DHA SUFFA UNIVERSITY
Department of Computer Science
Object Oriented Programming
CS-1002L
Spring 2020

LAB 01

Introduction To C++

Objectives

- To learn about basic input, output, data types,operators, branches, jumps and procedural part of C++

Why C++?

C++ is a general-purpose object-oriented programming (OOP) language, developed by Bjarne Stroustrup, and is an extension of the C language, C++ is considered to be an intermediate-level language, as it encapsulates both high- and low-level language features. Initially, the language was called "C with classes" as it had all the properties of the C language with an additional concept of "classes." However, it was renamed C++ in 1983. It is pronounced "C-plus-plus".

Building Blocks of C++

These are some common and fundamental concepts of C++ introduced below with relevant explanations and examples.

Variables

If the value of an item can be changed in the program then it is a variable. If it will not change then that item is a constant. The various variable types (also called data type) in C/C++ are: int, float, char, long, double etc.

Why do we need Data Types?

Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data-type with which it is declared. Every data type requires a different amount of memory.

The following table shows the variable type, how much memory it takes to store the value in memory, and what is maximum and minimum value which can be stored in such type of variables.

Table 1.0: Primitive Built in Data Types in C++

Type	Keyword
integer	`int`

character	**char**
floating point	**float**
double floating point	**double**
value less	**void**

Table 1.1: For data types and respective ranges in C++

Type	Typical Bit Width	Range
char	1 byte	-127 to 127 or 0 to 255
int	4 bytes	-2147483648 to 2147483647
signed int	4 bytes	-2147483648 to 2147483647
unsigned int	4 bytes	0 to 4294967295
signed long int	4 bytes	-4,294,967,295 to 0
unsigned long int	4 bytes	0 to 4,294,967,295
short int	2 bytes	-32768 to 32767
unsigned short int	2 bytes	0 to 65,535
float	4 bytes	+/- 3.4e +/- 38 (~7 digits)
double	8 bytes	+/- 1.7e +/- 308 (~15 digits)
long	4 bytes	-2,147,483,648to2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Variable Declaration and Initialization

C++ is a strongly-typed language, and requires every variable to be declared with its type before its first use. This informs the compiler the size to reserve in memory for the variable and how to interpret its value.

When the variables in the example above are declared, they have an undetermined value until they are assigned a value for the first time. But it is possible for a variable to have a specific value from the moment it is declared. This is called the initialization of the variable.

```
//variable declaration
int var;
int var1,var2;

//variable declaration and initialization
int var = 5;
or
//Syntax used in c++11 to initialize variable it is used to perform
bound checking so that no data will be lost incase of failure in
matching value of data with data type it will generate an error
int var {5};

//Its best practise to initialize variables while declaration
int var = 10;
```

Operators

Some of the common operators used in C++ and their categories are listed in the table below.

Table 1.3 : Operators and their Categories

Category	Operators
Arithmetic	+ - * / %
Relational	<, > ,<= ,>=, == ,!=
Assignment	= += -= *= /= %=
Bitwise	&,|,^,~,>>,<<

Logical	&&, ||, !

Table 1.4 : Operator Precedence Table

Category	Operator	Precedence
Unary	+ - ! ++ --	right to left
Multiplicative	* / %	left to right
Additive	+ -	left to right
shift	<< >>	left to right
Relational	< <= > >=	left to right
Equality	== !=	left to right
Assignment	= += -= *= /=	right to left
Logical	! && ||	left to right

First Basic Program in C++

```
#include <iostream>
using namespace std;
int main()
{
   //declaration and initialization of variable num
   int num=0;
   //to print on console
   cout << "Please enter a number: ";
   //to take input from user in num
   cin >> num;
   //to display value that user entered
   cout << "The number you entered is: " << num << endl;
   return 0;
}
```

Output

```
D:\Spring 2020\Lab Examples\Lab 1>g++ "Task 1.cpp" -o t1

D:\Spring 2020\Lab Examples\Lab 1>t1
Please enter a number: 10
The number you entered is: 10
```

Example 1.1

```
#include <iostream>
using namespace std;
//Defining main method
int main()
{
      //Declaring variables
       int a=2,b=5,c=7,x=5,y=0;
      //to output on screen
       cout << "Evaluating second degree polynomial"<<endl;
      //Algebraic form
      //y=ax^2+bx+c;
       y=a*x*x+b*x+c;
      //displaying result
      cout << y << endl;
      return 0;
}
```

Output

```
Evaluating second degree polynomial
82
```

Example 1.2

```
#include <iostream>
//Example to show working of bitwise(shift, AND, OR)operators

using namespace std;
int main()
{
```

```
   unsigned int a = 40;       // 40 = 0010 1000
   unsigned int b = 12;       // 12 = 0000 1100
   int c = 0;

   //AND OPERATOR on variable a and b
   c = a & b;              // 8 = 0000 1000
   cout << "Line 1 - Value of c is : " << c << endl ;

   //OR OPERATOR on variable a and b
   c = a | b;              // 44 = 0010 1100
   cout << "Line 2 - Value of c is: " << c << endl ;

   //Left shift OPERATOR on variable a
   c = a << 2;             // 160 = 1010 0000
   cout << "Line 5 - Value of c is: " << c << endl ;

   //Right shift OPERATOR on variable a
   c = a >> 2;             // 10 = 0000 1010
   cout << "Line 6 - Value of c is: " << c << endl ;

   return 0;
}
```

Output

```
Line 1 - Value of c is : 8
Line 2 - Value of c is: 44
Line 5 - Value of c is: 160
Line 6 - Value of c is: 10
```

Escape Sequences

There are many escape sequences in C++, which will give ability to exercise greater control over the way information is output by your program.

Table 1.5 : Escape Sequence

Escape Sequences	Character represented

\n	New Line
\\	Backslash
\'	Single Quotes
\"	Double Quotes

Selection Structures in C++

Normally, your program is sequential but, sometime it is required to perform a particular action if and only if a certain condition is true, i.e. you have to make a decision in your program. There are four major decision making structures which are listed below:

- `if statement`
- `if-else statement`
- `if-else-if`
- `switch case`

The `if` statement

The `if` statement enables you to test for a condition (such as whether two variables are equal) and branch to different parts of your code, depending on the result.

Syntax

```
if (expression)
  Statement;
```

The expression may consist of logical or relational operators like (`> , >=, < ,<= ,&&, ||`)

`if-else` statement

Often your program will want to take one branch if your condition is true, another if it is false.

Syntax

```
if (expression)
  statement;
else
   statement;
```

Example 1.4

```
#include<iostream>
//Using if-else condition to find if number is even or odd.
using namespace std;
int main()
{
         //declaring variables
        int num=0;

        //Printing on screen
        cout << "Enter number: "<<endl;

       //Taking number as input
        cin >> num;

        //To check and print if number is even
        if (num%2==0)
        {
          cout <<  "It is an Even Number";
        }
        //To print if number is not even
        else
        {
          cout <<  "It is an Odd number";
        }

       return 0;
}
```

Output

```
Enter number:
39
It is an Odd number
```

`if-else-if` statement

Often your program will want to take one branch if your condition is true, another if it is false. The keyword else can be used to perform this functionality:

Syntax

```
if (expression)
  statement;
else if(expression)
  statement;
else if(expression)
  statement;
else
  statement;
```

Example 1.5

```
#include <iostream>
using namespace std;
//Defining main method
int main()
{
     //Declaring variables
     int age=0;
     cout <<"Enter your age";
     //Taking input
     cin >> age;
     if(age <= 7)
     {
           cout<<"Congratulations, you have 100% discount!";
     }
     else if(age > 7 && age <= 18)
     {
           cout <<"Congratulations, you have 25% discount!";
     }
     else if (age > 18  && age <= 60)
     {
           cout <<"You can not avail any discount!";
     }
     else
```

```
     {
          cout <<"Avail 100% discount";
     }
     return 0;
}
```

Output

```
Enter your age
19
You can not avail any discount!
```

The `switch` Statement

Unlike if, which evaluates one value, switch statements allow you to branch on any of a number of different values.

Syntax

```
switch(expression)
{
  case 1:
    break;
  case 2:
    break;
  default:
}
```

Example 1.6

```
#include <iostream>
using namespace std;
//Defining main method
int main()
{
      //Declaring variables
      char op='';
      int num1=0,num2=0;
      cout << "Enter First Number: "<<endl;
      //Taking input
      cin >> num1;
```

```
	cout <<"Enter Second Number: "<<endl;
	//Taking input
	cin >> num2;
	cout <<"Enter Operator: "<<endl;
	//Taking input
	cin >> op;
switch(op)
	{
		case '+':
			cout <<"Sum = "<< num1+num2 <<endl;
			break;
		case '-':
			cout <<"Difference = "<< num1-num2 <<endl;
			break;
		case '*':
			cout <<"Product = "<< num1*num2 <<endl;
			break;
		case '/':
			cout <<"Divide = "<< num1/num2 <<endl;
			break;
		default:
			cout <<"Enter Valid Operator";
	}
	return 0;
}
```

Output

```
Enter First Number:
20
Enter Second Number:
40
Enter Operator:
-
Difference = -20
```

Iteration Structures

There are three types of Loops:

- for Loop
- while Loop
- do - while Loop
-

Nesting may extend these loops.

`for` Loop

`for` loop is a conditional iterative statement which is used to execute block of code repeatedly until specific condition is met.It is used when number of iterations are known before entering the loop

Syntax

```
for(initialize; condition; increment)
{
     do this;
}
```

Example 1.7

```
#include <iostream>
using namespace std;
//Defining main method
int main()
{
      //Declaring variables
      int factorial=0,result=1;
      cout <<"Enter Number: ";
      //Taking input
      cin >>factorial;
      for(int i=1;i<=factorial;++i)
      {
      //Calculating factorial
            result*=i;
      }
      //Printing result
      cout <<"Factorial of Number = "<< result;
      return 0;
}
```

Output

```
Enter Number: 3
Factorial of Number = 6
```

The `while` Loop

`while` loop is a conditional iterative statement which is used to execute block of code unknown number of times until the specific condition is met

Syntax

```
while(condition is true)
{
    do this;
}
```

Example 1.8

```
#include <iostream>
using namespace std;
//Defining main method
int main()
{
      //Variables declaration
      int factorial=0,result=1;
      //Printing Message
      cout << "Enter Number: ";
      cin >> factorial;
      int i=1;
      while(i<=factorial)
      {
            result*=i;
            ++i;
      }
      //Displaying result
      cout << "Factorial of Number = " << result;
      return 0;
}
```

Output

```
Enter Number: 4
Factorial of Number = 24
```

This loop will execute as long as the condition in parentheses is true. Note that there is no semicolon after the “while” statement. If there is only one statement in the “while” loop then the braces may be removed

The `do-while` Loop

`do-while` loop is a conditional iterative statement which is used to execute block of code at least once and then executing the block repeatedly until specific condition is met

Syntax

```
do
{
   //code
}
while(condition is true);
```

Example 1.9

```
#include <iostream>
using namespace std;
//Defining main method
int main()
{
      //Declaring and initializing variables
      int factorial=0,result=1;
      cout << "Enter Number: "<<endl;
      //Taking input
      cin >>factorial;
      //Loop initialization
      int i=1;
      do
      {
            result*=i;
            i++;
      }
      while(i<=factorial);
      //Displaying result
      cout << "Factorial of Number = " << result;
      return 0;
}
```

Output

```
Enter Number: 4
Factorial of Number = 24
```

This loop runs as long as the condition in the parentheses is true. Note that there is a semicolon after the "while" statement. The difference between the "while" and the "do-while" statements is that in the "while" loop the test condition is evaluated before the loop is executed, while in the "do" loop the test condition is evaluated after the loop is executed. This implies that statements in a "do" loop are executed at least once. However, the statements in the "while" loop are not necessarily executed.

Jump Statements

Jump statements allow altering the flow of a program by performing jumps to specific locations.

The `break` Statement

`break` leaves a loop, even if the condition for its end is not fulfilled. It can be used to end an infinite loop, or to force it to end before its natural end. In example 1.9, let's stop the countdown before its natural end.

Example 1.10

```
#include <iostream>
using namespace std;
//Defining main method
//use of break statement to end loop before its natural end
int main()
{
   int j=0;
   for (j=10; j>0; j--)
   {
      cout << j << ", ";
      if (j==4)
      {
        cout << "countdown aborted!";
        break;
      }
   }
   return 0;
}
```

Output

```
10 , 9 , 8 , 7 , 6 , 5 , 4 ,  countdown aborted!
```

The `continue` Statement

The `continue` statement causes the program to skip the rest of the loop in the current iteration, as if the end of the statement block had been reached, causing it to jump to the start of the following iteration. Below it is explained with an example.

Example 1.11

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

//Defining main method
int main()
{
    int i=0;
    // loop from 1 to 10
    for (i = 1; i <= 10; i++)
    {
        // If i is equals to 6,
        // continue to next iteration
        // without printing
        if (i == 5)
        {
            continue;
        }
        else
        {
            // otherwise print the value of i
            cout << i << " ";
        }
    }
    return 0;
}
```

Output

```
1 2 3 4 6 7 8 9 10
```

Lab Tasks

Task 1.1

Write a program in C++ to generate the following output. Take the score of the first quiz , second quiz , third quiz , midterm and final term as input. If the total score is 90% or above, Indicate that the student is eligible for a scholarship.

```
                    ============Result Sheet===============

Enter the score of the first quiz: 90

Enter the score of the second quiz: 75

Enter the score of the third quiz: 91

 =======Mid-term==============

Enter the score of the mid-term: 80

=======Final=================

Enter the score of the final: 89

Quiz Total:     256

Mid-term :      80

Final       :   89

……................

Total:          425
```

Task 1.2

Write a Program that takes integer numbers from 1 to n as an input and prints sum of odd numbers in this range.

Task 1.3

Write a program that takes two integers x & y as input and finds & displays their product without using '*' operator.

Task 1.4

Write a program in C++ to calculate the sum of the series (1*1) + (2*2) + (3*3) + (4*4) + (5*5) + ... + (n*n).

```
Sample Output:
Input the value for nth term: 5
1*1 = 1
2*2 = 4
3*3 = 9
4*4 = 16
5*5 = 25
The sum of the above series is: 55
```

Task 1.5 (HOME ASSIGNMENT)

Write a program to perform following:

1. Ask users to input the basic salary of an employee.
2. calculate gross salary according to given conditions:

In the end add a 5% bonus in basic salary if the service of an employee is more than 5 years. Take value of the year as an input.

Formula For gross salary: Basic salary + DA + HRA
DA = basic salary * % of DA according to the conditions given in the table.
HRA = basic salary * % of HRA according to the conditions given in the table.

Salary	DA	HRA
Basic Salary <= 10000	25%	80%
Basic Salary is between 10001 to 20000	25%	90%

Basic Salary >= 20001	50%	97%

Task 1.6 (HOME ASSIGNMENT)

Write a program to withdraw a amount from ATM.Ask user to input current balance,withdrawal amount,available balance in ATM and actual pin of Account.Apply following checks for the withdrawal of money

1) Verify pincode of the account by checking actual pin code with entered pin code
2) Amount to be withdrawn should not exceed the amount of available balance
3) Amount to be withdrawn should not exceed 25k
4) Amount to be withdrawn should not exceed the available balance in ATM

Submission Instructions

- Name your Task Files as T1,T2, etc (example : T1.cpp)
- Name your home assignment as Lab_0X .cpp (X is the lab number)
- Create a new folder named cs152abc where abc is your 3 digit roll #. e.g. cs152111.
- Copy all the task and assignment files into this folder.
- Now make a zip file of above made folder named cs152abc.zip is created e.g. cs152111.zip
- Upload this zipped folder on LMS under the assignment named Lab 01_Assignment – XX, where XX is your section name.