman ("Автоноя"). Woman ("Агава").
Woman ("Антигона"). Woman ("Афродита"). Woman ("Галатея").
Parent ("Агамемнон ", "Аид"). Parent ("Автоноя", "Аид").
Parent ("Гелиос", "Атлант"). Parent ("Галатея", " Атлант").
Parent ("Атлант", "Афродита"). Parent ("Антигона", "Афродита").
Mother(X,Y):-Parent(X,Y),Woman(X).
Father(X,Y):-Parent(X,Y),Man(X).
Daughter(X,Y):-Parent(X,Y),Woman(Y).
Sun(X,Y):-Parent(X,Y),Man(Y).
Predok(X,Y):-parent(X,Y).
Predok(X,Y):-Parent(Z,Y),Predok(X,Z).
?Sun("Гелиос",X).
```

1) X=Атлант; 2) X=Галатея; 3) X=Афродита; 4) X=Аид.

33. Каков будет результат выполнения программы:

```
clauses
Man ("Агамемнон"). Man ("Аид").
Man ("Атлант"). Man ("Гелиос").
Woman ("Автоноя"). Woman ("Агава").
Woman ("Антигона"). Woman ("Афродита"). Woman ("Галатея").
Parent ("Агамемнон ", "Аид"). Parent ("Автоноя", "Аид").
Parent ("Гелиос ", "Атлант"). Parent ("Галатея", "Атлант").
Parent ("Атлант ", "Афродита"). Parent ("Антигона", "Афродита").
Mother(X,Y):-Parent(X,Y),Woman(X).
```

```
Father(X,Y):-Parent(X,Y),Man(X).
Daughter(X,Y):-Parent(X,Y),Woman(Y).
Sun(X,Y):-Parent(X,Y),Man(Y).
Predok(X,Y):-parent(X,Y).
Predok(X,Y):-Parent(Z,Y),Predok(X,Z).
?Predok(X,"Афродита").
```

1) X=Гелиос;
2) X=Галатея;
3) 2 solution X=Гелиос X=Галатея;
4) 4 solution X=Гелиос X=Галатея X=Атлант X=Антигона.

Язык программирования Лисп

1. *Атом* — это:
   1) простейшие команды;
   2) простейшие неделимые элементы — символы и числа;
   3) функция без вложений;
   4) список из одного элемента.
2. *Символ* — это:
   1) одна буква, обозначающая переменную;
   2) упорядоченная последовательность, элементами которой являются либо атомы, либо списки (подсписки);
   3) имя, состоящее из букв, цифр и специальных знаков, или число;
   4) имя функции.
3. *Предикат* — это:
   1) выражения-условия, которые могут быть истинными (T) или ложными (NIL);
   2) основным средством разветвления обработки;
   3) переменная, которая может принимать значения 0 или 1;
   4) оператор циклической конструкции.
4. Переменная INPUT — это:
   1) переменная, которой присваивается выражение, выводимое на стандартный вывод;
   2) переменная, которой присваивается выражение, прочитанное функцией Read;
   3) переменная, которой присваивается значение True или False;
   4) символ, который подается на вход.
5. *Список* — это:
   1) упорядоченная последовательность, элементами которой являются либо атомы, либо списки (подсписки). Списки заключаются в круглые скобки, а их элементы разделяются пробелами;
   2) неупорядоченная последовательность, элементами которой являются либо атомы, либо списки (подсписки). Списки заключаются в круглые скобки, а их элементы разделяются запятыми;
   3) упорядоченная последовательность, элементами которой являются либо атомы, либо списки (подсписки). Списки заключаются в фигурные скобки, а их элементы разделяются точками;
   4) неупорядоченная последовательность, элементами которой являются либо атомы, либо списки (подсписки). Списки заключаются в фигурные скобки, а их элементы разделяются запятыми.

6. *Лямбда-выражение* — это:
   1) безымянная функция, которая может использоваться для связывания формальных и фактических параметров на время вычислений;
   2) функция с названием LAMBDA, которая может использоваться для связывания формальных и фактических параметров на время вычислений;
   3) безымянная функция, которая может использоваться для связывания глобальных и локальных переменных на время вычислений;
   4) функция с названием LAMBDA, которая может использоваться для связывания глобальных и локальных переменных на время вычислений.
7. Предложение COND — это:
   1) функция с названием COND, которая может использоваться для связывания формальных и фактических параметров на время вычислений;
   2) функция, которая используется для создания связи переменных внутри формы;
   3) функция, которая является основным средством разветвления обработки;
   4) функция, которая является основным средством образования циклов.
8. *Управляющие предложения* — это:
   1) выражения, которые проверяют тождественность символов-аргументов;
   2) выражения, первый элемент которых действует как аргумент, а остальные элементы — как имена управляющих структур;
   3) выражения для занесения значений в ячейку памяти, связанной с символом;
   4) скобочные выражения, первый элемент которых действует как имя управляющей структуры, а остальные элементы — как аргументы.
9. Используемый в Лиспе подход к программированию основывается:
   1) на логическом подходе;
   2) на том, что вся обработка информации и получение искомого результата могут быть представлены в виде списков;
   3) на том, что вся обработка информации и получение искомого результата могут быть представлены в виде вложенных и/или рекурсивных вызовов процедур;
   4) на том, что вся обработка информации и получение искомого результата могут быть представлены в виде вложенных и/или рекурсивных вызовов функций.
10. В языке Лисп программы состоят из:
    1) величин; 2) объектов; 3) атомов; 4) молекул.
11. В языке Лисп главной структурой являются:
    1) переменные; 2) списки; 3) строки; 4) процедуры.
12. В языке Лисп списки описываются с помощью:
    1) `Begin End`; 2) `Set`; 3) `()`; 4) `{ }`.
13. В языке Лисп принята форма записи:
    1) постфиксная; 2) префиксная; 3) структурная; 4) линейная.
14. Основными методами программирования на Лиспе являются:
    1) декомпозиция и циклы; 2) объектно-ориентированный метод;
    3) индукция и дедукция; 4) композиция и рекурсия.
15. S-выражение в формах Бэкуса — Наура выглядит:
    1) `<S-выражение>::=<атом> | <список> <список>::=(<внутренняя часть>) <внутренняя часть>::=NIL | <S-выражение> [{внутренняя часть}] <атом>::=цепочка алфавитно-цифровых символов без пробелов или специальных символов (,);`

2) <S-выражение>::=<S-выражение>|<список> <список>::=(<внешняя часть>) <внешняя часть>::=NIL | <список> [{внутренняя часть}];
3) <S-выражение>::=NIL | <список> [{внутренняя часть}];
4) S-выражение>::=< атом > | <внутренняя часть>.

16. В форме вызова лямбда-выражения (лямбда-выражение a1 a2 ... an), где ai — это:
1) формальные параметры; 2) фактические параметры;
3) глобальные параметры; 4) локальные параметры.

17. Определить новую функцию можно с помощью:
1) DEFUN; 2) LET; 3) LAMBDA; 4) NTH.

18. Определение функции дается:
1) именем; 2) отдельной подпрограммой;
3) специальным оператором; 4) списком.

19. Передача параметров происходит:
1) по контексту; 2) по идентификатору; 3) по значению; 4) по имени.

20. Формальные параметры функций являются:
1) статическими и глобальными; 2) динамическими и глобальными;
3) статическими и локальными; 4) динамическими и локальными.

21. Знак ' используется:
1) для подавления вычисления аргументов функции LIST;
2) для занесения значения в ячейку памяти, связанной с символом;
3) для логического отрицания;
4) для подавления вычисления аргументов функции SET.

22. Каков будет результат вычисления выражения:

```
(+ (* x x) (* y y)).
(x 4)
(y 8)
```

1) 80; 2) 12; 3) 32; 4) 128;

23. Каков будет результат вычисления выражения:

```
((lambda (x y) (+ (* x x) (* y y))) 4 8);
```

1) 12; 2) 80; 3) 32; 4) 128.

24. Значением формы (defun sumsquare (x y) (+ (* x x) (* y y))) будет:
1) Defun; 2) x*x+y*y;
3) sumsquare; 4) (+ (* x x) (* y y)).

25. Каков будет результат вычисления выражения:

```
(defun sumsquare (x y) (+ (* x x) (* y y)));
(sumsquare 3 4)
```

1) 25; 2) 12; 3) 7; 4) sumsquare.

26. Каков будет результат вычисления функции (CAR (1 2 3 4)):
1) 2 3 4; 2) 1; 3) 1 2; 4) 3 4.

27. Каков будет результат вычисления функции (CDR (1 2 3 4)):
1) (2 3 4); 2) (1 2 3); 3) (1); 4) (3 4).

28. Каков будет результат вычисления функции (CONS 1 (2 3 4)):
1) (2); 2) (3 4); 3) (1 2 3 4); 4) (4).

29. Каков будет результат вычисления функции (ATOM A):
1) T; 2) Nil; 3) A; 4) B.

30. Каков будет результат вычисления функции (ATOM A B C):
1) T; 2) Nil; 3) A; 4) B.

31. Каков будет результат вычисления функции `(EQ X (CAR (X Y Z)))`:
1) T; 2) X; 3) Y Z; 4) X Y Z.
32. Каков будет результат вычисления функции `(EQUAL (x y z) (x w z))`:
1) T; 2) Nil; 3) (x y w z); 4) (w).
33. Каков будет результат вычисления функции `(EQUAL (x y z) Cons(x (y z)))`:
1) T; 2) Nil; 3) X; 4) y z.
34. Каков будет результат вычисления функции `(= 3 0.3e1)`:
1) T; 2) Nil; 3) 3;4) 6.

Правильные ответы

Языки программирования высокого уровня

№	1	2	3	4	№	1	2	3	4
1	X				10		X		
2		X			11				X
3	X				12	X			
4	X				13		X		
5			X		14				X
6		X			15		X		
7		X			16	X			
8				X	17	X			
9				X	18				X

Язык программирования Паскаль

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1			X		19		X		X	37			X	
2		X			20			X		38			X	
3		X			21		X			39	X			
4	X				22				X	40		X		
5				X	23	X				41				X
6	X				24				X	42			X	
7			X		25			X		43	X			
8		X			26				X	44	X			
9				X	27		X			45				
10				X	28	X				46	X			
11				X	29				X	47	X			
12	X				30	X				48		X		
13		X			31			X		49				X
14	X				32			X		50				X
15			X		33				X	51			X	
16		X			34		X			52		X		
17			X		35			X		53	X			
18	X				36	X				54		X		

Методы и искусство программирования

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1			X		9				X	17			X	
2	X				10			X		18			X	
3				X	11	X				19	X			
4		X			12			X		20			X	
5	X				13	X				21				X
6		X			14				X	22			X	
7		X			15			X		23	X			
8	X				16		X			24		X		

Язык программирования Бейсик

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1	X				12		X			23	X			
2		X			13				X	24	X			
3		X			14			X		25				X
4		X			15			X		26		X		
5	X				16	X				27		X		
6				X	17			X		28	X			
7			X		18		X			29			X	
8		X			19			X		30	X			
9		X			20		X			31			X	
10				X	21		X							
11		X			22				X					

Язык программирования Си

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1				X	16		X			31			X		46				X
2		X			17	X				32			X		47			X	
3		X			18		X			33				X	48		X		
4		X			19	X				34		X			49		X		
5	X				20		X			35			X		50	X			
6			X		21		X			36	X				51				X
7	X				22	X				37			X		52	X			
8	X				23	X				38		X			53	X			
9				X	24		X			39	X				54	X			
10				X	25			X		40			X		55			X	
11	X				26				X	41	X				56			X	
12	X				27				X	42		X			57			X	
13		X			28		X			43				X					
14				X	29	X				44	X								
15			X		30	X				45				X					

Язык программирования Пролог

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1			X		12		X			23		X		
2	X				13	X				24	X			
3		X			14				X	25	X			
4	X				15	X				26	X			
5		X			16		X			27	X			
6			X		17	X				28		X		
7			X		18			X		29	X			
8				X	19	X				30		X		
9	X				20		X			31	X			
10		X			21		X			32	X			
11				X	22	X				33				X

Язык программирования Лисп

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1		X			13		X			25	X			
2			X		14				X	26		X		
3	X				15	X				27	X			
4		X			16		X			28			X	
5	X				17	X				29	X			
6	X				18				X	30		X		
7			X		19			X		31	X			
8				X	20			X		32		X		
9				X	21				X	33	X			
10			X		22	X				34	X			
11		X			23		X							
12			X		24			X						

Глава 4

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Изучение принципов устройства ЭВМ является обязательной составной частью подготовки учителя информатики. Уровень, на котором сосредоточено это изучение — архитектура вычислительных систем. Сюда включаются вопросы управления работой ЭВМ (программирования) на языке машинных команд, без представления о котором невозможно понять реальную работу компьютера.

При проведении практических занятий по этой теме встает вопрос о выборе модели процессора, на примере которой проводится обучение. В данном практикуме для этого используется не какой-либо из реальных процессоров, а учебная модель Е-97, описанная в базовом учебнике. Преимущество этой модели состоит в том, что при относительной простоте она передает основные черты реального процессора, понимание которых необходимо для данной категории студентов.

Хотя основная часть лабораторно-практических занятий по этому разделу приходится на указанную выше тему, в целом подготовка по вычислительной технике включает изучение истории развития ЭВМ, принципов работы внешних устройств, а также логических основ функционирования ЭВМ. Основные формы проведения занятий по этим темам — семинары и написание рефератов.

§ 1. ИСТОРИЯ РАЗВИТИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Рекомендации по проведению занятий

Данная тема представляется важной с точки зрения развития общего кругозора студентов. Многие особенности современного состояния вычислительной техники, тенденции ее развития невозможно понять вне исторического анализа.

Занятия по этой теме проводятся путем разработки студентами рефератов и, возможно, прослушивания докладов по лучшим из них на семинарском занятии.

Краткие сведения

Основные вехи в развитии вычислительной техники

1. 1642 — 1645 гг. — создание Паскалем первых счетных машин.
2. 1822 — 1850 гг. — работы Ч.Бэббиджа по созданию автоматических цифровых вычислительных машин, разработка, так называемых, «принципов Бэббиджа».

3. 1904—1906 гг. — создание лампового диода и триода (Дж. Флеминг, Великобритания; Ли де Форест, США).

4. 1918—1919 гг. — создание лампового триггера (М. А. Бонч-Бруевич, СССР; У. Экклз и Ф. Джордан, Великобритания).

5. 1936 г. — теоретическая работа А. Тьюринга по алгоритмически универсальным вычислительным устройствам.

6. 1939 г. — создание настольной модели ЭВМ (Дж. Атанасов, США).

7. 1943—1944 гг. — создание электронно-вычислительной машины «Колосс» (Великобритания).

8. 1945—1946 гг. — создание электронно-вычислительной машины ЭНИАК (США, руководители работ Дж. Моучли и П. Эккерт).

9. 1946 г. — теоретическая работа Дж. фон Неймана, Г. Голдстейга и А. Беркса «Предварительное обсуждение логической конструкции ЭВМ», включавшей в себя основополагающие требования к архитектуре ЭВМ (так называемые, «принципы фон Неймана»).

10. 1949 г. — создание первой ЭВМ с хранимой программой ЭДСАК (Великобритания, руководитель работ М. Уилкс).

11. 1951—1953 гг. — начало серийного выпуска ламповых ЭВМ с хранимой программой.

12. 1958—1960 гг. — начало серийного выпуска полупроводниковых ЭВМ.

13. 1965 г. — начало выпуска ЭВМ на интегральных схемах.

14. Середина 60-х годов — начало выпуска серии ЭВМ IBM 360 (впоследствии IBM 370).

15. Середина 70-х годов — начало выпуска ЭВМ на больших интегральных схемах.

16. Середина 70-х годов — начало выпуска супер-ЭВМ.

17. 1971 г. — создание первого микропроцессора (фирма «Intel», США).

18. 1976—1977 г. — создание первых персональных ЭВМ «Apple» (С. Возняк, С. Джобс, США).

19. Конец 80-х годов — начало выпуска «интеллектуальных» ЭВМ 5-го поколения.

Основные вехи в истории развития отечественной вычислительной техники

1. 1950—1951 гг. — создание первой отечественной ЭВМ МЭСМ (руководитель работ С. А. Лебедев).

2. 1952 г. — создание машины БЭСМ, наиболее быстродействующей в Европе ЭВМ.

3. 1953—1955 гг. — начало серийного производства ЭВМ («Стрела», «Урал» и др. конструкторов Ю. А. Базилевского, Б. И. Рамеева и др.).

4. Середина 60-х годов — создание ЭВМ БЭСМ-6, одной из крупнейших в мире (в свое время).

5. Середина 60-х годов — создание и серийный выпуск ЭВМ 3-го поколения «Минск», «Урал», М-220 и др.

6. Середина 70-х годов — начало выпуска серий ЕС ЭВМ, СМ ЭВМ, «Электроника», ориентированных на зарубежные модели.

Контрольные вопросы

1. Каковы основные вехи в «докомпьютерном» развитии вычислительной техники?

2. В чем состоят основные открытия Ч.Бэббиджа?
3. В чем заключаются принципы архитектуры ЭВМ фон Неймана?
4. Каким образом в современной вычислительной технике преодолеваются ограничения, связанные с принципами фон Неймана?
5. В чем заключается классификация ЭВМ по поколениям?
6. Как выглядит функциональная классификация ЭВМ?
7. Каковы перспективы совершенствования технической базы ЭВМ? их структурной организации?

Темы для рефератов

1. Докомпьютерная история развития вычислительной техники.
2. Вклад Ч. Бэббиджа в разработку принципов функционирования автоматических цифровых вычислительных машин.
3. Работы Дж. фон Неймана по теории вычислительных машин.
4. История создания и развития ЭВМ 1-го поколения.
5. История создания и развития ЭВМ 2-го поколения.
6. История создания и развития ЭВМ 3-го поколения.
7. История создания и развития ЭВМ 4-го поколения.
8. Микропроцессоры, история создания, использование в современной технике.
9. Персональные ЭВМ, история создания, место в современном мире.
10. Супер-ЭВМ, назначение, возможности, принципы построения.
11. Проект ЭВМ 5-го поколения: замысел и реальность.
12. Многопроцессорные ЭВМ и распараллеливание программ.

Темы семинарских занятий

1. «Докомпьютерная» история развития вычислительной техники.
2. История развития вычислительной техники с момента создания первой ЭВМ.

Дополнительная литература

1. *Апокин И.А., Майстров Л.Е.* История вычислительной техники. — М.: Наука, 1980.
2. *Апокин И.А., Майстров Л.Е., Эдлин И.С.* Чарльз Бэббидж. — М.: Наука, 1981.
3. *Васильев Б.М., Частиков А.П.* Микропроцессоры (история, применение, технология) // Информатика и образование. — 1993. — № 5. — С. 71—83.
4. *Гутер Р.С., Полунов Ю.Л.* От абака до компьютера. — М.: Знание, 1975.
5. *Данилов Ю.А.* Джон фон Нейман // Новое в науке, жизни, технике (Сер. «Математика, кибернетика»). — М.: Знание, 1990. — № 12.
6. *Данишевский Л.Н., Шкабара Е.А.* Как это начиналось (Воспоминания об истории создания первой отечественной вычислительной машины — МЭСМ) // Новое в науке, жизни, технике (Сер. «Математика, кибернетика»). — М.: Знание, 1981. — № 1.
7. Информатика. Энциклопедический словарь для начинающих. — М.: Педагогика-Пресс, 1994.
8. *Эндерлайн Р.* Микроэлектроника для всех. — М.: Мир, 1979.

§ 2. АРХИТЕКТУРА ЭВМ

Рекомендации по проведению занятий

Занятия по этой теме проводятся на семинарских занятиях, на которых отрабатываются принципы построения архитектуры ЭВМ, и путем разработки студентами рефератов. Эти занятия не предполагают анализа архитектуры конкретных ЭВМ и изучения конкретных наборов команд и методов адресации данных; соответствующие вопросы отнесены к следующим параграфам.

Краткие сведения

Под архитектурой ЭВМ понимают наиболее общие принципы построения вычислительных систем, реализующие программное управление работой и взаимодействие основных функциональных узлов.

К архитектуре относят:

- структуру памяти ЭВМ;
- способы доступа к памяти и внешним устройствам;
- возможности изменения конфигурации компьютера;
- систему команд;
- форматы данных;
- организацию интерфейса.

Классические принципы построения архитектуры ЭВМ были предложены в работе Дж. фон Неймана, Г. Голдстейга и А. Беркса в 1946 г. и известны как «принципы фон Неймана». Они таковы:

- использование двоичной системы представления данных;
- принцип хранимой программы;
- принцип последовательного выполнения операций;
- принцип произвольного доступа к ячейкам оперативной памяти.

В ходе эволюции ЭВМ, с созданием микропроцессоров, с появлением интеллектуальных контроллеров совершен переход к шинной архитектуре ЭВМ. Процессор перестал быть центром конструкции, стало возможным реализовывать прямые связи между устройствами.

Важную роль стали играть средства сетеобразования. Радикально увеличилась номенклатура и возможности периферийных устройств накопления, ввода и вывода информации.

Контрольные вопросы

1. Дайте развернутое описание того, что понимается под термином «архитектура ЭВМ».
2. Как выглядит структурная схема ЭВМ, построенной на принципах фон Неймана?
3. Как в неймановской архитектуре реализуются:
   - принцип хранимой программы;
   - принцип последовательного выполнения операций;
   - принцип произвольного доступа к ячейкам оперативной памяти?

4. Как выглядит структурная схема ЭВМ, построенной на принципах шинной архитектуры?

5. Как трансформируются принципы фон Неймана при переходе к шинной архитектуре ЭВМ?

6. Что такое шина данных? шина адреса? шина управления?

7. Для чего нужна видеопамять?

8. В чем состоит основной цикл работы ЭВМ неймановской структуры?

9. Какие группы команд обработки информации являются стандартными, не зависящими от конкретной ЭВМ?

10. В чем отличие CISC и RISC подходов в построении системы команд компьютера?

11. Какова структура команды ЭВМ?

12. Чем отличаются одно-, двух- и трехадресные команды?

Темы для рефератов

1. Детальное описание архитектуры фон-неймановских машин.
2. Детальное описание шинной архитектуры ЭВМ.
3. Системы команд машин различных поколений, адресация памяти.

Темы семинарских занятий

1. Развитие архитектуры ЭВМ.
2. Система команд ЭВМ и способы обращения к данным.

Дополнительная литература

1. *Александриди Т. М.* и др. Работа на микро-ЭВМ. — М.: Финансы и статистика, 1989.

2. *Брой М.* Информатика: В 3 т. Т.2. Вычислительные структуры и машинно-ориентированное программирование: Пер. с нем. — М.: Диалог-МИФИ, 1996.

3. *Брусенцов Н. П.* Микрокомпьютеры. — М.: Наука, 1985.

4. *Васильев Б. М., Частиков А. П.* Микропроцессоры (история, применение, технология) // Информатика и образование. — 1993. — № 5. — С. 71—83.

5. *Еремин Е.А.* Как работает современный компьютер. — Пермь: изд-во ПРИПИТ, 1997.

6. *Зальцман Ю.А.* Архитектура и программирование на языке ассемблера БК-0010 // Информатика и образование. — 1990. — № 4—6; 1991. — № 1—5.

7. Информатика. Энциклопедический словарь для начинающих. — М.: Педагогика-Пресс, 1994.

8. *Лин В.* PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера: Пер. с англ. — М.: Радио и связь, 1989.

9. *Майоров С.А., Кириллов В.В., Приблуда А.А.* Введение в микро-ЭВМ. — Л.: Машиностроение. Ленингр. отд-ние, 1988.

10. *Нортон П., Соухэ Д.* Язык ассемблера для IBM PC: Пер. с англ. — М.: Финансы и статистика, 1992.

11. *Перегудов М.А., Халамайзер А.Я.* Бок о бок с компьютером. — М.: Высш. шк., 1987.

12. *Смит Б.Э., Джонсон М.Т.* Архитектура и программирование процессора INTEL 80386: Пер. с англ. — М.: Конкорд, 1992.

13. *Частиков А.П.* История компьютера // Информатика и образование. — 1996.
14. *Ямпольский В.С.* Основы автоматики и электронно-вычислительной техники. — М.: Просвещение, 1991.

§ 3. АРХИТЕКТУРА МИКРОПРОЦЕССОРОВ

Рекомендации по проведению занятий

Как отмечалось во введении к данному разделу, программирование на уровне ассемблера в данном пособии отрабатывается на примере учебной ЭВМ, чему посвящен следующий параграф. В силу этого общие вопросы архитектуры микропроцессоров, рассматриваемые в данном параграфе, изучаются на семинарских занятиях и путем работы над рефератами, без решения задач на программирование.

Краткие сведения

Основные функции микропроцессора:
- выборка команд из ОЗУ;
- декодирование команд;
- выполнение команд;
- управление обменом информацией между различными разделами памяти (включая собственные регистры);
- обработка прерываний;
- обработка сигналов от внешних устройств;
- управление устройствами, входящими в состав компьютера.

Процессор, с функциональной точки зрения, состоит из устройства управления, арифметико-логического устройства и регистров памяти. Программист, работающий на уровне ассемблера, обращается к части регистров памяти непосредственно (так называемые, программно-доступные регистры), а другой частью (программно-недоступными регистрами) пользуется неявно. Некоторые регистры (например, счетчик адреса команд) в некоторых микропроцессорах являются программно-доступными, а в некоторых — недоступными. Программно-доступные регистры подразделяются на специализированные (указатель стека, регистр состояния) и общего назначения; в последние можно записывать команды программы.

Важной характеристикой процессора является ***разрядность***. Это понятие включает в себя:
- разрядность внутренних регистров;
- разрядность шины данных;
- разрядность шины адреса.

Наиболее значимым, с точки зрения программиста, является первый показатель.

Адресное пространство микропроцессора состоит из ячеек памяти ОЗУ, из которых можно брать информацию или засылать ее. Способы задания адресов в командах ЭВМ называют *методами адресации*. При *прямой адресации* адрес находится в самой команде; в современных процессорах этот метод практически не используется. При *косвенной адресации* адрес памяти заносится в один из регистров общего назначения, а в команде содержится лишь ссылка на этот регистр. Существует несколько вариантов косвенной адресации: регистровая, косвенно-регистровая,

автоинкрементная, автодекрементная и др. Наконец, данное (операнд) может находиться в одном из регистров процессора или даже входить в состав команды (так называемые, операции с константой).

Существует и неявный, без указания на адреса ОЗУ, способ адресации — *стек*, при котором информация записывается и считывается последовательным образом.

Важной характеристикой процессора является список используемых им форматов данных. Широко распространенными являются следующие форматы: *целые числа без знака*, *целые числа со знаком*, *коды символов*, *битовые поля*.

В наибольшей мере возможности процессора характеризуются его системой команд. Каждая команда включает код операции и от нуля до трех адресов. Команды различаются по функциям (команды переходов, команды арифметических действий, команды сдвига и т.д.) и по числу адресов. У современных микропроцессоров используется сочетание безадресных команд, одно- и двухадресных команд.

Важную роль в работе процессора играет *обработка прерываний*. Основные их виды — *внутрипроцессорные прерывания* и *прерывания от внешних устройств*. Часть прерываний могут быть замаскированы, т.е. их исполнение и обработка отложены. При обработке незамаскированного прерывания процессор делает следующее:

- запоминает состояние прерванной программы;
- распознает источник прерывания;
- запускает системную программу обработки прерываний;
- восстанавливает состояние прерванной программы и, при возможности, продолжает ее исполнение.

Контрольные вопросы

1. Каковы наиболее значимые этапы в истории развития микропроцессоров?
2. Какова внутренняя организация микропроцессора?
3. Каковы функции регистров: адреса команд, указателя стека, регистра состояния?
4. Как могут соотноситься разрядность шины управления, шины адресов и шины данных?
5. Какие бывают методы адресации данных и в чем они состоят?
6. В чем особенности адресации данных при работе со стеком?
7. В чем заключается обработка прерываний?
8. Как (в принципе) работает микропроцессор с внешними устройствами?
9. Приведите пример системы команд (частично) одного из реальных микропроцессоров.

Темы для рефератов

1. Архитектура процессоров машин 2-го и 3-го поколений.
2. Архитектура микропроцессора семейства PDP.
3. Архитектура микропроцессора семейства Intel.

Темы семинарских занятий

1. Форматы команд и данных.
2. Методы адресации. Стек. Обработка прерываний.

Дополнительная литература

1. *Брой М.* Информатика: В 3 т. Т.2. Вычислительные структуры и машинно-ориентированное программирование: Пер. с нем. — М.: Диалог-МИФИ, 1996.

2. *Брусенцов Н. П.* Микрокомпьютеры. — М.: Наука, 1985.

3. *Васильев Б. М., Частиков А. П.* Микропроцессоры (история, применение, технология) // Информатика и образование. — 1993. — № 5.

4. *Зальцман Ю. А.* Архитектура и программирование на языке ассемблера БК-0010 // Информатика и образование. — 1990. — № 4.

5. *Лин В.* PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера. — М.: Радио и связь, 1989.

6. *Майоров С. А., Кириллов В. В., Приблуда А. А.* Введение в микро-ЭВМ. — Л.: Машиностроение. Ленигр. отд-ние, 1988.

7. *Мальцева Л. А., Фромберг Э. М., Ямпольский В. С.* Основы цифровой техники. — М.: Радио и связь, 1986.

8. *Нортон П., Соухэ Д.* Язык ассемблера для IBM PC: Пер. с англ. — М.: Финансы и статистика, 1992.

9. *Паскалев Ж.* Первые шаги в вычислительной технике. — М.: Радио и связь, 1987.

10. *Смит Б. Э., Джонсон М. Т.* Архитектура и программирование процессора INTEL 80386: Пер. с англ. — М.: Конкорд, 1992.

11. *Тотхейм Я.* Основы цифровой электроники: Пер. с англ. — М.: Мир, 1988.

12. *Эндерлайн Р.* Микроэлектроника для всех: Пер. с нем. — М.: Мир, 1979.

13. *Ямпольский В. С.* Основы автоматики и электронно-вычислительной техники. — М.: Просвещение, 1991.

§ 4. УЧЕБНАЯ МОДЕЛЬ МИКРОКОМПЬЮТЕРА

Рекомендации по проведению занятий

Цели выполнения работ данного параграфа:

- знакомство с архитектурой ЭВМ и микропроцессоров четвертого поколения;
- выработка и закрепление практических навыков в освоении методологии программирования на низком уровне;
- практическая реализация принципов и способов представления информации в памяти ЭВМ, освоение на практике управления вычислительными и другими процессами в современных ЭВМ;
- освоение первичных навыков программирования при реализации простейших учебных задач.

При разработке и отладке программ для учебного микрокомпьютера целесообразно действовать так:

1) разработать план решения задачи (составить алгоритм решения);

2) распределить память для аргументов, результатов и промежуточных величин, которые будут использованы при работе программы;

3) подобрать набор тестовых данных для последующей проверки правильности программы;

4) составить программу согласно плану решения (алгоритму);

5) ввести программу и исходные данные;

6) проверить правильность алгоритма и программы с помощью тестов.

Если результаты тестирования окажутся положительными, то программа пригодна для проведения расчетов по ней. В противном случае следует приступить к ее отладке с целью выявления алгоритмических или других ошибок, после чего вернуться к тестированию.

Важной частью данного параграфа практикума являются лабораторные работы, выполняемые студентами практически самостоятельно. На них требуется порядка 20 аудиторных часов. Оценка трудоемкости каждой работы исходит из практического опыта их реализации и из того, что студенты:

- предварительно подготовились к выполнению работы, освоили соответствующий теоретический материал;
- имеют практически завершенное решение задачи в виде алгоритма и кода;
- в достаточной мере владеют навыками работы с компьютером;
- имеют устойчивые навыки самостоятельной работы.

Примечание. Работу № 5 выполняют только те студенты, которые успешно справились с предыдущими работами, и у которых имеется резерв времени.

В конце параграфа приведены задачи повышенной трудности, которые могут быть полезны при проведении семинарских занятий и самостоятельной работы студентов.

Краткие сведения

Микрокомпьютер «Е97» представляет собой учебную модель компьютеров четвертого поколения и предназначен для изучения основ функционирования современных персональных ЭВМ. Программа разработана Е. А. Ереминым в 1997 г.

В состав «Е97» (рис. 4.1) входят центральный процессор, оперативное и постоянное запоминающие устройства (ОЗУ и ПЗУ), дисплей и клавиатура.

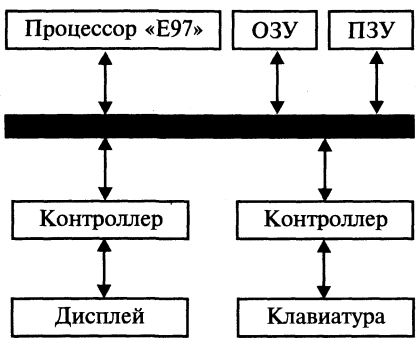


Рис. 4.1. Состав учебной машины «Е97»

Основным устройством в «Е97», построенном на основе неймановской архитектуры, является шестнадцатиразрядный процессор «Е97». Его система команд позволяет обрабатывать двухбайтовые машинные слова и отдельные байты.

Процессор (рис. 4.2) содержит программно доступные регистры — четыре регистра общего назначения (**R0—R3**), указатель стека (**SP** — stack pointer); служебные регистры — счетчик адреса команд (**PC** — programm counter), регистр состояния процессора (**PS**); и программно недоступные (внутренние), которые процессор использует в своих целях — регистр команд (**PK**), регистры операндов (**Рг1**, **Рг2**) и сумматор (**См**).

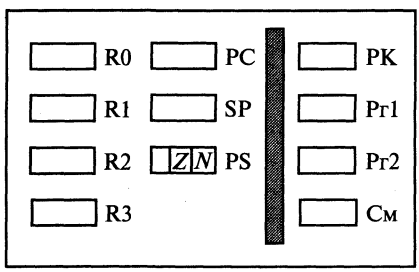


Рис. 4.2. Процессор «Е97»

Счетчик адреса команд **PC** при работе программы всегда указывает на адрес очередной команды, выполняя которую, автоматически изменяется согласно основному алгоритму работы процессора. Регистры об-

щего назначения можно рассматривать как внутреннюю память процессора. Они используются для промежуточного сохранения данных и адресов данных.

Напомним основной алгоритм работы процессора:

1) считывается команда из оперативной памяти по адресу, указанному в счетчике команд, и записывается в регистр команд;

2) счетчик команд автоматически увеличивается так, чтобы он содержал адрес следующей команды (в «E97» автоматически увеличивается на2);

3) содержимое регистра команд дешифруется, из памяти выбираются значения операндов и размещаются в регистрах операндов, выполняется нужная команда; в случае необходимости результат записывается в ОЗУ;

4) осуществляется переход к п.1.

Согласно этому алгоритму процессор исполняет программу до тех пор, пока не встретится команда СТОП.

Указатель стека **SP** чаще всего используется при организации подпрограмм, о чем речь пойдет ниже, а также для промежуточного хранения данных.

Регистр состояния процессора (точнее говоря, отдельные его биты) анализируется при организации переходов, прерываний и т.д. Реально далее будут использованы только два бита: **Z** и **N**. Значения этих битов при возможных значениях результатов операций указаны ниже в таблице.

Результат	Значение бита **Z**	Результат	Значение бита **N**
Нулевой (=0)	1	Отрицательный (<0)	1
Ненулевой (<>0)	0	Неотрицательный (=>0)	0

При работе программы в регистр команд считывается очередная команда для дешифрования и дальнейшего выполнения; в регистры операндов считываются согласно выполняемой команде операнды из регистров общего назначения или из оперативной памяти; выполняется необходимое действие, результат которого помещается в сумматор и сохраняется до тех пор, пока не будет выполнено следующее действие; из сумматора результат копируется в один из регистров общего назначения или в оперативную память.

Последние четыре регистра доступны лишь для наблюдения, их содержимое программист изменить не может (в отличие от программно доступных).

Минимальной адресуемой ячейкой памяти в микрокомпьютере «E97», как и в любой современной персональной ЭВМ, является байт. Номера байтов (адреса) лежат в диапазоне от $0000_{(16)}$ до $FFFF_{(16)}$. Гораздо чаще приходится работать с машинными словами (в «E97» они двухбайтовые). Напомним, что это связано с разрядностью процессора. Адрес машинного слова совпадает с номером младшего байта, входящего в слово, поэтому адреса всех машинных слов четные. Для размещения программы и данных в распоряжении программиста существует ОЗУ (с адреса 0000 до адреса 00FE включительно). Программа и данные записываются в память «E97» (в том числе и в регистры) в шестнадцатеричном коде. ПЗУ содержит полезные подпрограммы автора «E97» и размещается с адреса 4000 по адрес 4106. Список этих подпрограмм — в лабораторной работе №5.

Рассмотрим два основных способа адресации данных в «E97»: **прямой регистровый** — когда данные для обработки содержатся в регистрах, и **косвенный регистровый** — когда данные расположены в ОЗУ, а их адреса находятся в регистрах общего назначения. Указанные способы адресации кодируются следующим образом:

	Код операнда	
Регистр	Регистровая адресация	Косвенная адресация
R0	0	4
R1	1	5
R2	2	6
R3	3	7

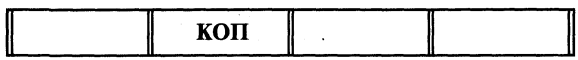
Условным обозначением косвенной адресации, в отличие от регистровой, служит название (идентификатор) регистра, заключенное в круглые скобки. Например: (R2).

Кроме рассмотренных способов адресации имеется еще адресация по командному счетчику PC. Если в качестве операнда указано значение **D**, то соответствующий операнд входит непосредственно в команду и расположен в ОЗУ по следующему за командой адресу; если — **E**, то по следующему за командой адресу указан адрес, где хранится величина. Более подробно об этих способах адресации — в примерах.

Замечание. Код **D** может выступать только в качестве первого операнда (в силу того, что результат всегда помещается во второй операнд).

Познакомимся с некоторыми командами процессора «E97».Все их можно разделить на безадресные (без операндов), одноадресные (с одним операндом) и двухадресные (с двумя операндами).

Безадресная команда имеет формат



Третий полубайт (отсчет ведется справа налево), как и в других командах, занимает **код операции (КОП)** — то действие, которое необходимо выполнить. Другие полубайты могут содержать все, что угодно, так как они не задействованы; по традиции туда записывают нули.

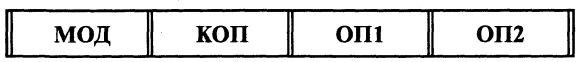
К безадресным относятся команды **нет операции (0)** и **стоп (F)**. Согласно принятым соглашениям в полной форме эти команды записываются так:

0000 (нет операции);

0F00 (стоп).

Всякую программу обязательно должна завершать команда СТОП.

Двухадресная команда имеет формат



На приведенной схеме **модификатор (МОД)** — некоторый вспомогательный код; ОП1 и ОП2 — операнды в команде.

Вот список таких команд:

Команда	Код	Команда	Код
Переписать	1	Логическое И (AND)	7
Сложить	2	Логическое ИЛИ (OR)	8
Вычесть	3	Исключающее ИЛИ (XOR)	9
Сравнить	4	Ввести в порт	A
Умножить	5	Вывести из порта	B
Разделить	6		

Действия выполняются по схеме

$$\text{ОП2} \oplus \text{ОП1} \rightarrow \text{ОП2},$$

где ⊕ — операция. Как видно из схемы, конечный результат операции всегда помещается во второй операнд.

Особое место занимает операция сравнения: при ее выполнении содержимое операндов не изменяется, а действие ОП2 − ОП1 выполняется лишь для установления значений управляющих битов **Z** и **N** регистра состояния **PS**.

В каждой из указанных здесь команд значение МОД (модификатора) равно нулю. Модификатор может быть отличен от нуля (об этом — позднее).

Приведем примеры использования команд:

1) 0123 — переслать содержимое регистра R2 в регистр R3, содержимое R2 сохраняется;

2) 0534 — умножить содержимое регистра R3 на содержимое ячейки памяти, адрес которой указан в регистре R0, результат поместить в эту ячейку памяти.

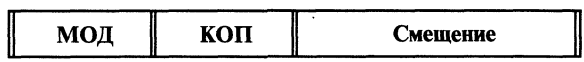
Для реализации ***развилок*** и ***циклов*** в системе команд «Е97» есть две ***команды перехода — абсолютного*** и ***относительного.***

Наиболее простой является команда абсолютного перехода. В этом случае просто необходимо указать адрес, куда надо попасть.

Структура команды следующая:

Код команды абсолютного перехода — **C**.

Команда относительного перехода имеет следующий формат:



Смещение указывает, на сколько необходимо изменить текущее значение счетчика команд при организации перехода.

Значение модификатора в команде в большинстве случаев зависит от управляющих битов **Z** и **N** регистра состояния **PS**. Следующая таблица показывает возможные значения модификатора:

МОД	Условие перехода
0	Возврат из подпрограммы
1	Безусловный переход
2	Результат неотрицателен (N = 0)
3	Результат отрицателен (N = 1)
4	Результат ненулевой (Z = 0)
5	Результат нулевой (Z = 1)
6	Результат неположительный (N = 1 или Z = 1)
7	Результат положительный (N = 0 или Z = 0)
9	Вызов подпрограммы

Код команды относительного перехода — **D**.

Если переход осуществляется в сторону увеличения адресов, то смещение будет положительным, в сторону уменьшения адресов — отрицательным. Смещение записывается в дополнительном коде.

Развилка может быть полной или неполной. Полная развилка имеет структуру

```
если <условие> то <серия 1> иначе <серия 2>
```

Здесь <условие> — это отношение между двумя величинами. Например, $x \le y$; $z > 10$. <серия 1>, <серия 2> — действия, которые необходимо осуществить соответственно в случае истинности или ложности условия. Например:

```
если a < b то min := a иначе min := b.
```

В примере выбирается наименьшее из двух целых чисел *a*, *b*.

Неполная развилка имеет структуру

```
если <условие> то <серия 1>
```

В этом случае действия выполняются только в случае истинности условия, в противном случае никаких действий выполнять не надо. Например:

```
если a < 0 то a := -a.
```

В примере данное число *a* заменяется модулем.

При реализации развилки (как полной, так и неполной) с помощью системы команд «Е97» обычно используется несколько различных команд (в их числе в большинстве случаев должна быть команда сравнения для установки управляющих битов **Z**, **N** регистра состояния **PS**; очень часто присутствуют команды условного и безусловного перехода).

При обращении к ***подпрограммам*** в «Е97» (и не только в ней) используется ***стек***. Стек позволяет по окончанию работы подпрограммы обеспечить возврат в ту точку основной программы, которая следует сразу же за вызовом подпрограммы. При обращении к вспомогательному алгоритму адрес в указателе стека (**SP**) уменьшается на 2, по вновь полученному адресу записывается адрес, следующий за вызовом подпрограммы. Подпрограмма исполняется и, когда встречается команда возврата (0D00), в счетчик команд помещается адрес, на который указывает **SP**, значение **SP** увеличивается на 2. Так обеспечивается продолжение исполнения основной программы.

Из сказанного выше следует, что в **SP** надо поместить адрес, свободный от данных и программы; несколько предшествующих адресов также должны быть свободны, поскольку стек строится в сторону уменьшения адресов.

Удобным способом выполнения ***действия с константой***, значение которой не превышает по абсолютной величине $15_{(10)}$ ($F_{(16)}$), является операция с *короткой константой*. В этом случае место первого операнда (ОП1) в двухадресной команде занимает модуль константы. На знак константы указывает модификатор. Если его значение $2_{(16)}$, то константа положительная; $3_{(16)}$ — отрицательная. Указанный способ оказывается экономичнее рассмотренного ранее способа адресации по **PC**, так как команда в этом случае занимает одно слово вместо двух.

Например, команда «переслать по адресу, записанному в регистре R0, число 3»

```
0020 01D4
0022 0003
```

может быть заменена на такую:

```
0020 2134
```

которая выполняет то же самое действие.

Контрольные вопросы

1. Что такое *дискретность памяти* ЭВМ?
2. Что такое *адресуемость памяти* ЭВМ?
3. Что такое *регистр процессора*?
4. Какие виды памяти можно выделить в ЭВМ? В чем различие между ними?
5. Сформулируйте и разъясните принципы Дж. фон Неймана функционирования ЭВМ.
6. Для чего нужна внешняя память?
7. Какие устройства используются в качестве внешней памяти ЭВМ?
8. Из чего состоит основная память? Дайте характеристику каждой компоненты.
9. В чем различие внутренней и внешней памяти ЭВМ?
10. Если выключить компьютер, что произойдет с содержимым внутренней и внешней памяти?
11. В какой (внутренней или внешней) памяти должна находиться выполняемая в данный момент программа?
12. Что такое *адресное пространство*?
13. Как определить *объем* адресного пространства?
14. Что такое *режимы* адресации к памяти?
15. Расскажите о *режимах адресации* персональных компьютеров.
16. Расскажите о *регистровой адресации.*
17. Расскажите о *косвенно-регистровой адресации.*
18. Расскажите об *автоинкрементной адресации.*
19. Расскажите об *автодекрементной адресации.*
20. Что такое микропроцессор (МП)?
21. Расскажите о поколениях МП и их основных характеристиках.
22. Из чего состоит МП?
23. Что такое РОН?
24. Что такое *регистр состояния микропроцессора*? Каково назначение битов **N** и **Z** регистра состояния **PS**? Расскажите о назначении других битов **PS**.
25. Какие регистры являются программно-управляемыми? программно-неуправляемыми?
26. Каково назначение следующих регистров: *счетчика команд*, *регистра команд*, *регистра адреса*?
27. Что такое *система команд* МП? классификация команд? Расскажите о назначении команд.
28. Расскажите об *алгоритме* работы микропроцессора.
29. Расскажите о *форматах* команд микропроцессора.
30. Что такое *безусловный переход*? В каких случаях он применяется?
31. Как организовать *условный переход*?
32. Как вычислить *смещение* при организации перехода?
33. Чем отличается абсолютный переход от относительного?
34. Какие *виды циклов* существуют?
35. Как организовать *цикл*?
36. Что такое *стек*?
37. Что такое *подпрограмма*?
38. В каких случаях целесообразно использовать подпрограммы?
39. Объясните механизм обращения к подпрограмме.
40. Почему использование подпрограмм облегчает отладку программы?

41. Существуют ли ограничения на вложенность подпрограмм друг в друга? Может ли одна подпрограмма вызывать другую?

42. Какие подпрограммы называются *рекурсивными*?

Тема для рефератов

Учебные ЭВМ.

Темы семинарских занятий

1. Принципы программирования на ассемблере; особенности «E97».
2. Особенности программирования развилок и циклов.
3. Использование подпрограмм.

Рекомендации по программному обеспечению

1. Компилятор «E97» (может быть получен при обращении к разработчику по адресу eremin@pspu.ac.ru).

Задачи и упражнения

Структура процессора и памяти. Способы адресации данных. Система команд

Пример 1. В регистрах R0, R1 содержатся целые числа. Получить сумму этих чисел в регистре R3, не изменяя содержимого R0, R1.

План решения:

1. Переслать R0 в R3.
2. Сложить R1 с R3.

Распределять память в этой задаче уже не требуется, так как это сделано согласно условию.

Тест

R0	R1	R2	R3
FFFE	5	—	3

Программа

Адрес	Команда	Действие	Замечание
0000	0103	R0 → R3	R3 := R0
0002	0213	R1 + R3 → R3	R3 := R3 + R1
0004	0F00	Стоп	

При решении задачи использовалась только регистровая адресация.

Пример 2. В последовательных ячейках памяти расположены пять целых чисел. Получить произведение этих чисел. Содержимое памяти не изменять.

Идея решения. Поскольку величины хранятся в памяти последовательно, в одном из регистров будем изменять адреса от адреса первого числа до последнего, а в другом — накапливать произведение.

Распределение памяти

R0	R1	R2	R3
Адрес очередного числа	—	—	P

Заданные числа будем хранить с адреса 0050.

Тест

Адрес	0050	0052	0054	0056	0058
Величина	0002	FEFE	000A	FEFE	0003

Ответ: $120_{(10)} = 78_{(16)}$.
Замечание. Целые числа записываются в память в дополнительном коде.

Программа

Адрес	Команда	Действие	Замечание
0000	0143	(R0) → R3	R3 := (R0)
0002	02D0	R0 + 2 → R0	Адрес второго числа
0004	0002	—	
0006	0543	R3 * (R0) → R3	R3 := R3 * (R0)
0008	02D0	R0 + 2 → R0	Адрес третьего числа
000A	0002	—	
000C	0543	R3 * (R0) → R3	R3 := R3 * (R0)
000E	02D0	R0 + 2 → R0	Адрес четвертого числа
0010	0002	—	
0012	0543	R3 * (R0) → R3	R3 := R3 * (R0)
0014	02D0	R0 + 2 → R0	Адрес пятого числа
0016	0002	—	
0018	0543	R3 * (R0) → R3	R3 := R3 * (R0)
001A	0F00	Стоп	

При решении задачи использованы способы адресации: регистровая, косвенная и по счетчику команд.

Пример 3. Дано натуральное трехзначное число, записанное в десятичной системе счисления. Получить число, записанное теми же цифрами, но расположенными в обратном порядке.

Идея решения. Пусть данное число имеет вид $N = \overline{abc}$, тогда ответ должен быть $M = cba = c \cdot 100 + b \cdot 10 + a$. Из исходного числа N цифры могут быть получены так: $a = N \operatorname{div} 100$; $b = (N - a \cdot 100) \operatorname{div} 10$; $c = N - a \cdot 100 - b \cdot 10$.

Здесь через div обозначена операция целочисленного деления одного целого числа на другое, которая и используется в «Е97».

План решения:

1. Запомнив исходное число N, разделить его на 100; полученную цифру a запомнить.
2. Получить часть числа M: $M := a$.
3. Вычесть из числа N произведение $a \cdot 100$; результат запомнить.
4. Разделить разность $N - a \cdot 100$ на 10; полученную цифру b запомнить.
5. Добавить к числу M очередную его часть: $M := M + b \cdot 10$.
6. Вычесть из разности $N - a \cdot 100$ число $b \cdot 10$; получим цифры c.
7. Добавить к числу M последнюю его часть: $M := M + c \cdot 100$.
8. Стоп.

Распределение памяти

R0	R1	R2	R3
N и разности	Цифры и произведение	—	M

Тест. Дано: $684_{(10)} = 2AC_{(16)}$. Ответ: $486_{(10)} = 1E6_{(16)}$.

Программа

Адрес	Команда	Действие	Замечание
0000	0101	R0 → R1	
0002	026D1	R1 / 100	$a := N / 100$
0004	0064		
0006	0113	R1 → R3	$M := a$
0008	05D1	R1 * 100	$a * 100$
000A	0064		
000C	0310	R0 := R0 − R3	$N := N - a * 100$
000E	0101	R0 → R1	
0010	06D1	R1 / 10	$b := (N - aq * 100) / 10$
0012	000A		
0014	05D1	R1 * 10	
0016	000A		
0018	0213	R3 := R3 + R1	$M := M + b * 10$
001A	0310	R0 := R0 − R1	$c := N - a * 100 - b * 10$
001C	05D0	R0 * 100	
001E	0064		
0020	0203	R3 := R3 + R0	$M := M + c * 100$
0022	0F00	Стоп	

При разработке программы были использованы регистровая адресация и адресация по PC.

Пример 4. Составить программу вычисления значения выражения

$$R = \frac{a}{bc - d^2} - \frac{ab - cd}{(ab)^2}.$$

Идея решения. Поскольку для хранения аргументов, результатов и промежуточных величин регистров недостаточно, то все эти величины поместим в последовательные ячейки памяти, а один из регистров используем для меняющихся адресов величин. Другие регистры можно использовать для некоторых промежуточных результатов.

План решения:

1. Вычислить $l = bc$.
2. Вычислить $m = d^2$.
3. Вычислить $n = l - m$.
4. Вычислить $k = a / n$.
5. Вычислить $p = ab$, запомнить эту величину.
6. Вычислить $q = cd$.
7. Вычислить $r = p - q$.
8. Вычислить $s = p^2$.
9. Вычислить $k_1 = r / s$.
10. Получить результат $R = k - k_1$.

Распределение памяти

R0	R1	R2	R3
Адрес величин	Промежуточный расчет	Промежуточный расчет	R

Адрес	0060	0062	0064	0066	0068	006A
Величина	a	b	c	d	k	k_1

Тест

$a = -8$ (FFF8); $b = -1$ (FFFF); $c = 4$; $d = 4$. Ответ: 1.

Программа

Адрес	Команда	Действие	Замечание
0000	02D0	R0 := R0 + 2	Адрес b
0002	0002		
0004	0141	(R0) → R1	
0006	02D0	R0 := R0 + 2	Адрес c
0008	0002		
000A	0541	R1 := R1 * (R0)	l := bc
000C	02D0	R0 := R0 + 2	Адрес d
000E	0002		
0010	0142	(R0) → R2	
0012	0522	R2 := R2 * R2	$m := d^2$
0014	0321	R1 := R1 − R2	$n := l - m$
0016	03D0	R0 := R0 − 6	Адрес a
0018	0006		
001A	0142	(R0) → R2	
001C	0612	R2 := R2 / R1	$k := a/n$

Адрес	Команда	Действие	Замечание
001E	02D0	R0 := R0 + 8	Адрес k
0020	0008		
0022	0124	R2 → (R0)	
0024	03D0	R0 := R0 − 8	Адрес a
0026	0008		
0028	0141	(R0) → R1	
002A	02D0	R := R0 + 2	Адрес b
002C	0002		
002E	0541	R1 := R1 + (R0)	$p := ab$
0030	02D0	R0 := R0 + 2	Адрес c
0032	0002		
0034	0142	(R0) → R2	
0036	02D0	R0 := R0 + 2	Адрес d
0038	0002		
003A	0542	RR2 := R2 * (R0)	$q := cd$
003C	02D0	R0 := R0 + 4	Адрес k_1
003E	0004		
0040	0114	R1 → (R0)	Запоминаем p
0042	0324	(R0) := (R0) − R2	$r := p - q$
0044	0511	R1 := R1 * R1	$s := p^2$
0046	0614	(R0) := (R0) / R1	$k_1 := r/s$
0048	0143	(R0) → R3	
004A	03D0	R0 := R0 − 2	Адрес k
004C	0002		
004E	0343	R3 := R3 − (R0)	$R := k_1 - k$
0050	05D3	R3 := R3 * (−1)	$R := -R$
0052	FFFE		
0054	0F00	Стоп	

Развилка и цикл

Пример 5. Найдите произведение наибольшего и наименьшего из трех чисел a, b, c.

Согласно условию задачи прежде всего необходимо определить наименьшее и наибольшее из трех чисел, после чего вычислить их произведение.

План решения:

1. Сравнить числа a, b. Если $a < b$, то $min := a$, иначе $min := b$.
2. Сравнить числа min, c. Если $c < min$, то $min := c$.
3. Сравнить числа a, b. Если $a > b$, то $max := a$, иначе $max := b$.
4. Сравнить числа max, c. Если $c > max$, то $max := c$.
5. Вычислить $p := min * max$.
6. Стоп.

Распределение памяти

R0	R1	R2	R3
	min	max	p

Адрес	0060	0062	0064
Величина	*a*	*b*	*c*

Тесты

1) $a = -8$ (FFF8); $b = -1$ (FFFF); $c = 4$ (0004). Ответ: -32 (FFE0).
2) $a = 5$ (0005); $b = 10$ (000A); $c = 2$(0002). Ответ: 20 (0014).

Программа

Адрес	Команда	Действие	Замечания
0000	0141	(R0) → R1	min := *a*
0002	02D0	R0 := R0 + 2	Адрес *b*
0004	0002		
0006	0441	Сравнить (R0) с R1	R1 − (R0)
0008	6D02	Если <= 0, переход на 1 слово (2 байта)	
000A	0141	(R0) → R1	min := *b*
000C	02D0	R0 := R0 + 2	Адрес *c*
000E	0002		
0010	0441	Сравнить (R0) с R1	R1 − (R0)
0012	6D02	Если <= 0, переход на 1 слово (2 байта)	
0014	0141	(R0) → R1	min := *c*
0016	03D0	R0 := R0 − 4	Адрес *a*
0018	0004		
001A	0142	(R0) → R2	max := *a*
001C	02D0	R0 := R0 + 2	Адрес *b*
001E	0002		
0020	0442	Сравнить (R0) с R2	R2 − (R0)
0022	2D02	Если >= 0, переход на 1 слово (2 байта)	
0024	0142	(R0) → R2	max := *b*
0026	02D0	R0 := R0 + 2	Адрес *c*
0028	0002		
002A	0442	Сравнить (R0) с R2	R2 − (R0)
002C	2D02	Если >= 0, переход на 1 слово (2 байта)	
002E	0142	(R0) → R2	max := *c*
0030	0113	R1 → R3	*p* := min
0032	0523	R3 := R3 * R2	*p* := *p* * max
0034	0F00	Стоп	

В том случае, когда некоторая последовательность действий повторяется несколько раз подряд, используется **цикл**. Действия в цикле выполняются до тех пор, пока истинно некоторое условие (значение логического выражения равно ИСТИНА). Как только значение истинности меняется на противоположное, исполнение цикла прекращается.

В «Е97» цикл можно организовать, если использовать команды сравнения, условного и безусловного перехода.

Пример 6. Найдите сумму первых n нечетных натуральных чисел, которые кратны 3.

Идея решения. Указанные в условии задачи числа образуют последовательность 3, 9, 15, ..., $6n-3$. Для получения суммы можно использовать цикл; очередное слагаемое будет на 6 больше предыдущего.

План решения:

1. $i := 1$.
2. $S := 0$.
3. $k := 3$.
4. Сравнить i с n.
5. Если $i > n$, перейти к п.10.
6. $S := S + k$.
7. $k := k + 6$.
8. $i := i + 1$.
9. Перейти к п. 4.
10. Стоп.

Распределение памяти

R0	R1	R2	R3
n	i	k	S

Тест. $n = 10$, $S = 300$.

Программа

Адрес	Команда	Действие	Замечания
0000	01D1	1 → R1	$i := 1$
0002	0001		
0004	01D3	0 → R3	$S := 0$
0006	0000		
0008	01D2	3 → R2	$k := 3$
000A	0003		
000C	0410	Сравнить R1 с R0	R0 − R1
000E	3D0C	Если $i > n$, переход на стоп	
0010	0223	R3 := R3 + R2	$S := S + k$
0012	02D2	R2 := R2 + 6	$k := k + 6$
0014	0006		
0016	02D1	R1 := R1 + 1	$i := i + 1$
0018	0001		
001A	1DF0	Переход на сравнение i с n	
001C	0F00	Стоп	

Расчет переходов:

1) Переход в случае $i > n$ на конец программы. При выполнении этой команды счетчик адреса команд (согласно алгоритму работы процессора) имеет значение

0010. Попасть необходимо на команду с адресом 001C. Поэтому смещение будет таким: 001C − 0010 = 0C.

2) Безусловный переход (возврат) на сравнение i с n. Адрес должен смениться с 001C на 000C. Имеем 000C−001C=−10. Это смещение необходимо записать в дополнительном коде. Имеем: прямой код $10_{(16)}=00010000_{(2)}$; дополнительный код $11101111_{(2)} + 1_{(2)} = 11110000_{(2)} = F0_{(16)}$.

Рассмотренная задача может быть решена и без использования цикла. В самом деле, полученная последовательность является арифметической прогрессией с разностью d=6. По формуле суммы первых n членов имеем

$$S_n = \frac{2a_1 + d(n-1)}{2} n = \frac{2 \cdot 3 + 6(n-1)}{2} n = 3n^2.$$

Последней формулой и надо воспользоваться для расчетов.

Вернемся к примеру 2 из предыдущей лабораторной работы. Как видно из представленного решения, задача более рационально должна быть решена с использованием цикла.

Пример 7. В последовательных ячейках памяти расположены пять целых чисел. Получите произведение этих чисел. Содержимое памяти не изменяйте.

Распределение памяти

R0	R1	R2	R3
Адрес очередного числа	i	—	P

Заданные числа будем хранить с адреса 0050.

Тест

Адрес	0050	0052	0054	0056	0058
Величина	0002	FFFE	000A	FFFF	0003

Ответ: $120_{(10)} = 78_{(16)}$.

Программа

Адрес	Команда	Действие	Замечания
0000	01D3	1 → R3	P:= 1
0002	0001		
0004	01D1	1 → R1	i:= 1
0006	0001		
0008	04D1	Сравнить i с 5	R1 − 5
000A	0005		
000C	7D0C	Если i > 5, переход на Стоп	
000E	0543	R3 := R3 * (R0)	$P := P * a_i$
0010	02D1	R1 := R1 + 1	$i := i + 1$
0012	0001		
0014	02D0	R0 := R0 + 2	Адрес следующего числа
0016	0002		
0018	1DEE	Переход на сравнение i с 5	
001A	0F00	Стоп	

Расчет переходов:
1) 1A − 0E = 0C; 2) 08 − 1A = −12 (EE).

Массивы

Пример 8. Поменяйте местами первый отрицательный элемент массива и его максимальный элемент. *Примечание.* В массиве есть хотя бы один отрицательный элемент.

Идея решения. Просматривая массив, необходимо запомнить адреса первого отрицательного элемента и максимального элемента. По окончанию просмотра совершить обмен.

План решения:

1. $i := 2$.
2. max := 1.
3. otr := 0.
4. Если $a[1] < 0$, то $k := 1$, otr := 1.
5. Сравнить i с n.
6. Если $i > n$, перейти к п. 11.
7. Если $a[i]<0$ и otr = 0, то $k := i$, otr := 1.
8. Если $a[i] > a$[max], то max := i.
9. $i := i + 1$.
10. Перейти к п. 5.
11. vsp := $a[k]$.
12. $a[k] := a$[max].
13. a[max] := vsp.
14. Стоп.

Распределение памяти

R0	R1	R2	R3
Адреса элементов массива	Адреса i, otr	Адреса k, n	max

Адрес	00A0	00A2	00A4	00A6
Величина	i	otr	k	n

Тест

Массив разместим с адреса 0080. Пусть $n = 8$.

Исходный массив		Преобразованный массив	
Адрес	Значение	Адрес	Значение
0080	0005	0080	0005
0082	0006	0082	0006
0084	FFFE	0084	0100
0086	0066	0086	0066
0088	0012	0088	0012
008A	0100	008A	FFFE
008C	FFFF	008C	FFFF
008E	0020	008E	0020

Программа

Адрес	Команда	Действие	Замечания
0000	01D5	2 → (R1)	$i := 2$
0002	0002		

Адрес	Команда	Действие	Замечания
0004	0103	R0 → R3	Адрес максимального элемента
0006	02D1	R1 := R1 + 2	Адрес otr
0008	0002		
000A	01D5	0 → (R1)	otr := 0
000C	0000		
000E	04D4	Сравнить *a*[1] с 0	(R0) − 0
0010	0000		
0012	2D06	Если *a*[1] >= 0, переход на $6_{(16)}$ байт	
0014	0106	R0 → (R2)	Адрес 1-го отрицательного элемента
0016	01D5	1 → (R1)	otr := 1
0018	0001		
001A	02D2	R2 := R2 + 2	Адрес *n*
001C	0002		
001E	03D1	R1 := R1 − 2	Адрес *i*
0020	0002		
0022	02D0	R0 := R0 + 2	Адрес 2-го элемента массива
0024	0002		
0026	0456	Сравнить (R1) с (R2) (*i* с *n*)	(R2) − (R1)
0028	3D32	Если < 0, переход на $32_{(16)}$ байта	Переход на обмен значений
002A	02D1	R1 := R1 + 2	Адрес otr
002C	0002		
002E	04D4	Сравнить (R0) с 0	(R0) − 0
0030	0000		
0032	2D14	Если >= 0, переход на $14_{(16)}$ байт	
0034	04D5	Сравнить (R1) с 0	(R1) − 0
0036	0000		
0038	4D0E	Если <> 0, переход на $E_{(16)}$ байт	
003A	03D2	R2 := R2 − 2	Адрес *k*
003C	0002		
003E	0106	R0 → (R2)	Адрес 1-го отрицательного элемента
0040	01D5	1 → (R1)	otr := 1
0042	0001		

Адрес	Команда	Действие	Замечания
0044	02D2	R2 := R2 + 2	Адрес *n*
0046	0002		
0048	0447	сравнить (R0) с (R3) ($a[i]$ с a[max])	(R3) − (R0)
004A	2D02	Если >= 0, переход на $2_{(16)}$ байта	Если a[max] >= $a[i]$
004C	0103	R0 → R3	Адрес максимального элемента
004E	03D1	R1 := R1 − 2	Адрес *i*
0050	0002		
0052	02D5	(R1) := (R1) + 1	$i := i + 1$
0054	0001		
0056	02D0	R0 := R0 + 2	Адрес *i*-го элемента
0058	0002		
005A	1DCA	Переход на $-36_{(16)}$ байт	На сравнение *i* с *n*
005C	03D2	R2 := R2 − 2	Адрес *k*
005E	0002		
0060	0162	(R2) → R2	
0062	0161	(R2) → R1	vsp := $a[k]$
0064	0176	(R3) → (R2)	$a[k] := a$[max]
0066	0117	R1 → (R3)	a[max] := $a[k]$
0068	0F00	Стоп	

Подпрограммы

Пример 9. На отрезке [*m*; *n*] вычислите сумму тех четных чисел, в десятичной записи которых три четные цифры.

В задаче можно выделить такие подзадачи:

1) определение четности-нечетности очередного числа;

2) подсчет количества четных цифр в десятичной записи очередного числа.

Каждую из этих подзадач решим с помощью подпрограммы.

1) Определить, является ли число четным.

Число *a* будет четным, если остаток от деления на 2 этого числа равен нулю, и нечетным в противном случае.

План решения:

1. $b := a$ div 2.
2. $b := b * 2$.
3. $a := a - b$.
4. Возврат из п/п.

Распределение памяти

R0	R1	R2	R3
–	–	Адрес *a*	Адрес *b*

Подпрограмма

Адрес	Команда	Действие	Замечания
0070	0167	(R3) := (R2)	$b := a$
0072	06D7	(R3) := (R3) / 2	$b := a$ div 2
0074	0002		
0076	05D7	(R3) := (R3) * 2	$b := b * 2$
0078	0002		
007A	0376	(R6) := (R6) − (R7)	$a := a - b$
007C	0D00	Возврат из п/п	

2) Подсчет количества четных цифр в десятичной записи числа.

Отделяем очередную цифру числа z (остаток от деления на 10) и проверяем ее на четность. Если четная, учитываем это. Число z делим на 10 и повторяем процедуру. Действия продолжаем до тех пор, пока последнее частное не станет равным нулю.

План решения:

1. $s := 0$.
2. Сравнить z с 0.
3. Если z=0, то к п.10.
4. $k := z$ div 10.
5. $k := k * 10$.
6. $m := z - k$.
7. Если m четное, то $s := s + 1$.
8. $z := z$ div 10.
9. Переход к п.2.
10. Возврат из п/п.

Распределение памяти

R0	R1	R2	R3
–	Адрес s, k	Адрес $a = m$	Адрес b, z

Подпрограмма

Адрес	Команда	Действие	Замечания
0080	01D5	0 → (R1)	$s := 0$
0082	0000		
0084	02D1	R1 := R1 + 2	Адрес k
0086	0002		
0088	04D7	Сравнить (R3) с 0	Сравнить z с 0
008A	0000		
008C	5D32	Если = 0, к возврату из п/п	
008E	0175	(R3) → (R1)	$k := z$
0090	06D5	(R1) := (R1) / 10	$k := k / 10$
0092	000A		
0094	05D5	(R1) := (R1) * 10	$k := k * 10$
0096	000A		
0098	0176	(R3) → (R2)	$m := z$
009A	0356	(R2) := (R2) − (R1)	$m := m - k$

Адрес	Команда	Действие	Замечания
009C	03D3	R3 := R3 − 2	Адрес *b*
009E	0002		
00A0	9C0D	Переход на п/п определения четности	
00A2	0070		
00A4	04D6	Сравнить (R2) с 0	
00A6	0000		
00A8	4D0C	Если <> 0, переход на С байт	
00AA	03D1	R1 := R1 − 2	Адрес *s*
00AC	0002		
00AE	02D5	(R1) := (R1) + 1	*s* := *s* + 1
00B0	0001		
00B2	02D1	R1 := R1 + 2	Адрес *k*
00B4	0002		
00B6	02D3	R3 := R3 + 2	Адрес *z*
00B8	0002		
00BA	06D7	(R3) := (R3) / 10	*z* := *z* / 10
00BC	000A		
00BE	1DC8	Переход на сравнение *z* с 0	
00C0	03D1	R1 := R1 − 2	Адрес *s*
00C2	0002		
00C4	03D3	R3 := R3 − 2	Адрес, предшествующий *z*
00C6	0002		
00C8	00C8	Возврат из п/п	

Приступим к составлению основной программы. Для решения задачи достаточно просмотреть все числа от *m* до *n* включительно. Проверяем очередное число, будет ли оно четным; если «да», то вычисляем количество четных цифр в его записи; если их будет 3 — добавляем число к сумме.

План решения:

1. *i* := *m*.
2. Sum := 0.
3. Сравнить *i* с *n*.
4. Если *i* > *n*, переход к п. 11.
5. Проверить четность числа *i*, если нечетное, перейти к п. 9.
6. Вычислить *R* — количество четных цифр в записи числа *i*.
7. Сравнить *R* с 3. Если *R* < 3, перейти к п. 9.
8. Sum := Sum + 1.
9. *i* := *i* + 1.
10. Перейти к п. 3.
11. Стоп.

Распределение памяти

R0	R1	R2	R3
Адрес *m*, Sum, *n*	Адрес *i*, *R*, в п/п	В п/п	В п/п

Адрес	0050	0052	0054	0056	0058	005A	005C	005E	0060
Переменная	*m*	Sum	*N*	*i*	*R*, *s*	*k*	*a*, *m*	*z*	*b*

Тесты:

1. $m = 400_{(10)} = 190_{(16)}$, $n = 420_{(10)} = 1A4_{(16)}$; $\text{Sum} = 2440_{(10)} = 988_{(16)}$.
2. $m = 1000_{(10)} = 3E8_{(16)}$, $n = 1030_{(10)} = 406_{(16)}$; $\text{Sum} = 10140_{(10)} = 279C_{(16)}$.

Программа

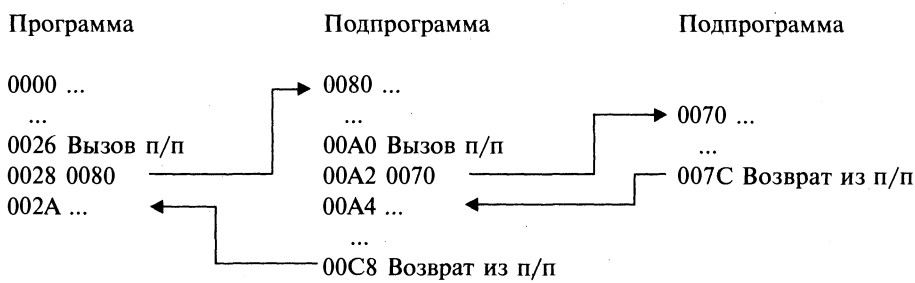
Адрес	Команда	Действие	Замечания
0000	0145	(R0) → (R1)	*i* := *m*
0002	02D0	R0 := R0 + 2	Адрес Sum
0004	0002		
0006	01D4	0 → (R0)	Sum := 0
0008	0000		
000A	02D0	R0 := R0 + 2	Адрес *n*
000C	0002		
000E	0454	Сравнить (R1) с (R0) (*i* с *n*)	(R0) − (R1)
0010	3D38	Если > 0, переход на 38 байт	На стоп
0012	0156	(R1) → (R2)	*a* := *i*
0014	9C0D	Переход на п/п	
0016	0070	Определения четности	
0018	04D6	Сравнить (R2) с 0	Сравнить *a* с 0
001A	4D28	Если <> 0, переход	На изменение *i*
001C	02D3	R3 := R3 + 2	Адрес *z*
001E	0002		
0020	0157	(R1) → (R3)	*z* := *i*
0022	02D1	R1 := R1 + 2	Адрес *R*
0024	0002		
0026	9C0D	Переход на п/п	Сколько четных цифр в числе
0028	0080		
002A	04D5	Сравнить (R1) с 3 (*R* с 3)	(R1) − 3
002C	0003		
002E	4D10	Если <> 0, переход на 10 байт	На увеличение *i*
0030	03D0	R0 := R0 − 2	Адрес Sum
0032	0002		

Адрес	Команда	Действие	Замечания
0034	03D1	R1 := R1 − 2	Адрес *i*
0036	0002		
0038	0254	(R0) := (R0) + (R1)	SUM := SUM + *I*
003A	02D0	R0 := R0 + 2	Адрес *n*
003C	0002		
003E	1D04	Переход на 4 байта	Адрес *i* уже вычислен
0040	03D1	R1 := R1 − 2	Адрес *i*
0042	0002		
0044	02D5	(R1) := (R1) + 1	*i* := *i* + 1
0046	0001		
0048	1DC4	Переход к сравнению *i* с *n*	
004A	0F00	Стоп	

Примечание. Расчет переходов предлагаем читателю проделать самостоятельно.

На примере данной задачи рассмотрим более подробно механизм обращения к подпрограмме. По схеме можно проследить, как осуществляются переход к вспомогательному алгоритму и возврат в вызывающую программную единицу.

Программа



Рассмотрим, как изменяется содержание регистра — указателя стека **SP** и памяти по тем адресам, на которые указывает **SP**, во время исполнения программы. Начальный адрес в **SP** — $00FE_{(16)}$.

До вызова первой п/п	После вызова первой п/п	После вызова второй п/п	После возврата из второй п/п	После возврата из первой п/п
SP: 00FE	SP: 00FC 00FC: 002A	SP: 00FA 00FA: 00A4 00FC: 002A	SP: 00FC 00FC: 002A	SP: 00FE

Следует заметить, что регистр **SP** может быть использован не только при работе с подпрограммами, но и для промежуточного хранения величин.

Заметим также, что подпрограммы могут обращаться и сами к себе — прямо или косвенно (через другие подпрограммы). Подпрограмма, вызывающая себя, называется *рекурсивной.*

Пример 10. Продемонстрируем это на классическом примере рекурсивного алгоритма — вычислении факториала натурального числа. С одной стороны, $n!$ определяется как произведение последовательных натуральных чисел от 1 до n включительно.

С другой стороны, $n! = \begin{cases} 1, & \text{если } n = 0, \\ n \cdot (n-1)!, & \text{если } n > 0. \end{cases}$

Это и есть рекурсивное определение факториала, которым мы воспользуемся.

План решения:

1. Сравнить n с 0.
2. Если n=0, переход к п. 7.
3. Запомнить n в стеке; $n := n - 1$.
4. Вызов п/п вычисления факториала.
5. $F := F * n$.
6. Переход к п. 1.
7. $F := 1$.
8. Возврат из п/п.

Распределение памяти

R0	R1	R2	R3
n	F	–	–

Подпрограмма

Адрес	Команда	Действие	Замечания
0070	2400	Сравнить R0 с 0	$n - 0$
0072	5D0E	Если равно 0, переход	Переход на $F := 1$
0074	0E20	$n \rightarrow$ стек	
0076	2310	R0 := R0 − 1	$n := n - 1$
0078	9C0D	Переход к п/п	
007A	0010	Вычисления факториала	
007C	0E30	Стек → R0	$n \rightarrow$ R0
001E	0501	R1 := R1 * R0	$F := F * n$
0020	1D02	Переход к возврату из п/п	
0022	2111	1 → R1	$F := 1$
0024	0D00	Возврат из п/п	

Распределение памяти

R0	R1	R2	R3
n	F	–	–

Основная программа

Адрес	Команда	Действие	Замечания
0000	0E6D	Установка указателя	
0002	00FE	стека SP на адрес 00FE	
0004	9C0D	Вызов п/п	
0006	0070	Вычисления факториала	
0008	0F00	Стоп	

Тест. $N = 7$; $F = 5040_{(10)} = 13B0_{(16)}$.

Вернемся к примеру 8. Решим задачу с использованием того же алгоритма и с тем же распределением памяти, только вместо адресации по **PC** по возможности будем использовать действие с короткой константой.

Пример 11. Поменять местами первый отрицательный элемент массива и его максимальный элемент.

Программа

Адрес	Команда	Действие	Замечания
0000	2125	2 → (R1)	*i* := 2
0002	0103	R0 → R3	Адрес максимального элемента
0004	2221	R1 := R1 + 2	Адрес otr
0006	2105	0 → (R1)	otr := 0
0008	2404	Сравнить *a*[1] с 0	(R0) − 0
000A	2D04	Если *a*[1] >= 0, переход на 4 байта	
000C	0106	R0 → (R2)	Адрес 1-го отрицательного элемента
000E	2115	1 → (R1)	otr := 1
0010	2222	R2 := R2 + 2	Адрес *N*
0012	2321	R1 := R1 − 2	Адрес *i*
0014	2220	R0 := R0 + 2	Адрес 2-го элемента массива
0016	0456	Сравнить (R1) с (R2) (*i* с *n*)	(R2) − (R1)
0018	3D20	Если < 0, переход на 20 байт	Переход на обмен значений
001A	2221	R1 := R1 + 2	Адрес otr
001C	2404	Сравнить (R0) с 0	(R0) − 0
001E	2D0C	Если >= 0, переход на *C* байт	
0020	2405	Сравнить (R1) с 0	(R1) − 0
0022	4D08	Если <>0, переход на 8 байт	
0024	2322	R2 := R2 − 2	Адрес *k*
0026	0106	R0 → (R2)	Адрес 1-го отрицательного элемента
0028	2115	1 → (R1)	OTR := 1
002A	2222	R2 := R2 + 2	Адрес *n*
002C	0447	Сравнить (R0) с (R3) (*a*[*i*] с *a*[max])	(R3) − (R0)
002E	2D02	Если >= 0, переход на 2 байта	Если *a*[max] >= *a*[*i*]
0030	0103	R0 → R3	Адрес максимального элемента
0032	2321	R1 := R1 − 2	Адрес i
0034	2215	(R1) := (R1) + 1	*i* := *i* + 1
0036	2220	R0 := R0 + 2	Адрес *i*-го элемента

Адрес	Команда	Действие	Замечания
0038	1DDC	Переход на $-24_{(16)}$ байт	На сравнение i с n
003A	2322	R2 := R2 − 2	Адрес k
003C	0162	(R2) → R2	
003E	0161	(R2) → R1	vsp := $a[k]$
0040	0176	(R3) → (R2)	$a[k]$:= a[max]
0042	0117	R1 → (R3)	a[max] := $a[k]$
0044	0F00	Стоп	

Как видно, в этом случае программа значительно сократилась.

Автор «Е97» реализовал некоторый набор полезных подпрограмм. Эти подпрограммы размещаются в ПЗУ, которое начинается с адреса $4000_{(16)}$. Среди этих программ особо стоит выделить те, которые позволяют работать с клавиатурой и дисплеем, т.е. организовывать ввод-вывод данных.

Перечислим их:

1. Вывод целого числа (подпрограмма WriteInteger). Адрес начала — 4068. В R1 — выводимое число, в R3 — адрес области памяти, свободной от программы и данных (организация буфера — для хранения промежуточных значений). Содержимое регистров R0−R3 сохраняется.

2. Вывод строки на дисплей (подпрограмма WriteString). Адрес начала — 4078. В R2 — число символов в выводимой строке, в R3 — адрес начала строки. Содержимое регистров R0, R1 сохраняется, а R2, R3 — нет.

3. Вывод символа на дисплей (подпрограмма OutSym). Адрес начала — 4088. В R0 — выводимый символ. Содержимое R0−R3 сохраняется.

4. Ввод символа с эхо-печатью (подпрограмма InSymE). Адрес начала — 40FA. В R0 — вводимый символ, содержимое регистров R1−R3 сохраняется.

5. Ввод целого числа (подпрограмма Input_Integer). Адрес начала — 4108. В R1 — введенное число, все другие регистры сохраняются.

6. Вывод логического значения (подпрограмма WriteBoolean). Адрес начала — 40C4. При R1 = 0 выводится FALSE, иначе — TRUE. В R1 — значение. R0, R1 сохраняются, а R2, R3 — нет.

7. Вывод текста, находящегося после вызова подпрограммы (п/п WritePasString). Адрес начала — 40DC. R0, R1 — сохраняются, а R2, R3 — нет.

8. Вывод целого числа (второй вариант подпрограммы, описание отсутствует в авторском изложении «Е97» — п/п NewWriteInteger). Адрес начала — 4152. В R1 — выводимое число; R0 = 0 — выравнивание выводимого числа по левому краю, во всех других случаях — по правому. Содержимое регистров R0−R3 сохраняется.

Рассмотрим пример, где организуется ввод данных с клавиатуры и результат выводится на экран.

Пример 12. Вычислить значение выражения $u = x^3y - y^2z + 12xyz$, организовав ввод данных с клавиатуры и вывод результатов на экран.

При решении необходимо позаботиться о распределении памяти с учетом того, что некоторые регистры, а также часть ОЗУ используется подпрограммами ввода-вывода.

План решения:

1. Ввод x, y, x.
2. $P := x^2$.
3. $R := 12z$.
4. $R := P + R$.
5. $R := R \cdot x$.
6. $S := yz$.

7. $R := R - S$.
8. $U := y \cdot R$.
9. Вывод U.
10. Стоп.

Распределение памяти. Значения x, y, z после ввода разместим в памяти по последовательным адресам $80 - 84_{(16)}$. Регистры R1, R2 будем использовать для хранения величин P, R, S. Значение U получим в R1. Кроме того, некоторые регистры будут использованы стандартными подпрограммами (см. выше). Указатель стека установим для удобства на последний адрес ОЗУ — $FE_{(16)}$.

Тесты:

1) $x = 1$, $y = 2$, $z = 3$; $u = 62$; 2) $x = -1$, $y = -2$, $z = -3$; $u = -58$.

Программа

Адрес	Команда	Действие	Замечания
0000	0E6D	FE → SP	Установка SP
0002	00FE		
0004	01D0	80 → R0	Адрес x
0006	0080		
0008	9C0D	Вызов п/п	
000A	40DC	вывода текста	
000C	7802	Текст "x?"	2 — длина текста
000E	003F		
0010	9C0D	Вызов п/п	
0012	4108	ввода целого числа	
0014	0114	R1 → (R0)	Сохраняем x
0016	2220	R0 := R0 + 2	Адрес y
0018	9C0D	Вызов п/п	
001A	40DC	вывода текста	
001C	7902	Текст "y?"	2 — длина текста
001E	003F		
0020	9C0D	Вызов п/п	
0022	4108	ввода целого числа	
0024	0114	R1 → (R0)	Сохраняем y
0026	2220	R0 := R0 + 2	Адрес z
0028	9C0D	Вызов п/п	
002A	40DC	вывода текста	
002C	7A02	Текст "z?"	2 — длина текста
002E	003F		
0030	9C0D	Вызов п/п	
0032	4108	ввода целого числа	
0034	0114	R1 → (R0)	Сохраняем z
0036	3240	R0 := R0 − 4	Адрес x
0038	0142	(R0) → R2	x → R2
003A	0522	R2 := R2 * R2	$P := x^2$
003C	2240	R0 := R0 + 4	Адрес z

Адрес	Команда	Действие	Замечания
003E	0141	(R0) → R1	$z \rightarrow$ R1
0040	25C1	R1 := R1 * 12	$R := 12z$
0042	0221	R1 := R1 + R2	$R := R + P$
0044	3240	R0 := R0 + (−4)	Адрес x
0046	0541	R1 := R1 * (R0)	$R := Rx$
0048	2220	R0 := R0 + 2	Адрес y
004A	0142	(R0) → R2	
004C	2220	R0 := R0 + 2	Адрес z
004E	0542	R2 := R2 * (R0)	$S := yz$
0050	0321	R1 := R1 − R2	$R := R - S$
0052	3220	R0 := R0 + (−2)	Адрес y
0054	0541	R1 := R1 * (R0)	$U := Ry$
0056	9C0D	Вызов п/п	
0058	40DC	вывода текста	
005A	7502	Текст "u="	2 — длина текста
005C	003D		
005E	01D3	В R3 помещаем	Используется в п/п
0060	0090	адрес 90	
0062	9C0D	п/п вывода	
0064	4068	целого числа	
0066	0F00	Стоп	

В «E97» используются команды для работы не только с машинными словами, но и отдельными байтами. Рассмотрим некоторые из них на примере программы обработки текста.

Пример 13. Составьте программу, которая определяет число заглавных русских букв в заданном тексте и выводит результат на экран.

Пусть текст находится в ОЗУ. Будем считать, что текст закончился, если встретится код $00_{(16)}$. Обратим внимание на то, что диапазон кодов заглавных русских букв $80_{(16)} - 9F_{(16)}$.

План решения:

1. $S := 0$.
2. Сравнить код символа с $00_{(16)}$.
3. Если равен, переход к п. 7.
4. Если код символа больше $7F_{(16)}$, но меньше $A0_{(16)}$, то $S := S + 1$.
5. Перейти к следующему символу.
6. Переход к п. 2.
7. Вывод S.
8. Стоп.

Распределение памяти

R0	*R1*	*R2*	*R3*
Адрес символа	S	—	—

Тест. Текст: НА ШАХМАТНОЙ доске 64 КЛЕТКИ. $S = 17$.

Программа

Адрес	Команда	Действие	Замечания
0000	2101	0 → R1	$S := 0$
0002	C44D	Сравнить (R0)b с (0)b	Не встретили ли
0004	0000		символ с кодом 0
0006	5D12	Если = 0, переход на $12_{(16)}$ байт	
0008	C44D	Сравнить (R0)b с (7F)b	7F − (R0)
000A	007F		
000C	2D08	Если >= 0, переход на $8_{(16)}$ байт	
000E	C44D	Сравнить (R0)b с (A0)b	A0 − (R0)
0010	00A0		
0012	6D02	Если <= 0, переход на $2_{(16)}$ байта	
0014	2211	R1 := R1 + 1	$S := S + 1$
0016	2210	R0 := R0 + 1	К следующему символу
0018	1DE8	Переход к адресу 0002	
001A	0111		
001C	9C0D	Вызов п/п	
001E	40DC	вывода последующего текста	
0020	AE06	Текст из 6 символов	
0022	A2E2	«Ответ:»	
0024	E2A5		
0026	003A		
0028	01D3	В R3 помещаем	Используется в п/п
002A	0090	адрес 90	
002C	9C0D	п/п вывода	
002E	4068	целого числа	
0030	0F00	Стоп	

Следующий пример иллюстрирует работу с логическими выражениями, командами «E97», выполняющими логические операции.

Пример 14. Построить таблицу истинности для логического выражения (Aor B) and C. Результаты вывести на экран.

Для решения задачи организуем цикл по каждой из переменных A, B, C. Результат обозначим D. Для дальнейших действий закодируем значение ИСТИНА (TRUE) с помощью числа −1 (FFFF), ЛОЖЬ (FALSE) — с помощью числа 0 (0000) (такой способ кодирования применяется, например, в языке BASIC). Для простоты будем хранить логическое значение не с помощью одного байта, как это реализовано в тех или иных языках программирования, а займем под него машинное слово. Получив очередное значение D, будем печатать A, B, C, D.

План решения:

1. $A := -1$.
2. Сравнить A с 0.
3. Если $A > 0$, переход к п. 19.
4. $B := -1$.
5. Сравнить B с 0.
6. Если $B > 0$, переход к п. 17.
7. $K := A$ or B.
8. $C := -1$.
9. Сравнить C с 0.
10. Если $C > 0$, переход к п. 15.
11. $D := K$ and C.
12. Вывод A, B, C, D.
13. $C := C + 1$.
14. Переход к п. 9.
15. $B := B + 1$.
16. Переход к п. 4.
17. $A := A + 1$.
18. Переход к п. 2.
19. Стоп.

Распределение памяти. В регистр R0 поместим адрес переменных A, B, C (перед запуском программы зададим там адрес величины A). В регистре R1 — величина D; R2 — K. Кроме того, регистры используются стандартными подпрограммами; перед переходом на подпрограммы будем сохранять содержимое регистров в стеке, восстанавливая его впоследствии.

Результатом работы программы будет следующая таблица:

A	B	C	D
TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	FALSE	FALSE
TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE
FALSE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE

Программа

Адрес	Команда	Действие	Замечания
0000	0E6D	FE → SP	Установка SP
0002	00FE		
0004	01D4	−1 → (R0)	$A := -1$
0006	FFFF		
0008	2404	Сравнить (R0) с 0	$(A - 0)$
000A	7D56	Если > 0, переход на стоп	
000C	2220	R0 + 2 → R0	Адрес B
000E	01D4	−1 → (R0)	$B := -1$
0010	FFFF		
0012	2404	Сравнить (R0) с 0	$(B - 0)$
0014	7D46	Если > 0, переход на $A := A + 1$	
0016	0142	(R0) → R2	B → R2

Адрес	Команда	Действие	Замечания
0018	3220	R0 − 2 → R0	Адрес *A*
001A	0842	(R0) or R2 → R2	*K* := *A* or *B*
001C	2240	R0 + 4 → R0	Адрес *C*
001E	01D4	−1 → (R0)	*C* := −1
0020	FFFF		
0022	2404	Сравнить (R0) с 0	(*C* − 0)
0024	7D30	Если > 0, переход на *B* := *B*+ + 1	
0026	0141	(R0) → R1	*C* → R2
0028	0721	R2 and R1 → R1	*D* := *K* AND *C*
002A	0E22	R2 → стек	*K* → стек
002C	0E21	R1 → стек	*D* → стек
002E	3240	R0 − 4 → R0	Адрес *A*
0030	0141	(R0) → R1	*A* → R1
0032	9C0D	П/п вывода значения	
0034	0090	логической константы	
0036	2220	R0 + 2 → R0	Адрес *B*
0038	0141	(R0) → R1	*B* → R1
003A	9C0D	П/п вывода значения	
003C	0090	логической константы	
003E	2220	R0 + 2 → R0	Адрес *C*
0040	0141	(R0) → R1	*C* → R1
0042	9C0D	П/п вывода значения	
0044	0090	логической константы	
0046	0E31	Стек → R1	*D* → *R*1
0048	9C0D	П/п вывода значения	
004A	0090	логической константы	
004C	9C0D	П/п перевода строки	
004E	40EC		
0050	0E32	Стек → R2	*K* → R2
0052	2214	(R0) + 1 → (R0)	*C* := *C* + 1
0054	1DCC	Переход к сравнению *C* с 0	
0056	3220	R0 − 2 → R0	Адрес B
0058	2214	(R0) + 1 → (R0)	*B* := *B* + 1
005A	1DB6	Переход к сравнению *B* с 0	
005C	3220	R0 − 2 → R0	Адрес *A*
005E	2214	(R0) + 1 → (R0)	*A* := *A* + 1
0060	1DA6	Переход к сравнению *A* с 0	
0062	0F00	Стоп	

С адреса $0090_{(16)}$ разместим измененный вариант авторской подпрограммы вывода значения логической константы. Изменения связаны с тем, что, во-первых, в рассмотренной программе логическая константа кодируется 2 байтами, во-вторых, для получения таблицы в подпрограмме выводятся в каждом случае 6 символов: TRUE дополняется двумя пробелами, FALSE — одним.

Адрес	Команда	Действие	Замечания
0090	2401	Сравнить R1 с 0	R1 − 0
0092	5D0E	Если 0, переход к выводу FALSE	
0094	9C0D	П/п вывода	
0096	40DC	следующего далее текста	
0098	5406	Число символов — 6;	
009A	5552	Текст TRUE	
009C	2045		
009E	0020		
00A0	1D0C	Переход к возврату из п/п	
00A2	9C0D	П/п вывода	
00A4	40DC	следующего далее текста	
00A6	4606	Число символов — 6;	
00A8	4C41	текст FALSE	
00AA	4553		
00AC	0020		
00AE	0D00	Возврат из п/п	

В некоторых предыдущих программах были использованы одноадресные команды. В этом случае кодом операции является **E**, и поскольку операнд всего один, то место одного из операндов (в двухадресных командах это ОП1) занимает дополнительный код операции. За счет этого появляется 12 одноадресных команд. Например, были использованы команды **E2** (пересылка операнда в стек), **E3** (возвращение операнда из стека), **E6** (помещение операнда в регистр **SP** — установка указателя стека). Полный список одноадресных команд представлен в приложении.

Задачи повышенной сложности

1. Заданы два линейных массива с различным числом элементов и натуральное число k. Объединить их в один массив, включив второй массив между k-м и $(k + 1)$-м элементами первого, не используя дополнительный массив.

2. Даны две последовательности $a_1 \le a_2 \le \ldots \le a_n$ и $b_1 \le b_2 \le \ldots \le b_m$. Образовать из них новую последовательность чисел так, чтобы она тоже была неубывающей. *Примечание.* Дополнительный массив не использовать.

3. *Сортировка обменами.* Дана последовательность чисел $a_1, a_2, \ldots, a_n$. Требуется переставить числа в порядке возрастания. Для этого сравниваются два соседних

числа a_i и a_{i+1}. Если $a_i > a_{i+1}$, то делается перестановка. Так продолжается до тех пор, пока все элементы не расположатся в порядке возрастания. Составить алгоритм сортировки, подсчитывая при этом число перестановок.

4. *Сортировка вставками.* Дана последовательность чисел $a_1, a_2, ..., a_n$. Требуется переставить числа в порядке возрастания. Делается это следующим образом. Пусть $a_1, a_2, ..., a_i$ — упорядоченная последовательность, т.е. $a_1 \le a_2 \le ... \le a_i$. Берется следующее число a_{i+1} и вставляется в последовательность так, чтобы новая последовательность была также возрастающей. Процесс производится до тех пор, пока все элементы от $i + 1$ до n не будут перебраны.

5. *Сортировка Шелла.* Дан массив n целых чисел. Требуется упорядочить его по возрастанию. Делается это следующим образом: сравниваются два соседних элемента a_i и a_{i+1}. Если $a_i \le a_{i+1}$, то продвигаются на один элемент вперед. Если $a_i > a_{i+1}$, то производится перестановка и сдвигаются на один элемент назад. Составить алгоритм этой сортировки.

6. Пусть даны неубывающая последовательность целых чисел $a_1 \le a_2 \le ... \le a_n$ и целые числа $b_1 \le b_2 \le ... \le b_m$. Требуется указать те места, на которые надо вставлять элементы последовательности $b_1 \le b_2 \le ... \le b_m$ в первую последовательность так, чтобы новая последовательность оставалась возрастающей.

7. Даны дроби $\frac{p_1}{q_1}, \frac{p_2}{q_2}, ..., \frac{p_n}{q_n}$ (p_i — целые, q_i — натуральные). Составить программу, которая приводит эти дроби к общему знаменателю и упорядочивает их в порядке возрастания.

8. *Алгоритм фон Неймана.* Упорядочить массив $a_1, a_2, ..., a_n$ по неубыванию с помощью алгоритма сортировки слияниями:

1) каждая пара соседних элементов сливается в одну группу из двух элементов (последняя группа может состоять из одного элемента);
2) каждая пара соседних двухэлементных групп сливается в одну четырехэлементную группу и т.д.

При каждом слиянии новая укрупненная группа упорядочивается.

9. Реализовать алгоритм быстрого поиска в отсортированном массиве. Если заданное число будет найдено, поместить по адресу 76 его адрес, иначе — число (−1).

10. Разработать программу ввода и размещения в памяти двумерного массива. Решить следующие задачи (результат вывести на экран):

а) упорядочить элементы каждой строки по возрастанию;
б) упорядочить элементы каждого столбца по возрастанию;
в) элементы каждой строки с четным номером упорядочить по возрастанию, с нечетным — по убыванию;
г) найти сумму элементов главной диагонали квадратной матрицы;
д) на месте каждого элемента записать частное от его деления на номер столбца, в котором он расположен;
е) на месте каждого элемента записать частное от его деления на минимальный элемент той строки, в которой он расположен;
ж) если сумма индексов максимального элемента массива четная, поместить по адресу 76 число 1, иначе — число −1;
з) каждый элемент массива увеличить на разность его первого и второго индекса;
и) все элементы квадратной матрицы, лежащие выше главной диагонали, умножить на данное число k;
к) все элементы квадратной матрицы, лежащие ниже главной диагонали, заменить на нуль;

л) если выше главной диагонали лежит больше чисел, кратных заданному k, чем ниже главной диагонали, то поместить по адресу 76 число 1, если поровну — 0, если меньше — −1;

м) порядок квадратной матрицы нечетный. Поменять местами среднюю строку и средний столбец.

11. Разработать способ представления в памяти «E97» действительных чисел. Написать следующие подпрограммы:

а) ввод действительного числа;

б) вывод действительного числа;

в) сложение двух действительных чисел;

г) вычитание двух действительных чисел;

д) умножение двух действительных чисел.

12. Разработать способ представления в памяти «E97» четырехбайтовых целых чисел со знаком. Написать следующие подпрограммы:

а) ввод числа; б) вывод числа;

в) сложение двух чисел; г) вычитание двух чисел;

д) умножение двух чисел;

е) целочисленное деление двух чисел;

ж) нахождение остатка от деления одного числа на другое.

13. Разработать способ представления в памяти «E97» величин типа «множество». Написать следующие подпрограммы:

а) объединение множеств;

б) пересечение множеств;

в) разность множеств;

г) вхождение элемента в множество (результат TRUE или FALSE).

Лабораторные работы

Лабораторная работа № 1. Структура процессора и памяти. Способы адресации данных. Система команд

Задания к лабораторной работе

Время выполнения 4 часа.

1. Записать математическую формулировку условия задачи.
2. Составить алгоритм решения задачи (план решения задачи).
3. Распределить память под данные.
4. Составить программу для «E97».
5. Подобрать тестовый набор данных для тестирования программы.
6. Ввести программу в память ЭВМ, сделать отладку, выполняя программу в пошаговом и автоматическом режиме.
7. Составить отчет о проделанной работе.

Варианты заданий

Вариант 1

а) Найдите сумму чисел, находящихся в регистрах с R0 по R2, накапливая ее в регистре R3.

Примечание. Содержимое регистров R0 — R2 не изменяйте.

б) Найдите произведение цифр заданного четырехзначного числа.

Вариант 2

а) Найдите разность суммы чисел, находящихся в регистрах R0 и R1, и числа из регистра R2. Результат — в регистре R3. *Примечание.* Содержимое регистров R0—R2 не изменяйте.

б) Дано целое число x. Не пользуясь никакими другими арифметическими операциями, кроме умножения, сложения и вычитания, вычислите за минимальное число операций $2x^4 - 3x^3 + 4x^2 - 5x + 6$.

Вариант 3

а) Найдите сумму чисел, находящихся в основной памяти с адреса $20_{(16)}$ по $26_{(16)}$. Результат поместите в регистр R3.

б) Дано целое число x. Получите значения $-2x + 3x^2 - 4x^3$ и $1 + 2x + 3x^2 + 4x^3$. Позаботьтесь об экономии операций.

Вариант 4

а) Найдите разность суммы чисел, находящихся в основной памяти с адреса $40_{(16)}$ по $44_{(16)}$, и суммы чисел, находящихся в регистрах R0 и R1. Результат поместите по адресу, который находится в регистре R3.

б) Дано a. Получите a^8 за три операции и a^{10} и a^{12} за четыре операции.

Вариант 5

а) Найдите сумму чисел, расположенных в основной памяти по адресам, которые находятся в регистрах с R0 по R2, накапливая ее в регистре R3.

б) Найдите сумму арифметической прогрессии, если известны ее первый член, разность и число слагаемых.

Вариант 6

а) Найдите разность суммы чисел, адреса которых заданы в регистрах R0 и R1, и числа из регистра R2. Результат — в регистре R3. *Примечание.* Содержимое регистров R0—R2 и основной памяти не изменяйте.

б) Вычислите значение выражения $\dfrac{a+b-c}{ab} - \dfrac{cd}{a-b}$.

Вариант 7

а) Найдите сумму четырех чисел, расположенных в основной памяти последовательно. Адрес первого числа хранится в основной памяти по адресу, заданному в регистре R0.

б) В регистре R1 хранится число единиц, в R2 — число десятков, в R3— число сотен некоторого трехзначного числа. Напишите программу, помещающую это число в память по адресу $54_{(16)}$.

Вариант 8

а) Даны два целых числа x и y. Вычислите их сумму, разность, произведение и частное.

б) Даны двузначные числа m, n. Вычислите отдельно сумму цифр каждого из этих чисел, затем вычислите разность этих сумм.

Вариант 9

а) Сложите два вектора с целочисленными координатами (a_1, b_1) и (a_2, b_2).

б) Составьте программу для вычисления пути, пройденного лодкой, если ее скорость в стоячей воде v км/ч, скорость течения реки v_1 км/ч, время движения по озеру t_1 ч, а против течения реки — t_2 ч.

Вариант 10

а) Найдите разность двух векторов с целочисленными координатами (a_1, b_1) и (a_2, b_2).

б) В заданном трехзначном числе, записанном в десятичной системе счисления, вычеркните среднюю цифру.

Вариант 11

а) Найдите произведение чисел, находящихся в регистрах с R0 по R2, накапливая его в регистре R3. *Примечание.* Содержимое регистров R0—R2 не изменяйте.

б) Дано натуральное число, записанное в десятичной системе счисления, меньшее 100. Допишите в начало и конец этого числа заданную в регистре R1 цифру k.

Вариант 12

а) Сложите два вектора с целочисленными координатами (a_1, b_1, c_1) и (a_2, b_2, c_2).

б) Дано двузначное натуральное число, записанное в десятичной системе счисления. Запишите вместо цифры, находящейся в этом числе в разряде десятков, заданную в регистре R0 цифру k ($k \neq 0$).

Вариант 13

а) Найдите разность двух векторов с целочисленными координатами (a_1, b_1, c_1) и (a_2, b_2, c_2).

б) Вычислите значение выражения $\dfrac{x + yz}{2} - \dfrac{xyz}{x - y}$.

Вариант 14

а) Вычислите периметр и площадь прямоугольного треугольника с катетами a, b и гипотенузой c.

б) Дано четырехзначное натуральное число, записанное в десятичной системе счисления. Найдите произведение сумм двух цифр из старших разрядов и двух цифр из младших разрядов.

Вариант 15

а) Дана длина ребра куба. Найдите площадь грани, площадь полной поверхности и объем этого куба.

б) Даны два двузначных натуральных числа, записанных в десятичной системе счисления. Получите четырехзначное число, дописав первое число ко второму в старшие разряды.

Вариант 16

а) Найдите площадь трапеции с основаниями a и b и высотой h.

б) Вычислите значение выражения $xyz + \dfrac{yz}{2} + \dfrac{10x}{z - y}$.

Вариант 17

а) Дана величина A, выражающая объем информации в байтах. Переведите A в биты и килобайты.

б) Дано четырехзначное натуральное число, записанное в десятичной системе счисления. Получите из него двузначное число, вычеркнув цифры из старшего и младшего разрядов.

Вариант 18

а) Вычислите объем и площадь полной поверхности прямоугольного параллелепипеда со сторонами a, b, c.

б) Дано двузначное натуральное число, записанное в десятичной системе счисления. Получите из него трехзначное число, записав в середину заданную цифру k.

Вариант 19

а) Вычислите площадь и периметр прямоугольника со сторонами a, b.

б) Дано четырехзначное натуральное число, записанное в десятичной системе счисления. Получите из него двузначное число, вычеркнув цифры из разрядов сотен и десятков.

Вариант 20

а) Умножьте число a на вектор $\vec{c}(c_1, c_2, c_3)$.

б) Вычислите значение выражения $x^2 - xyz + 10y^3 - \frac{xz}{3}$.

Вариант 21

а) Вычислите сопротивление R, если известны напряжение U и сила тока I.

б) Даны натуральные числа a и b. Найдите остаток от деления a на b.

Вариант 22

а) Даны a, h — основание и высота треугольника. Вычислите его площадь.

б) Дано однозначное натуральное число. Получите четырехзначное число, повторив цифру заданного числа четырежды.

Вариант 23

а) Найдите разность суммы чисел, находящихся в регистрах R0 и R1, и суммы трех чисел, хранящихся в оперативной памяти (адрес первого из них указан в регистре R2). Результат — в регистре R3. *Примечание.* Содержимое регистров R0—R2 и оперативной памяти не изменяйте.

б) Дано четырехзначное натуральное число, записанное в десятичной системе счисления. Получите из него число, записанное теми же цифрами, но расположенными в обратном порядке.

Вариант 24

а) Найдите разность числа из регистра R0 и суммы чисел, находящихся в основной памяти с адреса $40_{(16)}$ по $48_{(16)}$. Результат поместите в регистр R3.

б) Вычислите значение выражения $a^3 - b^3 + \frac{(c-1)\ c\ (c+1)}{6}$.

Вариант 25

а) В основной памяти с адреса $40_{(16)}$ расположены четыре десятичные цифры. Получите в регистре R0 четырехзначное число, в старшем разряде которого находится цифра, расположенная по адресу $40_{(16)}$, в следующем разряде — по адресу $42_{(16)}$ и т.д.

б) Дано четырехзначное натуральное число, записанное в десятичной системе счисления. Получите из него двузначное число, вычеркнув цифры из разрядов тысяч и десятков.

Лабораторная работа № 2. Развилка и цикл

Задания к лабораторной работе

Разработать программы, реализующие приведенные ниже задания.

Варианты заданий

Время выполнения 4 часа.

Вариант 1

а) Большее из трех натуральных чисел умножьте на 10, среднее по величине — на 50, меньшее — на 100.

б) Числовая последовательность задана рекуррентной формулой $a_n = a_{n-1} + a_{n-2}$. Найдите k-й член последовательности, если $a_0 = 1$ и $a_1 = 2$.

Вариант 2

а) В памяти по некоторому адресу хранится натуральное число. Напишите программу, которая позволяет переменной A присвоить значение 0, если число четное, и 1, если — нечетное.

б) Используя операцию вычитания, напишите программу нахождения частного и остатка от деления одного целого числа на другое.

Вариант 3

а) Результаты вычислений по формулам $y = 8A - 4B$ и $z = |A + 4B|$ запишите в память. Большее из них поместите в регистр R0.

б) На отрезке [1; 10] найдите такое целочисленное значение первого члена арифметической прогрессии, при котором один из ее членов равен c. Разность d ($d \neq 1$) задать самостоятельно. Сколько членов последовательности предшествуют члену со значением c?

Вариант 4

а) Найдите min {max (A, B), max (C, D)}.

б) Даны два натуральных числа. Определите, сколько натуральных чисел расположено между ними, и поместите эти числа в последовательно расположенные ячейки памяти.

Вариант 5

а) Проверьте, попадает ли точка $C(x, y)$ в квадрат $\{a < x < b;\ c < y < d\}$. Если попадает, то ее абсциссу занесите в регистр R0, иначе — в память по адресу $90_{(16)}$.

б) Вычислите 2^n.

Вариант 6

а) Даны три числа. Занесите их в память в порядке возрастания.

б) Найдите сумму всех целых чисел, принадлежащих отрезку $[a, b]$.

Вариант 7

а) Из трех чисел найдите наибольшее и вычтите из него все остальные.

б) В памяти хранятся числа A и B, причем $A < B$. Определите, сколько раз можно к числу A прибавить 4, чтобы результат не превышал B. Из полученной суммы вычтите B, результат запишите в память.

Вариант 8

а) Запишите число a в память по адресу $72_{(16)}$, а b — по адресу $74_{(16)}$. Вычислите $c = |a - 7b|$ и сравните с d. Если $c > d$, то очистите память с адресом $72_{(16)}$, иначе — с адресом $74_{(16)}$.

б) Вычислите $S = \sum_{i=1}^{n} i^2$.

Вариант 9

а) Функция задана формулой

$$f(n) = \begin{cases} 5, & \text{если } 1 \le n < 5, \\ n, & \text{если } 5 \le n < 10, \\ 20 - n, & \text{если } n \ge 10 \end{cases}$$

(n — натуральное число). Составьте программу вычисления значений этой функции.

б) В двух регистрах процессора находятся числа M и N, причем $M < N$. К ним начинают прибавлять соответственно 3 и 1. Через сколько повторений число в первом регистре будет больше, чем во втором?

Вариант 10

а) Заданы длины трех отрезков. Определите, могут ли эти отрезки служить сторонами треугольника. Если могут, то по адресу $76_{(16)}$ занесите 1, иначе — 2.

б) Заданы числа A, B и C ($A < B < C$; $B - A \ge 2$). Сколько раз надо вычесть 5 из C, чтобы результат попал на отрезок $[A; B]$? Предусмотрите случай, когда попадание на отрезок невозможно.

Вариант 11

а) В регистрах R1, R2, R3 находятся числа. Запишите номер регистра, в котором находится наибольшее значение, в память по адресу $84_{(16)}$.

б) Число A умножьте n раз на число B.

Вариант 12

а) Проверьте, удовлетворяют ли заданные числа M и N соотношениям $|M| < 10$ и $|N| > 5$. В случае, если оба соотношения справедливы, то очистите память с адресом $76_{(16)}$, в противном случае запишите туда 1.

б) Какой член числовой последовательности $a_n = 3a_{n-1} - 1$ превысит b, если $a_0 = 1$?

Вариант 13

а) Даны два угла треугольника. Проверьте, будет ли он прямоугольным. Если да, то по адресу $76_{(16)}$ занесите 1, иначе — 0.

б) Найдите сумму четных чисел от 2 до N.

Вариант 14

а) В памяти находятся пять чисел. Найдите наименьшее из них.

б) Даны два натуральных числа. Найдите первое нечетное число, следующее за большим из данных чисел.

Вариант 15

а) Даны четыре числа a_1, a_2, a_3, a_4. Вычислите $\min(a_1a_2,\ a_3 - a_4,\ a_4/a_1)$.

б) Даны две числовые последовательности: $a_n = 2n$ и $b_n = 2b_{n-1}$ ($b_0 = 1$). Определите, сколько членов этих последовательностей совпадают.

Вариант 16

а) Даны три числа a, b, c. Если a, b являются катетами, а c — гипотенузой прямоугольного треугольника, то поместите по адресу $76_{(16)}$ число 1, иначе — 0.

б) Вычислите $\text{НОК}(a, b) = \dfrac{ab}{\text{НОД}(a, b)}$.

Вариант 17

а) Вычислите $F(x) = \begin{cases} 0, & \text{если } x < 0, \\ x^3, & \text{если } 0 \le x \le 10, \\ 9x + 100, & \text{если } x > 10. \end{cases}$

б) Дано натуральное число n. Определите количество цифр в его записи.

Вариант 18

а) В памяти по адресам с $60_{(16)}$ по $76_{(16)}$ хранятся числа. Перепишите содержимое памяти с адресом $60 + 2N + M$, где M и N — произвольные целые числа, в регистр R3. Предусмотрите контроль правильности полученного адреса.

б) Вычислите произведение $P = n! = 1 \cdot 2 \cdot 3 \times \cdots \times n$ ($n \le 7$).

Вариант 19

а) Определите $\min(\max(\min(a_1,\ a_2),\ a_3),\ a_4)$.

б) Вычислите $S = 1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + \cdots + (n - 1)n$ ($n < 20$).

Вариант 20

а) Сумма двух первых цифр заданного четырехзначного числа равна сумме двух его последних цифр. Если «да», то по адресу $76_{(16)}$ занесите 1, иначе — 0.

б) Вычислите $S = 1 + x + x^2 + x^3 + \cdots + x^{n-1}$ ($n < 7$).

Вариант 21

а) Все цифры данного трехзначного числа N различны. Если «да», то по адресу $76_{(16)}$ занесите 1, иначе — 0.

б) Вычислите a^n.

Вариант 22

а) Даны три целых числа. Возведите в квадрат те из них, значения которых неотрицательны, и в четвертую степень—отрицательные.

б) Вычислите $\frac{a}{b} = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \cdots + \frac{n}{n+1}$.

Вариант 23

а) Подсчитайте количество отрицательных чисел среди чисел a, b, c.

б) Вычислите $S = 2 - 4 + 6 - 8 + \cdots + (-1)^n \cdot (2n)$ $(n < 20)$.

Вариант 24

а) В памяти хранятся числа a, b, c. Занесите в память адреса тех из них, которые делятся нацело на число k. Подсчитайте количество таких чисел.

б) Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определите, сколько амеб будет через 3, 6, 9, 12, ..., 24 часа?

Вариант 25

а) Грузовой автомобиль выехал из одного города в другой со скоростью v_1 км/ч. Через t ч в этом же направлении выехал легковой автомобиль со скоростью v_2 км/ч. Составьте программу, определяющую, догонит ли легковой автомобиль грузовой через t_1 ч после своего выезда. Если «да», то по адресу $76_{(16)}$ занесите 1, иначе — 0.

б) Составьте программу, вычисляющую значения по формуле $x^2 + x + 41$ при $0 \le x \le 20$ и записывающую их в память.

Лабораторная работа № 3. Массивы

Задания к лабораторной работе

Время выполнения 4 часа.

Составьте и отладьте программы, реализующие приведенные ниже задания.

Варианты заданий

Вариант 1

Подсчитайте число положительных элементов массива и занесите его в массив вместо первого элемента.

Вариант 2

Замените все элементы массива их модулями.

Вариант 3

Найдите сумму элементов массива S и сравните ее с заданным числом k. Если $S< k$, то все элементы массива с четными индексами поменяйте на нули.

Вариант 4

Адрес первого неотрицательного элемента массива поместите в регистр R0.

Вариант 5

Определите номера всех отрицательных элементов массива и сформируйте из них массив, расположенный сразу после заданного.

Вариант 6

В массиве есть положительные и отрицательные элементы. Найдите число элементов массива, которые больше суммы всех его элементов.

Вариант 7

Замените каждый элемент массива суммой всех последующих.

Вариант 8

Вычислите отдельно сумму отрицательных и положительных элементов массива.

Вариант 9

Замените k-й элемент массива адресом максимального по модулю элемента этого же массива.

Вариант 10

Задан массив A. Сформируйте массив B, элементами которого являются разности соседних элементов массива A. Массив B разместите перед массивом A.

Вариант 11

В массиве есть положительные и отрицательные элементы. Сравните модуль минимального элемента с максимальным. При совпадении занесите единицу в качестве последнего элемента массива, иначе — нуль.

Вариант 12

Исключите из массива все нулевые элементы, сформировав при этом новый массив.

Вариант 13

Из заданного массива сформируйте новый, состоящий только из положительных элементов исходного массива.

Вариант 14

Все отрицательные элементы массива замените нулями, число произведенных замен запишите в регистр R3.

Вариант 15

Произведите циклический сдвиг массива в сторону возрастания адресов. Последний элемент при этом сделайте первым.

Вариант 16

Найдите минимальный элемент массива и поставьте его на первое место, если он положителен, и на последнее, если он отрицателен.

Вариант 17

Определите, сколько раз в массиве встречаются элементы, равные данным числам k и l.

Вариант 18

Постройте массив, элементами которого являются числа n^2, $(n-1)^2$, ..., 1^2.

Вариант 19

Массив заканчивается нулем. Определите число элементов в массиве.

Вариант 20

В массиве есть единственный нулевой элемент. Уплотните массив, удалив нулевой элемент.

Вариант 21

Число положительных элементов в массиве поместите в R1, число отрицательных — в R2, число нулей — в R3.

Вариант 22

Даны два массива из одинакового числа элементов. Произведите обмен минимальными элементами.

Вариант 23

К положительным элементам массива прибавьте единицу, а отрицательные — уменьшите на единицу.

Вариант 24

Заданы два массива с одинаковым числом элементов. Перепишите тот массив, сумма элементов которого больше, в другой.

Вариант 25

Постройте массив, элементы которого вычисляются по формуле $a_n=2^n$ ($n = 0, 1, ..., 14$).

Лабораторная работа № 4. Подпрограммы

Задания к лабораторной работе

Время выполнения 4 часа.

Составьте и отладьте программы, соответствующие приведенным ниже задачам.

Варианты заданий

Вариант 1

Составьте программу, которая вычисляет НОД элементов данного массива натуральных чисел.

Вариант 2

Заполните массив $A(N)$ элементами, значение которых равно остатку от деления индекса этого элемента на 3. Например, для $N = 7$ будет получен массив 1, 2, 0, 1, 2, 0, 1.

Вариант 3

Составьте программу, которая подсчитывает, сколько в заданном массиве элементов таких, что $|A[i]| < t$, где t — данное число. В виде подпрограммы оформите вычисление модуля числа.

Вариант 4

Дан массив, элементами которого являются натуральные числа. Каждый элемент массива измените по следующему правилу: на месте элемента запишите остаток от деления элемента на индекс этого элемента. Например, для данного массива $A(5)$ {10, 7, 5, 11, 8} получаем ответ {0, 1, 2, 3, 3}.

Вариант 5

Найдите произведение всех элементов массива A, модуль которых меньше заданного числа b.

Вариант 6

Дан массив F. Найдите сумму наибольшего и наименьшего элементов этого массива. В виде подпрограмм оформите нахождение наибольшего и наименьшего значений из двух чисел.

Вариант 7

Дан массив. Подсчитайте, сколько раз встречается в нем максимальное по величине число. В виде подпрограммы оформите нахождение наибольшего из двух чисел.

Вариант 8

Составьте программу нахождения наименьшего общего кратного четырех натуральных чисел. Используйте следующее свойство $\text{НОК}(a,b) = \dfrac{ab}{\text{НОД}(a,b)}$.

Вариант 9

Составьте программу, которая в массиве A находит второе по величине число (т.е. выведите на печать число, которое меньше максимального элемента массива, но больше всех других элементов).

Вариант 10

Составьте программу, проверяющую, являются ли данные три числа взаимно простыми, т.е. НОД(a, b, c) = 1.

Вариант 11

Составьте программу вычисления суммы факториалов всех нечетных чисел от 1 до 5.

Вариант 12

Даны две дроби A/B и C/D (B, D — натуральные; A, C — целые числа). Составьте программу деления дроби на дробь. Ответ должен быть несократимой дробью.

Вариант 13

Даны две дроби A/B и C/D (B, D — натуральные; A, C — целые числа). Составьте программу умножения дроби на дробь. Ответ должен быть несократимой дробью.

Вариант 14

Даны две дроби A/B и C/D (B, D — натуральные; A, C — целые числа). Составьте программу вычитания из первой дроби второй. Ответ должен быть несократимой дробью.

Вариант 15

Даны две дроби A/B и C/D (B, D — натуральные; A, C — целые числа). Составьте программу сложения этих дробей. Ответ должен быть несократимой дробью.

Вариант 16

Составьте программу вычисления суммы факториалов всех четных чисел от 2 до 6.

Вариант 17

Дан массив A с четным числом элементов. Сформируйте массив B, элементами которого являются большие из двух рядом стоящих в массиве A чисел. (Например, если массив A состоит из элементов 1, 3, 5, −2, 0, 4, то элементами массива B будут 3, 5, 4.)

Вариант 18

Дано натуральное число n. Поменяйте порядок следования цифр в этом числе на обратный.

Вариант 19

Найдите число, большее 100, такое что сумма цифр при умножении его на a не меняется.

Вариант 20

Составьте программу, определяющую, в каком из данных двух чисел больше цифр.

Вариант 21

Дано натуральное число n. Если цифры этого числа образуют возрастающую последовательность, то поместите по адресу $76_{(16)}$ число один, иначе — нуль.

Вариант 22

Если данное натуральное число n делится на каждую из своих цифр, то поместите по адресу $76_{(16)}$ число один, иначе — нуль.

Вариант 23

Определите количество делителей данного натурального числа n.

Вариант 24

Напишите программу, определяющую сумму трехзначных чисел, содержащих только нечетные цифры. Определите также, сколько четных цифр в найденной сумме.

Вариант 25

Из заданного числа вычли сумму его цифр. Из результата вновь вычли сумму его цифр и т.д. Через сколько таких действий получится нуль?

Лабораторная работа № 5. Тексты, логические выражения, стек

Задания к лабораторной работе

Время выполнения 4 — 6 часов.

1) Введите значения переменных с клавиатуры, вычислите значение выражения и результат выведите на экран.
2) Выполните задание на обработку текста.
3) Составьте программу, которая позволяет построить таблицу истинности для заданного логического выражения и выведите ее на экран. Проверьте результаты работы программы с помощью составления таблицы истинности.

Примечание. Логическое значение ИСТИНА (TRUE) можно закодировать кодом FFFF, а ЛОЖЬ (FALSE) — кодом 0000. Возможны и другие варианты кодирования.

Варианты заданий

Вариант 1

а) $\dfrac{(n-1)n(n+1)}{6}$.

б) Подсчитайте число русских букв в данном тексте.

в) `(X and Y) or (Z and X)`.

Вариант 2

а) $a^3b + \dfrac{c(c-1)d}{2}$.

б) Подсчитайте число вхождений заданной буквы в данном тексте.

в) `not(X or Y or Z)`.

Вариант 3

1) $b^2 - 4ac$.

б) Подсчитайте число слов в строке. Слова отделяются друг от друга ровно одним пробелом.

в) `X and (not Y or Z).`

Вариант 4

а) $(a + b + c)^2$.

б) Определите, является ли данный текст числом, записанным в десятичной системе счисления, т.е. состоит только из арабских цифр от 0 до 9.

в) `not(X and Y) or Z.`

Вариант 5

а) $a^2 - (b - c)^2$.

б) Удалите из данной строки любой введенный с клавиатуры символ.

в) `not(X and Y and Z).`

Вариант 6

а) $a^4 - b^2 + \frac{c(c+1)^2}{4} - 17a$.

б) Составьте программу, удаляющую все лишние пробелы из данной строки, т.е. оставьте между словами не более одного пробела.

в) `X and not Y and Z.`

Вариант 7

а) $a^3b - ab^3$.

б) Определите длину самого короткого слова в заданном тексте.

в) `(X and Y) or not Z.`

Вариант 8

а) $\frac{c(c^2-4)(c^2-1)}{10}$.

б) Замените в заданном слове все буквы «о» пробелами.

в) `(X and Z) or (Y and Z).`

Вариант 9

а) $a^2 + b^2 + c^2$.

б) В тексте, состоящем из латинских букв и заканчивающемся точкой, подсчитайте число гласных букв.

в) `not(X or Y) and Z.`

Вариант 10

а) $2ab - c^2 + d^2$.

б) Дано слово. Переверните его.

в) `(X or Y) and not Z.`

Вариант 11

а) $\frac{n(n-1)}{2} - \frac{m(m^2-1)}{3}$.

б) Заданы фамилия, имя и отчество учащегося, разделенные пробелами. Оставьте только фамилию и инициалы.

в) `X or not(Y and Z).`

Вариант 12

а) $100a^2 - 4abcd$.

б) Вычеркните i-ю букву заданного слова.

в) `not(X and Y or Z).`

Вариант 13

а) $-10ab + 12cd - 4ad$.

б) Вычеркните из заданного слова все буквы, совпадающие с его последней буквой.

в) `not X and Y and Z.`

Вариант 14

а) $(x - y)z + z^2$.

б) Вычеркните из слова X те буквы, которые встречаются в слове Z.

в) `not(X or not Y or Z).`

Вариант 15

а) x^4y^4.

б) Составьте программу перевода строки строчных русских букв в прописные.

в) `(X and Z) or not Y.`

Вариант 16

а) $x^4 + y^2$.

б) Составьте программу, вычеркивающую каждую третью букву заданного слова.

в) `(X and Y) xor (X and Z).`

Вариант 17

а) $x^2(y^2 + z)$.

б) Составьте программу, позволяющую выяснить, на гласную или на согласную букву оканчивается заданное русское слово.

в) `(X xor Y) or (X xor Z).`

Вариант 18

а) $x(y^2 - z^2)$.

б) Дана строка. Подсчитайте в ней число вхождений букв **r**, **k**, **t**.

в) `not(X xor Y xor Z).`

Вариант 19

а) $xy^2 + yz$.

б) Дана строка. Определите, сколько в ней знаков «*», «;», «:».

в) `not X and (Y or Z).`

Вариант 20

а) $xy + z + z^2$.

б) Дана строка, среди символов которой есть двоеточие (:). Определите, сколько символов ему предшествует.

в) `X xor Y and not Z.`

Вариант 21

а) $(x + y)(y + z)x$.

б) Дана строка. Подсчитайте самую длинную последовательность подряд идущих букв **а**.

в) `not X xor Y xor Z.`

Вариант 22

а) $xy(z - x) + z$.

б) Дана строка, среди символов которой есть одна открывающаяся и одна закрывающаяся скобки. Выведите на экран все символы, расположенные внутри этих скобок.

в) `X and not Y xor Z.`

Вариант 23

а) $x + y(x + z)^2$.

б) Имеется строка, содержащая буквы латинского алфавита и цифры. Выведите на экран длину наибольшей последовательности цифр, идущих подряд.

в) `(X or Y) xor not Z.`

Вариант 24

а) $(x + y)^2 - z^2$.

б) В строке заменить все двоеточия (:) точкой с запятой (;). Подсчитайте число замен.

в) `X or Y or (X xor Z).`

Вариант 25

а) $z(x + y)(y - z)$.

б) Строка содержит одно слово. Проверьте, будет ли оно читаться одинаково справа налево и слева направо (т.е. является ли оно палиндромом).

в) `X xor (not Y or Z).`

§ 5. ВНЕШНИЕ УСТРОЙСТВА ЭВМ: ФИЗИЧЕСКИЕ ПРИНЦИПЫ И ХАРАКТЕРИСТИКИ

Рекомендации по проведению занятий

Занятия по этой теме проводятся на семинарских занятиях, а также путем разработки студентами рефератов.

Краткие сведения

Внешние (или, по-другому, периферийные) устройства ЭВМ прошли огромный путь в своем развитии — не меньший, чем процессоры и ОЗУ. Возможности компьютера, отнесение его к тому или иному поколению в значительной степени определяются номенклатурой и производительностью внешних устройств.

Внешние запоминающие устройства (ВЗУ) обеспечивают долговременное хранение программ и данных. Наиболее распространены следующие типы ВЗУ: ***накопители на магнитных дисках*** (НМД); их разновидности — накопители на гибких магнитных дисках (НГМД) и накопители на жестких магнитных дисках (НЖМД); ***накопители на оптических дисках*** (НОД); ***накопители на магнитных лентах*** (НМЛ).

Соответственно, физическими носителями информации, с которыми работают эти устройства, являются магнитные диски (МД), магнитные ленты (МЛ) и оптические диски (ОД).

Важнейшая, с точки зрения пользователя, характеристика любого диска — информационная емкость. Для гибких дисков (дискет) она чаще всего находится в диапазоне от одного до полутора мегабайт, хотя созданы дискеты с емкостью до 10 Мбайт. Специальные дискеты для резервного копирования (так называемые Zip-диски, для работы с которыми нужны особые дисководы) имеют емкость 100 Мбайт и более. Для жестких дисков в настоящее время (начало 2000 г.) наиболее характерна емкость порядка 10 Гбайт. Стандартная емкость наиболее распространенных оптических дисков категории CD ROM порядка 800 Мбайт. Такие характеристики дисков, как радиальная плотность записи (дорожек/см) и продольная плотность записи (байт/см) с точки зрения пользователя мало существенны. Еще по одной важной характеристике — надежности и долговечности хранения информации — оптические диски вне конкуренции.

Важнейшие пользовательские характеристики любого из накопителей — скорость доступа к определенному участку информации на диске и скорость записи или считывания информации. Для НГМД время доступа лежит в диапазоне 200—1000 мс, скорость чтения-записи 30—40 Кбайт/с. Для НЖМД эти параметры, соответственно, 5—100 мс и 0,5—1 Мбайт/с. Для НОД — 30—100 Мс и порядка нескольких Мбайт/с.

Устройства ввода информации определяются видом вводимых данных. При вводе символьной информации важнейшую роль играет ***клавиатура***. Важная характеристика клавиатуры — эргономичность, определяющая удобства пользователя. Современная клавиатура имеет 4 группы клавиш: пишущей машинки, служебные, функциональные и малой цифровой клавиатуры, за каждой из которых закреплено определенное назначение. Различные манипуляторы (***мышь, шар, джойстик*** и т.д.) обеспечивают перемещение курсора на экране и позволяют вводить управляющую информацию при работе с программами, использующими символьные или графические меню (т.е. практически со всеми современными программами). Для ввода графической информации используются ***сканеры***. Их важнейшие характеристики: разрешающая способность, возможность ввода цветных изображений, быстродействие, размер обрабатываемых изображений. Сканер выдает информацию в виде файла графического (растрового) формата; все остальное (например, распознание текстов, если они есть в исходном изображении) — дело программной обработки. *Средства речевого ввода* — микрофон и звуковая карта — также являются неотъемлемой принадлежностью современных компьютеров.

Устройства вывода информации включают дисплеи, принтеры, синтезаторы звука и др. ***Дисплей*** — устройство визуального отображения текстовой и графической информации. Дисплей относится к числу неотъемлемых принадлежностей компь-

ютера. Дисплеи классифицируются по нескольким разным параметрам, отражающим их назначение в конкретной компьютерной системе и возможности. Бывают дисплеи монохромные и цветные. Монохромный дисплей производит отображение в двух цветах — черном и белом, либо зеленом и черном и т.д. Высококачественный цветной дисплей может воспроизводить десятки основных цветов и сотни оттенков.

По физическим принципам, лежащим в основе конструкций дисплеев, подавляющее большинство их относится к дисплеям на базе электронно-лучевых трубок и к жидкокристаллическим дисплеям (последние особенно часто встречаются у портативных компьютеров). Выпускаются и плазменные дисплеи, пока не получившие широкого распространения.

Основные характеристики дисплеев с точки зрения пользователя таковы: разрешающая способность, число воспроизводимых цветов (для цветного дисплея) или оттенков яркости (для монохромного). Для алфавитно-цифрового дисплея разрешающая способность — число строк на экране и символов в каждой строке. К примеру, характеристики цветного графического дисплея SVGA (Super Video Grapics Adapter — видеографический адаптер повышенного разрешения) в нескольких из возможных режимах работы таковы: при разрешающей способности 768×1824 он позволяет различать 16 цветов, при разрешающей способности 480×640 — 64 оттенка разных цветов.

Огромную роль при выводе информации играют разнообразные печатающие устройства — ***принтеры***. Принтеры бывают ударные (точечно-матричные) и безударные (струйные, лазерные, термографические). Важнейшие пользовательские характеристики принтеров — цветной или монохромный, быстродействие, формат бумаги. Качество печати (и скорость работы принтера) во многом определяются программной поддержкой, так как в настоящее время печать большей частью происходит загружаемыми шрифтами, а не теми, которые встроены в сам принтер.

Контрольные вопросы

1. Какие существуют категории внешних устройств?
2. Каковы профессиональные (качественные и количественные) характеристики накопителей на гибких магнитных дисках? жестких магнитных дисках? оптических дисках?
3. Какие физические принципы положены в основу магнитной записи? оптической записи?
4. Каковы профессиональные (качественные и количественные) характеристики дисплеев?
5. Какие физические принципы положены в основу дисплеев на электронно-лучевых трубках? на жидких кристаллах?
6. Каковы профессиональные (качественные и количественные) характеристики принтеров?
7. Какие физические принципы положены в основу устройства струйных принтеров? лазерных принтеров? термографических принтеров?

Темы для рефератов

1. Современные накопители информации, используемые в вычислительной технике.
2. Дисплеи, их эволюция, направления развития.
3. Печатающие устройства, их эволюция, направления развития.

4. Сканеры и программная поддержка их работы.
5. Средства ввода и вывода звуковой информации.

Темы семинарских занятий

1. Устройство и характеристики внешних запоминающих устройств.
2. Устройство и характеристики устройств ввода и вывода информации.

Дополнительная литература

1. *Богумирский Б.С.* Руководство пользователя ПЭВМ: В 2 ч. Ч. 1. — СПб.: Печатный Двор, 1994.
2. Документация по устройствам ЭВМ фирм-изготовителей.

§ 6. ЛОГИЧЕСКИЕ ОСНОВЫ ФУНКЦИОНИРОВАНИЯ ЭВМ

Рекомендации по проведению занятий

Понимание логических основ работы ЭВМ является существенным для подготовки специалиста по информатике. При этом достигается понимание процессов, происходящих уже на уровне ниже архитектуры ЭВМ. Логические принципы лежат между архитектурой и реализацией электронных устройств — области, заведомо не относящейся к информатике.

При изучении данной темы будущие учителя должны освоить элементарные навыки в понимании логических схем базовых логических элементов, таких как сумматор, триггер и др., провести связи между математической логикой и устройством ЭВМ. Такие же вопросы, как устройство операционных узлов цифровой техники — регистров, счетчиков и др., — лежат за пределами этой темы (и собственно информатики). Им должен быть посвящен специальный курс типа «Введение в цифровую технику и микроэлектронику», входящий в подготовку учителя информатики.

Занятия по теме проводятся на семинарских и лабораторных занятиях, а также путем работы над рефератами.

Краткие сведения

Логические выражения

Логическое выражение состоит из логических операндов, соединенных с помощью логических операций. В качестве логических операндов могут выступать логические константы, переменные, а также отношения (сравнения) между двумя величинами. Логические выражения могут принимать одно из двух значений: ИСТИНА (TRUE или 1), ЛОЖЬ (FALSE или 0).

Существует несколько логических операций, все возможные значения которых описывают обычно с помощью таблиц истинности (это возможно по той причине, что все сочетания значений логических операндов очень легко перечислить) (см. ниже — табл. 4.1).

Приоритет операций при вычислении значения логического выражения следующий (в порядке понижения):

1) отрицание (NOT, НЕ);

2) конъюнкция (AND, И);

3) дизъюнкция и исключающее ИЛИ (OR, ИЛИ; XOR, ИСКЛЮЧАЮЩЕЕ ИЛИ);

4) операции отношения (равно, не равно, больше, меньше, больше или равно, меньше или равно).

Если существует необходимость изменения порядка вычисления значения выражения, надо использовать круглые скобки. Чаще всего это применяется к операциям отношения, поскольку они имеют самый низкий приоритет, а их чаще всего необходимо вычислить в первую очередь.

Например, вычислим значение выражения $(a \le b)$ OR $(c \ne b)$ при a=2, b=3, c = 3:

1) $2 \le 3 \rightarrow$ TRUE;

2) $3 \ne 3 \rightarrow$ FALSE;

3) TRUE OR FALSE $\rightarrow$ TRUE.

Логические элементы

При всей сложности устройства электронных блоков современных ЭВМ выполняемые ими действия осуществляются с помощью комбинаций относительно небольшого числа типовых логических узлов. Основные из них таковы:

- регистры;
- комбинационные преобразователи кодов (шифратор, дешифратор, мультиплексор и др.);
- счетчики (кольцевой, синхронный, асинхронный и др.);
- арифметико-логические узлы (сумматор, узел сравнения и др.).

Из этих узлов строятся интегральные микросхемы очень высокого уровня интеграции: микропроцессоры, модули ОЗУ, контроллеры внешних устройств и т.д.

Сами указанные узлы собираются из основных базовых логических элементов — как простейших, реализующих логические функции И, ИЛИ, НЕ, И–НЕ, ИЛИ–НЕ и им подобных (элементы комбинационной логики, для которых значение функции на выходе однозначно определяется комбинацией входных переменных в данный момент времени), так и более сложных, таких как триггеры (элементы последовательностной логики, для которых значение функции зависит не только от текущих значений переменных на входе, но и от их предшествующих значений).

Условные обозначения основных элементов комбинационной логики приведены на рис. 4.3, соответствующие значения переменных («таблицы истинности») — в табл. 4.1. Отметим, что кружочек на схеме на выходе из логического элемента означает, что элемент производит логическое отрицание результата операции, указанной внутри прямоугольника.

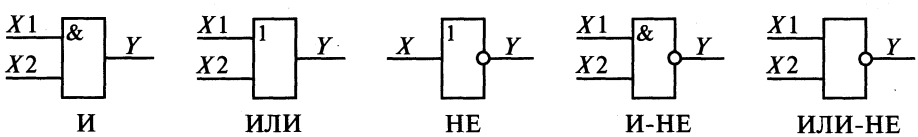


Рис. 4.3. Основные элементы комбинационной логики

Таблица 4.1

Таблица истинности логических операций

$X1$	$X2$	$X1 \wedge X2$ (И)	$X1 \vee X2$ (ИЛИ)	$\overline{X1 \wedge X2}$ (И-НЕ)	$\overline{X1 \vee X2}$ (ИЛИ-НЕ)
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	0

Контрольные вопросы

1. Назовите основные логические операции и приведите их таблицы истинности.
2. Что такое логическое выражение?
3. Каков порядок выполнения операций при вычислении значения логического выражения?
4. Приведите примеры логических выражений и вычисления их значений.
5. Назовите элементарные логические элементы и приведите их обозначения на схемах.
6. Изобразите электрические схемы, реализующие элементарные логические элементы.
7. Приведите примеры построения схем на логических элементах на основе логического выражения.
8. Приведите примеры построения логических выражений по заданным логическим схемам.
9. Что такое триггер? Перечислите виды триггеров и коротко их охарактеризуйте.
10. Чем отличается синхронный триггер от несинхронного?
11. Проиллюстрируйте на примерах хранение информации в триггере и запись нуля или единицы.
12. Какое состояние триггера называют недопустимым?
13. Расскажите об элементе «Исключающее ИЛИ», приведите таблицу истинности для соответствующей логической операции.
14. Расскажите о полусумматоре.
15. Расскажите о сумматоре и организации переноса «запасного» разряда.

Темы для рефератов

1. Различные виды триггеров и их сопоставление.
2. Операционные узлы ЭВМ.

Темы семинарских занятий

1. Элементы комбинационной логики.
2. Элементы последовательностной логики.

Задачи и упражнения

Упражнение № 1

Найти значение приведенных ниже выражений:

1) $x > y$	при а) $x = 2$, $y = 2$; б) $x = 2$, $y = -8$;
2) A OR B AND NOT C	при A = False, B = True, C = False;
3) NOT $(A < B)$	при а) $A = 7$, $B = 9$; б) $A = 0$, $B = 2$;
4) $(x < y)$ OR $(x = z)$	при а) $x = 0$, $y = 0$, $z = 0$; б) $x = 0$, $y = -8$, $z = 0$;
5) $(a \le z)$ AND $(z > 2)$ AND $(a \ne 5)$	при а) $a = 2$, $z = 4$; б) $a = -5$, $z = 0$;
6) $A \le B$	при а) $A = 2$, $B = 2$; б) $A = 2$, $B = -8$;
7) A AND B OR NOT C	при A = False, B = True, C = False;
8) NOT $(x \ge y)$	при а) $x = 7$, $y = 9$; б) $x = 0$, $y = 2$;
9) $(x < y)$ AND $(x = z)$	при а) $x = 0$, $y = 0$, $z = 0$; б) $x = 0$, $y = -8$, $z = 0$;
10) $(a \le z)$ OR $(z > 2)$ OR $(a \ne 5)$	при а) $a = 5$, $z = -4$; б) $a = -5$, $z = 0$;
11) $(x = y)$ OR $(z < 4)$	при а) $x = 5$, $y = 7$, $z = 0$; б) $x = 5$, $y = -7$, $z = 10$;
12) $(x \ne y)$ AND $(z < 4)$	при а) $x = 5$, $y = 7$, $z = 0$; б) $x = 5$, $y = -7$, $z = 10$;
13) NOT $(x > z)$	при а) $x = 5$, $z = -2$; б) $x = -5$, $z = 2$;
14) NOT A OR B	при A = True, B = False;
15) $(A$ OR $B)$ AND C	при A = True, B = False, C = True;
16) $(x \le y)$ OR $(z > -4)$	при а) $x = 5$, $y = 7$, $z = 0$; б) $x = 5$, $y = -7$, $z = 10$;
17) $(x \ge y)$ AND $(z \le 4)$	при а) $x = 5$, $y = 7$, $z = 0$; б) $x = 5$, $y = -7$, $z = 10$;
18) NOT $(x \le z)$	при а) $x = 5$, $z = -2$; б) $x = 2$, $z = 2$;
19) A OR NOT B	при A = False, B = False;
20) A OR B AND C	при A = True, B = False, C = True;
21) $(x \ge y)$ OR $(z > -4)$	при а) $x = 5$, $y = 7$, $z = 0$; б) $x = 5$, $y = -7$, $z = 10$;
22) $(x \le y)$ AND $(z \le 4)$	при а) $x = -5$, $y = -7$, $z = 0$; б) $x = 5$, $y = -7$, $z = -10$;
23) NOT $(x \ne z)$	при а) $x = 5$, $z = -2$; б) $x = 2$, $z = 2$;
24) A AND NOT B	при A = True, B = False;
25) NOT $(A$ OR $B)$ AND C	при A = True, B = False, C = True.

Упражнение № 2

По заданной логической схеме (рис. 4.4) составить логическое выражение и заполнить для него таблицу истинности.

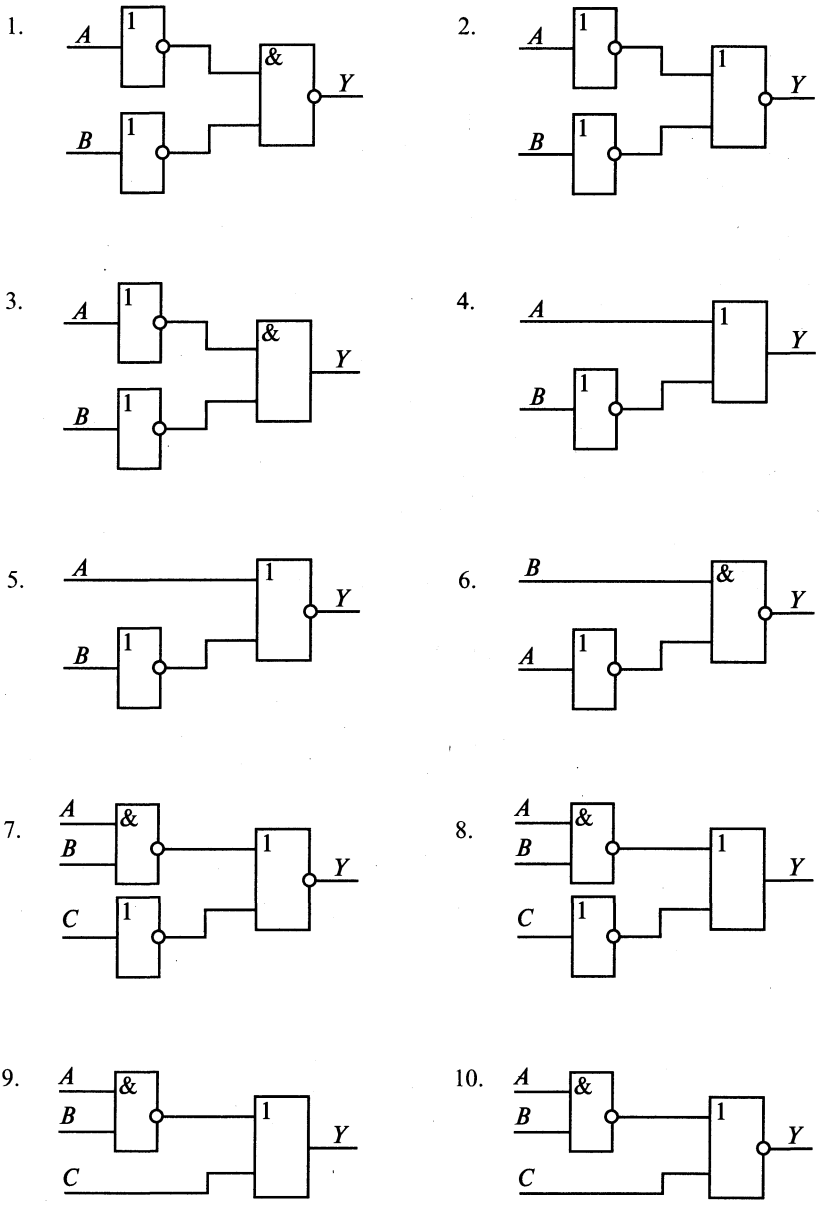


Рис. 4.4. Логические схемы

Упражнение № 3

По заданному логическому выражению составить логическую схему и построить таблицу истинности:

1. *A* AND *B* OR NOT *C*;

2. *A* AND NOT *B* OR *C*;

3. NOT(*A* AND NOT *B*) OR *C*;
4. *A* OR NOT *B* AND *C*;
5. *A* OR NOT(NOT *B* AND *C*);
6. NOT(*A* OR *B*) AND NOT *C*;
7. NOT(*A* AND *B*) OR NOT *C*;
8. NOT *A* OR *B* AND *C*;
9. NOT(NOT *A* OR *B* OR *C*);
10. NOT(NOT *A* OR *B* AND NOT *C*).

Упражнение № 4

Логические элементы И–НЕ и ИЛИ–НЕ называют базовыми, поскольку любой из перечисленных на рис. 4.3 логических элементов можно выразить только через И–НЕ (или ИЛИ–НЕ). Соответствующие схемы для одного из этих случаев приведены на рис. 4.5.

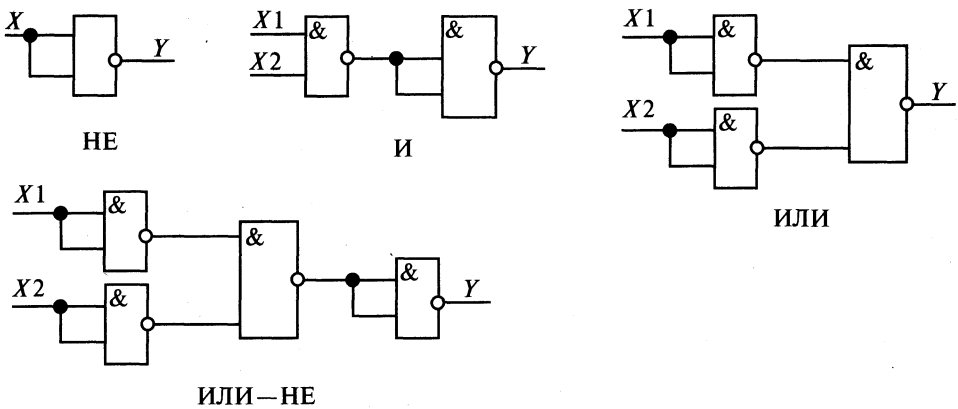


Рис. 4.5. Реализация логических элементов через базовый И–НЕ

Для того чтобы убедиться в справедливости сформулированного выше утверждения, достаточно перебрать все возможные комбинации входных сигналов и найти результат. Покажем это на примере схемы для «И»; промежуточный результат обозначим через *Z* (табл. 4.2).

Таблица 4.2

Реализация схемы «И»

*X*1	*X*2	*Z*	*Y*
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Таким образом, сравнивая с табл. 4.1, убеждаемся в справедливости высказанного выше утверждения.

Задание 1.

Выполнить указанную проверку для всех схем на рис. 4.5.

Задание 2.

Разработать схемы реализации элементов НЕ, И, ИЛИ, И–НЕ через базовый логический элемент ИЛИ–НЕ.

Упражнение № 5

Кроме указанных выше одно- и двухвходовых элементов комбинационной логики, используют и более сложные — трех-, четырехвходовые и др., реализующие определенные логические функции более чем двух аргументов. Один из таких элементов изображен на рис. 4.6, *а*; он реализует действие $Y = \overline{X1 \wedge X2 \wedge X3 \wedge X4}$.

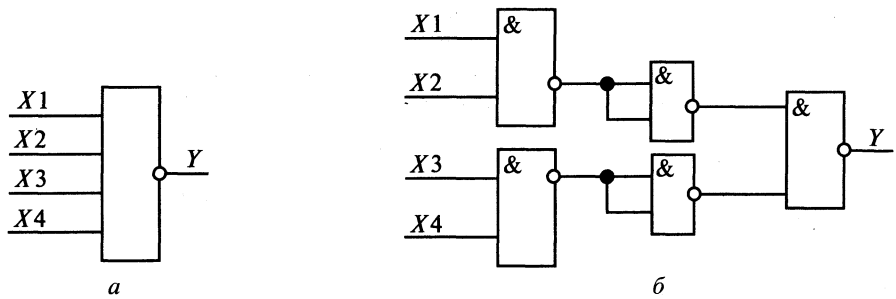


Рис. 4.6. Один из четырехвходовых элементов комбинационной логики (*а*) и его реализация через двухвходовые элементы (*б*)

Задание 3. Проверить, что четырехвходовый элемент, изображенный на рис. 4.6, *а*, эквивалентен комбинации двухвходовых элементов, изображенной на рис. 4.6, *б*.

Упражнение № 6

Для сложения двух одноразрядных чисел применяется так называемый полусумматор, логическая схема которого изображена на рис. 4.7. Схема реализует арифметическое действие $A + B = C_0S$, где A и B — одноразрядные двоичные числа, C_0 и S — соответственно старший и младший двоичные разряды суммы (например, если $A = 0$ и $B = 1$, то $C_0 = 0$ и $S = 1$).

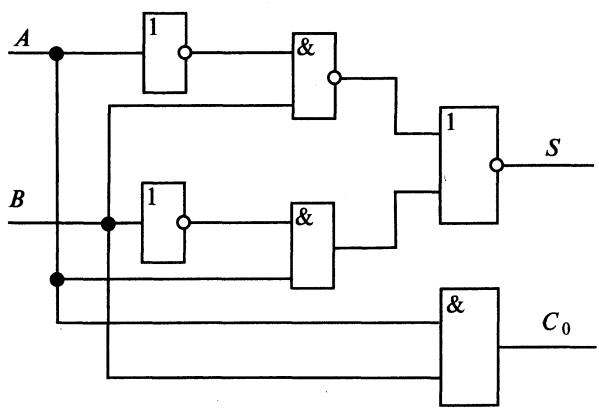


Рис. 4.7. Логическая схема полусумматора

Задание 4.
Проверить, что имеют место логические формулы

$$S = (\bar{A} \wedge B) \vee (A \wedge \bar{B}), \quad Co = A \wedge B$$

Примечание. Цифра «1» отождествляется с логическим «да» («истина», или 1), цифра «0» — с логическим «нет» («ложь», или 0).

Упражнение № 7

Для сложения двух двоичных разрядов A и B многоразрядного числа с учетом возможного добавления цифры C_i, оставшейся от сложения предыдущих разрядов используется так называемый одноразрядный сумматор.

Пример. Складываем «столбиком» $101_{(2)} + 111_{(2)}$. Для сложения крайних правых цифр достаточно использовать полусумматор; согласно обозначениям, принятым в упражнении № 6, имеем: $A = 1$, $B = 1 \Rightarrow C_0 = 0$, $S = 1$. Продолжаем сложение — теперь уже полусумматором не обойтись, ибо надо фактически сложить три цифры: 0 и 1 (вторые справа разряды слагаемых) и 1, «пришедшую» из сложения предыдущих разрядов.

Эта задача решается с помощью одноразрядного сумматора (рис. 4.8).

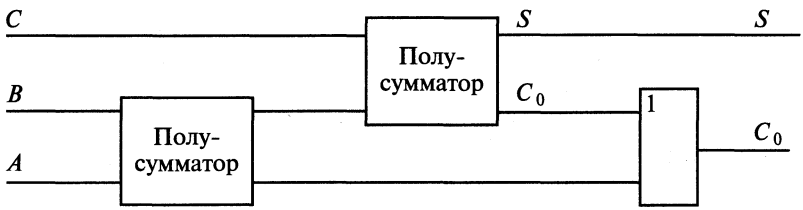


Рис. 4.8. Логическая схема одноразрядного сумматора

Задание 5.

Проверить перебором всех возможных вариантов, что схема на рис. 4.8 действительно реализует указанное выше действие.

Упражнение № 8

Для сложения многоразрядных чисел используют устройства — сумматоры. Логическая схема трехразрядного сумматора приведена на рис. 4.9.

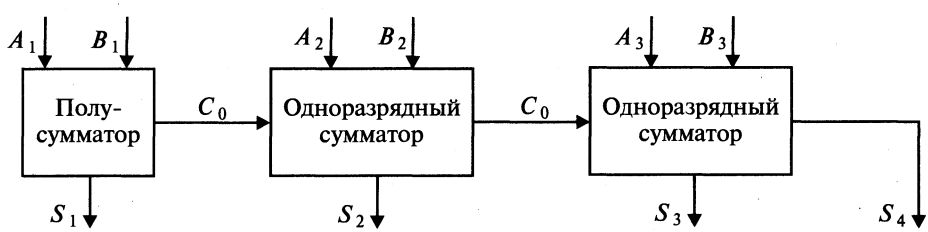


Рис. 4.9. Логическая схема трехразрядного сумматора

Задание 6.

Разобрать на примерах работу трехразрядного сумматора.

Задание 7.

Построить схему восьмиразрядного сумматора и разобрать его действие на примерах.

Упражнение № 9

Основное устройство последовательностной логики — триггер. На рис. 4.10 — схема простейшего RS-триггера. R и S — входы, Q и $\overline{Q}$ — выходы (прямой и инверсный соответственно).

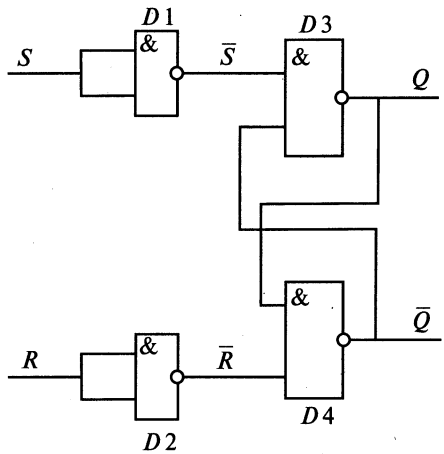


Рис. 4.10. Логическая схема RS-триггера

Состояние на выходе триггера зависит не только от значений R и S на входе, но и от того, в каком состоянии находится триггер. Благодаря этому его можно использовать для записи и хранения информации (одного бита).

Под действием входных сигналов триггер может переключаться из одного устойчивого состояния в другое. RS-триггер является асинхронным, поскольку информация в нем может изменяться в любой момент при изменении входных сигналов (в отличие от синхронизируемых триггеров, в которых информация на выходе может меняться только в определенные моменты времени).

Вход S (Set) — вход установки триггера в единичное состояние, вход R (Reset) — сброса в нулевое состояние. Допустим, на входе S= 1 и R= 0. Тогда на выходе будет Q= 1 и $\overline{Q}$ = 0. После исчезновения выходного сигнала (т.е. задания S= 0, R= 0) сохранится указанный выходной сигнал — произошла запись информации.

Задание 8.

Отследить по схеме на рис. 4.10 справедливость сформулированного выше утверждения.

Задание 9.

Найти, каким будет состояние RS-триггера при входном сигнале R= 1 и S= 0 и каким оно станет после исчезновения сигнала.

Задание 10.

Проверить, что при входном сигнале S= 1, R= 1 оба выходных сигнала равны нулю, т.е. состояние системы не определено (в силу чего комбинация S= 1, R= 1 является запрещенной).

Упражнение № 10

Альтернативная схема RS-триггера на элементах И–НЕ имеет вид, изображенный на рис. 4.11 (входные сигналы R и S при этом замещены на инверсные $\overline{R}$ и $\overline{S}$).

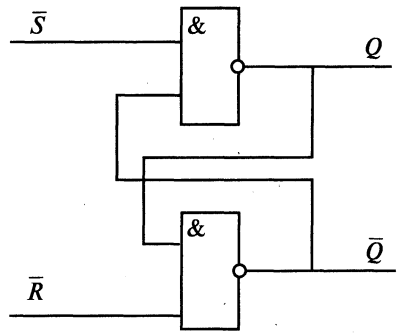


Рис. 4.11. Альтернативная схема RS-триггера

Задание 11.

Проанализировать работу RS-триггера, основанного на схеме рис. 4.11. Подтвердить, что утверждения табл. 4.3 верны.

Таблица 4.3

Таблица истинности для RS-триггера

S	R	S	R	Q	Q	Примечания
0	0	1	1	0	0	Хранение
0	1	1	0	0	1	Запись 0
1	0	0	1	1	0	Запись 1
1	1	0	0	–	–	Запрещено

Упражнение № 11

Найти в литературе логическую схему одного из синхронных (синхронизируемых) триггеров и разобрать его работу.

Упражнение № 12

Сконструируйте устройство, собранное только из базовых двухвходных элементов И–НЕ, реализующее операцию: а) НЕ; б) И; в) ИЛИ; г) ИЛИ–НЕ; д) И–ИЛИ–НЕ (NOT (*A* AND *B* OR *C* AND *D*)); е) сложения по модулю два (NOT (NOT(*A* AND *B*) OR NOT(*C* AND *D*))).

Дополнительная литература

1. *Брой М.* Информатика: В 3 т. Т. 2. Вычислительные структуры и машинно-ориентированное программирование: Пер. с нем. — М.: Диалог-МИФИ, 1996.

2. *ВершининО. Е.* За страницами учебника информатики. — М.: Просвещение, 1992.

3. *ЕреминЕ.А.* Как работает современный компьютер. — Пермь: Изд-во ПРИПИТ, 1997.
4. *Мальцева Л.А., Фромберг Э.М., Ямпольский В.С.* Основы цифровой техники. — М.: Радио и связь, 1986.
5. *Мнеян М.Г.* Физические принципы работы ЭВМ. — М.: Просвещение, 1987.
6. *Паскалев Ж.* Первые шаги в вычислительной технике. — М.: Радио и связь, 1987.
7. *Савельев А.Я.* Арифметические и логические основы цифровых автоматов. — М.: Высш. шк., 1980.
8. *Тотхейм Я.* Основы цифровой электроники: Пер. с англ. — М.: Мир, 1988.
9. *Эндерлайн Р.* Микроэлектроника для всех: Пер. с нем. — М.: Мир, 1979.
10. *Ямпольский В.С.* Основы автоматики и электронно-вычислительной техники. — М.: Просвещение,1991.

Тесты к главе 4

История развития вычислительной техники

1. Кто изобрел первую действующую суммирующую машину:
   1) Паскаль; 2) Ньютон; 3) Воль; 4) Нейман?
2. В каком году построили первую действующую автоматическую цифровую вычислительную машину:
   1) 1941; 2) 1946; 3) 1942; 4) 1944?
3. Кто изобрел «аналитическую машину»:
   1) Кант; 2) Бэббидж; 3) Нортон; 4) Нейман?
4. Арифмометр — это:
   1) механическое вычислительное устройство, способное выполнять 4 арифметических действия;
   2) устройство, выполняющее основные логические действия;
   3) логическое устройство, являющееся прототипом машины Тьюринга;
   4) арифметико-логическое устройство, выполняющее арифметические и логические действия.
5. Первая релейная машина называлась:
   1) ЭРВМ-1; 2) ЕС-ВМ; 3) БЭВМ; 4) МАРК-I.
6. Основным недостатком первых ЭВМ была:
   1) неспособность сохранять программу;
   2) неспособность выводить информацию;
   3) неспособность взаимодействовать с оператором;
   4) неспособность взаимодействовать между собой.
7. Первая действующая ЭВМ называлась:
   1) Марк-I; 2) Колосс; 3) Урал; 4) ENIAC.
8. Какой принцип лежал изначально в основе ЭВМ:
   1) нейрофизический; 2) нейротехнологический;
   3) физико-технологический; 4) физико-информационный?
9. Когда была создана первая ЭВМ с использованием транзисторов:
   1) 1948; 2) 1956; 3) 1949; 4) 1957?
10. С какого поколения машины стали делиться на большие, средние и малые:
    1) с 1-го; 2) со 2-го; 3) с 3-го; 4) с 4-го?
11. В каком поколении появились первые унифицированные ЭВМ:
    1) в 1-м; 2) во 2-м; 3) в 3-м; 4) в 4-м?

12. Когда был создан первый микрокомпьютер:
1) 1971; 2) 1968; 3) 1972; 4) 1980?
13. Когда был создан первый компьютер Apple:
1) 1969; 2) 1976; 3) 1981; 4) 1983?
14. Когда был создан первый компьютер IBM PC:
1) 1963; 2) 1971; 3) 1973; 4) 1981?
15. Первое поколение ПК были:
1) 16-битные; 2) 32-битные; 3) 8-битные; 4) 4-битные.
16. Самые мощные суперЭВМ представлены серией:
1) PC; 2) Macintosh; 3) Cray; 4) Apple.
17. Мини-ЭВМ появились:
1) в начале 70-х; 2) в середине 80-х; 3) в начале 80-х; 4) в начале 90-х.
18. Примерами супермини-ЭВМ является семейство:
1) рабочая станция; 2) CRAY-3; 3) ES; 4) VAX-11.
19. Особенность аналоговой вычислительной машины:
1) цифровая, обрабатывает информацию в непрерывной форме;
2) нецифровая, обрабатывает информацию в непрерывной форме;
3) нецифровая, обрабатывает информацию в дискретной форме;
4) цифровая, обрабатывает информацию в дискретной форме.
20. Основные учения об архитектуре вычислительных машин заложил:
1) Паскаль; 2) Фон Нейман; 3) Вуль; 4) Лейбниц.
21. Принцип хранимой программы предложил:
1) Бэббидж; 2) Тьюринг; 3) Фон Нейман; 4) Ньютон.
22. Появление 3-го поколения ЭВМ было обусловлено:
1) переходом от ламп к транзисторам;
2) переходом от транзисторов к интегральным микросхемам;
3) переходом от интегральных микросхем к микропроцессору;
4) переходом от транзисторов к большим интегральным схемам.
23. Первая интегральная микросхема родилась в:
1) 1959; 2) 1947; 3) 1974; 4) 1961.
24. Процессор Intel 4004 запущен в серийное производство в:
1) 1968; 2) 1971; 3) 1973; 4) 1979.
25. Intel 4004 является:
1) 4-разрядным процессором; 2) 8-разрядным процессором;
3) 16-разрядным процессором; 4) 32-разрядным процессором.
26. Intel 8008 появился в:
1) 1972; 2) 1973; 3) 1977; 4) 1980.
27. Intel 8008:
1) 4-разрядный; 2) 8-разрядный; 3) 16-разрядный; 4) 31-разрядный.
28. Intel 8080 появился в:
1) 1974; 2) 1975; 3) 1978; 4) 1980.
29. Intel 8080 имел память:
1) до 8 Кбайт; 2) до 16 Кбайт;
3) до 32 Кбайт; 4) до 64 Кбайт.
30. Время появления процессора Pentium:
1) 1992; 2) 1993; 3) 1994; 4) 1995.
31. Время появления Celeron:
1) 1995; 2) 1996; 3) 1997; 4) 1998.
32. Процессор Pentium-II впервые появился в:
1) 1994; 2) 1997; 3) 1998; 4) 1999.

Архитектура ЭВМ

1. Основное требование архитектурной совместимости ЭВМ:
   1) все программы данной модели выполнимы на более старших моделях, но не обязательно наоборот;
   2) все программы данной модели выполнимы на более старших моделях и наоборот;
   3) все машины одного семейства, независимо от их конкретного устройства и фирмы производителя, должны быть способны выполнять одну и ту же программу;
   4) все машины данного семейства должны работать одинаково.
2. Архитектура — это:
   1) общие принципы построения ЭВМ, реализующие программное управление работой и взаимодействием основных ее функциональных узлов;
   2) общие принципы построения ЭВМ, не реализующие программное управление работой;
   3) дизайн внешнего вида ЭВМ;
   4) принцип соединения внешних устройств к ЭВМ.
3. В современных компьютерах устройство управления и АЛУ объединены:
   1) в процессоре; 2) в материнской плате; 3) в ВЗУ; 4) в ПЗУ.
4. Подавляющее большинство современных машин являются:
   1) арифметико-логическими машинами; 2) машинами Тьюринга;
   3) фон-неймановскими машинами; 4) релейными машинами.
5. Контроллеры возникли в связи с решением проблемы:
   1) разгрузки процессора;
   2) загрузки процессора;
   3) медленная работа устройств ввода-вывода;
   4) медленная работа процессора.
6. Важной отличительной чертой машин 3-го и 4-го поколений является:
   1) мультипроцессорная среда; 2) интеллектуальные контроллеры;
   3) сопроцессоры; 4) устройство ввода-вывода.
7. Основная часть дисплея — это:
   1) монитор; 2) видеокарта;
   3) люминоформный слой; 4) электронно-лучевая трубка.
8. Изображение хранится:
   1) на экране монитора; 2) в ОЗУ;
   3) в видеоОЗУ; 4) в ВЗУ.
9. Сопроцессоры используются для:
   1) ускорения ввода информации на экран; 2) ускорения передачи данных;
   3) ускорения обработки данных; 4) операции с плавающей точкой.
10. Видеопроцессоры используются для:
   1) ускорения ввода информации на экран; 2) ускорения передачи данных;
   3) ускорения обработки данных; 4) операции с плавающей точкой.
11. При включении питания или нажатии на кнопку сброса счетчик адреса команд:
   1) указывает на 0;
   2) аппаратно устанавливается на 0;
   3) программно устанавливается на стартовый адрес, находящийся в ПЗУ программы инициализации всех устройств начальной загрузки;
   4) аппаратно заносится в стартовый адрес, находящийся в ПЗУ программы инициализации всех устройств начальной загрузки.

12. При методе конвейеризации внутреннее устройство процессора работает:
1) последовательно; 2) линейно;
3) параллельно; 4) используя буфер обмена данных.
13. Команды передачи данных:
1) копируют информацию из одного места в другое;
2) сдвигают двоичный код влево или вправо;
3) обмениваются информацией с внешними устройствами;
4) реализуют нелинейные алгоритмы.
14. Операции умножения или деления:
1) копируют информацию из одного места в другое;
2) сдвигают двоичный код влево или вправо;
3) обмениваются информацией с внешними устройствами;
4) реализуют нелинейные алгоритмы.
15. Команды ввода-вывода:
1) копируют информацию из одного места в другое;
2) сдвигают двоичный код влево или вправо;
3) обмениваются информацией с внешними устройствами;
4) реализуют нелинейные алгоритмы.
16. Команды управления:
1) копируют информацию из одного места в другое;
2) сдвигают двоичный код влево или вправо;
3) обмениваются информацией с внешними устройствами;
4) реализуют нелинейные алгоритмы.
17. Операционная часть команды указывает:
1) на код операций; 2) на адрес кода операций;
3) на адрес хранения кода; 4) на номер кода в таблице операций.
18. Команды могут быть одноадресные, двухадресные, трехадресные, в зависимости от:
1) разрядности процессора; 2) разрядности шины данных;
3) разрядности адресной шины;
4) количества участвующих в них операндов.
19. Адресная часть команды описывает:
1) где используемая информация хранится;
2) где хранится блок информации;
3) где хранится код операции;
4) указывает на адрес начала выполнения операции.
20. Первые ЭВМ имели:
1) нуль-адресную систему команд; 2) одноадресную систему команд;
3) двухадресную систему команд; 4) трехадресную систему команд.
21. Нуль-адресная система машин использует:
1) буфер; 2) стек; 3) процессорное ОЗУ; 4) программную очередь.
22. Ячейка ЭВМ 2-го поколения состояла:
1) из 32 двоичных разрядов; 2) из 24 двоичных разрядов;
3) из 16 двоичных разрядов; 4) из 8 двоичных разрядов.
23. В ЭВМ 3-го поколения минимальная порция информации для обмена с ОЗУ равна:
1) 32 двоичным разрядам; 2) 24 двоичным разрядам;
3) 16 двоичным разрядам; 4) 8 двоичным разрядам.
24. Машинное слово в ЭВМ 3-го поколения:
1) 1 байт; 2) 2 байта; 3) 3 байта; 4) 4 байта.

1. МП 4004 содержит:
   1) 2000 транзисторов; 2) 2200 транзисторов;
   3) 2400 транзисторов; 4) 2600 транзисторов.
2. МП 8080 содержит:
   1) 4000 транзисторов; 2) 4400 транзисторов;
   3) 4800 транзисторов; 4) 5200 транзисторов.
3. Intel 80486 содержит:
   1) около 1,2 млн транзисторов; 2) около 800 транзисторов;
   3) около 1,7 млн транзисторов; 4) около 2 млн транзисторов.
4. МП 8086 является:
   1) 8-разрядным; 2) 16-разрядным; 3) 24-разрядным; 4) 32-разрядным.
5. МП 80386 является:
   1) 8-разрядным; 2) 16-разрядным; 3) 24-разрядным; 4) 32-разрядным.
6. Процессор Power PC разработан в:
   1) 1993; 2) 1995; 3) 1989; 4) 1990.
7. Транспьютер содержит процессорное ОЗУ:
   1) от 2 до 4 Кбайт; 2) от 2 до 8 Кбайт;
   3) от 2 до 16 Кбайт; 4) от 2 до 32 Кбайт.
8. Процессор Pentium имеет:
   1) 16-разрядную магистраль; 2) 32-разрядную магистраль;
   3) 64-разрядную магистраль; 4) 128-разрядную магистраль.
9. В отличие от семейства процессоров PC-486 в процессоре Pentium одно исполнительное устройство заменено:
   1) на 2 устройства; 2) на 3 устройства;
   3) на 4 устройства; 4) не заменено.
10. Внутренний КЭШ процессора Pentium разделен на:
   1) КЭШ команд и КЭШ разрядов; 2) КЭШ разрядов и КЭШ данных;
   3) КЭШ команд и КЭШ адресов; 4) КЭШ команд и КЭШ данных.
11. Первое поколение процессоров Pentium имели тактовые частоты:
   1) 60 и 66 МГц; 2) 66 и 70 МГц; 3) 60 и 70 МГц; 4) 100 и 166 МГц.
12. Частота шины у Pentium была равна:
   1) частоте ядра; 2) удвоенной частоте ядра;
   3) утроенной частоте ядра; 4) регулировалась пользователем.
13. Процессоры Celeron имеют интегральный КЭШ второго уровня размером:
   1) 16 Кбайт; 2) 32 Кбайта; 3) 64 Кбайта; 4) 128 Кбайт.
14. КЭШ процессора Celeron встроен:
   1) в ядро процессора; 2) в материнскую плату;
   3) является съемным; 4) является заменимым.
15. Процессор Pentium поддерживает:
   1) 2-процессорную систему; 2) 3-процессорную систему;
   3) 4-процессорную систему; 4) многопроцессорную систему.
16. Pentium-II изготовлен:
   1) по 0,25-микронной технологии; 2) по 0,33-микронной технологии;
   3) по 0,46-микронной технологии; 4) по 0,128-микронной технологии.
17. Для программиста доступна:
   1) вся рабочая память процессора;
   2) внутренняя память процессора недоступна;
   3) внутренняя память доступна через регистры;

4) внутренняя память доступна через информационную магистраль.

18. Какова роль счетчика адреса команд:
1) сохраняет адрес очередной команды программы;
2) счетчик операций процессора;
3) счетчик внутренних операций внутри системы;
4) указатель на адрес контрольной суммы команд.

19. В регистре состояния процессора хранится (хранятся):
1) сведения о текущих режимах работы процессора;
2) информация о результатах выполняемых программ;
3) режим работы процессора и результат;
4) флаги, описывающие состояние внутренней памяти.

20. Аккумулятор используется:
1) для указания на стек;
2) для битового сложения;
3) как место для проведения операций и сохранения их результатов;
4) как регистр приемника.

21. Говоря о 16-разрядной ЭВМ, имеют в виду:
1) разрядность шины данных 16 бит; 2) разрядность шины адреса 16 бит;
3) размер слова 16 бит; 4) размер внутренних регистров памяти 16 бит.

22. В защищенном режиме работы процессора начальные адреса сегментов вычисляются:
1) умножением на 16 содержимого сегментных регистров;
2) умножением на 32 содержимого сегментных регистров;
3) извлекаются из таблиц сегментных дескрипторов;
4) извлекаются из таблиц сегментных дескрипторов, индексируемых теми же сегментными регистрами.

23. Каждый сегментный дескриптор занимает:
1) 2 байта; 2) 3 байта; 3) 6 байт; 4) 12 байт.

24. В сегментном регистре под индекс таблицы сегментных дескрипторов отводится:
1) 4 двоичных разряда; 2) 8 двоичных разрядов;
3) 14 двоичных разрядов; 4) 24 двоичных разряда.

25. Полный логический адрес адресуемой ячейки состоит:
1) из 14-разрядного индекса сегмента и 16-разрядного относительного адреса;
2) из 16-разрядного индекса сегмента и 14-разрядного относительного адреса;
3) из 24-разрядного индекса сегмента и 14-разрядного относительного адреса;
4) из 14-разрядного индекса сегмента и 24-разрядного относительного адреса.

26. Минимальной адресной единицей является:
1) бит; 2) байт; 3) слово; 4) двойное слово.

27. В методе косвенной адресации адрес памяти содержится:
1) в одном из регистров; 2) в команде;
3) в стеке; 4) в ссылке на команду.

28. Если адрес находится в самой команде, то мы имеем дело:
1) с косвенной адресацией; 2) с основной адресацией;
3) с прямой адресацией; 4) с двойной косвенной адресацией.

29. При индексном доступе памяти адрес равен:
1) базовому адресу;
2) базовый адрес * смещение;
3) базовый адрес + смещение;
4) базовый адрес * K, где K — размер страницы.

30. Базовый адрес является:
1) начальной точкой массива данных; 2) конечной точкой массива данных; 3) промежуточной точкой массива данных; 4) массивом данных.
31. Наиболее широко используется:
1) косвенная адресация; 2) прямая адресация; 3) сегментный способ; 4) двойная косвенная адресация.
32. Стек — это:
1) неявный способ адресации, при котором информация записывается и считывается только последовательным образом;
2) способ адресации, при котором информация записывается и считывается по принципу очереди;
3) неявный способ адресации, в котором информация записывается по принципу иерархий;
4) способ адресации, при котором информация записывается по старшинству;
33. 8-битовые целые числа без знака лежат в диапазоне:
1) от 0 до 65535; 2) от 0 до 255; 3) от −128 до +127; 4) от −32768 до + 32767.
34. 8-битовые целые числа со знаком лежат в диапазоне:
1) от 0 до 65535; 2) от 0 до 255; 3) от −128 до +127 ; 4) от −32768 до + 32767.
35. 16-битовые целые числа со знаком лежат в диапазоне:
1) от 0 до 65535; 2) от 0 до 255; 3) от −128 до +127; 4) от −32768 до + 32767.
36. 16-битовые целые числа без знака лежат в диапазоне:
1) от 0 до 65535; 2) от 0 до 255; 3) от −128 до +127; 4) от −32768 до + 32767.
37. 8-битовые целые числа без знака занимают:
1) 1 байт и воспринимаются как положительное число;
2) 2 байта и воспринимаются как положительное число;
3) 2 байта и воспринимаются как отрицательное число;
4) 4 байта и воспринимаются как положительное число.
38. Прерывания — это:
1) события, возбуждаемые программами;
2) события, вызванные аппаратным сбоем;
3) события, которые делают дальнейшую работу невозможной или требуют специальной реакции;
4) события, вызванные логическими операциями.
39. Запрет прерывания называется:
1) откатом; 2) маскировкой; 3) указанием процессора; 4) «stoper».
40. Для запоминания состояния прерванной программы используется:
1) очередь; 2) стек; 3) ОЗУ; 4) ПЗУ.
41. Способ решения задачи приема и передачи данных решается следующим образом:
1) устройства ввода-вывода имеют собственное адресное пространство или включаются в общее адресное пространство;
2) для устройств ввода-вывода процессором выделяется свободное адресное пространство памяти;
3) для устройства ввода-вывода зарезервировано специальное адресное пространство;
4) устройства ввода-вывода используют совместную выделяемую процессором память постранично, с использованием двухступенчатой системы контроля со стороны процессора.

42. Размер страницы памяти равен:
1) 16 Кбайт; 2) 28 Кбайт; 3) 64 Кбайта; 4) 256 Кбайт.
43. Обмен с подключенными к ЭВМ печатающими устройствами производится через:
1) порт принтера; 2) порт состояния и порт данных;
3) LPT-порт; 4) COM-порт.
44. Буфер служит:
1) для записи информации о происшедшем прерывании;
2) для хранения информации о состоянии клавиатуры;
3) буферизации потока ввода-вывода данных;
4) для создания копии аккумулятора в процессе обработки команды процесса.
45. Команда `INC A`:
1) увеличивает аккумулятор на 1;
2) увеличивает содержимое аккумулятора на 1;
3) очищает аккумулятор;
4) вызывает прерывание.
46. Команда `MOV`:
1) сравнивает аккумулятор с регистром SP;
2) очищает аккумулятор;
3) увеличивает аккумулятор на 1;
4) загружает один регистр значением другого.
47. Команда `CMP A, B`:
1) загружает в аккумулятор содержимое B;
2) сравнивает содержимое аккумулятора с содержимым регистра;
3) если A = 0, то вызывает процедуру по адресу B;
4) выполняет побитовую операцию `XOR` над содержимым регистров A, B, C последующим сдвигом во флаг состояний.
48. В процессоре Intel команда `XOR AX`:
1) зануляет содержимое аккумулятора; 2) вызывает процедуру AX;
3) помещает AX в стек; 4) перезагружает компьютер.
49. В процессоре Intel команда `MOV DS,AX`:
1) записывает в `DS` содержимое `AX`;
2) записывает в `AX` содержимое `DS`;
3) сравнивает `DS` и `AX`;
4) отнимает от `DS AX` и результат помещает в аккумулятор.

Учебный микрокомпьютер «E97»

1. «E97» является:
1) 8-разрядным; 2) 16-разрядным;
3) 24-разрядным; 4) 32-разрядным.
2. Видеопамять в «E97» размещается:
1) в ОЗУ; 2) в видеокарте;
3) в контроллере дисплея; 4) в контроллере видеокарт.
3. Слово в «E97» состоит:
1) из 1 байта; 2) из 2 байтов;
3) из 3 байтов; 4) из 4 байтов.
4. В «E97» каждому внешнему устройству соответствуют:
1) 2 порта ввода-вывода; 2) 1 порт ввода;
3) 2 порта ввода и 1 порт вывода; 4) 2 пары портов ввода и вывода.

5. «E97» состоит:
   1) из четырех регистров общего назначения PC, SP, PS;
   2) из шести регистров общего назначения PC, PS;
   3) из четырех регистров общего назначения SP, PC;
   4) из шести регистров общего назначения PC, PS, SP.
6. В регистре состояния процессора «E97» используется:
   1) весь регистр; 2) 4 младших бита;
   3) 3 младших бита; 4) 2 младших бита.
7. В наиболее полной форме команда «E97» состоит:
   1) из двух частей по 4 бита каждая;
   2) из четырех частей по 4 бита каждая;
   3) из четырех частей по 2 бита каждая;
   4) из четырех частей по 8 бит каждая.
8. В «E97» команды переходов бывают:
   1) относительные и смешанные;
   2) абсолютные и смешанные;
   3) абсолютные и относительные;
   4) абсолютные, относительные и смешанные.
9. Модификатор команд в «E97» показывает:
   1) дополнительные условия перехода;
   2) адрес подпрограммы;
   3) по какому условию осуществляется переход;
   4) куда осуществляется переход.
10. В «E97» условный переход осуществляется:
   1) если условие справедливо;
   2) если условие несправедливо;
   3) если условие справедливо и разрешен переход;
   4) если условие справедливо, разрешен переход и известен адрес подпрограммы.
11. В «E97» сдвиг влево эквивалентен:
   1) делению на 2; 2) делению на 16;
   3) умножению на 2; 4) умножению на 16.
12. В «E97» существует:
   1) 1 метод адресации по PC; 2) 2 метода адресации по PC;
   3) 3 метода адресации по PC; 4) 4 метода адресации по PC.
13. В «E97» для сохранения точки возврата из подпрограммы используется:
   1) очередь; 2) стек;
   3) PC; 4) SP.
14. В конце любой подпрограммы в «E97» должна стоять команда:
   1) 0A00; 2) 0B00;
   3) 0C00; 4) 0E00.
15. В «E97» команда с кодом E1 выполняет:
   1) над операндом ОП1 логическую операцию И;
   2) над операндом ОП1 логическую операцию ИЛИ;
   3) над операндом ОП1 логическую операцию НЕ;
   4) над операндом ОП1 логическую операцию ИСКЛЮЧАЮЩЕЕ ИЛИ.
16. В «E97» команды с кодом E2—E9 обеспечивают:
   1) работу со стеком;
   2) работу с внутренней памятью регистра;
   3) работу с очередью;

4) работу с SP.

17. В «Е97» команды Е4 и Е5 позволяют:
1) изменять значение PS на величину ОП2;
2) изменять значение SP на величину ОП1;
3) изменять значение SP на величину ОП2;
4) изменять значение PS на величину ОП1.

18. В «Е97» команды Е6 и Е7 задают:
1) новое значение в SP, читают его текущее значение в ОП1;
2) новое значение в SP, читают его текущее значение в ОП2;
3) новое значение в PS, читают его текущее значение в ОП1;
4) новое значение в PS, читают его текущее значение в ОП2.

19. В «Е97» команды Е8, Е9 сохраняют:
1) в очереди и восстанавливают для последующего анализа регистра состояния процессора PS;
2) в стеке и восстанавливают для последующего анализа регистра состояния процессора PS;
3) в очереди и восстанавливают для последующего анализа регистра состояния процессора SP;
4) в очереди и восстанавливают для последующего анализа регистра состояния процессора SP.

20. В «Е97» под кодирование каждого операнда отводится:
1) 1 двоичный разряд; 2) 2 двоичных разряда;
3) 3 двоичных разряда; 4) 4 двоичных разряда.

21. Модификатор в «Е97» состоит:
1) из 2 бит; 2) из 4 бит;
3) из 8 бит; 4) из 16 бит.

22. В «Е97» если операнд является содержимым указанного регистра, то это:
1) регистровый метод адресации; 2) резерв;
3) метод косвенной адресации; 4) адресация по PC.

23. В «Е97» если операндом является содержимое ячейки ОЗУ, то это:
1) регистровый метод адресации; 2) резерв;
3) метод косвенной адресации; 4) адресация по PC.

24. В «Е97» регистр состояния имеет:
1) 1 бит; 2) 2 бита; 3) 4 бита; 4) 8 бит.

25. В «Е97» реальная информация в порте ввода-вывода располагается:
1) в младшем байте; 2) в младшем байте со смещением;
3) в старшем байте; 4) в старшем байте со смещением.

26. В «Е97» порту состояния клавиатуры соответствует номер:
1) 0; 2) 1; 3) 2; 4) 3.

27. В «Е97» порту данных клавиатуры соответствует номер:
1) 0; 2) 1; 3) 2; 4) 3.

28. В «Е97» порту состояния дисплея соответствует адрес:
1) 0; 2) 1; 3) 2; 4) 3.

29. В «Е97» порту данных дисплея соответствует адрес:
1) 0; 2) 1; 3) 2; 4) 3.

Внешние устройства ЭВМ

1. Информация на дискету наносится вдоль:
1) дорожек; 2) секторов; 3) кластеров; 4) цилиндров.

2. Каждая дорожка разбита:
1) на модули памяти; 2) на сектора; 3) на кластеры; 4) на цилиндры.
3. Стандартная емкость сектора:
1) 256 байт; 2) 512 байт; 3) 1024 байта; 4) 2048 байт.
4. Информационная емкость НГМД:
1) от 1 до 1,5 Мбайт; 2) от 1 до 2 Мбайт;
3) от 1 до 2,5 Мбайт; 4) от 1 до 3 Мбайт.
5. Доступ к информации на НГМД осуществляется:
1) за 0,01 — 1 с; 2) за 0,1 — 1 с; 3) за 1 — 2 с; 4) за 1,5 — 2,5 с.
6. Емкость жесткого диска:
1) от 5 до 50 Мбайт; 2) от 10 до 95 Мбайт;
3) от 10 Мбайт до 1 Гбайт; 4) от 20 Мбайт до 10 Гбайт.
7. Время доступа к нужной записи на жестком диске:
1) 1 — 10 мс; 2) 1 — 25 мс; 3) 1 — 50 мс; 4) 1 — 100 мс.
8. Процедура разметки нового диска называется:
1) архивацией; 2) компиляцией;
3) форматированием; 4) дефрагментацией.
9. Информация на оптических дисках наносится посредством:
1) изменения магнитного уровня; 2) изменения физической структуры;
3) изменения рельефа; 4) изменения химической структуры.
10. Чтение с оптического диска происходит с помощью:
1) лазерного луча; 2) магнитной головки;
3) мини-сканера; 4) системы магнитно-оптических контроллеров.
11. Специальный кассетный накопитель:
1) драйвер; 2) НОД; 3) стриммер; 4) плоттер.
12. Группа клавиш клавиатуры, относящихся к служебным, имеет назначение:
1) перенацеливающие действия остальных;
2) для ввода букв, цифр и других знаков;
3) назначение задается разработчиком программы;
4) для формирования новых шрифтов.
13. Важным свойством клавиатуры является:
1) экономичность; 2) эргономичность;
3) легитимность; 4) функциональность.
14. Устройство для ввода с листа бумаги документов называется:
1) драйвер; 2) плоттер; 3) стриммер; 4) сканер.
15. Всякую информацию сканер воспринимает:
1) как линейную; 2) как асинхронную;
3) как текстовую; 4) как графическую.
16. Монохромный дисплей производит отображение:
1) в двух цветах; 2) в трех цветах;
3) в четырех цветах; 4) в 8-битовом разрешении.
17. Наиболее прогрессивным способом представления графической информации является:
1) векторный; 2) синхронный; 3) растровый; 4) асинхронный.
18. Печатающее устройство называется:
1) плоттер; 2) принтер; 3) стриммер; 4) дигитайзер.
19. В основе функционирования точечно-матричного принтера лежит использование:
1) печатающих игл;
2) головки со специальной краской и микросоплом;

3) лазера;
4) красящих пузырьков.

20. В основе струйного принтера лежит использование:
1) печатающих игл;
2) головки со специальной краской и микросоплом;
3) лазера;
4) красящих пузырьков.

21. В основе лазерного принтера лежит использование:
1) печатающих игл;
2) головки со специальной краской и микросоплом;
3) лазера;
4) красящих пузырьков.

22. Устройство для вывода на бумагу чертежей и рисунков называется:
1) принтером; 2) плоттером;
3) сканером; 4) стриммером.

Логические основы функционирования ЭВМ

1. Высказывание — это:
1) отношение между формулами;
2) всякая выводимая формула;
3) любое утверждение, относительно которого можно сказать, истинно оно или ложно;
4) всякое формулированное утверждение, относительно которого можно сказать, что оно ложно.

2. Логическое И называется:
1) конъюнкцией; 2) дизъюнкцией;
3) логической разностью; 4) дополнением.

3. Логическое ИЛИ называется:
1) конъюнкцией; 2) дизъюнкцией;
3) логической разностью; 4) дополнением.

4. Операция И имеет результат «истина», если:
1) оба операнда истинны; 2) оба операнда ложны;
3) хотя бы один истинный; 4) хотя бы один ложный.

5. Операция ИЛИ имеет результат «истина», если:
1) оба операнда истинны; 2) оба операнда ложны;
3) хотя бы один истинный; 4) хотя бы один ложный.

6. Установка — это:
1) запись в операционный элемент двоичного кода;
2) перезапись кода из одного элемента в другой;
3) изменение положения кода относительно исходного;
4) перекодирование.

7. Прием-передача — это:
1) запись в операционный элемент двоичного кода;
2) перезапись кода из одного элемента в другой;
3) изменение положения кода относительно исходного;
4) перекодирование.

8. Сдвиг — это:
1) запись в операционный элемент двоичного кода;
2) перезапись кода из одного элемента в другой;

3) изменение положения кода относительно исходного;
4) перекодирование.

9. Преобразование — это:
1) запись в операционный элемент двоичного кода;
2) перезапись кода из одного элемента в другой;
3) изменение положения кода относительно исходного;
4) перекодирование.

10. Триггер — это:
1) устройство для сложения чисел;
2) устройство для хранения информации;
3) устройство для передачи данных;
4) основа устройства оперативного хранения информации.

11. Сумматор — это:
1) устройство для сложения чисел;
2) устройство для хранения информации;
3) устройство для передачи данных;
4) основа устройства оперативного хранения информации.

12. Триггер имеет:
1) 2 входа;
2) 2 входа и 2 выхода;
3) 1 вход и 2 выхода;
4) 2 входа и 1 выход.

13. Что делает триггер при отсутствии входных сигналов:
1) сбрасывается в 0;
2) устанавливается в 1;
3) сохраняет свое предыдущее состояние;
4) меняется по падающему фронту?

14. В основе схемы реализации логического элемента И–НЕ лежат:
1) *n-p-n* резисторы; 2) *n-p-n* транзисторы;
3) *p-n-p* резисторы; 4) *p-n-p* транзисторы.

Правильные ответы

История развития вычислительной техники

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1	X				12	X				23	X			
2				X	13		X			24		X		
3		X			14				X	25	X			
4	X				15			X		26	X			
5				X	16			X		27		X		
6	X				17	X				28	X			
7				X	18				X	29				X
8			X		19		X			30		X		
9		X			20		X			31				X
10		X			21			X		32		X		
11			X		22		X							

Архитектура ЭВМ

№	1	2	3	4	№	1	2	3	4
1	X				13	X			
2	X				14		X		
3	X				15			X	
4			X		16				X
5			X		17	X			
6		X			18				X
7				X	19	X			
8			X		20				X
9				X	21		X		
10	X				22	X			
11				X	23				X
12			X		24				X

Архитектура микропроцессора

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1		X			18	X				35				X
2			X		19			X		36	X			
3	X				20			X		37	X			
4		X			21				X	38			X	
5				X	22				X	39		X		
6	X				23			X		40		X		
7			X		24			X		41	X			
8			X		25	X				42	X			
9	X				26		X			43		X		
10				X	27	X				44	X			
11	X				28			X		45		X		
12	X				29			X		46				X
13				X	30	X				47		X		
14	X				31			X		48	X			
15	X				32	X				49	X			
16	X				33		X							
17			X		34			X						

Учебный микрокомпьютер «Е97»

№	1	2	3	4	№	1	2	3	4
1		X			16	X			
2			X		17		X		
3		X			18	X			
4	X				19		X		
5	X				20				X
6				X	21		X		
7		X			22	X			
8			X		23			X	
9			X		24	X			
10	X				25	X			
11			X		26	X			
12		X			27		X		
13		X			28			X	
14			X		29				X
15			X						

Внешние устройства ЭВМ

№	1	2	3	4	№	1	2	3	4
1	X				12	X			
2		X			13			X	
3		X			14				X
4	X				15				X
5		X			16	X			
6				X	17			X	
7				X	18		X		
8			X		19	X			
9			X		20		X		
10	X				21			X	
11			X		22		X		

Логические основы функционирования ЭВМ

№	1	2	3	4	№	1	2	3	4
1			X		8			X	
2	X				9				X
3		X			10				X
4	X				11	X			
5			X		12		X		
6	X				13			X	
7		X			14		X		

Глава 5

КОМПЬЮТЕРНЫЕ СЕТИ И ТЕЛЕКОММУНИКАЦИИ

В данном разделе практикума осваивается работа в локальных сетях и в Internet. Поскольку компьютерные сети становятся (если уже не стали) общедоступными и во многом именно они обеспечивают информационную революцию, практическое освоение навыков работы с ними совершенно необходимо.

При разработке практикума по этой теме основной проблемой является то, что она отражает самую динамичную сферу информатики.

Программные средства поддержки сетей меняются очень быстро, и излишняя конкретизация указаний по работе с ними может быстро сделать руководство неактуальным.

Рекомендации по организации практических занятий приведены с учетом этого обстоятельства.

Кроме того, учтено, что следующая глава практикума, посвященная информационным системам, также активно использует Internet и способствует отработке соответствующих практических навыков.

§ 1. ЛОКАЛЬНЫЕ СЕТИ

Рекомендации по проведению занятий

Тема локальных сетей оказывается одной из важнейших тем по архитектуре вычислительных систем и операционным системам в связи с быстрым распространением и развитием корпоративных сетей компьютеров.

В учебном процессе вуза имеется возможность, как правило, проиллюстрировать локальные сети персональных компьютеров, приемы их построения, сетевое оборудование, приемы администрирования сетей, коллективное использование ресурсов, групповую работу над проектами. Все эти элементы сетевых технологий могут быть изучены уже на простейшей одноранговой сети на основе Windows 95/98.

При освоении данной темы весьма полезны не только семинарские занятия, подготовка рефератов, но и выполнение лабораторных работ. Лабораторные работы могут выполняться в реальном работающем учебном классе персональных компьютеров, соединенных в локальную сеть, однако при их постановке приходится преодолевать сопротивление учебно-вспомогательного персонала компьютерного класса, особенно при неудачном выполнении задания, требующем восстановления работоспособности сети.

Контрольные вопросы

1. Что называется локальной сетью компьютеров?
2. Каковы причины создания локальных сетей? Для чего они создаются?
3. Какие схемы соединения компьютеров в локальную сеть существуют?
4. Какие сети называются одноранговыми? Что такое сервер локальной сети?
5. Какие кабели используются для соединения компьютеров в локальную сеть? Охарактеризуйте условия применения таких соединений.
6. Какие виды электронного оборудования используются для создания локальных сетей?
7. Какое программное обеспечение требуется для создания и работы в локальной сети?
8. Охарактеризуйте основные операционные системы, используемые для создания локальных сетей.
9. Что называется администрированием локальной сети?
10. Каким образом происходит идентификация пользователя локальной сети? Для чего она нужна?
11. Что называется рабочей группой?
12. Какие сетевые приложения называют клиент-серверными?

Темы для рефератов

1. Развитие технологий соединения компьютеров в локальные сети.
2. Развитие операционных систем для локальных сетей.
3. Сетевые приложения клиент-серверной архитектуры.
4. Защита информации и администрирование в локальных сетях.

Темы семинарских занятий

1. Условия создания и архитектура локальных сетей компьютеров.
2. Кабельное хозяйство и аппаратное обеспечение локальных сетей.
3. Программное обеспечение локальных сетей.
4. Администрирование локальных сетей.

Рекомендации по программному обеспечению

Несмотря на сравнительно небольшое количество распространенных в России операционных систем для организации локальных сетей — а первенство здесь держат (по состоянию на конец 2000 г.) для простейших одноранговых сетей Windows 95/98, а для сетей с выделенным сервером — Novell Netware (версий 4 и 5) и Windows NT 4.0, — при изучении общего курса информатики нет возможности подробно изучать сетевые возможности всех этих операционных систем, а также операционных систем UNIX, LINUX и т.п. Сам выбор сетевой операционной системы в лаборатории, как правило, определяется вкусами и подготовкой сетевого администратора учебного класса или наличием лицензионного программного обеспечения для компьютерного класса. Переустановка, переконфигурирование сетевой операционной системы, как правило, оказывается трудоемкой работой, ме-

шающей параллельно ведущемуся в компьютерном классе учебному процессу по другим темам. Поэтому общие рекомендации, которые могут быть даны по изучению локальных сетей, — изучение сетевых возможностей, приемов администрирования и настройки имеющейся в учебном классе операционной системы.

Наиболее простым вариантом знакомства с локальной сетью является одноранговая сеть на основе Windows 95/98. На ее примере могут быть проиллюстрированы установка сетевой платы и подключение компьютера в сеть, администрирование общих ресурсов компьютеров, использование общего принтера. Могут быть проиллюстрированы и приемы групповой работы в сети, например, сетевая дискуссия с помощью программного средства Microsoft NetMeeting, электронный документооборот и электронная почта (при помощи *proxy*-сервера или программы Winproxy).

Задачи и упражнения

1. Зарегистрируйтесь в локальной сети. Имя и пароль для входа в сеть узнайте у администратора сети. Узнайте также имена сетевых дисков и доступных разделов на них. Проверьте сетевое окружение вашего компьютера.

2. Создайте файл PROBA.TXT, содержащий ваши имя и фамилию. Сохраните его в доступном разделе сетевого диска. Загрузите его с сетевого диска.

3. Выведите файл PROBA.TXT на печать на сетевом принтере.

4. Разместите файл PROBA.TXT в папке EXCHANGE (предварительно созданной) на локальном диске вашего компьютера. Разрешите сетевой доступ к этому разделу: а) только для чтения; б) для чтения и записи. Проверьте с других компьютеров в сети читаемость и изменяемость файла PROBA.TXT в разделе EXCHANGE.

5. Установите и сконфигурируйте для работы в локальной сети программу Microsoft NetMeeting. Свяжитесь с несколькими компьютерами и поработайте в чате, аудио, или видеоконференции, перейдите в режим общего рабочего стола, продемонстрируйте в локальной сети содержимое файла PROBA.TXT.

Лабораторные работы

Для освоения темы «Локальные сети» можно рекомендовать выполнение трех лабораторных работ.

1. Аппаратное обеспечение локальных сетей, кабельное хозяйство.

Данная работа включает ознакомление с видами кабелей, используемых для создания локальных сетей (ознакомление с последующим определением типа кабеля), с разъемами, коннекторами, сетевыми адаптерами, а также другим сетевым оборудованием (при его наличии) — с хабами, репитерами, маршрутизаторами.

2. Основные возможности операционных систем для локальных сетей.

Лабораторная работа предполагает установку драйвера сетевого адаптера компьютера, установку и настройку IPX- и IP-протоколов. В качестве контрольного задания следует выполнить задачи 1 — 4.

3. Групповая работа в локальных сетях.

Лабораторная работа предполагает установку и общение в локальной сети с помощью программы NetMeeting, а также с использованием электронной почты.

В качестве контрольного задания может быть выполнено упражнение № 5.

Дополнительная литература

1. *Веттиг Д.* Novell Netware: Пер. с англ. — Киев: BHV; М: Бином, 1994.
2. *Кульгин М.* Технологии корпоративных сетей. Энциклопедия. — СПб.: Питер, 1999.
3. *Нанс Бэрри.* Компьютерные сети: Пер. с англ. — М.: Бином, 1996.
4. Сетевые средства Windows NT. Windows NT Workstation и Windows NT Server версия 3.5. — СПб.: BHV—Санкт-Петербург, 1996.

§ 2. ГЛОБАЛЬНЫЕ СЕТИ

Рекомендации по проведению занятий

Глобальные компьютерные сети — условное название темы. В действительности глобальные сети, возникшие независимо друг от друга когда-то, давно объединились в одну всемирную сеть сетей — Internet. Значение этой сети исключительно велико — она заставляет взглянуть на компьютер по-новому, не как на средство переработки информации, а как на средство связи между людьми, находящимися в разных географических точках.

Internet вызывает повышенный интерес у молодежи, поэтому надо ожидать, что студенты будут иметь определенные навыки работы в сети Internet и даже создания ее ресурсов. Задача данного параграфа практикума — систематизировать имеющиеся знания и упрочить практические навыки. При этом важно помнить, что нельзя раз и навсегда «научиться Internet». Это очень подвижная, быстро меняющаяся информационно-технологическая среда. Здесь как никогда важны умения самостоятельно находить нужную информацию, осваивать способы управления информационными системами, творчески подходить к решению разнообразных задач.

При работе над данной частью практикума полезны не только семинарские занятия и лабораторные работы, но и самостоятельная работа над контрольными вопросами, а также над рефератами.

Контрольные вопросы

1. Какие сети называют глобальными?
2. Какова структура сети Internet?
3. Что такое протокол? Какова роль стандартизации протоколов для создания сети Internet?
4. Какова транспортная основа Internet? Какие каналы связи он использует?
5. Какие в настоящее время существуют способы связи пользователя с Internet?
6. Что означает аббревиатура TCP/IP? Какой механизм передачи пактов предполагает этот протокол?
7. Какие виды сервиса Internet предоставляет?
8. Что такое E-mail? Телеконференции USENET? FTP? WWW?
9. Какова структура Internet-адреса в числовой форме? в доменной форме?
10. Какова структура IP-пакета?
11. Какие протоколы используются при обмене электронными письмами?
12. Какова структура электронного письма?

13. Какова структура электронного адреса?
14. Охарактеризуйте известные клиентские программы электронной почты.
15. Как используется ftp-протокол в сети Internet?
16. Приведите примеры ftp-серверов.
17. Какие клиентские программы для работы по протоколу ftp Вы знаете?
18. Что такое WWW (World Wide Web)?
19. Что называется броузером (программой-просмотрщиком)? Охарактеризуйте наиболее распространенные просмотрщики.
20. Охарактеризуйте протокол HTTP.
21. Что такое HTML? Приведите примеры основных тегов HTML. Как определяется гипертекстовая ссылка с помощью HTML?
22. Какие графические форматы используются при оформлении Web-страниц?
23. Охарактеризуйте распространенные средства разработки Web-страниц.
24. Как обеспечивается интерактивное взаимодействие пользователя с Web-сайтом?
25. Приведите примеры поисковых Web-сайтов.
26. Приведите примеры поисковых запросов, содержащих конъюнкцию и дизъюнкцию ключевых фраз.
27. Охарактеризуйте новые виды сервиса Internet: ICQ, IP-телефония.

Темы для рефератов

1. История формирования всемирной сети Internet. Современная статистика Internet.
2. Структура Internet. Руководящие органы и стандарты Internet.
3. Каналы связи и способы доступа в Internet.
4. Модемы и протоколы обмена.
5. Оборудование и цифровые технологии доступа в Internet.
6. Программное обеспечение сети Internet: операционные системы серверов.
7. Программное обеспечение сети Internet: операционные системы серверов.
8. Программное обеспечение сети Internet: серверное программное обеспечение.
9. Протоколы и сервисы сети Internet.
10. Развитие стандартов кодирования сообщений электронной почты.
11. Телеконференции системы Usenet.
12. Клиентские программы для работы с электронной почтой. Особенности их использования и конфигурирования.
13. Клиентские программы для просмотра Web-страниц, их конфигурирование.
14. Основы HTML и его развитие.
15. Интерактивные элементы Web-страниц и скрипты.
16. Графические форматы при оформлении Web-страниц.
17. Средства разработки Web-страниц.
18. Элементы Web-дизайна.
19. Поисковые сайты и технологии поиска информации в Internet.
20. Образовательные ресурсы сети Internet.
21. Досуговые ресурсы сети Internet.
22. Новые виды сервиса Internet — ICQ, IP-телефония, видеоконференция.
23. Электронная коммерция и реклама в сети Internet.
24. Проблемы защиты информации в Internet.
25. Авторское право и Internet.

Темы семинарских занятий

1. Архитектура сети Internet. Каналы связи и технологии доступа в Internet.
2. Руководящие органы и стандарты сети Internet. IP-адресация.
3. Протоколы и сервисы сети Internet.
4. Программное обеспечение сети Internet: операционные системы, серверное программное обеспечение.
5. Электронная почта — структура и кодировка сообщений, клиентское программное обеспечение. Телеконференции.
6. Принципы WWW. Броузер, HTML, просмотр Web-страниц.
7. Web-технологии и создание Web-ресурсов.
8. Технологии поиска информации в Internet. Образовательные и досуговые ресурсы.
9. Новые виды сервиса Internet — ICQ, видеоконференции, IP-телефония.
10. Вопросы использования и регулирования Internet. Защита информации.

Рекомендации по программному обеспечению

Для изучения данной темы в минимально необходимом объеме достаточно стандартного программного обеспечения, имеющегося на компьютерах, используемых для работы в сети Internet. Это операционная система **Windows 95/98** или **NT4.0 Workstation**, стандартная клиентская программа электронной почты **Outlook Express** и броузер (программа-просмотрщик) **Internet Explorer**. Полезными являются также широко распространенные почтовая система **The bat**, комплекс **Netscape Communicator**. При изучении вопросов, посвященных разработке Web-страниц, необходимым является графический **редактор Adobe Photoshop** (или **Corel Draw**), средства разработки Web-страниц **Front Page** (или **Macromedia Dreamweaver**). Данные программы легко доступны и не представляют большой трудности в их установке и конфигурировании.

При более продвинутом варианте курса требуется иметь в распоряжении серверы на основе **Windows NT Server** или **Linux (UNIX Free BSD)**, комплекс серверных программ электронной почты, FTP, WWW. Данные системы также легко доступны, однако они представляют значительные трудности в установке и конфигурировании, требуют высокой квалификации учебно-вспомогательного персонала, отвечающего за сопровождение компьютеров и программного обеспечения учебного процесса.

Задачи и упражнения

1. Проверьте настройки сетевых протоколов вашего компьютера. Для этого, работая в среде Windows, откройте панель управления, приложение «Сеть». С помощью какого адаптера ваш компьютер подключен к Internet? Каковы настройки протокола IP — как установлен IP-адрес вашего компьютера, маска подсети, DNS-сервер?
2. Проверьте параметры обозревателя Internet. Для этого откройте панель управления, свойства обозревателя. Какое используется соединение? Каковы настройки локальной сети? Каковы общие настройки обозревателя? Какие программы используются для работы в Internet? Каков уровень безопасности?

3. Установите и сконфигурируйте почтовую клиентскую программу **Outlook Express** на рабочей станции. Параметры для входа на POP-сервер (имя, электронный адрес, пароль) узнайте у системного администратора сети.

4. Создайте тестовое сообщение электронной почты, пошлите его по известному вам адресу (студенту вашей группы). Ответьте на тестовое сообщение, пришедшее на ваш адрес.

5. Сохраните пришедшее письмо в отдельной папке почтовой системы. Напечатайте его. Удалите. Очистите папку для мусора.

6. Просмотрите список доступных телеконференций. Подпишитесь на тестовую телеконференцию. Пошлите в нее тестовое сообщение. Подпишитесь на образовательную телеконференцию (например, relcom.education). Просмотрите сообщения в ней.

7. С помощью клиентской программы (**CuteFTP** или **FAR**) подключитесь к FTP-серверу. Определите, какие файлы можно получить с этого сервера. Получите файл наименьшего размера.

8. Соединитесь с помощью броузера **Intrenet Explorer** или **Netscape Navigator** с официальным Web-сервером Министерства образования http://www.informika.ru. Ознакомьтесь с его ресурсами. Какие последние нормативные акты по управлению образованием выпустило министерство?

9. Выполните поиск на сайте http://www.informika.ru информации, относящейся к вашему вузу.

10. Выполните поиск в Internet виртуальных электронных магазинов. Проверьте, имеется ли в них в продаже данная книга. Сделайте заказ. Оставьте запись в гостевой книге.

11. Примите участие в чате по адресу http://chat.ru.

12. Соединитесь с поисковым сайтом http://yandex.ru. Сформируйте поисковый запрос для методических материалов по информатике. Выполните поиск. Ознакомьтесь с найденными страницами. Можно ли конкретизировать запрос? Выполните поиск с уточненным запросом.

13. Создайте простую Web-страницу, содержащую информацию о вас, пользуясь языком HTML. Просмотрите ее с помощью имеющегося броузера.

14. Создайте собственную визитную карточку для Internet, состоящую как минимум из трех страниц, связанных гипертекстовыми ссылками. Просмотрите ее.

15. Установите и сконфигурируйте программу **ICQ**. Зарегистрируйтесь. «Поболтайте» в сети с кем-нибудь.

Лабораторные работы

Решению предложенных выше задач полезно придать форму лабораторных работ. При этом их можно структурировать в 5 лабораторных работ.

1. Протоколы и настройки обозревателей рабочей станции для работы в Internet (предполагает решение задач 1, 2).

2. Работа с электронной почтой и телеконференциями (предполагает решение задач 3 — 6).

3. FTP- и WWW-сервисы сети Internet (предполагает решение задач 7 — 9).

4. Поиск информации в Internet. Работа с интерактивными элементами Web-страниц (предполагает решение задач 10, 12).

5. Создание Web-страниц (предполагает решение задач 13 — 14).

Дополнительная литература

1. Internet. Всемирная компьютерная сеть. Практическое пособие и путеводитель. — М.: Синтез, 1995
2. *Кирсанов Д.* Web-дизайн. — СПб.: Символ-Плюс, 1999.
3. *Крол Эд.* Все от INTERNET. Руководство и каталог: Пер. с англ. — Киев: BHV, 1995.
4. *Нанс Бэрри.* Компьютерные сети: Пер. с англ. — М.: Бином, 1996.
5. *Храмцов П.* Лабиринты Internet. — М.: Электроинформ, 1996.
6. Ресурсы сайта http://www.citforum.ru

Тесты к главе 5

Локальные сети

1. Компьютерная сеть — это:
   1) группа компьютеров, размещенных в одном помещении;
   2) объединение нескольких ЭВМ для совместного решения задач;
   3) комплекс терминалов, подключенных каналами связи к большой ЭВМ;
   4) мультимедийный компьютер с принтером, модемом и факсом.
2. Сетевые технологии — это:
   1) основная характеристика компьютерных сетей;
   2) формы хранения информации;
   3) технологии обработки информации в компьютерных сетях;
   4) способ соединения компьютеров в сети.
3. Информационные системы — это:
   1) компьютерные сети;
   2) хранилище информации;
   3) системы, управляющие работой компьютера;
   4) системы хранения, обработки и передачи информации в специально организованной форме.
4. Локальная сеть — это:
   1) группа компьютеров в одном здании;
   2) комплекс объединенных компьютеров для совместного решения задач;
   3) слаботочные коммуникации;
   4) система Internet.
5. Что не характерно для локальной сети:
   1) большая скорость передачи информации;
   2) возможность обмена информацией на большие расстояния;
   3) наличие связующего для всех абонентов высокоскоростного канала для передачи информации в цифровом виде;
   4) наличие канала для передачи информации в графическом виде?
6. Какие линии связи используются для построения локальных сетей:
   1) только витая пара;
   2) только оптоволокно;
   3) только толстый и тонкий коаксильный кабель;
   4) витая пара, коаксильный кабель, оптоволокно и беспроводные линии связи?
7. Сетевой адаптер выполняет следующую функцию:
   1) реализует ту или иную стратегию доступа от одного компьютера к другому;

2) кодирует информацию;
3) распределяет информацию;
4) переводит информацию из числового вида в текстовый, и наоборот.

8. Типы сетевых адаптеров:
1) Arcnet, Internet; 2) SoundBlaster, Token Ring;
3) Ethernet, винчестер; 4) Arcnet, Token Ring, Ethernet.

9. Сервер — это:
1) один или несколько мощных компьютеров для обслуживания сети;
2) высокопроизводительный компьютер;
3) хранитель программы начальной загрузки;
4) мультимедийный компьютер с модемом.

10. Основная функция сервера:
1) выполняет специфические действия по запросам клиента;
2) кодирует информацию, предоставляемую клиентом;
3) хранит информацию;
4) пересылает информацию от клиента к клиенту.

11. Для передачи данных в сети используются основные схемы:
1) конкурентная и логическая;
2) конкурентная и с лексическим доступом;
3) конкурентная с маркерным доступом;
4) с маркерным доступом и с лексическим доступом?

12. Какую схему сеть Ethernet использует для передачи данных по сети:
1) с маркерным доступом; 2) конкурентную схему;
3) логическую схему; 4) с лексическим доступом.

13. Сеть Token Ring использует следующую схему:
1) логическую; 2) конкурентную;
3) с маркерным доступом; 4) с лексическим доступом?

14. По какой схеме ведется передача данных в сети Arcnet:
1) по логической; 2) с лексическим доступом;
3) с маркерным доступом; 4) по конкурентной?

15. Какие бывают конфигурации (топологии) ЛС:
1) древовидная, односвязная, полносвязная, параллельная;
2) шинная, односвязная, звездообразная, полносвязная;
3) кольцевая, шинная, звездообразная, полносвязная и древовидная;
4) древовидная, многосвязная, малокольцевая, последовательная?

16. Какие методы доступа от компьютера к компьютеру используются в ЛС:
1) маркерный метод, прямой доступ;
2) метод резервации времени, кодировочный метод;
3) прямой доступ, кодировочный метод;
4) маркерный метод, метод резервации времени?

17. Компоненты, участвующие в передачи данных по сети:
1) компьютер-источник, передатчик, кабельная сеть, приемник;
2) компьютер-источник, кабельная сеть, приемник и компьютер-адресат;
3) файл-сервер, блок проколов, кабельная сеть, компьютер-адресат;
4) компьютер-источник, блок протокола, передатчик, кабельная сеть, приемник и компьютер-адресат.

18. Протокол — это:
1) пакет данных;
2) правила организации передачи данных в сети;
3) правила хранения данных в сети;

4) структуризация данных в сети.

19. Специфические функции ЛС учебного назначения:
1) поддержка файловой системы, защита данных и разграничение доступа;
2) система контроля и ведения урока;
3) определение рабочей системы, декодирование данных, система контроля;
4) разграничение данных, защита данных, система доступа, определение рабочей системы, разграничение доступа, система контроля и ведения урока.

Операционные системы локальных сетей

1. Каково назначение операционных систем ЛС:
1) обучающие функции;
2) прикладная программа для клиента;
3) обеспечивает совместное использование аппаратных ресурсов сети и использование распределенных коллективных технологий при выполнении работ;
4) специальная компонента ЛС для настройки передачи данных по заданному протоколу?

2. ОС NetWare — это:
1) сетевая ОС с централизованным управлением;
2) сетевая ОС с демократическим принципом управления;
3) иерархическая ОС для одноранговой и многоранговой систем;
4) специфическая ОС для связи с Internet.

3. Что используется при запуске ОС NetWare:
1) текстовый файл; 2) ядро — файл server.exe;
3) nlm-модуль; 4) системный том SYS?

4. Каково назначение утилиты syscon.exe:
1) хранилище индивидуальных подкаталогов пользователей сети;
2) с ее помощью администратор системы выполняет всю работу по разграничению доступа пользователей;
3) заводит для каждого пользователя сети отдельный подкаталог;
4) содержит программу подключения пользователя к сети?

5. Возможности файловой системы NetWare:
1) обеспечивает прозрачный доступ к разделам диска файл-сервера;
2) поддерживает разветвленную систему разграничения доступа к файлам и каталогам файл-сервера с различных рабочих станций;
3) создает системный том SYS;
4) разделяет пользователей сети на группы.

6. В какой директории содержатся сетевые программы и утилиты для пользователя в NetWare:
1) SYSTEM; 2) USERS; 3) MAIL; 4) PUBLIC?

7. Каково назначение команды LOGIN (ОС NetWare):
1) пользователь подключается к файл-серверу;
2) отображает каталоги файл-сервера на локальные диски рабочей станции;
3) отключает от файл-сервера;
4) позволяет получить детальную информацию о файлах?

8. Какая команда производит отключение от файл-сервера (ОС NetWare):
1) `map`; 2) `Login`; 3) `Logout`; 4) `ndir`?

9. Для чего предназначена диалоговая утилита salvage (ОС NetWare):
1) для управления сервером;

2) позволяет посылать короткие сообщения с одной рабочей станции на другую;
3) для восстановления случайно удаленных файлов;
4) для просмотра информации о группе пользователей?

10. Какая утилита предназначена для управления сервером (ОС NetWare):
1) `session`; 2) `syscon`; 3) `send`; 4) `filer`?

Глобальные сети

1. В глобальных сетях существуют два режима информационного обмена — это:
1) пользовательский и сетевой; 2) информируемый и скрытый;
3) диалоговый и пользовательский; 4) диалоговый и пакетный.
2. On-line — это:
1) информационная сеть; 2) команда;
3) режим реального времени; 4) утилита.
3. Крупнейшая российская телекоммуникационная сеть:
1) BITNET; 2) APRANET; 3) NET; 4) RELCOM.
4. Of-line — это:
1) режим информационного пакетного обмена; 2) команда;
3) телекоммуникационная сеть; 4) операционная система.
5. Мировая система телеконференций:
1) Eunet; 2. Fidonet; 3. Relcom; 4. Usenet.
6. BBS — это:
1) компьютерная сеть; 2) система телеконференций;
3) электронная доска объявлений; 4) режим работы.
7. BBS предназначена:
1) для определения маршрута информации;
2) для обмена файлами между пользователями;
3) для просмотра адресов;
4) для управления информацией.
8. Хост-машина — это:
1) банк информации; 2) компьютерные узлы связи;
3) мультимедийный компьютер; 4) машина-хранилище информации.
9. Модем — это:
1) устройство преобразования цифровых сигналов в аналоговые, и наоборот;
2) транспортная основа сети;
3) хранилище информации;
4) устройство, которое управляет процессом передачи информации.
10. Функции модема:
1) соединяет компьютер с ближайшим узлом;
2) служит сетевой платой для соединения компьютеров в локальную сеть;
3) осуществляет протоколирование передающей информации;
4) защищает информацию.
11. Транспортная основа глобальных сетей — это:
1) витая пара; 2) коаксиальный кабель;
3) телефонные линии и спутниковые каналы; 4) телеграф.
12. Для связи компьютеров через модемы используются:
1) только телефонные линии;
2) только спутниковые каналы;
3) только радиоволны;

4) телефонные линии, оптоволокно, спутниковые каналы и радиоволны.

13. По способу общения различают следующие режимы передачи данных:
1) дуплексный и полудуплексный; 2) одновременный и поэтапный;
3) скоростной и одновременный; 4) дуплексный и одновременный.

14. По способу группирования данных различают режимы:
1) однозначную и одноблочную передачи;
2) многосложную и односложную передачи;
3) последовательную и параллельную;
4) синхронную и асинхронную.

15. Что такое MNP-модемы:
1) модемы с аппаратным сжатием и коррекцией информации;
2) модемы с кодированием информации;
3) модемы с защитой информации;
4) модифицированные по скоростям модемы?

16. Что является более важным для организации сети:
1) наличие большого количества компьютеров; 2) система протоколов;
3) несколько сетевых операционных систем; 4) высокоскоростные модемы?

17. Что обеспечивают протоколы сетевого уровня:
1) обеспечивают сетевые режимы передачи данных;
2) доступ к сетевым ресурсам;
3) соединяют различные сети;
4) тестируют работу в сети?

18. Транспортные протоколы выполняют следующие функции:
1) группируют сообщения;
2) кодируют пакеты информации;
3) отвечают за обмен между хост-машинами;
4) контролируют вход и выход данных.

19. За что отвечают прикладные протоколы:
1) за передачу данных и доступ к сетевым ресурсам;
2) формируют пакеты данных;
3) контролируют работу хост-машин;
4) тестируют правильность работы сети?

20. Маршрутизатор (роутер) — это:
1) мощные компьютеры, соединяющие сети или участки сети;
2) отслеживают путь от узла к узлу;
3) определяют адресатов сети;
4) программа маршрутизации пакетов данных.

21. Техническая структура E-mail — это:
1) совокупность узловых станций, связывающихся друг с другом для обмена;
2) совокупность компьютеров локальной сети;
3) компьютеры, хранящие и кодирующие информацию;
4) компьютеры, пересылающие информацию по запросам.

22. Типичная абонентская станция электронной почты состоит:
1) из нескольких сетевых компьютеров;
2) из компьютера, специальной программы и модема;
3) из компьютера и почтового сервера;
4) из хост-машин.

23. Типичная структура электронного письма:
1) заголовок, тема сообщения, ФИО адресата;
2) заголовок, тема сообщения, тип письма, адрес отправителя;

3) дата отправления, адрес, обратный адрес, тема сообщения и текст;
4) тема сообщения, адресная книга, текст и заголовок.

24. Домен — это:
1) название файла в почтовом ящике; 2) почтовый ящик узловой станции;
3) код страны; 4) короткое имя адресата.

25. Что является протокольной основой Internet:
1) система IP-адресов; 2) протоколы тестирования сетевого компьютера;
3) последовательность адресов; 4) адресная книга?

26. Из чего состоит IP-адрес:
1) адреса сети; 2) последовательности адресов;
3) протоколов; 4) адреса сети и номера хоста?

27. Какой протокол поддерживает Internet:
1) SCP/IP; 2) SCP; 3) TCP/IP; 4) QCP/IP?

28. Основные компоненты IP-технологии:
1) идентификация, длина IP-заголовка;
2) формат IP-пакета, IP-адрес, способ маршрутизации IP-пакетов;
3) формат ASCII и формат IP-адреса;
4) формат IP-пакета, способ общения на английском языке.

29. Что обеспечивает серверная программа DNS:
1) кодировку информации;
2) поиск числовых адресов;
3) устанавливает соответствие между доменными именами и IP-адресами;
4) занимается поиском IP-адресов?

30. Для чего используются программы Ping:
1) для трассировки пакетов;
2) для проверки прохождения IP-пакетов;
3) для идентификации повреждения пакета при передаче;
4) для определения IP-адреса?

31. Для поддержки E-mail в Internet разработан протокол:
1) STTP; 2) SMTP; 3) SCTP; 4) SSTP.

32. Какой стандарт кодировок используется в Internet:
1) UUCD; 2) MIME; 3) RFC-822; 4) WHOIS?

33. Кодирование писем применяется:
1) для ускорения передачи информации;
2) для передачи секретной информации;
3) для передачи бинарных файлов и некоторых текстовых;
4) исторические «правила игры» электронной почты.

34. Архив FTP — это:
1) сервер Archie; 2) хранилище файлов; 3) база данных; 4) WEB-сайт.

35. Начальная команда сеанса работы с сервером FTP:
1) `close`; 2) `get`; 3) `open`; 4) `ftp`.

36. Регистрация пользователя сервера FTP:
1) `ftp`; 2) `cd`; 3) `ls`; 4) `user`.

37. Команда **cd** протокола FTP используется:
1) для изменения текущего каталога; 2) для регистрации пользователя;
3) для навигации по дереву файловой системы; 4) для начала сеанса.

38. Какой командой следует пользоваться для просмотра каталогов FTP:
1) `ls`; 2) `mget`; 3) `bin`; 4) `get`?

39. По какой команде FTP можно принять или передать один файл:
1) `get, put`; 2) `mget, mput`; 3) `bin`; 4) `ls, cd`?

40. Для приема/передачи набора файлов FTP используется команда:
1) `get, put`; 2) `ls, cd`; 3) `user`; 4) `mget, mput`.
41. WWW — это:
1) распределенная информационная система мультимедиа, основанная на гипертексте;
2) электронная книга;
3) протокол размещения информации в Internet;
4) информационная среда обмена файлами.
42. Гипертекст — это:
1) информационная оболочка;
2) текст, содержащий иллюстрации;
3) информация в виде документов, имеющих ссылки на другие документы;
4) информационное хранилище.
43. Взаимодействие клиент—сервер при работе на WWW происходит по протоколу:
1) HTTP; 2) URL;
3) Location; 4) Uniform.
44. Какие программы не являются броузерами WWW:
1) Mosaic; 2) Microsoft Internet Explorer;
3) Microsoft Outlook Express; 4) Netscape Navigator?
45. HTML — это:
1) программа просмотра WWW-документов;
2) прикладная программа;
3) язык разметки гипертекстов;
4) протокол взаимодействия клиент — сервер.
46. В HTML можно использовать:
1) текст в ASCII-формате; 2) текст любого формата и графические рисунки;
3) любые мультимедиа-файлы; 4) любые типы данных.
47. Для чего служат в HTML символы `<HEAD><TITLE> </TITLE></HEAD>`:
1) для выделения абзаца; 2) для выделения параграфа, пункта;
3) для выделения глав; 4) для выделения заголовка?
48. Какими символами в HTML основной текст отделяется от сопроводительного:
1) `<TITLE></TITLE>`; 2) `<H1></H1>`; 3) `<BODY></BODY>`; 4) `<P></P>`?
49. Как в HTML описывается ссылка на другой документ:
1)`<A HREF="имя файла"></A>`; 2) с указанием их URL;
3) `<A NAME="имя файла"></A>`; 4) `<UL><LI></UL>`?
50. Как в HTML записываются ссылки на документы, хранящиеся на других серверах:
1) с указанием их URL; 2) `<UL><LI></UL>`;
3) `<A NAME="имя файла"></A>`; 4) `<A HREF="имя файла"></A>`?
51. Как в HTML задается положение рисунка:
1) `<IMG SRC=...>`; 2) `<ALIGN=...>`; 3) `<URL>`; 4) `<HR>`?
52. Что лежит в основе системы Gopher:
1) поиск информации с использованием логических запросов;
2) поиск по ключевым словам;
3) идея иерархических каталогов;
4) бинарный поиск?
53. На чем основана система WAIS:
1) на поиске информации с использованием логических запросов;
2) на поиске по ключевым словам;
3) на идее иерархических каталогов;
4) на бинарном поиске?

Операционная система UNIX

1. Какой язык программирования тесно связан с ОС UNIX:
   1) HTML; 2) Паскаль; 3) Си; 4) Java?
2. ОС UNIX — это:
   1) сетевая ОС для работы в Internet;
   2) многофункциональная сетевая ОС универсального значения;
   3) ОС для закрытых систем;
   4) ОС для поддержки среды Windows.
3. Какой командой в ОС UNIX можно узнать имя текущего каталога:
   1) `is`; 2) `change directory`;
   3) `cat`; 4) `pwd (print working directory)`?
4. Команда ls в ОС UNIX используется:
   1) для изменения рабочего каталога;
   2) для объединения нескольких файлов для печати;
   3) для вывода на экран содержимого каталога;
   4) для копирования файлов.
5. Изменение рабочего каталога в ОС UNIX производится командой:
   1) `cd`; 2) `cat`; 3) `pwd`; 4) `is`.
6. Что выполняет команда *cat* в ОС UNIX:
   1) определяет имя текущего каталога; 2) вывод на печать;
   3) объединяет файлы и направляет результат на вывод;
   4) вырезает фрагменты данных из файла?
7. Для чего служат метасимволы в ОС UNIX:
   1) для уничтожения всех файлов; 2) для уничтожения каталогов;
   3) для подстановки любых строк и символов в имена файлов;
   4) для переименования файлов?
8. Как образуется программный канал в ОС UNIX:
   1) назначением стандартного вывода одной команды вводом следующей команды;
   2) перенаправлением вывода команды с добавлением;
   3) введением произвольной строки в команду; 4) объединением команд?
9. Для получения почты в ОС UNIX вводится команда:
   1) `write`; 2) `mail`; 3) `delete`; 4) `who`.

Правильные ответы

Локальные сети

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1		X			8				X	15			X	
2			X		9	X				16				X
3				X	10	X				17				X
4		X			11			X		18		X		
5		X			12		X			19				X
6				X	13			X						
7	X				14			X						

Сетевые операционные системы

№	1	2	3	4	№	1	2	3	4
1			X		6				X
2	X				7	X			
3		X			8			X	
4		X			9			X	
5		X			10		X		

Операционная система UNIX

№	1	2	3	4	№	1	2	3	4
1			X		6			X	
2		X			7			X	
3				X	8	X			
4			X		9		X		
5	X								

Глобальные сети

№	1	2	3	4	№	1	2	3	4	№	1	2	3	4
1				X	19	X				37			X	
2			X		20	X				38	X			
3				X	21	X				39	X			
4	X				22		X			40				X
5				X	23			X		41	X			
6			X		24		X			42			X	
7		X			25	X				43	X			
8		X			26				X	44			X	
9	X				27			X		45			X	
10	X				28		X			46	X			
11			X		29			X		47				X
12				X	30		X			48			X	
13	X				31		X			49	X			
14				X	32		X			50			X	
15	X				33	X				51		X		
16		X			34		X			52			X	
17	X				35			X		53	X			
18			X		36				X					

Глава 6

ИНФОРМАЦИОННЫЕ СИСТЕМЫ

В данном разделе практикума, как и в базовом учебнике, затрагиваются лишь те аспекты информационных систем, которые не нашли отражения в других главах.

Подготовка специалиста по информатике, работающего в сфере образования, подразумевает ознакомление с работой классических информационных систем:

- информационно-справочных систем;
- систем автоматизации документооборота и учета;
- автоматизированных систем управления;
- экспертных систем;
- систем автоматизации научных исследований;
- систем автоматизированного проектирования;
- геоинформационных систем.

Учитывая высокий профессиональный уровень и сложность указанных систем, задача развития навыков их разработки для студентов той категории, которой адресован данный практикум, не ставится (эта задача может частично реализовываться в спецкурсах). Даже ознакомление с указанными выше информационными системами и отработка простых пользовательских навыков является в настоящее время трудно реализуемым с точки зрения разработки единого практикума, поскольку не существует общераспространенных программ, на которых такой практикум можно было бы единообразно базировать. Более того, высокая стоимость профессиональных программных продуктов этого класса может сделать практически бесполезной конкретные рекомендации. Наилучшим выходом из положения было бы создание простых программ-имитаторов классических информационных систем, назначение которых ограничивалось бы сферой обучения. Пример такой программы приводится ниже — имитатор геоинформационной системы, созданный в ПГПУ в ходе разработки серии программ-имитаторов информационных систем по заказу Минобразования РФ.

Значительную роль в проведении практических занятий по данному разделу могут сыграть рефераты и курсовые работы.

§ 1. БАНКИ ИНФОРМАЦИИ

Рекомендации по проведению занятий

Практическое знакомство с банками информации и приобретение навыков поиска и обработки информации, хранящейся в банках, требует их физического наличия. При этом есть три пути:

• работа с локализованными программами, хранящимися на жестком диске или на CD ROM-диске;
• работа в локальной сети учреждения;
• работа по сети Internet.

В первых двух случаях какие-либо универсальные рекомендации вряд ли возможны. В то же время работа в Internet открывает огромные возможности для ознакомления с различными типами банков информации, и при наличии соответствующей возможности этот путь представляется оптимальным.

Контрольные вопросы

1. Каковы основные функции банков информации?
2. Какие компоненты включают в себя информационно-поисковые системы?
3. Каковы режимы функционирования банков информации?
4. Каковы компоненты банка документов?
5. Какие иерархические классификационные системы наиболее распространены?
6. На чем основаны дескрипторные информационно-поисковые языки?
7. Каковы функции администратора банка информации?

Темы для рефератов

1. Информационно-справочные и информационно-поисковые системы.
2. Системы автоматизации документооборота и учета.
3. Банки данных.
4. Банки документов.
5. Иерархические классификационные системы.
6. Дескрипторные информационно-поисковые языки.

Темы семинарских занятий

1. Банки данных и банки документов.
2. Способы создания поисковых образов документов в банках информации.
3. Формирование информационных модулей и запросов в конкретном банке информации.

Рекомендации по программному обеспечению

В силу причин, отмеченных выше, в настоящий момент не представляется возможным выделить единое для различных вузов программное обеспечение для проведения практических работ по этой теме. Точнее говоря, трудно указать на *локализованные* банки информации, так как они в вузах либо отсутствуют, либо разнородны. Так, описанный в базовом пособии банк педагогической информации мог бы стать весьма подходящим объектом для выработки практических навыков работы с информационными системами, но у большинства вузов нет к нему доступа.

Тем не менее быстрый прогресс российского Internet позволяет организовать практическую работу по этой теме с хранящимися в Internet банками информации.

Таким образом, программным обеспечением, с которым студент будет работать явно, будет один из броузеров.

Ниже рассматривается вариант, связанный с работой в сети Internet. Программное средство — броузер Internet Explorer.

Задачи и упражнения

Упражнение № 1. Работа с банком документов

Как и в базовом пособии, положим в основу упражнения работу с банком педагогической информации. При наличии доступа к Internet это может быть один из банков, в которых собирается информация по образовательной системе России и мира.

Набрав адрес www.informika.ru, попадаем на сервер Министерства образования РФ, представляющий собой сочетание информационно-справочной системы, информационно-поисковой системы и банка документов. Заглавный кадр сайта содержит информацию об основных разделах банка (воспроизводим лишь содержимое, а не дизайн).

Минобразования России ГосНИИ ИТТ «Информика» Международное сотрудничество (англ.) Дистанционное образование Информационные технологии Конференции, выставки, семинары Фонды, гранты, конкурсы Путеводитель по Internet Базы данных Подготовка управленческих кадров Партнеры Газеты и журналы Книгоиздание Поиск на сервере	**НОВОСТИ** <ниже — три постоянные рубрики> • Новые поступления на сервер • Российская пресса об образовании • Российская пресса о науке <далее — порядка 10 сменных рубрик, например,> • Курсы повышения квалификации работников образования в Госкоорцентре • Международная научно-методическая конференция «Телематика-2000» и т.д.

В левой колонке этой таблицы — постоянные рубрики, отражающие основную деятельность Министерства образования. В правой колонке — основные информационно-справочные рубрики. Некоторые из них являются постоянными (с обновляемым содержанием), большая часть — сменными, в них отражается актуальная на настоящий момент информация с относительно коротким «временем жизни».

Войдем в раздел «Базы данных». Перед нами открывается обширный перечень баз данных по различным видам деятельности в системе образования. Приводим их перечень (с некоторыми сокращениями). Следует учесть, что номенклатура баз данных время от времени обновляется, и вы можете увидеть несколько отличный список.

Вузы России

Научные организации и высшие учебные заведения России

Перечень направлений подготовки и специальностей высшего профессионального образования

Государственные образовательные стандарты

Номенклатура специальностей научных работников

Интерактивный классификатор ГРНТИ

Геоинформация

Международное образование в России

Нормативно-правовая база (дистанционное образование)

Информационные ресурсы высшей школы России (автоматизированный кадастр)

Интерактивная картографическая информационная система «Вузы Москвы»

База данных Диссертационные советы вузов России

Международная поддержка Российской науки и высшей школы («Электронный справочник»)

Финансирование проектов в сфере муниципального и университетского самоуправления

Адреса электронной почты вузов и ЦНИТов по городам в алфавитном порядке

E-mail местных органов управления образованием

E-mail российских школ, лицеев, колледжей, гимназий, учебных центров

Нормативные документы (1990—1996 гг.)

ЮСИС-Internet («Законодательство и практика применения»)

База данных ИНФОПАРТНЕР

Абитуриент

Учитесь за рубежом

Ваша новая профессия

Детские сады Москвы

Отдых с детьми

****

Нормативные документы для регистрации баз данных в Государственном реестре баз данных

Войдем в одну из предлагаемых баз данных и произведем в ней:
а) просмотр содержимого;
б) поиск с помощью предлагаемых средств.

Упражнение № 2. Освоение иерархической классификационной системы — классификатора ГРНТИ

Один из разделов приведенного выше банка — интерактивный классификатор ГРНТИ (Государственный регистр научно-технической информации).

Войдем в этот раздел. Нам предлагается два меню: по номерам классификатора (которые нам поначалу ничего не говорят) и по предметным областям. Воспользуемся вначале вторым и наберем запрос — например, «Образование». Система найдет подразделы и коды. Затем по кодам уточним классификацию разделов и т.д.

Упражнение № 3. Работа с библиотечными каталогами

В системе образования головной является Государственная научная педагогическая библиотека им. К.Д.Ушинского. В качестве упражнения выполним работу с ее каталогами.

Набрав адрес http://www.gnpbu.iip.net, попадаем на первую страницу Web-сайта библиотеки. На ней, кроме красочного оформления и посторонних ссылок, нам предлагается меню из пяти пунктов:

О библиотеке
База данных
Методическая работа
Издания библиотеки
Услуги библиотеки

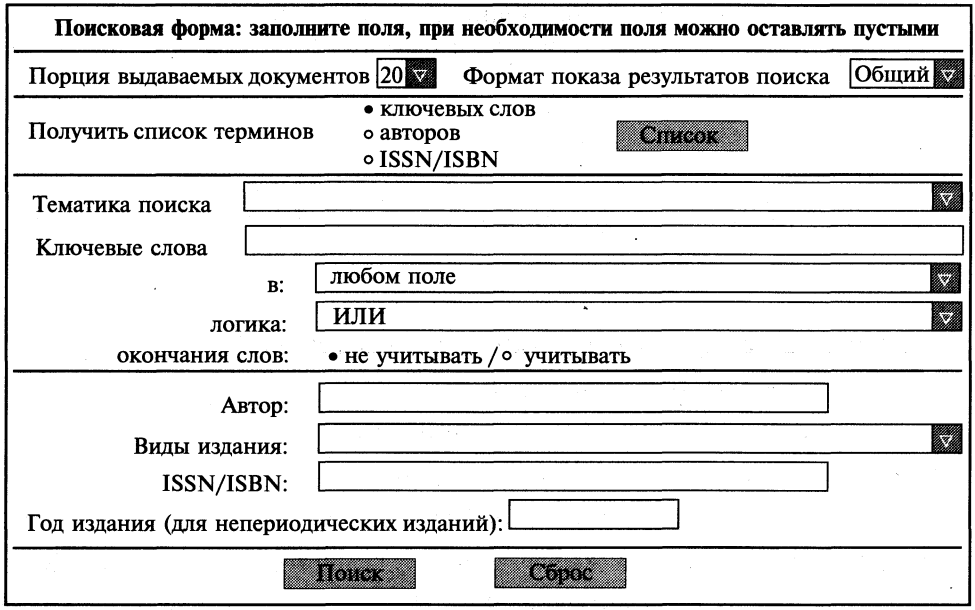
Выберем раздел «Базы данных» и попадем на страницу, на которой содержится описание раздела «Базы данных».

Раздел БД (базы данных) состоит:

из электронного каталога ГНПБ им. К.Д.Ушинского
коллекции ссылок на Интернет-ресурсы, ориентирующиеся на источники информации в области образования,
педагогики и психологии
сведенных в единое целое основных поисковых систем с запросными формами

Войдем в электронный каталог — нам предлагается запросная форма. В ней для поиска нужной литературы следует заполнить (или оставить пустыми) ряд полей, перечисленных ниже.

Согласимся с тем, что порция выдаваемых документов — 20, а формат показа результатов поиска — общий (оба показателя можно было бы при желании изменить в соответствующих меню). «Кликнем» по меню «Тематика поиска» и выберем из появившегося списка раздел «Образование педагогическое». Введем ключевое слово «Информатика» и не будем ничего более уточнять, т.е. начнем поиск. Система выдаст соответствующие документы — бегло их просмотрим.



Упражнение № 4. Работа с региональной информационно-справочной системой

В настоящее время большинство регионов России и многие крупные города имеют собственные информационно-справочные системы. Опишем в качестве примера упражнение по работе с информационно-справочной системой Пермской области. На практике целесообразно проделать аналогичную работу с информационно-справочной системой своего региона (города).

Запускаем программу Internet Explorer. После загрузки вводим адрес регионального сайта (в данном случае это http://www.perm.ru; адреса других региональных сайтов устроены схожим образом).

На экране появляется заглавный кадр. В целях экономии места приводим ниже аналог кадра в текстовом формате без рисунков (которые есть в оригинале), выделяя разделы, допускающие дальнейший поиск, подчеркиванием.

Поиск по Пермским серверам: Поиск по этому серверу:

Сервер находится под патронажем Главного управления экономики областной администрации.
E-mail для контактов gue@pstu.ac.ru
Регион — Губернатор — Новости — Погода — Бизнес — Власть — Политика — Инвестиции — Справочники — Проблема-2000 — История — Религия — Культура — Образование — Отдых — Транспорт — Туризм — Спорт — СМИ ...

WIN – KOI – DOS

All questions and comments please mail to webmaster@perm.ru

Перед нами обширная информационно-справочная система общего назначения. Из нее можно многое узнать о жизни региона. Входя, к примеру, в раздел Справочники, получаем следующий кадр:

Искать:

Справочник по улицам Расписание транспорта Справочная лекарственных препаратов

Издательство «М плюс Б»

Телефонно-адресный справочник «Пермь. Желтые страницы»

Пермь от А до Я — телефонный справочник

РИА «Навигатор». Телефонный бизнес-справочник

ИПС «Кодекс» — законодательство России, Перми

«Консультант Плюс» — региональный информационный центр

Справочник предприятий Пермской области на сервере группы предприятий «КИТ»

Справочник предприятий Пермской области (более 1900 предприятий) на сервере Пермского ЦНТИ

«Дизайн. Строительство. Отделка» — справочник по фирмам и магазинам строительного профиля

«Строительная индустрия Прикамья» — списки фирм, предприятий по производству стройматериалов; советы по покупке жилья

«Компьютерный мир Перми» — справочник по пермским компьютерным фирмам

Пермская Универсальная Торговая Система — оперативная информация с пермских рынков: векселя и долговые обязательства, мониторинг цен, операции с недвижимостью, продукты, товары, услуги

Пермская Торговая Система — база данных по коммерческим предложениям на товары и услуги

«Товары. Фирмы. Цены» — ежемесячное информационно-справочное издание, содержащее информацию о ценах на различные категории товаров

«Товары со склада» — еженедельная справочно-аналитическая газета (уральский региональный выпуск)

«Колесо» — электронный автосправочник

«Реклама в Пермской области» — электронный справочник

Выполните в качестве упражнения просмотр нескольких разделов системы.

Лабораторные работы

Лабораторная работа № 1. Банки документов

Задания к лабораторной работе

Время выполнения 6 часов.

Произвести поиск в банке документов системы образования. За основу принять либо региональный банк педагогической информации, либо описанный выше банк, находящийся на сервере Министерства образования РФ.

Если результат поиска в этих банках окажется отрицательным (информация не найдена или ее очень мало), проведите поиск указанной информации в Internet с помощью поисковой машины Rambler.

Составить описание поиска, включающее:

1) общее описание банка;
2) перечень объектов поиска;
3) траекторию поиска;
4) результаты поиска.

Варианты заданий

Вариант 1

Документы, имеющие отношение к вашему вузу. Если их нет в базе данных, то по ближайшему территориально педагогическому вузу.

Вариант 2

Документы по высшему педагогическому образованию.

Вариант 3

Документы по научной деятельности в сфере педагогики.

Вариант 4

Документы по международному образованию в России в целом и в нескольких вузах в отдельности.

Вариант 5

Документы по трудовому законодательству в сфере образования.

Вариант 6

Документы, регламентирующие содержание школьной информатики.

Вариант 7

Документы по организации конкурсов по научно-технической деятельности в сфере образования.

Вариант 8

Документы, связанные с защитой диссертаций (тематика, советы по защите и т.д.).

Вариант 9

Документы по международному сотрудничеству в системе образования.

Вариант 10

Документы по начальному образованию.

Вариант 11

Документы по начальному профессиональному образованию.

Вариант 12

Документы по реформе образования.

Вариант 13

Документы по 12-летнему образованию.

Вариант 14

Документы по истории образования в России.

Вариант 15

Стандарты образования по информатике различных ступеней системы образования.

Вариант 16

Материалы и документы по изучению геоинформатики.

Вариант 17

Материалы и документы по законодательству в системе образования.

Вариант 18

Статьи Гражданского кодекса, связанные с отношениями в сфере образования.

Вариант 19

Материалы и документы по государственным образовательным стандартам.

Вариант 20

Материалы и документы по поступлению в вузы России.

Вариант 21

Материалы и документы по финансированию и финансовой поддержке системы образования.

Вариант 22

Материалы и документы по организации управления образованием.

Вариант 23

Материалы и документы по учебному книгоизданию.

Вариант 24

Материалы и документы по профессиональной переподготовке и повышению квалификации в системе образования.

Лабораторная работа № 2. Каталоги электронных библиотек

Задания к лабораторной работе

Время выполнения 4 часа.

Произвести поиск литературы в каталоге указанной электронной библиотеки на заданную тему (тема сформулирована ориентировочно, возможны вариации). Если число источников, предъявляемых системой в ходе поиска, велико, то ограничиться просмотром части из них.

Составить описание поиска, включающее:

1) общее описание электронного каталога;
2) траекторию поиска;
3) результаты поиска.

Варианты заданий 1 — 8

Поиск в каталоге ГНПБ им. К.Д.Ушинского
(http://www.gnpbu.iip.net)

Вариант 1. Литература по теме «Педагогика и психология высшей школы».
Вариант 2. Литература по теме «Информатика в школе».
Вариант 3. Литература по теме «Математика в школе».
Вариант 4. Литература по теме «Физика в школе».
Вариант 5. Литература по теме «История образования».
Вариант 6. Литература по теме «Филология».
Вариант 7. Литература по теме «Дошкольное образование».
Вариант 8. Литература по теме «Начальная школа».

Варианты заданий 9 — 16

Поиск в каталоге Российской государственной библиотеки
(http://www.rsl.ru)

Вариант 9. Базы данных.
Вариант 10. Геоинформатика.
Вариант 11. САПР.
Вариант 12. АСНИ.
Вариант 13. Информационно-справочные системы.
Вариант 14. Экспертные системы.
Вариант 15. Базы знаний.
Вариант 16. Искусственный интеллект.

Варианты заданий 17 — 24

Поиск в каталоге Государственной публичной научно-технической библиотеки
(http://www.gpntb.ru)

Вариант 17. Базы данных.
Вариант 18. Геоинформатика.
Вариант 19. САПР.
Вариант 20. АСНИ.

Вариант 21. Информационно-справочные системы.
Вариант 22. Экспертные системы.
Вариант 23. Базы знаний.
Вариант 24. Искусственный интеллект.

Лабораторная работа № 3. Информационно-справочные системы

Задания к лабораторной работе

Время выполнения 4 часа.

Осуществить поиск информации в информационно-справочной системе своего региона (города). Составить описание поиска, включающее:
1) общее описание информационно-справочной системы;
2) перечень объектов поиска;
3) траекторию поиска;
4) результаты поиска.

Варианты заданий

Каждый студент выбирает свой перечень объектов поиска и согласует его с преподавателем.

Дополнительная литература

1. *Айламзян А.К.* Информация и информационные системы. — М.: Радио и связь, 1982.
2. *Блюменау Д.И.* Проблемы свертывания научной информации. — Л.: Наука, 1982.
3. ГОСТ 7.27 — 80. Научно-информационная деятельность: Основные термины и определения. — М.: Изд-во стандартов, 1983.
4. *Звежинский С.М.* Информационное обеспечение научно-технических разработок. — Львов: Выща шк., 1982.
5. *Иванов Р.Н.* Организация и методика информационной работы. — М.: Радио и связь, 1982.
6. *Корюкова А.А., Дера В.Г.* Основы научно-технической информации. — М.: Высш. шк., 1985.
7. *Котов Р.Г., Якунин Б.В.* Языки информационных систем. — М.: Наука, 1979.
8. *Монастырский И.М.* Информационно-поисковые системы. — М.: Экономика, 1983.
9. *Назаров С.В., Першиков В.И. и др.* Компьютерные технологии обработки информации. — М.: Финансы и статистика, 1995.
10. *Овчинников В.Г.* Автоматизированные системы информационного обеспечения. — М.: Энергия, 1977.
11. *Поваляев С.А.* Основы информатики. — М.; Минск, 1987.
12. *Полищук Ю.М., Хон В.Б.* Теория автоматизированных банков информации. — М.: Высш. шк., 1989.
13. *Севастьянов Ю.С., Фокин В.М., Махотенко Ю.А.* Научные и организационные основы информационной деятельности. — М.: Радио и связь, 1983.
14. *Советов Б.Я.* Информационная технология. — М.: Высш. шк., 1994.
15. *Соколов А.В.* Информационно-поисковые системы. — М.: Радио и связь, 1983.
16. *Ходаков В.Е.* Системы информационного обслуживания. — Киев: Выща шк., 1983.

§ 2. АВТОМАТИЗИРОВАННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Рекомендации по проведению занятий

При изучении информационных систем на уровне, необходимом учителю информатики, и тем более при ознакомлении с принципами их организации студентов — будущих педагогов других специальностей — возникают трудности, связанные:

- с необъятностью материала по каждому из таких классов систем;
- с неподготовленностью студентов, ориентированных на иную сферу деятельности;
- со сложной организацией профессиональных информационных систем, их малой доступностью (во всех смыслах, включая необходимость специальных знаний, высокие требования к аппаратной части, высокую стоимость программ).

Для указанных целей желательны относительно простые программы, позволяющие понять именно принципы организации и работы указанных систем, а не быстро меняющиеся детали реализации конкретных технологий и не требующие специальных знаний из предметных сфер деятельности.

Характер и уровень подготовки студентов педагогического вуза делают более целесообразным использование в учебном процессе не профессиональных информационных систем (АСУ, САПР, АСНИ, ГИС и т.д.), а их учебных аналогов-имитаторов, воспроизводящих основные функции таких систем в объеме, достаточном для понимания основ их функционирования. Требования к таким имитаторам — современный интерфейс, высокая учебная технологичность, относительно невысокие требования к компьютерной технике и т.д. Использование таких программ педагогически и экономически полностью оправдано и соответствует традициям использования в учебном процессе школ и вузов специального учебного оборудования.

Отметим, что изучение учебных имитаторов, как правило, значительно проще, чем соответствующих им профессиональных программ, поскольку имитатор содержит только самые необходимые возможности и свободен от избыточных технических деталей.

К сожалению, в настоящее время существуют имитаторы далеко не всех перечисленных выше информационных систем. При отсутствии такой программы можно рекомендовать базировать практические занятия по соответствующей теме на самых простых из доступных программ данного класса (как правило, из числа программ, легально предоставляемых бесплатно). Ниже приведены примеры таких программ.

В данном пункте ограничимся достаточно детальным описанием практикума по автоматизированным информационным системам на примере ГИС. Этот выбор обусловлен следующими обстоятельствами:

- современные ГИС вобрали в себя черты ряда классических информационных систем;
- быстрый прогресс в разработке и использовании ГИС, признание за ними роли наиболее универсальных из автоматизированных информационных систем;
- возможность предложить имитатор ГИС, специально разработанный для знакомства с ними, той категории студентов, которой адресовано данное пособие.

Такой выбор не означает, что в ходе подготовки учителя не нужен практикум по другим автоматизированным информационным системам. Напротив, желательно проведение лабораторно-практических занятий с каждой из них. Однако конкретный выбор программ — за учебным заведением; некоторые рекомендации даны ниже.

Краткие сведения

Различают следующие основные классы автоматизированных информационных систем:

- автоматизированные системы управления;
- автоматизированные системы научных исследований;
- системы автоматизированного проектирования;
- геоинформационные системы;
- экспертные системы.

Автоматизированная система управления (АСУ) — комплекс технических и программных средств, обеспечивающий управление объектом в производственной или административной среде.

Автоматизированные системы научных исследований (АСНИ) — программно-аппаратные комплексы, обрабатывающие данные, поступающие от экспериментальных установок и измерительных приборов.

Системы автоматизированного проектирования (САПР) реализуют принципы геометрического моделирования и компьютерной графики, служат для подготовки чертежей (так называемые CAD-системы); более специализированные CAM-системы сосредоточены на технологии изготовления изделий конкретного назначения. Наиболее совершенные САПР являются интегрированными CAD/CAM-системами.

Геоинформационные системы

Современная ГИС — это автоматизированная система, имеющая большое количество графических и тематических баз данных, соединенная с модельными и расчетными функциями для манипулирования ими и преобразования их в пространственную картографическую информацию для принятия на ее основе решений и осуществления контроля. Базы данных являются обязательными компонентами ГИС, в которых хранятся любые данные (графическая основа, объекты на карте и дополнительные сведения), связанные с определенной картой.

Современные ГИС сочетают в себе черты АСУ, информационно-справочных систем, картографических информационных систем, баз данных, САПР, АСНИ, систем документационного обеспечения. Высказывается точка зрения, что ГИС является интегрированной информационной системой, объединяющей концептуально, структурно и методически названные выше информационные системы. Авторы современных обзоров подчеркивают, что, говоря о ГИС, разные люди часто подразумевают различные системы как структурно, так и по-разному ориентированные — на экологию и природопользование, земельный кадастр и землеустройство (с этого ГИС начинались), управление городским хозяйством, демографию и трудовые ресурсы, управление дорожным движением, социологию и политологию и т.д.

Программа, относящаяся к классу ГИС, в обязательном порядке реализует следующие функции:

- ввод картографических данных путем преобразования в подходящий цифровой формат;

• манипулирование данными, включая представление карт в разных масштабах;
• управление базами данных (обычно реляционного типа);
• обслуживание запросов на информацию;
• визуализацию информациии, основанную как на географических картах, так и на построении таблиц, графиков.

Контрольные вопросы

1. Какие существуют виды автоматизированных информационных систем?
2. Каковы функции и структурные схемы автоматизированных систем управления?
3. Каковы функции и структурная схема автоматизированных систем управления технологическими процессами?
4. Каковы функции и структурная схема автоматизированных систем научных исследований?
5. Каковы функции и структурная схема экспертных систем?
6. Каково назначение ГИС?
7. Какие бывают разновидности ГИС?
8. Какие функции являются обязательными для каждой программы класса ГИС?

Темы для рефератов

1. Автоматизированные системы управления.
2. Автоматизированные системы управления в образовании.
3. Автоматизированные системы управления технологическими процессами.
4. Системы автоматизированного проектирования в строительстве.
5. Системы автоматизированного проектирования в машиностроении.
6. Геоинформационные системы в экологии и природопользовании.
7. Геоинформационные системы в ведении земельных кадастров.
8. Экспертные системы в медицине.
9. Инструментальные программные средства для создания экспертных систем.

Темы семинарских занятий

1. Автоматизированные системы управления и информационные системы управления.
2. Системы автоматизированного проектирования и автоматизированные системы научных исследований.
3. Геоинформационные системы.
4. Экспертные системы.

Рекомендации по программному обеспечению

Геоинформационные системы

Существует большое число программных ГИС-систем. В принципе для ознакомления можно использовать любую универсальную ГИС, доступную в данном вузе, однако в силу причин, указанных выше, в данной ситуации предпочтителен имитатор ГИС.

Ниже приводится краткое описание двух программ. Обе вполне доступны, полностью «русскоязычны», что для проведения занятий также немаловажно.

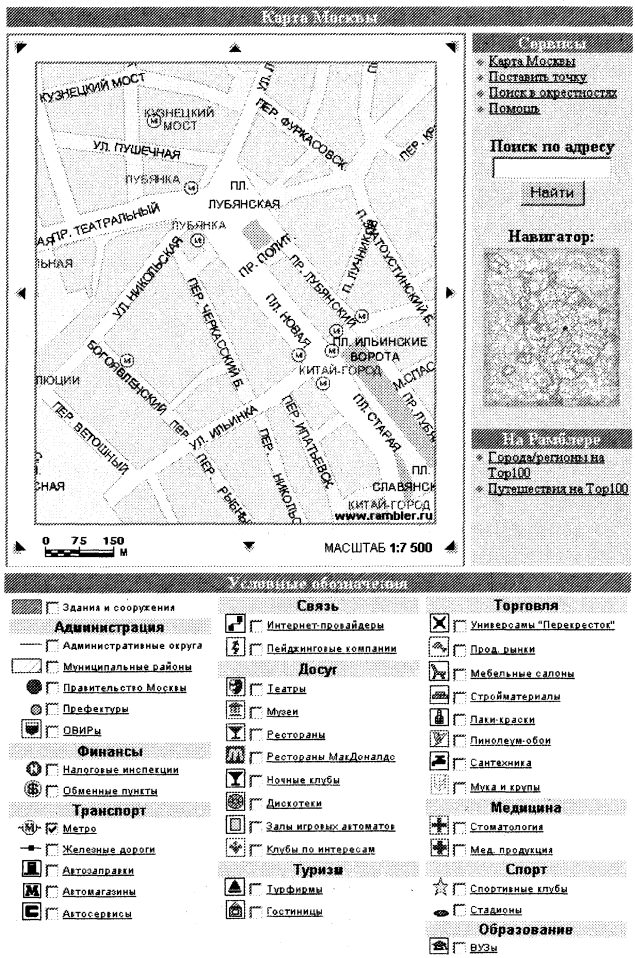
ГИС «Карта Москвы»

Настольная ГИС «Карта Москвы» распространяется на CD ROM; одна из ее версий расположена в открытом доступе на сайте информационно-поисковой системы Рамблер по адресу www.rambler.ru/map.

ГИС «Карта Москвы» обладает стандартными возможностями навигации и масштабирования, кроме того, карта:

- *связана с базами данных,* т.е. позволяет отображать объекты разных сфер жизни города;
- *может быть персонифицирована,* т.е. на ней можно отобразить только интересующие вас объекты;
- *поддерживает функцию поиска* по названиям улиц, адресам, названиям объектов и организаций, роду деятельности и близлежащим объектам.

Рабочее окно системы имеет следующий вид:



Справка по работе с системой

Навигация. Имеются два средства навигации: сама карта, расположенная в центре страницы, и *Навигатор*, который находится на панели *Сервисов*, справа от карты. Красный прямоугольник на *Навигаторе* отмечает положение фрагмента, показанного на большой карте. Щелчок мыши по любой точке карты или *Навигатора* приведет к перемещению выбранной точки в центр отображаемого участка.

Вокруг карты расположены стрелки. Нажатие на них приведет к перемещению фрагмента карты в направлении, указанном стрелками.

С помощью выпадающего списка *«При нажатии на карту масштаб»* можно отобразить участки карты в разных масштабах. Возможные варианты:

- *сохранить* — производится перемещение указанного мышью фрагмента в центр отображаемого участка карты; изменения масштаба при этом не происходит;
- *увеличить* — увеличивается масштаб карты, т.е. выбранный участок выводится более подробно;
- *уменьшить* — масштаб карты уменьшается;
- *увеличить до* max — позволяет максимально увеличить отображаемый участок карты;
- *уменьшить до* min — позволяет увидеть карту в наименьшем масштабе.

Работа со слоями. Объекты, информацию о которых содержит программа, объединены в тематические группы — *слои*. Можно выбирать слои, которые будут отображаться на карте Москвы.

Список всех доступных объектов находится в рабочем окне программы ниже карты. Каждому объекту соответствует символ, который отображается на карте.

Чтобы включить объект в карту, нужно выбрать его из списка и нажать на кнопку *«Обновить»*.

Щелкнув мышью по названию объекта в списке, можно получить список всех объектов.

Щелкнув мышью по объекту на карте, можно получить информацию по выбранному объекту. По различным слоям выдается разная информация, но, как правило, выдается название объекта и его адрес.

Адресный поиск. В поле *«Поиск по адресу»* набрать несколько букв, с которых начинается название нужной улицы или название улицы полностью, и нажать на кнопку *«Найти»*. Появится список улиц, названия которых удовлетворяют условию поиска. Щелкнуть мышью по нужной улице — появится список домов, которые находятся на данной улице. Щелкнуть по нужному дому мышью — и на карте красным цветом отобразится интересующий вас адрес.

Поиск в окрестностях. Для проведения поиска в окрестностях необходимо выбрать ссылку *«Поиск в окрестностях»* на панели Сервисов. В этом режиме работы можно выбрать объекты, которые вы хотели бы найти, с помощью списка *«Произвести выборку по»*, задать *«Радиус выборки»*, в пределах которого требуется выполнить поиск объектов. Щелкнув мышью по заданной точке, вы получите интересующую вас информацию: карту с нанесенными на ней объектами. Под картой будет отображен список найденных объектов. Нажав на объект из списка, вы увидите детальную карту окрестностей данного объекта.

Нанесение собственных объектов. Система позволяет наносить собственные объекты на карту. Для этого необходимо выбрать на панели Сервисов *«Поставить точку»*, задать форму, цвет, размер, надпись. Щелчком мыши нанести значок в нужное место карты.

Имитатор ГИС Geo-Perm 2000

Программа создана в 1999 — 2000 гг. (при поддержке Минобразования РФ) в Пермском государственном педагогическом университете (А. В. Князев, Е. К. Хеннер). Программа в той версии, которая обеспечивает проведение указанных ниже работ, может быть получена бесплатно при запросе по адресу postmaster@pspu.ac.ru. Она позволяет отработать основы представлений о ГИС у тех категорий специалистов, для которых на сегодняшний день в силу характера подготовки актуальным является создание общего представления о структуре и возможностях ГИС, провести несколько лабораторно-практических занятий, сконструировать простейшую «собственную» ГИС (поскольку является инструментальным средством).

Программа Geo-Perm 2000 создана для работы в среде Windows'95 (Windows'98, Windows NT). Для инсталляции программы требуется около 10 Мбайт дискового пространства. Она использует традиционный интерфейс ОС Windows'95, что существенно облегчает работу с программой и ускоряет процесс обучения. Программа позволяет создавать электронные тематические атласы на основе представления цифровых карт и связанных с ними атрибутивных цифровых данных.

Имитатор геоинформационной системы обеспечивает создание и поддержку баз данных, являющихся основой для построения ГИС города, региона, страны.

Программа способна связывать с картографическими (графическими) объектами некоторую описательную, атрибутивную информацию, представленную в виде таблиц реляционной СУБД. То есть каждому точечному графическому объекту ставится в соответствие строка таблицы — запись в БД. В Geo-Perm 2000 информация хранится в различных связанных таблицах для объектов: *Населенный пункт*, *Водный объект*, *Объект инфраструктуры* и *Другие*. Перечисленные объекты создаются на сканированной карте (готовые графические файлы с расширением .bmp, .emf, .wmf). Объект также может быть создан пользователем путем сканирования собственной карты.

Программа-имитатор позволяет осуществлять манипулирование объектами ГИС путем перемещения, переименования, изменения параметров шрифта, выделения цветом и удаления объектов.

Наличие реляционной базы данных позволяет осуществлять связь между пространственными объектами посредством связанных таблиц. В режиме *Базы данных* возможен просмотр введенной информации по любому объекту, редактирование или удаление данных.

Программа позволяет осуществлять поиск по названию объекта в таблицах базы данных.

Проект сохраняется в файле с расширением **.gsm**.

Имитатор ГИС позволяет просматривать, редактировать данные, находить объекты и реализовывать другие функции современной ГИС, такие как ввод, манипулирование, управление, запрос и анализ, визуализацию, рассмотренные выше.

В процессе доработки программы планируется реализовать возможность осуществления обмена с другими программными средствами (например, Corel Draw, Adobe PhotoShop и др.), использовать различные способы отображения объектов (цвет, тип линии и др.), распечатывать готовые файлы на принтере и другое. Дополнительно будет разработана обучающая Демо-версия и электронный справочник программы.

Автоматизированные системы управления

Рекомендуется использовать для ознакомления элементы АСУ-ВУЗ, используемые в конкретном вузе, включая подсистемы «Абитуриент», «Деканат», «Учебная часть» и др.

Системы автоматизированного проектирования

Среди универсальных САПР (систем конструирования изделий, или CAD-систем) широко известен пакет ***AutoCad*** фирмы «Autodesk». В 1994 г. была выпущена уже 13-я версия этого продукта. Это высокопроизводительное программное обеспечение с Windows-интерфейсом позволяет одинаково хорошо решать машиностроительные, архитектурные и иные задачи для широкого класса приложений. Перечень задач, решаемых в AutoCad, таков (из фирменного сообщения):

- выполнение архитектурно-строительных чертежей;
- проектирование интерьера и планировка помещений;
- изготовление технологических схем и организационных диаграмм;
- изготовление чертежей для электронных, химических, строительных, машиностроительных и аэрокосмических приложений;
- изготовление топографических и морских карт;
- проектирование судов;
- графическое и другое представление математических функций;
- выполнение театральных декораций;
- запись партитуры музыкальных произведений;
- изготовление технических иллюстраций и схем, торговых марок и фирменных знаков, поздравительных открыток, а также других художественно-графических работ.

Возможности AutoCad не ограничиваются созданием статических рисунков. Он позволяет их «оживлять», добиваться киноэффекта.

Если вуз имеет данную программу, то на ее основе можно разработать несколько элементарных упражнений и лабораторных работ. Однако, учитывая высокую стоимость программы (порядка $4000) и неспецифичность ее для педобразования, такое решение не может быть массовым. Кроме того, обилие функций этой САПР становится, как ни парадоксально, препятствием для тех целей, которые ставятся в данном случае. Тем не менее на AutoCad можно с успехом ориентировать спецкурс, особенно связанный с изучением черчения.

Для ознакомления с универсальными САПР лучше подходит отечественный пакет **КОМПАС-ГРАФИК LT**, являющийся упрощенной версией профессиональной системы КОМПАС-ГРАФИК. Поскольку предприятие АСКОН (разработчик и владелец программы) предлагает бесплатную некоммерческую версию, у вуза не возникают финансовые проблемы. По своим возможностям программа более чем достаточна для проведения ознакомительных занятий по универсальным САПР, в которых главным является автоматизация чертежно-графических работ. Немаловажно для наших целей, что эта программа является полностью русскоязычной. Первоначальную информацию о программе можно получить по адресу http://www.ascon.ru.

Чрезвычайно удобным для использования этой программы в учебном процессе является наличие в ней приложения — обучающей части. Фирма предлагает пользователям набор упражнений для изучения КОМПАС-ГРАФИК LT и соответствующую книгу. Часть этих упражнений может быть непосредственно

превращена в лабораторные работы (в том смысле, как это принято в данном пособии).

Кроме универсальных САПР, представляется полезным практическое ознакомление с технологическими, специализированными САПР (CAM-системами). Препятствием к этому обычно служат следующие обстоятельства:

- высокая стоимость таких программ;
- высокие требования к аппаратной части;
- необходимость наличия специальных знаний в предметной области, которых нет у студентов — будущих педагогов (и которые им не нужны, по существу).

Из программ, которые предлагаются в сети на некоммерческих условиях, обратим внимание на отечественную САПР для раскроя листовых материалов ***«Астра-Д»***.

Демоверсия программы, позволяющая выполнять некоторые самостоятельные работы, выложена в сети по адресу http://www.chat.ru/~astranet. Программа имеет вполне современный интерфейс, встроенную систему помощи, является полностью русскоязычной. Постановка задачи об оптимальном раскрое не требует специальных знаний, что в данном случае очень важно. Даже с помощью всего лишь демоверсии этой программы вполне можно провести несколько упражнений и лабораторную работу.

Задачи и упражнения

Геоинформационные системы (с использованием программы Geo-Perm 2000)

Упражнение № 1. Ввод и загрузка карты

Как правило, создание ГИС начинается с ввода карты (подложки). В данной программе используется заранее подготовленный набор карт, которые хранятся в каталоге **Maps** (в дальнейшем планируется создание возможности автоматического ввода карт путем сканирования из программы). Имитатор позволяет использовать карты с расширением **bmp, emf, wmf**.

Для осуществления операции загрузки карты необходимо выполнить команду ***Файл\Загрузить карту*** (ALT + M) и выбрать из каталога **Maps** (по умолчанию программа обращается к каталогу **Maps**) нужный файл.

Пример.

1. Запуск программы Geo-Perm 2000.

Из главного меню Windows'95 выполните команду ***Пуск\Программы\Geo-Perm_2000***.

2. Помещение в окно программы географической карты (например, карты Евразии).

Выполните команду ***Файл\Загрузить_карту*** или используйте сочетание клавиш Alt + M. В каталоге **Maps** (он автоматически открывается при вызове этой команды) хранятся географические карты в виде графических файлов с расширением .bmp. Выберите файл evrasia.bmp.

3. Изучение окна программы.

Если вы все сделали верно, то на экране появится картина, изображенная на рис. 6.1.

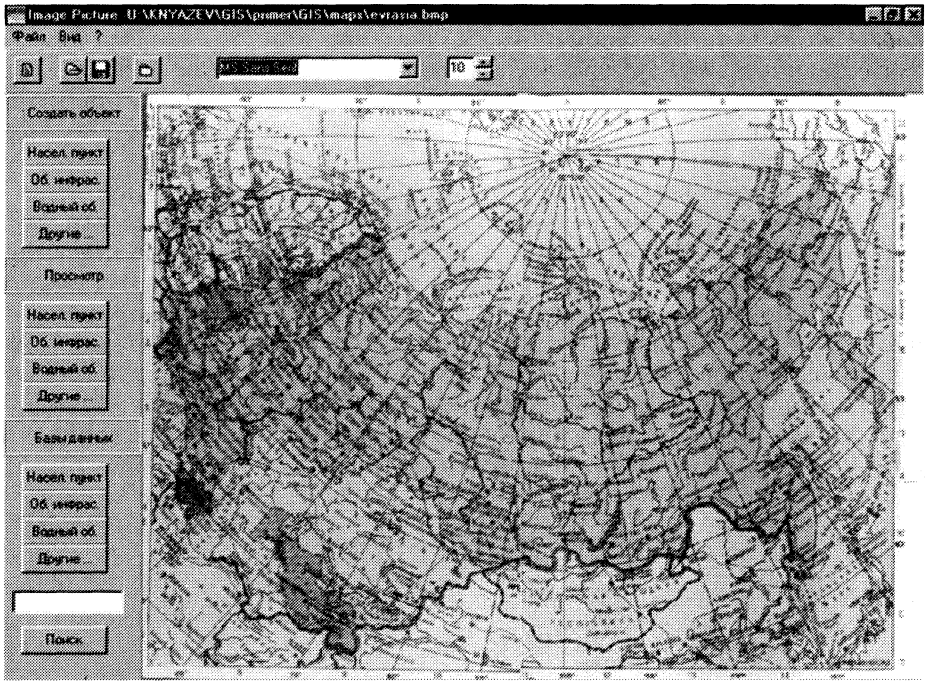


Рис. 6.1. Пример загрузки карты

Упражнение № 2. Создание объекта

Программа позволяет наносить на карту объекты, которые разделены на четыре группы: ***Населенный пункт***, ***Водный объект***, ***Объект инфраструктуры***, ***Другие***. Каждая группа объектов связана с отдельной таблицей.

Создание и привязка объекта к карте осуществляется достаточно просто. Для этого необходимо в меню ***Создать_объект*** выбрать кнопку с названием группы, к которой относится создаваемый объект. После нажатия на кнопку в верхнем левом углу карты возникнет объект (S, T, R или Object). Удерживая левую кнопку мыши в нажатом состоянии, необходимо перенести объект в нужную точку карты. Щелчком правой кнопки мыши по объекту вызывается меню. Выбрав пункт меню ***Имя_объекта***, можно задать имя рассматриваемого объекта. Для записи в базу данных информации по созданному объекту нужно выбрать пункт меню ***Ввести_данные***. После выполнения этой команды в поле ***Объект*** базы данных автоматически заносится имя рассматриваемого объекта, и она переходит в режим ввода данных.

Используя меню объекта, кроме рассмотренных функций, можно осуществлять просмотр и редактирование информации по объекту в базе данных, а также удаление объекта.

Пример.

1. Создание нового населенного пункта.

• Нажмите кнопку ***Насел._пункт*** в режиме создания объекта. На карте появится объект T1.

Замечание. Для любого объекта ГИС существует меню, вызываемое правой кнопкой мыши. При помощи этого меню в любой момент можно переименовать объект

командой ***Имя объекта***, ***Ввести данные*** (новое поле), ***Редактировать данные***, ***Удалить объект*** и получить информацию о данном объекте.

• На созданном объекте правой кнопкой мыши вызовите меню и выберите команду ***Имя объекта***. Вместо T1 введите слово *Москва*. Удерживая в нажатом состоянии левую кнопку мыши, перенесите объект на то место карты, где расположен город Москва.

2. Создание нового водного объекта.

• Нажмите кнопку ***Водный объект*** в режиме ***Создания объекта***. На карте появится объект R1. Вместо R1 напечатайте *Кама* (командой ***Имя_объекта***). Перенесите этот объект на карту в любое место, где протекает река Кама.

Замечание. Новый объект инфраструктуры создается аналогично предыдущим. Обычно этот объект создают на карте бóльшего масштаба. Новый объект ***Другие...*** размещают на карте с любым масштабом.

3. Изменение шрифта названий объектов.

Используя панель инструментов, установите следующие параметры: размер шрифта — 12, тип шрифта — произвольный.

Упражнение № 3. Просмотр ГИС

Программа позволяет просматривать группы объектов, расположенных на карте, путем их выделения.

Пример.

• Нажмите кнопку ***Насел._пункт*** в режиме просмотра. На экране все населенные пункты будут выделены красным цветом.

Замечание. Чтобы отменить выделение цветом, повторно нажмите на эту же кнопку.

• Выделите и просмотрите только водные объекты.

Упражнение № 4. Ввод и редактирование данных

Программа позволяет выполнять просмотр и редактирование базы данных. Для осуществления этих функций используется меню ***Базы_данных***. Встроенный в окно базы данных *Навигатор* позволяет легко осуществлять просмотр и редактирование данных.

Пример.

1. Заполнение базы данных сведениями о населенных пунктах.

• Нажмите кнопку ***Насел._пункт*** в режиме ***Базы_данных***.

• Изучите следующие режимы базы данных:

 — Отослать изменения в таблицу

 — Отменить изменения

 — Редактировать запись

 — Вставить новую запись

 — Удалить запись

 — Следующая запись

 — Предыдущая запись

 — Последняя запись

 — Первая запись

 — Обновить отображаемые значения

• Введите следующие данные для объекта Москва (рис. 6.2).

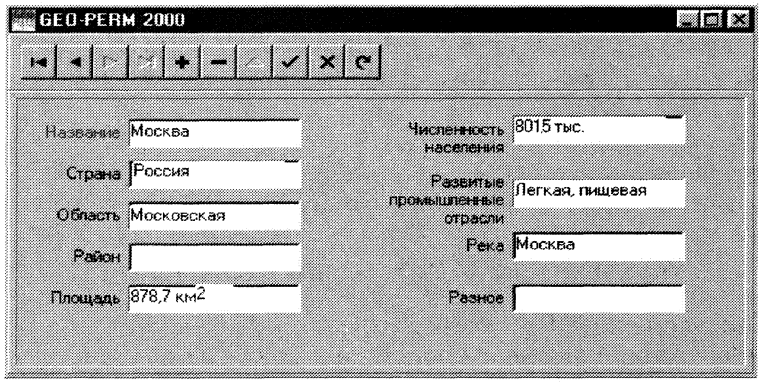


Рис. 6.2. Пример ввода данных для создания объекта

2. Ввод данных для водных объектов.

• Нажмите кнопку ***Водный_объект*** в режиме ***Базы_данных***.

• В появившемся окне введите запрошенную информацию, отраженную на рис. 6.3.

Замечание. Лучше всего данные о водном объекте вводить в левую колонку, их можно взять из Справочных материалов. При заполнении поля ***Крупный населенный пункт*** в правой колонке (поля красного цвета) автоматически появляется ранее введенная вами информация об этом населенном пункте, так как таблицы в базе данных связаны.

• Закройте окно базы данных.

3. Редактирование водных объектов.

В режиме ***Базы_данных*** измените данные в поле ***Судоходность***: вместо *Есть* напечатайте *Да*.

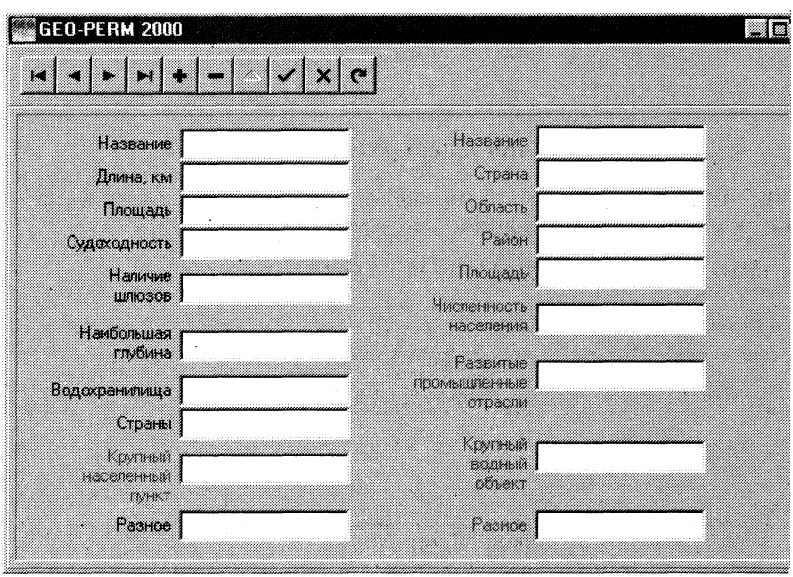


Рис. 6.3. Вид окна для ввода информации о водном объекте

Упражнение № 5. Поиск объектов в базе данных

Поиск объектов в базе данных осуществляется путем ввода в окно поиска имени нужного объекта. В данном режиме программа осуществляет поиск путем просмотра записей в поле ***Объект*** всех таблиц.

Пример.

• В окне режима ***Поиск*** наберите *Москва.*

Замечание. Поиск осуществляется по имени объекта.

• После просмотра информации об этом объекте выйдите из базы данных.

Упражнение № 6. Сохранение созданной ГИС

Сохранение созданной ГИС осуществляется в файл с расширением **.gsm**. При выполнении операции сохранения достаточно ввести название созданного проекта. Привязка расширения осуществляется автоматически.

Пример.

• Выполните команду ***Файл\Сохранить.***

• Введите имя файла: *evrasia.*

• Подтвердите сохранение.

По умолчанию проект сохраняется в каталоге **projects**.

Упражнение № 7. Открытие проекта ГИС

Этот режим позволяет осуществлять загрузку ранее созданного проекта. Для чего достаточно выполнить команду ***Файл\Открыть_проект*** и выбрать из каталога **projects** нужный файл.

Лабораторные работы

Лабораторная работа № 1. Геоинформационная система «Карта Москвы»

Время выполнения 4 часа.

Задания к лабораторной работе

1) Осуществить поиск объекта по указанному адресу, найти ближайшую станцию метро, отметить на карте место, соответствующее заданному адресу.

Музеи, находящиеся в радиусе 1 км от указанного адреса.

№ варианта	1	2	3	4	5	6
Адрес	Петровка, 17	Арбат, 12	Страстной бульвар, 10	Столешников переулок, 10	Кузнецкий мост, 17	Успенский переулок, 3

Вузы, находящиеся в радиусе 1 км от указанного адреса.

№ варианта	7	8	9	10	11	12
Адрес	Покровский бульвар, 4	Тверской бульвар, 20	Ленинский проспект, 10	Бронная Б, 17	Леонтьевский переулок, 12	Университетский проспект, 14

Пункты обмена валют, находящиеся в радиусе 1 км от указанного адреса.

№ варианта	13	14	15	16	17	18
Адрес	Кузнецкий мост, 6	Покровский бульвар,13	Рождественский бульвар, 20	Неглинная, 16	Страстной бульвар, 12	Цветной бульвар, 11

Гостиницы, находящиеся в радиусе 1 км от указанного адреса.

№ варианта	19	20	21	22	23	24
Адрес	Кузнецкий мост, 17	Лубянка, 5	Университетский проспект, 4	Дмитровский переулок, 8	Дегтярный переулок, 7	Трехпрудный переулок, 4

2) Найти адреса указанных в варианте вузов.

Варианты заданий

Вариант 1

Академия реставрации.

Вариант 2

Российская государственная академия труда и занятости.

Вариант 3

Военный авиационный технический университет им. Н. Е. Жуковского.

Вариант 4

Всероссийский государственный институт кинематографии им. С. А. Герасимова.

Вариант 5

Высшая коммерческая школа Министерства экономического развития и торговли РФ.

Вариант 6

Высшее театральное училище им. Б. В. Щукина.

Вариант 7

Высший институт управления.

Вариант 8

Дипломатическая академия МИД Российской Федерации.

Вариант 9

Институт мировой экономики.

Вариант 10

Литературный институт им. А. М. Горького.

Вариант 11

Московская академия экономики и права.

Вариант 12

Московская государственная консерватория им. П. И. Чайковского.

Вариант 13

Московская медицинская академия им. И. М. Сеченова.

Вариант 14

Московская сельскохозяйственная академия им. К. А. Тимирязева.

Вариант 15

Московский государственный авиационный технический университет.

Вариант 16

Московский государственный инженерно-физический технический университет.

Вариант 17

Московский государственный институт международных отношений.

Вариант 18

Московский государственный институт стали и сплавов.

Вариант 19

Московский государственный строительный университет.

Вариант 20

Московский государственный технический университет им. Н. Э. Баумана.

Вариант 21

Московский государственный университет им. М. В. Ломоносова.

Вариант 22

Московский государственный университет путей сообщения.

Вариант 23

Московский гуманитарно-экономический институт.

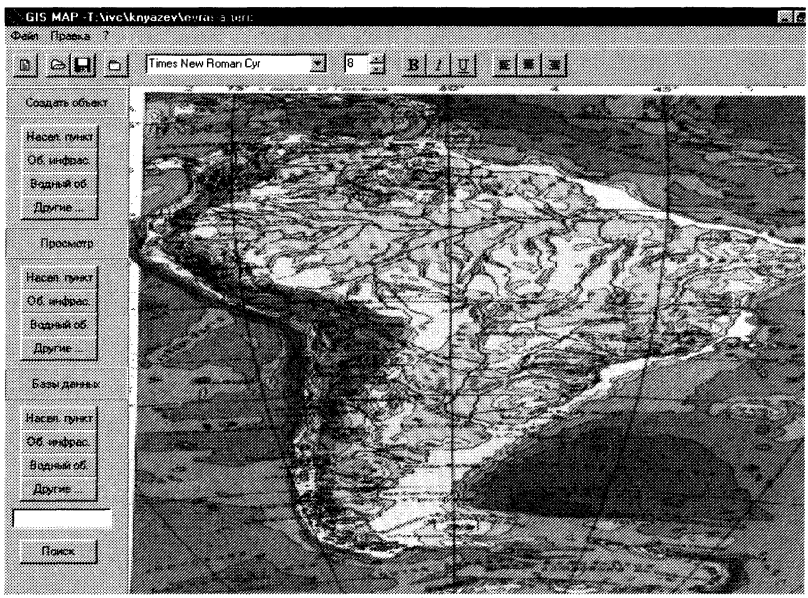
Вариант 24

Московский государственный медико-стоматологический университет.

Лабораторная работа № 2. Геоинформационная система Geo-Perm 2000

Время выполнения 6 — 8 часов.

1. Откройте созданный вами в упражнениях файл ***evrasia.gsm.***
2. Создайте новые объекты (кнопкой *Другие)*: Россия, Польша, Украина.
3. Определите, каким цветом выделяются эти объекты на карте.
4. Внесите в базу данных сведения о созданных объектах.
5. Осуществите поиск *водного объекта* ***Кама*** и отредактируйте информацию о нем.
6. Сохраните файл.
7. Создайте новую ГИС по карте, указанной на рисунке (файл jg_amer.bmp):



8. Создайте любой файл с объектами инфраструктуры.

Приложение: справочные материалы

Города

Название	Страна	Область	Район	Площадь	Численность нас.	Развитые промышленные отрасли	Разное
Москва	Россия	Московская	—	878,7 км^2	8015 тыс.	Легкая, пищевая, металлург. пром-ть	

Название	Страна	Область	Район	Площадь	Численность нас.	Развитые промышленные отрасли	Разное
Пермь	Россия	Пермская	—		1018 тыс.	Машин-ие, нефте- и газопереработ. пром-ть	
Санкт-Петербург	Россия	Ленинградская	—		4156 тыс.	Машин-ие, черная и цветная металлург.	
Варшава	Польша			0,5 тыс.км^2	2,1 млн	Машин-ие, металлообработка, пищевая, швейная пром-ть	

Водные объекты

Название	Длина, км	Площадь тыс. км^2	Наличие шлюзов	Судоходность	Наибольшая глуб., м	Водохранилища	Страны	Разное
р. Кама	1805	басс. 507	Есть	Есть		Камское	Россия	
р. Волга	3534	басс. 1360	Есть	Регулярное судоходство		Волжское	Россия	
р. Обь	3650	басс. 2990	Есть	Есть		Новосибирское	Россия	
оз. Онежское		9720		Есть	127	Свирское	Россия	С островами

Страны

Название	Площадь тыс. кв. км	Столица	Крупнейший нас. пункт	Наивысшая точка	Официальный язык	Госп. религии	Денежн. единица	Гос. строй
Россия	17 075	Москва	Санкт-Петер.	Эльбрус (5642 м)	Русский	Христианство, ислам, иудаизм	Рубль	Федеративная республика

Название	Площадь тыс.кв. км	Столица	Крупнейший нас. пункт	Наивысшая точка	Официальный язык	Госп. религии	Денежн. единица	Гос. строй
Украина	603,7	Киев		Говерла (2061 м)	Украинский	Христианство	Гривна	Республика
Польша	312	Варшава	Краков	Рысы (2499 м)	Польский	Христианство	Злотый	Многопартийная республика
Перу	22,454	Лима	Кальяло	Уаскаран (6768 м)	Испанский, кечуа	Христианство	Новый соль	Многопартийная республика
Чили	756	Сантьяго	Консельпсон	Охос-дель-Саладо (6880 м)	Испанский	Христианство	Чилийское песо	Республика

Дополнительная литература

1. *Герман О. В.* Введение в теорию экспертных систем и обработку знаний. — Минск: Дизайн-ПРО, 1995.
2. *Глушков В. М.* Основы безбумажной информатики. 2-е изд. — М.: Наука, 1987.
3. *Гохман В., Андрианов В.* Обзор по ГИС (СП Дата+, 1999, Internet).
4. *Гриценко В. Н., Паньшин Б. Н.* Информационная технология: Вопросы развития и применения. — Киев: Наук. думка, 1988.
5. *Коновалов Н. В., Капралов Е. Г.* Введение в ГИС. — Петрозаводск: Изд-во Петрозав. ун-та, 1995.
6. *Корячко В. П., Курейчик В. М., Норенков И. П.* Теоретические основы САПР. — М.: Энергоатомиздат, 1987.
7. *Кошкарев А. В., Тикунов В. С.* Геоинформатика. — М.: Картгеоцентр «Геоиздат», 1993.
8. *Овчинников В. Г.* Автоматизированные системы информационного обеспечения. — М.: Энергия, 1977.
9. *Свириденко С. С.* Современные информационные технологии. — М.: Радио и связь, 1989.
10. *Сербенюк С. Н.* Картография и геоинформатика — их взаимодействие. — М.: Изд-во МГУ, 1990.
11. Системы автоматизированного проектирования: Пер. с англ. / Под ред. Д. А. Аветисяна. — М.: Мир, 1985.
12. Системы автоматизированного проектирования / Под ред. Дж. Аллана. — М.: Мир, 1985.
13. Системы автоматизированного проектирования / Под ред. И. П. Норенкова. — Минск: Вышэйш. шк., 1997.
14. *Советов Б. Я.* АСУ. Введение в специальность. — М.: Высш. шк., 1989.
15. *Советов Б. Я.* Информационная технология. — М.: Высш. шк., 1992.
16. Справочник по САПР / Под ред. В. И. Скурихина. — Киев: Техника, 1988.

17. *Трофимов А.М., Панасюк М.В.* Геоинформационные системы и проблемы управления окружающей средой. — Казань: Изд-во Казан. ун-та, 1984.
18. *Фурунжиев Р.И., Гугля В.А.* САПР, или Как ЭВМ помогает конструктору. — Минск: Вышэйш. шк., 1997.
19. *Цветков В.Я.* Геоинформационные системы и технологии. — М.: Финансы и статистика, 1998.

Тесты к главе 6

Банки информации

1. Основой банка данных является:
   1) база данных; 2) совокупность информационных документов;
   3) СУБД; 4) система хранения данных.
2. Классы банков информации по назначению:
   1) информационно-справочные системы, БД в автоматизированных системах управления, БД в системах автоматизации научных исследований;
   2) БД в системах поиска, БД в системах управления;
   3) БД в пользовательских системах, БД в системах управления, БД информационных систем;
   4) исследовательские системы, информационные системы, логические системы.
3. Основные типы банков информации по режиму функционирования:
   1) пакетный, диалоговый и смешанный тип;
   2) логический, информационный и смешанный;
   3) пакетный и логический;
   4) информационный, диалоговый и логический.
4. По архитектуре вычислительной среды различают следующие банки информации:
   1) логические и диалоговые;
   2) централизованные и распределенные;
   3) распределенные и автоматизированные;
   4) автоматизированные и диалоговые.
5. Предварительная обработка документа для его размещения в банк данных называется:
   1) кодированием; 2) индексацией;
   3) автоматизацией; 4) поисковой интерпретацией.

Базы данных в структуре информационных систем

1. Основные типы моделей данных:
   1) логический, физический;
   2) иерархический, эмпирический, физический;
   3) сетевой, иерархический, реляционный;
   4) реляционный, физический, логический.
2. Процесс построения концептуальной модели разделяется на следующие этапы:
   1) сбор и дублирование информации, кодирование;
   2) сбор и априорный анализ информации, концептуальный анализ данных и синтез концептуальной модели;

3) сбор информации, организация ПО (предметной области);
4) определение сферы применения БД, организация данных.

3. При проектировании БД в первую очередь необходимо определить:
1) способ интерпретации отчетов;
2) реализацию операций обработки и управления;
3) структуру данных и их отношения;
4) ключевые поля.

4. Как представлена информация в реляционной БД:
1) в виде списка;
2) в виде совокупности прямоугольных таблиц;
3) блоками; 4) в виде совокупности файлов?

5. Поле в реляционной БД — это:
1) единица информации;
2) совокупность связанных по какому-либо признаку записей;
3) наименьший поименованный элемент информации;
4) совокупность записей, соответствующих одному объекту.

6. Определение понятия «запись» в реляционной БД:
1) наименьший поименованный элемент информации;
2) совокупность полей, соответствующих одному объекту;
3) совокупность элементов файлов;
4) совокупность связанных по какому-либо признаку списков.

7. Как различаются поля в БД:
1) по типам; 2) по количеству информации;
3) по доменам; 4) по алфавиту?

8. Каждое поле в БД может быть отнесено к одному из следующих типов:
1) символьный, лексический, цифровой тип данных;
2) логический, символьный, числовой, тип даты и тип примечаний;
3) лексический, конкурентный, логический, физический;
4) лексический, символьный, конкурентный, тип примечаний и логический.

9. Поле логического типа содержит:
1) величины, принимающие значения «истинно» или «ложно»;
2) логические высказывания;
3) суть логических рассуждений;
4) логические знаки.

10. Что представляет собой символьное поле:
1) текст, выровненный по левому краю, без выполнения вычислений;
2) числа, выровненные по правому краю;
3) содержит величины, принимающие значения «истинно» или «ложно»;
4) реальные даты?

11. Для чего служит поле типа *Memo*:
1) для хранения секретной информации;
2) для реализации других полей;
3) для хранения больших массивов текстовой информации в отдельном файле;
4) для графики?

12. Какие связи допускают многотабличные БД:
1) один — ко всем;
2) все — ко всем;
3) один — к одному;
4) все перечисленные в 1, 2, 3?

Автоматизированные информационные системы

1. АСУ (автоматизированные системы управления) — это:
   1) комплекс технических и программных средств, обеспечивающий управление объектом в производственной, научной или общественной жизни;
   2) робот-автомат;
   3) компьютерная программа на рабочем столе руководителя завода;
   4) система принятия управленческих решений с привлечением компьютера.
2. Современный принцип построения информационных систем управления:
   1) совершенствование математических моделей системы;
   2) распределенность информационных ресурсов и технология «клиент — сервер»;
   3) персонализация и автоматизация рабочего места;
   4) массовая разработка прикладных программ для управленческого персонала.
3. АСНИ (автоматизированная система научных исследований) — это:
   1) программно-аппаратный комплекс, связанный с экспериментальными установками;
   2) комплекс программ для проведения расчетов научного характера;
   3) компьютерная программа на рабочем столе научного работника;
   4) комплекс программ для проведения компьютерного моделирования.
4. САПР (система автоматизированного проектирования) — это:
   1) программы типа AUTOCAD;
   2) программно-аппаратный комплекс моделирования объектов предметной области;
   3) комплекс программ компьютерной графики для инженера-проектировщика;
   4) компьютерная программа на рабочем столе конструктора.
5. ГИС (геоинформационные системы) — это:
   1) информационные системы в предметной области — география;
   2) системы, содержащие топологические базы данных на электронных картах;
   3) электронные географические карты;
   4) глобальные фонды и архивы географических данных.

Экспертные системы

1. Экспертные системы по своей сути — это:
   1) операционные системы; 2) системы программирования;
   3) системы искусственного интеллекта; 4) авторские системы.
2. Описанием ситуации по информации, поступающей от датчиков, занимаются экспертные системы категории:
   1) диагностики; 2) проектирования; 3) наблюдения; 4) интерпретации.
3. Экспертная система MYCIN предназначена:
   1) для медицинской диагностики;
   2) для определения структуры химического вещества;
   3) для поиска месторождений полезных ископаемых;
   4) для анализа цепей.
4. Экспертная система DENDRAL предназначена:
   1) для медицинской диагностики;
   2) для определения структуры химического вещества;
   3) для поиска месторождений полезных ископаемых;

4) для анализа цепей.

5. Экспертная система PROSPECTOR предназначена:
   1) для медицинской диагностики;
   2) для определения структуры химического вещества;
   3) для поиска месторождений полезных ископаемых;
   4) для анализа цепей.
6. Выберите экспертную систему, занимающуюся построением экспертных систем:
   1) EXPERT; 2) ROSIE; 3) TEIRESIAS; 4) MOLGEN.
7. Выберите экспертную систему, занимающуюся обучением электронике:
   1) EXPERT; 2) MOLGEN; 3) SOPHIE; 4) GUIDON.
8. Выберите экспертную систему, занимающуюся медицинской диагностикой:
   1) PUFF; 2) EL; 3) MOLGEN; 4) MYCIN.

Правильные ответы

Банки информации

№	1	2	3	4
1	X			
2	X			
3	X			
4		X		
5		X		

Базы данных в структуре информационных систем

№	1	2	3	4	№	1	2	3	4
1			X		7	X			
2		X			8		X		
3			X		9	X			
4		X			10	X			
5			X		11			X	
6		X			12				X

Автоматизированные информационные системы

№	1	2	3	4
1	X			
2		X		
3	X			
4		X		
5		X		

Экспертные системы

№	1	2	3	4
1			X	
2				X
3	X			
4		X		
5			X	
6		X		
7			X	
8				X

Глава 7

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

§ 1. ВВЕДЕНИЕ В КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

Тематика и уровень лабораторных работ, предлагаемых в данном разделе, соответствуют гл. 7 базового учебного пособия. Теоретический материал, необходимый для выполнения этих работ, можно найти как в этом пособии, так и в указанной в нем дополнительной литературе.

Цели выполнения работ данного раздела:

- выработка и закрепление практических навыков в освоении методологии компьютерного математического моделирования;
- практическая реализация межпредметных связей;
- освоение элементов самостоятельной научно-исследовательской работы;
- укрепление навыков программирования при реализации практически значимых задач;
- освоение специальных приемов программирования, связанных с моделированием.

Рекомендации по проведению занятий по компьютерному моделированию

Особенность большинства работ данного раздела — отсутствие полных инструкций о ходе выполнения работы и возможность для студента проявить значительную самостоятельность, уточнить (с помощью преподавателя или самостоятельно) постановку задачи, выбрать метод реализации модели, форму представления результатов и т.д. Это придает работам исследовательский характер. Каждую работу можно рассматривать как небольшой проект.

Выполнение работ данного раздела опирается на математический аппарат, входящий в стандартный курс «Численные методы». Задачами студента являются выбор адекватного метода (здесь вполне уместно использование библиотеки стандартных математических программ) и получение достоверного результата с контролем его точности.

Первостепенную важность при выполнении работ по моделированию имеет форма представления результатов. До начала выполнения каждой работы необходимо спроектировать (возможно, с помощью преподавателя) интерфейс пользователя моделирующей программы. Идеальным является наличие нескольких видов отображения результатов моделирования: численного, табличного, графического, динамического, звукового сопровождения и т.д. Некоторые требования по форме представления результатов указаны в инструкциях к работам. Эти требования мо-

гут быть дополнены и конкретизированы преподавателями, проводящими занятия; все остальное — на усмотрение студентов.

Важной частью каждой работы является отчет. Он должен быть выполнен в стиле, приближенном к стандартному стилю научно-технического отчета.

Обязательными частями отчета являются:

- постановка задачи;
- математическая модель;
- описание метода исследования модели;
- программа для ЭВМ;
- описание тестирования программы;
- результаты (в различных формах представления);
- содержательный анализ результатов.

Выполнение всех приведенных ниже работ в полном объеме подразумевает выделение на лабораторные работы порядка 100 часов (аудиторных и внеаудиторных вместе). Оценка исходит из практического опыта их реализации и из того, что студенты:

- предварительно подготовились к выполнению работы, освоили соответствующий теоретический материал;
- имеют практически завершенную математическую модель процесса;
- достаточно свободно владеют математическими методами, необходимыми для выполнения данной работы;
- имеют устойчивые навыки программирования и/или использования необходимых для выполнения данной работы программных средств.

Рекомендации по программному обеспечению при проведении занятий по компьютерному моделированию

При проведении лабораторных работ по компьютерному математическому моделированию можно опираться на различные виды программного обеспечения.

1. Трансляторы с языков высокого уровня.

Соответствующий способ проведения занятий ориентирован на активно программирующих студентов и позволяет наряду с отработкой навыков моделирования углубить программистскую подготовку. Недостаток этого способа — относительно высокая трудоемкость, особенно если речь идет об оформлении диалогового интерфейса, адекватного современным требованиям, предъявляемым к прикладным программам. Этот недостаток может быть устранен, если наряду с языком (типа Паскаль) использовать современные средства визуального программирования (типа Delphi).

2. Офисные пакеты (текстовый редактор и электронные таблицы).

С помощью электронных таблиц (ЭТ) можно произвести моделирование большей части процессов, описанных в данной главе. Текстовый редактор позволит сделать отчет, в который программы, составленные с помощью ЭТ, и результаты моделирования (численные и графические) войдут органично. Недостаток этого способа — не всегда удобно реализовывать достаточно сложные вычислительные алгоритмы в ЭТ.

3. Специальные пакеты для решения математических задач.

Программы типа «MathCad», «Mathematica» и им подобные позволяют обойти трудность, связанную с программированием математических алгоритмов и (частично) представлением результатов моделирования. Это является одновременно и недостатком, так как снижает образовательный эффект от занятий.

4. Специальные пакеты для моделирования соответствующих типов процессов. К примеру, созданная в ПГПУ среда Model Designer позволяет моделировать процессы, описываемые обыкновенными дифференциальными уравнениями (скрывая детали их решения), и отображать результаты моделирования в табличной и графической формах. Подобный способ — самый простой, но высказанное в предыдущем пункте замечание применимо к нему в еще большей мере.

В любом случае работы рассчитаны на самостоятельное выполнение студентами, включая разработку программ, их отладку и тестирование. Выбор программного средства — в руках преподавателей и студентов. Наилучшее решение — использование каждым студентом в ходе реализации практикума нескольких программных средств.

Темы для рефератов

1. Моделирование как метод познания.
2. Информационное моделирование.
3. Компьютерное моделирование физических процессов.
4. Компьютерное моделирование в биологии и экологии.
5. Компьютерное моделирование в химии.
6. Математические методы в медицине.

Тема семинарских занятий

Основные приемы и методы компьютерного математического моделирования.

Дополнительная литература

1. *Белошапка В.К.* Информационное моделирование в примерах и задачах. — Омск: ОГПИ, 1992.
2. *Бейли Н.* Математика в биологии и медицине. — М.: Мир, 1970.
3. *Бейли Н.* Статистические методы в биологии. — М.: ИЛ, 1962.
4. *Беллман Р.* Математические методы в медицине. — М.: Мир, 1987.
5. *Бусленко Н.П.* Моделирование сложных систем. — М.: Наука, 1978.
6. *Веденов А.А.* Моделирование элементов мышления. — М.: Наука, 1988.
7. *Геронимус Ю.В.* Игра. Модель. Экономика. — М.: Знание, 1989.
8. *Глинский Б.А. и др.* Моделирование как метод научного исследования. — М.: Изд-во МГУ, 1965.
9. *Горстко А.Б.* Познакомьтесь с математическим моделированием. — М.: Знание, 1991.
10. Колебания и бегущие волны в химических системах / Под ред. Р.Филд и М.Бургер. — М.: Мир, 1988.
11. Компьютеры, модели, вычислительный эксперимент: Введение в информатику с позиций математического моделирования. — М.: Наука, 1988.
12. *Кунин С.* Вычислительная физика. — М.: Мир, 1992.
13. *Марчук Г.И.* Математическое моделирование в иммунологии. — М.: Наука, 1991.
14. Математическое моделирование / Под ред. Дж.Эндрюса, Р.Мак-Лоуна. — М.: Мир, 1979.

15. *Могилев А.В., Злотникова И.Я.* Элементы математического моделирования. — Омск: ОмГПУ, 1995.
16. *Морозов К.Е.* Математическое моделирование в научном познании. — М.: Мысль, 1965.
17. *Пак Н.И.* Компьютерное моделирование в примерах и задачах. — Красноярск: КрГПУ, 1994.
18. *Пейре Р., Тейлор Т.Д.* Вычислительные методы в задачах механики жидкостей. — Л.: Гидрометеоиздат, 1986.
19. *Поттер Д.* Вычислительные методы в физике. — М.: Мир, 1975.
20. *Романовский Ю.М., Степенов И.В., Чернавский Д.С.* Что такое математическая биофизика? — М.: Просвещение, 1971.

§ 2. МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ

Краткие сведения

Движение тел в среде с учетом трения

Второй закон Ньютона. В рассматриваемых ниже физических задачах фундаментальную роль играет второй закон Ньютона. Он гласит, что ускорение, с которым движется тело, прямо пропорционально действующей на него силе (если их несколько, то равнодействующей, т.е. векторной сумме сил) и обратно пропорционально его массе: $\vec{a} = \dfrac{\vec{F}}{m}$.

Чтобы исследовать ситуации, когда сила или масса не являются величинами постоянными, необходимо записать этот закон в более общей математической форме.

Допустим, что сила или масса (или и то, и другое) непостоянны и заданным образом зависят от времени, скорости движения или перемещения: $\vec{F}(t, v, s)$ и $m(t, v, s)$. Достаточно наличия хотя бы одной из указанных зависимостей, чтобы ускорение было величиной переменной. В этом случае приведенная выше формула определяет его значение в тот момент времени, которому соответствуют сила и масса. Реальный интерес представляет временная зависимость перемещения $\vec{s}(t)$ и скорости $\vec{v}(t)$. Поскольку ускорение есть приращение скорости, а скорость — приращение перемещения, то

$$\vec{a}(t) = \frac{d\vec{v}}{dt}, \quad \vec{v}(t) = \frac{d\vec{s}}{dt}, \tag{7.1}$$

а сам второй закон Ньютона приобретает вид

$$\frac{d^2\vec{s}}{dt^2} = \frac{\vec{F}(t, \vec{v}, \vec{s})}{m(t, \vec{v}, \vec{s})}, \tag{7.2}$$

или, что то же самое,

$$\frac{d\vec{s}}{dt} = \vec{v}(t), \quad \frac{d\vec{v}}{dt} = \frac{\vec{F}(t, \vec{v}, \vec{s})}{m(t, \vec{v}, \vec{s})}. \tag{7.3}$$

Еще раз подчеркнем, что совсем необязательно чтобы сила и/или масса зависели каждая от трех указанных переменных. Чаще всего в конкретных задачах присутствует в явном виде одна из указанных зависимостей.

Произведем дискретизацию по времени простейшим возможным способом. Если в некоторый момент времени t_0 величина s имеет значение s_0, а величина v — значение v_0, то в некоторый последующий момент времени $t_1 = t_0 + \Delta t$ будем иметь

$$\vec{v}(t_1) \approx \vec{v}(t_0) + \vec{a}(t_0) \cdot \Delta t = \vec{v}(t_0) + \frac{\vec{F}_0}{m_0} \cdot \Delta t, \quad \vec{s}(t_1) \approx \vec{s}(t_0) + \vec{v}(t_0) \cdot \Delta t. \qquad (7.4)$$

Здесь индекс «0» означает величины в начальный момент времени.

При вычислениях значений $\vec{v}$ и $\vec{s}$ в последующие моменты времени можно поступать аналогично (7.4). Так, если известны значения $\vec{v}_i$ и $\vec{s}_i$ в момент t_i, то

$$\vec{v}_{i+1} \approx \vec{v}_i + \frac{\vec{F}_i}{m_i} \cdot \Delta t, \quad \vec{s}_{i+1} \approx \vec{s}_i + \vec{v}_i \cdot \Delta t. \qquad (7.5)$$

Вопрос о выборе конкретного значения Δt весьма непрост и определяется следующими соображениями. При компьютерном моделировании можно получить решение задачи о движении тела на некотором конечном отрезке времени $[t_0, T]$. Чем больше величина Δt, тем:

а) меньше вычислений требуется для того, чтобы пройти весь заданный временной интервал;

б) меньшая точность в передаче значений непрерывных функций $\vec{s}(t)$ и $\vec{v}(t)$ их дискретными представлениями — наборами чисел $\vec{s}_i$ и $\vec{v}_i$.

Вопрос о точности результатов является в описываемом моделировании одним из центральных. Он распадается на два: как оценить эту точность и можно ли, уменьшая Δt, достигать все большей точности?

Остановимся вначале на первом. Формулы (7.4), (7.5) представляют собой применение метода Эйлера для приближенного решения системы дифференциальных уравнений (7.3). Наиболее приемлемой при использовании этого и родственного ему методов (например, Рунге — Кутта) является эмпирическая оценка точности. Для этого отрезок $[t_0, T]$ проходится с некоторым шагом Δt, а затем с существенно меньшим (например, в два раза) шагом. Сравнение результатов в точках $t_1, t_2, \ldots, T$ позволяет составить представление о реальной точности результатов. Если она недостаточна, то следует повторить процесс с еще меньшим шагом.

Однако уменьшение шага Δt не всегда ведет к улучшению результатов моделирования. Одна из причин заключается в том, что чем меньше шаг, тем больше арифметических действий, ведущих к увеличению глобальной погрешности округления. Другая причина глубже и связана со способом дискретизации— перехода от описания реально непрерывного процесса движения тел к описанию по простейшим формулам (7.4), (7.5). Обе причины могут привести к неустойчивости решения, т.е. к получению результатов, не имеющих реально ничего общего с истинными. Обычно неустойчивость становится заметной при повторениях процесса с уменьшением шага Δt.

Более эффективными при моделировании процессов, описываемых дифференциальными уравнениями, являются методы Рунге — Кутта более высокого порядка аппроксимации, чем метод Эйлера, неявные методы, методы типа «предиктор—корректор», отличающиеся повышенной устойчивостью, и другие, описанные в специальной литературе.

Сила сопротивления. В ряде представленных ниже задач необходимо знать, от чего зависит сила сопротивления при движении в среде. При реальных физических движениях тел в газовой или жидкостной среде трение сильно влияет на характер движения.

Соответствующие закономерности носят эмпирический характер и отнюдь не имеют столь строгой и четкой формулировки, как второй закон Ньютона. При относительно малых скоростях величина силы сопротивления пропорциональна скорости и имеет место соотношение

$$F_{\text{сопр}} = k_1 v_1, \tag{7.6}$$

где k_1 определяется свойствами среды и формой тела. Например, для шарика $k_1 = 6\pi\mu r$ — так называемая формула Стокса, где μ — динамическая вязкость среды, r — радиус шарика. Так, для воздуха при $t = 20\,^{\circ}\text{C}$ и давлении 1 атм $\mu = 0{,}0182 \frac{\text{Н} \cdot \text{с}}{\text{м}^2}$, для воды $\mu = 1{,}002 \frac{\text{Н} \cdot \text{с}}{\text{м}^2}$, для глицерина $\mu = 1480 \frac{\text{Н} \cdot \text{с}}{\text{м}^2}$.

При более высоких скоростях сила сопротивления становится пропорциональной квадрату скорости:

$$F_{\text{сопр}} = k_2 v^2. \tag{7.7}$$

Разумеется, линейная по скорости часть силы сопротивления формально также сохранится, но если $k_2 v^2 \gg k_1 v$, то вкладом $k_1 v$ можно пренебречь. О величине k_2 известно следующее: она пропорциональна площади сечения тела S, поперечного по отношению к потоку, плотности среды $r_{\text{среды}}$ и зависит от формы тела. Обычно представляют

$$k_2 = \frac{1}{2} c S \rho_{\text{среды}}, \tag{7.8}$$

где c — безразмерный коэффициент лобового сопротивления (рис. 7.1).

При достижении достаточно большой скорости, когда образующиеся за обтекаемым телом вихри газа или жидкости начинают интенсивно отрываться от тела, значение c в несколько раз уменьшается; для шара оно становится приблизительно равным 0,1.

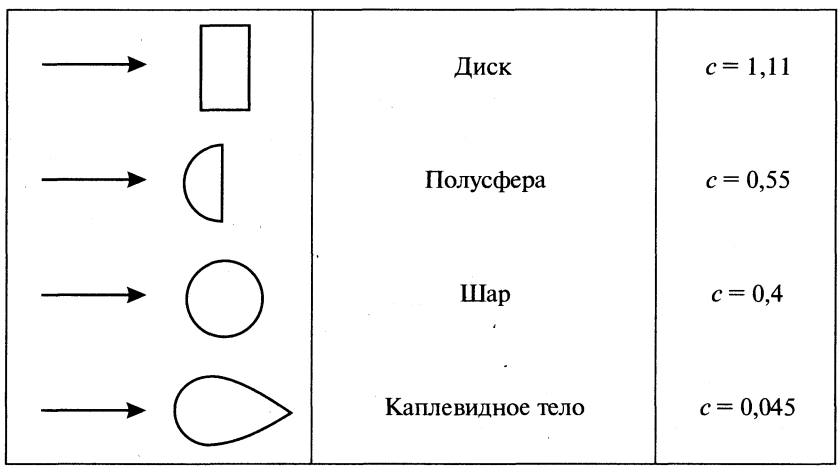


Рис. 7.1. Значения коэффициента лобового сопротивления для некоторых тел, поперечное сечение которых имеет указанную на рисунке форму

Свободное падение тела. Математическая модель свободного падения тела — уравнение второго закона Ньютона с учетом двух сил, действующих на тело — силы тяжести и силы сопротивления среды. Движение является одномерным; проецируя силу, скорость и перемещение на ось, направленную вертикально вниз, из (7.3) получаем

$$\begin{cases} \dfrac{dh}{dt} = v, \\ \dfrac{dv}{dt} = \dfrac{mg - k_1 v - k_2 v^2}{m}. \end{cases} \tag{7.9}$$

В конкретных задачах можно одной из составляющих силы сопротивления пренебречь (если она заведомо много меньше другой).

Частичное тестирование моделирующей программы можно провести для движения без трения. Аналитическое решение в этом случае общеизвестно.

Входные параметры модели:

- начальная высота тела;
- начальная скорость тела;
- величины, определяющие коэффициенты сопротивления среды k_1 и k_2.

Взлет ракеты. Построим простейшую модель вертикального взлета ракеты, приняв гипотезу, что ее масса уменьшается во время взлета по линейному закону:

$$m(t) = \begin{cases} m_0 - \alpha t, & \text{если } m(t) \le m_{\text{кон}}, \\ m_{\text{кон}}, & \text{если } m(t) = m_{\text{кон}}. \end{cases} \tag{7.10}$$

Силу тяги двигателя будем считать постоянной на всем участке взлета.

Однако при самом простом моделировании данного процесса необходимо принять во внимание, что плотность воздуха ρ, входящая в коэффициент k_2, убывает по мере подъема ракеты по закону $\rho = \rho_0 \cdot 10^{-\beta h}$, где h — высота, $\beta \approx 5{,}6 \cdot 10^{-5}\ \text{м}^{-1}$ — иначе модель будет совершенно неадекватна реальности. Таким образом, модель будет описываться системой двух дифференциальных уравнений для функций $v(t)$ и $h(t)$:

$$\begin{cases} \dfrac{dh}{dt} = v, \\ \dfrac{dv}{dt} = \dfrac{F_{\text{тяги}} - m(t) c \rho_0 10^{-\beta h} s v^2(t)}{m(t)}. \end{cases} \tag{7.11}$$

Входные параметры модели:

- m_0 — начальная масса ракеты, заправленной топливом;
- $m_{\text{кон}}$ — остаточная масса после полного выгорания топлива;
- α — расход топлива;
- величины, определяющие k_2, — коэффициент сопротивления воздуха (линейной составляющей силы сопротивления можно заведомо пренебречь);
- $F_{\text{тяги}}$ — сила тяги двигателя (принять постоянной).

Движение тела, брошенного под углом к горизонту. Дифференциальные уравнения модели получаются из второго закона Ньютона проецированием скорости и перемещения на горизонтальную и вертикальную оси координат:

$$\begin{cases} \dfrac{dv_x}{dt} = \dfrac{k_1 + k_2\sqrt{v_x^2 + v_y^2}}{m} v_x, \\ \dfrac{dv_y}{dt} = \dfrac{k_1 + k_2\sqrt{v_x^2 + v_y^2}}{m} v_y, \\ \dfrac{dx}{dt} = v_x, \\ \dfrac{dy}{dt} = v_y. \end{cases} \tag{7.12}$$

Входные параметры модели:
- m — масса тела;
- v — начальная скорость;
- α — угол начального наклона вектора скорости к горизонту;
- величины, определяющие коэффициенты сопротивления среды k_1 и k_2.

Контрольные вопросы

1. Каковы альтернативные формы записи второго закона Ньютона?
2. Как связаны сила трения при движении тела в среде со скоростью движения при относительно небольших (дозвуковых) скоростях?
3. Как (качественно) меняется сила трения со скоростью при околозвуковых скоростях движения?
4. При каких значениях скорости становятся равными линейная и квадратичная составляющие силы сопротивления при падении шарика диаметром 5 см:
а) в воде; б) в керосине; в) в глицерине?

Тема для рефератов

Виды трения. Трение покоя и трение движения. Зависимость силы трения от условий движения.

Темы семинарских занятий

1. Основные законы механики движения тела. Движение тел в среде с учетом трения.
2. Обезразмеривание величин, входящих в дифференциальные уравнения, и установление законов подобия.

Лабораторная работа

Общие рекомендации

1. Целесообразно до начала компьютерной реализации модели провести обезразмеривание переменных, входящих в уравнения, выявить безразмерные комбинации параметров модели и дальнейшие действия производить в безразмерных величинах.

2. Необходим контроль точности результатов и устойчивости применяемого численного метода. Для этого достаточно ограничиться эмпирическими приемами (например, сопоставлением решений, полученных с несколькими разными шагами по времени).

3. Целесообразно применять для моделирования стандартные методы интегрирования систем дифференциальных уравнений, описанные в математической литературе. Простейшие методы (метод Эйлера) часто бывают неустойчивы, и их применение ведет к лишнему расходу времени.

4. Результаты моделирования следует выводить на экран компьютера в следующих видах: таблицы зависимостей перемещения и скорости от времени, графики этих зависимостей, траектории. Желательны динамические иллюстрации движения тел (скажем, изображение движений по траекториям в некотором условном масштабе времени через равные промежутки). Уместны звуковые сигналы (одни — в критические моменты для моделируемого движения, другие — через некоторый фиксированный отрезок пройденного пути и т.д.).

5. При выводе результатов в табличном виде следует учитывать, что соответствующий шаг по времени не имеет практически ничего общего с шагом интегрирования и определяется удобством и достаточной полнотой для восприятия результатов на экране. Экран, сплошь забитый числами, не поддается восприятию. Выводимые числа следует разумным образом форматировать, чтобы незначащие цифры практически отсутствовали.

6. При выводе результатов в графической форме графики должны быть построены так, как это принято в математической литературе (с указанием того, какие величины отложены по осям, в каких масштабах и т.д.).

7. Поскольку таблицы, графики и траектории на одном экране обычно не помещаются, удобно сделать меню, в котором пользователь выбирает желаемый в настоящий момент вид представления результатов.

Примерное время выполнения 16 часов.

Задания к лабораторной работе

1) Выписать математическую модель, определить состав набора входных параметров и их конкретные числовые значения.

2) Если моделирование будет производиться в безразмерных переменных (решение — на усмотрение студента и преподавателя), произвести обезразмеривание и найти набор значений безразмерных параметров.

3) Спроектировать пользовательский интерфейс программы моделирования, обращая особое внимание на формы представления результатов.

4) Выбрать метод интегрирования системы дифференциальных уравнений модели, найти в библиотеке стандартных программ или разработать самостоятельно программу интегрирования с заданной точностью.

5) Произвести отладку и тестирование полной программы.

6) Выполнить конкретное задание из своего варианта работы.

7) Качественно проанализировать результаты моделирования.

8) Создать текстовый отчет по лабораторной работе, включающий:

- титульный лист (указать название работы, исполнителя, номер группы и т.д.);
- постановку задачи и описание модели;
- результаты тестирования программы;
- результаты, полученные в ходе выполнения задания (в различных формах);
- качественный анализ результатов.

Варианты заданий

Вариант 1

Парашютист прыгает с некоторой высоты и летит, не открывая парашюта; на какой высоте (или через какое время) ему следует открыть парашют, чтобы иметь к моменту приземления безопасную скорость (не большую 10 м/с)?

Вариант 2

Изучить, как связана высота прыжка с площадью поперечного сечения парашюта, чтобы скорость приземления была безопасной.

Вариант 3

Промоделировать падение тела с заданными характеристиками (массой, формой) в различных вязких средах. Изучить влияние вязкости среды на характер движения. Скорость движения должна быть столь невелика, чтобы квадратичной составляющей силы сопротивления можно было пренебрегать.

Вариант 4

Промоделировать падение тела с заданными характеристиками (массой, формой) в различных плотных средах. Изучить влияние плотности среды на характер движения. Скорость движения должна быть достаточно большой, чтобы линейной составляющей силы сопротивления можно было пренебрегать (на большей части пути).

Вариант 5

Промоделировать движение исследовательского зонда, «выстреленного» вертикально вверх с уровня земли. В верхней точке траектории над зондом раскрывается парашют, и он плавно спускается в точку старта.

Вариант 6

Промоделировать движение исследовательского зонда, «выстреленного» вертикально вверх с летящего над землей самолета. В верхней точке траектории над зондом раскрывается парашют, и он плавно спускается на землю.

Вариант 7

Глубинная бомба, установленная на взрыв через заданное время, сбрасывается со стоящего неподвижно противолодочного корабля. Исследовать связь между глубиной, на которой произойдет взрыв, и формой корпуса (сферической, полусферической, каплевидной и т.д.).

Вариант 8

Глубинная бомба, установленная на взрыв на заданной глубине, сбрасывается со стоящего неподвижно противолодочного корабля. Исследовать связь между временем достижения заданной глубины и формой корпуса (сферической, полусферической, каплевидной и т.д.).

Вариант 9

Провести моделирование взлета ракеты при значениях параметров $m_0 = 2 \cdot 10^7$ кг, $m_{\text{кон}} = 2 \cdot 10^5$ кг, $\alpha = 2 \cdot 10^5$ кг/с, $F_{\text{тяги}} = 4 \cdot 10^8$ Н. Ответить на вопрос, достиг-

нет ли ракета при этих значениях параметров первой космической скорости 7,8 км/с?

Вариант 10

Провести исследование соотношения входных параметров m_0 и $F_{тяги}$, при которых ракета достигнет первой космической скорости (и в соответствующий момент исчерпает горючее). Остальные входные параметры фиксировать произвольно. Построить соответствующую фазовую диаграмму в переменных (m_0, $F_{тяги}$).

Вариант 11

Разработать и исследовать усовершенствованную модель взлета ракеты, приняв во внимание, что реальные космические ракеты обычно двух- и трехступенчатые и двигатели разных ступеней имеют разную силу тяги.

Вариант 12

Промоделировать движение исследовательского зонда, снабженного разгонным двигателем небольшой мощности, «выстреленного» вертикально вверх с уровня земли. В верхней точке траектории двигатель выключается, над зондом раскрывается парашют, и он плавно спускается в точку старта.

Вариант 13

Промоделировать движение исследовательского зонда, снабженного разгонным двигателем небольшой мощности, «выстреленного» вертикально вверх с летящего над землей самолета. В верхней точке траектории над зондом раскрывается парашют, и он плавно спускается на землю.

Вариант 14

Глубинная бомба-торпеда, снабженная разгонным двигателем, установленная на взрыв через заданное время, сбрасывается со стоящего неподвижно противолодочного корабля. Исследовать связь между глубиной, на которой произойдет взрыв, и формой корпуса (сферической, полусферической, каплевидной и т.д.).

Вариант 15

Глубинная бомба-торпеда, снабженная разгонным двигателем, установленная на взрыв на заданной глубине, сбрасывается со стоящего неподвижно противолодочного корабля. Исследовать связь между временем достижения заданной глубины и формой корпуса (сферической, полусферической, каплевидной и т.д.).

Вариант 16

Торпеда, снабженная разгонным двигателем, нацеливается с подводной лодки на стоящий вертикально над ней надводный корабль. Исследовать связь между временем поражения цели и формой корпуса (сферической, полусферической, каплевидной и т.д.).

Вариант 17

Построить траектории и найти временные зависимости горизонтальной и вертикальной составляющих скорости и перемещения для тела массой 1 кг, брошенного под углом 45° к горизонту с начальной скоростью 10 м/с:

1) в воздухе; 2) в воде.

Сравнить результаты с теми, которые получились бы без учета сопротивления среды (последние можно получить либо численно из той же модели, либо аналитически).

Вариант 18

Найти вид зависимости горизонтальной длины полета тела и максимальной высоты траектории от одного из коэффициентов сопротивления среды, фиксировав все остальные параметры. Представить эту зависимость графически и подобрать подходящую аналитическую формулу, определив ее параметры методом наименьших квадратов.

Вариант 19

Разработать модель подводной охоты. На расстоянии r под углом α подводный охотник видит неподвижную акулу. На сколько метров выше нее надо целиться, чтобы гарпун попал в цель?

Вариант 20

Поставить и решить задачу о подводной охоте при дополнительном условии: акула движется.

Вариант 21

Промоделировать движение исследовательского зонда, «выстреленного» под углом к горизонту. В верхней точке траектории над зондом раскрывается тормозной парашют, затем зонд плавно движется до земли.

Вариант 22

Глубинная бомба, установленная на взрыв через заданное время, сбрасывается с движущегося противолодочного корабля. Исследовать связь между глубиной, на которой произойдет взрыв, пройденным расстоянием по горизонтали и формой корпуса (сферической, полусферической, каплевидной и т.д.).

Вариант 23

Глубинная бомба-торпеда, снабженная разгонным двигателем, установленная на взрыв на заданной глубине, сбрасывается с движущегося противолодочного корабля. Исследовать связь между временем достижения заданной глубины, пройденным расстоянием по горизонтали и формой корпуса (сферической, полусферической, каплевидной и т.д.).

Вариант 24

Торпеда, снабженная разгонным двигателем, нацеливается с лежащей на дне подводной лодки на поражение движущегося надводного корабля. Пуск торпеды производится в момент прохождения корабля над лодкой. Исследовать связь между глубиной залегания лодки, временем поражения цели и расстоянием, который корабль успеет пройти по горизонтали.

Дополнительная литература

1. *Архангельский М.М.* Курс физики. Механика. — М.: Просвещение, 1975.

2. *Гулд Х., Тобочник Я.* Компьютерное моделирование в физике: Пер. с англ. Т. 1, 2. — М.: Мир, 1990.

3. *Савельев И.В.* Курс общей физики: В 3 т. Т. 1. — М.: Наука, 1977.

4. *Сивухин Д.В.* Общий курс физики: В 5 т. Т. 1. — М.: Наука, 1974.
5. *Стрелков С.П.* Механика. — М.: Наука, 1975.
6. *Хайкин С.Э.* Физические основы механики. — М.: Наука, 1976.

Краткие сведения

Моделирование движения небесных тел и заряженных частиц

Движение небесных тел. Рассмотрим модель движения космического тела (планеты, кометы, спутника) под действием силы всемирного тяготения в гравитационном поле, создаваемом телом с многократно большей массой.

Входные параметры модели:

- масса «большого» тела;
- начальные координаты «малого» тела, движение которого изучается;
- начальная скорость «малого» тела.

В системе координат, начало которой привязано к «большому» телу, дифференциальные уравнения модели имеют вид

$$\begin{cases} \dfrac{dx}{dt} = v_x, \\ \dfrac{dy}{dt} = v_y, \\ \dfrac{dv_x}{dt} = -GM\dfrac{x}{\sqrt{(x^2+y^2)^3}}, \\ \dfrac{dv_y}{dt} = -GM\dfrac{y}{\sqrt{(x^2+y^2)^3}}. \end{cases} \tag{7.13}$$

Они получаются из второго закона Ньютона и закона всемирного тяготения.

$G = 6{,}67 \cdot 10^{-11}$ м^3/кг с^2 — гравитационная постоянная.

Движение заряженных частиц. Рассмотрим модель движения заряженной частицы в кулоновском поле другой заряженной частицы, положение которой фиксировано.

Входные параметры модели:

- q и Q — соответственно заряды движущейся и закрепленной частиц;
- m — масса движущейся частицы;
- начальные координаты движущейся частицы;
- начальная скорость движущейся частицы.

В системе координат, начало которой привязано к «большому» телу, дифференциальные уравнения модели имеют вид

$$\begin{cases} \dfrac{dx}{dt} = v_x, \\ \dfrac{dy}{dt} = v_y, \\ \dfrac{dv_x}{dt} = -\dfrac{1}{4\pi\varepsilon_0}\cdot\dfrac{Qq}{m}\cdot\dfrac{x}{\sqrt{(x^2+y^2)^3}}, \\ \dfrac{dv_y}{dt} = -\dfrac{1}{4\pi\varepsilon_0}\cdot\dfrac{Qq}{m}\cdot\dfrac{y}{\sqrt{(x^2+y^2)^3}}. \end{cases} \tag{7.14}$$

Эти уравнения получаются из второго закона Ньютона и закона Кулона; $\varepsilon_0 = 0{,}85 \cdot 10^{-12}$ Ф/м — электрическая постоянная. Знак «минус» в двух последних уравнениях соответствует разноименно заряженным частицам; в случае одноименных зарядов он меняется на «плюс».

Взаимные движения разноименно заряженных частиц и движения двух небесных тел качественно очень схожи (это становится совершенно очевидным после обезразмеривания уравнений (7.13) и (7.14)).

Контрольные вопросы

1. Как формулируется закон всемирного тяготения?
2. Как формулируется закон Кулона?

Тема для рефератов

Движение небесных тел. Задача двух тел. Возмущения. Задача трех тел.

Темы семинарских занятий

1. Движение небесных тел. Закон всемирного тяготения. Законы Кеплера.
2. Закон Кулона. Единицы измерения электрических величин. Характеристики электрического поля.

Лабораторная работа

Общие рекомендации

1. Целесообразно до начала компьютерной реализации модели провести обезразмеривание переменных, входящих в уравнения, выявить безразмерные комбинации параметров модели и дальнейшие действия производить в безразмерных величинах.
2. Необходим контроль точности результатов и устойчивости применяемого численного метода. Для этого достаточно ограничиться эмпирическими приемами (например, сопоставлением решений, полученных с несколькими разными шагами по времени).
3. Целесообразно применять для моделирования стандартные методы интегрирования систем дифференциальных уравнений, описанные в математической литературе. Простейшие методы (метод Эйлера) часто бывают неустойчивы, и их применение ведет к лишнему расходу времени.
4. Результаты моделирования следует выводить на экран компьютера в следующем виде: в виде таблиц зависимостей перемещения и скорости от времени, в виде графиков этих зависимостей, в виде траекторий. Желательны динамические иллюстрации движения тел (скажем, изображение движения по траекториям в некотором условном масштабе времени через равные промежутки). Уместны звуковые сигналы (одни — в критические моменты для моделируемого движения, другие — через каждый фиксированный отрезок пройденного пути и т.д.).
5. При выводе результатов в табличном виде следует учитывать, что соответствующий шаг по времени не имеет практически ничего общего с шагом интегрирования и определяется удобством и достаточной полнотой для восприятия ре-

зультатов на экране. Экран, сплошь забитый числами, зрительно плохо воспринимается. Выводимые числа следует разумным образом форматировать, чтобы незначащие цифры практически отсутствовали.

6. При выводе результатов в графической форме графики должны быть построены так, как это принято в математической литературе (с указанием величин, отложенных по осям, их размерности и масштаба и т.д.).

7. Поскольку таблицы, графики и траектории на одном экране обычно не помещаются, удобно сделать меню, в котором пользователь выбирает желаемый в настоящий момент вид представления результатов.

Примерное время выполнения 16 часов.

Задания к лабораторной работе

1) Выписать математическую модель, определить состав набора входных параметров и их конкретные числовые значения.

2) Если моделирование будет производиться в безразмерных переменных (решение — на усмотрение студента и преподавателя), то произвести обезразмеривание и найти набор значений безразмерных параметров.

3) Спроектировать пользовательский интерфейс программы моделирования, обращая особое внимание на форму представления результатов.

4) Выбрать метод интегрирования системы дифференциальных уравнений модели, найти в библиотеке стандартных программ или разработать самостоятельно программу интегрирования с заданной точностью.

5) Произвести отладку и тестирование полной программы.

6) Выполнить конкретное задание из своего варианта работы.

7) Качественно проанализировать результаты моделирования.

8) Создать текстовый отчет по лабораторной работе, включающий:

- титульный лист (название работы, исполнителя, группу и т.д.);
- постановку задачи и описание модели;
- результаты тестирования программы;
- результаты, полученные в ходе выполнения задания (в различной форме);
- качественный анализ результатов.

Варианты заданий

Вариант 1

Найти траекторию полета кометы, залетевшей в Солнечную систему, у которой на расстоянии от Солнца 100 астрономических единиц (1 а.е. = $1{,}50 \cdot 10^{11}$ м — расстояние от Земли до Солнца) скорость (v = 10 км/с) направлена под углом α = = 30° к оси «комета — Солнце». Является ли эта траектория замкнутой? Если «да», то сколько длится для нее период полета?

Вариант 2

В условиях предыдущей задачи подобрать то значение угла α, при котором траектория из незамкнутой превращается в замкнутую (скорость v фиксирована).

Вариант 3

В условиях задачи из варианта 1 подобрать то значение скорости v, при котором траектория из незамкнутой превращается в замкнутую (угол α фиксирован).

Вариант 4

Проверить в компьютерном эксперименте выполнимость второго закона Кеплера, определяющего движение небесных тел по замкнутой траектории.

Вариант 5

Проверить в компьютерном эксперименте выполнимость третьего закона Кеплера, определяющего движение небесных тел по замкнутой траектории.

Вариант 6

Промоделировать траекторию движения малого космического аппарата, запускаемого с борта космической станции, относительно Земли. Запуск осуществляется путем толчка в направлении, противоположном движению станции, по касательной к ее орбите.

Вариант 7

Промоделировать траекторию движения малого космического аппарата, запускаемого с борта космической станции, относительно Земли. Запуск осуществляется путем толчка в направлении, перпендикулярном к плоскости орбиты движения станции.

Вариант 8

Как будет выглядеть полет искусственного спутника Земли, если учесть возмущающее действие Луны?

Вариант 9

Разработать и реализовать модель движения искусственного спутника Земли при учете воздействия на него малой постоянной силы, обусловленной «солнечным ветром». Считать, что плоскость орбиты движения спутника изначально перпендикулярна к «солнечному ветру».

Вариант 10

Считая, что движение Луны вокруг Земли происходит практически по круговой орбите, проанализировать воздействие на эту орбиту со стороны Солнца для малого участка движения, на котором плоскость орбиты перпендикулярна к оси «Солнце—Земля».

Вариант 11

Проанализировать особенности движения искусственного спутника Земли, движущегося практически по круговой орбите на высоте порядка 300 км, связанные с малым сопротивлением атмосферы.

Вариант 12

Проанализировать изменение круговой орбиты астероида, движущегося вокруг Солнца, под влиянием вулканического выброса с его поверхности.

Вариант 13

Найти траекторию движения тела массой 1 г, несущего заряд величиной $q = 1 \cdot 10^{-2}$ К, в поле заряда величиной $Q = 5 \cdot 10^{-2}$ К. Начальное расстояние между

зарядами 1 м, начальная скорость равна $1 \cdot 10^{-1}$ м/с и направлена под углом 30° к оси, соединяющей заряды. Провести моделирование для случая зарядов одного знака.

Вариант 14

В условиях предыдущей задачи провести моделирование для случая зарядов разных знаков.

Вариант 15

Разработать модель движения практически невесомой заряженной частицы в электрическом поле, созданном системой нескольких фиксированных в пространстве заряженных тел, в случае когда заряженные тела находятся в одной плоскости и в ней же находится движущаяся частица.

Вариант 16

То же, что и в предыдущем варианте, но частица находится вне плоскости расположения зарядов; ее начальная скорость перпендикулярна к этой плоскости.

Вариант 17

Имеется неподвижная заряженная частица с зарядом Q и экран (рис. 7.2). В точке A экрана находится мишень. При каких соотношениях величины начальной скорости v_0 движущейся частицы (заряд q) и угла прицеливания α она попадет в мишень? Расстояния обозначены на рисунке. Заряды частиц — разных знаков.

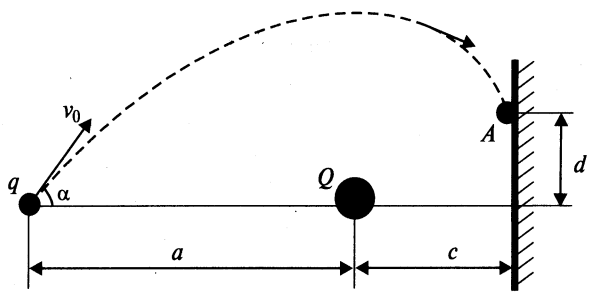


Рис. 7.2. К задаче варианта 17

Вариант 18

То же условие, что и в предыдущей задаче, но расположение частиц и экрана соответствует рис. 7.3; заряды частиц имеют одинаковые знаки.

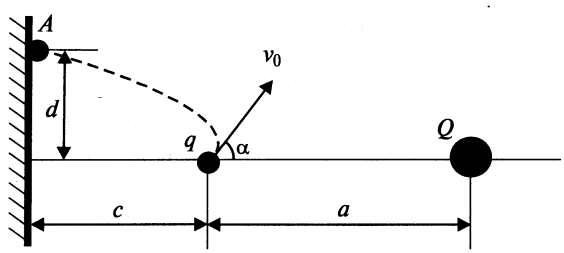


Рис. 7.3. К задаче варианта 18

Вариант 19

Промоделировать движение заряженной частицы между пластинами плоского конденсатора. Поле конденсатора считать однородным, начальная скорость частицы направлена параллельно пластинам. Частицу считать практически невесомой.

Вариант 20

Промоделировать движение легкого (практически невесомого) заряженного тела сферической формы между горизонтальными пластинами плоского конденсатора с учетом сопротивления воздуха, находящегося между пластинами.

Вариант 21

Легкая заряженная частица падает вертикально вниз (под влиянием силы тяжести) на одноименно заряженную пластину (начальная скорость обеспечивает движение вниз независимо от соотношения силы тяжести и силы электростатического отталкивания). Промоделировать движение частицы, считая поле, созданное пластиной, однородным.

Вариант 22

Легкая заряженная частица влетает в однородное поле, созданное горизонтально расположенными пластинами конденсатора. Промоделировать ее траекторию, учитывая силу тяжести и электростатическую силу.

Вариант 23

То же, что и в предыдущем варианте, но пластины конденсатора расположены вертикально.

Вариант 24

То же, что и в варианте 22, то пластины конденсатора расположены наклонно.

Дополнительная литература

1. *Гулд Х., Тобочник Я.* Компьютерное моделирование в физике: Пер. с англ. Т. 1, 2. — М.: Мир, 1990.
2. *Калашников С.Г.* Электричество. — М.: Наука, 1977.
3. *Левантовский В.И.* Механика космического полета. М.: Наука, 1970.
4. *Савельев И.В.* Курс общей физики: В 3 т. Т. 1, 2. — М.: Наука, 1977.
5. *Сивухин Д.В.* Общий курс физики: В 5 т. Т. 1, 3, 5. — М.: Наука, 1974.
6. *Стрелков С.П.* Механика. — М.: Наука, 1975.
7. *Хайкин С.Э.* Физические основы механики. — М.: Наука, 1976.

Краткие сведения

Колебательные процессы

Рассмотрим модель движения математического маятника (рис. 7.4) при произвольном (не малом) начальном угле отклонения.

Поскольку нить подвеса считается нерастяжимой, то у маятника одна степень свободы. Удобно принять за нее угол между нитью подвеса и вертикалью.

Уравнения движения имеют вид

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\sin\theta. \qquad (7.15)$$

В случае **малых колебаний** (колебаний с малой амплитудой) оно превращается в так называемое уравнение малых колебаний, отличающееся от полного заменой в правой части $\sin(\theta)$ на θ . Задача о малых колебаниях имеет простое аналитическое решение

$$\theta = A\cos(\omega t + \varphi), \qquad (7.16)$$

где A — амплитуда колебаний, ω — частота, φ — начальная фаза. A и φ можно выразить через начальные условия — угол θ_0 и скорость v_0:

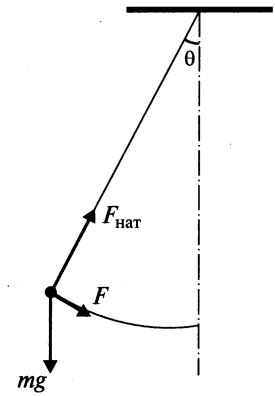


Рис. 7.4. Математический маятник

$$A = \sqrt{\theta_0^2 + \frac{v_0^2}{(l\omega)^2}},\quad \mathrm{tg}(\varphi) = -\frac{v_0}{l\omega\theta_0}. \qquad (7.17)$$

Частота колебаний $\omega = \sqrt{\frac{l}{g}}$; отметим, что она, равно как и период колебаний $T = \frac{2\pi}{\omega}$, в приближении малых колебаний не зависит от начальной амплитуды.

Малые колебания маятника — пример так называемого гармонического движения, описываемого простой тригонометрической функцией (см. (7.16)).

Для численного эксперимента удобно представить уравнение колебаний в форме системы двух уравнений первого порядка:

$$\begin{cases} \frac{d\theta}{dt} = x, \\ \frac{dx}{dt} = -\frac{g}{l}\sin\theta. \end{cases} \qquad (7.18)$$

Входные параметры модели:

l — длина нити подвеса;

θ_0 — начальное отклонение маятника;

x_0 — начальная угловая скорость.

Колебания маятника с трением в точке подвеса описываются следующим уравнением:

$$\frac{d^2\theta}{dt^2} = -2\eta\frac{d\theta}{dt} - \omega^2\sin\theta, \qquad (7.19а)$$

где, как и выше, $\omega = \sqrt{\frac{g}{l}}$, η — коэффициент трения.

Уравнение (7.19а) равносильно системе уравнений

$$\begin{cases} \dfrac{d\theta}{dt} = x, \\ \dfrac{dx}{dt} = -2\eta x - \omega^2 \sin\theta. \end{cases} \qquad (7.19б)$$

Трение приводит, в частности, к тому, что в зависимости от соотношения η и ω появляются разные режимы движения: затухающие колебания и затухание без колебаний. Одна из задач исследования — найти на фазовой плоскости (η, ω) зависимость линии, разделяющей два режима, от начального отклонения маятника.

Входные параметры модели:

ω — частота собственных малых колебаний маятника;

θ_0 — начальное отклонение маятника;

x_0 — начальная угловая скорость;

η — коэффициент трения.

Вынужденные колебания маятника описываются уравнением

$$\frac{d^2\theta}{dt^2} = -2\eta\frac{d\theta}{dt} - \omega^2 \sin\theta + f\cos\lambda, \qquad (7.20а)$$

где f — амплитуда, λ — частота вынуждающей силы.

Уравнение (7.20а) равносильно системе уравнений

$$\begin{cases} \dfrac{d\theta}{dt} = x, \\ \dfrac{dx}{dt} = -2\eta x - \omega^2 \sin\theta + f\cos\lambda. \end{cases} \qquad (7.20б)$$

При $\lambda \approx \omega$ наступает резкое возрастание амплитуды вынужденных колебаний. При $\lambda = \omega$ эта амплитуда в приближении малых колебаний формально бесконечна, однако само приближение при этом не работает.

Вынужденные колебания характеризуются двумя этапами — переходным процессом и стационарными колебаниями с частотой вынуждающей силы. При $\lambda \approx \omega$ переходный процесс сопровождается биениями — особым видом пульсирующих колебаний.

Входные параметры модели:

ω — частота собственных малых колебаний маятника;

θ_0 — начальное отклонение маятника;

x_0 — начальная угловая скорость;

η — коэффициент трения;

f — амплитуда вынуждающей силы;

λ — частота вынуждающей силы.

При **периодическом изменении длины нити подвеса** уравнение колебаний принимает вид

$$\frac{d^2\theta}{dt^2} = -2\eta\frac{d\theta}{dt} - \omega_0^2(1 + \alpha\cos(\lambda t))\sin\theta, \qquad (7.21а)$$

где λ — частота колебаний нити подвеса.

Уравнение (7.21а) равносильно системе уравнений

$$\begin{cases} \dfrac{d\theta}{dt} = x, \\ \dfrac{d^2\theta}{dt^2} = -2\eta x - \omega_0^2(1 + \alpha\cos(\lambda t))\sin\theta. \end{cases} \qquad (7.21б)$$

Одно из принципиальных явлений, связанных с этими колебаниями, — появление так называемого параметрического резонанса при некоторых соотношениях частот λ и ω_0:

$$\lambda \approx \omega_0 / 2,\ \lambda \approx \omega_0,\ \lambda \approx 3\omega_0/2,\ \ldots .$$

Входные параметры модели:
ω_0 — частота собственных малых колебаний маятника;
θ_0 — начальное отклонение маятника;
x_0 — начальная угловая скорость;
η — коэффициент трения;
α — амплитуда модуляции;
λ — частота модуляции.

Контрольные вопросы

1. Как выглядят математические модели следующих движений:
   - колебаний математического маятника без трения;
   - малых колебаний математического маятника без трения;
   - колебаний математического маятника с трением;
   - вынужденных колебаний математического маятника;
   - параметрически возбуждаемых колебаний математического маятника?
2. Как качественно влияет наличие трения на вид колебаний? Являются ли соответствующие колебания гармоническими?
3. В чем заключается процедура Фурье-анализа периодических процессов?

Темы для рефератов

1. Маятники различных видов. Свободные, вынужденные и параметрические колебания.
2. Спектральный анализ периодических процессов.
3. Колебания пружинного маятника.
4. Колебания крутильного маятника.

Тема семинарских занятий

Гармонические колебания. Спектральный анализ периодических процессов.

Лабораторная работа

Общие рекомендации

1. Целесообразно до начала компьютерной реализации модели провести обезразмеривание переменных, входящих в уравнения, выявить безразмерные комби-

нации параметров модели и дальнейшие действия производить в безразмерных величинах.

2. Необходим контроль точности результатов и устойчивости применяемого численного метода. Для этого достаточно ограничиться эмпирическими приемами (например, сопоставлением решений, полученных с несколькими разными шагами по времени).

3. Целесообразно применять для моделирования стандартные методы интегрирования систем дифференциальных уравнений, описанные в математической литературе. Простейшие методы (метод Эйлера) часто бывают неустойчивы, и их применение ведет к лишнему расходу времени.

4. Результаты моделирования следует выводить на экран компьютера в следующем виде: в виде таблиц зависимостей перемещения и скорости от времени, в виде графиков этих зависимостей, в виде траекторий. Желательны динамические иллюстрации движения тел (скажем, изображение движений по траекториям в некотором условном масштабе времени через равные промежутки). Уместны звуковые сигналы (одни — в критические моменты для моделируемого движения, другие — через каждый фиксированный отрезок пройденного пути и т.д.).

5. При выводе результатов в табличном виде следует учитывать, что соответствующий шаг по времени не имеет практически ничего общего с шагом интегрирования и определяется удобством и достаточной полнотой для восприятия результатов на экране. Экран, сплошь забитый числами, зрительно плохо воспринимается. Выводимые числа следует разумным образом форматировать, чтобы незначащие цифры практически отсутствовали.

6. При выводе результатов в графической форме графики должны быть построены так, как это принято в математической литературе (с указанием величин, отложенных по осям, их размерности и масштаба и т.д.).

7. Поскольку таблицы, графики и траектории на одном экране обычно не помещаются, удобно сделать меню, в котором пользователь выбирает желаемый в настоящий момент вид представления результатов.

Примерное время выполнения 16 часов.

Задания к лабораторной работе

1) Выписать математическую модель, определить состав набора входных параметров и их конкретные числовые значения.

2) Если моделирование будет производиться в безразмерных переменных (решение — на усмотрение студента и преподавателя), произвести обезразмеривание и найти набор значений безразмерных параметров.

3) Спроектировать пользовательский интерфейс программы моделирования, обращая особое внимание на формы представления результатов.

4) Выбрать метод интегрирования системы дифференциальных уравнений модели, найти в библиотеке стандартных программ или разработать самостоятельно программу интегрирования с заданной точностью.

5) Произвести отладку и тестирование полной программы.

6) Выполнить конкретное задание из своего варианта работы.

7) Качественно проанализировать результаты моделирования.

8) Создать текстовый отчет по лабораторной работе, включающий:
   - титульный лист (название работы, исполнителя, группу и т.д.);
   - постановку задачи и описание модели;

- результаты тестирования программы;
- результаты, полученные в ходе выполнения задания (в различных формах);
- качественный анализ результатов.

Варианты заданий

Вариант 1

Установить зависимость периода колебаний маятника T от начальной амплитуды в диапазоне амплитуд $\theta_0 \in [0, \pi]$ и его отклонение от периода малых колебаний T_0.

Вариант 2

Установить зависимость периода колебаний маятника T от длины нити подвеса при амплитуде колебаний, равной $\pi/2$.

Вариант 3

Ограничиваясь тремя членами ряда Фурье, исследовать зависимость амплитуд гармоник a_1, a_2 и a_3 от начальной амплитуды колебаний.

Вариант 4

Ограничиваясь тремя членами ряда Фурье, исследовать зависимость амплитуд гармоник a_1, a_2 и a_3 от длины нити подвеса при амплитуде колебаний, равной $\pi/2$.

Вариант 5

Заменить в (7.19) $\sin(\theta)$ на θ и изучить, как трение влияет на малые колебания математического маятника. Фиксировать параметр l и найти то критическое значение коэффициента трения η^*, при котором движение перестает быть колебательным и становится монотонно затухающим (апериодический режим).

Вариант 6

В условиях предыдущей задачи построить зависимость η^* от l при фиксированном значении ω.

Вариант 7

Изучить, как значение начальной амплитуды немалых колебаний математического маятника с трением сказывается на переходе режима затухающих колебаний в режим затухания без колебаний.

Вариант 8

Построить зависимость амплитуды малых колебаний без трения от частоты вынуждающей силы λ при приближении ее к частоте собственных колебаний ω_0.

Вариант 9

Построить зависимость амплитуды немалых колебаний маятника без трения от частоты вынуждающей силы λ при приближении ее к частоте собственных колебаний ω_0.

Вариант 10

Построить зависимость амплитуды немалых колебаний маятника без трения от амплитуды вынуждающей силы при ее частоте, приблизительно равной половине частоты собственных колебаний маятника.

Вариант 11

Получить картину процесса биений в системе с близкими значениями частот λ и ω_0 (в приближении малых колебаний и без наличия трения).

Вариант 12

Получить картину процесса биений в системе с близкими значениями частот λ и ω_0 (для амплитуды колебаний, равной $\pi/2$, и без наличия трения).

Вариант 13

Исследовать, как возрастание коэффициента трения влияет на процесс биений в системе с близкими значениями частот λ и ω_0 (для произвольной амплитуды колебаний).

Вариант 14

Исследовать колебания маятника с периодически меняющейся длиной нити подвеса. Построить на фазовой плоскости (λ/ω_0, α) границы нескольких зон параметрического резонанса (без учета трения).

Вариант 15

В условиях задания из предыдущего варианта исследовать влияние трения на границы нескольких зон параметрического резонанса.

Вариант 16

Построить модель колебаний шарика массы m, висящего на пружинке (пружинного маятника), движущегося под влиянием силы тяжести и упругой силы, без учета трения. Исследовать зависимость периода колебаний маятника от длины l при фиксированном значении параметров m и k (жесткости пружины).

Вариант 17

Для маятника, описанного в предыдущей задаче, исследовать зависимость периода колебаний от массы при фиксированных значениях параметров k и l.

Вариант 18

Для маятника, описанного в варианте 16, добавить учет сопротивления окружающей среды (при конечном размере шарика) и исследовать зависимость периода колебаний от вязкости среды при движении маятника в воде (значения остальных параметров фиксировать). Найти границу перехода периодического движения в апериодическое.

Вариант 19

Для маятника, описанного в варианте 16, добавить учет воздействия периодической вынуждающей силы и исследовать зависимость амплитуды колебаний

от частоты вынуждающей силы при прохождении через резонанс (без учета трения).

Вариант 20

Построить модель колебаний шарика массы m, лежащего на горизонтальной поверхности, под действием пружины, создающей упругую силу $F_{\text{упр}} = -ax - bx^3$, где x — смещение из положения равновесия. Трения не учитывать. Исследовать зависимость периода колебаний такого маятника от параметра b (при фиксированном значении других параметров).

Вариант 21

Для маятника, описанного в предыдущем варианте, добавить учет трения шарика о поверхность (сила трения пропорциональна весу шарика) и исследовать зависимость периода колебаний от коэффициента трения. Найти границу перехода периодического движения в апериодическое.

Вариант 22

Для маятника, описанного в варианте 20, добавить учет наличия вынуждающей периодической силы и исследовать зависимость периода колебаний от амплитуды вынуждающей силы при ее частоте, равной приблизительно половине частоты собственных колебаний (без учета трения).

Вариант 23

Для маятника, описанного в варианте 20, добавить учет наличия вынуждающей периодической силы и исследовать зависимость периода колебаний от частоты вынуждающей силы при прохождении через резонанс (без учета трения).

Вариант 24

Исследовать процесс биений для маятника, описанного в варианте 20, в отсутствии трения.

Дополнительная литература

1. *Мигулин В. В. и др.* Основы теории колебаний. — М.: Наука, 1988.
2. *Савельев И. В.* Курс общей физики: В 3 т. Т. 1, 2. — М.: Наука, 1977.
3. *Сивухин Д. В.* Общий курс физики: В 5 т. Т. 1. — М.: Наука, 1974.
4. *Стрелков С. П.* Механика. — М.: Наука, 1975.
5. *Стрелков С. П.* Введение в теорию колебаний. — М.: Наука, 1964.
6. *Хайкин С. Э.* Физические основы механики. — М.: Наука, 1976.

Краткие сведения

Описание физических процессов в приближении сплошной среды

Наиболее удобные способы наглядного **изображения электрического поля** связаны с двумя взаимодополняющими картинами: силовых линий и линий равного потенциала.

Для построения эквипотенциальных линий (в трехмерном случае — поверхностей) поля, созданного системой зарядов, можно воспользоваться принципом суперпозиции: потенциалы полей, созданных разными зарядами, алгебраически складываются. Поскольку потенциал поля, созданного зарядом q на расстоянии r от него, равен $-\frac{1}{4\pi\varepsilon_0}\frac{q}{r}$, легко определить общий потенциал в любой точке.

В задачах моделирования достаточно стандартная проблема — построение линий (поверхностей), вдоль которых некоторая функция имеет одинаковое значение, называемых изолиниями (изоповерхностями). Это очень распространенная задача визуализации характеристик некоторого скалярного поля в приближении сплошной среды.

Пусть поле создается системой точечных электрических зарядов Q_1, ..., Q_p с координатами соответственно (x_1, y_1), ..., (x_p, y_p). Типичная процедура построения изолиний на экране компьютера состоит в следующем. Выберем по осям x и y некоторые шаги h_x и h_y и покроем плоскость сеткой, образованной прямыми, параллельными осям x и y и отстоящими друг от друга на расстояниях h_x и h_y соответственно. Точки пересечения этих прямых — узлы сетки. Пронумеруем их так: начало координат (0,0), следующий по оси x вправо — (0,1), влево — (0,−1); по оси y вверх — (1,0), вниз (−1,0) и т.д. Значения потенциала, создаваемого системой зарядов Q_1, ..., Q_p в узле (i, k), согласно принципу суперпозиции, таково (обратим внимание, что здесь и ниже i — номер строки, k — номер столбца сетки):

$$\Phi_{ik} = -\sum_{i=1}^{p}\frac{1}{4\pi\varepsilon_0}\frac{q}{r}\frac{Q_1}{\sqrt{(x_i - kh_x)^2 + (y_i - ih_y)^2}}. \qquad (7.22)$$

Ограничимся прямоугольной областью в плоскости xy: $[-mh_x, mh_x]$ по оси x и $[-nh_y, nh_y]$ по оси y. В этой области $(2m + 1)(2n + 1)$ узлов. Вычислим значения потенциала в каждом из них по указанным формулам. В результате получим матрицу значений потенциала.

Фиксируем некоторое значение потенциала $\tilde{\Phi}$ и построим изолинию, соответствующую этому значению. Для этого проходим, к примеру, по i-й горизонтальной линии сетки и ищем среди ее узлов такие соседние значения потенциала, в которых «захватывают» $\tilde{\Phi}$ между собой; признаком этого может служить выполнение неравенства $(\Phi_{ik} - \tilde{\Phi})(\Phi_{i,k+1} - \tilde{\Phi}) < 0$. Если такая пара узлов найдена, то координату точки, в которой $\Phi = \tilde{\Phi}$, найдем приближенно с помощью линейной интерполяции:

$$\begin{aligned} x &= kh_x + \frac{\tilde{\Phi} - \Phi_{ik}}{\Phi_{i,k+1} - \Phi_{ik}}h_x, \\ y &= ih_y. \end{aligned} \qquad (7.23)$$

Найдя в данной горизонтали все такие точки, перейдем к следующей горизонтали, пока не исчерпаем их все. Для этого надо совершить двойной циклический проход: во внешнем цикле перебирать i от $-n$ до $+n$, во внутреннем перебирать k от $-m$ до $+m$.

После этого следует аналогично заняться поиском нужных точек на вертикальных линиях сетки. Детали процедуры очевидны; формулы, аналогичные (7.23), имеют вид

$$x = kh_x,$$
$$y = ih_y + \frac{\tilde{\Phi} - \Phi_{ik}}{\Phi_{i+1,k} - \Phi_{ik}} h_y. \qquad (7.24)$$

После прохождения всех горизонтальных и вертикальных линий сетки находятся все те точки на этих линиях, в которых потенциал равен $\tilde{\Phi}$. Проведя — мысленно или на экране (или на бумаге) — кривую, плавно проходящую через ближайшие точки (прибегая, например, к интерполяции сплайнами), получим искомую изолинию (разумеется, лишь в том случае, если значение $\tilde{\Phi}$ выбрано разумно и такая линия есть в пределах рассматриваемой области). Затем берем другие значения $\tilde{\Phi}$ и повторяем указанную процедуру, получая таким образом семейство изолиний.

Один из способов построения ***объемной*** картины электрического поля состоит в том, чтобы построить системы изолиний в нескольких параллельных равноотстоящих плоскостях для одного и того же набора значений потенциала. Квазитрехмерная картина совокупности указанных плоскостей с изображенными на них изолиниями создает представление об объемной структуре электрического поля.

Для построения изолиний поля, созданного однородно заряженными ***нитями***, ***пластинами***, можно представить их как совокупности большого числа одинаковых «точечных зарядов», в совокупности воспроизводящих форму нити или пластины.

Процесс теплопроводности возникает, если тело неоднородно нагрето. Простейшая для изучения теплопроводности система — линейный однородный стержень (рис. 7.5). В простой модели боковая поверхность стержня считается теплоизолированной, т.е. через нее нет обмена теплом с окружающей средой.

Обозначим температуру стержня в точке с координатой x в момент времени t через $u(x,t)$. Уравнение теплопроводности имеет вид

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial t^2}, \qquad (7.25)$$

где a — коэффициент температуропроводности, зависящий в первую очередь от вещества, из которого сделан стержень.

Уравнение теплопроводности сопровождается начальными и краевыми условиями, делающими постановку задачи физически однозначной. Начальное условие задает распределение температуры в стержне в начальный момент времени (считаем его равным нулю):

$$u(x,0) = f(x). \qquad (7.26)$$

Краевые условия (их должно быть в данном случае два) указывают в простейшем варианте, какая температура поддерживается на концах стержня:

$$u(0,t) = u\big|_{x=0} = \tilde{u}_0(t), \quad u(l,t) = u\big|_{x=l} = \tilde{u}_l(t). \qquad (7.27)$$

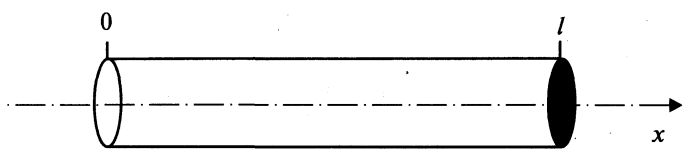


Рис. 7.5. К вопросу о теплопроводности стержня

Моделирование процесса теплопроводности связано с дискретизацией как временного изменения температуры, так и пространственного. Если для пространственных производных использовать простейшие центрально-разностные аппроксимации, а по времени — схему Эйлера, то величины $u_i^k = u(t_k,x_i)$ находятся из системы линейных алгебраических уравнений

$$u_i^{k+1} = u_i^k + \frac{a^2\Delta t}{(\Delta x)^2}(u_{i+1}^k - 2u_i^k + u_{i-1}^k), \tag{7.28}$$

$k = 0, 1, \ldots; i = 1, 2, \ldots, n-1$ — для внутренних узлов пространственной сетки; в силу начального условия $u_i^{(0)} = f(x_i)$. Шаг по времени обозначен Δt, по пространству — Δx.

Описанный метод устойчив при выполнении условия

$$\frac{a^2\Delta t}{(\Delta x)^2} \le \frac{1}{2}. \tag{7.29}$$

Это следует учитывать, выбирая шаги по времени и пространству.

Существенно более устойчива следующая неявная схема второго порядка (схема Кранка—Николсона):

$$u_i^{k+1} = u_i^k + \frac{a^2\Delta t}{2(\Delta x)^2}(u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1}) + \frac{a^2\Delta t}{2(\Delta x)^2}(u_{i+1}^k - 2u_i^k + u_{i-1}^k). \tag{7.30}$$

Это система линейных алгебраических уравнений с трехдиагональной матрицей. Для ее решения наиболее эффективен метод прогонки.

Другие численные схемы решения одномерной задачи теплопроводности можно найти в специальной литературе.

Контрольные вопросы

1. Какие примеры сплошных сред и проистекающих в них процессов Вам известны?
2. Как построить на экране компьютера пространственное распределение электрического поля?
3. Как выглядит уравнение теплопроводности в общем случае? Как к нему ставить начальные и граничные условия?
4. Как построить пятиточечную аппроксимацию первой и второй производных на одномерной сетке?

Темы для рефератов

1. Моделирование процессов тепломассопереноса в приближении сплошной среды.
2. Описание процесса диффузии.
3. Моделирование процесса распространения упругих волн в твердом теле.
4. Моделирование простых течений жидкости.

Тема семинарских занятий

Визуализация физических процессов, проистекающих в сплошной среде.

Лабораторная работа

Общие рекомендации

1. При проведении расчетов необходим контроль точности результатов и устойчивости применяемых численных методов. Для этого достаточно ограничиться эмпирическими приемами (например, сопоставлением решений, полученных с несколькими разными шагами на пространственной и/или временной сетках).

2. Результаты моделирования электрического поля удобно выводить на экран компьютера в следующем виде:

- с помощью изолиний потенциала, построенных для простоты в одной плоскости (в которой лежат заряды или другой); количество изолиний, поддающихся эмпирическому анализу, — от 5 до 8;
- с помощью таблиц координат точек, рассчитанных указанным выше образом на каждой из изолиний;
- используя прием условной раскраски, изображая поле внутри той области, где потенциал особо велик, красным цветом, там, где он мал, синим, а в промежуточных областях — последовательностью цветов спектра.

3. Результаты моделирования процесса теплопроводности в стержне удобно выводить на экран в виде:

- графиков зависимостей температуры от координат точек стержня, располагая на одном графике несколько кривых, относящихся к различным моментам времени — от начала эволюции до завершения наблюдения (моделирования);
- изображения стержня с условной раскраской, отражающей временную эволюцию температуры;
- таблиц зависимостей температуры от времени в нескольких точках стержня;
- графиков зависимостей температуры от времени в нескольких точках стержня.

4. При выводе результатов в табличном виде следует учитывать, что соответствующие табличные шаги не имеют практически ничего общего с шагами по времени и пространству, использованными при моделировании, и определяются удобством и достаточной полнотой для восприятия результатов на экране. Экран, сплошь забитый числами, зрительно плохо воспринимается. Выводимые числа следует разумным образом форматировать, чтобы незначащие цифры практически отсутствовали.

5. При выводе результатов в графической форме графики должны быть построены так, как это принято в математической литературе (с указанием величин, отложенных по осям, их размерности и масштаба и т.д.).

6. Поскольку таблицы, графики и визуальные изображения на одном экране обычно не помещаются, удобно сделать меню, в котором пользователь выбирает желаемый в настоящий момент вид представления результатов.

Примерное время выполнения 16 часов.

Задания к лабораторной работе

1) Выписать математическую модель, определить состав набора входных параметров и их конкретные числовые значения.

2) Спроектировать пользовательский интерфейс программы моделирования, обращая особое внимание на формы представления результатов.

3) Разработать программу моделирования, используя при необходимости и возможности библиотечные программы (например, построения изолиний, метода прогонки и т.д.).

4) Произвести отладку и тестирование полной программы.

5) Выполнить конкретное задание из своего варианта работы.
6) Качественно проанализировать результаты моделирования.
7) Создать текстовый отчет по лабораторной работе, включающий:
   - титульный лист (название работы, исполнителя, группу и т.д.);
   - постановку задачи и описание модели;
   - результаты тестирования программы;
   - результаты, полученные в ходе выполнения задания (в различных формах);
   - качественный анализ результатов.

Варианты заданий

Вариант 1

Построить изолинии поля, созданного четырьмя одноименными и равными по величине зарядами, находящимися в вершинах прямоугольника.

Вариант 2

Построить изолинии поля, созданного четырьмя разноименными и равными по величине зарядами, находящимися в вершинах прямоугольника. Знаки зарядов чередуются циклически по соседним вершинам прямоугольника.

Вариант 3

Построить изолинии поля, созданного четырьмя одноименными зарядами, расположенными в вершинах прямоугольника. Значения зарядов (при последовательном обходе вершин) есть q, $2q$, $3q$, $4q$.

Вариант 4

Построить изолинии поля, созданного четырьмя равноименными и равными по величине зарядами, находящимися в вершинах правильного треугольника и в его центре.

Вариант 5

Построить изолинии поля, созданного шестью равноименными и равными по величине зарядами, находящимися в вершинах правильного шестиугольника.

Вариант 6

Провести моделирование объемной картины электрического поля, созданного тремя равными и одноименными зарядами, находящимися в вершинах равностороннего треугольника.

Вариант 7

Провести моделирование объемной картины электрического поля, созданного четырьмя равными и одноименными зарядами, находящимися в вершинах квадрата.

Вариант 8

Разработать метод построения силовых линий электрического поля, созданного системой зарядов, находящихся в одной плоскости.

Вариант 9

Разработать метод построения изолиний поля, созданного совокупностью однородно заряженных плоских нитей и точечных зарядов. Получить с его помощью изображение поля, созданного нитью, имеющей форму полуокружности (в той же плоскости, в которой находится нить).

Вариант 10

Построить изолинии поля, созданного двумя параллельно расположенными заряженными нитями при условии, что на нитях равные и одноименные заряды.

Вариант 11

Построить изолинии поля, созданного двумя параллельно расположенными заряженными нитями при условии, что на нитях — равные и разноименные заряды.

Вариант 12

Построить изолинии поля, созданного нитью, имеющей форму полуокружности, и зарядом в ее центр. Совокупный заряд на нити и заряд в ее центре равны по величине и имеют разные знаки.

Вариант 13

Изучить динамику изменения температуры в стержне длиной l с теплоизолированными концами, температура на которых поддерживается постоянной и равной T_0, с начальным условием

$$f(x) = -\frac{12T_0}{l^2}x^2 + \frac{12T_0}{l}x + T_0$$

при некотором фиксированном значении коэффициента температуропроводности. Шаг по сетке принять равным $\frac{l}{10}$. Построить графики выхода на стационарное значение температуры в каждом из узлов пространственной сетки.

Вариант 14

В условиях предыдущего варианта исследовать влияние шага пространственной сетки на точность результатов моделирования.

Вариант 15

В условиях задания варианта 13 изучить, как влияет на динамику установления стационарного распределения температуры в стержне коэффициент температуропроводности (путем перебора различных его значений).

Вариант 16

В условиях задания варианта 13 изучить сравнительную эффективность методов, выражаемых формулами (7.28) и (7.30).

Вариант 17

В начальный момент времени стержень длиной 5 м имеет температуру 20°С. На левом конце стержня включается источник тепла, который модулирует температу-

ру по закону $u(0, t) = 20 + 10 \sin(\omega t)$. Произвести моделирование изменения температуры в средней точке стержня при различных соотношениях a и ω вплоть до значения времени $t^* = 5\left(\frac{a}{l\omega}\right)^2$. Есть ли качественные различия в процессе при быстрой ($\omega \gg \left(\frac{a}{l}\right)^2$) и при медленной ($\omega \ll \left(\frac{a}{l}\right)^2$) модуляции?

Вариант 18

Разработать метод максимально наглядной иллюстрации на экране компьютера динамики процесса теплопроводности в стержне, используя сочетание различных приемов, включая цветную раскраску.

Вариант 19

Изучить динамику изменения температуры в стержне длиной l с теплоизолированными концами, который в начальный момент времени имел одинаковую вдоль всего стержня температуру T_0, при условии, что на левом конце температура скачком изменилась и поддерживается равной $4T_0$.

Вариант 20

Изучить динамику изменения температуры в стержне длиной l с теплоизолированными концами, который в начальный момент времени имел одинаковую вдоль всего стержня температуру T_0, при условии, что на обоих концах температура скачком изменилась и поддерживается равной $4T_0$.

Вариант 21

Изучить динамику изменения температуры в стержне длиной l с теплоизолированными концами, который в начальный момент времени имеет одинаковую вдоль всего стержня температуру T_0, если температура на его концах скачком изменилась и поддерживается равной $2T_0$ на левом конце и нулю на правом.

Вариант 22

Изучить динамику изменения температуры в стержне длиной l с теплоизолированными концами, который в начальный момент времени имеет одинаковую температуру T_0, а затем нагревается в центре источником с температурой $4T_0$. Концы стержня при этом сохраняют температуру T_0.

Вариант 23

Изучить динамику изменения температуры в стержне длиной l с теплоизолированными концами, который в начальный момент времени имеет одинаковую температуру T_0, а затем нагревается в центре и на концах источником с температурой $4T_0$.

Вариант 24

Изучить динамику изменения температуры в стержне длиной l с теплоизолированными концами, который в начальный момент времени разбит на 3 равных участка с температурами на концах участков T_0, $2T_0$ и $3T_0$ соответственно, а затем температура на конце стержня с температурой $3T_0$ скачком становится равной T_0.

Дополнительная литература

1. *Араманович И.Г., Левин В.И.* Уравнения математической физики. — М.: Наука, 1969.
2. *Калашников С.Г.* Электричество. — М.: Наука, 1977.
3. *Кикоин А.К., Кикоин И.К.* Молекулярная физика. — М.: Наука, 1976.
4. *Пейре Р., Тейлор Т.Д.* Вычислительные методы в задачах механики жидкостей: Пер. с англ. — Л.: Гидрометеоиздат, 1986.
5. *Поттер Д.* Вычислительные методы в физике: Пер. с англ. — М.: Мир, 1975.
6. *Савельев И.В.* Курс общей физики: В 3 т. Т. 2, 3. — М.: Наука, 1977.
7. *Сивухин Д.В.* Общий курс физики: В 5 т. Т. 2, 3. — М.: Наука, 1974.

§ 3. КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ В ЭКОЛОГИИ

Краткие сведения

В данном практикуме, равно как и в базовом пособии, рассматриваются лишь модели классической экологии (взаимодействие популяций).

Популяция — совокупность особей одного вида, существующих в одно и то же время и занимающих определенную территорию.

Взаимодействие особей внутри популяции определяется внутривидовой конкуренцией, взаимодействие между популяциями — межвидовой конкуренцией.

Внутривидовая конкуренция в популяции с дискретным размножением. Для популяций с дискретным размножением (некоторые виды растений, насекомых и т.д.) поколения четко разнесены во времени и особи разных поколений не сосуществуют. Численность такой популяции можно характеризовать числом N_t и считать t величиной дискретной — номером популяции.

Одна из моделей межвидовой конкуренции в этом случае выражается уравнением

$$N_{t+1} = \frac{N_t R}{1 + (aN_t)^b}. \qquad (7.31)$$

Здесь R — скорость воспроизводства популяции в отсутствие внутривидовой конкуренции (математически это соответствует случаю $a = 0$). Тогда уравнение определяет просто изменение численности популяции по закону геометрической прогрессии: $N_t = N_0 R^t$, где N_0 — начальная численность популяции.

Знаменатель в уравнении отражает наличие конкуренции, делающей скорость роста тем меньше, чем больше численность популяции; a и b — параметры модели.

Исходные параметры модели:

- R — скорость воспроизводства;
- N_0 — начальная численность популяции;
- a — параметр, характеризующий интенсивность внутривидовой конкуренции.

Характерная черта эволюции при $b = 1$ — выход численности популяции на стационарное значение при любых значениях других параметров. Однако в природе так бывает не всегда, и более общая модель при $b \neq 1$ отражает другие, более сложные, но реально существующие виды эволюции. Этих видов модель описывает четыре:

1) монотонное установление стационарной численности популяции;
2) колебательное установление стационарной численности популяции;

3) устойчивые предельные циклы изменения численности популяции;

4) случайные изменения численности популяции без наличия явных закономерностей (динамический хаос).

Внутривидовая конкуренция в популяции с непрерывным размножением. Математическая модель в данном случае строится на основе дифференциальных уравнений. Наиболее известна так называемая логистическая модель

$$\frac{dN}{dt} = rN\left(\frac{K-N}{K}\right). \tag{7.32}$$

Исходные параметры модели:

- r — скорость роста численности популяции в отсутствие конкуренции;
- K — предельное значение численности популяции, при котором скорость роста становится равной нулю;
- N_0 — начальная численность популяции.

Межвидовая конкуренция. В этом случае исследуется конкуренция популяций, потребляющих общий ресурс. Пусть N_1 и N_2 — численности конкурирующих популяций. Модель (называемая также моделью Лотки—Вольтерры) выражается уравнениями

$$\begin{cases} \dfrac{dN_1}{dt} = r_1 N_1 \dfrac{K_1 - N_1 - \alpha_{12} N_2}{K_1}, \\ \dfrac{dN_2}{dt} = r_2 N_2 \dfrac{K_2 - N_2 - \alpha_{21} N_1}{K_2}. \end{cases} \tag{7.33}$$

Содержательный смысл параметров можно понять из сравнения с предыдущей моделью. Дополнительные параметры α_{12} и α_{21} отражают интенсивность межвидовой конкуренции.

Главный вопрос, который интересует исследователя межвидовой конкуренции, — при каких условиях увеличивается или уменьшается численность каждого вида? Данная модель предсказывает следующие режимы эволюции взаимодействующих популяций: устойчивое сосуществование или полное вытеснение одной из них.

Система «хищник—жертва». В этой системе ситуация значительно отличается от предыдущей. В частности, если в случае конкурирующих популяций исчезновение одной означает выигрыш для другой (дополнительные ресурсы), то исчезновение «жертвы» влечет за собой и исчезновение «хищника», для которого в простейшей модели «жертва» является единственным кормом.

Введем обозначения: C — численность популяции хищника, N — численность популяции жертвы. Одна из известных моделей выражается следующими уравнениями:

$$\begin{cases} \dfrac{dN}{dt} = rN - aCN, \\ \dfrac{dC}{dt} = faCN - qC. \end{cases} \tag{7.34}$$

В первое уравнение заложен следующий смысл: в отсутствие хищников (т.е. при $C = 0$) численность жертв растет экспоненциально со скоростью r, так как модель не учитывает внутривидовой конкуренции; скорость роста числа жертв (т.е. $\dfrac{dN}{dt}$) уменьшается тем больше, чем чаще происходят встречи представителей видов; a — коэффициент эффективности поиска.

Второе уравнение говорит о следующем: в отсутствие жертв численность хищников экспоненциально убывает со скоростью q; положительное слагаемое в правой части уравнения компенсирует эту убыль; f — коэффициент эффективности перехода пищи в потомство хищников.

Контрольные вопросы

1. В чем состоит предмет исследований классической экологии?
2. В чем сущность процессов:
   - внутривидовой конкуренции;
   - межвидовой конкуренции;
   - отношений «хищник—жертва»?
3. Каковы цели математического моделирования в экологии?
4. В чем различие приемов моделирования популяций с непрерывным и с дискретным размножением?

Темы для рефератов

1. Задачи классической экологии и математическое моделирование.
2. Математическое моделирование процессов распространения загрязнений окружающей среды.

Тема семинарских занятий

Динамика развития популяций. Математические модели внутривидовой и межвидовой конкуренции и системы «хищник — жертва».

Лабораторная работа

Общие рекомендации

1. При проведении расчетов необходим контроль точности результатов и устойчивости применяемого численного метода. Для этого достаточно ограничиться эмпирическими приемами (например, сопоставлением решений, полученных с несколькими разными шагами по времени).

2. Целесообразно применять для моделирования стандартные методы интегрирования систем дифференциальных уравнений, описанные в математической литературе. Простейшие методы (метод Эйлера) часто бывают неустойчивы и их применение ведет к лишнему расходу времени.

3. Результаты моделирования следует выводить на экран компьютера в следующем виде: в виде таблиц зависимостей численности популяций от времени, в виде графиков этих зависимостей. Уместны звуковые сигналы (одни — в критические моменты для моделируемого процесса, другие — через каждый фиксированный отрезок пройденного пути и т.д.).

4. При выводе результатов в табличном виде следует учитывать, что соответствующий шаг по времени не имеет практически ничего общего с шагом интегрирования и определяется удобством и достаточной полнотой для восприятия результатов на экране. Экран, сплошь забитый числами, зрительно плохо восприни-

мается. Выводимые числа следует разумным образом форматировать, чтобы незначащие цифры практически отсутствовали.

5. При выводе результатов в графической форме графики должны быть построены так, как это принято в математической литературе (с указанием величин, отложенных по осям, их размерности и масштаба и т.д.).

6. Поскольку таблицы и графики на одном экране обычно не помещаются, удобно сделать меню, в котором пользователь выбирает желаемый в настоящий момент вид представления результатов.

Примерное время выполнения — 16 часов.

Задания к лабораторной работе

1) Выписать математическую модель, определить состав набора входных параметров и их конкретные числовые значения.

2) Спроектировать пользовательский интерфейс программы моделирования, обращая особое внимание на формы представления результатов.

3) Выбрать метод интегрирования дифференциальных уравнений модели, найти в библиотеке стандартных программ или разработать самостоятельно программу интегрирования с заданной точностью.

4) Произвести отладку и тестирование полной программы.

5) Выполнить конкретное задание из своего варианта работы.

6) Качественно проанализировать результаты моделирования.

7) Создать текстовый отчет по лабораторной работе, включающий:
- титульный лист (название работы, исполнителя, группу и т.д.);
- постановку задачи и описание модели;
- результаты тестирования программы;
- результаты, полученные в ходе выполнения задания (в различных формах);
- качественный анализ результатов.

Варианты заданий

Вариант 1

Изучить характер эволюции популяции, описываемый моделью (7.31), при значениях параметров $b = 1$, $R = 1$, $N_0 = 100$ в зависимости от значения параметра a в диапазоне $0{,}1 \le a \le 10$.

Есть ли качественные различия в характере эволюции в зависимости от значения a?

Вариант 2

Изучить характер эволюции популяции, описываемый моделью (7.31), при значениях параметров $b = 1$, $R = 4$, $N_0 = 100$ в зависимости от значения параметра a в диапазоне $0{,}1 \le a \le 10$.

Есть ли качественные различия в характере эволюции в зависимости от значения a?

Вариант 3

Изучить характер эволюции популяции, описываемый моделью (7.31), при значениях параметров $b = 4$, $R = 1$, $N_0 = 100$ в зависимости от значения параметра a в диапазоне $0{,}1 \le a \le 10$.

Есть ли качественные различия в характере эволюции в зависимости от значения a?

Вариант 4

Изучить характер эволюции популяции, описываемый моделью (7.31), при значениях параметров $a = 1$, $R = 1$, $N_0 = 100$ в зависимости от значения параметра b в диапазоне $0{,}1 \le b \le 10$.

Есть ли качественные различия в характере эволюции в зависимости от значения b?

Вариант 5

Изучить характер эволюции популяции, описываемый моделью (7.31), при значениях параметров $a = 1$, $R = 4$, $N_0 = 100$ в зависимости от значения параметра b в диапазоне $0{,}1 \le b \le 10$.

Есть ли качественные различия в характере эволюции в зависимости от значения b?

Вариант 6

Изучить характер эволюции популяции, описываемый моделью (7.31), при значениях параметров $a = 3$, $R = 1$, $N_0 = 100$ в зависимости от значения параметра b в диапазоне $0{,}1 \le b \le 10$.

Есть ли качественные различия в характере эволюции в зависимости от значения b?

Вариант 7

Изучить характер эволюции популяции, описываемый моделью (7.31), при значениях параметров $a = 3$, $b = 1$, $N_0 = 100$ в зависимости от значения параметра R в диапазоне $1 \le R \le 4$.

Есть ли качественные различия в характере эволюции в зависимости от значения R?

Вариант 8

Изучить характер эволюции популяции, описываемый моделью (7.31), при значениях параметров $a = 3$, $b = 4$, $N_0 = 100$ в зависимости от значения параметра R в диапазоне $1 \le R \le 4$.

Есть ли качественные различия в характере эволюции в зависимости от значения R?

Вариант 9

Реализовать модель (7.31) при следующих наборах значений параметров:

1) $N_0 = 100$, $a = 1$, $R = 2$, $b = 1$;
2) $N_0 = 100$, $a = 1$, $R = 2$, $b = 4$;
3) $N_0 = 100$, $a = 1$, $R = 4$, $b = 3{,}5$;
4) $N_0 = 100$, $a = 1$, $R = 4$, $b = 4{,}5$

и изучить вид соответствующих режимов эволюции.

Вариант 10

Для модели (7.31) в фазовой плоскости (b, R) найти границы зон, разделяющих режимы монотонного и колебательного установления стационарной численности популяции изучаемой системы.

Вариант 11

Для модели (7.31) в фазовой плоскости (b, R) найти границы зон, разделяющих режим колебательного установления стационарной численности популяции изучаемой системы и режим устойчивых предельных циклов.

Вариант 12

Реализовать моделирование межвидовой конкуренции по формулам (7.33) при значениях параметров $r_1 = 2$, $r_2 = 2$, $K_1 = 200$, $K_2 = 200$, $\alpha_{12} = 0{,}5$, $\alpha_{21} = 0{,}5$. Проанализировать зависимость судьбы популяций от соотношения значений их начальной численности N_1^0, N_2^0.

Вариант 13

Реализовать моделирование межвидовой конкуренции по формулам (7.33) при значениях параметров $r_1 = 2$, $r_2 = 2$, $K_1 = 200$, $K_2 = 200$, $N_1^0 = 100$, $N_2^0 = 100$. Проанализировать зависимость судьбы популяций от соотношения значений коэффициентов конкуренции α_{12} и α_{21}.

Вариант 14

Построить в фазовой плоскости (N_1^0, N_2^0) границы зон, разделяющих какие-либо два режима эволюции конкурирующих популяций (в соответствии с моделью (7.33)). Остальные параметры модели выбрать произвольно. Учесть при этом, что режим устойчивого сосуществования популяций может в принципе реализоваться только при $\alpha_{12}\alpha_{21} < 1$.

Вариант 15

Провести моделирование динамики численности популяций в системе «хищник—жертва» (модель (7.34)) при значениях параметров $r = 5$, $a = 0{,}1$, $q = 2$, $f = 0{,}6$. Проанализировать зависимость исхода эволюции от соотношения значений параметров N_0 и C_0.

Вариант 16

Провести моделирование динамики численности популяций в системе «хищник—жертва» (модель (7.34)) при значениях параметров $r = 5$, $a = 0{,}1$, $q = 2$, $N_0 = 100$, $C_0 = 6$. Проанализировать зависимость результатов моделирования от значения параметра f в диапазоне $0{,}1 \le f \le 2$.

Вариант 17

Провести моделирование динамики численности популяций в системе «хищник—жертва» (модель (7.34)) при значениях параметров $r = 5$, $a = 0{,}1$, $f = 2$, $N_0 = 100$, $C_0 = 6$. Проанализировать зависимость результатов моделирования от значения параметра q в диапазоне $0{,}1 \le q \le 2$.

Вариант 18

Провести моделирование динамики численности популяций в системе «хищник—жертва» (модель (7.34)) при значениях параметров $a = 0{,}1$, $f = 2$, $q = 2$, $N_0 = 100$, $C_0 = 6$. Проанализировать зависимость результатов моделирования от значения параметра r в диапазоне $0{,}1 \le r \le 2$.

Вариант 19

Провести моделирование динамики численности популяций в системе «хищник—жертва» (модель (7.34)) при значениях параметров $r = 5$, $q = 2$, $f = 2$, $N_0 = 100$, $C_0 = 6$. Проанализировать зависимость результатов моделирования от значения параметра a в диапазоне $0{,}1 \le a \le 2$.

Вариант 20

Модель (7.34) предсказывает сопряженные колебания численности жертв и хищников. Исследовать зависимость запаздывания амплитуд колебаний численности хищников от амплитуд колебаний численности жертв в зависимости от значений параметра a. Значения остальных параметров фиксировать по усмотрению.

Вариант 21

Модель (7.34) предсказывает сопряженные колебания численности жертв и хищников. Исследовать зависимость запаздывания амплитуд колебаний численности хищников от амплитуд колебаний численности жертв в зависимости от значений параметра q. Значения остальных параметров фиксировать по усмотрению.

Вариант 22

Модель (7.34) предсказывает сопряженные колебания численности жертв и хищников. Исследовать зависимость запаздывания амплитуд колебаний численности хищников от амплитуд колебаний численности жертв в зависимости от значений параметра f. Значения остальных параметров фиксировать по усмотрению.

Вариант 23

Модель (7.34) предсказывает сопряженные колебания численности жертв и хищников. Исследовать зависимость запаздывания амплитуд колебаний численности хищников от амплитуд колебаний численности жертв в зависимости от значений параметра r. Значения остальных параметров фиксировать по усмотрению.

Вариант 24

Модель (7.34) предсказывает сопряженные колебания численности жертв и хищников. Исследовать зависимость запаздывания амплитуд колебаний численности хищников от амплитуд колебаний численности жертв в зависимости от соотношения значений начальных численностей популяций N_0 и C_0. Значения остальных параметров фиксировать по усмотрению.

Дополнительная литература

1. *Бейли Н.* Математика в биологии и медицине: Пер. с англ. — М.: Мир, 1970.
2. *Бейли Н.* Статистические методы в биологии: Пер. с англ. — М.: ИЛ, 1962.
3. *Бигон М., Харпер Дж., Таунсенд К.* Экология. Особи, популяции и сообщества: Пер. с англ.: В двух книгах. — М.: Мир, 1989.
4. *Горстко А.Б., Угольницкий Г.А.* Введение в моделирование эколого-экономических систем. — Ростов: РГУ, 1990.
5. *Рифлекс Р.* Основы общей экологии. Пер. с англ. — М.: Мир, 1979.

§ 4. МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ ПРОЦЕССОВ

Краткие сведения

К числу случайных процессов, изучаемых методом имитационного моделирования (методом Монте-Карло) относятся, в частности, процессы, связанные с формированием и обслуживанием очередей (так называемые процессы **массового обслуживания**). Простейшая задача данного класса такова. Имеется система массового обслуживания с одним узлом обслуживания (магазин с одним продавцом, ремонтная зона в автохозяйстве, травмопункт с одним врачом, телефонная станция с одним входом, сервер с одним входным каналом и т.д.). К услугам системы клиенты прибегают случайным образом (с заданной функцией распределения отрезков времени между приходами). Если система свободна, то начинает обслуживать клиента сразу, иначе ставит его в очередь. Длительность обслуживания каждого клиента — случайная величина с известным законом распределения.

В ходе решения данной задачи требуется дать ответ на вопросы типа «какова функция распределения вероятностей времени ожидания клиента в очереди?» «каково время простоя системы в ожидании клиентов?», «если сами эти функции определять сложно, то каковы их наиболее важные характеристики (т.е. математическое ожидание, дисперсия и т.д.)?».

Основа этой задачи — случайный процесс прихода клиентов в систему обслуживания. Промежутки между приходами любой последовательной пары клиентов — независимые случайные события, распределенные по некоторому закону. Реальный характер этого закона может быть установлен лишь путем многочисленных наблюдений; в качестве простейшей модельной функции плотности вероятности можно взять равновероятное распределение в диапазоне времени от 0 до некоторого T — максимально возможного промежутка между приходами двух последовательных покупателей. При этом распределении вероятность того, что между приходами двух покупателей пройдет 1 минута, 3 минуты или 8 минут, одинакова (если $T > 8$ мин).

Такое распределение, конечно, малореалистично; реально для большинства процессов массового обслуживания функция распределения растет от $t = 0$, имеет при некотором значении $t = \tau$ максимум и быстро спадает при больших t, т.е. имеет вид, изображенный на рис. 7.6.

Можно, конечно, подобрать немало элементарных функций, имеющих качественно такой вид. В теории массового обслуживания широко используется семейство функций Пуассона

$$p_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}, \qquad (7.35)$$

где λ — некоторая постоянная, n — произвольное целое.

Функции (7.35) имеют максимум при $\tau = n/\lambda$ и нормированы.

Второй случайный процесс в этой задаче, никак не связанный с первым, определяется последовательностью случайных событий — длительностями обслуживания каждого из покупателей. Распределение вероятностей длительности обслуживания имеет тот же качественный вид, что и в предыдущем случае.

Для примера в таблице в колонке A записаны случайные числа — промежутки между приходами клиентов (в минутах), в колонке B — случайные числа — длительности обслуживания (в минутах). Для определенности взято $a_{max} = 10$ и $b_{max} = 5$.

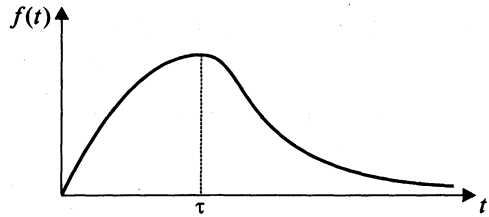


Рис. 7.6. Схематическое изображение плотности вероятности распределения времени между появлениями клиентов в системе массового обслуживания

Из этой короткой таблицы, разумеется, невозможно установить, какие законы распределения приняты для величин A и B. Остальные колонки предусмотрены для удобства анализа; входящие в них числа находятся путем элементарного расчета. В колонке C представлено условное время прихода клиента; D — момент начала обслуживания; E — момент конца обслуживания; F — длительность времени, проведенного клиентом в системе в целом; G — время, проведенное в очереди в ожидании обслуживания; H — время, проведенное системой в ожидании клиентов (если их нет). Таблицу удобно заполнять по горизонтали, переходя от строчки к строчке. Так как начало обслуживания очередного клиента определяется либо временем его прихода, если система не занята, либо временем ухода предыдущего клиента, приведем для удобства соответствующие формулы (в них $i = 1, 2, 3, ...$):

$$c_1 = 0,\ c_{i+1} = c_i + a_{i+1};\ d_1 = 0,\ d_{i+1} = \max(c_{i+1},\ e_i); \tag{7.36а}$$

$$e_1 = b_1,\ e_i = d_i + b_i;\ f_i = e_i - c_i;\ g_1 = 0,\ g_{i+1} = f_{i+1} - b_{i+1};\ h_1 = 0,\ h_{i+1} = d_{i+1} - e_i. \tag{7.36б}$$

Таким образом, при данных случайных наборах чисел в колонках A и B и клиентам приходилось стоять в очереди (колонка G), и система простаивала в ожидании клиента (колонка H).

№ п/п	A	B	C	D	E	F	G	H
1	0	3	0	0	3	3	0	0
2	1	1	1	3	4	3	2	0
3	8	2	9	9	11	2	0	5
4	3	3	12	12	15	3	0	1
5	5	4	17	17	21	4	0	2

При моделировании систем такого вида прежде всего возникает вопрос, какое среднее время приходится стоять в очереди? Ответить на него, кажется, несложно — надо найти

$$\bar{g} = \frac{1}{n}(g_1 + g_2 + \ldots + g_n) \tag{7.37}$$

в некоторой серии испытаний. Аналогично можно найти среднее значение величины h. Труднее ответить на вопрос о достоверности полученных результатов; для этого надо провести несколько серий испытаний и использовать стандартные методы математической статистики (часто уместна обработка с помощью распределения Стьюдента).

Более сложный вопрос — каково распределение случайных величин G и H при заданных распределениях случайных величин A и B? Качественный ответ на него можно попытаться получить, построив соответствующие гистограммы по результатам моделирования. Затем делается некоторая гипотеза о виде распределения и используются один или несколько статистических критериев проверки достоверности этой гипотезы.

Располагая функцией распределения (пусть даже эмпирической, но достаточно надежной), можно ответить на любой вопрос о характере процесса ожидания в очереди. Например: какова вероятность прождать дольше m минут? Ответ будет получен, если найти отношение площади криволинейной трапеции, ограниченной графиком плотности распределения, прямыми $x = m$ и $y = 0$, к площади всей фигуры.

Контрольные вопросы

1. Что такое «случайный процесс»?
2. Каковы принципы компьютерного генерирования равномерно распределенных случайных чисел?
3. Как можно получить последовательность случайных чисел с пуассоновским законом распределения?
4. Что такое «система массового обслуживания»? Приведите примеры.
5. В чем заключается метод Монте-Карло вычисления площадей плоских фигур? объемов тел?
6. Какие примеры случайных процессов Вы можете привести?

Темы для рефератов

1. Принципы компьютерной генерации последовательностей случайных чисел и статистические критерии определения свойств последовательностей.
2. Методы статистической обработки результатов, полученных при компьютерном моделировании случайных процессов.

Тема семинарских занятий

Получение последовательностей случайных чисел с заданным законом распределения.

Лабораторная работа

Общие рекомендации

1. При выполнении данной работы необходима генерация длинных последовательностей псевдослучайных чисел с заданным законом распределения вероятностей. Ее можно основывать на стандартном датчике равномерно распределенных случайных чисел, встроенном в применяемую систему программирования, с использованием одной из процедур пересчета данной последовательности в последовательность с нужным законом распределения (например, процедуру «отбор — отказ»).

2. Одна из центральных задач при моделировании случайных процессов — нахождение характеристик случайных величин, являющихся объектом моделирования. Главная такая характеристика — функция распределения. Ее вид можно качественно оценить по гистограмме, построенной в ходе моделирования, а гипотезу о функциональной форме проверить с помощью одного из стандартных критериев, используемых в математической статистике (например, критерия χ^2). Однако это не всегда целесообразно, особенно если в задаче требуется определить лишь некоторые характеристики случайной величины — чаще всего среднее значение и дисперсию. Их можно найти без моделирования самой функции распределения. При этом статистическая оценка достоверности результатов является обязательной.

3. Результаты моделирования уместно выводить на экран компьютера в следующем виде: в виде таблиц значений рассчитываемой величины (как правило, в нескольких выборках), в виде гистограмм распределения случайных величин, построенных в ходе моделирования.

4. Целесообразно там, где это возможно, сопровождать имитационное моделирование визуальным отображением соответствующего процесса на экране компьютера (процесс формирования очереди, рождение и исчезновение объектов в задачах моделирования популяций и т.д.).

Примерное время выполнения 16 часов.

Задание к лабораторной работе

Произвести имитационное моделирование указанного случайного процесса и оценить достоверность полученных результатов, пользуясь статистическими критериями.

Варианты заданий

Вариант 1

Провести моделирование очереди в магазине с одним продавцом при равновероятных законах распределения описанных выше случайных величин: прихода покупателей и длительности обслуживания (при некотором фиксированном наборе параметров). Получить устойчивые характеристики: средние значения ожидания в очереди покупателем и простой продавца в ожидании прихода покупателей. Оценить их достоверность. Оценить характер функции распределения величин g и h.

Вариант 2

Провести то же моделирование при пуассоновских законах распределения вероятностей входных событий: прихода покупателей и длительности обслуживания (при некотором фиксированном наборе параметров).

Вариант 3

Провести то же моделирование при нормальном законе распределения вероятностей входных событий: прихода покупателей и длительности обслуживания (при некотором фиксированном наборе параметров).

Вариант 4

В рассмотренной выше системе может возникнуть критическая ситуация, когда очередь неограниченно растет со временем. В самом деле, если покупатели заходят в магазин очень часто (или продавец работает слишком медленно), очередь начи-

нает расти, и в рассматриваемой системе с конечным временем обслуживания наступит кризис.

Построить зависимость между величинами (a_{max}, b_{max}), отражающую границу указанной критической ситуации, при равновероятном распределении входных событий.

Вариант 5

На междугородней телефонной станции две телефонистки обслуживают общую очередь заказов. Очередной заказ обслуживает та телефонистка, которая первой освободилась. Если обе в момент поступления заказа заняты, то звонок аннулируется и требуется звонить снова. Смоделировать процесс, считая входные потоки пуассоновскими.

Вариант 6

Смоделировать ситуацию, описанную в предыдущем варианте, но считать, что, если в момент попытки сделать заказ обе телефонистки заняты, формируется очередь.

Вариант 7

Пусть на телефонной станции с одним входом используется обычная система: если абонент занят, то очередь не формируется и надо звонить снова. Смоделировать ситуацию: три абонента пытаются дозвониться до одного и того же владельца номера и в случае успеха разговаривают с ним некоторое (случайное по длительности) время. Какова вероятность того, что некто, пытающийся дозвониться, не сможет сделать это за определенное время T?

Вариант 8

Смоделировать ситуацию, описанную в предыдущем варианте, но считать, что, если в момент попытки связаться телефон абонента занят, формируется очередь.

Вариант 9

На травмопункте работает один врач. Длительность лечения больного и промежутки времени между поступлениями больных — случайные величины, распределенные по пуассоновскому закону. По тяжести травм больные делятся на три категории, поступление больного любой категории — случайное событие с равновероятным распределением. Врач вначале занимается больными с максимально тяжелыми травмами (в порядке их поступления), затем, если таковых нет, — больными с травмами средней тяжести (в порядке их поступления) и лишь затем — больными с легкими травмами. Смоделировать процесс и оценить средние времена ожидания в очереди больных каждой из категорий.

Вариант 10

Смоделировать ситуацию, описанную в предыдущем варианте, при условии, что в травмопункте работают два врача, а больные делятся не на три, а на две категории.

Вариант 11

Одна ткачиха обслуживает группу станков, осуществляя по мере необходимости краткосрочное вмешательство, длительность которого — случайная величина.

Какова вероятность простоя сразу двух станков? Как велико среднее время простоя одного станка?

Вариант 12

Смоделировать ситуацию, описанную в предыдущем варианте, если группу станков совместно обслуживают две ткачихи.

Вариант 13

В городском автохозяйстве две ремонтные зоны. Одна — обслуживает ремонты краткой и средней продолжительности, другая — средней и долгой (т.е. среднесрочный ремонт может осуществлять каждая из зон). По мере поломок в автохозяйство доставляют транспорт; промежуток времени между доставками — случайная пуассоновская величина. Продолжительность ремонта — случайная величина с нормальным законом распределения. Смоделировать описанную систему. Каковы средние времена ожидания в очереди транспорта, требующего соответственно краткосрочного, среднесрочного и длительного ремонта?

Вариант 14

Реализовать имитационную модель статистического моделирования для решения задачи Бюффона (XVIII в.). Автор аналитически нашел, что если на поле, разграфленное параллельными прямыми, расстояние между которыми L, бросается наугад игла длиной l, то вероятность того, что игла пересечет хотя бы одну прямую, определяется формулой $p = \frac{2l}{\pi L}$.

Эта задача дала способ имитационному определению числа π. Действительно, если $L = 2l$, то $p = \frac{1}{\pi}$. В ходе моделирования выполнить этот расчет.

Вариант 15

Разработать модель случайного одномерного блуждания (модель «пьяницы»). Блуждание задается по правилу: если случайное число из отрезка [0,1] меньше 0,5, то делается шаг вправо на расстояние h, в противном случае — влево. Распределение случайных чисел принять равновероятным.

Решить задачу: какова вероятность при таком блуждании удалиться от начальной точки на n шагов?

Вариант 16

В условиях задачи из предыдущего варианта получить ответ на вопрос: какова вероятность «пьянице» вернуться через n шагов в начальную точку?

Вариант 17

Точка хаотически блуждает на плоскости по узлам квадратной сетки с возможностью делать с равной вероятностью шаги влево-вправо-вверх-вниз на фиксированный (за один ход) шаг. Движение происходит в замкнутом прямоугольном объеме, и при соприкосновении со стенкой происходит зеркальное отражение от нее.

Ответить в ходе моделирования на вопрос: как связана частота посещения каждого узла с расстоянием от него до того узла, из которого начинается движение?

Вариант 18

Смоделировать ту же ситуацию, что и в задании к варианту 17, при условии неограниченной области блуждания и ответить на заданный вопрос.

Вариант 19

Смоделировать полет пчелы. На плоскости (поляне) случайным образом растут медоносные растения с заданной концентрацией (на 1 $м^2$). В центре — улей, из которого вылетает пчела. Пчела может долететь от одного растения до любого другого растения, но вероятность выбора монотонно уменьшается с увеличением расстояния между растениями (по некоторому закону). Какова вероятность посещения пчелой конкретного заданного растения за заданное количество элементарных перелетов?

Вариант 20

Реализовать модель плоского броуновского движения n частиц в прямоугольнике. Частицы считать шариками конечного размера. Удары частиц друг о друга и о стенки моделировать как абсолютно упругие. Определить в этой модели зависимость давления газа на стенки от числа частиц.

Вариант 21

Разработать в деталях и реализовать модель перемешивания (диффузии) газов в замкнутом сосуде. В начальный момент времени каждый газ занимает половину сосуда. Изучить с помощью этой модели зависимость скорости диффузии от различных входных параметров.

Вариант 22

Реализовать имитационную модель системы «хищник—жертва» по следующей схеме.

«Остров» размером 20×20 заселен дикими кроликами, волками и волчицами. Имеется по несколько представителей каждого вида. Кролики в каждый момент времени с одинаковой вероятностью 1/9 передвигаются в один из восьми соседних квадратов (за исключением участков, ограниченных береговой линией) или просто сидят неподвижно. Каждый кролик с вероятностью 0,2 превращается в двух кроликов. Каждая волчица передвигается случайным образом, пока в одном из соседних восьми квадратов не окажется кролик, за которым она охотится. Если волчица и кролик оказываются в одном квадрате, волчица съедает кролика и получает одно очко. В противном случае она теряет 0,1 очка.

Волки и волчицы с нулевым количеством очков умирают. В начальный момент времени все волки и волчицы имеют 1 очко. Волк ведет себя подобно волчице до тех пор, пока в соседних квадратах не исчезнут все кролики; тогда, если волчица находится в одном из восьми близлежащих квадратов, волк гонится за ней.

Если волк и волчица окажутся в одном квадрате и там нет кролика, которого можно съесть, они производят потомство случайного пола.

Пронаблюдать за изменением популяции в течение некоторого периода времени. Проследить, как сказываются на эволюции популяций изменения параметров модели.

Вариант 23

Промоделировать процесс распространения инфекции стригущего лишая по участку кожи размером $n \times n$ (n — нечетное) клеток.

Предполагается, что исходной зараженной клеткой кожи является центральная. В каждый интервал времени пораженная инфекцией клетка может с вероятностью 0,5 заразить любую из соседних здоровых клеток. По прошествии шести единиц времени зараженная клетка становится невосприимчивой к инфекции, возникший иммунитет действует в течение последующих четырех единиц времени, а затем клетка оказывается здоровой. В ходе моделирования описанного процесса выдавать текущее состояние моделируемого участка кожи в каждом интервале времени, отмечая зараженные, невосприимчивые к инфекции и здоровые клетки.

Проследить, как сказываются на результатах моделирования изменение размеров поля и вероятность заражения.

Вариант 24

Разработать в деталях и реализовать модель распространения загрязняющих окружающую среду частиц вещества, испускаемых в атмосферу заводской трубой (например, золы, получающейся после сжигания угля на электростанции). Считать движение частицы состоящим из двух компонент: в горизонтальной плоскости — под влиянием случайных порывов ветра, в вертикальном — под действием силы тяжести.

Дополнительная литература

1. *Бейли Н.* Статистические методы в биологии: Пер. с англ. — М.: ИЛ, 1962.
2. *Гнеденко Б.В., Коваленко И.Н.* Введение в теорию массового обслуживания. — М.: Наука, 1966.
3. *Саати Т.* Элементы теории массового обслуживания и ее приложения: Пер. с англ. — М.: Сов. радио, 1991.
4. *Шеннон Р.* Имитационное моделирование систем — искусство и наука: Пер. с англ. — М.: Мир, 1978.

Тесты к главе 7

Модели и технология компьютерного моделирования

1. Какое высказывание наиболее точно определяет понятие «модель»:
   1) точная копия оригинала;
   2) оригинал в миниатюре;
   3) образ оригинала с наиболее присущими ему свойствами;
   4) начальный замысел будущего объекта?
2. Компьютерное моделирование — это:
   1) процесс построения модели компьютерными средствами;
   2) процесс исследования объекта с помощью его компьютерной модели;
   3) построение модели на экране компьютера;
   4) решение конкретной задачи с помощью компьютера.
3. Вербальной моделью является:
   1) модель автомобиля; 2) сборник правил дорожного движения;
   3) формула закона всемирного тяготения;
   4) номенклатура списка товаров на складе.

4. Математической моделью является:
   1) модель автомобиля;
   2) сборник правил дорожного движения;
   3) формула закона всемирного тяготения;
   4) номенклатура списка товаров на складе.
5. Информационной моделью является:
   1) модель автомобиля;
   2) сборник правил дорожного движения;
   3) формула закона всемирного тяготения;
   4) номенклатура списка товаров на складе.
6. К детерминированным моделям относится:
   1) модель случайного блуждания частицы;
   2) модель формирования очереди;
   3) модель свободного падения тела в среде с сопротивлением;
   4) модель игры «орел — решка».
7. К стохастическим моделям относится:
   1) модель движения тела, брошенного под углом к горизонту;
   2) модель броуновского движения;
   3) модель таяния кусочка льда в стакане;
   4) модель обтекания газом крыла самолета.
8. Последовательность этапов моделирования:
   1) цель, объект, модель, метод, алгоритм, программа, эксперимент, анализ, уточнение;
   2) цель, модель, объект, алгоритм, программа, эксперимент, уточнение выбора объекта;
   3) объект, цель, модель, эксперимент, программа, анализ, тестирование;
   4) объект, модель, цель, алгоритм, метод, программа, эксперимент.
9. Индуктивное моделирование предполагает:
   1) гипотетическое описание модели;
   2) решение задачи методом индукции;
   3) решение задачи дедуктивным методом;
   4) построение модели как частного случая глобальных законов природы.
10. Дедуктивное моделирование предполагает:
   1) гипотетическое описание модели;
   2) решение задачи методом индукции;
   3) решение задачи дедуктивным методом;
   4) построение модели как частного случая глобальных законов природы.
11. Компьютерный эксперимент — это:
   1) решение задачи на компьютере;
   2) исследование модели с помощью компьютерной программы;
   3) подключение компьютера для обработки физических экспериментов;
   4) автоматизированное управление физическим экспериментом.

Моделирование физических процессов

1. Модель свободного падения тела в среде с трением:
   1) $ma = mg - kV$, m — масса, a — ускорение, V — скорость, k — коэффициент;
   2) $ma = mg - kX$, m — масса, a — ускорение, X — перемещение, k — коэффициент;

3) $ma = mg - kP$, m — масса, a — ускорение, P — давление, k — коэффициент;
4) $ma = mg - kR$, m — масса, a — ускорение, R — плотность, k — коэффициент.

2. Модель движения тела, брошенного под углом к горизонту в системе координат, в которой ось x направлена по горизонту, y — вертикально вверх:
1) $ma_x = -kV_x$, $ma_y = mg - kV_y$, $V_{0x} = V_0\cos A$, $V_{0y} = V_0\sin A$, где a_x, a_y, V_x, V_y — проекции ускорения и скорости, m — масса, A — угол бросания;
2) $ma_x = mg - kV_x$, $ma_y = mg - kV_y$, $V_{0x} = V_0\cos A$, $V_{0y} = V_0\sin A$, где a_x, a_y, V_x, V_y — проекции ускорения и скорости, m — масса, A — угол бросания;
3) $ma_x = mg - kV_x$, $ma_y = -kV_y$, $V_{0x} = V_0\cos A$, $V_{0y} = V_0\sin A$, где a_x, a_y, V_x, V_y — проекции ускорения и скорости, m — масса, A — угол бросания;
4) $ma_x = mg - kV_x$, $ma_y = mg - kV_y$, $V_{0x} = V_0\sin A$, $V_{0y} = V_0\cos A$, где a_x, a_y, V_x, V_y — проекции ускорения и скорости, m — масса, A — угол бросания.

3. Модель движения небесного тела относительно Земли (плоский случай):
1) $d^2x/dt^2 = -GMx/\sqrt{(x^2 + y^2)^3}$; $d^2y/dt^2 = -GMy/\sqrt{(x^2 + y^2)^3}$; где G — гравитационная постоянная, M— масса Земли, x, y — координаты тела;
2) $dx/dt = -GMm/\sqrt{(x^2 + y^2)^3}$; $dy/dt = -GMm/\sqrt{(x^2 + y^2)^3}$; где G — гравитационная постоянная, M— масса Земли, x, y — координаты тела, m — масса тела;
3) $d^2V_x/dt^2 = -GMV_x/\sqrt{(x^2 + y^2)^3}$; $d^2V_y/dt^2 = -GMV_y/\sqrt{(x^2 + y^2)^3}$; где G — гравитационная постоянная, M— масса Земли, V_x, V_y — скорость тела;
4) $d^2x/dt^2 = -GM/mx^2$; $d^2y/dt^2 = -GM/my^2$, где G — гравитационная постоянная, M— масса Земли, x, y — координаты тела, m — масса тела.

4. Для краевой задачи теплопроводности в одномерном стержне, концы которого имеют координаты $x = 0$ и $x = L$, в случае, когда на границах задана температура, уравнение теплопроводности дополняют граничными условиями вида ($u(x, t)$ — температура в стержне):
1) $u(0, t) = 0$; $u(L,t) = 0$;
2) $u(0,t) = T_0$; $u(L,t) = T_L$;
3) $\partial u/\partial x|_{x=0} = T_0$; $\partial u/\partial x|_{x=L} = T_L$;
4) $\partial u/\partial x|_{x=0} = 0$; $\partial u/\partial x|_{x=L} = 0$.

5. Для краевой задачи теплопроводности в одномерном стержне, концы которого имеют координаты $x = 0$ и $x = L$, в случае, когда границы теплоизолированы, уравнение теплопроводности дополняют граничными условиями вида ($u(x, t)$ — температура в стержне):
1) $u(0, t) = 0$; $u(L,t) = 0$;
2) $u(0,t) = T_0$; $u(L,t) = T_L$;
3) $\partial u/\partial x|_{x=0} = T_0$; $\partial u/\partial x|_{x=L} = T_L$;
4) $\partial u/\partial x|_{x=0} = 0$; $\partial u/\partial x|_{x=L} = 0$.

6. Для краевой задачи теплопроводности в одномерном стержне, концы которого имеют координаты $x = 0$ и $x = L$, в случае, когда на границах задан тепловой поток, уравнение теплопроводности дополняют граничными условиями вида ($u(x, t)$ — температура в стержне):
1) $u(0, t) = 0$; $u(L,t) = 0$;
2) $u(0,t) = T_0$; $u(L,t) = T_L$;
3) $\partial u/\partial x|_{x=0} = Q_0$; $\partial u/\partial x|_{x=L} = Q_L$;
4) $\partial u/\partial x|_{x=0} = 0$; $\partial u/\partial x|_{x=L} = 0$.

Компьютерное моделирование в экологии

1. Дискретная модель численности популяции, зависящей в основном от чистой скорости воспроизводства (без учета внутривидовой конкуренции, R — скорость воспроизводства):

1) $N_{t+1} = RN_t$;
2) $N_t = RN_{t+1}$;
3) $N_{t+1} = RN_t + RN_{t+1}$;
4) $N_t = RN_t/(1+N_t)$.

2. Дискретная модель роста популяций, ограниченная внутривидовой конкуренцией (R — скорость воспроизводства, a, b — коэффициенты):

1) $N_{t+1} = RN_t$;
2) $N_t = RN_{t+1}$;
3) $N_{t+1} = RN_t + RN_{t+1}$;
4) $N_{t+1} = RN_t/(1 + (aN_t)^b)$.

3. Непрерывная модель численности популяций, без учета внутривидовой конкуренции (r — скорость роста численности, K — предельная плотность насыщения):

1) $dN/dt = rN/(1 + N)$;
2) $dN/dt = rN$;
3) $dN/dt = r(K - N)$;
4) $dN/dt = r$.

4. Непрерывная (логистическая) модель численности популяций с учетом внутривидовой конкуренции (r — скорость роста численности, K — предельная плотность насыщения):

1) $dN/dt = rN/(1+N)$; 2) $dN/dt = rN(K - N)/K$;
3) $dN/dt = r(K-N)$; 4) $dN/dt = r$.

5. Модель межвидовой конкуренции для случая двух популяций с численностью N_1 и N_2 (r_1, r_2 — врожденные скорости роста популяций; K_1, K_2 — предельные плотности насыщения; a_{12}, a_{21} — коэффициенты конкуренции):

1) $dN_1/dt = r_1N_1$; $dN_2/dt = r_2N_2$;
2) $dN_1/dt = r_1N_1(K_1 - a_{12}N_1)/K_1$; $dN_2/dt = r_2N_2(K_2 - a_{21}N_2)/K_2$;
3) $dN_1/dt = r_1N_1(K_1 - N_1 - a_{12}N_2)/K_1$; $dN_2/dt = r_2N_2(K_2 - N_2 - a_{21}N_1)/K_2$;
4) $dN_1/dt = r_1N_1(K_1 - N_2)/K_1$; $dN_2/dt = r_2N_2(K_2 - N_1)/K_2$.

6. Модель межвидовой конкуренции «хищник—жертва» (N_1, r, a — численность, скорость роста и коэффициент смертности популяции жертвы; N_2, b, q — численность, эффективность добычи и коэффициент смертности популяции хищника):

1) $dN_1/dt = rN_1 - aN_1N_2$, $dN_2/dt = bN_1 - qN_2$;
2) $dN_1/dt = rN_1 - aN_1N_2$, $dN_2/dt = abN_1N_2 - qN_2$;
3) $dN_1/dt = rN_1(N_1 - N_2 - aN_2)$, $dN_2/dt = aN_2(N_2 - N_1 - qN_1)$;
4) $dN_1/dt = rN_1 - aN_2$, $dN_2/dt = bN_1 - qN_2$.

7. В имитационной модели «Жизнь» (Д. Конвей) количество стационарных конфигураций:

1) 2; 2) 3; 3) 4; 4) более 10.

Моделирование случайных процессов

1. Компьютерная модель «очередь» не может быть применена для оптимизации в следующих задачах:

1) обслуживание в магазине;
2) телефонная станция;

3) компьютерная сеть с выделенным сервером;
4) спортивные соревнования.

2. В модели «очередь» случайный процесс формирования очереди является:
1) марковским;
2) немарковским;
3) линейным;
4) квазистационарным.

3. Для моделирования очереди менее всего подходит распределение длительности ожидания:
1) равновероятностное;
2) пуассоновское;
3) нормальное;
4) экспоненциальное.

4. Пусть автобусы двигаются с интервалом в 10 минут. Каково среднее время ожидания транспорта на остановке при наличии одного маршрута:
1) 10 мин; 2) 0 мин;
3) 5 мин; 4) не определено?

5. Пусть автобусы двигаются с интервалом в 10 минут. Каково среднее время ожидания транспорта на остановке при наличии двух маршрутов:
1) 5 мин; 2) менее 5 мин;
3) более 5 мин; 4) 10 мин?

6. Методом случайных испытаний (метод Монте-Карло) невозможно вычислить:
1) число π; 2) площадь;
3) числа Фибоначчи; 4) корень уравнения.

7. С помощью имитационной модели случайного блуждания точек невозможно изучать:
1) законы идеального газа;
2) броуновское движение;
3) законы кинематики;
4) тепловые процессы.

Правильные ответы

Модели и технология компьютерного моделирования

№	1	2	3	4
1			X	
2		X		
3		X		
4			X	
5				X
6			X	
7		X		
8	X			
9	X			
10				X
11		X		

Моделирование физических процессов

№	1	2	3	4
1	X			
2	X			
3	X			
4		X		
5				X
6			X	

Компьютерное моделирование в экологии

№	1	2	3	4
1	X			
2				X
3		X		
4		X		
5			X	
6		X		
7				X

Моделирование случайных процессов

№	1	2	3	4
1				X
2	X			
3				X
4			X	
5		X		
6			X	
7			X	

Оглавление

Учебное издание

Могилев Александр Владимирович,
Пак Николай Инсебович,
Хеннер Евгений Карлович

Практикум по информатике

Учебное пособие

2-е издание, стереотипное

Редактор *Н. А. Носова*
Технический редактор *Е. Ф. Коржуева*
Компьютерная верстка: *Н. В. Соколова*
Корректоры *О. А. Королева, А. Б. Слоненко*

Изд. № А-390-II. Подписано в печать 16.02.2005. Формат 70×100/16. Печать офсетная.
Гарнитура «Таймс». Бумага тип. № 2. Усл. печ. л. 49,4. Тираж 3000 экз.
Заказ № 5213.

Издательский центр «Академия».
Санитарно-эпидемиологическое заключение № 77.99.02.953.Д.004796.07.04 от 20.07.2004.
117342, Москва, ул. Бутлерова, 17-Б, к. 360. Тел./факс: (095) 330-1092, 334-8337.

Отпечатано с готовых диапозитивов издательства
на ОАО «Тверской полиграфический комбинат»
170024, г. Тверь, пр-т Ленина, 5. Телефон: (0822) 44-42-15
Интернет/Home page - www.tverpk.ru Электронная почта (E-mail) - sales@tverpk.ru