For B.E. B.C.A. M.C.A. M.C.M. B.S.C. Polythenic

Computer Fundamentals

Algorithm & Flowcharts

by Choudhari Sir

25 Years ANNIVERSARY

CCIT

Keeping Pace with Technology

An ISO 9001 : 2008 Certified Company

website: www.ccitindia.com

Rajapeth: 0721-2563615 GadgeNagar: 0721-2552289

A History of Computers

Generation	**First**	
Vacuum tube technology, Unreliable, Supported machine language only, Very costly, Generated lot of heat, Slow input and output devices, Huge size, Need of A.C., Non-portable Consumed lot of electricity		
Generation	**Second**	
Use of transistors,Smaller size as compared to first gen.,Generated less heat as compared to first gen., Consumed less electricity as than first gen, Faster than first gen., Still very costly, A.C. needed,Supported machine and assembly languages		
Generation	**Third**	
IC used, More reliable in comparison to previous two generations ,Smaller size, Generated less heat, Faster, Lesser maintenance, Still costly, A.C needed, Consumed lesser, electricity, Supported high level languages		
Generation	**Fourth**	
VLSI technology used, Very cheap, Portable and reliable, Use of PC's, Very small size, Pipeline processing, No A.C. needed, Concept of internet was introduced, Great developments in the fields of networks, Computers became easily available		

Computer

It is an electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals

Basic characteristics about computer are:

1. **Speed:** - A computer can perform millions of instructions per second.
2. **Accuracy**: - The degree of accuracy of computer is very high and every calculation is performed with the same accuracy.
3. **Diligence**: - A computer is free from tiredness, lack of concentration etc.
4. **Versatility**: - It means the capacity to perform completely different type of work.
5. **Power of Remembering**: - Computer has the power of storing any amount of information or data.
6. **No IQ**: - Computer is a dumb machine and it cannot do any work without instruction from the user.
7. **No Feeling**: - It does not have feelings or emotion, taste, knowledge and experience.
8. **Storage**: - The Computer has an in-built memory where it can store a large amount of data.

Computer Organization

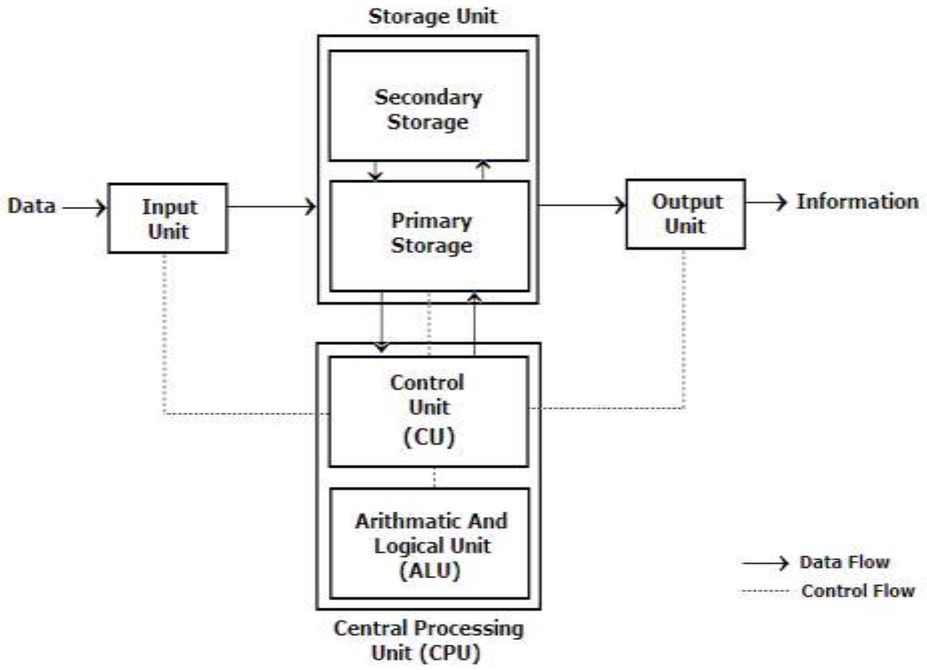
A computer can be organized into 4 blocks

1] Input Devices

2] Output Devices

3] Central Processing [A.L.U. & C.U.]

4] Memory



CPU

A **central processing unit** (**CPU**) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.

CPU features:

- CPU is considered as the brain of the computer.
- CPU performs all types of data processing operations.
- It stores data, intermediate results and instructions (program).
- It controls the operation of all parts of computer.

CPU performs four basic Steps / Operations:

1. **Fetch** Each instruction is stored in memory and has its own address. The processor takes this address number from the program counter, which is responsible for tracking which instructions the CPU should execute next.
2. **Decode** All programs to be executed are translated to into Assembly instructions. Assembly code must be decoded into binary instructions, which are understandable to your CPU. This step is called decoding.
3. **Execute** While executing instructions the CPU can do one of three things: Do calculations with its ALU, move data from one memory location to another, or jump to a different address.
4. **Store** The CPU must give feedback after executing an instruction and the output data is written to the memory.

The Control Unit

This unit controls the operations of all parts of computer but does not carry out any actual data processing operations.

Functions of this unit are:

- It is responsible for controlling the transfer of data and instructions among other units of a computer.
- It manages and coordinates all the units of the computer.
- It obtains the instructions from the memory, interprets them, and directs the operation of the computer.
- It communicates with Input/Output devices for transfer of data or results from storage.
- It does not process or store data.

The Arithmetic and Logic Unit

The arithmetic/logic unit (ALU) contains the electronic circuitry that executes all arithmetic and logical operations.

This unit consists of two subsections namely

- Arithmetic section
- Logic Section

Arithmetic Section :

Function of arithmetic section is to perform arithmetic operations like addition, subtraction, multiplication and division. All complex operations are done by making repetitive use of above operations.

Logic Section :

Function of logic section is to perform logic operations such as comparing, selecting, matching and merging of data.

Memory and Storage

Memory is used to store data and instructions. Computer memory is the storage space in computer where data is to be processed and instructions required for processing are stored. The memory is divided into large number of small parts called cells. Each location or cell has a unique address which varies from zero to memory size minus one.

Memory is primarily of three types

- Cache Memory
- Primary Memory/Main Memory
- Secondary Memory

Cache Memory

Cache memory is a very high speed semiconductor memory which can speed up CPU. It acts as a buffer between the CPU and main memory. It is used to hold those parts of data and program which are most frequently used by CPU. The parts of data and programs are transferred from disk to cache memory by operating system, from where CPU can access them.

Advantages

The advantages of cache memory are as follows:

- Cache memory is faster than main memory.
- It consumes less access time as compared to main memory.
- It stores the program that can be executed within a short period of time.
- It stores data for temporary use.

Disadvantages

The disadvantages of cache memory are as follows:

- Cache memory has limited capacity.
- It is very expensive.

Primary Memory (Main Memory)

Primary memory holds only those data and instructions on which computer is currently working. It has limited capacity and data is lost when power is switched off. It is generally made up of semiconductor device. These memories are not as fast as registers. The data and instruction required to be processed reside in main memory. It is divided into two subcategories RAM and ROM.

Characteristics of Main Memory

- These are semiconductor memories
- It is known as main memory.
- Usually volatile memory.
- Data is lost in case power is switched off.
- It is working memory of the computer.
- Faster than secondary memories.
- A computer cannot run without primary memory.

Types of Primary Memory

- RAM
- ROM
- PROM
- EPROM
- EEPROM
- REGISTERS

RAM [RANDOM ACCESS MEMORY]

RAM is the best example of primary storage. RAM is a volatile memory because it loses its contents when there is a power failure in the computer system. The memories which lose their contents on power failure are called volatile memories.

ROM [READ ONLY MEMORY]:

ROM is also formed by Integrated Circuits. The data which is stored in ROM is permanent. The ROM can only read the data by CPU but can’t be edited or manipulated.ROM is a non-volatile memory because it will not lose its contents when there is a power failure in the computer system. The contents in the ROM can neither be changed nor deleted.

PROM [PROGRAMMABLE READ ONLY MEMORY]:

PROM we can store our programs in PROM chip. Once the programs are written it cannot be changed and remains intact even if the power is switched off.

EPROM [ERASABLE PROM]:

EPROM chip can be programmed time and again by erasing the information stored earlier in it. EPROM chip has to be exposed to sunlight for some time so that ultra violet rays fall on the chip and that erases the data on the chip and the chip can be re-programmed using a special programming facility. There is another type memory called **EEPROM** that stands for Electrically Erasable Programmable Read Only Memory in which we can erase the data and re-program it with a fresh content.

REGISTERS:

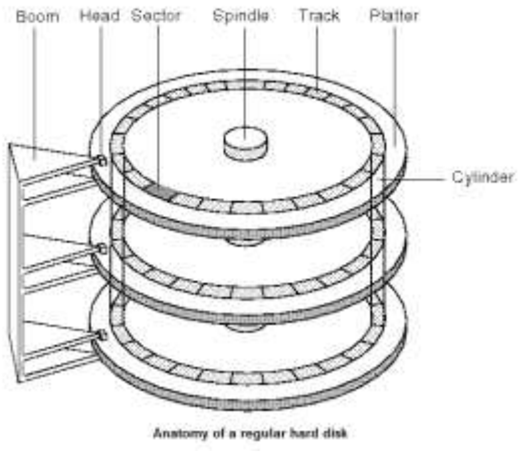
Actually computer system uses a number of memory units called registers. Registers store data or information temporarily and pass it on as directed by the control unit.

Secondary Memory

This type of memory is also known as external memory or non-volatile. It is slower than main memory. These are used for storing data/Information permanently. CPU directly does not access these memories instead they are accessed via input-output routines. Contents of secondary memories are first transferred to main memory, and then CPU can access it. For example : disk, CD-ROM, DVD etc.

Characteristic of Secondary Memory

- These are magnetic and optical memories
- It is known as backup memory.
- It is non-volatile memory.
- Data is permanently stored even if power is switched off.
- It is used for storage of data in a computer.
- Computer may run without secondary memory.
- Slower than primary memories.



Anatomy of a regular hard disk

Difference between Primary and Secondary memory

Subject	Primary Memory	Secondary Memory
Volatility	This is volatile.	This is Non-volatile.
Access Time	Access Time is higher Than Secondary memories.	Access Time is lower Than Secondary memories.
Memory Status	This is a Temporary Memory.	This is a Permanent Memory.
Capacity	At present time, 512 MB to 8 GB RAMs are available.	At Present time 80 GB to 4 TB Hard Disc Drive are available.
Price	Higher than HDD/Secondary Memory	Lower Than primary Memory.
Connection	Connected via Slots.	Connected Via Cables.
Accessible	Primary memory is directly accessible to the CPU	Secondary memory is not directly accessible to the CPU

RAM VS ROM

Options	RAM	ROM
Accessibility	In reference with the processor, the information stored in the RAM is easily accessed	The processor cannot directly access the information that is stored in the ROM. To access the ROM information, first the information will be transferred into the RAM and then it gets executed by the processor
Working type	Both the read and write operations can be performed over the information that is stored in the RAM	The ROM memory only allows the user to read the information. User cannot make any changes to the information.
Storage	RAM memory is only used to store the temporary information.	ROM memory is used to store permanent information and cannot be deleted.
Speed	the accessing speed of RAM is faster, it assist the processor to boost up the speed	Speed is slower in comparison with RAM, ROM cannot boost up the processor speed
Data preserving	Electricity is needed in RAM to flow to preserving information	Electricity is not needed in ROM to flow to preserving information
Cost	The price of RAMs are comparatively high	The price of ROMs are comparatively low
Types	The RAM memory is categorized into two types they are the: Statistic RAM (SRAM) and Dynamic RAM (DRAM)	The ROM memory is categorized into three types, they are: PROM (Programmable Read Only memory), EPROM (Erasable Programmable Read Only memory) and EEPROM (Electrically Erasable Programmable Read Only memory)

Input Devices

An input device feeds data to the computer system for processing. Input is any data that we send to a computer for processing. That can be an image from a Digital Camera, or some letters types via keyboard in a word document.

List of Input Devices

- Keyboard
- Mouse
- Scanner
- Digital Camera
- Gamepad, Joystick, Steering wheel.
- Mic
- Barcode Reader
- Pen / Stylus
- Touch Screen
- Webcam
- Biometrics (Thumb impression / Face detection)

Output Devices

Output is the result of the data we can see through some output device like a picture displayed by the Monitor, a word documented printed by a printer etc. Output devices displays the processed form of data to the end user.

Common Output devices include;

- Monitor
- Printer
- Speaker

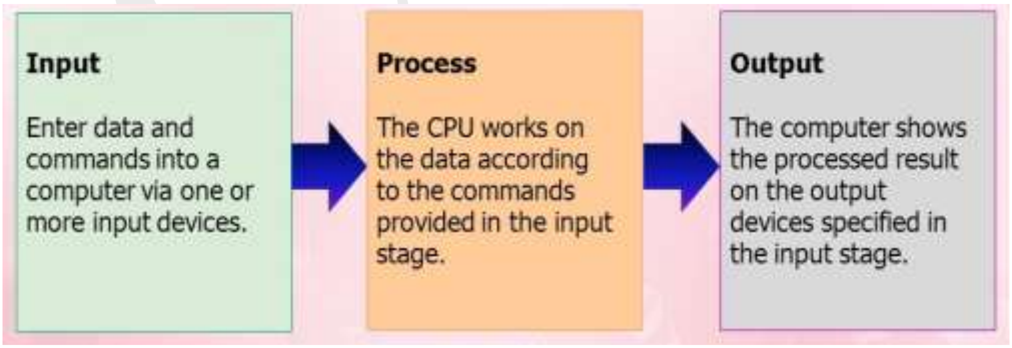
Input-Process-Output cycle

During the functioning of a computer it performs the I-P-O cycle that is; it needs certain *input*, carries out a *process* and produces the *output*.

Input Stage: Input means to put in or into and that is exactly what an input device does. An input device enables one to communicate with the computer. Input devices are used to enter information and issue commands. Examples of input devices are the keyboard, mouse, scanners, video cameras and microphones.

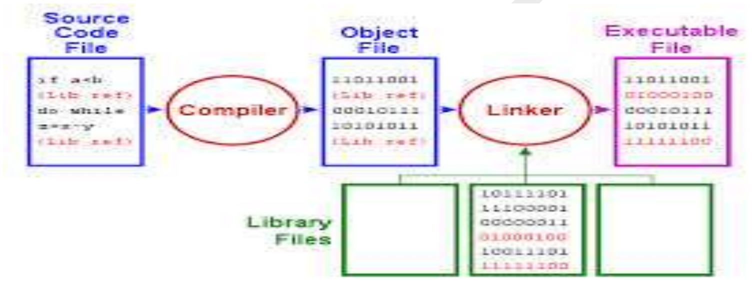
Process Stage: Information received from the input devices will then be processed in the Central Processing Unit (CPU). The CPU's job is to process instructions, perform calculations and manage the flow of information through a computer system. The CPU not only communicates with the input devices, but also the output and storage devices to perform tasks.

Output stage: After the information has been processed, or the necessary tasks completed on the information, it will be outputted to you. The output device enables a computer to communicate with you. These devices display information on a screen, create printed copies or generate sound. Some of these devices are the monitor, printer and speakers.


Input
Enter data and commands into a computer via one or more input devices.
Process
The CPU works on the data according to the commands provided in the input stage.
Output
The computer shows the processed result on the output devices specified in the input stage.

Compiler

A **compiler** is a computer program (or a set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language), with the latter often having a binary form known as object code. The most common reason for converting a source code is to create an executable program.



Interpreter

An interpreter is a computer program that reads the source code of another computer program and executes that program. Because it is interpreted line by line, it is a much slower way of running a program than one that has been compiled but is easier for learners because the program can be stopped, modified and rerun without time-consuming compiles.

Assembler

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.

Difference between Compiler and Interpreter

#	COMPILER	INTERPRETER
1	Compiler works on the complete program at once. It takes the **entire program** as input.	Interpreter program works line-by-line. It takes **one statement at a time** as input.
2	Compiler generates intermediate code, called the o**bject code or machine code.**	Interpreter does not generate intermediate object code or machine code.
3	Compiler executes conditional control statements **faster than interpreter.**	Interpreter execute conditional control statements at a much **slower speed**.
4	**Compiled programs take more memory** because the entire object code has to reside in memory.	Interpreter does not generate intermediate object code. As a result, **interpreted programs are more memory efficient.**
5	Compile once and run anytime. Compiled program does not need to be compiled every time.	Interpreted programs are interpreted line-by-line every time they are run.
6	Errors are reported after the **entire program is checked** for syntactical and other errors.	Error is reported as soon as the first error is encountered.
7	A compiled language is more difficult to debug.	Debugging is easy because interpreter stops and reports errors as it encounters them.
8	Compiler does not allow a program to run until it is completely error-free.	Interpreter runs the program from first line and stops execution only if it encounters an error.
9	Compiled languages are more efficient but difficult to debug.	Interpreted languages are less efficient but easier to debug. This makes such languages an ideal choice for new students.
10	**Examples** : C, C++, COBOL	**Examples** : BASIC, Python,PHP

Linker

A **linker** or **link editor** is a computer program that takes one or more object files generated by a compiler and combines them into a single executable file, library file, or another object file.

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.

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.

Algorithm

An Algorithm is well defined set of computational instructions designed to solve a computational problem.
Algorithms are expressed in English like language called as pseudo code and designed by using basic programming constructs such as sequence, branching and looping etc. The Algorithm can be graphically represented by using flow charts where different symbols are used to represent different programming constructs.

Characteristics of Algorithm:

- **Unambiguous** − Algorithm should be clear and unambiguous. Each of its steps and their input/outputs should be clear and must lead to only one meaning.
- **Input** − An algorithm should have 0 or more well defined inputs.
- **Output** − An algorithm should have 1 or more well defined outputs, and should match the desired output.
- **Finiteness** − Algorithms must terminate after a finite number of steps.
- **Feasibility** − Should be feasible with the available resources.
- **Independent** − An algorithm should have step-by-step directions which should be independent of any programming code.

Conventions for designing Algorithm

Start - start of algo

Stop - end of algo

Read / accept / input - to read input

Print / write / display - to display out

Operators

Arithmetic	**+ - * / *mod* ++ -- ^**
Relational	**< > <= >= = !=**
Logical	***AND OR NOT***

Goto Step n	***- to jump at a step***

```
if conditional then                    - for conditional execution
        ...
else
        ...
```

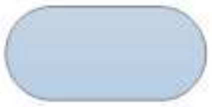
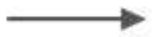
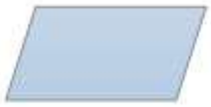
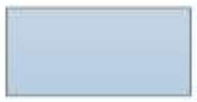
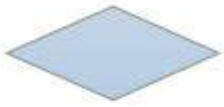
```
repeat while condition                 - for Iteration
        ……
```

```
repeat for iterator = lowerbound to upperbound step increment     -
for Iteration
        ……..
```

Flow Chart

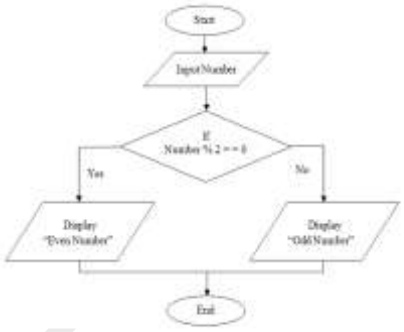
A flowchart is a visual representation of the sequence of steps and decisions needed to perform a process. Each step in the sequence is noted within a diagram shape. Steps are linked by connecting lines and directional arrows. This allows anyone to view the flowchart and logically follow the process from beginning to end.

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 output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

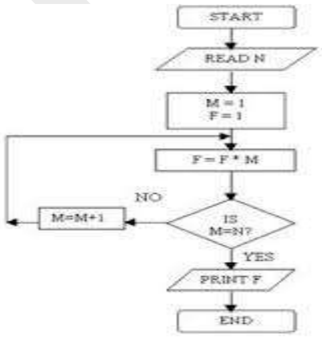
Flow Chart and Algorithm for Even/Odd

```
Step 1: Start
Step 2: read num
Step 3: if num mod 2 =0 then
                  Print "Even"
         Else
                  Print "Odd"
Step 4: End
```



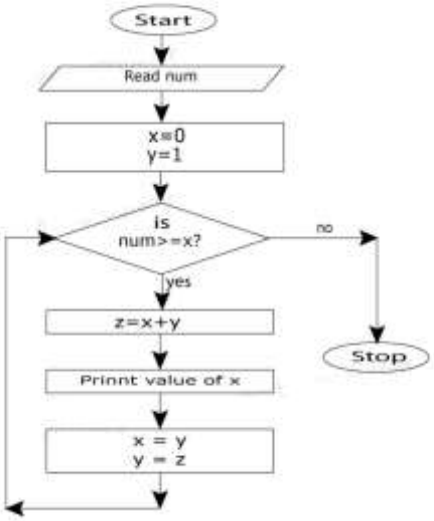
Flow Chart and Algorithm for Factorial of Number

```
Step 1: start
Step 2: read n
Step 3: set m:=1 and f:=1
Step 4: repeat step 4,5 while m<=n
Step 5:           set f := f x m
Step 6:           set m := m+1
Step 7: Print F
Step 8: end
```



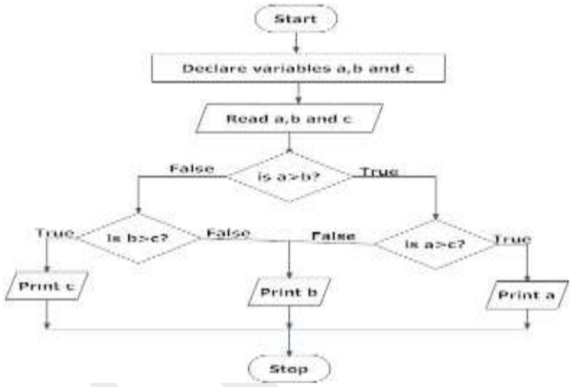
Flow chart and Algorithm for fabonacci Series

```
Step 1: start
Step 2: read n
Step 3: set x :=0 and y :=1
Step 4: repeat step 5 to 8 while x<= n
Step 5:           set z := x+y
Step 6:           print x
Step 7:           set x := y
Step 8:           set y := z
Step 9: end
```



FlowChart and Algorithm for Greatest of 3 numbers

```
Step1 : start
Step 2: read a,b,c
Step3 : if a > b and a>c then
                Print a
        Else if b> c then
                Print b
        Else
                Print c
Step 4: end
```



Algorithm to check for Prime Number

```
Step 1: start
Step 2: read n
Step 3: set dflag :=false
Step 4: repeat step 5for i:=2 to n-1
                If n mod I = 0 then
                        set dflag := true
                        goto step 5
step 5: if dflag=false then
                print "Prime Number"
        else
                print "Not a Prime Number"
step 6: end
```

Algorithm for sum of first n numbers

```
Step 1: start
Step 2: read n
Step 3: set sum := 0
Step 4: repeat step 5 for i = 1 to n
Step 5:                 set sum := sum + i
Step 6: print sum
Step 7: end
```

Algorithm for sum of series s= $1^2 + 2^2 + 3^2 + \ldots\ldots + n^2$

```
Step 1: start
Step 2: read n
Step 3: set sum := 0
Step 4: repeat step 5 for i = 1 to n
Step 5:                 set sum := sum + i*i
Step 6: print sum
Step 7: end
```

Algorithm to print first n even numbers

```
Step 1: start
Step 2: read n
Step 3: repeat step 4 for i = 1 to n
Step 4:         print i*2
Step 5: end
```

Algorithm to print n even numbers upto n

```
Step 1: start
Step 2: read n
Step 3: set i := 2
Step 4: repeat step 5,6 while i <= n
Step 5:         print i
Step 6:         Set i := i+2
Step 7: end
```

Algorithm to find sum of digits of given numbers

```
Step 1: start
Step 2: read n
Step 3: set sum := 0
Step 4: repeat step 5 to 7 while n >0
Step 5:         set d := n mod 10
Step 6:         set sum := sum + d
Step 7:         set n = INT(n/10)
Step 8: print sum
Step 9: end
```

Linear search or Sequential search

Linear search or Sequential search is a method for finding a target value within a list.
It sequentially checks each element of the list for the target value until a match is found or until all the elements have been searched.

Linear_Search(data , N, Item, Loc)
Where

Data	– array containing values
N	– size of Array
Item	- value to be searched
Loc	- variable in which position / result is to be returned

```
Step 1: set I:=1 and Loc := 0
Step 2: repeat step 3 and 4 while Loc=0 and I <= N
Step 3:        If data[I]=Item then Set Loc:=I
Step 4:                set I := I+1
Step 5: return
```

Binary Search

It is used to search an Item in a sorted array

- Binary search looks for a particular item by comparing the middle most item of the collection.
- If a match occurs, then the index of item is returned.
- If the middle item is greater than the item, then the item is searched in the sub-array to the right of the middle item.
- Otherwise, the item is searched for in the sub-array to the left of the middle item.
- This process continues on the sub-array as well until the size of the subarray reduces to zero.

Binary_Search(data , LB, UB, Item, Loc)

Where

Data	– array containing values in sorted order
LB	– lower bound of Array
UB	– upper bound of Array
Item	- value to be searched
Loc	- variable in which position / result is to be returned

```
Step 1: set MID := INT ((LB+UB)/2)
Step 2: repeat step 3 and 4 while LB<UB and data[MID] <> ITEM
Step 3:     if Item< data[MID] then
                Set UB := MID +1
            Else
                Set LB := MID -1
Step 4:     MID :=INT ( (LB+UB )/2)
Step 5: if data[MID] = Item then
            Set LOC := MID
        Else
            Set Loc := NULL
Step 6:  Return
```

Bubble Sort

This sorting algorithm is comparison-based algorithm in which

- Each pair of adjacent elements is compared and the elements are swapped if they are not in order.
- This process is repeated as many times as necessary, until the array is sorted.

It is called Bubble sort, because with each iteration the smaller element in the list bubbles up towards the first place, just like a water bubble rises up to the water surface.

```
Bubble ( data, N)
Where
        Data   – array containing values to be sorted
        N      – size of Array
Step 1: start
Step 2: repeat step 3 and 4  for I:=1 to N-1
Step 3:         repeat step 4 for J:= 1 to N-I
Step 4:                 if data[J] > data[J+1] then
                                Set temp := data[J]
                                Set data[J] := data[J+1]
                                Set data[J+1] := temp
Step 5: return
```

Selection Sort

- This sorting algorithm is an in-place comparison-based algorithm.
- In this the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end.
- Initially, the sorted part is empty and the unsorted part is the entire list.
- The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array.
- This process continues moving unsorted array boundary by one element to the right.

```
Selection_Sort(A, N)
Where
        A       – array containing values to be sorted
        N       – size of Array
Step 1: Repeat Step 2 to 5 for I:=1 to N-1
Step 2:         Set J:=I
Step 3:         Repeat step 4 for K:=I+1 to N
Step 4:                 if A[K]<A[J] then
                                J:=K
Step 5:  if I <> J then
                Set Temp   :=A[I]
                Set A[I]   := A[J]
                Set A[J]   :=Temp
Step 6: Return
```

Insertion Sorting

- In the First iteration, second element A[1] is compared with the first element A[0].
- In the second iteration third element is compared with first and second element.
- In general, in every iteration an element is compared with all the elements before it.
- While comparing if it is found that the element can be inserted at a suitable position, then space is created for it by shifting the other elements one position up and inserts the desired element at the suitable position.
- This procedure is repeated for all the elements in the list.

INSERTION_SORT(A,N)

Where

A	– array containing values to be sorted
N	– size of Array

```
Step 1:Repeat step 2 to 7 for I:=2 to N
Step 2:     Set J:=I-1
Step 3:     Set X:=A[I]
Step 4:     Repeat step 5and 6 while X<A[J] and J>0
Step 5:           Set A[J+1] := A[J]
Step 6:           Set J:=J-1
Step 7:     Set A[J+1]:=X
Step 8: Return
```

Programming Languages

FIRST GENERATION OF PROGRAMMING LANGUAGE

1GL, is machine language. Machine language is a set of instructions and data that a computer's central processing unit can execute directly. Machine language statements are written in binary code, and each statement corresponds to one machine action.

SECOND GENERATION PROGRAMMING LANGUAGE

2GL, is assembly language. Assembly language is the human-readable notation for the machine language used to control specific computer operations. An assembly language programmer writes instructions using symbolic instruction codes that are meaningful abbreviations or mnemonics. An assembler is a program that translates assembly language into machine language.

THIRD GENERATION PROGRAMMING LANGUAGE

3GL uses a series of English-like words, that are closer to human language, to write instructions. Programs written in a high-level programming language must be translated into machine language by a compiler or interpreter. For ex: PASCAL, FORTRAN, COBOL, C and C++.

FOURTH GENERATION PROGRAMMING LANGUAGE

4GL, enables users to access data in a database. A very high-level programming language is often referred to as goal-oriented programming language because it is usually limited to a very specific application and it might use syntax that is never used in other programming languages.

For ex : SQL, NOMAD and FOCUS etc.

FIFTH GENERATION PROGRAMMING LANGUAGE

The fifth generation programming language or visual programming language provides a visual or graphical interface, called a visual programming environment, for creating source codes. Fifth generation programming allows people to interact with computers without needing any specialized knowledge.

For ex: Prolog and Mercury etc.

Software

Software is a set of programs, which is designed to perform a well-defined function. A program is a sequence of instructions written to solve a particular problem.

There are two types of software

- System Software
- Application Software

System Software

The system software is collection of programs designed to operate, control, and extend the processing capabilities of the computer itself.

Some examples of system software are Operating System, Compilers, Interpreter, Assemblers etc.

Features of system software are as follows:

- Close to system
- Fast in speed
- Difficult to design
- Difficult to understand
- Less interactive
- Smaller in size
- Difficult to manipulate
- Generally written in low-level language

Application Software

Application software products are designed to satisfy a particular need of a particular environment.

Examples of Application software are following:

- Payroll Software
- Student Record Software
- Inventory Management Software
- Income Tax Software
- Railways Reservation Software
- Microsoft Word

Features of application software are as follows:

- Close to user
- Easy to design
- More interactive
- Slow in speed
- Generally written in high-level language
- Easy to understand
- Easy to manipulate and use
- Bigger in size and requires large storage space

Program

An organized list of instructions that, when executed, causes the computer to behave in a predetermined manner. Without programs, computers are useless.

Programs can be writtern in different programming languages

- high-level languages : C , C++, BASIC, FORTRAN etc
- low-level languages *: assembly languages etc*

Note: C language can also be called as middle level language as it can perform high level as well as low level operation.

High Level Languages Vs Low Level languages

High Level Languages	**Low Level languages**
High-level languages are easy to learn.	Low-level languages are difficult to learn.
High0level languages are near to human languages.	Low-level languages are far from human languages.
Programs in high-level languages are slow in execution.	Programs in low-level languages are fast in execution.
Programs in high-level languages are easy to modify.	Programs in low-level languages are difficult to modify.
Knowledge of hardware is not required to write programs.	Low-level languages provide facility to write programs at hardware level.
These languages are normally used to write application programs	These languages are normally used to write hardware programs / Drivers etc.
For ex: C++, Basic , Fortran	For ex : Assembly Language