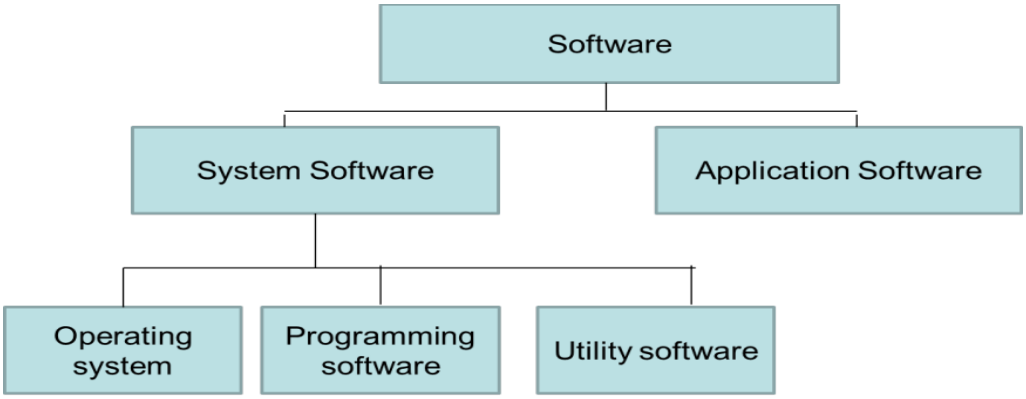
Introduction to Programming Environment:

- Prog. Envt. is the envt, in which programs are created & tested.
- It is much closer to O.S. than the design of a programming language.
- Prog. Envt. consists of a set of support tools & a command language (instruction Set) for invoking the tools.
- Each support tool is another program that may be used by programmer as an aid during ≥1 stages of program-creation.

Tools example : - Editors, debuggers, verifiers, compilers, test data generators & pretty printers etc.

Types of Software



Program: A sequence of instructions for the computer is called a program. The data that are manipulated by the program create the database.

1. **System software-**It is the software used to manage and control the hardware components and which allow interaction between the hardware and the other types of software. The most obvious type of system software is **operating system** and **device drivers**.
   Utility software-It is software such as anti-virus software, firewalls, disk defragmenters and so on.
   which helps to maintain and protect the computer system but does not directly interface with the hardware.
2. **Applications software** (also known as 'apps') are designed to allow the user of the system to complete a specific task(application). They include programs such as web browsers, MS office, games, media players and so on.

COMPUTER LANGUAGES

 Languages are a means of communication. Normally people interact with each other through a language. On the same pattern, communication with computers is carried out through a language. This language is understood both by the user and the machine.

 Just as every language like English, Hindi has its own grammatical rules; every computer language is also bounded by rules known as syntax of that language. The user is bound by that syntax while communicating with the computer system.

Computer languages are broadly classified as:

 Low Level Language: The term low level highlights the fact that it is closer to a language which the machine understands

The low level languages are classified as:

 Machine Language: This is the language (in the form of 0's and 1's, called binary numbers) understood directly by the computer. It is machine dependent. It is difficult to learn and even more difficult to write programs.

 Assembly Language: This is the language where the machine codes comprising of 0'sand 1's are substituted by symbolic codes (called mnemonics) to improve their understanding. It

is the first step to improve programming structure. Assembly language programming is simpler and less time consuming than machine level programming, it is easier to locate and correct errors in assembly language than in machine language programs. It is also machine dependent. Programmers must have knowledge of the machine on which the program will run.

 High Level Language: Low level language requires extensive knowledge of the hardware since it is machine dependent. To overcome this limitation, high level language has been evolved which uses normal English, which is easy to understand to solve any problem. High level languages are computer independent and programming becomes quite easy and simple. Various high level languages are given below:

 C: Structured Programming Language used for all purpose such as scientific application, commercial application, developing games etc.

Algorithm:

The Fundamental knowledge needed to solve problems using a computer is the notion of an algorithm.

*"**An Algorithm is a precise specification of a seq. of instructions (tells what task is to be executed) to be carried out in order solve a given problem".***

Features of a good algorithm:

1. **Finiteness:** An algorithm must terminate after a finite number of steps and further each step must be executable in finite amount of time

2. **Definiteness** (no ambiguity): Each steps of an algorithm must be precisely defined; the action to be carried out must be rigorously and unambiguously specified for each case.
3. **Inputs/ Output**: An algorithm has zero or more (but only finite), number of inputs and (atleast 1)output.
4. **Effectiveness(efficient):** An algorithm should be effective. This means that each of the operation to be performed in an algorithm must be sufficiently basic that it can be done exactly and in a finite length of time.

FLOWCHARTS

- Flowchart is developed for showing the steps involved in a process(algorithm).
- A flowchart is a diagram made up of *boxes*, *diamonds* and other shapes, connected by arrows.
- Flowchart combines symbols and flow lines, to show the operation of an algorithm in a figure.

Flocharting Symbols

Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or ouptut.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

Advantages of Using FLOWCHARTS: -

 Communication: - Flowcharts are better way of communicating the logic of a system to all concerned.

 Effective analysis: - With the help of flowchart, problem can be analyzed in more effective way.

 Proper documentation: - Program flowcharts serve as a good program documentation, which is needed for various purposes.

 Efficient Program Maintenance: - The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part

Disadvantages Of Using FLOWCHARTS: -

1. Complex logic: - Sometimes, the program logic is quite complicated. In that case,

flowchart becomes complex and clumsy.

2. Alterations and Modifications: - If alterations are required the flowchart may require redrawing completely.

3. Reproduction: - As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.

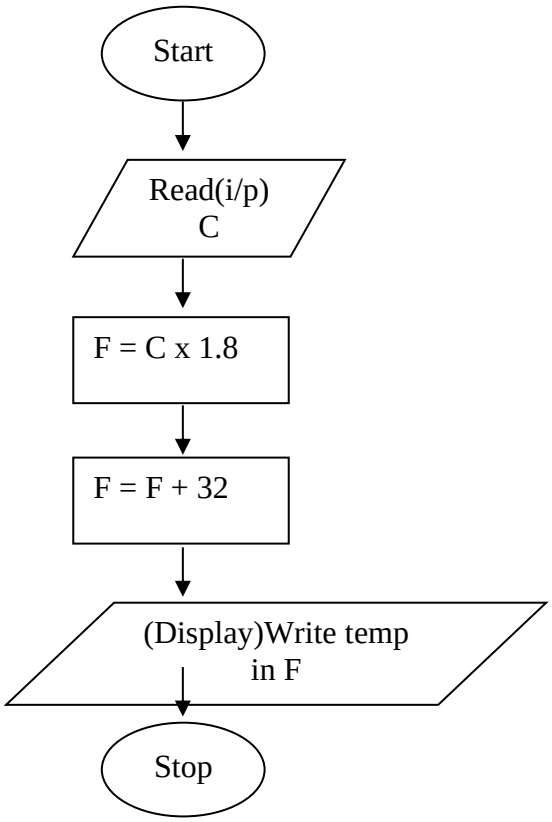
4. The essentials of what is done can easily be lost in the technical details of how it is done.

Example 1:Algo. designed **to convert temperature in degree Celsius $\xrightarrow{to}$ degree Fahrenheit.**

Input: Temp. in degree Celsius. **Output: Temp. in degree Fahrenheit.**

1. Start
2. Obtain value of temp. in °C.
3. Apply mathematical formula of conversion(Multiply this value by 1.8 & add 32)
4. Assign value obtained of temp. in °F.
5. Display result.
6. Stop.

Flowchart: **to convert temperature in degree Celsius $\xrightarrow{to}$ degree Fahrenheit.**



Example 2: *Design Algo. for* ***choosing the largest of 3 given positive integers.***

Input : Three integers :– A, B, C. **o/p : Largest no. among A, B, C.**

Step1: Read three numbers A, B, C

Step2: Compare A with B

Step3: If A is larger compare it with C

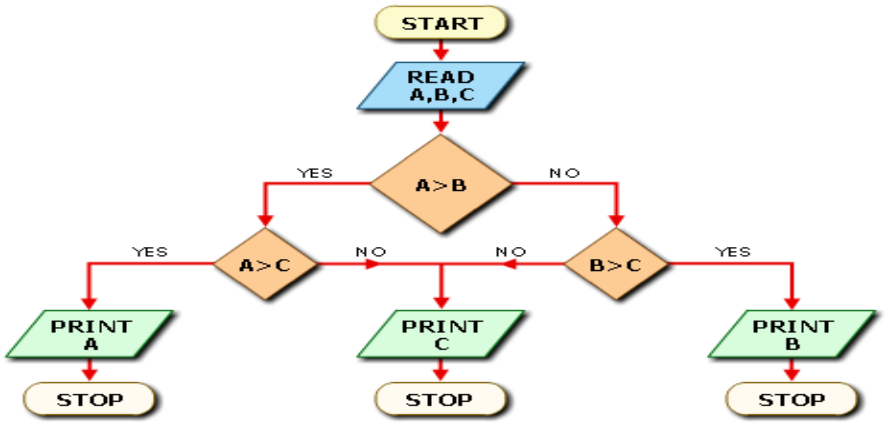
Step4: If A is larger than C then A is the largest otherwise C is the largest.

Step5: If A is smaller than or equal to B in the first step then B is compared with C.

Step6: If B is larger than C then B is the largest number otherwise C is the largest number.

Step7: Stop

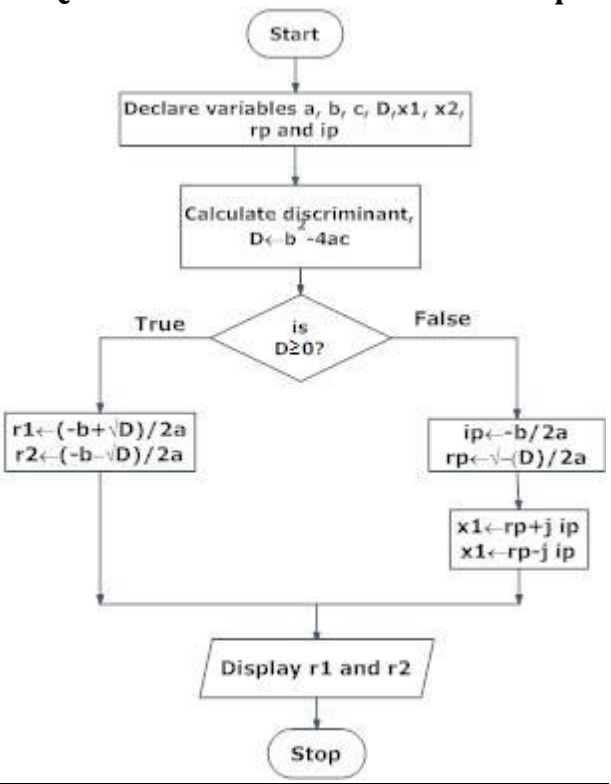
The above algorithm may be expressed much more clearly and concisely using a flowchart.



Example 3*. Design Algo for finding the sum of given N nos. & display the result.*
I/p: N nos. entered (one by one) by keyboard. O/p: sum of N elements (entered by keyboard).

1. start
2. obtain the value N(total nos. to be entered)
3. Initialize variable sum & counter to 0.
4. Read the no. as num
5. Add this num to sum and update sum.
6. Increment the value of the counter by one
7. Continue steps 4 to 6 till counter < = N.
8. Print the sum value.
9. Stop

Q: Draw flowchart to find roots of quadratic equation.



The way of data processing For Program

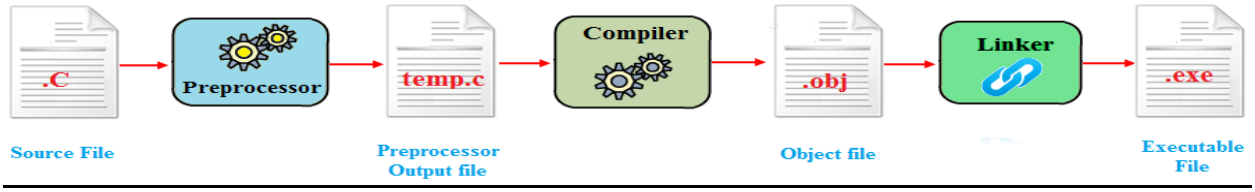
Before discussing our program we should understand the way of data processing by computer:

Steps of Data-Processing using Computer:

1. Analysis of the requirement (problem).
2. Designing of method after understanding the- to do task.
3. Expressing the method in a step-wise procedure to design algorithm (in user-language)
4. Expressing the Algorithm in a std. notation using a programming language so a program is designed to be executed.
5. Input the program (& data) to be executed & save (store) it in the memory of computer.
6. Issue the command to execute the program in computer.
7. The computer interprets/compiles the program saved in memory (RAM). As per the instructions Written in the program (to need or store or manipulate) the processor of computer performs accordingly.
8. After executing the program, the result is written to the O/p unit if execution is successful or error message will be displayed.
9. Then error removal (modify and recompile the program) or further processing continues.

C Program: Generally three basic phases occurs when we execute any C prog:

- **Preprocessing**
- **Assembling & Compiling**
- **Linking & Loading**



Compile, link and execute stages of **hello.c** in Linux OS(using GCC)

Assembler: A computer will not understand any program written in a language, other than its machine language. The programs written in other languages must be translated into the machine language. Such translation is performed with the help of software. A program which translates an assembly language program into a machine language program is called an assembler. If an assembler which runs on a computer and produces the machine codes for the same computer then it is called *self assembler* or *resident assembler*. If an assembler that runs on a computer and produces the machine codes for other computer then it is called *Cross Assembler.*

Assemblers are further divided into two types: One Pass Assembler and Two Pass Assembler. One pass assembler is the assembler which assigns the memory addresses to the variables and translates the source code into machine code in the first pass simultaneously. A Two Pass Assembler is the assembler which reads the source code twice. In the first pass, it reads all the variables and assigns them memory addresses. In the second pass, it reads the source code and translates the code into object code.

Compiler: It is a program which translates a high level language program into a machine language program. A compiler is more intelligent than an assembler. It checks all kinds of limits, ranges, errors etc. But its program run time is more and occupies a larger part of the memory. It has slow speed. Because a compiler goes through the entire program and then translates the entire program into machine codes. If a compiler runs on a computer and produces the machine codes for the same computer then it is known as a *self compiler* or *resident compiler*. On the other hand, if a compiler runs on a computer and produces the machine codes for other computer then it is known as a *cross compiler.*

Interpreter: An interpreter is a program which translates statements of a program into machine code. It translates only one statement of the program at a time. It reads only one

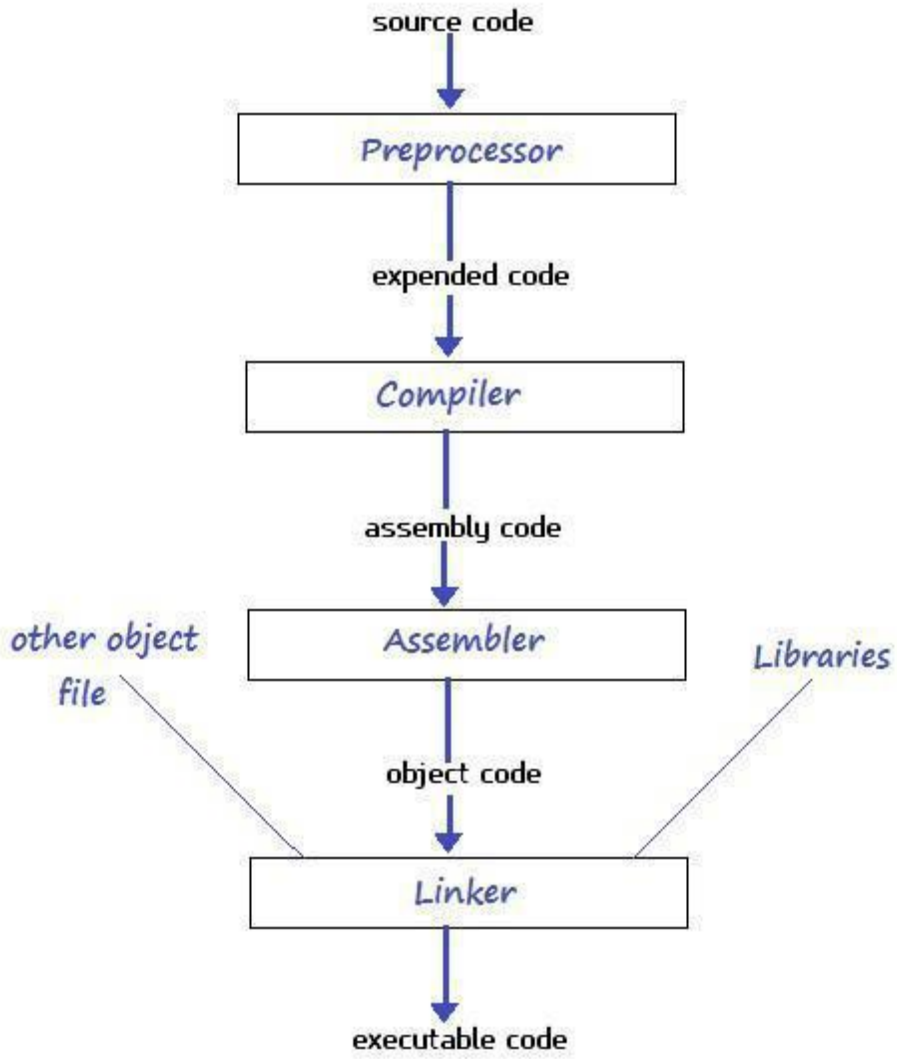
statement of program, translates it and executes it. Then it reads the next statement of the program again translates it and executes it. In this way it proceeds further till all the statements are translated and executed. On the other hand, a compiler goes through the entire program and then translates the entire program into machine codes. A compiler is 5 to 25 times faster than an interpreter.

By the compiler, the machine codes are saved permanently for future reference. On the other hand, the machine codes produced by interpreter are not saved. An interpreter is a small program as compared to compiler. It occupies less memory space, so it can be used in a smaller system which has limited memory space.

Linker: In high level languages, some built in header files or libraries are stored. These libraries are predefined and these contain basic functions which are essential for executing the program. These functions are linked to the libraries by a program called Linker. If linker does not find a library of a function then it informs to compiler and then compiler generates an error. The compiler automatically invokes the linker as the last step in compiling a program.

Not built in libraries, it also links the user defined functions to the user defined libraries. Usually a longer program is divided into smaller subprograms called modules. And these modules must be combined to execute the program. The process of combining the modules is done by the linker.

Loader: Loader is a program that loads machine codes of a program into the system memory. In Computing, a **loader** is the part of an Operating System that is responsible for loading programs. It is one of the essential stages in the process of starting a program. Because it places programs into memory and prepares them for execution. Loading a program involves reading the contents of executable file into memory. Once loading is complete, the operating system starts the program by passing control to the loaded program code. All operating systems that support program loading have loaders. In many operating systems the loader is permanently resident in memory.



Difference between Compiler and Interpreter:

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.

Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Ex-Programming language like Python, Ruby use interpreters.	Ex-Programming language like C, C++ use compilers.