**Chapter 2 Lab**

**C++ Fundamentals**

# Lab Objectives

* Write arithmetic expressions to accomplish a task
* Use casting to convert between primitive types
* Use a value-returning library method and a library constant
* Use string functions to manipulate string data
* Read string input containing spaces
* Generate random numbers

**Introduction**

This lab is designed to give you practice with some of the basics in C++. We will continue ideas from lab 1 by correcting logic errors while looking at mathematical formulas in C++. We will explore the difference between integer division and division on your calculator as well as reviewing the order of operations. We will also look at the C++ APIs to examine the documentation on prewritten functions that are available for our use.

We will learn how to use mathematical formulas that are preprogrammed in C++. On your calculator there are buttons to be able to do certain operations, such as raise a number to a power or use trig functions. Similarly, in C++, we will have programs that are available for our use that will also do these operations. Mathematical operations that can be performed with the touch of a button on a calculator are also available in the math library. We will learn how to use a math function to cube the radius in the formula for finding the volume of a sphere.

This lab also introduces another way to read user input. We have already seen how to use cin to read numbers and one-word strings. However, to read in spaces, we need to use another way of reading input.

Finally, we will simulate the Wisconsin Lottery Pick 3. You will ask the user for 3 numbers from 0 to 9. We will use a random number generator to select the winning numbers in the lottery. You will then check the user’s choices against the winning numbers and report how many numbers matched.

# Task #1 Correcting Logic Errors in Formulas

1. Create a project in Visual Studio and add the file **numericTypes.cpp**.
2. Compile the source file, run the program, and observe the output. Some of the output is ***incorrect***. You need to correct **logic errors** in the average formula and the temperature conversion formula. The logic errors could be due to conversion between data types, order of operations, or formula problems. DO NOT change the data type on the named constants, instead use type casting. The necessary formulas are

*average* =  C = 

1. Each time you make changes to the program code, you must compile again for the changes to take effect before running the program again.
2. Make sure that the output makes sense before you continue. The average of 95 and 100 should be 97.5 and the temperature that water boils is 100 degrees Celsius.

## Task #2 Using the *getline* Function for User Input

1. Prompt the user to enter his/her first and last name.
2. Read the name from the keyboard using the getline method and store it into a variable called fullName (you will need to declare any variables you use).
3. Print out the fullName.
4. Compile, debug, and run, using your name as test data.
5. Since we are adding on to the same program, each time we run the program we will get the output from the previous tasks before the output of the current task.

**Task #3 Working with Strings**

1. Examine the string class in the C++ API. Look up the documentation on each of the methods that you will be using below.
2. Use the **at** method to get the first character in fullName and store it in a variable called firstInitial (you will need to declare any variables that you use).
3. Print out the user’s first initial.
4. Add a line that prints out how many characters (including the space) are in the string stored in fullName (use the function **length** to obtain that information).
5. Compile, debug, and run. The new output added on after the output from the previous tasks should have your first initial and the number of characters in your name including the space.

**Task #4 Using Predefined Math Functions**

1. Add a line that prompts the user to enter the diameter of a sphere.
2. Read in and store the number into a variable called diameter (you will need to declare any variables that you use).
3. The diameter is twice the radius, so calculate and store the radius in an appropriately named variable.
4. The formula for the volume of a sphere is

*π r3*

Convert the formula to C++ and add a line which calculates and stores the value of volume in an appropriately named variable. Use 3.14159265359 for π as a named constant and pow to cube the radius. Be sure to #include <cmath>.

1. Print your results to the screen with an appropriate message and using only 1 decimal point. Don’t forget to include the needed library file.
2. Compile, debug, and run using the following test data and record the results.

|  |  |  |
| --- | --- | --- |
| Diameter | Volume (hand calculated) | Volume (resulting output) |
| 2 |  |  |
| 25.4 |  |  |
| 875,000 |  |  |
| two |  |  |

1. Entering the word when the program was expecting a number resulted in what is called a **runtime error**. Notice that runtime can occur on programs which compile and run on many other sets of data. This emphasizes the need to thoroughly test you program with all possible kinds of data.

**Task #5 Using a Random Number Generator**

1. Use a random number generator to select 3 winning numbers and print them out.
2. Compile, debug, and run several times. Do you get the same 3 numbers each time? If so, add a line of code to seed the random number generator. Don’t forget to include the needed library file. Compile, debug and run to make sure you get a different set of 3 numbers each time.

**Post Lab Questions**

1. String is class (user-defined type) in the C++ library rather than a primitive type, therefore has functions that can operate on string objects. List 2 functions that you used in this lab and what they do.
2. The cmath library contain functions to do math operations that you typically used your calculator to perform.
   1. List the function that we used.
   2. List 3 other functions available in cmath.