C# Programming: From Problem Analysis to Program Design, 5th edition

Chapter 6

1. b. 1000

2. c. to allow statements to be repeated

3. a. foreach

4. d. do…while

5. b. for

6. d. continue

7. b. while (counter < 10)

8. b. for (num = 0; num < 10; num++)

9. d. 2

10. c. computes the sum of even integers from1 through n

11. b. n < 10

12. a. 0

13. e. a and c

14. d. curly braces

15. d. five times

16. c. the number of negative items entered

17. a. counter-controlled loop

18. a. 20

19. d. 12

20. d. an infinite loop

21. for (int counter = 100; counter > 0; counter--)

{

Console.WriteLine(counter);

}

int counter = 100; // while example

while (counter > 0)

{

Console.WriteLine(counter);

counter--;

}

Final result is the same for all three loop constructs. Both the for statement and the while statement are tested before entry into the loop.

22. for (int val = 10; val < 101; val += 3)

{

Console.WriteLine(val);

}

23. int temp = 0;

int totalTemp = 0;

int tempCount = 0;

double averageTemp;

string inValue;

Console.WriteLine(“You may enter any number of temperatures.”);

Console.WriteLine(“To stop entering values, type -99”);

inValue = Console.ReadLine( );

temp = int.Parse(inValue);

while (temp != -99)

{

tempCount++;

totalTemp += temp;

Console.WriteLine(“Enter next temperature.”);

Console.WriteLine(“To stop entering values, type -99”);

inValue = Console.ReadLine( );

temp = int.Parse(inValue);

}

averageTemp = totalTemp / tempCount;

Console.WriteLine(“Average Temperature: ” + averageTemp);

24. Random rNum = new Random( );

int randomValue,

totalRandomValues = 0,

sumOfRandomValues = 0;

randomValue = rNum.Next(25, 75);

while (randomValue < 60)

{

sumOfRandomValues += randomValue;

totalRandomValues++;

randomValue = rNum.Next(25, 76);

}

Console.WriteLine(“Total number of Random Values “ +

“generated is {0}”, totalRandomValues);

Console.WriteLine(“Sum of all Random Values “ +

“entered is {0}”, sumOfRandomValues);

25.

i j Output area:

0 4 0 4

0 3 0 3

0 2 0 2

0 1 0 1

0 0

1 4 1 4

1 3 1 3

1 2 1 2

1 1 1 1

1 0

2 4 2 4

2 3 2 3

2 2 2 2

2 1 2 1

2 0

3