Lab 2: Simple I/O, variables, and expressions

Simple I/O

Input and output (or I/O) in C++ is handled via streams. (Plain C handles input and output very differently. For now we will not worry about C’s approach.) Streams can be thought of as sources of or destinations for values. For example, for your program to receive input it must receive values from some source, typically the keyboard. So, in C++ we would set up a ‘stream’ to receive data from the keyboard.

Each stream has to have a name so the program can refer to it. The two most common streams are called cin and cout. cin ‘streams’ input from the keyboard. cout ‘streams’ data to the screen. Later we will learn how to create streams that direct values to and from files (file streams). The put to (double less than) (<<) and get from (double greater than) (>>) operators are used in conjunction with the I/O streams. For example, the statement:

```cpp
cout << ''Hello, please enter an integer.'';
```

sends the string in (double) quotes to the (standard) output stream, typically the computer screen (or console).

Variables and Expressions

Variables are used to store values. Each variable must be declared before it can be used. Declaring a variable means giving it a name and a type. A variable’s type determines how much space is set aside in the computer’s memory to store the value and how the value being stored is interpreted. At the most basic level all values are stored as binary numbers; a variable’s type determines whether that binary number is interpreted as an integer, a character or as some other type of value. For example, to declare a variable of type integer called sampleVariable we would write:

```cpp
int sampleVariable;
```

You can also initialize a variable with a value (that is assign it a value) when you declare it:

```cpp
int sampleVariable = 15;
```

For the most part expressions in C/C++ are handled the way we would expect from our mathematics background. For example, the statement:

```cpp
sampleVariable = 5 + 2 * anotherVariable;
```
assigns 2 times the value of anotherVariable then adds 5 to the variable sampleVariable. The most difficult feature of C/C++ expressions is what happens when different types are mixed together in one expression (for example, trying to divide integers with floating point numbers). This is explored in some of the following exercises.

**Exercise 2.1 (I/O and Variables)**

This exercise introduces the ideas of simple input and output (I/O), variable types and expressions. We begin with a simple program that appears to perform some simple calculations. However, a little testing shows that the program does not function properly and needs to be improved. Pay attention to the tests that are being made, in the future you will be expected to perform similar tests to make sure that your program is functioning correctly.

1. Create a new program file and enter the following program (Don’t forget to add comments to the beginning of the program containing your name, the date and the assignment number. This is required for all programs.):

```cpp
#include <iostream>

using namespace std;

int main()
{
    int x, y; // declare input variables
    int sum = 0, average; // declare two more variables

    cout << "Please enter a number: " << flush;
    cin >> x;
    cout << "Please enter another number: " << flush;
    cin >> y;
    sum = x + y; // Calculate the sum
    cout << "The sum of " << x << " plus " << y;
    cout << " is " << sum << "."; // Notice how the spaces are placed in the cout statement.
    average = sum/2; // Calculate the average
    cout << "The average of your numbers is ";
    cout << average << "." << endl;

    return 0;
}
```

2. Create a separate file (include your name, the date and the assignment number) to record the answers to the following questions:
• What does the program print for the sum and the average if you enter 5 and 17?
• What does the program print for the sum and the average if you enter 6 and 17?
• What does the program print for the sum and the average if you enter 4.5 and 5.5?

3. Try fixing the program by changing the types of x and y to `double` (or `float`).

4. Repeat the previous questions and record the answers.

5. If the program is still not giving the correct answer for the average of 6 and 17. Fix the program so that it does.

6. Turn in the corrected program and the answers to the questions.

**Exercise 2.2 (Variables and Memory)**

This exercise explores what happens in memory when variables are defined and the size limits of variables. Both of these frequently lead to programming errors.

• Write a program or programs to help answer the following questions (you may use a previous program as a starting point):

1. What happens if you declare a variable of type `int` and print its value before assigning it a value?

2. What happens if you declare a variable of type `float` and print its value before assigning it a value?

3. What is the difference (if any) between the answers for questions 1 and 2 and how would you explain the results?

4. Is it possible to try to store a value that is too large for an integer variable to hold? What happens if you try? (Hint: to test this with code you can write a series of statements like:

```cpp
int largeInt = 10;
cout << largeInt << endl;
largeInt = largeInt*largeInt;
cout << largeInt << endl;
largeInt = largeInt*largeInt;
cout << largeInt << endl;
cout << largeInt << endl;
... repeat the multiplication and output as many times as necessary to get an error
```
and see what happens.)

5. Is it possible to try to store a value that is too large for a float (double) variable to hold? Try it for float first, with an initial value of 1000?

6. Try to find the largest value that an integer variable can correctly hold.

• Record (type) your answers and turn them in. There is no code to turn in for this exercise.