Unit 1: Introduction to C Language

Module.1: Introduction

CONTENTS

Learning Objectives

1. To understand fundamentals of C language
2. To know structure of C program
3. To develop algorithm/flowcharts
4. To know applications of C language

1. Introduction

Programming in C is one of the most important papers for Electronic Science subject at post graduate level. Every student planning to make career in Electronics Science should have some kind of programming skills. One may ask a question like "***What are the different computer languages an electronics student must learn for his career***?" It is like asking a question "How many cars a driver should practice to learn driving? Practice driving only one car. It is sufficient. Similarly, mastering one programming language is enough. Learn the better implementation strategies in programming.

Overall, in the language learning what is most important is to have deep knowledge about what to tell computer to do and follow the different implementation strategies like how to write algorithm, how to work with computer memory and operating system effectively. Ultimately, you should learn only as much as required to get solution to a problem. Remember that ***language is not the solution but it is a tool to solve the problem***. This course is designed to teach you how to program in C. It assumes no previous background of any programming language. In this module, the focus is on understanding the fundamentals of computer programming and general introduction to C language.

2. Fundamentals of computer programming

Computers are really dumb machines because they do only what they are asked to do. Many computers perform operations on a very basic level like addition/subtraction of numbers etc. These basic operations are decided by the instruction set of the computer. To solve a problem using a computer, it is necessary to express the method to solve the problem using the primitive instructions of the computer. A sequence of instructions given to the computer is known as program. The method or strategy used for solving the problem is known as algorithm. With the algorithm in hand, it is possible to write the instructions necessary to implement the algorithm. Programs are developed to solve some problem. These programs can be written using some computer language like Basic, C, C++, Java etc.

2.1 Language levels

When computers were first developed, the only way to program them was in terms of ***binary numbers***. These binary numbers are corresponds to machine instructions and referred to as ***Machine language*** and are stored in computer's memory. The next step in programming was the development of ***assembly languages***. This has enabled the programmer to work with the computers on a slightly higher level. Instead of specifying the sequences of binary numbers to carry out particular tasks, the assembly language permits the programmer to use symbolic names to perform various operations. An **assembler** translates the assembly language program into the machine instructions i.e in binary.

Table-1: World of programming languages.

High level languages	Ada
	Modula-2
	Pascal
	COBOL
	FORTRAN
	BASIC
Middle level languages	Java
	C++
	C
	FORTH
	Macro-assembler
Low level language	Assembler

The machine language and the assembly language is popularly known as **Low level language**. The programmer must learn the instruction set of the particular computer system to write a program in assembly language, and the program is not portable. The program will not run on a different processor. This is because different processors have different instruction sets. As these assembly language programs are written in terms of seperate instruction sets, they are **machine dependent**.

For writing the program once independent of the processor, it was necessary to standardize the syntax of a language. Higher-level language was introduced so that a program could be written in the language to be **machine independent**. That is, a program could run on any machine that supported the language with few or no changes. To execute a higher-level language, a special computer program must be developed that translates the statements of the higher-level language into a form that the computer can understand. This program is known as a **compiler**. Table -1 indicates the position of C in programming languages.

2.2 Operating System

Operating system is program that controls the entire operations of computer system. Access to all the resource of the computer e.g. memory and input/output devices is provided through the operating system. This program looks after the file management, memory management and I/O management. It is a program that allows all other program to use the computing capability of the computer and execute them. The most popular operating systems are Windows, Linux, Unix, Mac OS X, MSDOS etc.

2.3 Program Execution

Before writing a computer program one must be clear about the steps to be performed by the computer for processing it. To produce an effective computer program it is necessary that every instruction is written in the proper sequence. For this reason, the program must be planned before writing. An algorithm represents the logic of the program. When an algorithm is expressed in the programming language, it becomes the program.

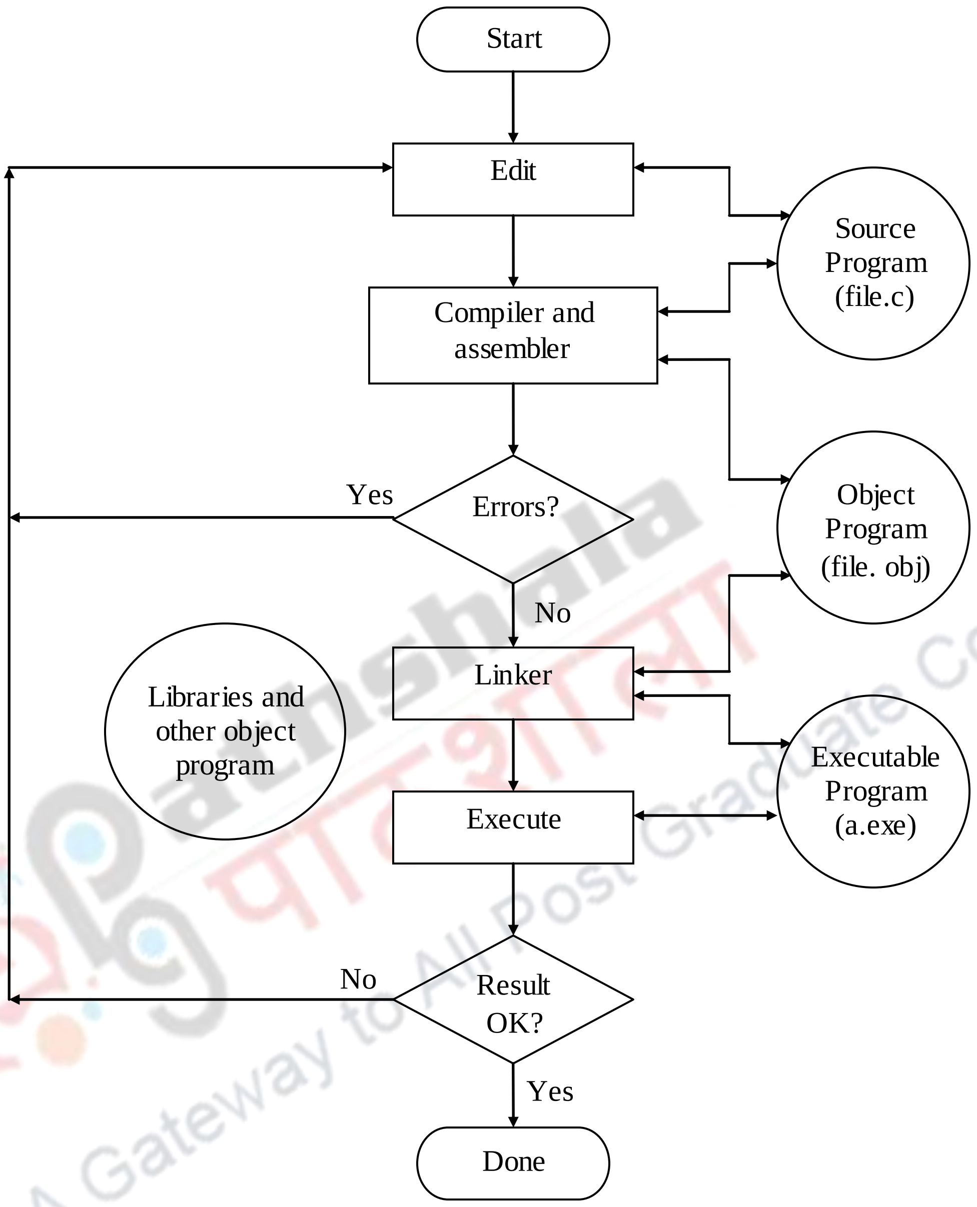


Figure 1: Program development stages

A compiler is simply a program developed in a particular computer language usually a higher level language and then translates it into a form that is suitable for execution on a particular computer system. Fig. 1 indicates the different stages right form program coding to execution,

Initially, the **text editor** will help the programmer to enter the C program and store it as file with and extension ".c". The program written by the user with the help of text editor is known as the source program. Every source program must have a valid filename e.g. “file.c”. Once the source program is entered into a file then one can proceed to the compilation,

To start the compilation process, the file containing the source C program must be specified. Compiler examines each program statement present in the source programs for correct syntax and semantics of the language. If there are any mistakes or errors detected by the compiler then these are reported to the user and compilation process is terminated. The errors must be removed from the source program and then compilation process may be reinitiated.

Once all the syntax error (e.g. missing parentheses) or semantic error (e.g. variable not defined) have been removed then compiler restarts it process. Compiler takes the statements of the program and translates it into lower language. After the program has been translated into lower level language like assembly language, the next stage is to translate these assembly instructions into machine language. In many compilers, this stage is automatically executed along with the compilation process. The result of this compilation process is the generation of the object file from the given source file. This object file has an extension “.obj” and contains the object code. The filename is just same as source filename but with different extension name.

After the program is converted into object code, the **linker** automatically starts the process. The purpose of linker phase is to link the present file with any previously compiled file and system’s library. At the end of linking the system generates a file with the same source filename but with extension “.exe”. Once executable file is generated then the program is ready to execute or run. Computer system execute each program statement sequentially. If any external data is required then user must input it. Once the processing is over the result is displayed as the output of the program.

If the desired results are obtained the complete process is over else go back to the editor and check for the logical error. At this state, the built in debugger can help programmer to remove the bugs from the program. In this case the entire processor of editing, compiling, linking and executing the program. Usually, the process of editing, compilation, executing and debugging is managed under one umbrella by a single Integrated Development Environment (IDE).

3. Introduction to C language

Before learning how to write programs in C it would be important to know a brief history, features, structure and applications of C language. In this section, the attempt is made to present this information in short.

3.1 History of C Language

The origin of C is closed related with the development of UNIX operating system for PDP-7 computers. UNIX operating system was originally written by Ritchie and Thompson using **assembly language**. Even for PDP-11, the operating system was developed using assembly language. Developers were planning to rewrite OS using the **B language** using the Thompson's simplified version of BCPL (Basic Combined Programming Language).

The development of C language started in 1972 on PDP-11 UNIX system by Dennis Ritchie. The name of C language simply continued the alphabetic order. C is the result of a development process that started with an older language called BCPL. It was developed by Martin Richards, and it influenced a language called B, which was implemented by Ken Thompson. B language led to the development of C language in the 1970s.

In the summer of 1983 a committee was established to create an ANSI (American National Standards Institute) standard that would define the C language. This process took much time but the ANSI C standard was finally adopted in December 1989, with the first copies becoming available in early 1990. The ANSI C was also adopted by ISO (International Standards Organization). The 1989 standard for C, along with Amendment 1, became a base document for Standard C++, defining the C subset of C++. The version of C defined by the 1989 standard is commonly referred to as C89.

During the 1990s, the development of the C++ standard consumed most programmers' attention. However, work on C continued quietly along, with a new standard for C being developed. The end result was the 1999 standard for C, usually referred to as C99.

3.2 Features of C

C is a general-purpose high level language that was originally developed for the UNIX operating system. The UNIX operating system and virtually all UNIX applications are written in C language. C has now become a widely used professional language for various reasons. Figure 2 indicates feature of c at a glance.

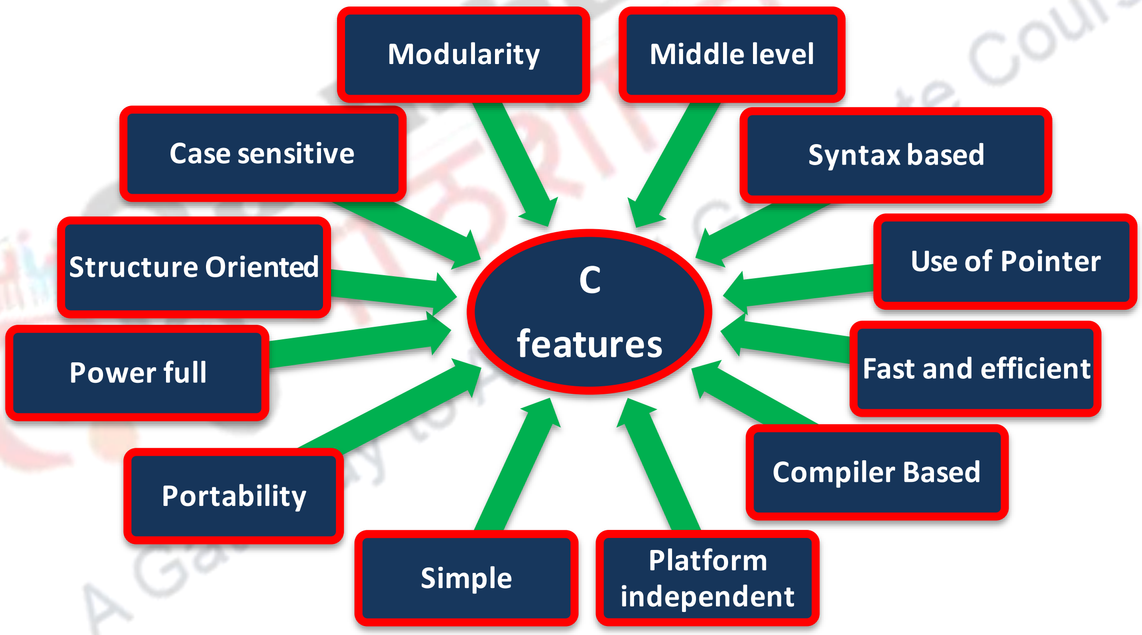


Figure 2: Features of C – at a glance

It is a very simple and easy language. C language is mainly used for develop desktop based application. All other programming languages were derived directly or indirectly from C programming concepts. C language has following features;

1. **Simple**

Every c program can be written in simple English language so that it is very easy to understand and developed by programmer.

2. **Portability**

C programs are portable. It is the concept of carrying the instruction from one system to another system. In C language **.C** file contain source code, we can edit also this code. **.exe** file contain application, only we can execute this file. When we write and compile any C program on window operating system that program easily run on other window based system.

3. **Powerful**

C is a very powerful programming language, it have a wide verity of data types, functions, control statements, decision making statements, etc.

4. **Structure oriented**

C is a Structure oriented programming language.Structure oriented programming language aimed on clarity of program, reduce the complexity of code, using this approach code is divided into sub-program/subroutines. These programming have rich control structure

5. **Case sensitive**

It is a case sensitive programming language. In C programming 'break and BREAK' both are different

6. **Modularity**

It is concept of designing an application in subprogram that is procedure oriented approach. In c programming we can break our code in subprogram.

For example we can write a calculator programs in C language with divide our code in subprograms.

7. **Middle level language**

C programming language can supports two level programming instructions with the combination of low level and high level language that's why it is called middle level programming language.

8. **Syntax based language**

C is a strongly tight syntax based programming language.

9. Efficient use of pointers

Pointers is a variable which hold the address of another variable, pointer directly direct access to memory address of any variable due to this performance of application is improve. In C language also concept of pointer are available.

10. Fast and Efficient

C programs are fast to compile and run efficiently.

11. Compiler based

C is a compiler based programming language that means without compilation no C program can be executed. First we need compiler to compile our program and then execute

12. Platform independent

A language is said to be platform independent when the program executes independent of the computer hardware and operating system.
Therefore C is platform independent programming language.
Note: .obj file of C program is platform dependent.

c a Middle-Level Language

C is often called a middle-level computer language. It does not mean that C is less powerful, difficult to use, or less developed than a high-level language such as BASIC or Pascal. Rather, C is thought of as a middle-level language because it combines the best elements of high-level languages with the control and flexibility of assembly language. As a middle-level language, C allows the manipulation of bits, bytes, and addresses, the basic elements with which the computer functions. Despite this fact, C code is also very portable. C language is very often known as a **System Programming Language**, because it is used for writing assemblers, compilers, editors and even operating systems,

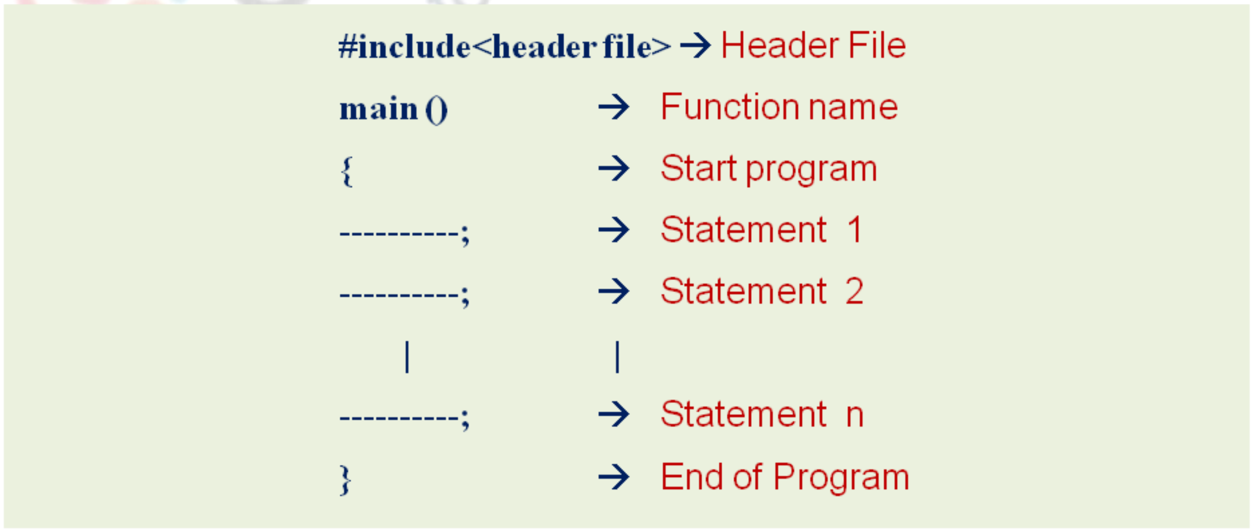
3.3 Applications of C

C language is used in wide variety of applications. Practical applications of C language are several, right form writing the operating systems like UNIX, Windows to creating antivirus programs. Some application of C language is given below.

- To design the system software like operating system and compiler.
- To develop application software like database and spread sheets.
- For development of Graphical related application like computer simulators and mobile games.
- C programming language can be used to design Network Devices Drivers
- For writing embedded system software
- Development of Text editors and language interpreters.
- Creating video games, 3 D animations, Multimedia applications, Image processing, modelling and simulations etc.

3.4 Structure of C program

Programming structure of C language consists of header file, main function and body of function as shown in figure 3.



```
#include<header file> → Header File
main ()               → Function name
{                     → Start program
----------;           → Statement 1
----------;           → Statement 2
     |                  |
----------;           → Statement n
}                     → End of Program
```

Figure 3: Structure of c program

The C language program starts form main() function which is **Function name.** The program function start with **{** (open brace) and ends with **}** (close brace). Then valid c statements are to be written between open brace and close brace. The functions written between open and close brace called as **function body. Semicolon (;)** is used to end the valid statement.

Let us consider a very simple program; here programmer wants to display some text message.

```
#include <stdio.h>                 //library function for standard input output
main ( )                           //Execution begins
{                                  //Opening brace
printf("Fergusson College");       //Executable statement
}                                   //Closing brace
```

Output of program is

```
            Fergusson College
```

3.5 Problem solving process

Any scientific and engineering problem can be solved using some Problem Solving process. Here total six important steps are given to solve any general problem.

1. Understanding a problem
2. Identifying necessary inputs and expected output
3. Design an algorithm/flowchart
4. Writing or coding the program
5. Testing the code
6. Debugging

Everything begins with problem therefore it is necessary to understand the problem clearly. With a given problem, the second step is to know necessary inputs and expected outputs. By knowing the input and output parameter the next step is to design an algorithm to solve the problem or one prefer to draw the design in a graphical way using flowchart. Once the algorithm or flow chart is ready, the obvious step is to start Writing or coding the program in an appropriate programming language which is suitable for given a application Any program is not complete without testing the code with possible inputs. This stage is also referred to as program execution or running the program. If there are any errors in the program then debugging is an important step in problem solving i.e. removing the bugs or error.

4. The algorithm

An algorithm is a finite sequence of instructions which can be carried out to solve a particular problem in order to obtain the desired results.

- A finite sequence of instructions
- The program must be planned before writing.
- Represents the logic of the program
- Instructions are to be written in the proper sequence.
- Expressed in the programming language, becomes the program

4.1 Characteristics of an algorithm

An algorithm must satisfy the following criteria:

1) Input: Zero or more quantities are externally supplied as inputs.

2) Output: At least one quantity as output is produced.

3) Finiteness: An algorithm terminates after a finite number of steps.

4) Definiteness: Each instruction must be clear and unambiguous.

5) Effectiveness: Every instruction must be very basic so that it can b carried out in principle, by a person using just a pencil and paper.

4.2 Examples

Let us consider few example of creating algorithm for a given problem.

Example 1: Algorithm for sum of two numbers

```
Step 1: Take the first number from the user (input).
Step 2: Take the second number from the user (input).
Step 3: Add the two numbers to get the sum (process).
Step 4: Display the sum on the screen (output).
```

Example 2: Write an algorithm to check whether the entered number is positive, negative or zero.

```
Step 1: Take the number from the user
Step 2: If the number is less than 0,
            then display the entered number is negative.
Step 3: If the number is greater than 0, display the entered
          number is positive.
Step 4: otherwise display the entered number is zero.
```

Example 3: calculating the sum of 10 numbers**.**

The formula for the sum of 10 numbers

$$\text{sum} = \sum_{i=1}^{10} \text{number i}$$

```
Step 1: Let the sum be initially set to 0.
Step 2:  Let i be initially 1.
Step 3: Take the i th  number from the user.
Step 4: Add i th number to the sum.
Step 5: Increase the value of i by 1.
Step 6: Check if the value of i is less than 10; then go to step 4.
            Otherwise go to step 7.
Step 7: Output the sum
```

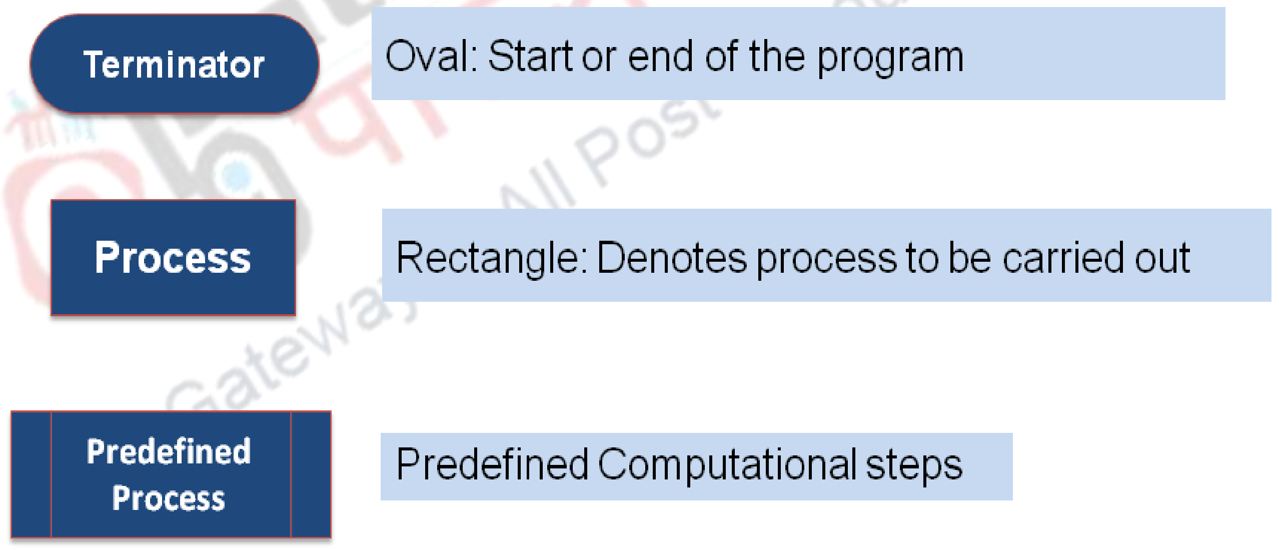
5. Flow charts

A flowchart is a pictorial representation of an algorithm. Flow chart consists of set of flow chart symbols connected by arrows. Each symbol has its own meaning, tells to compiler what must be done at that point. The sequence of flow chart symbols can be considered as a program.

Flow charts are better way of communicating logic of system. Problem can be analyzed in effective way with the flowchart. If the algorithms are represented in a graphical or pictorial form they are easy to understand. Efficient coding acts as Guide or Blue print during system analysis or development phase.

5.1 Guidelines for flowchart

Usually flow charts are drawn using some standard symbols. Some special symbols can also be developed when required. Standard symbols, which are frequently required for flowcharting are shown as follows.


Terminator
Oval: Start or end of the program
Process
Rectangle: Denotes process to be carried out
Predefined Process
Predefined Computational steps

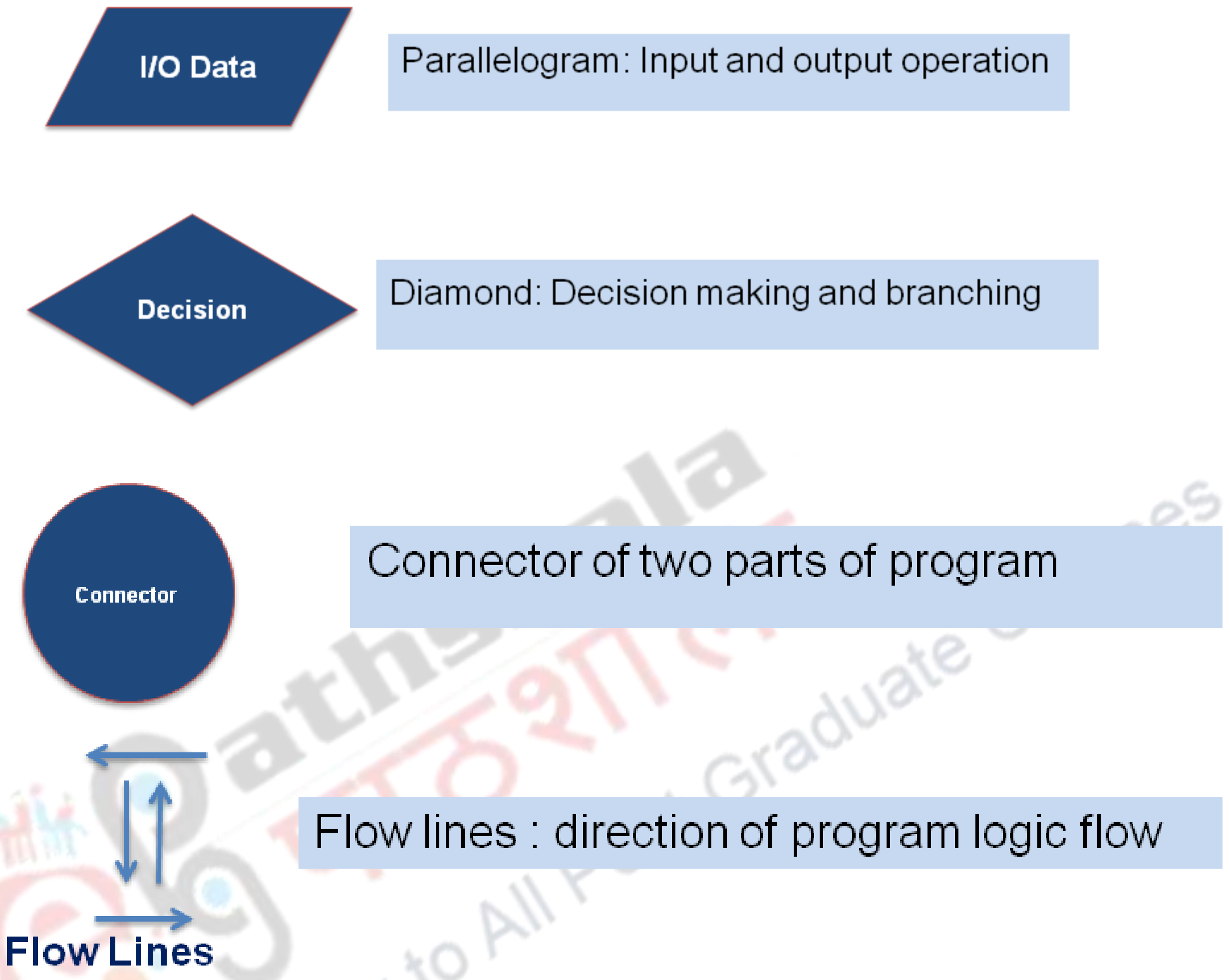
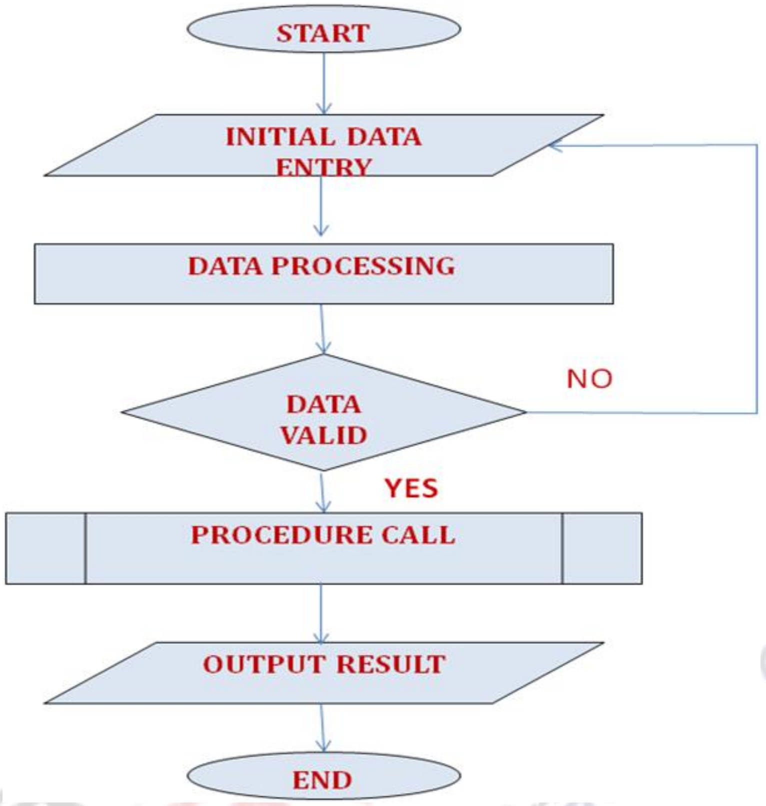

I/O Data
Parallelogram: Input and output operation
Decision
Diamond: Decision making and branching
Connector
Connector of two parts of program
Flow lines : direction of program logic flow
Flow Lines

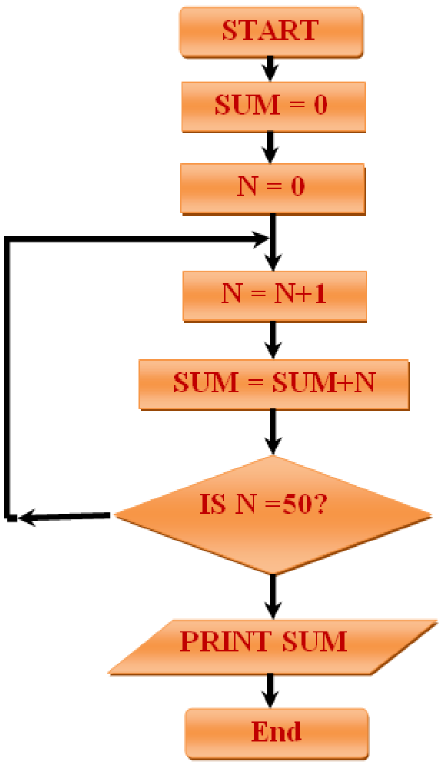
Figure 4: Flowchart symbols

Let us now consider simple general flowcharting technique. Let us start drawing a flowchart for a given problem.



Here different flowchart symbols are shown and are interconnected by the flow line. Start, input, processing, decision, predefined process, output and terminator are interconnected.

1. **Example 1:** Flow chart to find the sum of first 50 natural numbers



In this example we can draw simple flowchart for sum of first 50 natural numbers. So we need flowchart symbols for input,output,process,decision,start and end. The first and last symbols are starting and end of flowcharting. Initial sum of numbers and number assumed to be zero which are inputted in input symbol. The sum is added to the number which is initialized as zero. The decision block used in the flowchart takes the decision when number is equal to 50, if not then natural number is incremented by one and added in to the sum. This process continues till number N goes up to 50.If number is 50 then final sum will be calculated and displayed using output symbol.

Example using Algorithm and Flow chart.

Write an algorithm and draw a flowchart for computing factorial n (n!) Where n! = 1 x 2 x 3 x …… x n

Algorithm :

Step 1 : Input the value of n

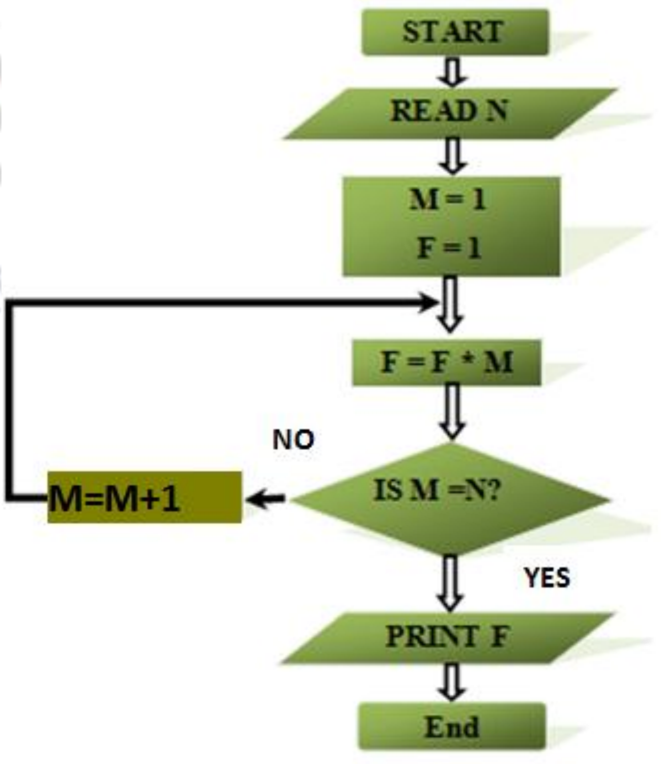
Step 2 : Let factorial F and m initialized to 1

Step 3 : Multiply F by M (F = F x M)

Step 4 : Check whether M = n if no then go to step 5 otherwise go to step 6

Step 5 : Increase the value of M by 1 and go to step 3.

Step 6 : output factorial F .



6. **Summary**

In the language learning it is most important is to have deep knowledge about what to tell computer to do and use different implementation strategies to write algorithm. With the algorithm in hand, it is possible to write the instructions necessary to implement the algorithm. Programs are developed to solve some problem. Programming language levels are: Machine language, Assembly language, Middle level language and High level language.

C is a general-purpose high level language that was originally developed for the UNIX operating system. C language is mainly used for develop desktop based application. All other programming languages were derived directly or indirectly from C programming concepts. C is often called a middle-level computer language. C language is very often known as a **System Programming Language**, because it is used for writing assemblers, compilers, editors and even operating systems. C language is used in wide variety of applications. Practical applications of C language are several, right form writing the operating systems like UNIX, Windows to creating antivirus programs.