**ITCS 1212L**

**Prelab and Lab 9**

# Arrays

**Learning Objectives:**

* **Practice with arrays.**

**For prelab, make sure to come to the lab with at least the basic algorithm, main variables and functions.**

**Lab 9A:**

1. Write the code to declare array to hold ten test grades.
2. Now write the ‘for loop’ that will prompt and allow the user to enter 10 grades.
3. After that, write the ‘for loop’ to print the values of each of the ten elements in the array.
4. Now, add to this, the code to validate that each grade is between 0 and 100 before assigning it a slot in the array. (This is a do-while loop inside the ‘for loop’).
5. Calculate and print the average of the 10 grades.
6. Find and print the minimum grade in the array and the maximum grade in the array.

**TA Check: \_\_\_\_\_\_\_\_\_\_\_**

**Lab 9B:**

Write a program that declares two arrays. Each array will hold ten integers.Read in values for the first array from the user. Then assign the values into the second array in reverse order. Print the contents of both arrays to verify the first array was reversed in the second array.

**Sample Output:**

Array 1: 1 2 3 4 5 6 7 8 9 10

Array 2: 10 9 8 7 6 5 4 3 2 1

**TA Check: \_\_\_\_\_\_\_\_\_\_\_\_**

**Lab 9C:**

Write a program that declares an array of 15 integers. Fill the array with random integers from 1 to 20, (generate the random numbers, the user does not enter them).Print the contents of the array.

Once you have done that, count how many of the values in the array are even and how many are odd. Print each of these counts.

**Sample Output:**

Array: 3 4 8 19 12 15 11 16 2 1 4 9 10 12 15

Even Count: 8

Odd Count: 7

**TA Check: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**Parallel Arrays:**

In this lab you will use arrays in parallel. This means that several arrays are used together to store related data. The data in the arrays are related by the subscript.

For example, look at how these three arrays that store sales, costs and profits by quarters are used in parallel.

**double costs[4] = {300, 400, 500, 800}; double sales[4] = {500, 600, 800,1200}; double profits[4];**

**for(int i = 0; i < 4; i++) {**

**profits[i] = sales[i] – costs[i];**

**}**

In the example above **sales[0]** stores the sales amount for the first quarter for some company. This works with the costs array, because **costs[0]** stores the sales for the first quarter, and the memory location at **profits[0]** stores the net profit for this company for the first quarter.

**Lab 9D:**

In this program you will use two arrays in parallel and perform some useful tasks that can be done using arrays.

1. Create an array to hold the ID’s (int) of 10 sales employees. Then create an array to hold the sales amounts for each employee that will parallel the array of IDs.
2. Write a loop to read in data for both arrays. (what type of loop is this?)
3. Write a second loop to display the data from both arrays lined up **with column headings**.

ID Sales

2121 $3456

2235 $5324

1. Write the code to find which salesperson had the highest sales. Print the employee’s ID and sales amount.
2. Write the code to find which salesperson had the lowest sales. Print the employee’s ID and sales amount.
3. Write the code to find the average sales for all 10 salespeople. Print this average.

Print the IDs of the salespeople who had below average sales and print the IDs of the salespeople who had above average sales. Make the output meaningful.

1. Allow the user to enter an ID. Search the ID array for that id number and then print the ID and the sales for that salesperson. If the ID is not present in the array of IDs print a message stating the search ID was not found.

**TA Check: \_\_\_\_\_\_\_\_\_\_\_**

**Lab 9E:**

In this program you will use arrays in parallel to calculate your numeric grade in this course using test data you create. Use three arrays in parallel. The first array will hold values for each component of this class; three exam grades, a lab grade, an assignment grade, a lecture grade and a final exam grade. The second array will hold the weights for each value in the first array. The third array will hold a description/prompt of what each element contains. Make sure you can calculate a weighted average on paper before trying to code it in C++!

Supply values for each element in the first array and then use the following arrays for the prompts and the weights:

string labels[7] = {“Exam 1”, “Exam 2”, “Exam 3”, “Final Exam”, “Programming Assignment”, “Other Activities”, “Lab Test Grade”};

double weights[7] = {.10, .10, .10, .10, .20, .10, .30};

double myScores[7] ; //you can supply fictitious values for each category as input

**Sample:**

for (int index = 0; index < 7; index++)

{

cout << “Enter you score for “ << labels[index] << “: “;

cin >> myScores[index];

}

Print the final average correct to one decimal place.

**TA Check: \_\_\_\_\_\_\_\_\_\_\_\_**

**Lab 9F:**

1. Write a webservice program to get the stock market for 3 different stocks of your choice for the 12 months of year 2014 into a 2-D array (two-dimensional array) of doubles. Automate your program so that it does not prompt the user for the month. It rather automatically loads the array from the internet for each of the stock values for January to December of 2014. Do this part in a function called initArray(). In addition, you need to develop the following functions.
2. A function to calculate and display the total value of all stocks as well as total value of each of the stocks for the year 2014. Test the function by calling it in the main function.

Total value of all stocks for year 2014 is: $XX.XX

Total value of stock XXX for year 2014 is: $XX.XX

And so on…

1. A function to calculate and display the average monthly value of all of the stocks as well as the average value of each of the stocks for the year 2014. Test the function by calling it in the main function.

Average value of all stocks for year 2014 is: $XX.XX

Average value of stock XXX for year 2014 is: $XX.XX

And so on…

1. A function to calculate and display the average monthly value of the 3 stocks for each month. Again, test the result in the main.

The average stocks value for month 1 is: $XX.XX

The average stocks value for month 2 is: $XX.XX

And so on…

1. A function to calculate and display the month with the highest (and the month with the lowest) value for all of the stocks for the entire year. Test the function in the main.

The month with the highest value of stocks for year 2014 is: X

The month with the lowest value of stocks for year 2014 is: X

1. A function to calculate and display the stock with the lowest value of all of the stocks for each month. Test your function and show the results.

Stock XXX had the lowest stocks value for month 1: $YY.ZZ

Stock XXX had the lowest stocks value for month 2: $YY.ZZ

,and so on…

1. A function to calculate and display the stock with the highest value of all of the stocks for each month. Test your function and show the results.

Stock XXX had the highest stocks value for month 1: $YY.ZZ

Stock XXX had the highest stocks value for month 2: $YY.ZZ

,and so on…

1. Create a menu() function to display all the above options (and an exit option). Test the main program again.
2. What do you think is the difference between top-down design and bottom-up design of large programs? Which method do you think it was? How about the method that we used to do before?

**TA Check: \_\_\_\_\_\_\_\_\_\_\_\_**