LAB # 6

Expressions, Input, Output and Data Type Conversions

Objective:

- To learn input and formatted output statements
- To learn data type conversions (coercion and casting)
- To work with constants and mathematical functions

Theory:

The cout Statement:

The cout statement invokes an output stream, which is a sequence of characters to be displayed to the screen.

The insertion operator << inserts the string of characters, say “Hi there” into the output stream that goes to the screen.

Input Instructions:

The cin statement transfers data into the computer from the keyboard. The extraction operator >> extracts an item from the input stream.

Just as cout is of type ostream, cin is considered to be an istream (input stream) data type. In order to use cin and cout in a C++ program, the #include <iostream> directive should be included in the header.

Every cin statement should be preceded by a cout statement that indicates to the user the data to be input. Such a cout statement is called a prompt.

Strings:

String is a sequence of characters. One way to store this is through an array of char-acters, often referred to as a C-string in C++. When using this method to define a string, the programmer must indicate how many characters it can hold. The last char-acter must be reserved for the end-of-string character ‘\0’ which marks the end of the string. In Example 2 below, the variable name can hold up to 11 characters even though the size of the array indicates 12. The extra character is reserved for the end-of-string marker. We can define a variable to be a string object: Example 1 below.

Example 1 (using a string object)

```
string name;
cout << "What is your
name"; cin >> name;
cout << "Hi " << name << endl;
```

Example 2 (using a C-string)

```
char name[12]
cout << "What is your
name"; cin >> name;
cout << "Hi " << name << endl;
```

“Mary Lou”, would not be read into one variable using cin >>. We can get around this restriction by using special functions for reading whole lines of input.

getline(cin, name);

Formatted Output

C++ has special instructions that allow the user to control output attributes such as spacing, decimal point precision, data formatting and other features

Example:

cout << fixed

This displays the output in decimal format rather than scientific notation.
cout << showpoint;
This forces all floating-point output to show a decimal point, even if the values are whole numbers
cout << setprecision(2);
This rounds all floating-point numbers to 2 decimal places

Data Type Conversions:

Whenever an integer and a floating point variable or constant are mixed in an operation, the integer is changed temporarily to its equivalent floating point. This automatic conversion is called implicit type coercion.

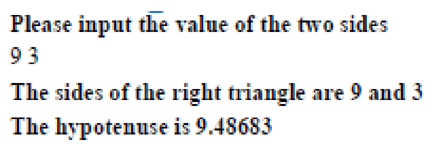
Type conversions can be made explicit (by the programmer) by using the following general format: static_cast<DataType>(Value). This is called type casting or type conversion.

Lab Task:

Do task 6.1, 6.2 and 6.3 given in file attached on LMS and attach your source code and output window with this file. And write your observation with lab task.

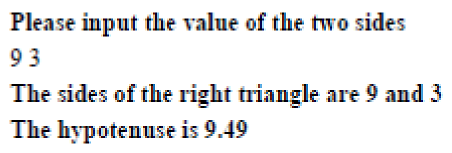
Post Lab:

Task1: Write a program that will take the measure of two sides of a right angled triangle from user and will calculate the hypotenuse of the triangle. The sample run should look like this:



```
Please input the value of the two sides
9 3
The sides of the right triangle are 9 and 3
The hypotenuse is 9.48683
```

Task 2: Alter the program so that the sample run now looks like the following:



```
Please input the value of the two sides
9 3
The sides of the right triangle are 9 and 3
The hypotenuse is 9.49
```

Learning Outcomes:

Upon successful completion of the lab, students will be able to:

LO1: Do data type conversions
LO2: Work with constants and mathematical functions
LO3: Format Output Statements