Lecture 1

1- C++ Programming Language:

For the last couple of decades, the C programming language has been widely accepted for all applications, and is perhaps the most powerful of structured programming languages. Now, C++ has the status of a structured programming language with object oriented programming (OOP).

C++ has become quite popular due to the following reasons:

1. It supports all features of both structured programming and OOP.

2. C++ focuses on function and class templates for handling data types.

2-Getting Ready to Program

Programs are written to instruct machines to carry out specific tasks or to solve specific problems. A step-by-step procedure that accomplishes a desired task is called an *algorithm*. Thus, programming is the activity of communicating algorithms to computers. The programming process is analogous, except that machines have no tolerance for ambiguity and must have all steps specified in a precise language and in tedious detail.

The Programming Process

1. Specify the task.
2. Discover an algorithm for its solution.
3. Code the algorithm in C++.
4. Test the code.

3-A First Program

First task for anyone learning to program is to print on the screen. Let us begin by writing the traditional first C++ program which prints the phrase *Hello, world!* On the screen. The complete program is

```
#include <iostream>
#include <conio>
 main ()
{
cout << "Hello World!";
getch();
}
```

Ex/write a program to print the following message on the screen .

C++ is a programming language

C++,based on the C programming

Sol ||

```
#include <iostream>
#include <conio>
 main ()
{
cout << " C++ is a programming language\n";
cout << " C++ based on the C programming " ;
getch();}
```

Ex:-Write program to read two numbers and then calculate the sum of square each number and the sum of cubic each number.
Sol//

```
#include <iostream>

#include <conio>

main()

{

int x ,y,z,w;

cout<<"enter two numbers" <<endl;;

cin>>x>>y;

z=x*x+y*y;

w=x*x*x+y*y*y;

cout << "z= " << z << endl;

cout << "w= " << w <<endl;

getch();

}
```

Ex :- Write a program to read the number x and then find the value of y from the following equation: $y = 3x^3 - 8x^2 - 7x + 8$.

Sol//

```
#include <iostream>

#include <conio>

main()

{

int x ,y;

cout<<"enter number" <<endl;;
```

```
cin>>x;
y=3*x*x*x- 8*x*x-7*x+8;
cout << "y= " << y << endl;
getch();
}
```

4- Control flow introduction

When a program is run, the CPU begins execution at the top of main(), executes some number of statements, and then terminates at the end of main(). The sequence of statements that the CPU executes is called the program's **path**. Most of the programs you have seen so far have been **straight-line programs**. Straight-line programs have **sequential flow** — that is, they take the same path (execute the same statements) every time they are run (even if the user input changes).

However, often this is not what we desire. For example, if we ask the user to make a selection, and the user enters an invalid choice, ideally we'd like to ask the user to make another choice. This is not possible in a straight-line program. Fortunately, C++ provides **control flow statements** (also called *flow control statements*), which allow the programmer to change the CPU's path through the program. There are quite a few different types of control flow statements, so we will cover them briefly here, and then in more detail throughout the rest of the section

5-Conditional branches

A **conditional branch** is a statement that causes the program to change the path of execution based on the value of an expression.

If statements

The most basic kind of conditional branch in C++ is the **if statement**. An *if statement* takes the form:

```
if (expression)
```

statement

or

if (expression)

statement

else

statement2

Here is a simple program that uses an if statement to check then number is smallest or largest then 10 number:

```
#include <iostream>
#include <conio>
 main ()
{
cout << "Enter a number: ";
int nX;
cin >> nX;
if (nX > 10)
cout << nX << "is greater than 10" << endl;
else
cout << nX << "is not greater than 10" << endl;
getch();
}
```

Nested if

It is also possible to nest if statements within other if statements:

Here is a simple program to input the grade of the student then check it and output the result.

Grade Result

G<50 Failed

G<60 Acceptance

G<70 median

G<80 good

G<90 Very good

G<=100 Excellent

The switch Statement

The switch statement is a multiway conditional statement generalizing the if-else statement. The general form of the switch statement is given by switch (expression)

{

case constant1:

group of statements 1;

break;

case constant2:

group of statements 2;

break;

.

.

.

default:

default group of statements

}

Loops

A **loop** causes the program to repeatedly execute a series of statements until a given condition is false.

for loop

Its main function is to repeat statement while condition remains true, like the while loop. But in addition, the for loop provides specific locations to contain an initialization statement and an increase statement. So this loop is specially designed to perform a repetitive action with a counter which is initialized and increased on each iteration.

Its format is:

for (initialization; condition; increase) statement

Here is an example of print numbers between 1 to 10 using a for loop

```
#include <iostream>
#include <conio>
 main ()
{
for (int i = 1; i <= 10; i++)
cout<<i<<"," ;
getch();
}
```

Ex:- Write C++ program to print the following:

```
#include <iostream>
#include <conio>
void main( )
{
int x;
```

```
1   10
2   9
3   8
4   7
5   6
6   5
```

```
for ( x = 1; x < 7; ++ x )
cout << x <<"\t" << 11 - x << endl;
getch();
}
```

The while Statement

The general form of a while statement is

while (*condition*)

statement

First, *condition* is evaluated. If it is true, *statement* is executed, and control passes back to the beginning of the while loop. The result: The body of the while loop, namely, *statement*, is executed repeatedly until *condition* is false. At that point, control passes to the next statement. In this way, *statement* can be executed zero or more times.

Ex2:- Write C++ program to find the cub of a number, while it is positive:

```
#include<iostream>
#include<conio>
void main( )
{
int num, cubenum;
cout << "Enter positive number \n";
cin >> num;
while ( num > 0 )
{
cubenum = num * num * num;
```

```
cout << "cube number is :" << cubenum << endl;
cin >> num;
getch();
}
}
```