Lab Experiment 1

Objectives

Objective(s):
To be familiar with syntax and structure of C++ programming.
To investigate writing programs using control statements including the syntax and the process.

High-Level Language Programs

C++is in contrast to assembly languages that are closely tied to machine languages and therefore are closer to the needs of the hardware.

The process of programming using a high-level language is a sequence of steps that is repeated until we are satisfied that the program meets the specified requirements:

Editing The high-level language program is a text file we create using a **text editor**. Any **text editor** can be used as long as it can store the program as an ASCII file. The file is referred to as the *source file* and contains the *source program*.

Compiling The *source program* is translated into machine language by the compiler. The compiler outputs a file which contains the machine language program. This called an *object file*.

Linking The *object file* is combined with other needed *object files by the linker*. The resulting file is referred to as the *executable file* and contains the *executable program*.

This is an important step that beginning programmers are often unaware of. It often is done automatically by the tool or program that does the compiling, or right before the program executes.

Executing The *executable program* is loaded into the memory of the computer and the instructions in it are carried out. This is also called *running* the program.

Programming Language and Compiler

The high-level programming language used in this lab is C++. In order to write programs in C++ we need a compiler.

Source files in C++ are named with a `.cpp` extension. Object files are named with a `.obj` extension.

One of the best software development skills that you can learn is to start with a minimal program, get it to compile and run, and add to it in small, incremental steps with a new compile and run between each addition.

Comments There are two ways to comment a C++ program. 1) anything between **/*** and ***/** is a comment, and 2) anything after // until the end of the line is a comment

Header Comments Typically there are some comments at the top of the *source file* called the header comments. It is a good habit to put in meaningful header comments including the name of the file, a brief description of what it does, and the name of the person who wrote the program (so you can blame them when it doesn't work).

#include - C++ programs are built from a core programming language, combined with various add-on features. These features are specified in a special statement called a pre-processor directive. The #include <iostream> statement in our first program includes add-on features which allow us to do various input and output operations. This capability is not included in the core C++ language.

main Every C++ program must have a **main** function. The **main** function contains the statements that will be executed by the computer when the program is run. The body of the main function is between the { } braces.

Getting Started

This is the description of the cycle of steps that you will perform for all the projects of this lab.

1. Use the C++ editor and edit the C++ file (file name.*cpp*) or make corrections to your file and save it on your local machine.
2. Compile your file. If compiling the program produces any errors (for now you can ignore the warnings) you will need to fix these before you go on.
3. If the file does not compile repeat step 1 and make corrections, then continue with step 2 until your program is correct and runs properly.

Implementing the Project

There are a couple of small changes to make to this program to change it to a good program that prints out your name.

When a programmer is trying something new, the best thing to do is to take a short example, make small changes to it, and make sure that those small changes worked.

To make sure that it works we need to be able to compile and link the program without any errors. We also need to run the program and make sure that it produces the correct answer (at least for the part that we changed).

If you learn to do this in small steps, you can be successful at programming. If not, then you won't be.

The "I/O"

I/O is shorthand for input and output. For input we just expect that the data got there somehow, and for output we are not very concerned with the format of how it appears.

However, in writing programs we are very concerned about input and output. This includes both what the input and output are and how they appear.

Let’s see how to do input and output and how to use **variables** in a C++ program:

I: Loops with break, continue statements

To understand the programming using Loop & nested loop Statements (for, while, do-while)

Example #1

```
// a pyramid of $ using nested loops
#include <iostream>
using namespace std;

int main()
{
        int i, j;
        cout<<"Let have money pyramid!\n"<<endl;
        // first for loop, set the rows...
        for(i=1; i<=10; i++)
        {
                // second for loop, set the space...
                for(j=1; j<=10-i; j++)
                     cout<<" ";
               // third for loop, print the $ characters...
                for(j=1; j<=2*i-1; j++)
                       // print character...
                        cout<<"$";
                // go to new line...
                cout<<"\n";
        }       return 0;  }
```

Output:

Example #2

```
                    // using break statement in a for structure
#include <iostream>

using namespace std;

int main()
{
      int x; int y=0;
      for(x = 1; x <= 10; x++)
      {y+=2*x;
           // break loop only if x > 5&& y>20
            if (x >5 &&y>20)
               break;
            cout << " X "<<x<<"  Y  "<<y<<endl;
      }
      cout<<"Broke out of loop at x == "<< x<<" and y=  "<<y<<endl;
         return 0;}
```

Output:

Example #3

```
          // using the continue statement in a for structure
#include<iostream>
using namespace std;

// use continue inside loop
 main( )
{ int i,j,k;
          int y=5;    float x;
    for(i= 2;i<4; i++)
    {cout<<"   inside first loop   "<<endl;
cout<<"  i  "<<i<<endl;

    for(j=2;j<6;j=j+2)
    {  cout<<"    inside second loop   "<<endl;
       cout<<"  j   "<<j<<endl;

      k=i/j;  x=float(i)/j;cout<<"  k=  "<<k<<"   x=   "<<x<<endl;
      y+=i+j;
      if(y<20){cout<<"  y  "<<y<<endl;
      cout<<"  use continue  "<<endl;continue;
      // end of if_true
      }

      y+=2; cout<<"continue was not used"<<endl<<" y= "<<y<<endl; }

      // end of internal loop
            }
                       //end of outer loop
                        }
```

Output:

Example #4

```
        // using go to statement in loops


#include<iostream>

using namespace std;


int main()
{   int i, j, k;
            i=1;j=1;k=1;

      label:
  cout<<i<<"  "<<j<<"  "<<k<<endl<<endl; ;
   j=j * i ;
    k=k*(2*i);
    i=i+1;
  cout<<i<<"  "<<j<<"  "<<k<<endl<< endl;

  if (j>20 || k>40)
     cout<<"  goto is not used  "<<endl<<i<<"  "<<j<<"  "<<k<<endl ;
  else
    {  cout<<"  go to is used   "<<endl;
                    goto  label;       }
  return 0;  }
```

Example #5

II. Switch statement

```
// counting letter grades using while, switch and multiple cases
#include <iostream>
 using namespace std;
int main()
{ int EOF;
       char grade;
       int aCount=0, bCount=0, cCount=0, dCount=0, eCount=0, fCount = 0;
       cout<<aCount<<" " <<bCount<<"  "<< cCount<<"  "<< dCount<<"  "<<
eCount<<"  "<< fCount;
       cout<<"Enter the EOF";
      cin>> EOF;
       cout<<" EOF FALSE to end input.\n";
              cout<<"Enter the letter grades. \n";

       while (  EOF)
       {cin>> grade;
             // switch nested in while
              switch(grade)
              {
                     // grade was uppercase A or lowercase a
                     case 'A': case 'a':
                     ++aCount;  break;
                     // grade was uppercase B or lowercase b
                     case 'B': case 'b':
                     ++bCount;  break;
                     // grade was uppercase C or lowercase c
                     case 'C': case 'c':
                     ++cCount;  break;
                     // grade was uppercase D or lowercase d
                     case 'D': case 'd':
                     ++dCount;  break;
                     // grade was uppercase E or lowercase e
                     case 'E': case 'e':
                     ++eCount;  break;
                     // grade was uppercase F or lowercase f
                     case 'F': case 'f':
                     ++fCount;   break;
                     // ignore these input
                       default:
                     {cout<<"Incorrect letter grade entered.\n";
                           cout<<"Enter a new grade.\n";}
```

```
                                  }
     cin>>EOF;                  }       // do the counting...
   cout<< "\nTotals for each letter grade are:\n";

        cout<< "\A:" <<  aCount<<endl;
   cout<< "\B:" <<  bCount<<endl;
   cout<< "\C:" <<  cCount<<endl;
   cout<< "\D:" <<  dCount<<endl;
   cout<< "\E:" <<  eCount<<endl;
   cout<< "\F:" <<  fCount<<endl;
   return 0;}
```

Output:

Example #6

(Students are to code the following programs in the lab and show the output to instructor/course co-ordinator)

Instructions

- *Write comment to make your programs readable.*
- *Use descriptive variables in your programs(Name of the variables should show their purposes)*

Programs List

1. Write a program that tests whether an input positive integer n is prime or not. (A prime number is divisible only by itself and 1, such as 5, 7, 11, etc).

2. Write a program to compute grade of students using **if else adder**. The grades are assigned as followed:

Marks	**Grade**
marks<50	F
50≤marks< 60	C
60≤marks<70	B
70≤marks<80	B+
80≤marks<90	A
90≤mars≤ 100	A+