12.14 Briefly describe what each of the following statements does:

```
file.seekp(100L, ios::beg);
file.seekp(-10L, ios::end);
file.seekg(-25L, ios::cur);
file.seekg(30L, ios::cur);
```

12.15 Describe the mode that each of the following statements causes a file to be opened in:

```
file.open("info.dat", ios::in | ios::out);
file.open("info.dat", ios::in | ios::app);
file.open("info.dat", ios::in | ios::out | ios::ate);
file.open("info.dat", ios::in | ios::out | ios::binary);
```

For another example of this chapter's topics, see the High Adventure Travel Part 3 Case Study, available on the book's companion Web site at www.pearsonhighered.com/gaddis.

Review Questions and Exercises

Short Answer

1. What capability does the `fstream` data type provide that the `ifstream` and `ofstream` data types do not?
2. Which file access flag do you use to open a file when you want all output written to the end of the file's existing contents?
3. Assume that the file `data.txt` already exists, and the following statement executes. What happens to the file?

   ```
   fstream file("data.txt", ios::out);
   ```

4. How do you combine multiple file access flags when opening a file?
5. Should file stream objects be passed to functions by value or by reference? Why?
6. Under what circumstances is a file stream object's `ios::hardfail` bit set? What member function reports the state of this bit?
7. Under what circumstances is a file stream object's `ios::eofbit` bit set? What member function reports the state of this bit?
8. Under what circumstances is a file stream object's `ios::badbit` bit set? What member function reports the state of this bit?
9. How do you read the contents of a text file that contains whitespace characters as part of its data?
10. What arguments do you pass to a file stream object's `write` member function?
11. What arguments do you pass to a file stream object's `read` member function?
12. What type cast do you use to convert a pointer from one type to another?
13. What is the difference between the `seekg` and `seekp` member functions?
14. How do you get the byte number of a file's current read position? How do you get the byte number of a file's current write position?
15. If a program has read to the end of a file, what must you do before using either the `seekg` or `seekp` member functions?
16. How do you determine the number of bytes that a file contains?
17. How do you rewind a sequential-access file?

Fill-in-the-Blank

18. The _____ file stream data type is for output files, input files, or files that perform both input and output.
19. If a file fails to open, the file stream object will be set to _____.
20. The same formatting techniques used with _____ may also be used when writing data to a file.
21. The _____ function reads a line of text from a file.
22. The _____ member function reads a single character from a file.
23. The _____ member function writes a single character to a file.
24. _____ files contain data that is unformatted and not necessarily stored as ASCII text.
25. _____ files contain data formatted as _____.
26. A(n) _____ is a complete set of data about a single item and is made up of _____.
27. In C++, _____ provide a convenient way to organize data into fields and records.
28. The _____ member function writes "raw" binary data to a file.
29. The _____ member function reads "raw" binary data from a file.
30. The _____ operator is necessary if you pass anything other than a pointer-to-char as the first argument of the two functions mentioned in questions 26 and 27.
31. In _____ file access, the contents of the file are read in the order they appear in the file, from the file's start to its end.
32. In _____ file access, the contents of a file may be read in any order.
33. The _____ member function moves a file's read position to a specified byte in the file.
34. The _____ member function moves a file's write position to a specified byte in the file.
35. The _____ member function returns a file's current read position.
36. The _____ member function returns a file's current write position.
37. The _____ mode flag causes an offset to be calculated from the beginning of a file.
38. The _____ mode flag causes an offset to be calculated from the end of a file.
39. The _____ mode flag causes an offset to be calculated from the current position in the file.
40. A negative offset causes the file's read or write position to be moved _____ in the file from the position specified by the mode.

Algorithm Workbench

41. Write a statement that defines a file stream object named `places`. The object will be used for both output and input.

42. Write two statements that use a file stream object named `people` to open a file named `people.dat`. (Show how to open the file with a member function and at the definition of the file stream object.) The file should be opened for output.
43. Write two statements that use a file stream object named `pets` to open a file named `pets.dat`. (Show how to open the file with a member function and at the definition of the file stream object.) The file should be opened for input.
44. Write two statements that use a file stream object named `places` to open a file named `places.dat`. (Show how to open the file with a member function and at the definition of the file stream object.) The file should be opened for both input and output.
45. Write a program segment that defines a file stream object named `employees`. The file should be opened for both input and output (in binary mode). If the file fails to open, the program segment should display an error message.
46. Write code that opens the file `data.txt` for both input and output, but first determines if the file exists. If the file does not exist, the code should create it, then open it for both input and output.
47. Write code that determines the number of bytes contained in the file associated with the file stream object `dataFile`.
48. The `infoFile` file stream object is used to sequentially access data. The program has already read to the end of the file. Write code that rewinds the file.

True or False

49. T F Different operating systems have different rules for naming files.
50. T F `fstream` objects are only capable of performing file output operations.
51. T F `ofstream` objects, by default, delete the contents of a file if it already exists when opened.
52. T F `ifstream` objects, by default, create a file if it doesn't exist when opened.
53. T F Several file access flags may be joined by using the | operator.
54. T F A file may be opened in the definition of the file stream object.
55. T F If a file is opened in the definition of the file stream object, no mode flags may be specified.
56. T F A file stream object's `fail` member function may be used to determine if the file was successfully opened.
57. T F The same output formatting techniques used with `cout` may also be used with file stream objects.
58. T F The >> operator expects data to be delimited by whitespace characters.
59. T F The `getline` member function can be used to read text that contains whitespaces.
60. T F It is not possible to have more than one file open at once in a program.
61. T F Binary files contain unformatted data, not necessarily stored as text.
62. T F Binary is the default mode in which files are opened.
63. T F The `tellp` member function tells a file stream object which byte to move its write position to.
64. T F It is possible to open a file for both input and output.

Find the Error

Each of the following programs or program segments has errors. Find as many as you can.

65.
```
fstream file(ios::in | ios::out);
file.open("info.dat");
if (!file)
{
   cout << "Could not open file.\n";
}
```

66.
```
ofstream file;
file.open("info.dat", ios::in);
if (file)
{
   cout << "Could not open file.\n";
}
```

67.
```
fstream file("info.dat");
if (!file)
{
   cout << "Could not open file.\n";
}
```

68.
```
fstream dataFile("info.dat", ios:in | ios:binary);
int x = 5;
dataFile << x;
```

69.
```
fstream dataFile("info.dat", ios:in);
char stuff[81];
dataFile.get(stuff);
```

70.
```
fstream dataFile("info.dat", ios:in);
char stuff[81] = "abcdefghijklmnopqrstuvwxyz";
dataFile.put(stuff);
```

71.
```
fstream dataFile("info.dat", ios:out);
struct Date
{
   int month;
   int day;
   int year;
};
Date dt = { 4, 2, 98 };
dataFile.write(&dt, sizeof(int));
```

72.
```
fstream inFile("info.dat", ios:in);
int x;
inFile.seekp(5);
inFile >> x;
```

Programming Challenges

myprogramminglab

Visit www.myprogramminglab.com to complete many of these Programming Challenges online and get instant feedback.

1. **File Head Program**

   Write a program that asks the user for the name of a file. The program should display the first 10 lines of the file on the screen (the "head" of the file). If the file has fewer

than 10 lines, the entire file should be displayed, with a message indicating the entire file has been displayed.

NOTE: Using an editor, you should create a simple text file that can be used to test this program.

2. **File Display Program**

   Write a program that asks the user for the name of a file. The program should display the contents of the file on the screen. If the file's contents won't fit on a single screen, the program should display 24 lines of output at a time, and then pause. Each time the program pauses, it should wait for the user to strike a key before the next 24 lines are displayed.

NOTE: Using an editor, you should create a simple text file that can be used to test this program.

3. **Punch Line**

   Write a program that reads and prints a joke and its punch line from two different files. The first file contains a joke, but not its punch line. The second file has the punch line as its last line, preceded by "garbage." The `main` function of your program should open the two files and then call two functions, passing each one the file it needs. The first function should read and display each line in the file it is passed (the joke file). The second function should display only the last line of the file it is passed (the punch line file). It should find this line by seeking to the end of the file and then backing up to the beginning of the last line. Data to test your program can be found in the `joke.txt` and `punchline.txt` files.

4. **Tail Program**

   Write a program that asks the user for the name of a file. The program should display the last 10 lines of the file on the screen (the "tail" of the file). If the file has fewer than 10 lines, the entire file should be displayed, with a message indicating the entire file has been displayed.

NOTE: Using an editor, you should create a simple text file that can be used to test this program.

5. **Line Numbers**

   (This assignment could be done as a modification of the program in Programming Challenge 2.) Write a program that asks the user for the name of a file. The program should display the contents of the file on the screen. Each line of screen output should be preceded with a line number, followed by a colon. The line numbering should start at 1. Here is an example:

```
1: George Rolland
2: 127 Academy Street
3: Brasstown, NC  28706
```

If the file's contents won't fit on a single screen, the program should display 24 lines of output at a time, and then pause. Each time the program pauses, it should wait for the user to strike a key before the next 24 lines are displayed.

NOTE: Using an editor, you should create a simple text file that can be used to test this program.

6. **String Search**

   Write a program that asks the user for a file name and a string to search for. The program should search the file for every occurrence of a specified string. When the string is found, the line that contains it should be displayed. After all the occurrences have been located, the program should report the number of times the string appeared in the file.

NOTE: Using an editor, you should create a simple text file that can be used to test this program.

7. **Sentence Filter**

   Write a program that asks the user for two file names. The first file will be opened for input and the second file will be opened for output. (It will be assumed that the first file contains sentences that end with a period.) The program will read the contents of the first file and change all the letters to lowercase except the first letter of each sentence, which should be made uppercase. The revised contents should be stored in the second file.

NOTE: Using an editor, you should create a simple text file that can be used to test this program.

8. **Array/File Functions**

   Write a function named `arrayToFile`. The function should accept three arguments: the name of a file, a pointer to an `int` array, and the size of the array. The function should open the specified file in binary mode, write the contents of the array to the file, and then close the file.

   Write another function named `fileToArray`. This function should accept three arguments: the name of a file, a pointer to an `int` array, and the size of the array. The function should open the specified file in binary mode, read its contents into the array, and then close the file.

   Write a complete program that demonstrates these functions by using the `arrayToFile` function to write an array to a file, and then using the `fileToArray` function to read the data from the same file. After the data are read from the file into the array, display the array's contents on the screen.

9. File Encryption Filter

VideoNote
Solving the File Encryption Filter Problem

File encryption is the science of writing the contents of a file in a secret code. Your encryption program should work like a filter, reading the contents of one file, modifying the data into a code, and then writing the coded contents out to a second file. The second file will be a version of the first file, but written in a secret code.

Although there are complex encryption techniques, you should come up with a simple one of your own. For example, you could read the first file one character at a time, and add 10 to the ASCII code of each character before it is written to the second file.

10. File Decryption Filter

Write a program that decrypts the file produced by the program in Programming Challenge 9. The decryption program should read the contents of the coded file, restore the data to its original state, and write it to another file.

11. Corporate Sales Data Output

Write a program that uses a structure to store the following data on a company division:

Division Name (such as East, West, North, or South)
Quarter (1, 2, 3, or 4)
Quarterly Sales

The user should be asked for the four quarters' sales figures for the East, West, North, and South divisions. The data for each quarter for each division should be written to a file.

Input Validation: Do not accept negative numbers for any sales figures.

12. Corporate Sales Data Input

Write a program that reads the data in the file created by the program in Programming Challenge 11. The program should calculate and display the following figures:

- Total corporate sales for each quarter
- Total yearly sales for each division
- Total yearly corporate sales
- Average quarterly sales for the divisions
- The highest and lowest quarters for the corporation

13. Inventory Program

Write a program that uses a structure to store the following inventory data in a file:

Item Description
Quantity on Hand
Wholesale Cost
Retail Cost
Date Added to Inventory

The program should have a menu that allows the user to perform the following tasks:

- Add new records to the file.
- Display any record in the file.
- Change any record in the file.

Input Validation: The program should not accept quantities, or wholesale or retail costs, less than 0. The program should not accept dates that the programmer determines are unreasonable.

14. Inventory Screen Report

Write a program that reads the data in the file created by the program in Programming Challenge 13. The program should calculate and display the following data:

- The total wholesale value of the inventory
- The total retail value of the inventory
- The total quantity of all items in the inventory

15. Average Number of Words

If you have downloaded this book's source code from the companion Web site, you will find a file named `text.txt` in the Chapter 12 folder. (The companion Web site is at www.pearsonhighered.com/gaddis.) The text that is in the file is stored as one sentence per line. Write a program that reads the file's contents and calculates the average number of words per sentence.

Group Project

16. Customer Accounts

This program should be designed and written by a team of students. Here are some suggestions:

- One student should design function `main`, which will call other program functions. The remainder of the functions will be designed by other members of the team.
- The requirements of the program should be analyzed so each student is given about the same workload.

Write a program that uses a structure to store the following data about a customer account:

Name
Address
City, State, and ZIP
Telephone Number
Account Balance
Date of Last Payment

The structure should be used to store customer account records in a file. The program should have a menu that lets the user perform the following operations:

- Enter new records into the file.
- Search for a particular customer's record and display it.
- Search for a particular customer's record and delete it.
- Search for a particular customer's record and change it.
- Display the contents of the entire file.

Input Validation: When the data for a new account is entered, be sure the user enters data for all the fields. No negative account balances should be entered.