Computer Programming I

Communication Engineering Department

Lecturer 1

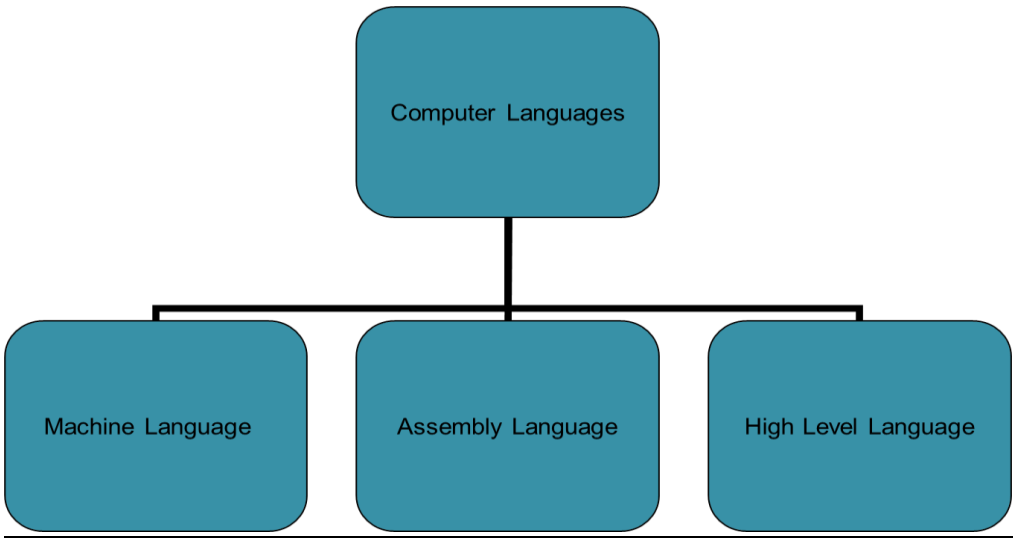
Flowchart and Pseudo-code

- Computer Language
- Algorithm and Flowchart
- Rules for constructing an Algorithm

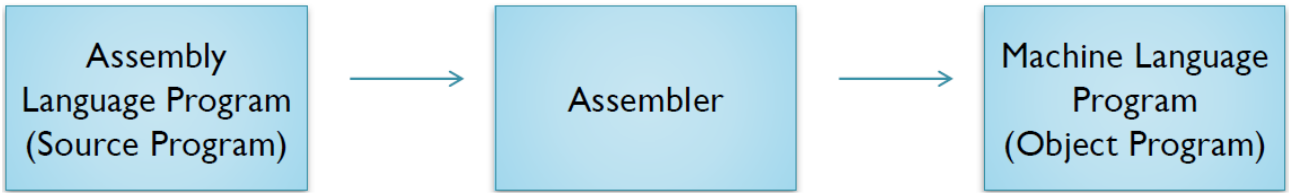
Computer Language

Means of communication used to communicate between people and the computer

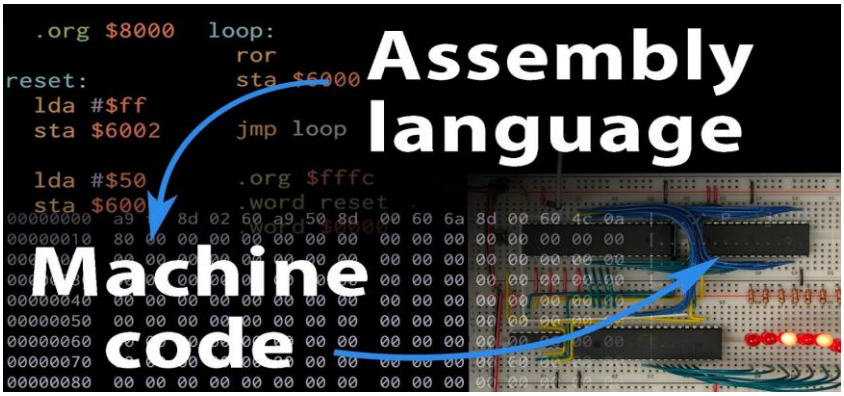
Classification of Computer Languages



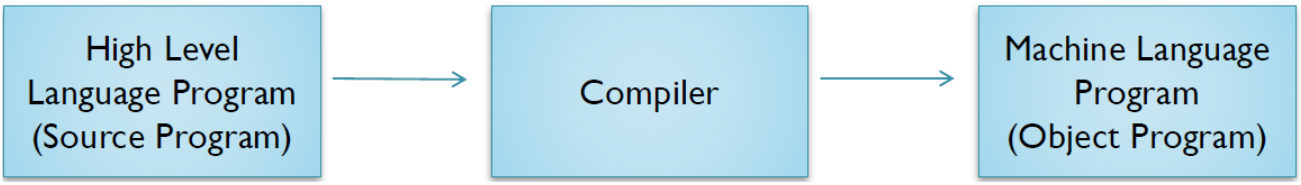
- ✓ **Machine language:** These language instructions are directly executed by CPU.
- ✓ **Assembly language:** The endeavor of giving machine language instructions a name structure that means bit strings of instructions of machine language are given name here



Assembler is needed to convert assembly language into machine language.



✓ **High Level Language:** The user-friendly language ...more natural language than assembly language.



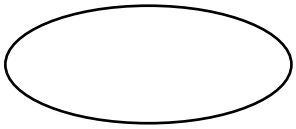
Compiler is needed to convert high level to machine language.

Algorithm and Flowchart

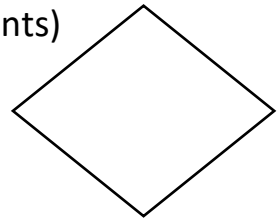
Before we study C++ in details, we need to understand the flow of a program and how to analyze a given problem before writing its code. There are two useful tools one may apply called the algorithm and flowchart.

Algorithm and flowcharts are two different tools used for creating new programs, especially in computer programming. An **algorithm** is a step-by-step analysis of the process, while a **flowchart** explains the steps of a program in a graphical way. Algorithms can be presented by natural languages, pseudo code and flowcharts.

Several standard graphics are applied in a flowchart as following:

- Terminal Box - Start / End 
- Input / Output 
- Process / Instruction 

- Decision (conditional statements)



- Connector / Arrow

Rules for constructing an Algorithm:

- ✓ Input: There should be zero or more values which are to be supplied.
- ✓ Output: At least one result is to be produced.
- ✓ Definiteness: Each step must be clear and unambiguous.
- ✓ Finiteness: If we trace the steps of an algorithm, then for all cases, the algorithm must terminate after a finite number of steps.
- ✓ Effectiveness: Each step must be sufficiently basic that a person using only paper and pencil can in principle carry it out. In addition, not only each step is definite, it must also be feasible.
- ✓ Comment Session: Comment is additional info of program for easily modification. In algorithm comment would be appear between two square bracket [ ].

Rules of Drawing Flowcharts for Algorithms:

- All boxes of flowcharts are connected with arrows to show the logical connection between them
- Flowcharts will ow from top to bottom
- All flowcharts start with a start box(ellipse) and end with a terminal box(ellipse).

Example: Write algorithm to calculate the sum and average of two numbers.

Algorithm:

[procedure for calculate sum and average of two numbers]

Step 1: Start
Step 2: Read two numbers n, m
Step 3: Calculate sum=n + m
Step 4: Calculate avg= sum/2
Step 5: Print sum, avg
Step 6: End

[End of procedure for calculate sum and average of two numbers]

Example:

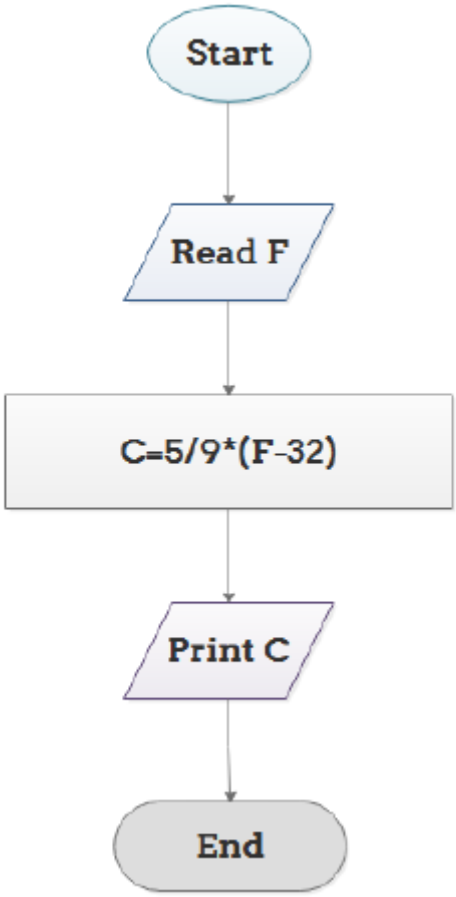
Convert temperature from Fahrenheit °F to Celsius °C by using the formula C = 5/9 * (F - 32).

Algorithm:

[Procedure for Converting Temperature from Fahrenheit °F to Celsius °C]

Step 1: Start,
Step 2: Read temperature in Fahrenheit,
Step 3: Calculate temperature with formula C=5/9*(F-32),
Step 4: Print C
Step 5: End

Flowchart:

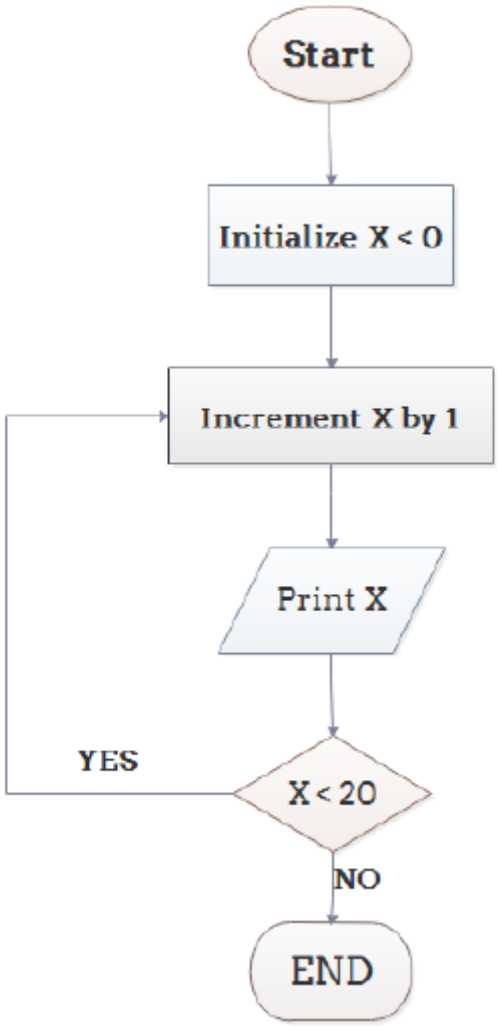


Example:
Let x be a fixed negative integer. Print the integer values in the interval (x,21).

Algorithm:

Step 1: Start,
Step 2: Initialize integer x < 0,
Step 3: Increment x by 1 (x = x + 1),
Step 4: Print x,
Step 5: If x is less than 20 then go back to step 3,
Step 6: End.

Flowchart:



Computer Programming I

Communication Engineering Department

Lecturer 2

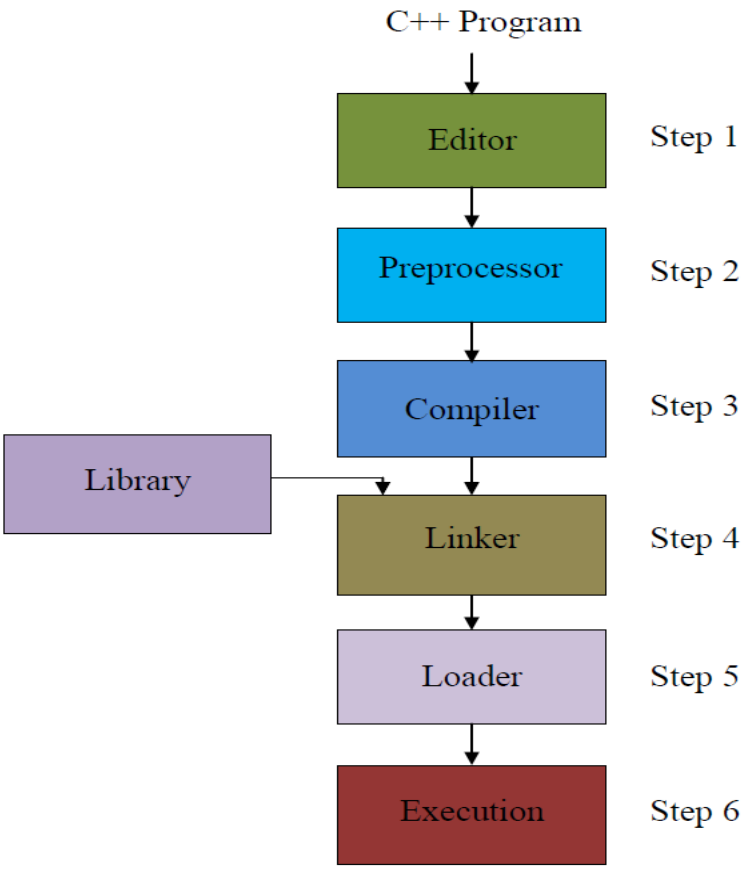
Introduction of computer C++ programming
Arithmetic operation

- **Data types**
- **Variables**
- **Constants**
- **Input statement**
- **Initializing variables in declarations**
- **Assignment statement**
- **Output statement**
- **Basic program construction**

A program is a set of instructions in proper sequence, that causes a computer to perform a particular task.
The following are the general steps to execute a C++ program

1. **Editor**
   The source code is written using a text editor.
2. **Preprocessor**
   It processes directives that start with # (such as #include and #define).
3. **Compiler**
   The compiler converts the source code into machine language, creating an object file.
4. **Linker**
   It links the object file with the required libraries (like I/O libraries) to form the executable program.
5. **Loader**
   The loader loads the executable into the main memory, preparing it for execution.
6. **Execution**
   Finally, the program is executed by the CPU.

Library:
The library contains ready-to-use functions and code that might be used by the program during execution.



Data types

Data type is a type of container that can hold a specific kind of program data. C++ language permits the use of several basic data types

Data type	Meaning	Size
Char	Character	8 bit
Int	Integer	16 bit
long	Long integer	32 bit
float	Floating point	16 bit
double	Double floating point	32 bit

- ✓ **Integer data type (int)**
  Integers are whole numbers without a fractional part. (zero and negative whole numbers are integers)
- ✓ **Long integer type (long)**
  It is the same as the integer, except it reserves twice the space for the regular integer.
- ✓ **Floating point data types**
  The data types that are used to represent real numbers (numbers with fractional parts) are **float** and **double**. The difference between these data types lies in the maximum number of significant digits that is, the number of decimal places in float values is 6 or 7 while the maximum number of significant digits in values belonging to the double is 15. This means that a double can store a number approximately ten times larger than float.
- ✓ **Character data type (char)**
  char is the smallest data type; it occupies only 1 byte (eight bits) of memory. The **char** data type is used to represent ASCII characters. When using the char data type, you enclose each character (letter, digit or special symbol) with single quotation marks. Examples: 'A', 'a', '0', '+', ' ', '$'. Each of the 128 values of the ASCII character set represents a different character. For example, the value 65 represents 'A', the value 48 represents '0' and the value 97 represents 'a'.
- ✓ **String data type**
  The **string** is a collection of characters, numbers or special symbols **enclosed in double quotation marks**. Examples: "Hello!", "1+2=", "a-b=", "what is your name?"

Variables

The variable is a **memory location** whose **content may change** during program execution.

The general form of a variable declaration is:

```
data type  variable name;
```

NOTE: every C++ statement must end with a semicolon (;).

Examples:

int number**;**

char ch;

string name;

When there is more than one variable in a declaration, the variables are separated by **comma** as illustrated in the following declaration statement:

float a, b, c**;**
int first, middle, last;
There are several rules to name a variable.

- Names may contain letters (A to Z, a to z), numbers (0 to 9), or underscores (_).
- The first character must be a letter or an underscore.
- Names cannot contain any symbols, such as - ! @ # $ % ^ & * ( ) " + = | ' \ , nor can they have any spaces.

Constants

The constant is a memory location whose content is not allowed to change during program execution.

The syntax to declare a constant is:

```
const data type  constant name = value;
```

Examples:

const int number = 55;

const char letter= 'a';

const string name = "mohammed";

❖ Input statement

The first way to put data into a variable is to use the standard input device (usually the keyboard), this is accomplished via the use of **cin** and the operator >>. The syntax of **cin** and the operator >> is:

cin>>variable$_1$>>variable$_2$ ….;

Examples:

cin>> number**;**

cin>> ch;

cin>>name;

The **cin** statement can be used to enter more than one value of the **same data type**, it can also be used to enter multiple values of **different data types** as shown in the following **examples**:

1.
string name1, name2;
cin>>name1>>name2;
2.
char a;
double b;
cin>>a>>b;

Initializing variables in declarations

The second way to put data into a variable is to **initialize** the variable (that is, **give it a value**) at the time that you declare the variable.

Examples:
int cont = 0;
int a=3, b, c=5, d;
char letter = 'm';
string s= "welcome to C++";
Notice that, in the second declaration statement only the variables **a** and **c** are initialized.

Assignment statement

```
variable = expression;
```

Examples:

z= 3*x +5;

a= b;

x= m+n;

y=9;

Output statement

In C++, output on the standard output device which is usually the screen is accomplished via the use of **cout** and the operator <<. The general syntax of **cout** together with << is:

```
cout<<variable or expression or string or manipulator<<variable or expression or string
or manipulator,…;
```

Examples:

cout<<z;

cout<<2+3;

cout<<x+y-5;

cout<<"x="<<x<<endl;

cout<<"Hello!";

Basic program construction

```
#include <iostream>          // تضمين مكتبة الإدخال والإخراج
using namespace std;         // استخدام فضاء الأسماء القياسي

int main()                   // الدالة الرئيسية التي يبدأ منها تنفيذ البرنامج
{
    // تعليمات البرنامج تكتب هنا

    return 0;                // إنهاء البرنامج بنجاح
}
```

Example 1: write a program to print "Hello World"

```
#include <iostream>          // تضمين مكتبة الإدخال والإخراج
using namespace std;         // استخدام فضاء الأسماء القياسي

int main()                   // الدالة الرئيسية التي يبدأ منها تنفيذ البرنامج
{
    cout << "Hello World!" << endl;  // طباعة النص على الشاشة متبوعًا بسطر جديد
    return 0;                // إنهاء البرنامج بنجاح
}
```

Example 2: write a program to calculate the area of a rectangle.

```
#include <iostream>
Using namespace std;
int main ()
{
int length, width, area;
cout << "enter the length: ";
cin >> length;
cout << "enter the width: ";
cin >> width;
area = length*width;
cout << "the area of the rectangle is: "<< area;
return 0;
}
```

Program output

```
enter the length: 2
enter the width: 7
the area of the rectangle is: 14
```

Homework:

1. **Program to Add Two Numbers and Display the Result**
2. **Program to Convert Fahrenheit to Celsius**

Computer Programming I

Communication Engineering Department

Lecturer 3

Conditional Statements in C++

- **Introduction**
- **if Statement**
- **if-else Statement**
- **if-else-if ladder Statement**

Introduction:

Boolean (bool): represent logic values.
Values: false and true.
Operators: not, and, or.
The logical operators are often used to combine relational expressions into more complicated Boolean expressions:

operator	meaning
&&	and
||	or
!	not

Arithmetic operators: +, -, *, /

operator	meaning
==	equal to
! =	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

In C++ programming, if statement is a logical expression and it is used to test a certain condition. There are various types of if statements in C++ as follows:

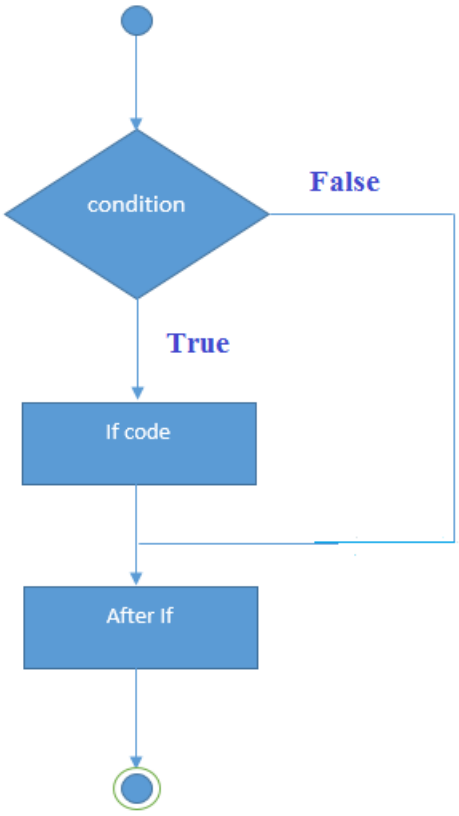
- if statement
- if-else statement
- if-else-if ladder

if Statement

The C++ if statement tests the condition. It is executed if condition is true otherwise the next step after the condition shall be executed.

```
if(condition)
{
//the executed code
}
```

if Statement flowchart



- **Condition**: A logical (Boolean) expression that results in either **true** or **false**.
- If the condition is **true**, the code inside {} will run.
- If it is **false**, the code inside {} will be skipped.

Example 1:

Check if a real number x is greater than 5.

```
#include<iostream>
using namespace std;
int main () {
        cout<<"Enter a real number: ";
        float num;
        cin >> num;
        if (num > 5)
        {
        cout << num << "is greater than 5";
        }
return 0;
}
```

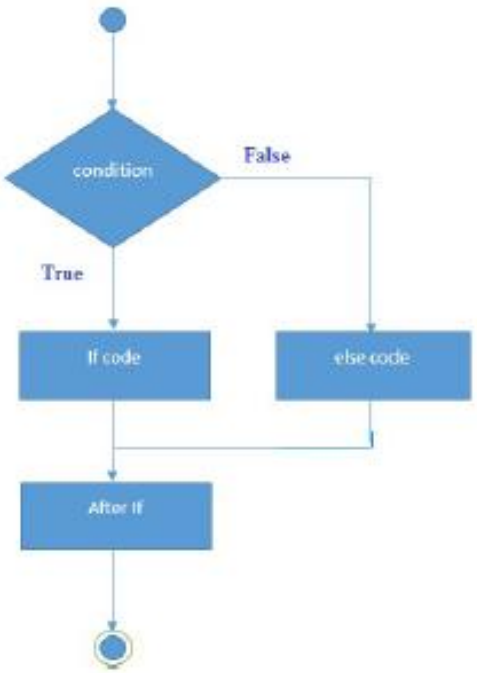
The output: suppose that you enter x=6.5

```
Enter a real number: 6.5
6.5 is greater than 5
```

if-else Statement

```
if(condition)
{
//code if condition is true
}
else
{
//code if condition is false
}
```

if-else Statement flowchart



Example 2:

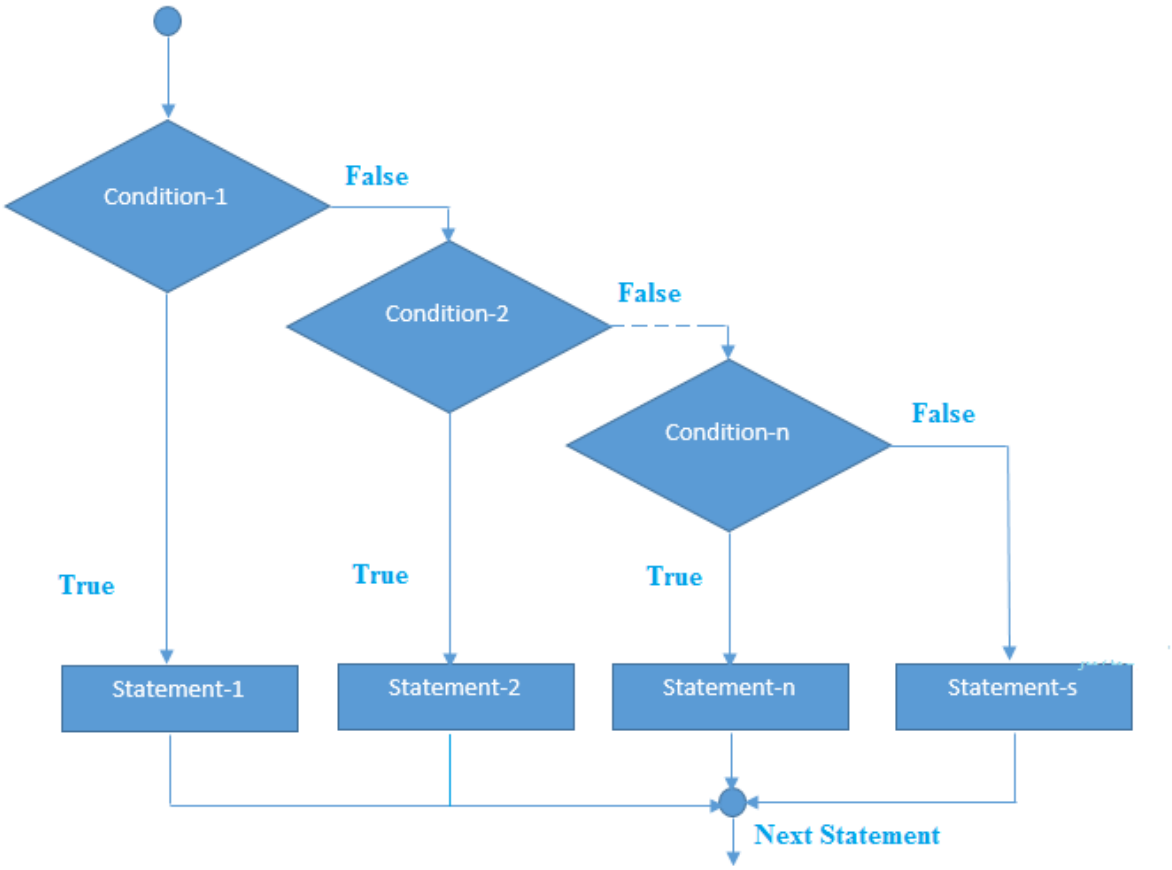
Check whether a number is even or odd.

```
#include<iostream>
using namespace std; //A code for checking whether a number
is even or odd
int main () {
      cout<<"Enter a number: ";
      int num;
      cin >> num;
      if (num % 2 == 0) {
         cout << num << "is even number"; }
      else {
         cout<< num <<" is odd number"<<endl; }
return 0; }
```

- **if-else-if ladder Statement**

```
if(condition 1){
//code to be executed if condition1 is true }
else if(condition 2){
//code to be executed if condition2 is true }
else if(condition 3){
//code to be executed if condition3 is true }
...
else{
//code to be executed if all the conditions are false }
```

if-else-if ladder Statement flowchart



Example: Check whether a student's grade is fail, accepted, middle, good, very good, or excellent.

```
#include <iostream>
using namespace std;
int main () {
        int num;
        cout<<"Enter a number to check grade:";
        cin>>num;
        if (num < 0 ||  num > 100) { cout<<"wrong number"; }
        else if(num >= 0 && num < 50){ cout<< "Fail"; }
        else if (num >= 50 &&  num < 60){cout<< "accepted";}
        else if (num >= 60 &&  num < 70){cout<< "middle"; }
        else if (num >= 70 &&  num < 80) { cout<< "good"; }
        else if (num >= 80 &&  num < 90)
                { cout<< "very good"; }
        else                                {cout<< "excellent";
}
return 0; }
```

Computer Programming I

Communication Engineering Department

Lecturer 4

C++ Loops

- **The C++ For Loop**
- **While loop**
- **Do - While loop**

Loops are used to repeat a block of code. Being able to have your program repeatedly execute a block of code is one of the most basic but useful tasks in programming. There are three types of loops:

- ✓ for
- ✓ while
- ✓ do- while

Each of them has their specific uses.

The C++ For Loop

The C++ for loop is used to iterate a part of the program several times. The syntax for a for loop is given by

```
for(initialization; condition; incr/decr){
//code to be executed
}
```

It executes initial statement once, the test takes place before each iteration, then executes statement and iteration expression repeatedly, until the value of condition becomes false.

Example: Print the integer numbers from 1 to 10.

```
#include<iostream>
using namespace std;
int main ( ) {
      for(int i=1; i<=10; i++){
      cout<< i <<"\ n";
          }
return 0; }
```

If we replace the statement

cout << i << "\ n";

by

cout << i << " "; output????

While loop

In C++, while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop than for loop. The syntax of while loops is given by

```
while(condition){
//code to be executed
}
```

Example: Print the real numbers from 1 to 2 by 0.2 increment using while loop.

```
#include<iostream>
using namespace std;
int main ( ) {
      float x=1.0;
        while(x<=2){
           cout<<x<<" ";
           x=x+0.2; }
return 0; }
```

Output:

```
1 1.2 1.4 1.6 1.8
```

if we set the step cout << x << " "; after the step x = x + 0:2;

then the output is

```
1 1.2 1.4 1.6 1.8 2
```

Do - While loop

Do-While loop the C++ do-while loop is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop. The C++ do-while loop is executed at least once because condition is checked after loop body. The syntax of do-while loops is given by

```
do{
//code to be executed
}
while(condition);
```

Example: print the numbers from 1 to 10 using do-while loop.

```
#include<iostream>
using namespace std;
int main ( ) {
      int i=1;
        do{ cout<<i<<" ";
          i++;}
        while(i<=10);
return 0; }
```

Output:

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

Homework:

write a program in C++ using (for, while) to:

1- Print numbers from 1 to 10.
2- Print even numbers from 2 to 10.
3- Find the Sum of numbers from 1 to 10.