

# 5

## Control Statements: Part I

© 2006 Pearson Education, Inc. All rights reserved.

### OBJECTIVES

In this chapter you will learn:

- To use the `if` and `if...else` selection statements to choose between alternative actions.
- To use the `while` repetition statement to execute statements in an application repeatedly.
- To use counter-controlled repetition and sentinel-controlled repetition.
- To develop algorithms through the process of top-down, stepwise refinement.
- To use the increment, decrement and compound assignment operators.

© 2006 Pearson Education, Inc. All rights reserved.

### 5.4 Control Structures

- **Program control**
  - Specifies the order in which actions execute in a program
- **Sequential execution**
  - Statements are executed one after the other in the order in which they are written
- **Transfer of control**
  - Specifying the next statement to execute that is not necessarily the next one in order
    - Selection structure
    - Repetition structure

© 2006 Pearson Education, Inc. All rights reserved.

### 5.4 Control Structures (Cont.)

- **All programs could be written with three control structures**
  - Sequence structure
  - Selection structure
  - Repetition structure

© 2006 Pearson Education, Inc. All rights reserved.

### 5.4 Control Structures (Cont.)

- **Selection Statements**
  - `if` statement
    - Single-selection statement
  - `if...else` statement
    - Double-selection statement
  - `switch` statement
    - Multiple-selection statement

© 2006 Pearson Education, Inc. All rights reserved.

### 5.4 Control Structures (Cont.)

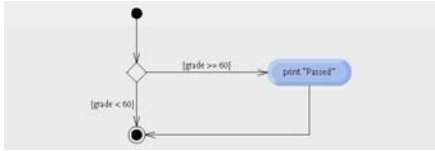
- **Repetition statements**
  - Also known as looping statements
  - Repeatedly performs an action while its loop-continuation condition remains true
  - `while` statement
    - Performs the actions in its body zero or more times
  - `do...while` statement
    - Performs the actions in its body one or more times
  - `for` statement
    - Performs the actions in its body zero or more times
  - `foreach` statement
    - Performs the actions in its body zero or more times

© 2006 Pearson Education, Inc. All rights reserved.

## 5.5 if Single-Selection Statement

- **if statements**

- Execute an action if the specified condition is true

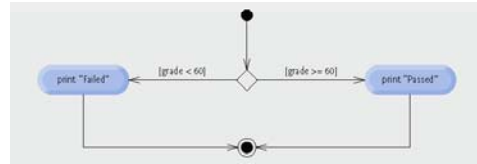


© 2006 Pearson Education, Inc. All rights reserved.

## 5.6 if...else Double-Selection Statement

- **if...else statement**

- Executes one action if the specified condition is true or a different action if the specified condition is false



© 2006 Pearson Education, Inc. All rights reserved.

## 5.6 if...else Double-Selection Statement (Cont.)

- **Nested if...else statements**

- if...else statements can be put inside other if...else statements

© 2006 Pearson Education, Inc. All rights reserved.

## 5.6 if...else Double-Selection Statement (Cont.)

- **Dangling-else problem**

- What is the output when  $x=60$
- ```

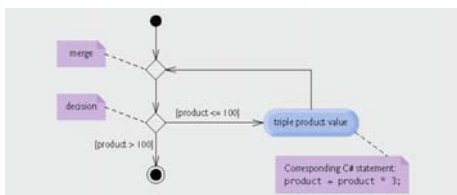
if (x > 70) if (x < 65) output ('a') else
output ('b')
  
```
- else's are always associated with the immediately preceding if unless otherwise specified by braces { }

© 2006 Pearson Education, Inc. All rights reserved.

## 5.7 while Repetition Statement

- **while statement**

- Repeats an action while its loop-continuation condition remains true
- `while (condition) { ... }`



© 2006 Pearson Education, Inc. All rights reserved.

## Exercise

- **Example: write all even numbers up to 200**
- `int i = 0; while (i <= 200) { Console.WriteLine("{0}", i); i=i+2; }`

© 2006 Pearson Education, Inc. All rights reserved.

## 5.8 Formulating Algorithms: Counter-Controlled Repetition

- Counter-controlled repetition
  - Also known as a definite repetition
  - Use a counter variable to count the number of times a loop is iterated

## 5.3 Pseudocode

- Pseudocode
  - An informal language similar to English
  - Helps programmers develop algorithms
  - Does not run on computers
  - Should contain input, output and calculation actions
  - Should not contain variable declarations

```
1 set total to zero
2 set grade counter to one
3
4 while grade counter is less than or equal to ten
5   prompt the user to enter the next grade
6   Input the next grade
7   add the grade into the total
8   add one to the grade counter
9
10 set the class average to the total divided by ten
11 print the class average
```

Fig. 5.5 | Pseudocode algorithm that uses counter-controlled repetition to solve the class-average problem.

```
1 // Fig. 5.6: GradeBook.cs
2 // GradeBook class that solves class-average problem using
3 // counter-controlled repetition.
4 using System;
5
6 public class GradeBook
7 {
```

Outline

GradeBook.cs

(1 of 3)

```
28 // display a welcome message to the GradeBook user
29 public void DisplayMessage()
30 {
31     // property CourseName gets the name of the course
32     Console.WriteLine("Welcome to the grade book!");
33 }
34 // determine class average based on 10 grades entered by user
35 public void DetermineClassAverage()
36 {
37     // initialization phase
38     int total; // sum of the grades entered by user
39     int gradeCounter; // number of the grade to be entered next
40     int grade; // grade value entered by the user
41     int average; // average of the grades
42     // initialization phase
43     total = 0; // initialize the total
44     gradeCounter = 1; // initialize the loop counter
45 }
46
47
```

Outline

GradeBook.cs

(2 of 3)

Initialize counter to 1

Counter to control while loop

Declare local int variables total, gradeCounter, grade and average

```
48 // processing phase
49 while (gradeCounter <= 10) // loop 10 times
50 {
51     Console.WriteLine("Enter grade: "); // prompt the user
52     grade = Convert.ToInt32(Console.ReadLine()); // read grade
53     total = total + grade; // add the grade to total
54     gradeCounter = gradeCounter + 1; // increment the counter by 1
55 } // end while
56 // termination phase
57 average = total / 10; // integer division yields integer result
58
59 // display total and average of grades
60 Console.WriteLine("\nTotal of all 10 grades is {0}", total);
61 Console.WriteLine("Class average is {0}", average);
62 } // end method DetermineClassAverage
63 // end class GradeBook
64
```

Outline

GradeBook.cs

(3 of 3)

Display results

Calculate average grade

Increment the counter variable gradeCounter

while loop iterates as long as gradeCounter <= 10

```

1 // Fig. 5.7: GradeBookTest.cs
2 // Create GradeBook object and invoke its DetermineClassAverage method.
3 public class GradeBookTest
4 {
5     public static void Main( string[] args )
6     {
7         // create GradeBook object myGradeBook and
8         // pass course name to constructor
9         GradeBook myGradeBook = new GradeBook( );
10        myGradeBook.DisplayMessage(); // display welcome message
11        myGradeBook.DetermineClassAverage(); // find average
12    } // end Main
13 } // end class GradeBookTest

```

**Outline**  
GradeBookTest.cs

- Create a new GradeBook object
- Pass the course's name to the GradeBook constructor as a string
- Call GradeBook's DetermineClassAverage method

```

Welcome to the grade book
Enter grade: 88
Enter grade: 79
Enter grade: 95
Enter grade: 100
Enter grade: 48
Enter grade: 88
Enter grade: 92
Enter grade: 83
Enter grade: 90
Enter grade: 85
Total of all 10 grades is 848
Class average is 84

```

© 2006 Pearson Education, Inc. All rights reserved.

## 5.9 Formulating Algorithms: Sentinel-Controlled Repetition

- Sentinel-controlled repetition
  - Also known as indefinite repetition
  - Use a sentinel value (also known as a signal, dummy or flag value)
    - A sentinel value cannot also be a valid input value
- Choosing a sentinel value that is also a legitimate data value is a logic error.

© 2006 Pearson Education, Inc. All rights reserved.

## Software Engineering Observation 5.3

Many applications can be divided logically into three phases: an initialization phase that initializes the variables; a processing phase that inputs data values and adjusts application variables (e.g., counters and totals) accordingly; and a termination phase that calculates and outputs the final results.

© 2006 Pearson Education, Inc. All rights reserved.

```

1 Initialize total to zero
2 Initialize counter to zero
3
4 prompt the user to enter the first grade
5 Input the first grade (possibly the sentinel)
6
7 while the user has not yet entered the sentinel
8     add this grade into the running total
9     add one to the grade counter
10    prompt the user to enter the next grade
11    Input the next grade (possibly the sentinel)
12
13 If the counter is not equal to zero
14     set the average to the total divided by the counter
15     print the average
16 else
17     print "No grades were entered"

```

Fig. 5.8 | Class-average problem pseudocode algorithm sentinel-controlled repetition.

© 2006 Pearson Education, Inc. All rights reserved.

```

28 // display a welcome message to the GradeBook user
29 public void DisplayMessage()
30 {
31     Console.WriteLine( "Welcome to the grade book for\n{0}\n",
32         CourseName );
33 } // end method DisplayMessage
34
35 // determine the average of an arbitrary number of grades
36 // (2 of 3)
37 public void DetermineClassAverage()
38 {
39     int total; // sum of grades
40     int gradeCounter; // number of grades entered
41     int grade; // grade value
42     double average; // number with decimal point for average
43
44     // initialization phase
45     total = 0; // initialize total
46     gradeCounter = 0; // initialize loop counter
47
48     // processing phase
49     // prompt for input and read grade from user
50     Console.WriteLine( "Enter grade or -1 to quit: ");
51     grade = Convert.ToInt32( Console.ReadLine() );
52
53     // loop until sentinel value read from user
54     while ( grade != -1 )
55     {
56         total = total + grade; // add grade to total
57         gradeCounter = gradeCounter + 1; // increment counter

```

**Outline**  
GradeBook.cs

- Declare method DisplayMessage
- Declare method DetermineClassAverage
- Declare local int variables total, gradeCounter and grade and double variable average
- while loop iterates as long as grade != the sentinel value, -1

© 2006 Pearson Education, Inc. All rights reserved.

```

58 // prompt for input and read next grade from user
59 Console.WriteLine( "Enter grade or -1 to quit: ");
60 grade = Convert.ToInt32( Console.ReadLine() );
61 } // end while
62
63 // termination phase
64 // if user entered at least one grade...
65 if ( gradeCounter != 0 )
66 {
67     // calculate average of all grades entered
68     average = (double) total / gradeCounter;
69
70     // display total and average (with two digits of precision)
71     Console.WriteLine( "\nTotal of the {0} grades entered is {1}",
72         gradeCounter, total );
73     Console.WriteLine( "Class average is {0:F2}; average );
74 } // end if
75 else // no grades were entered, so output error message
76     Console.WriteLine( "No grades were entered" );
77 } // end method DetermineClassAverage
78 } // end class GradeBook

```

**Outline**  
GradeBook.cs

- Calculate average grade using (double) to perform explicit conversion
- Display average grade
- Display "No grades were entered" message

© 2006 Pearson Education, Inc. All rights reserved.

25

```

1 // Fig. 5.10: GradeBookTest.cs
2 // Create GradeBook object and invoke its DetermineClassAverage method.
3 public class GradeBookTest
4 {
5     public static void Main( string[] args )
6     {
7         // create GradeBook object myGradeBook and
8         // pass course name to constructor
9         GradeBook myGradeBook = new GradeBook( "CS" );
10        myGradeBook.DisplayMessage(); // display welcome message
11        myGradeBook.DetermineClassAverage(); // find average
12    } // end Main
13 } // end class GradeBookTest

```

Outline

GradeBookTest.cs

Create a new GradeBook object

Pass the course's name to the GradeBook constructor as a string

Call GradeBook's DetermineClassAverage method

```

Welcome to the grade book
Enter grade or -1 to quit: 96
Enter grade or -1 to quit: 88
Enter grade or -1 to quit: 79
Enter grade or -1 to quit: -1
Total of the 3 grades entered is 263
Class average is 87.67

```

© 2006 Pearson Education, Inc. All rights reserved.

26

```

1 Initialize passes to zero
2 Initialize failures to zero
3 Initialize student counter to one
4
5 while student counter is less than or equal to 10
6     prompt the user to enter the next exam result
7     input the next exam result
8
9     if the student passed
10        add one to passes
11    else
12        add one to failures
13
14    add one to student counter
15
16 print the number of passes
17 print the number of failures
18
19 if more than eight students passed
20     print "Raise tuition"

```

Fig. 5.11 | Pseudocode for the examination-results problem.

© 2006 Pearson Education, Inc. All rights reserved.

27

## 5.11 Compound Assignment Operators

- Compound assignment operators
  - An assignment statement of the form:  $variable = variable\ operator\ expression;$  where *operator* is +, -, \*, / or % can be written as:  $variable\ operator = expression;$
  - example:  $c = c + 3;$  can be written as  $c += 3;$ 
    - This statement adds 3 to the value in variable *c* and stores the result in variable *c*

© 2006 Pearson Education, Inc. All rights reserved.

28

## 5.12 Increment and Decrement Operators

- Unary increment and decrement operators
  - Unary increment operator (++) adds one to its operand
  - Unary decrement operator (--) subtracts one from its operand
  - Prefix increment (and decrement) operator
    - Changes the value of its operand, then uses the new value of the operand in the expression in which the operation appears
  - Postfix increment (and decrement) operator
    - Uses the current value of its operand in the expression in which the operation appears, then changes the value of the operand

© 2006 Pearson Education, Inc. All rights reserved.

29

| Operator | Called            | Sample expression | Explanation                                                                                                  |
|----------|-------------------|-------------------|--------------------------------------------------------------------------------------------------------------|
| ++       | prefix increment  | ++a               | Increment <i>a</i> by 1, then use the new value of <i>a</i> in the expression in which <i>a</i> resides.     |
| ++       | postfix increment | a++               | Use the current value of <i>a</i> in the expression in which <i>a</i> resides, then increment <i>a</i> by 1. |
| --       | prefix decrement  | --b               | Decrement <i>b</i> by 1, then use the new value of <i>b</i> in the expression in which <i>b</i> resides.     |
| --       | postfix decrement | b--               | Use the current value of <i>b</i> in the expression in which <i>b</i> resides, then decrement <i>b</i> by 1. |

Fig. 5.15 | Increment and decrement operators.

© 2006 Pearson Education, Inc. All rights reserved.

30

```

1 // Fig. 5.16: Increment.cs
2 // Prefix increment and postfix increment operators.
3 using System;
4
5 public class Increment
6 {
7     public static void Main( string[] args )
8     {
9         int c;
10
11        // demonstrate postfix increment operator
12        c = 5; // assign 5 to c
13        Console.WriteLine( c ); // print 5
14        Console.WriteLine( c++ ); // print 5 again, then increment
15        Console.WriteLine( c ); // print 6
16
17        Console.WriteLine(); // skip a line
18
19        // demonstrate prefix increment operator
20        c = 5; // assign 5 to c
21        Console.WriteLine( c ); // print 5
22        Console.WriteLine( ++c ); // increment, then print 6
23        Console.WriteLine( c ); // print 6 again
24    } // end Main
25 } // end class Increment

```

Outline

Increment.cs

Postincrementing the c variable

Preincrementing the c variable

© 2006 Pearson Education, Inc. All rights reserved.

## Common Programming Error 5.8

---

Attempting to use the increment or decrement operator on an expression other than one to which a value can be assigned is a syntax error. For example, writing `++(x + 1)` is a syntax error because `(x + 1)` is not a variable.

---

