UMass

Electrical and Computer Engineering Department

ECE 264, Object-Oriented Software Development

Lab Assignment 7

Lab 7 :: 100 points (see Grading Notes for details) ::
Wednesday lab session (March 27) Due March 29, Friday by 5:00 pm
Monday lab session (April 1) Due April 3, Wednesday by 5:00 pm

1. Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 7 of *C++ How To Program, 8th 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.
- Using double-subscripted arrays to store tables of information.
- Nesting for loops to access multiple-subscripted arrays.

2. Deliverables

Create "lab7" sub-directory on your M:\ drive. Submit your file to this sub-directory on the M:\ drive. Call your project *lab7_ Dice and lab7_Sales* respectively. You should place all the source files (.h and .cpp) on the"lab7" sub-directory. Failure to meet this specification will reduce your grade, as described in the ECE 264 lab grading handout, which you are strongly encouraged to read before starting the lab.

3. Description of the Problem 1 (*lab7_ Dice*)

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).

	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.

UMass
Electrical and Computer
Engineering Department

ECE 264, Object-Oriented Software Development

Lab Assignment 7

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
```

ECE 264, Object-Oriented Software Development

Lab Assignment 7

```
19     int y; // second die
20     /* Write declaration for array sum here. Initialize all
21        elements to zero. Use SIZE for the number of elements */
22
23     srand( time( 0 ) );
24
25     // roll dice 36,000 times
26     /* Write a for statement that iterates ROLLS times. Randomly
27        generate values for x (i.e., die1) and y (i,e, die2)
28        and increment the appropriate counter in array sum that
29        corresponds to the sum of x and y */
30
31     cout << setw( 10 ) << "Sum" << setw( 10 ) << "Total" << setw( 10 )
32        << "Expected" << setw( 10 ) << "Actual\n" << fixed << showpoint;
33
34
35     // display results of rolling dice
36     for ( int j = 2; j < SIZE; j++ )
37        cout << setw( 10 ) << j << setw( 10 ) << sum[ j ]
38           << setprecision( 3 ) << setw( 9 )
39           << 100.0 * expected[ j ] / 36 << "%" << setprecision( 3 )
40           << setw( 9 ) << 100.0 * sum[ j ] / 36000 << "%\n";
41  } // 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%.

4. Description Problem 2 (*lab7_Sales)*

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

ECE 264, Object-Oriented Software Development

Lab Assignment 7

Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all of the slips for last monrth 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 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
3 3 5.99
2 2 4.99
-1

The total sales for each sales person 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      0.00      0.00      0.00      0.00      9.99
2      0.00      4.99      0.00      0.00      0.00      4.99
3      0.00      0.00      5.99      0.00      0.00      5.99
4      0.00      0.00      0.00      0.00      0.00      0.00

Total  9.99      4.99      5.99      0.00      0.00
```

Template

ECE 264, Object-Oriented Software Development

Lab Assignment 7

```
1  // Lab 3: sales.cpp
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5
6  int main()
7  {
8     const int PEOPLE = 5;
9     const int PRODUCTS = 6;
10    /* Write the declaration for array sales here */
11    double value;
12    double totalSales;
13    double productSales[ PRODUCTS ] = { 0.0 };
14    int salesPerson;
15    int product;
16
17    // enter sales slips
18    cout << "Enter the salesperson (1 - 4), product number (1 - 5), and "
19         << "total sales.\nEnter -1 for the salesperson to end input.\n";
20
21    cin >> salesPerson;
22
23    // continue receiving input for each salesperson until -1 is entered
24    while ( salesPerson != -1 )
25    {
26       cin >> product >> value;
27       /* Write a statement that adds values to the proper
28          element in the sales array */
29       cin >> salesPerson;
30    } // end while
```

Fig. L 7.4 | sales.cpp

ECE 264, Object-Oriented Software Development

Lab Assignment 7

```
31
32     cout << "\nThe total sales for each salesperson are displayed at the "
33          << "end of each row,\n" << "and the total sales for each product "
34          << "are displayed at the bottom of each column.\n " << setw( 12 )
35          << 1 << setw( 12 ) << 2 << setw( 12 ) << 3 << setw( 12 ) << 4
36          << setw( 12 ) << 5 << setw( 13 ) << "Total\n" << fixed << showpoint;
37
38     // display salespeople and sales
39     for ( int i = 1; /* Write condition here */; i++ )
40     {
41        totalSales = 0.0;
42        cout << i;
43
44        // add total sales, and display individual sales
45        for ( int j = 1; /* Write condition here */; j++ )
46        {
47           /* Write a statement that adds the current sales element
48              to totalSales */
49           cout << setw( 12 ) << setprecision( 2 ) << sales[ i ][ j ];
50           /* Write a statement that adds the current sales element
51              to productSales */
52        } // end inner for
53
54        cout << setw( 12 ) << setprecision( 2 ) << totalSales << '\n';
55     } // end outer for
56
57     cout << "\nTotal" << setw( 8 ) << setprecision( 2 )
58        << productSales[ 1 ];
59
60     // display total product sales
61     for ( int j = 2; j < PRODUCTS; j++ )
62        cout << setw( 12 ) << setprecision( 2 ) << productSales[ j ];
63
64     cout << endl;
65  } // 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 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.

5. Testing Your Program

- For this program, there is no user input so the only way to test your program is to run it and see if it displays all of the information correctly.
- In all of your programs, but especially a program where there isn't any user input, you should focus on making the output easy to read. One of the most difficult things for a user of your program to deal with is poorly formatted output. The easier your output is to read, the easier it is to identify the relevant information that you're producing.