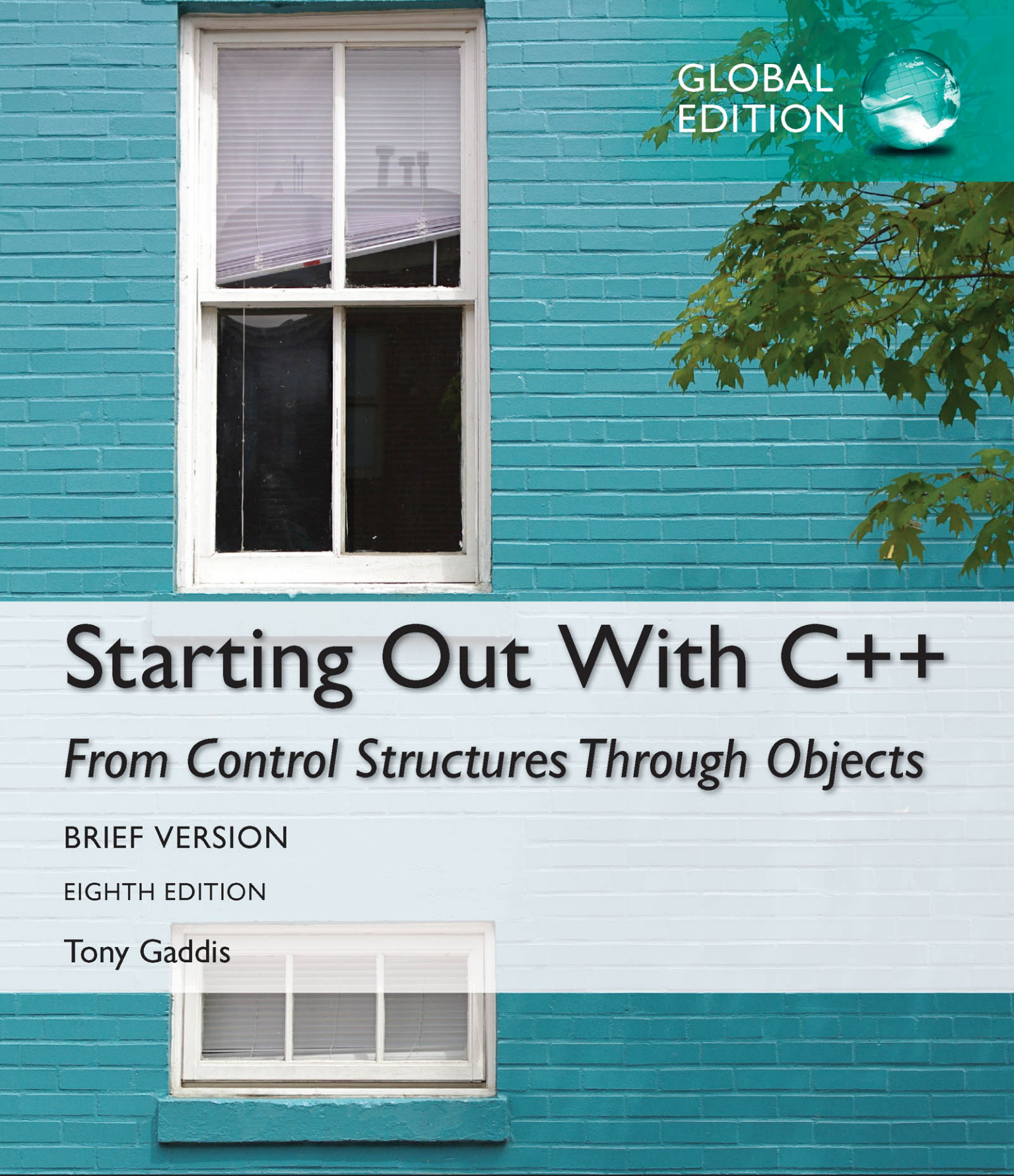

Starting Out With C++

From Control Structures Through Objects

BRIEF VERSION

EIGHTH EDITION

Tony Gaddis

ALWAYS LEARNING

PEARSON

STARTING OUT WITH

C++

From Control Structures through Objects

BRIEF VERSION
EIGHTH EDITION
GLOBAL EDITION

Sometimes it's necessary to stop a loop before it goes through all its iterations. The `break` statement, which was used with `switch` in Chapter 4, can also be placed inside a loop. When it is encountered, the loop stops, and the program jumps to the statement immediately following the loop.

The `while` loop in the following program segment appears to execute 10 times, but the `break` statement causes it to stop after the fifth iteration.

```
int count = 0;
while (count++ < 10)
{
     cout << count << endl;
     if (count == 5)
          break;
}
```

Program 5-25 uses the `break` statement to interrupt a `for` loop. The program asks the user for a number and then displays the value of that number raised to the powers of 0 through 10. The user can stop the loop at any time by entering Q.

Program 5-25

```
1  // This program raises the user's number to the powers
2  // of 0 through 10.
3  #include <iostream>
4  #include <cmath>
5  using namespace std;
6
7  int main()
8  {
9       double value;
10      char choice;
11
12      cout << "Enter a number: ";
13      cin >> value;
14      cout << "This program will raise " << value;
15      cout << " to the powers of 0 through 10.\n";
16      for (int count = 0; count <= 10; count++)
17      {
18           cout << value << " raised to the power of ";
19           cout << count << " is " << pow(value, count);
20           cout << "\nEnter Q to quit or any other key ";
21           cout << "to continue. ";
22           cin >> choice;
23           if (choice == 'Q' || choice == 'q')
24                break;
25      }
26      return 0;
27 }
```

(program output continues)

Program 5-25 *(continued)*

Program Output with Example Input Shown in Bold

```
Enter a number: 2 [Enter]
This program will raise 2 to the powers of 0 through 10.
2 raised to the power of 0 is 1
Enter Q to quit or any other key to continue. C [Enter]
2 raised to the power of 1 is 2
Enter Q to quit or any other key to continue. C [Enter]
2 raised to the power of 2 is 4
Enter Q to quit or any other key to continue. Q [Enter]
```

Using `break` in a Nested Loop

In a nested loop, the `break` statement only interrupts the loop it is placed in. The following program segment displays five rows of asterisks on the screen. The outer loop controls the number of rows, and the inner loop controls the number of asterisks in each row. The inner loop is designed to display 20 asterisks, but the `break` statement stops it during the eleventh iteration.

```
for (int row = 0; row < 5; row++)
{
     for (int star = 0; star < 20; star++)
     {
          cout << '*';
          if (star == 10)
               break;
     }
     cout << endl;
}
```

The output of the program segment above is:

```
***********
***********
***********
***********
***********
```

The `continue` Statement

The `continue` statement causes the current iteration of a loop to end immediately. When `continue` is encountered, all the statements in the body of the loop that appear after it are ignored, and the loop prepares for the next iteration.

In a `while` loop, this means the program jumps to the test expression at the top of the loop. As usual, if the expression is still true, the next iteration begins. In a `do-while` loop, the program jumps to the test expression at the bottom of the loop, which determines whether the next iteration will begin. In a `for` loop, `continue` causes the update expression to be executed and then the test expression to be evaluated.

The following program segment demonstrates the use of `continue` in a `while` loop:

```
int testVal = 0;
while (testVal++ < 10)
{
     if (testVal == 4)
          continue;
     cout << testVal << " ";
}
```

This loop looks like it displays the integers 1 through 10. When `testVal` is equal to 4, however, the `continue` statement causes the loop to skip the `cout` statement and begin the next iteration. The output of the loop is

```
1 2 3 5 6 7 8 9 10
```

Program 5-26 demonstrates the `continue` statement. The program calculates the charges for DVD rentals, where current releases cost $3.50 and all others cost $2.50. If a customer rents several DVDs, every third one is free. The `continue` statement is used to skip the part of the loop that calculates the charges for every third DVD.

Program 5-26

```
1  // This program calculates the charges for DVD rentals.
2  // Every third DVD is free.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
6
7  int main()
8  {
9       int dvdCount = 1;     // DVD counter
10      int numDVDs;          // Number of DVDs rented
11      double total = 0.0;   // Accumulator
12      char current;         // Current release, Y or N
13
14      // Get the number of DVDs.
15      cout << "How many DVDs are being rented? ";
16      cin >> numDVDs;
17
18      // Determine the charges.
19      do
20      {
21           if ((dvdCount % 3) == 0)
22           {
23                cout << "DVD #" << dvdCount << " is free!\n";
24                continue; // Immediately start the next iteration
25           }
26           cout << "Is DVD #" << dvdCount;
27           cout << " a current release? (Y/N) ";
28           cin >> current;
29           if (current == 'Y' || current == 'y')
30                total += 3.50;
31           else
32                total += 2.50;
33      } while (dvdCount++ < numDVDs);
34
```

(program continues)

Program 5-26 *(continued)*

```
35          // Display the total.
36          cout << fixed << showpoint << setprecision(2);
37          cout << "The total is $" << total << endl;
38          return 0;
39    }
```

Program Output with Example Input Shown in Bold

```
How many DVDs are being rented? 6 [Enter]
Is DVD #1 a current release? (Y/N) y [Enter]
Is DVD #2 a current release? (Y/N) n [Enter]
DVD #3 is free!
Is DVD #4 a current release? (Y/N) n [Enter]
Is DVD #5 a current release? (Y/N) y [Enter]
DVD #6 is free!
The total is $12.00
```

Case Study: See the Loan Amortization Case Study on this book's companion Web site at www.pearsonglobaleditions.com/gaddis.

Review Questions and Exercises

Short Answer

1. Why should you indent the statements in the body of a loop?
2. Describe the difference between pretest loops and posttest loops.
3. Why are the statements in the body of a loop called conditionally executed statements?
4. What is the difference between the `while` loop and the `do-while` loop?
5. Which loop should you use in situations where you wish the loop to repeat until the test expression is false, and the loop should not execute if the test expression is false to begin with?
6. Which loop should you use in situations where you wish the loop to repeat until the test expression is false, but the loop should execute at least one time?
7. Explain the concept of a user-controlled loop. Which loop(s) can be used as a user-controlled loop? Give an example to support your answer.
8. Why is it critical that counter variables be properly initialized?
9. Why is it critical that accumulator variables be properly initialized?
10. Why should you be careful not to place a statement in the body of a `for` loop that changes the value of the loop's counter variable?
11. What header file do you need to include in a program that performs file operations?
12. What data type do you use when you want to create a file stream object that can write data to a file?
13. What data type do you use when you want to create a file stream object that can read data from a file?
14. Why should a program close a file when it's finished using it?

15. What is a file's read position? Where is the read position when a file is first opened for reading?

Fill-in-the-Blank

16. To _____ a value means to increase it by one, and to _____ a value means to decrease it by one.
17. In the _____ mode of increment operator, the precedence of the increment operator is higher than all other operators in the expression except brackets.
18. When the increment or decrement operator is placed after the operand (or to the operand's right), the operator is being used in _____ mode.
19. The three looping structures that C++ supports are _____, _____, and _____.
20. The first line of a while loop or a `for` loop is generally called a _____.
21. A loop that evaluates its test expression before each repetition is a(n) _____ loop.
22. A _____ loop is a type of posttest loop.
23. A loop that does not have a way of stopping is a(n) _____ loop.
24. A block of statements is enclosed within _____.
25. A(n) _____ is a sum of numbers that accumulates with each iteration of a loop.
26. A(n) _____ is a variable that is initialized to some starting value, usually zero, and then has numbers added to it in each iteration of a loop.
27. A(n) _____ is a special value that marks the end of a series of values.
28. The _____ loop always iterates at least once.
29. The _____ and _____ loops will not iterate at all if their test expressions are false to start with.
30. The _____ loop is ideal for situations that require a counter.
31. Inside the `for` loop's parentheses, the first expression is the _____ , the second expression is the _____ , and the third expression is the _____.
32. A loop that is inside another is called a(n) _____ loop.
33. The header file `fstream` contains all declarations necessary for _____.
34. An object of _____ data type is created when an existing file is opened such that the data can be read from it.

Algorithm Workbench

35. Write a `while` loop that asks the user for a number (integer) and finds the sum of all the numbers entered. The process is continued till the sum of the numbers exceeds 100. Initialize the sum as 0 and display all partial sums.
36. Write a `do-while` loop that asks the user for a number. The square root of the number is then computed and displayed. The process continues as long as the user inputs a positive number; otherwise the loop terminates. Assume that the first number is 1.
37. Write a `for` loop to display the following set of numbers:

    ```
    2, 4, 8, 16 ... 1024, 2048
    ```

38. Write a `for` loop to display all the even numbers from 100 to 250. Also include a counter variable that counts the number of even numbers generated.
39. Write a nested loop that displays 10 rows of '#' characters. There should be 15 '#' characters in each row.

40. Convert the following `while` loop to a `do-while` loop:

```
int x = 1;
while (x > 0)
{
     cout << "enter a number: ";
     cin >> x;
}
```

41. Convert the following `do-while` loop to a `for` loop:

```
int response;
do
{
     cout << "Hello! Press 1 to exit loop...";
     cin >> response;
} while ( response != 1 );
```

42. Convert the following `while` loop to a `for` loop:

```
int count = 0;
while (count < 50)
{
     cout << "count is " << count << endl;
     count++;
}
```

43. Convert the following `for` loop to a `while` loop:

```
for (int x = 50; x > 0; x--)
{
     cout << x << " seconds to go.\n";
}
```

44. Write code that does the following: Opens an output file with the filename Numbers.txt, uses a loop to write the numbers 1 through 100 to the file, and then closes the file.

45. Write code that does the following: Opens the Numbers.txt file that was created by the code you wrote in question 44, reads all of the numbers from the file and displays them, and then closes the file.

46. Modify the code that you wrote in question 45 so it adds all of the numbers read from the file and displays their total.

True or False

47. T F The operand of the increment and decrement operators can be any valid mathematical expression.

48. T F The value of the variable `z` after execution of the following statements will be 11:

```
x = y = 5;
z = x + ++y;
```

49. T F The value of the variable `z` after execution of the following statements will be 11:

```
x = y = 5;
z = x++ + y;
```

50. T F The `while` loop is a pretest loop.

51. T F The `do-while` loop is a pretest loop.

52. T F The `for` loop is a posttest loop.

53. T F It is not necessary to initialize counter variables.