7

Arrays and Vectors

Now go, write it
before them in a table,
and note it in a book.
—Isaiah 30:8

Begin at the beginning, …
and go on till you come to
the end: then stop.
—Lewis Carroll

To go beyond is as
wrong as to fall short.
—Confucius

OBJECTIVES

In this chapter you'll learn:

- To use the array data structure to represent a set of related data items.
- To use arrays to store, sort and search lists and tables of values.
- To declare arrays, initialize arrays and refer to the individual elements of arrays.
- To pass arrays to functions.
- Basic searching and sorting techniques.
- To declare and manipulate multidimensional arrays.
- To use C++ Standard Library class template `vector`.

Assignment Checklist

Name: ____________________________ Date: ____________________________

Section: ____________________________

Exercises	Assigned: Circle assignments	Date Due
Prelab Activities Matching	YES NO	
Fill in the Blank	12, 13, 14, 15, 16, 17, 18	
Short Answer	19, 20, 21, 22, 23	
Programming Output	24, 25, 26, 27, 28, 29	
Correct the Code	30, 31, 32, 33, 34, 35	
Lab Exercises Lab Exercise 1—`Rolling Dice`	YES NO	
Follow-Up Questions and Activities	1, 2, 3, 4	
Lab Exercise 2—`Bubble Sort`	YES NO	
Follow-Up Question and Activity	1	
Lab Exercise 3—`Salespeople`	YES NO	
Follow-Up Questions and Activities	1, 2, 3	
Debugging	YES NO	
Labs Provided by Instructor 1.		
2.		
3.		
Postlab Activities Coding Exercises	1, 2, 3, 4, 5, 6	
Programming Challenges	1, 2, 3, 4	

Prelab Activities

Matching

Name: ______________________ **Date:** ______________________

Section: ______________________

After reading Chapter 7 of *C++ How to Program, Seventh Edition*, answer the given questions. These questions are intended to test and reinforce your understanding of key concepts and may be done either before the lab or during the lab.

For each term in the column on the left, write the corresponding letter for the description that best matches it from the column on the right.

	Term	Description
___	1. Subscript	a) Qualifier that prevents modification of a variable’s value.
___	2. Zeroth element	b) Refers to a particular location or element in an array.
___	3. Scalar quantities	c) Single pieces of data such as individual array elements.
___	4. Insertion sort	d) String termination character.
___	5. Sorting	e) Sequence of characters enclosed in double quotes.
___	6. Search key	f) Value that the program attempts to find by searching.
___	7. Two-dimensional arrays	g) First element in an array.
___	8. Null character	h) Discrepancy between “i^{th} element of the array” and “array element *i*.”
___	9. `const`	i) An algorithm for ordering the elements in an array.
___	10. Off-by-one error	j) Arrays that require two subscripts to identify an array element.
___	11. String	k) Placing data in ascending or descending order.

Prelab Activities

Name:

Fill in the Blank

Name: ______________________________ **Date:** ______________________________

Section: ______________________________

Fill in the blank for each of the following statements:

12. To pass an array to a function, the __________ of the array is passed.

13. To pass one row of a double-subscripted array to a function that receives a single-subscripted array, pass the name of the array followed by the __________.

14. An array can be initialized in its declaration may using a(n) __________.

15. C++ stores lists of values in __________.

16. A(n) __________ variable must be initialized in its declaration.

17. Arrays are passed to functions by __________.

18. A(n) __________ may be an integer or an integer expression and identifies a particular array element.

Prelab Activities

Name:

Short Answer

Name: ______________________ **Date:** ______________________

Section: ______________________

In the space provided, answer each of the given questions. Your answers should be as concise as possible; aim for two or three sentences.

19. What is an "off-by-one error?" Give an example.

20. Describe how a linear search works. On average, how many comparisons must a linear search perform?

21. What is the `const` qualifier? What happens when the programmer tries to modify the contents of an array that is passed to a function that receives the array as a `const` parameter?

22. How is an insertion sort implemented? Why is insertion sort inefficient for sorting large arrays?

Prelab Activities

Name:

Short Answer

23. Describe how multidimensional arrays might represent a table in which information is arranged in rows and columns.

Prelab Activities

Name:

Programming Output

Name: ______________________________ **Date:** ______________________________

Section: ______________________________

For each of the given program segments, read the code and write the output in the space provided below each program. [*Note:* Do not execute these programs on a computer.]

24. What is output by the following program segment?

```
int i;
int values[ 10 ] = { 4, 1, 1, 3, 4, 9, 9, 2, 1, 7 };

cout << "Element" << setw( 13 ) << "Value" << endl;

for ( i = 0; i < 10; i++ )
   cout << setw( 7 ) << i << setw( 13 ) << values[ i ] << endl;
```

Your answer:

25. What is output by the following code segment?

```
char string1[] = "How are you?";

cout << "string1 is: " << string1 << endl;

for ( int i = 0; string1[ i ] != '\0'; i++ )
   cout << string1[ i ] << "_";
```

Your answer:

Prelab Activities

Name:

Programming Output

26. What is output by the following program?

```
#include <iostream>
using namespace std;

void mystery();

int main()
{
   cout << "First call to mystery:\n";
   mystery();

   cout << "\n\nSecond call to mystery:\n";
   mystery();
   cout << endl;
} // end main

// function mystery definition
void mystery()
{
   static int array1[ 3 ];
   int i;

   cout << "\nValues on entering mystery:\n";

   for ( i = 0; i < 3; i++ )
      cout << "array1[" << i << "] = " << array1[ i ] << "  ";

   cout << "\nValues on exiting mystery:\n";

   for ( i = 0; i < 3; i++ )
      cout << "array1[" << i << "] = "
           << ( array1[ i ] += 2 ) << "  ";

} // end function mystery
```

Your answer:

27. What is output by the following program? What algorithm does this program implement?

```
#include <iostream>
using namespace std;

int main()
{
```

Prelab Activities

Name:

Programming Output

```
   const int arraySize = 10;
   int a[ arraySize ] = { 2, 62, 4, 33, 10, 12, 89, 68, 45, 7 };
   int i;
   int insert;

   cout << "Data items in original order\n";

   for ( i = 0; i < arraySize; i++ )
      cout << setw( 4 ) << a[ i ];

   for ( int next = 1; next < arraySize; next++ )
   {
      insert = data[ next ];

      int moveItem = next;

      while ( ( moveItem > 0 ) && ( data[ moveItem - 1 ] < insert ) )
      {
         data[ moveItem ] = data[ moveItem - 1 ];
         moveItem--;
      }

      data[ moveItem ] = insert;
   }

   cout << "\nData items in new order\n";

   for ( i = 0; i < arraySize; i++ )
      cout << setw( 4 ) << a[ i ];

   cout << endl;
} // end main
```

Your answer:

28. What is output by the following program segment?

```
const int arraySize = 10;
int a[ arraySize ] = { 4, 3, 7, 1, 13, 6, 0, 2, 9, 5 };

for ( int i = 0; i < arraySize; i++ )
{

   for ( int j = 0; j < a [ i ]; j++ )
      cout << "*";

   cout << endl;
```

Prelab Activities

Name:

Programming Output

```
}
```

Your answer:

29. What is output by the following program?

```
#include <iostream>
using namespace std;

int main()
{
   int array[ 3 ][ 4 ] = { { 1, 2, 3, 4 }, { 2, 3, 4, 5 }, { 3, 4, 5, 6 } };

   for ( int i = 0; i < 3; i++ )
   {
      for ( int j = 0; j < 4; j++ )
      {
         cout << array[ i ][ j ] << " ";
      }

      cout << endl;
   }
} // end main
```

Your answer:

Prelab Activities

Name:

Correct the Code

Name: ____________________ **Date:** ____________________

Section: ____________________

For each of the given program segments, determine if there is an error in the code. If there is an error, specify whether it is a logic or compilation error, circle the error in the program, and write the corrected code in the space provided after each problem. If the code does not contain an error, write "no error." [*Note:* It is possible that a program segment may contain multiple errors.]

30. The following code should assign `8` to the fifth element in `array`:

```
array[ 5 ] = [ 8 ];
```

Your answer:

31. The `for` loop should initialize all array values to `-1`.

```
int array[ 10 ];

for ( int i = 0; i < 9; i++ )
   array[ i ] = -1;
```

Your answer:

32. Array `array` should contain all the integers from `0` through `10`, inclusive.

```
int array[ 10 ] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

Your answer:

33. The following code segment should declare two arrays containing five and six elements, respectively:

```
const int arraySize = 5;
int a[ arraySize ];

arraySize = 6;

int b[ arraySize ];
```

Your answer:

Prelab Activities

Name:

Correct the Code

34. The `for` loop that follows should print `array`'s values:

```
int array[ 10 ] = { 0 };

for ( int i = 0; i <= 10; i++ )
   cout << array[ i ];
```

Your answer:

35. The `for` loop that follows should print all of `array`'s values:

```
int array[ 3 ][ 3 ] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

for ( int i = 0; i < 3; i++ )

   for ( int i = 0; i < 3; i++ )
      cout << array[ i ][ i ];
```

Your answer:

Lab Exercises

Lab Exercise 1 — Rolling Dice

Name: ____________________ Date: ____________________

Section: ____________________

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 7.2)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 7 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Using `rand` to generate random numbers and using `srand` to seed the random-number generator.
- Declaring, initializing and referencing arrays.

The follow-up questions and activities also will give you practice:

- Remembering that arrays begin with subscript 0 and recognizing off-by-one errors.
- Preventing array out-of-bounds errors.
- Using two-dimensional arrays.

Description of the Problem

Write a program that simulates the rolling of two dice. The program should call `rand` to roll the first die, and should call `rand` again to roll the second die. The sum of the two values should then be calculated. [*Note:* Each die has an integer value from 1 to 6, so the sum of the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 being the least frequent sums.] Figure L 7.1 shows the 36 possible combinations of the two dice. Your program should roll the two dice 36,000 times. Use a one-dimensional array to tally the numbers of times each sum appears. Print the results in a tabular format. Also, determine if the totals are reasonable (i.e., there are six ways to roll a 7, so approximately one sixth of all the rolls should be 7).

Lab Exercises

Name:

Lab Exercise 1 — Rolling Dice

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Fig. L 7.1 | 36 possible outcomes of rolling two dice.

Sample Output

```
Sum      Total   Expected    Actual
  2      1000     2.778%     2.778%
  3      1958     5.556%     5.439%
  4      3048     8.333%     8.467%
  5      3979    11.111%    11.053%
  6      5007    13.889%    13.908%
  7      6087    16.667%    16.908%
  8      4996    13.889%    13.878%
  9      3971    11.111%    11.031%
 10      2996     8.333%     8.322%
 11      2008     5.556%     5.578%
 12       950     2.778%     2.639%
```

Template

```
// Lab 1: dice.cpp
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
   const long ROLLS = 36000;
   const int SIZE = 13;

   // array exepected contains counts for the expected
   // number of times each sum occurs in 36 rolls of the dice
   /* Write the declaration of array exprected here. Assign an
      initializer list containing the expected values here. Use
      SIZE for the number of elements */
   int x; // first die
```

Fig. L 7.2 | dice.cpp (Part 1 of 2)

Lab Exercises

Name:

Lab Exercise 1 — Rolling Dice

```
   int y; // second die
   /* Write declaration for array sum here. Initialize all
      elements to zero. Use SIZE for the number of elements */

   srand( time( 0 ) );

   // roll dice 36,000 times
   /* Write a for statement that iterates ROLLS times. Randomly
      generate values for x (i.e., die1) and y (i,e, die2)
      and increment the appropriate counter in array sum that
      corresponds to the sum of x and y */

   cout << setw( 10 ) << "Sum" << setw( 10 ) << "Total" << setw( 10 )
      << "Expected" << setw( 10 ) << "Actual\n" << fixed << showpoint;


   // display results of rolling dice
   for ( int j = 2; j < SIZE; j++ )
      cout << setw( 10 ) << j << setw( 10 ) << sum[ j ]
         << setprecision( 3 ) << setw( 9 )
         << 100.0 * expected[ j ] / 36 << "%" << setprecision( 3 )
         << setw( 9 ) << 100.0 * sum[ j ] / 36000 << "%\n";
} // end main
```

Fig. L 7.2 | dice.cpp. (Part 2 of 2.)

Problem-Solving Tips

1. Remember that array subscripts always begin with zero. This is also true for each dimension of a multiple-subscripted array (which this lab does not use).
2. The actual percentage is the likelihood, based on the results of your program, that a dice roll produced a certain result. In other words, if you roll the dice 36,000 times the actual percentage will be the *(number of times a result occurred / 36000) * 100*.
3. The expected percentage is the statistical probability that a dice roll will produce a certain result. This can be calculated from the diagram "36 possible outcomes of rolling two dice," shown in the problem description. For example, there is only one combination that will produce the sum of 2 and there are 36 total combinations that occur with equal likelihood. Therefore, the expected percentage of rolling a 2 is 1/36 or 2.778%.

Lab Exercises

Name:

Lab Exercise 1 — Rolling Dice

Follow-Up Questions and Activities

1. Why is the variable SIZE initialized to 13 when there are only 11 possible die-roll outcomes?

2. What happens if the < operator on line 47 of the program template is changed to <=?

3. What happens if the elements of array sum are not initialized to zero? Try running the program without initializing the array. Show your results.

4. Modify the program to use a two-dimensional array similar to the diagram in Figure L 7.1. Now, rather than counting the number of times each sum appears, increment the correct cell in the array. Print this array with the number of times each dice combination occurred. A sample output may look like the following:

```
          1         2         3         4         5         6
1      1011       971      1027      1025       971      1015
2      1013       968       990       968      1081       993
3       993      1014       983       973      1019       977
4       980      1004       974      1022       946      1046
5      1003      1021      1019       979      1004      1056
6      1026      1015       931       989      1014       979
```

Lab Exercises

Name:

Lab Exercise 2 — Bubble Sort

Name: ______________________________ **Date:** ______________________________

Section: ______________________________

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 7.3–Fig. L 7.4)
5. Problem-Solving Tips
6. Follow-Up Question and Activity

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up question. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 7 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Sorting data using the bubble sort algorithm.

The follow-up question and activity also will give you practice:

- Optimizing a program to be more efficient.

Description of the Problem

In the bubble sort algorithm, smaller values gradually "bubble" their way upward to the top of the array like air bubbles rising in water, while the larger values sink to the bottom. The bubble sort makes several passes through the array. On each pass, successive pairs of elements are compared. If a pair is in increasing order (or the values are identical), we leave the values as they are. If a pair is in decreasing order, their values are swapped in the array. Write a program that sorts an array of 10 integers using bubble sort.

```
Data items in original order
   2   6   4   8  10  12  89  68  45  37
Data items in ascending order
   2   4   6   8  10  12  37  45  68  89
```

Lab Exercises

Name:

Lab Exercise 2 — Bubble Sort

Template

```
// Lab 2: bubblesort.cpp
// This program sorts an array's values into ascending order.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   const int arraySize = 10; // size of array a
   int a[ arraySize ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
   int hold; // temporary location used to swap array elements

   cout << "Data items in original order\n";

   // output original array
   for ( int i = 0; i < arraySize; i++ )
      cout << setw( 4 ) << a[ i ];

   // bubble sort
   // loop to control number of passes
   /* Write a for header to loop for one iteration less than the size
      of the array */
   {
      // loop to control number of comparisons per pass
      /* Write a for header to iterate j from 0 and keep
         looping while j is less than arraySize - 1 */
      {
         // compare side-by-side elements and swap them if
         // first element is greater than second element
         /* Write an if statement to test if element j is greater than
            element j + 1 */
         {
            /* Write code to swap the values in element j and
               element j + 1, using hold as temporary storage */
         } // end if
      } // end for
   } // end for

   cout << "\nData items in ascending order\n";

   // output sorted array
   for ( int k = 0; k < arraySize; k++ )
      cout << setw( 4 ) << a[ k ];

   cout << endl;
} // end main
```

Fig. L 7.3 | bubblesort.cpp.

Lab Exercises

Name:

Lab Exercise 2 — Bubble Sort

Problem-Solving Tips

1. Each "bubbling" pass through the array brings one element, the *i*th up to its correct position. This means that the program will require `arraySize - 1` passes through the array to sort the entire array.
2. Each bubbling pass will look at each pair of adjacent elements and swap them if they are not already in sorted order.
3. To swap two elements, the value of one element will have to be stored in a temporary storage variable while the value of the other element is placed in the first, and then the second element can be replaced with the temporary storage value.

Follow-Up Question and Activity

1. This bubble sort algorithm is inefficient for large arrays. Make the following simple modifications to improve the performance of the bubble sort:

   a) After the first pass, the largest number is guaranteed to be in the highest-numbered element of the array; after the second pass, the two highest numbers are "in place," and so on. Instead of making nine comparisons on every pass, modify the bubble sort to make eight comparisons on the second pass, seven on the third pass, and so on.

   b) The data in the array may already be in the proper order or near-proper order, so why make nine passes if fewer will suffice? Modify the sort to check at the end of each pass if any swaps have been made. If none have been made, then the data must already be in the proper order, so the program should terminate. If swaps have been made, then at least one more pass is needed.

Lab Exercise 3 — Salespeople

Name: ______________________ **Date:** ______________________

Section: ______________________

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 7.4)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 7 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Using double-subscripted arrays to store tables of information.
- Nesting `for` loops to access multiple-subscripted arrays.

The follow-up question and activities also will give you practice:

- Using `const int`s to declare identifiers that are used in an array declaration.
- Initializing multidimensional arrays.
- Using character arrays to store strings.

Description of the Problem

Use a double-subscripted array to solve the following problem. A company has four salespeople (1 to 4) who sell five different products (1 to 5). Each salesperson passes in slips for each different type of product sold. Each slip contains the following:

a) The salesperson number

b) The product number

c) The total dollar value of that product sold that day

Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all of the slips for last month are available. Write a program that reads all this information for last month's sales and summarize the total sales by salesperson by product. All totals should be stored in the two-dimensional array `sales`. After processing all the information for last month, print the results in tabular format with each of the columns representing a particular salesperson and each of the rows representing a particular product. Cross total each row to get the total sales of each product for last month; cross total each column to get the total sales by salesperson

Lab Exercises

Name:

Lab Exercise 3 — Salespeople

for last month. Your tabular printout should include these cross totals to the right of the totaled rows and to the bottom of the totaled columns.

Sample Output

```
Enter the salesperson (1 - 4), product number (1 - 5), and total sales.
Enter -1 for the salesperson to end input.
1 1 9.99
1 2 12.49
1 5 19.98
2 3 15.75
2 4 21.23
3 1 17.33
4 5 22.22
4 1 19.65
-1

The total sales for each salesperson are displayed at the end of each row,
and the total sales for each product are displayed at the bottom of each column.

            1           2           3           4           5       Total
1        9.99       12.49        0.00        0.00       19.98       42.46
2        0.00        0.00       15.75       21.23        0.00       36.98
3       17.33        0.00        0.00        0.00        0.00       17.33
4       19.65        0.00        0.00        0.00       22.22       41.87

Total   46.97       12.49       15.75       21.23       42.20
Press any key to continue . . .
```

Template

```cpp
// Lab 3: sales.cpp
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   const int PEOPLE = 5;
   const int PRODUCTS = 6;
   /* Write the declaration for array sales here */
   double value;
   double totalSales;
   double productSales[ PRODUCTS ] = { 0.0 };
   int salesPerson;
   int product;

   // enter sales slips
   cout << "Enter the salesperson (1 - 4), product number (1 - 5), and "
        << "total sales.\nEnter -1 for the salesperson to end input.\n";

   cin >> salesPerson;

```

Fig. L 7.4 | sales.cpp. (Part 1 of 2.)

Lab Exercises

Name:

Lab Exercise 3 — Salespeople

```
   // continue receiving input for each salesperson until -1 is entered
   while ( salesPerson != -1 )
   {
      cin >> product >> value;
      /* Write a statement that adds values to the proper
         element in the sales array */
      cin >> salesPerson;
   } // end while

   cout << "\nThe total sales for each salesperson are displayed at the "
        << "end of each row,\n" << "and the total sales for each product "
        << "are displayed at the bottom of each column.\n " << setw( 12 )
        << 1 << setw( 12 ) << 2 << setw( 12 ) << 3 << setw( 12 ) << 4
        << setw( 12 ) << 5 << setw( 13 ) << "Total\n" << fixed << showpoint;

   // display salespeople and sales
   for ( int i = 1; /* Write condition here */; i++ )
   {
      totalSales = 0.0;
      cout << i;

      // add total sales, and display individual sales
      for ( int j = 1; /* Write condition here */; j++ )
      {
         /* Write a statement that adds the current sales element
            to totalSales */
         cout << setw( 12 ) << setprecision( 2 ) << sales[ i ][ j ];
         /* Write a statement that adds the current sales element
            to productSales */
      } // end inner for

      cout << setw( 12 ) << setprecision( 2 ) << totalSales << '\n';
   } // end outer for

   cout << "\nTotal" << setw( 8 ) << setprecision( 2 )
      << productSales[ 1 ];

   // display total product sales
   for ( int j = 2; j < PRODUCTS; j++ )
      cout << setw( 12 ) << setprecision( 2 ) << productSales[ j ];

   cout << endl;
} // end main
```

Fig. L 7.4 | `sales.cpp`. (Part 2 of 2.)

Problem-Solving Tips

1. This problem asks the reader to input a series of numbers representing the salesperson number, product number and the dollar amount. The product number and salesperson number represent the row subscript and column subscript in the `sales` array where the dollar amount is added. Each array begins with subscript zero; therefore, it is recommended that you oversize the array by one element in each dimension. This allows you to map the product number and salesperson number directly to a subscript without having to subtract one.
2. Table columns contain the total sales for each product. Table rows contain the sales figures for each salesperson. To create the output, the table header must first be printed. (See template.) When program

Lab Exercises

Name:

Lab Exercise 3 — Salespeople

control reaches the outer for loop, the salesperson number is printed. The inner for loop prints the amount of each product that the salesperson sold. When the inner loop finishes, control returns to the outer loop and the \n character is printed.

3. To display totals in the right-most column, simply sum each element in the row and display the total. This is best done when the array is output. To display the totals at the bottom, declare a one-dimensional array of five elements. While outputting sales, simply add the current column's value to the appropriate element of the single-subscripted array. After outputting sales and the totals for each row, iterate through the single-subscripted array and output its values.

Follow-Up Questions and Activities

1. Explain why keyword const must be present when declaring the variables PRODUCTS and PEOPLE. Why do these two constants have the values 6 and 5 rather than 5 and 4?

2. Change the declaration of productSales, in your solution that corresponds to line 18 in the program template, so that salesperson 1 has sold $75.00 of product 3 initially and so that salesperson 4 has sold $63.00 of product 1 initially. All other array values should be initialized to 0.0. [*Hint:* Use an initializer list to initialize the array.]

3. Create an additional array that stores the names of all of the salespeople. Allow the user to input the first names of the four employees. Limit the names to 20 characters. When generating the output table, use the names of the salespeople rather than numbers.

Lab Exercises

Name:

Debugging

Name: ______________________________ **Date:** ______________________________

Section: ______________________________

The program (Fig. L 7.5) in this section does not run properly. Fix all the compilation errors so that the program will compile successfully. Once the program compiles, compare the output with the sample output, and eliminate any logic errors that may exist. The sample output demonstrates what the program's output should be once the program's code has been corrected.

Sample Output

```
Here is the grade database

      Name   1   2   3   4   5   6   7   8   9  10
       Bob  56  67  83  81  70  84  94  64  68  86
      John  76  89  81  42  66  93 104  91  71  85
       Joe  65  69  91  89  82  93  72  76  79  99

Bob's highest grade is: 94
Bob's lowest grade is: 56
John's highest grade is: 104
John's lowest grade is: 42
Joe's highest grade is: 99
Joe's lowest grade is: 65
```

Broken Code

```
// Debugging: grades.cpp
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;

const int NUM_GRADES = 10;
const int NUM_SUDENTS = 3;

int findHighest( int );
int findLowest( int * );
void printDatabase( const int [][], const char [][ 20 ] );

int main()
{
   int student1[ NUM_GRADES ] = { 0 };
   int student2[ NUM_GRADES ] = { 76, 89, 81, 42, 66, 93, 104,
                                 91, 71, 85, 105 };
   int student3[ NUM_GRADES ] = { 65, 69, 91, 89, 82, 93, 72,
                                 76, 79, 99 };
   char names[ NUM_SUDENTS ][ 20 ] = { "Bob", "John", "Joe" };

```

Fig. L 7.5 | grades.cpp (Part 1 of 3.)

Lab Exercises

Name:

Debugging

```
   int database[ NUM_SUDENTS ][ NUM_GRADES ];
   int i = 0;

   srand( time( 0 ) );

   // initialize student1
   for ( i = 0; i < NUM_GRADES; i++ )
      student1[ NUM_GRADES ] = rand() % 50 + 50;

   // initialize database
   for ( i = 1; i < NUM_GRADES; i++ ) {
      database[ 0 ][ i ] = student1[ i ];
      database[ 1 ][ i ] = student2[ i ];
      database[ 2 ][ i ] = student3[ i ];

   } // end for

   printDatabase( database, studentNames );

   for ( i = 0; i < NUM_SUDENTS; i++ ) {
      cout << studentNames[ i ] << "'s highest grade is: "
           << findHighest( student1 ) << endl
           << studentNames[ i ] << "'s lowest grade is: "
           << findLowest( database[ i ] ) << endl;

   } // end for
} // end main

// determine largest grade
int findHighest( int )
{
   int highest = a[ 0 ];

   for ( int i = 1; i <= NUM_GRADES; i++ )
      if ( a[ i ] > highest )
         highest = a[ i ];

   return highest;

} // end function findHighest

// determine lowest grade
int findLowest( int a[] )
{
   int lowest = a[ 0 ];

   for ( int i = 1; i < NUM_GRADES; i++ )
      if ( a[ i ] < lowest )
         lowest = a[ i ];

   return lowest;

} // end lowestGrade

```

Fig. L 7.5 | grades.cpp. (Part 2 of 3.)

Lab Exercises

Name:

Debugging

```
// output data
void printDatabase( int a[][ NUM_GRADES ], char names[][ 20 ] )
{
   cout << "Here is the grade database\n\n"
        << setw( 10 ) << "Name";

   for ( int n = 1; n <= NUM_GRADES; n++ )
      cout << setw( 4 ) << n;

   cout << endl;

   for ( int i = 0; i < NUM_SUDENTS; i++ ) {
      cout << setw( 10 ) << names[ i ];

      for ( int j = 0; j < NUM_GRADES; j++ )
         cout << setw( 4 ) << a[ i, j ];

      cout << endl;

   } // end for

   cout << endl;

} // end printDatabase
```

Fig. L 7.5 | `grades.cpp`. (Part 3 of 3.)

Postlab Activities

Coding Exercises

Name: ____________________________ **Date:** ____________________________

Section: ____________________________

These coding exercises reinforce the lessons learned in the lab and provide additional programming experience outside the classroom and laboratory environment. They serve as a review after you have completed the *Prelab Activities* and *Lab Exercises* successfully.

For each of the following problems, write a program or a program segment that performs the specified action:

1. Write a line of code that declares a 101-element array, `a`.

2. Initialize all elements of the array in *Coding Exercise 1* to `-1`.

3. Write a line of code that accesses the seventh element of the array in *Coding Exercise 2* and sets its value to `7`.

4. Use the `rand` function to randomly select an element of the array created in *Coding Exercise 1*. Assign to that element the value `2`.

Postlab Activities

Name:

Coding Exercises

5. Write a function `printArray` that can print the contents of the array created in *Coding Exercise 1*. Assume that the array's size is passed as a second argument to the function. Place a space between every number that is printed. In addition, print a new line after every 20 elements.

6. Write a program that generates a multiplication table. Use a double-subscripted array to represent your table. The numbers used in the calculations can be in the range 1–5 (i.e., 5 * 5 = 25 is the largest value in this table). Use a nested `for` statement to populate the array with the results of each calculation.

Postlab Activities

Name:

Programming Challenges

Name: ______________________ **Date:** ______________________

Section: ______________________

The *Programming Challenges* are more involved than the *Coding Exercises* and may require a significant amount of time to complete. Write a C++ program for each of the problems in this section. The answers to these problems are available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`. Pseudocode, hints and/or sample outputs are provided to aid you in your programming.

1. Use a one-dimensional array to solve the following problem. A company pays its salespeople on a commission basis. The salespeople each receive $200 per week plus 9 percent of their gross sales for that week. For example, a salesperson who grosses $5000 in sales in a week receives $200 plus 9 percent of $5000, or a total of $650. Write a program (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson's salary is truncated to an integer amount):

   a) $200–299

   b) $300–399

   c) $400–499

   d) $500–599

   e) $600–699

   f) $700–799

   g) $800–899

   h) $900–999

   i) $1000 and over

Hints:

- Calculate salary as a `double`. Then use `static_cast< int >` to truncate the salaries and convert them to integers. Divide by `100` to obtain an array index.

Postlab Activities

Name:

Programming Challenges

- Sample output:

```
Enter employee gross sales (-1 to end): 10000
Employee Commission is $1100.00

Enter employee gross sales (-1 to end): 4235
Employee Commission is $581.15

Enter employee gross sales (-1 to end): 600
Employee Commission is $254.00

Enter employee gross sales (-1 to end): 12500
Employee Commission is $1325.00

Enter employee gross sales (-1 to end): -1
Employees in the range:
$200-$299 : 1
$300-$399 : 0
$400-$499 : 0
$500-$599 : 1
$600-$699 : 0
$700-$799 : 0
$800-$899 : 0
$900-$999 : 0
Over $1000: 2
```

2. Use a one-dimensional array to solve the following problem: Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, validate it and store it in the array only if it is not a duplicate of a number already read. After reading all the values, display only the unique values that the user entered. Provide for the "worst case" in which all 20 numbers are different. Use the smallest possible array to solve this problem.

Hints:

- Compare every value input to all existing array elements. If it is a duplicate, set a flag variable to 1. This flag should be used to determine whether it is necessary to print the value.
- Use a counter variable to keep track of the number of elements entered into the array and the array position where the next value should be stored.

Postlab Activities

Name:

Programming Challenges

- Sample output:

```
Enter 20 integers between 10 and 100:
10
5
Invalid number.
20
30
40
50
60
70
80
90
100
110
Invalid number.
10
Duplicate number.
11
22
33
44
55
66
77
88
99
45

The nonduplicate values are:
10 20 30 40 50 60 70 80 90 100 11 22 33 44 55 66 77 88 99 45
```

3. *(The Sieve of Eratosthenes)* A prime integer is any integer greater than 1 that is evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:

   a) Create an array with all elements initialized to 1 (true). Array elements with prime subscripts will remain 1. All other array elements will eventually be set to zero. You will ignore elements 0 and 1 in this exercise.

   b) Starting with array subscript 2, every time an array element is found whose value is 1, iterate through the remainder of the array and set to zero every element whose subscript is a multiple of the subscript for the element with value 1. For array subscript 2, all elements beyond 2 in the array that are multiples of 2 will be set to zero (subscripts 4, 6, 8, 10, etc.); for array subscript 3, all elements beyond 3 in the array that are multiples of 3 will be set to zero (subscripts 6, 9, 12, 15, etc.); and so on.

   When this process is complete, the array elements that are still set to one indicate that the subscript is a prime number. These subscripts can then be printed. Write a program that uses an array of 1000 elements to determine and print the prime numbers between 2 and 999. Ignore elements 0 and 1 of the array.

Hints:

- Use a loop to find all elements that are set to 1. (This must be done in order.) Set all multiples of that element to 0.
- Print the primes by looping through the array searching for elements equal to 1. Print their subscript. Increment a counter by one each time a prime number is printed.

Postlab Activities

Name:

Programming Challenges

- Sample output:

```
 2 is a prime number.
 3 is a prime number.
 5 is a prime number.
 7 is a prime number.
11 is a prime number.
13 is a prime number.
17 is a prime number.
19 is a prime number.
23 is a prime number.
29 is a prime number.
31 is a prime number.
...
929 is a prime number.
937 is a prime number.
941 is a prime number.
947 is a prime number.
953 is a prime number.
967 is a prime number.
971 is a prime number.
977 is a prime number.
983 is a prime number.
991 is a prime number.
997 is a prime number.
A total of 168 prime numbers were found.
```

4. Write a recursive function `recursiveMinimum` that takes an integer array, a starting subscript and an ending subscript as arguments and returns the smallest element of the array.The function should stop processing and return when the starting subscript equals the ending subscript. [*Note:* This problem is intended for those students who have studied recursion in Sections 6.19–6.21 of *C++ How to Program, Seventh Edition*.]

Hints:

- Write a program to test your function. Populate an array with randomly generated integers.
- Function `recursiveMinimum` should take as its arguments the array, a low value and a high value. The low and high values represent the boundaries of the array, respectively.
- Recursive functions involving arrays approach their base case by reducing the size of the array using boundaries, not by literally passing a smaller array.Your function should approach the base case in the following manner: increment `low` by one each time until `low` equals `high`.
- Sample output:

```
Array members are:
  309  893   72  270  109  830  338  487  240  505

Smallest element is: 72
```