Chapter 4 Lab

**Loops**

**Lab Objectives**

* Be able to convert an algorithm using control structures into C++
* Be able to write a while loop
* Be able to write an do-while loop
* Be able to write a for loop
* Be able to read from an input file and write to an output file

**Introduction**

This is a simulation of rolling dice. Actual results approach theory only when the sample size is large. So we will need to repeat rolling the dice a large number of times (we will use 10,000). The theoretical probability of rolling doubles of a specific number is 1 out of 36 or approximately 278 out of 10,000 times that you roll the pair of dice. Since this is a simulation, the numbers will vary a little each time you run it. Recall how we used a random number generator in lab 2 to pick lottery numbers, this time using it to generate a number 1-6 for each roll of a die.

We will continue to use control structures that we have already learned, while exploring control structures used for repetition. We shall also continue our work with algorithms, translating a given algorithm to C++ in order to complete our program. We will start with a while loop, then use the same program, changing the while loop to a do-while loop, and then a for loop.

We will be introduced to file input and output. We will read a file and use the numbers to calculate the mean and standard deviation.

First, we will learn how to use file output to get results printed to a file. Next we will use file input to read the numbers from a file and calculate the mean. Finally, we will see that when the file is closed and then reopened, we will start reading from the top again so that we can calculate the standard deviation.

**Task #1 While loop**

1. Create a project in Visual Studio and add the file **diceSimulation.cpp**. Since there is a large part of the program missing, the output will be incorrect if you run diceSimulation.cpp.
2. I have declared all the variables. You need to add code to simulate rolling the dice and keeping track of the doubles. Convert the algorithm below to C++ and place it in the main method after the variable declarations, but before the output statements. You will be using several control structures: a **while** loop and an if-else-if statement nested inside another if statement. Use the indenting of the algorithm to help you decide what is included in the loop, what is included in the if statement, and what is included in the nested if-else-if statement.

*Repeat while the number of dice rolls are less than the number of times the dice should be rolled.*

*Roll the first die*

*Get the value of the first die*

*Roll the second die*

*Get the value of the second die*

*If the value of the first die is the same as the value of the second die*

*If value of first die is 1*

*Increment the number of times snake eyes were rolled*

*Else if value of the first die is 2*

*Increment the number of times twos were rolled*

*Else if value of the first die is 3*

*Increment the number of times threes were rolled*

*Else if value of the first die is 4*

*Increment the number of times fours were rolled*

*Else if value of the first die is 5*

*Increment the number of times fives were rolled*

*Else if value of the first die is 6*

*Increment the number of times sixes were rolled*

*Increment the number of times the dice were rolled*

1. Compile and run. You should get numbers that are somewhat close to 278 for each of the different pairs of doubles. Run it several times. You should get different results than the first time, but again it should be somewhat close to 278.

**Task #2 Using Other Types of Loops**

1. Copy the entire while loop and paste it below the original while loop. Comment out the original while loop. Change the while loop to a **do-while** loop. Compile and run. You should get the same results.
2. Copy the entire do-while loop and paste it below the original do-while loop. Comment out the original do-while loop. Change the do loop to a **for** loop. Compile and run. You should get the same results.

**Task #3 Writing Output to a File**

1. Copy the files statsDemo.cpp and numbers.txt.
2. First, we will write output to a file
   * 1. Open the file results.txt.
     2. Use a condition to test that it is opened (if not open, return 1 to indicate error)
     3. Format the output to have 3 decimal places
     4. Print the mean and the standard deviation to the file (both labeled)
     5. Close the file
   1. Compile, debug, and run. You will need to type in the filename Numbers.txt. You should get no output to the console but running the program will create a file called Results.txt with your output, formatted to 3 decimal places. The output you should get at this point is: mean = 0.000, standard deviation = 0.000. This is not the correct mean or standard deviation for the data, but we will fix this in the next tasks.

**Task #4 Calculating the Mean**

1. Now we need to add lines to allow us to read from the input file and calculate the mean.
2. Open the file the user requested
3. Use a condition to test that it is opened (if not open, return 1 to indicate error)
4. Write a loop that continues until you are at the end of the file.
5. The body of the loop will
   1. Read the value
   2. If you were able to read the value
      1. Add the value to the accumulator
      2. Increment the counter
6. When the program exits the loop, calculate and store the mean. The mean is calculated by dividing the accumulator by the counter.
7. Close the input file.
8. Compile, debug, and run. You should now get a mean of 77.444, but the standard deviation will still be 0.000 at this time.

**Task #5 Calculating the Standard Deviation**

1. Reinitialize both the sum and the counter to 0.
2. We need to reconnect to the file so that we can start reading from the top again, so we can reopen the file.
3. Use a condition to test that it is opened (if not open, return 1 to indicate error)
4. Write a loop that continues until you are at the end of the file.
5. The body of the loop will
   1. Read the value
   2. If you were able to read the value
      1. Subtract the mean from the value and store the result in the difference.
      2. Add the square of the difference to the accumulator
      3. Increment the counter
6. When the program exits the loop, calculate and store the standard deviation. The variance is calculated by dividing the accumulator (sum of the squares of the difference) by the counter. Calculate the standard deviation by taking the square root of the variance.
7. Close the input file.
8. Compile, debug, and run. You should get a mean of 77.444 and a standard deviation of 10.021.