EXPERIMENT NO 2

Fundamentals of Programming

Objectives:

- To learn more about the "cout" and "cin" statement
- To learn about different types of variables

Equipment required:

- Dev-C++/Eclipse/Visual Studio installed in PC/Windows

DISCUSSION

1. Pre-Lab

"cout" Statement

In C++ programming the standard output by default is the screen (output console), and the C++ stream object defined to access it is "cout". For formatted output operations, "cout" is used together with the insertion operator, which is written as << (i.e., two "less than" signs).

cout<< "Output sentence";	// prints Output sentence on screen
cout<< 120;	// prints number 120 on screen
cout<< x;	// prints the value of x on screen

The << operator inserts the data that follows it into the stream that precedes it. In the examples above, it inserted the literal string output sentence, the number 120, and the value of variable x into the standard output stream "cout". Notice that the sentence in the first statement is enclosed in double quotes (") because it is a string literal, while in the last one, x is not. The double quoting is what makes the difference; when the text is enclosed between them, the text is printed literally; when they are not, the text is interpreted as the identifier of a variable, and its value is printed instead. For example, these two sentences have very different results:

cout<< "Hello";	// prints Hello
cout<< Hello;	// prints the content of variable Hello

Multiple insertion operations (<<) may be chained in a single statement:
cout<< "This " << " is a " << "single C++ statement";
This last statement would print the text: **This is a single C++ statement**.
Chaining insertions is especially useful to mix literals and variables in a single statement:
cout<< "I am " << age << " years old and my zipcode is " <<zipcode;

Assuming the age variable contains the value 24 and the zipcode variable contains 90064, the output of the previous statement would be: **I am 24 years old and my zipcode is 90064**

What “cout” does not do automatically is add line breaks at the end, unless instructed to do so. For example, take the following two statements inserting into “cout”:

```
cout<< "This is a sentence.";
cout<< "This is another sentence.";
```

Output: This is a sentence.This is another sentence.

The output would be in a single line, without any line breaks in between as above.
To insert a line break, a new-line character shall be inserted at the exact position the line should be broken. In C++, a new-line character can be specified as \n (i.e., a backslash character followed by a lowercase n). For example:

```
cout<< "First sentence.\n";
cout<< "Second sentence.\nThird sentence.";
```

Output:

```
First sentence
Second sentence
Third sentence
```

Alternatively, the “endl” manipulator can also be used to break lines. For example:

```
cout<< "First sentence." <<endl;
cout<< "Second sentence." <<endl;
```

Output:

```
First sentence
Second sentence
```

The “endl” manipulator produces a newline character, exactly as the insertion of '\n' does; but it also has an additional behavior: the stream's buffer (if any) is flushed, which means that the output is requested to be physically written to the device, if it was not already. This affects mainly fully buffered streams, and “cout” is (generally) not a fully buffered stream. Still, it is generally a good idea to use endl only when flushing the stream would be a feature and '\n' when it would not. Bear in mind that a flushing operation incurs a certain overhead, and on some devices, it may produce a delay.

“cin” Statement

In C++ programming, the standard input by default is the keyboard, and the C++ stream object defined to access it is “cin”. For formatted input operations, “cin” is used together with the extraction operator, which is written as >> (i.e., two "greater than" signs). This operator is then followed by the variable where the extracted data is stored.

For example:

```
int age;
cin>>age;
```

The first statement declares a variable of data type int called age, and the second extracts from “cin” a value to be stored in it. This operation makes the program wait for input from “cin”; generally, this means that the program will wait for the user to enter some sequence with the keyboard. In this case, note that the characters introduced using the keyboard are only transmitted to the program when the “ENTER” or “RETURN” key is pressed. Once the statement with the extraction operation on cin is reached, the program will wait for as long as needed until some input is introduced.
The extraction operation on “cin” uses the type of the variable after the >> operator to determine how it interprets the characters read from the input; if it is an integer, the format expected is a series of digits, if a string a sequence of characters, etc.

```
#include <iostream>
using namespace std;

int main ()
{
int i;
cout<<"Please enter an integer value: ";
cin>>i;
cout<<"The value you entered is "<<i;
cout<<" and its double is "<<i*2
<<".\n";
return 0;
}
```

```
Please enter an integer value: 702
The value you entered is 702 and its double
is 1404.
```

As you can see, extracting from “cin” seems to make the task of getting input from the standard input simple and straightforward. But this method also has a big drawback. What happens in the example above if the user enters something else that cannot be interpreted as an integer? Well, in this case, the extraction operation fails. And this, by default, lets the program continue without setting a value for variable i, producing undetermined results if the value of variable i is used later.
This is very poor program behavior. Most programs are expected to behave in an expected manner no matter what the user types, handling invalid values appropriately. Only very simple programs should rely on values extracted directly from “cin” without further checking. A little later we will see how string streams can be used to have better control over user input.
Extractions on “cin” can also be chained to request more than one datum in a single statement:
cin>> a >> b;

This is equivalent to:
cin>> a;
cin>> b;
In both cases, the user is expected to introduce two values, one for variable a, and another for variable b. Any kind of space is used to separate two consecutive inputs operations; this may either be a space, a tab, or a new-line character.

Variables

A variable provides us with named storage that our programs can manipulate. Each variable in C++ has a specific data type, which determines the size, layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.
The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C++ is case-sensitive language.

Data Type	Size	Description
Int	4 bytes	Stores whole numbers, without decimals. `int Number = 5; // Integer (whole number)`
Float	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 7 decimal digits. `float Number = 5.99; // Floating point number`
Double	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits. `double Number = 9.98; // Floating point number`
Bool	1 byte	Stores true or false values `bool myBoolean = true; // Boolean`
Char	1 byte	Stores a single character/letter/number, or ASCII values. `char myLetter = 'D'; // Character`
String	1 byte / character	Stores a string. `string myText = "Hello"; // String`

2. Post-Lab (Lab Tasks)

1. Write a program that inputs name, age and address from the user and then displays these values on the screen.

2. Write a C++ program that takes two floating point numbers and shows the sum, difference, multiplication, division and modulus result.

3. Write a program that calculates the gravitational force.

$$F = G\left(\frac{m_1 m_2}{r^2}\right)$$

F= Gravitation force of attraction between two objects in newton(N)
G=Universal Gravitational Constant($6.67 * 10^{-11} N - m^2/kg^2$**)**

m1 and m2 = Mass of two objects(Kg)

r= separation in meters(m) between the objects,as measured from their centres of mass.

4. Write a program that inputs time in seconds and converts it into hh-mm-ss format.

5. Write a program that inputs 4 digit number from user and displays it in reverse order.

6. Write a program that inputs two numbers, swap the values and then displays them.

END