Chapter 6

Control Structures

Control Structures

Objectives

After completing this unit you will be able to:

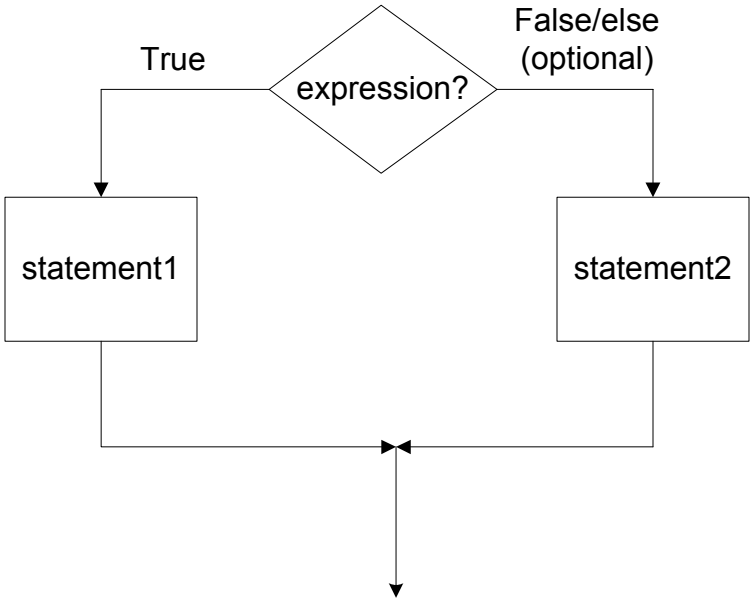
- **Use the common C# control structures to perform tests and loops.**

If Test

- **In an *if test* a *bool* expression is evaluated, and depending on the result, the "true branch" or "false branch" is executed.**

```
if (expression)
    statement 1;
else // optional
    statement 2;
```

- **If the *else* is omitted, then if the test is false, the control simply passes to the next statement after the *if* test.**



  - See **LeapYear**

Blocks

- **Several statements may be combined into a *block*, which is semantically equivalent to a single statement.**
  - A block is enclosed in curly braces.
  - Variables declared inside a block are local to that block.
- **The program *Swap* illustrates a block and the declaration of a local variable *temp* within the block.**
  - An attempt to use **temp** outside the block is a compiler error.

```
// Swap.cs

using System;

public class Swap
{
   public static int Main(string[] args)
   {
      int x = 5;
      int y = 12;
      Console.WriteLine("Before: x = {0}, y = {1}",
                        x, y);
      if (x < y)
      {
         int temp = x;
         x = y;
         y = temp;
      }
      Console.WriteLine("After: x = {0}, y = {1}",
                        x, y);
      // Console.WriteLine("temp = {0}", temp);
      return 0;
   }
}
```

Loops

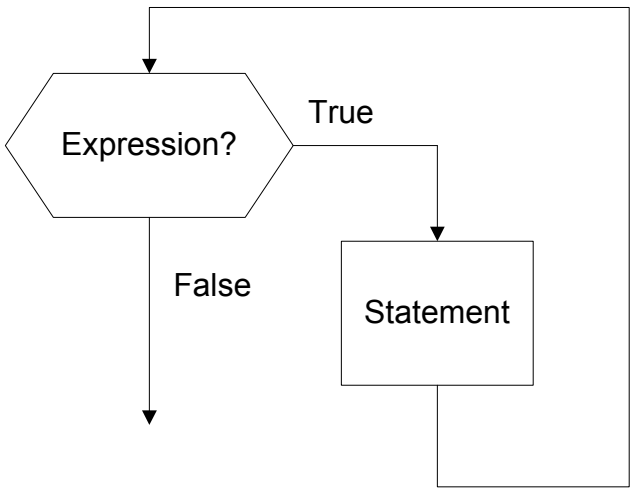
- **while**
- **for**
- **do/while**
- **foreach**
- **break**
- **continue**
- **goto**
- **switch**

While Loop

- **The most basic type of loop in C# is a** while **loop.**

```
while (expression)
{
    statements;
    ...
}
more statements;
```

  - Recommendation: Use blocks (in curly braces) even if there is only one statement in a loop.

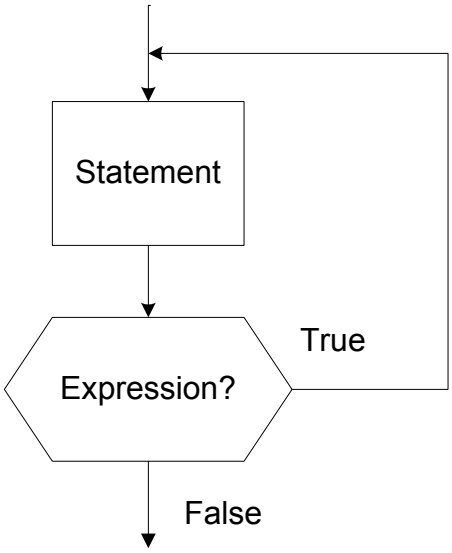


  - See **LeapYearLoop**

Do/While Loops

- **In the while loop, if the condition is initially false, then the loop is skipped.**
- **If you want a loop in which the body is always executed, use a do/while.**

```
do
{
   ...
}
while (expression); // ← note semicolon!
```

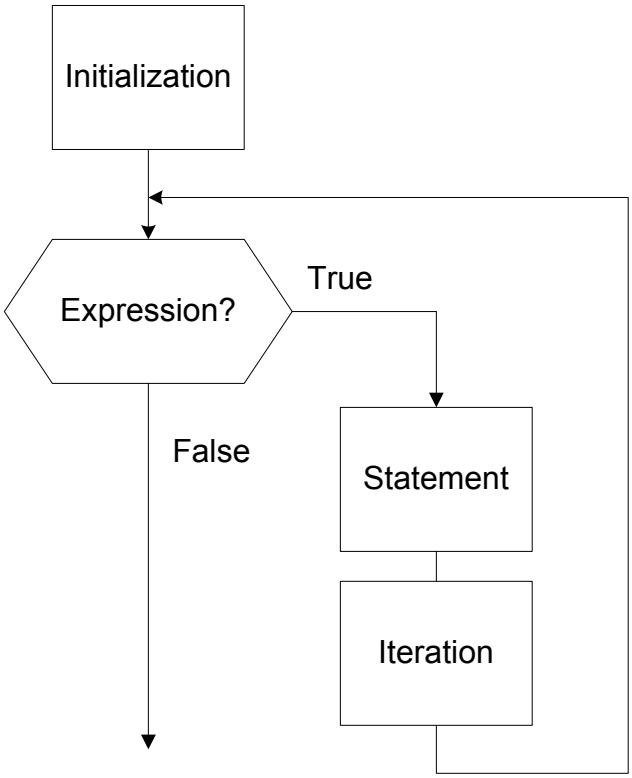


For Loops

- **A perennial favorite of C/C++/Java programmers, the *for* loop is the most flexible of the loop control structures.**

```
for (initialization; test; iteration)
{
   statements;
   ...
}
more statements;
```

  - The test must be a Boolean expression. Initialization and iteration can be nearly any kind of expression.



  - See **ForUp** and **ForDown**

Foreach Loop

- **The *foreach* loop is familiar to VB programmers, but is not present in C/C++/Java.**
- **It is a special loop for iterating through collections.**
- **In C#, an array is a collection, so you can use a *foreach* loop to iterate through an array.**

```
// ForEachLoop.cs

using System;

public class ForEachLoop
{
   public static int Main(string[] args)
   {
      int [] primes = {2, 3, 5, 7, 11, 13};
      int sum = 0;
      foreach (int prime in primes)
      {
         Console.Write("{0} ", prime);
         sum += prime;
      }
      Console.WriteLine();
      Console.WriteLine("sum = {0}", sum);
      return 0;
   }
}
```

- ***Foreach* will be covered in greater detail in a later chapter.**

Control Flow – Break and Continue

- **The *break* statement allow immediate exit from a loop.**
  - See **BreakSearch**
- **The *continue* statement bypasses the remainder of a loop, and transfers control to the beginning of the loop.**
  - See **ContinueLoop**

goto

- **Considered by purists to be evil, the infamous *goto* was even completely banned from some languages.**
  - Use **goto** sparingly and with great care.

```
goto label;
  ...
label:
  ...
```

```
// GotoSearch.cs
using System;
public class GotoSearch
{
   public static int Main(string[] args)
   {
      int [] primes = {2, 3, 5, 7, 11, 13};
      foreach (int prime in primes)
         Console.Write("{0} ", prime);
      Console.WriteLine();
      int target = 7;
      int i;
      for(i = 0; i < primes.Length; i++)
      {
         if (target == primes[i])
            goto found;
      }
      Console.WriteLine("{0} not found", target);
      return 0;
   found:
      Console.WriteLine("{0} found at {1}",
                        target,i);
      return 0;
   }
}
```

Switch

- **The *switch* statement can be substituted in some cases for a sequence of if tests.**
- **There are comparable control structures in other languages, such as**
  - **Select** in Visual Basic
  - **case** in Pascal
  - "computed goto" in FORTRAN.
  - **switch** in C/C++
- **Example Program:**
  - **SwitchDemo**

Switch in C# and C/C++

- **In C# after a particular case statement is executed, control does not automatically continue to the next statement.**
  - You must explicitly specify the next statement, typically by **break** or **goto** *label*.
  - This avoids a "gotcha" in C/C++

```
switch (code)
{
  case 1:
    goto case 2;
  case 2:
    Console.WriteLine("Low");
    break;
  case 3:
    Console.WriteLine("Medium");
    break;
  case 4:
    Console.WriteLine("High");
    break;
  default:
    Console.WriteLine("Special case");
    break;
}
```

- **In C# you may also switch on a *string* data type.**

Summary

- **If, while, do/while, for, and switch**
- **Avoid the use of goto.**
- **Special loop: foreach, used for collections**