Review Questions

1. A structure that allows repeated execution of a block of statements is a(n) _____ .

   a. sequence
   b. selection
   c. array
   d. loop

2. The body of a `while` loop can consist of _____ .

   a. a single statement
   b. a block of statements within curly braces
   c. either a or b
   d. neither a nor b

3. A loop that never ends is called an _____ loop.

   a. indefinite
   b. interminable
   c. infinite
   d. intermediate

4. Which of the following is not required of a loop control variable in a correctly working loop?

   a. It is reset to its initial value before the loop ends.
   b. It is initialized before the loop starts.
   c. It is tested.
   d. It is altered in the loop body.

5. A `while` loop with an empty body contains no _____ .

   a. statements
   b. loop control variable
   c. curly braces
   d. test within the parentheses of the `while` statement

6. A loop for which you do not know the number of iterations when you write it is a(n) _____ .

   a. indefinite loop
   b. definite loop
   c. counted loop
   d. `for` loop

7. What is the major advantage of using a `for` loop instead of a `while` loop?

   a. With a `for` loop, it is impossible to create an infinite loop.
   b. The loop control variable is initialized, tested, and altered all in one place.
   c. It is the only way to achieve an indefinite loop.
   d. Unlike with a `while` loop, the execution of multiple statements can depend on the test condition.

Copyright 2018 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. WCN 02-200-202

8. A `for` loop statement must contain _____ .
   a. two semicolons
   b. three commas
   c. four dots
   d. five pipes

9. In a `for` statement, the section before the first semicolon executes _____ .
   a. once
   b. once prior to each loop iteration
   c. once after each loop iteration
   d. one less time than the initial loop control variable value

10. The three sections of the `for` loop are most commonly used for _____ the loop control variable.
    a. testing, outputting, and incrementing
    b. initializing, testing, and incrementing
    c. incrementing, selecting, and testing
    d. initializing, converting, and outputting

11. Which loop is most convenient to use if the loop body must always execute at least once?
    a. a `while` loop
    b. a `for` loop
    c. a `do` loop
    d. an `if` loop

12. The loop control variable is checked at the bottom of which kind of loop?
    a. a `while` loop
    b. a `for` loop
    c. a `do` loop
    d. all of the above

13. A `for` loop is an example of a(n) _____ loop.
    a. untested
    b. pretest
    c. posttest
    d. infinite

14. A `while` loop is an example of a(n) _____ loop.
    a. untested
    b. pretest
    c. posttest
    d. infinite

15. When a loop is placed within another loop, the loops are said to be _____ .
    a. infinite
    b. bubbled
    c. overlapping
    d. nested

Copyright 2018 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. WCN 02-200-202

16. What does the following code segment display?

```
a = 1;
while (a < 5);
{
   Write("{0} ", a);
   ++a;
}
```

a. 1 2 3 4
b. 1
c. 4
d. nothing

17. What is the output of the following code segment?

```
s=1;
while(s < 4)
   ++s;
   Write("{0} ", s);
```

a. 1
b. 4
c. 1 2 3 4
d. 2 3 4

18. What is the output of the following code segment?

```
j = 5;
while(j > 0)
{
   Write("{0} ", j);
   j--;
}
```

a. 0
b. 5
c. 5 4 3 2 1
d. 5 4 3 2 1 0

19. What does the following code segment display?

```
for(f = 0; f < 3; ++f);
   Write("{0} ", f);
```

a. 0
b. 0 1 2
c. 3
d. nothing

20. What does the following code segment display?

```
for(t = 0; t < 3; ++t)
   Write("{0} ", t);
```

a. 0
b. 0 1
c. 0 1 2
d. 0 1 2 3

Copyright 2018 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. WCN 02-200-202

Exercises

Programming Exercises

1. Write an application named **SumFiveInts** that allows the user to enter five integers and displays their sum.

2. Write an application named **SumInts** that allows the user to enter any number of integers continuously until the user enters *999*. Display the sum of the values entered, not including 999.

3. Write an application named **EnterUppercaseLetters** that asks the user to type an uppercase letter from the keyboard. If the character entered is an uppercase letter, display *OK*; if it is not an uppercase letter, display an error message. The program continues until the user types an exclamation point.

4. Write an application named **DailyTemps** that continuously prompts a user for a series of daily high temperatures until the user enters a sentinel value. Valid temperatures range from −20 through 130 Fahrenheit. When the user enters a valid temperature, add it to a total; when the user enters an invalid temperature, display an error message. Before the program ends, display the number of temperatures entered and the average temperature.

5. Danielle, Edward, and Francis are three salespeople at Holiday Homes. Write an application named **HomeSales** that prompts the user for a salesperson initial (*D*, *E*, or *F*). Either uppercase or lowercase initials are valid. While the user does not type *Z*, continue by prompting for the amount of a sale. Issue an error message for any invalid initials entered. Keep a running total of the amounts sold by each salesperson. After the user types *Z* or *z* for an initial, display each salesperson's total, a grand total for all sales, and the name of the salesperson with the highest total.

6. Write an application named **DisplayMultiplicationTable** that displays a table of the products of every combination of two integers from 1 through 10.

7. Write an application named **MultiplicationTable** that prompts the user for an integer value, for example *7*. Then display the product of every integer from 1 through 10 when multiplied by the entered value. For example, the first three lines of the table might read $1 \times 7 = 7$, $2 \times 7 = 14$, and $3 \times 7 = 21$.

8. Write an application named **OddNums** that displays all the odd numbers from 1 through 99.

9. Write an application named **Sum200** that sums the integers from 1 through 200. Display the running total when the program is halfway complete (after the first 100 numbers), and at the end.

Copyright 2018 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. WCN 02-200-202

10. Write an application named **Perfect** that displays every perfect number from 1 through 10,000. A number is perfect if it equals the sum of all the smaller positive integers that divide evenly into it. For example, 6 is perfect because 1, 2, and 3 divide evenly into it and their sum is 6.

11. In a "You Do It" section of this chapter, you created a tipping table for patrons to use when analyzing their restaurant bills. Now, create a modified program named **TippingTable3** in which each of the following values is obtained from user input:
    - The lowest tipping percentage
    - The highest tipping percentage
    - The lowest possible restaurant bill
    - The highest restaurant bill

12. Write a program named **WebAddress** that asks a user for a business name. Suggest a good Web address by adding *www.* to the front of the name, removing all spaces from the name, and adding *.com* to the end of the name. For example, a good Web address for Acme Plumbing and Supply is *www.AcmePlumbingandSupply.com*.

13. Write a program named **CountVowels** that accepts a phrase from the user and counts the number of vowels in the phrase. For this exercise, count both uppercase and lowercase vowels, but do not consider *y* to be a vowel.

14. In Chapter 4, you created a program that generates a random number, allows a user to guess it, and displays a message indicating whether the guess is too low, too high, or correct. Now, create a modified program called **GuessingGame2** in which the user can continue to enter values until the correct guess is made. After the user guesses correctly, display the number of guesses made.

Recall that you can generate a random number whose value is at least `min` and less than `max` using the following statements:

```
Random ranNumber = new Random();
int randomNumber;
randomNumber = ranNumber.Next(min, max);
```

15. Modify the GuessingGame2 program to create a program called **GuessingGame3** in which the player is criticized for making a "dumb" guess. For example, if the player guesses that the random number is 4 and is told that the guess is too low, and then the player subsequently makes a guess lower than 4, display a message that the user should have known not to make such a low guess.

Copyright 2018 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. WCN 02-200-202