Review Questions and Exercises

Short Answer

1. What is the difference between a size declarator and a subscript?
2. Look at the following array definition.

   ```
   int values[10];
   ```

   How many elements does the array have?

   What is the subscript of the first element in the array?

   What is the subscript of the last element in the array?

   Assuming that an int uses four bytes of memory, how much memory does the array use?
3. Why should a function that accepts an array as an argument, and processes that array, also accept an argument specifying the array's size?
4. Consider the following array definition:

   ```
   int values[5] = { 4, 7, 6, 8, 2 };
   ```

   What does each of the following statements display?

   ```
   cout << values[4] << endl; _____
   cout << (values[2] + values[3]) << endl; _____
   cout << ++values[1] << endl; _____
   ```
5. How do you define an array without providing a size declarator?
6. Look at the following array definition.

   ```
   int numbers[5] = { 1, 2, 3 };
   ```

   What value is stored in `numbers[2]`?

   What value is stored in `numbers[4]`?
7. Assuming that `array1` and `array2` are both arrays, why is it not possible to assign the contents of `array2` to `array1` with the following statement?

   ```
   array1 = array2;
   ```
8. Assuming that `numbers` is an array of `doubles`, will the following statement display the contents of the array?

   ```
   cout << numbers << endl;
   ```
9. Is an array passed to a function by value or by reference?
10. When you pass an array name as an argument to a function, what is actually being passed?
11. How do you establish a parallel relationship between two or more arrays?
12. Look at the following array definition.

    ```
    double sales[8][10];
    ```

    How many rows does the array have?

    How many columns does the array have?

How many elements does the array have?

Write a statement that stores a number in the last column of the last row in the array.

13. When writing a function that accepts a two-dimensional array as an argument, which size declarator must you provide in the parameter for the array?
14. What advantages does a `vector` offer over an array?

Fill-in-the-Blank

15. The _____ indicates the number of elements, or values, an array can hold.
16. The size declarator must be a(n) _____ with a value greater than _____.
17. Each element of an array is accessed and indexed by a number known as a(n) _____.
18. Subscript numbering in C++ always starts at _____.
19. The number inside the brackets of an array definition is the _____, but the number inside an array's brackets in an assignment statement, or any other statement that works with the contents of the array, is the _____.
20. C++ has no array _____ checking, which means you can inadvertently store data past the end of an array.
21. Starting values for an array may be specified with a(n) _____ list.
22. If an array is partially initialized, the uninitialized elements will be set to _____.
23. If the size declarator of an array definition is omitted, C++ counts the number of items in the _____ to determine how large the array should be.
24. By using the same _____ for multiple arrays, you can build relationships between the data stored in the arrays.
25. You cannot use the _____ operator to copy data from one array to another in a single statement.
26. Any time the name of an array is used without brackets and a subscript, it is seen as _____.
27. To pass an array to a function, pass the _____ of the array.
28. A(n) _____ array is like several arrays of the same type put together.
29. It's best to think of a two-dimensional array as having _____ and _____.
30. To define a two-dimensional array, _____ size declarators are required.
31. When initializing a two-dimensional array, it helps to enclose each row's initialization list in _____.
32. When a two-dimensional array is passed to a function the _____ size must be specified.
33. The _____ is a collection of programmer-defined data types and algorithms that you may use in your programs
34. The two types of containers defined by the STL are _____ and _____.

35. The `vector` data type is a(n) _____ container.
36. To define a `vector` in your program, you must `#include` the _____ header file.
37. To store a value in a `vector` that does not have a starting size, or that is already full, use the _____ member function.
38. To determine the number of elements in a `vector`, use the _____ member function.
39. Use the _____ member function to remove the last element from a `vector`.
40. To completely clear the contents of a `vector`, use the _____ member function.

Algorithm Workbench

41. `names` is an integer array with 20 elements. Write a `for` loop that prints each element of the array.
42. The arrays `numberArray1` and `numberArray2` have 100 elements. Write code that copies the values in `numberArray1` to `numberArray2`.
43. In a program you need to store the identification numbers of 10 employees (as `int`s) and their weekly gross pay (as `double`s).
    A) Define two arrays that may be used in parallel to store the 10 employee identification numbers and gross pay amounts.
    B) Write a loop that uses these arrays to print each employee's identification number and weekly gross pay.
44. Define a two-dimensional array of integers named `grades`. It should have 30 rows and 10 columns.
45. In a program you need to store the populations of 12 countries.
    A) Define two arrays that may be used in parallel to store the names of the countries and their populations.
    B) Write a loop that uses these arrays to print each country's name and its population.
46. The following code totals the values in two arrays: `numberArray1` and `numberArray2`. Both arrays have 25 elements. Will the code print the correct sum of values for both arrays? Why or why not?

```
int total = 0;          // Accumulator
int count;              // Loop counter
// Calculate and display the total of the first array.
for (count = 0; count < 24; count++)
   total += numberArray1[count];
cout << "The total for numberArray1 is " << total << endl;
// Calculate and display the total of the second array.
for (count = 0; count < 24; count++)
   total += numberArray2[count];
cout << "The total for numberArray2 is " << total << endl;
```

47. Look at the following array definition.

```
int numberArray[9][11];
```

Write a statement that assigns 145 to the first column of the first row of this array.

Write a statement that assigns 18 to the last column of the last row of this array.

48. `values` is a two-dimensional array of `floats` with 10 rows and 20 columns. Write code that sums all the elements in the array and stores the sum in the variable `total`.
49. An application uses a two-dimensional array defined as follows.

```
int days[29][5];
```

Write code that sums each row in the array and displays the results.

Write code that sums each column in the array and displays the results.

True or False

50. T F An array's size declarator can be either a literal, a named constant, or a variable.
51. T F To calculate the amount of memory used by an array, multiply the number of elements by the number of bytes each element uses.
52. T F The individual elements of an array are accessed and indexed by unique numbers.
53. T F The first element in an array is accessed by the subscript 1.
54. T F The subscript of the last element in a single-dimensional array is one less than the total number of elements in the array.
55. T F The contents of an array element cannot be displayed with `cout`.
56. T F Subscript numbers may be stored in variables.
57. T F You can write programs that use invalid subscripts for an array.
58. T F Arrays cannot be initialized when they are defined. A loop or other means must be used.
59. T F The values in an initialization list are stored in the array in the order they appear in the list.
60. T F C++ allows you to partially initialize an array.
61. T F If an array is partially initialized, the uninitialized elements will contain "garbage."
62. T F If you leave an element uninitialized, you do not have to leave all the ones that follow it uninitialized.
63. T F If you leave out the size declarator of an array definition, you do not have to include an initialization list.
64. T F The uninitialized elements of a `string` array will automatically be set to the value `"0"`.
65. T F You cannot use the assignment operator to copy one array's contents to another in a single statement.
66. T F When an array name is used without brackets and a subscript, it is seen as the value of the first element in the array.
67. T F To pass an array to a function, pass the name of the array.
68. T F When defining a parameter variable to hold a single-dimensional array argument, you do not have to include the size declarator.
69. T F When an array is passed to a function, the function has access to the original array.
70. T F A two-dimensional array is like several identical arrays put together.

71. T F It's best to think of two-dimensional arrays as having rows and columns.
72. T F The first size declarator (in the declaration of a two-dimensional array) represents the number of columns. The second size definition represents the number of rows.
73. T F Two-dimensional arrays may be passed to functions, but the row size must be specified in the definition of the parameter variable.
74. T F C++ allows you to create arrays with three or more dimensions.
75. T F A `vector` is an associative container.
76. T F To use a `vector`, you must include the `vector` header file.
77. T F `vectors` can report the number of elements they contain.
78. T F You can use the `[]` operator to insert a value into a `vector` that has no elements.
79. T F If you add a value to a `vector` that is already full, the `vector` will automatically increase its size to accommodate the new value.

Find the Error

Each of the following definitions and program segments has errors. Locate as many as you can.

80.
```
int size;
double values[size];
```

81.
```
int collection[-20];
```

82.
```
int table[10];
for (int x = 0; x < 20; x++)
{
   cout << "Enter the next value: ";
   cin >> table[x];
}
```

83.
```
int hours[3] = 8, 12, 16;
```

84.
```
int numbers[8] = {1, 2, , 4, , 5};
```

85.
```
float ratings[];
```

86.
```
char greeting[] = {'H', 'e', 'l', 'l', 'o'};
cout << greeting;
```

87.
```
int array1[4], array2[4] = {3, 6, 9, 12};
array1 = array2;
```

88.
```
void showValues(int nums)
{
   for (int count = 0; count < 8; count++)
      cout << nums[count];
}
```

89.
```
void showValues(int nums[4][])
{
   for (rows = 0; rows < 4; rows++)
      for (cols = 0; cols < 5; cols++)
         cout << nums[rows][cols];
}
```

Programming Challenges

mylab programming

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

1. **Largest/Smallest Array Values**

   Write a program that lets the user enter 10 values into an array. The program should then display the largest and smallest values stored in the array.

2. **Rainfall Statistics**

   Write a program that lets the user enter the total rainfall for each of 12 months into an array of `doubles`. The program should calculate and display the total rainfall for the year, the average monthly rainfall, and the months with the highest and lowest amounts.

   Input Validation: Do not accept negative numbers for monthly rainfall figures.

3. **Chips and Salsa**

   VideoNote
   Solving the Chips and Salsa Problem

   Write a program that lets a maker of chips and salsa keep track of sales for five different types of salsa: mild, medium, sweet, hot, and zesty. The program should use two parallel 5-element arrays: an array of strings that holds the five salsa names and an array of integers that holds the number of jars sold during the past month for each salsa type. The salsa names should be stored using an initialization list at the time the name array is created. The program should prompt the user to enter the number of jars sold for each type. Once this sales data has been entered, the program should produce a report that displays sales for each salsa type, total sales, and the names of the highest selling and lowest selling products.

   Input Validation: Do not accept negative values for number of jars sold.

4. **Monkey Business**

   A local zoo wants to keep track of how many pounds of food each of its three monkeys eats each day during a typical week. Write a program that stores this information in a two-dimensional 3×7 array, where each row represents a different monkey and each column represents a different day of the week. The program should first have the user input the data for each monkey. Then it should create a report that includes the following information:

   - Average amount of food eaten per day by the whole family of monkeys.
   - The least amount of food eaten during the week by any one monkey.
   - The greatest amount of food eaten during the week by any one monkey.

   Input Validation: Do not accept negative numbers for pounds of food eaten.

5. **Rain or Shine**

   An amateur meteorologist wants to keep track of weather conditions during the past year's three-month summer season and has designated each day as either rainy ('R'), cloudy ('C'), or sunny ('S'). Write a program that stores this information in a 3×30 array of characters, where the row indicates the month (0 = June, 1 = July, 2 = August) and the column indicates the day of the month. Note that data are not being collected for the 31st of any month. The program should begin by reading the weather data in from a file. Then it should create a report that displays, for each month and for the whole three-month period, how many days were rainy, how many were cloudy, and how many were sunny. It should also report which of the three months had the largest number of rainy days. Data for the program can be found in the `RainOrShine.dat` file.

6. Number Analysis Program

Write a program that asks the user for a file name. Assume the file contains a series of numbers, each written on a separate line. The program should read the contents of the file into an array and then display the following data:

- The lowest number in the array
- The highest number in the array
- The total of the numbers in the array
- The average of the numbers in the array

If you have downloaded this book's source code from the companion Web site, you will find a file named numbers.txt in the Chapter 06 folder. You can use the file to test the program. (The companion Web site is at www.pearsonhighered.com/gaddis.)

7. Quarterly Sales Statistics

Write a program that lets the user enter four quarterly sales figures for six divisions of a company. The figures should be stored in a two-dimensional array. Once the figures are entered, the program should display the following data for each quarter:

- A list of the sales figures by division
- Each division's increase or decrease from the previous quarter (This will not be displayed for the first quarter.)
- The total sales for the quarter
- The company's increase or decrease from the previous quarter (This will not be displayed for the first quarter.)
- The average sales for all divisions that quarter
- The division with the highest sales for that quarter

The program should be modular, with functions that calculate the statistics above.

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

8. Payroll

Write a program that uses the following arrays:

- `empId`: an array of seven long integers to hold employee identification numbers. The array should be initialized with the following numbers:

```
5658845   4520125   7895122   8777541
8451277   1302850   7580489
```

- `hours`: an array of seven integers to hold the number of hours worked by each employee
- `payRate`: an array of seven `double`s to hold each employee's hourly pay rate
- `wages`: an array of seven `double`s to hold each employee's gross wages

The program should relate the data in each array through the subscripts. For example, the number in element 0 of the `hours` array should be the number of hours worked by the employee whose identification number is stored in element 0 of the `empId` array. That same employee's pay rate should be stored in element 0 of the `payRate` array.

The program should display each employee number and ask the user to enter that employee's hours and pay rate. It should then calculate the gross wages for that employee (hours times pay rate) and store them in the `wages` array. After the data has

been entered for all the employees, the program should display each employee's identification number and gross wages.

Input Validation: Do not accept negative values for hours or numbers less than 6.00 for pay rate.

9. Driver's License Exam

The local Driver's License Office has asked you to write a program that grades the written portion of the driver's license exam. The exam has 20 multiple choice questions. Here are the correct answers:

1. B	6. A	11. B	16. C
2. D	7. B	12. C	17. C
3. A	8. A	13. D	18. B
4. A	9. C	14. A	19. D
5. C	10. D	15. D	20. A

Your program should store the correct answers shown above in an array. It should ask the user to enter the student's answers for each of the 20 questions, and the answers should be stored in another array. After the student's answers have been entered, the program should display a message indicating whether the student passed or failed the exam. (A student must correctly answer 15 of the 20 questions to pass the exam.) It should then display the total number of correctly answered questions, the total number of incorrectly answered questions, and a list showing the question numbers of the incorrectly answered questions.

Input Validation: Only accept the letters A, B, C, or D as answers.

10. Exam Grader

One of your professors has asked you to write a program to grade her final exams, which consist of only 20 multiple-choice questions. Each question has one of four possible answers: A, B, C, or D. The file CorrectAnswers.txt contains the correct answers for all of the questions, with each answer written on a separate line. The first line contains the answer to the first question, the second line contains the answer to the second question, and so forth. (Download the book's source code from the companion Web site at www.pearsonhighered.com/gaddis. You will find the file in the Chapter 07 folder.)

Write a program that reads the contents of the CorrectAnswers.txt file into a `char` array, and then reads the contents of another file, containing a student's answers, into a second `char` array. (You can use the file StudentAnswers.txt for testing purposes. This file is also in the Chapter 07 source code folder, available on the book's companion Web site.) The program should determine the number of questions that the student missed, and then display the following:

- A list of the questions missed by the student, showing the correct answer and the incorrect answer provided by the student for each missed question
- The total number of questions missed
- The percentage of questions answered correctly. This can be calculated as

  Correctly Answered Questions ÷ Total Number of Questions

- If the percentage of correctly answered questions is 70% or greater, the program should indicate that the student passed the exam. Otherwise, it should indicate that the student failed the exam.

11. Grade Book

A teacher has five students who have taken four tests. The teacher uses the following grading scale to assign a letter grade to a student, based on the average of his or her four test scores.

Test Score	Letter Grade
90–100	A
80–89	B
70–79	C
60–69	D
0–59	F

Write a program that uses an array of `string` objects to hold the five student names, an array of five characters to hold the five students' letter grades, and five arrays of four `doubles` to hold each student's set of test scores.

The program should allow the user to enter each student's name and his or her four test scores. It should then calculate and display each student's average test score and a letter grade based on the average.

Input Validation: Do not accept test scores less than 0 or greater than 100.

12. Grade Book Modification

Modify the grade book application in Programming Challenge 13 so it drops each student's lowest score when determining the test score averages and letter grades.

13. Lottery Application

Write a program that simulates a lottery. The program should have an array of five integers named `lottery`, and should generate a random number in the range of 0 through 9 for each element in the array. The user should enter five digits which should be stored in an integer array named `user`. The program is to compare the corresponding elements in the two arrays and keep a count of the digits that match. For example, the following shows the `lottery` array and the `user` array with sample numbers stored in each. There are two matching digits (elements 2 and 4).

lottery array:

7	4	9	1	3

user array:

4	2	9	7	3

The program should display the random numbers stored in the `lottery` array and the number of matching digits. If all of the digits match, display a message proclaiming the user as a grand prize winner.

14. `vector` Modification

Modify the National Commerce Bank case study presented in Program 7-21 so `pin1`, `pin2`, and `pin3` are `vectors` instead of arrays. You must also modify the `testPIN` function to accept a `vector` instead of an array.

15. Tic-Tac-Toe Game

Write a program that allows two players to play a game of tic-tac-toe. Use a two-dimensional `char` array with three rows and three columns as the game board. Each element of the array should be initialized with an asterisk (*). The program should run a loop that

- Displays the contents of the board array
- Allows player 1 to select a location on the board for an X. The program should ask the user to enter the row and column number.
- Allows player 2 to select a location on the board for an O. The program should ask the user to enter the row and column number.
- Determines whether a player has won, or a tie has occurred. If a player has won, the program should declare that player the winner and end. If a tie has occurred, the program should say so and end.

Player 1 wins when there are three Xs in a row on the game board. The Xs can appear in a row, in a column, or diagonally across the board. A tie occurs when all of the locations on the board are full, but there is no winner.

16. 2D Array Operations

Write a program that creates a two-dimensional array initialized with test data. Use any data type you wish. The program should have the following functions:

- **`getTotal`**. This function should accept a two-dimensional array as its argument and return the total of all the values in the array.
- **`getAverage`**. This function should accept a two-dimensional array as its argument and return the average of all the values in the array.
- **`getRowTotal`**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the total of the values in the specified row.
- **`getColumnTotal`**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a column in the array. The function should return the total of the values in the specified column.
- **`getHighestInRow`**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the highest value in the specified row of the array.
- **`getLowestInRow`**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the lowest value in the specified row of the array.

Demonstrate each of the functions in this program.

Group Project

17. Theater Seating

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 the other functions in the program. 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 work load.
- The parameters and return types of each function should be decided in advance.
- The program can be implemented as a multi-file program, or all the functions can be cut and pasted into the main file.

Here is the assignment: Write a program that can be used by a small theater to sell tickets for performances. The theater's auditorium has 15 rows of seats, with 30 seats in each row. The program should display a screen that shows which seats are available and which are taken. For example, the following screen shows a chart depicting each seat in the theater. Seats that are taken are represented by an * symbol, and seats that are available are represented by a # symbol:

```
                Seats
          123456789012345678901234567890
Row 1     **********##########**********
Row 2     ##############################
Row 3     **************##############***
Row 4     ************************#######
Row 5     ********#########################
Row 6     ##############################
Row 7     ################################
Row 8     ************#####################
Row 9     ##########*************************
Row 10    ################################
Row 11    **********########################**
Row 12    ##############################*
Row 13    ##############################
Row 14    ################################
Row 15    ################################
```

Here is a list of tasks this program must perform:

- When the program begins, it should ask the user to enter the seat prices for each row. The prices can be stored in a separate array. (Alternatively, the prices may be read from a file.)
- Once the prices are entered, the program should display a seating chart similar to the one shown above. The user may enter the row and seat numbers for tickets being sold. Every time a ticket or group of tickets is purchased, the program should display the total ticket prices and update the seating chart.
- The program should keep a total of all ticket sales. The user should be given an option of viewing this amount.

- The program should also give the user an option to see a list of how many seats have been sold, how many seats are available in each row, and how many seats are available in the entire auditorium.

Input Validation: When tickets are being sold, do not accept row or seat numbers that do not exist. When someone requests a particular seat, the program should make sure that seat is available before it is sold.