

Computer

Computer can be defined on an electronic device that accept data, process then at a high speed according to a set of instructions provide to it and produces the desired output. So computer is a programmable machine.

Generation of computer

Generation means variation between different hardware technologies

1st generation: (1946-1956)

First generation computers are made with the use of vacuum tubes. These computers used machine language for programming.

Disadvantages:

1. Occupied lot of space
2. Consumed lot of power
3. Produced lot of heat
4. Costly system

Example: ENIAC(Electronic numeric integrator and computer)

UNIVAC(Universal Accounting company)

2nd generation: (1957-1963)

The computer in which vacuum tubes were replaced by transistors are called second generation of computers these computers used assembly language for programming.

Advantages:

1. Less expensive
2. Consumed less power
3. Produced little heat
4. Less cost and work at higher speed

Example: IBM 7090, IBM 7094

3rd generation: (1964-1981)

The computers using the integrated circuits came to be known on the third generation of computers. These used high level language.

Example: IBM 370, Cyber 175

4th generation: (1982-1989)

The computers which were built with microprocessor are identified on the fourth generation computers these computers use VLSI chips for both cpu and memory.

Example: CRAY-2, IBM 3090

5th generation: (1990- till now)

Fifth generation computers are under development stage these computers use ULSI chips. ULSI chip contains thousands of components into a single IC. Aim of these computers is develop the artificial intelligence.

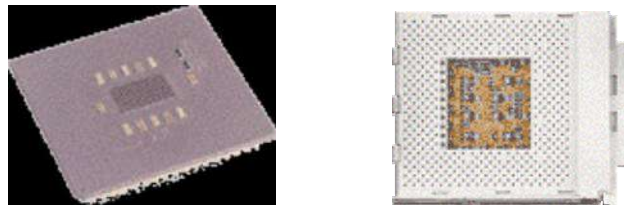
The Computer Hardware

A computer is made up of many parts:

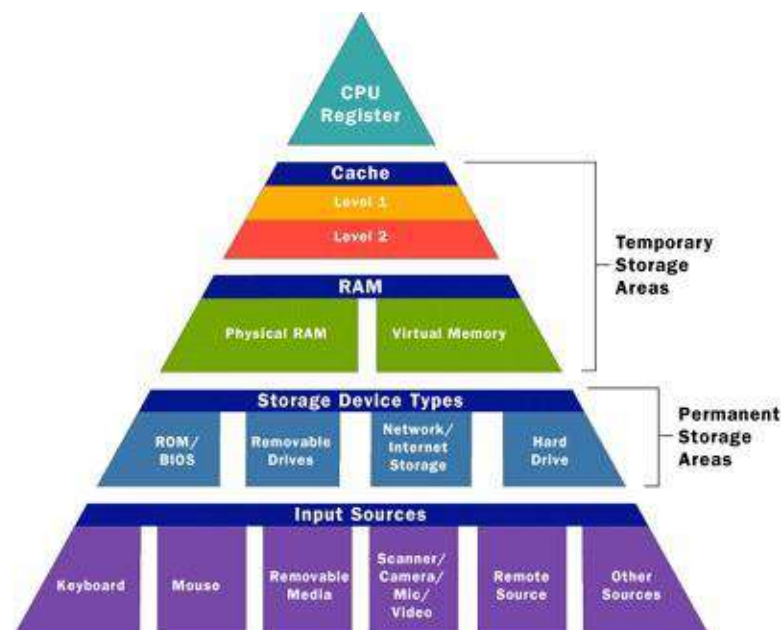
1. **Input/Output (I/O) devices** – These allow you to send information to the computer or get information from the computer.



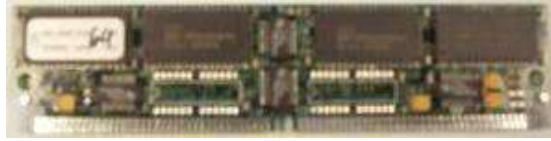
2. **Central Processing Unit** – CPU or Processor for short. The brain of a computer. Approximately 1.5 in X 1.5 in. Does all the computation/work for the computer.



3. **Memory** – Although memory is technically any form of electronic storage, it is used most often to identify fast, temporary forms of storage. Accessing the hard drive for information takes time. When the information is kept in memory, the CPU can access it much more quickly.



- a. **Random Access Memory** – RAM. Where information is stored temporarily when a program is run. Information is automatically pulled into memory, we cannot control this. RAM is cleared automatically when the computer is shutdown or rebooted. RAM is volatile (non-permanent).



- b. **Read Only Memory** – ROM. More permanent than RAM. Data stored in these chips is nonvolatile -- it is not lost when power is removed. Data stored in these chips is either unchangeable or requires a special operation to change. The BIOS is stored in the CMOS, read-only memory.



- c. **Hard Drive** – Where you store information permanently most frequently. This is also nonvolatile.



4. **Motherboard** – A circuit board that allows the CPU to interact with other parts of the computer.



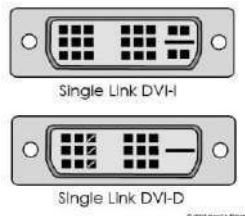
5. **Ports** – Means of connecting peripheral devices to your computer.
- a. **Serial Port** – Often used to connect a older mice, older external modems, older digital cameras, etc to the computer. The serial port has been replaced by USB in most cases. 9-pin connector. Small and short, often gray in color. Transmits data at 19 Kb/s.



- b. **Monitor Ports** – Used to connect a monitor to the computer. PCs usually use a VGA (Video Graphics Array) analog connector (also known as a D-Sub connector) that has 15 pins in three rows. Typically blue in color.



Because a VGA (analog) connector does not support the use of digital monitors, the Digital Video Interface (DVI) standard was developed.



LCD monitors work in a digital mode and support the DVI format. At one time, a digital signal offered better image quality compared to analog technology. However, analog signal processing technology has improved over the years and the difference in quality is now minimal.

- c. **Parallel Port** – Most often used to connect a printer to the computer. 25-pin connector. Long and skinny, often pink in color. Transmits data at 50-100 Kb/s.



- d. **USB Port** – Universal Serial Bus. Now used to connect almost all peripheral devices to the computer. USB 1.1 transmits data at 1.5 Mb/s at low speed, 12 Mb/s at full speed. USB 2.0 transmits data at 480 Mb/s.



- e. **Firewire/ IEEE 1394 Port** – Often found on Apple Computers. Often used with digital camcorders. Firewire transmits data at 400 Mb/s. Firewire 1394B (the new firewire) transmits data at 3.2 Gb/s.



- f. **PS/2 Port** - sometimes called a mouse port, was developed by IBM. It is used to connect a computer mouse or keyboard. Most computers come with two PS/2 ports.



- g. **Ethernet Port** – This port is used for networking and fast internet connections. Data moves through them at speeds of either 10 megabits or 100 megabits or 1 gigabit (1,000 megabits) depending on what speed the network card in the computer supports. Little monitor lights on these devices flicker when in use.



6. **Power Supply** – Gives your computer power by converting alternating current (AC) supplied by the wall connection to direct current (DC).



7. **Expansion Cards** – Used to add/improve functionality to the computer.
- a. **Sound Card** – Used to input and output sound under program control. Sound cards provide better sound quality than the built in sound control provided with most computers.



- b. **Graphics Card** – Used to convert the logical representation of an image to a signal that can be used as input for a monitor.



- c. **Network Card** – Used to provide a computer connection over a network. Transmit data at 10/100/1000 Mb/s.



8. **CD ROM** – A device used to read CD-ROMs. If capable of writing to the CD-ROM, then these are usually referred to as a 'burner' or CD-RW.



9. **DVD ROM** – A device that is used to read DVDs/CDs. If capable of writing to the DVD, then it is often referred to as a DVD-burner or a DVD-RW.



10. **Floppy Drive** – A device that is used to read/write to floppy diskettes.



11. **Fan** – Keeps your computer cool. If the inside of your computer becomes too hot, then the computer can overheat and damage parts.



12. **Heatsink** – Used to disperse the heat that is produced inside the computer by the CPU and other parts by increasing surface area.



13. **The little parts** – Capacitors – store energy, Resistors – allows a current through, Transistors – a valve which allows currents to be turned on or off.



14. **Case** – (Tower if standing upright.) What your motherboard, CPU, etc is contained in.



The three main components of a computer:

1. **CPU** – Central Processing Unit, coordinates all actions that occur in the system, executes program instructions.
2. **I/O Devices** – Input/Output devices, which allow you to obtain or display data.
3. **Memory** – Used to store information.

Bits and Bytes:

Bits:

The term bit is an acronym of **B**inary **D**igit. By definition, it is “a single digit in binary number scheme”, meaning it can take on one of two values: 0 and 1 (a binary condition). It is a mutually exclusive state: Something either isn’t (‘0’), or it is (‘1’). It is also the basic unit of information storage.

State 0: OFF

State 1: ON

Byte:

A pattern eight bits makes up a BYTE. A BYTE represents characters (letters, numbers & symbols).

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	1	1	3
0	0	0	0	0	1	0	0	4
0	0	0	0	0	1	0	1	5
0	0	0	0	0	1	1	0	6
0	0	0	0	0	1	1	1	7
0	0	0	0	1	0	0	0	8
0	0	0	0	1	0	0	1	9

- 8 bits are 1 **byte**
- 16 bits are 2 **bytes** = 1 **halfword**
- 32 bits are 4 **bytes** = 1 **word**
- 1024 bytes are 1 **kilobyte**
- 1024 kilobytes are 1 **megabyte**
- 1024 mega bytes are 1 **gigabyte**
- 1024 gigabytes are 1 **terabyte**
- 1024 terabytes are 1 **petabyte**
- 1024 petabytes are 1 **zettabyte**.

Programming Languages:

To write a computer program, a standard programming language is used. A programming language is composed of a set of instructions in a language understandable to the programmer and recognizable by a computer. Programming languages can be classified as high-level and low level.

Low-level programming languages were the first category of programming languages to evolve. Gradually, high-level languages were developed and put to use.

a) Low-level Languages:

A low-level computer programming language is one that is closer to the native language of the computer, which is 1's and 0's.

Machine Language:

Only computer can directly understand only its own machine language. Machine language is the natural language of a particular computer. It is defined by the hardware design of that computer. This is considered to be the *First Generation Language* (1GL).

Advantages:

The CPU directly understands machine instructions and hence no translation is required. Therefore, the computer directly starts execute the machine language instructions, and it takes less execution time.

Disadvantages:

- Difficult to use
- Machine dependent
- Error prone
- Difficult to debug and modify

Assembly Language:

When symbols as letters, digits, or special characters are employed for the operation, operand and other parts of instruction code. The representation is called an assembly language instruction. Such representations are known as mnemonic codes; they are used instead of binary codes. A program written with mnemonic codes forms an assembly language program. This is considered to be a *Second Generation Language* (2GL).

Advantages:

- Changes could be made easier and faster.
- Easier to read and write.
- Error checking is provided.

Disadvantages:

- Machine dependent.
- Programming is difficult and time consuming.
- The programmer should know all about the logical structure of the computer.

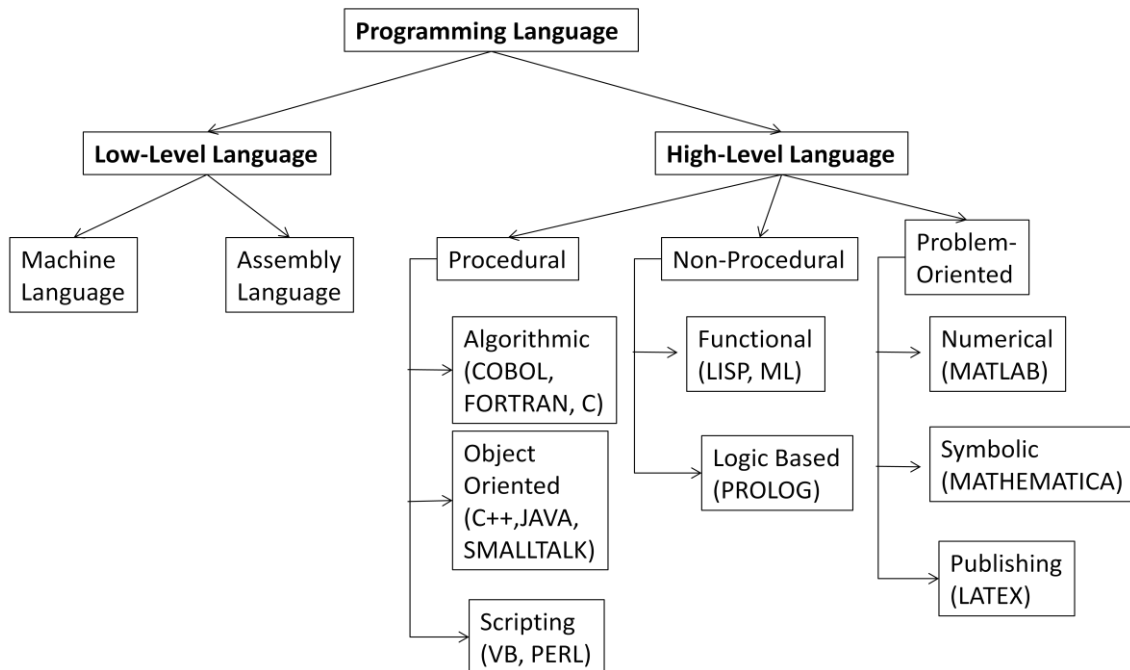


Fig: Programming Languages Classification

b) High-Level Languages:

Since a computer prefers machine language while a person wants to use a natural language special “high level computer languages have been developed to bridge this communication gap. A high level language like C, uses English words plus symbols and numbers. Very first two high level languages are FORTRAN and COBOL. This is considered to be a *Third Generation Language* (3GL).

Advantages:

- **Readability** Programs written in these languages are more readable than those written in assembly and machine languages.
- **Portability** High-level programming languages can be run on different machines with little or no change.
- **Easy debugging** error can be easily detected and removed.
- **Ease in the development of Software** Since commands of these programming languages are closer to the English language, software can be developed with ease.

Translation of high level language to machine language.

High level language

Assembly language

Machine language

```
sum=0
for(i=1;i<10;i++)
sum=sum+i;
```

```
mov sum,0
sub AX,AX
mov CX,10
top: add AX,CX
loop top
mov sum,AX
```

```
C70600000000
2 BC 0
B90A00
03C1
E2F9
A30000
```

i) Procedural Languages:**Algorithmic Languages:**

These are high-level languages designed for forming convenient expression of procedures, used in the solution of wide class of problems. In this language, the programmer must specify the steps the computer has to follow while executing a program. Some of languages that fall in the category are C, COBOL, and FORTRAN.

Object-oriented language:

The basic philosophy of object-oriented programming is to deal with objects rather than functions or subroutines as in strictly algorithmic languages. Objects are self-contained modules that contain data as well as the functions needed to manipulate the data within the same module. The difference affects the way a programmer goes about writing a program as well as how information is represented and activated in the computer. The most important object-oriented programming features are

- abstraction
- encapsulation and data hiding
- polymorphism
- inheritance
- reusable code

C++, JAVA, SMALLTALK, etc. are examples of object-oriented languages.

Scripting Languages: These languages assume that a collection of useful programs, each performing a task, already exists. It has facilities to combine these components to performing a complex task. A scripting language may thus be thought of as a glue language, which sticks a variety of components together. One of the earliest scripting languages is the UNIX shell. Now there are several scripting languages such as VB script and Perl.

ii) Problem-oriented languages

Problem-oriented Languages were designed to solve specific problems e.g. MATLAB is used to design circuit diagrams, and allowed the programmer to concentrate more on the problem rather than spending time learning the complex syntax of the language.

iii) Non-procedural Languages

Functional languages: These functional languages solve a problem by applying a set of functions to the initial variables in specific ways to get the answer. LISP, ML, etc. are examples of functional languages.

Logic Based Programming Language: A logic program is expressed as a set of atomic sentences, known as fact, and horn clauses, such as if-the rules. A query is then posed. The execution of the program now begins and the system tries to find out if the answer to the query is true or false. Such languages include PROLOG.

Software:

A set of computer programs are called software. Computer program is a series of instructions telling the computer what to do.

Types of software:

1. System software
2. Application software

1. System software:

- System software exists in the functioning of a computer system and includes the operating system, assembler, interpreter, compiler, linker and loader.
- Operating system is the interface between user applications and system hardware.
- Assembler is a translator that converse assembly language code into machine language.
- Interpreter converts source language program into executable code at once.
- Linker performs the important task of linking together several objects modules.
- The task of loading the linked object modules is performed by the loader.

System software includes three types of programs:

- **Operating System:** The combination of a particular hardware configuration and system software package is known as a computer system platform. System platforms are commonly termed as operating system (OS). Some common operating systems are DOS, UNIX, Mac, and Windows platform.
- **Language Translators:** These are interpreters and compilers for programs such as Pascal, BASIC, COBOL, C, and C++.
- **Common Utility Programs:** Communication tools, disk formatter, etc.

Examples: Language Translator, Operating System, Special Purpose Program, Utilities

2. Application software:

Application software is written to enable the computer to solve a specific data processing task. There are two categories of application software: pre-written software packages and user application programs.

A number of powerful application software packages that do not require significant programming knowledge have been developed. These are easy to learn and use compared to programming languages.

Examples:

- Database Management Software
- Spreadsheet Software
- Word processing, Desktop Publishing (DTP), and Presentation Software
- Multimedia Software
- Data Communication Software
- Statistical and operational research Software

Algorithms:

Computer Scientist 'Niklaus Wirth' stated that

$$\text{Programs} = \text{Algorithms} + \text{Data}$$

An algorithm is a part of the plan for the computer program; an algorithm is 'an effective procedure for solving problem in a finite number of steps'.

A sequential solution of any program that written in human language, called algorithm.

Algorithm is first step of the solution process, after the analysis of problem, programmer writes the algorithm of that problem.

Good Qualities of algorithm:

- Inputs and outputs should be defined precisely.
- Each step in algorithm should be clear and unambiguous.
- Algorithm should be most effective among many different ways to solve a problem.
- Algorithm should be complete or finiteness.
- Algorithm should be correctness.

Steps for algorithm:

- Identify the processes
- Identify the major decisions
- Identify the loops
- Identify the variables

Example Algorithm for Making a Tea:

1. Put the kettle in the stove.
2. Fill the kettle with water, sugar and tea powder.
3. Boil the water in the kettle.
4. Add milk to the decoction.
5. Filter the tea into cup.
6. Drink the tea.

1. Example Algorithm for Addition of two numbers:

Step 1: START
Step 2: Read a, b
Step 3: Sum = a + b
Step 4: PRINT Sum
Step 5: STOP

2. Example Algorithm for Biggest of two numbers:

Step 1: START
Step 2: Read a, b
Step 3: IF a>b Then PRINT "a is Big"
ELSE IF a<b Then PRINT "b is Big"
ELSE PRINT "both are equal"
Step 4: STOP

3. Example Algorithm for find Leap Year:

Step 1: START
Step 2: Read year
Step 3: IF year%4 == 0 Then PRINT "Leap year"
ELSE PRINT "NOT Leap Year"
Step 4: STOP

4. Example Algorithm for Increment of Employee salary:

Step 1: START
Step 2: Read salary
Step 3: IF salary is greater than 5000
 THEN step 4
 ELSE IF salary is equal to 5000
 Then step5
 ELSE step 6
Step 4: Salary = Salary + (0.05*Salary)
Step 5: Salary = Salary + 200
Step 6: Salary = Salary
Step 7: Print Salary
Step 8: STOP

5. Example Algorithm for Factorial of Number:

Step 1: START	Step 1: START
Step 2: Read n	Step 2: Read n
Step 3: [Initialize] i = 1, fact = 1	Step 3: [Initialize] fact = 1
Step 4: WHILE i<=n	Step 4: WHILE n>=1
BEGIN	BEGIN
fact = fact * i	fact = fact * n
i = i + 1	n = n - 1
END	END
Step 5: PRINT fact	Step 5: PRINT fact
Step 6: STOP	Step 6: STOP

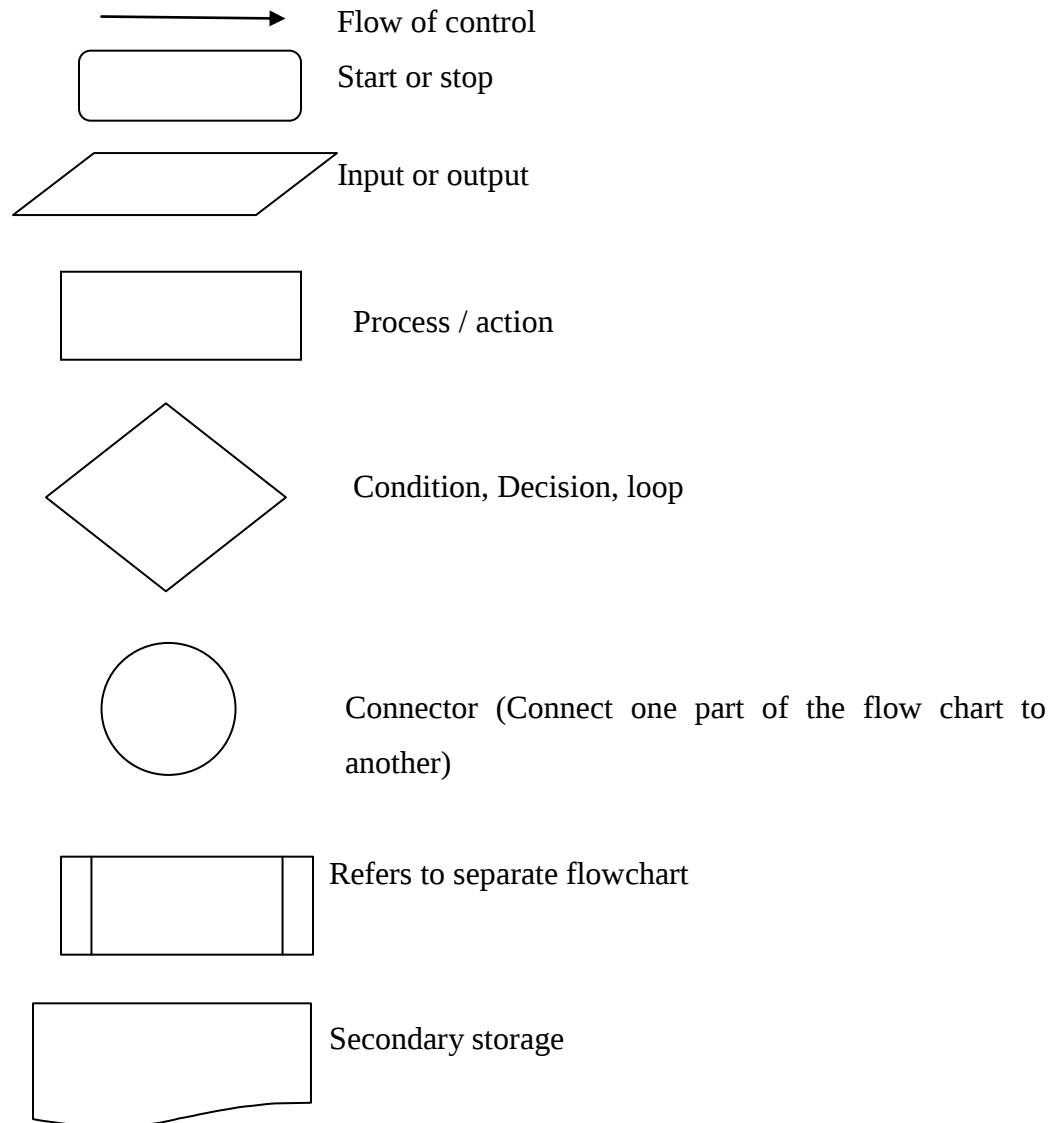
6. Example Algorithm for Swapping of Two Numbers:

Step 1: START	Step 1: START
Step 2: Read a, b	Step 2: Read a, b
Step 3: c = a	Step 3: a = a + b
Step 4: a = b	Step 4: b = a - b
Step 5: b = c	Step 5: a = a - b
Step 6: PRINT a, b	Step 6: PRINT a, b
Step 7: STOP	Step 7: STOP

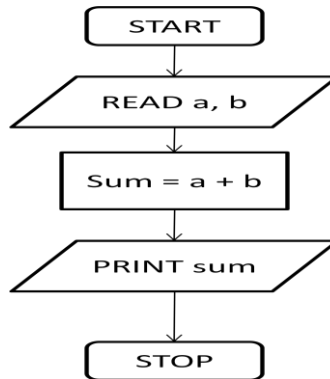
Flowchart:

A flowchart is a pictorial representation of an algorithm. It shows the logic of the algorithm and the flow of control. The flowchart uses symbols to represent specific actions and arrows to indicate the flow of control.

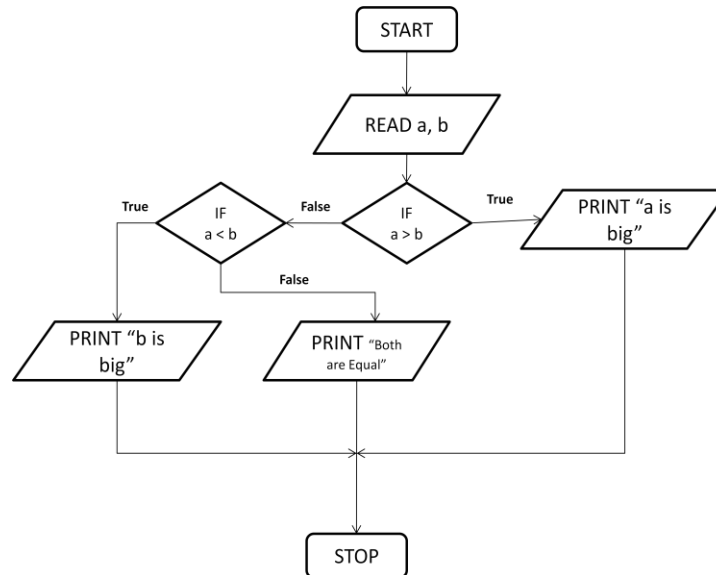
Flowchart symbols are



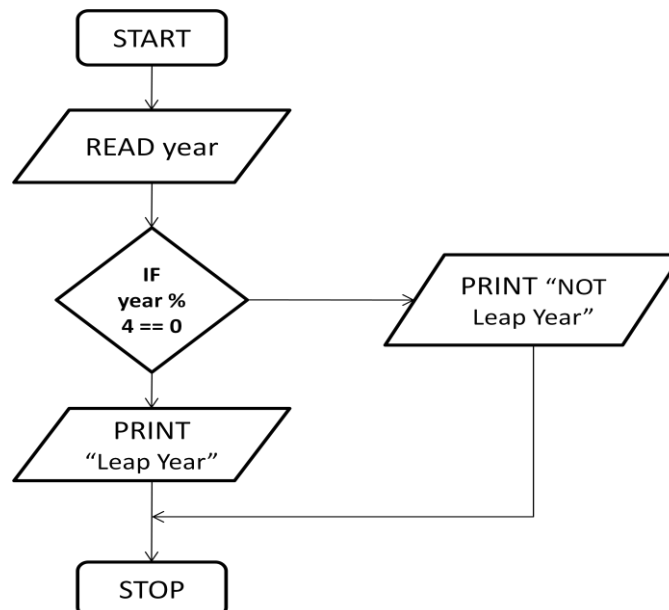
1. Example Flowchart for Addition of two numbers:



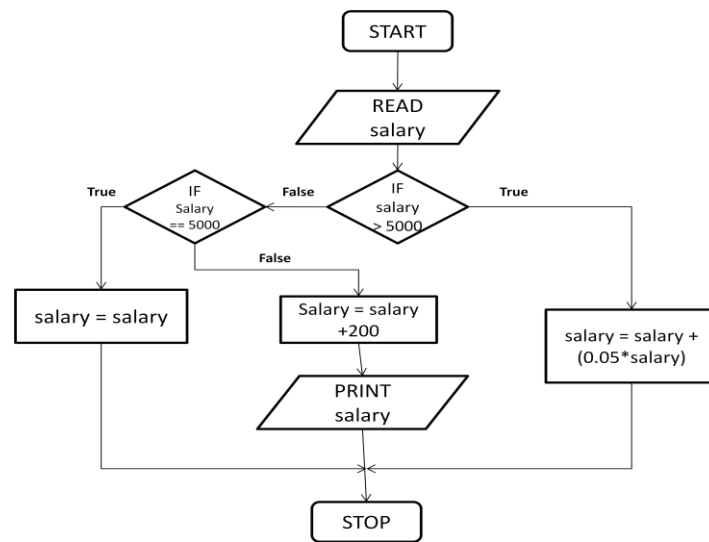
2. Example Flowchart for Biggest of two numbers:



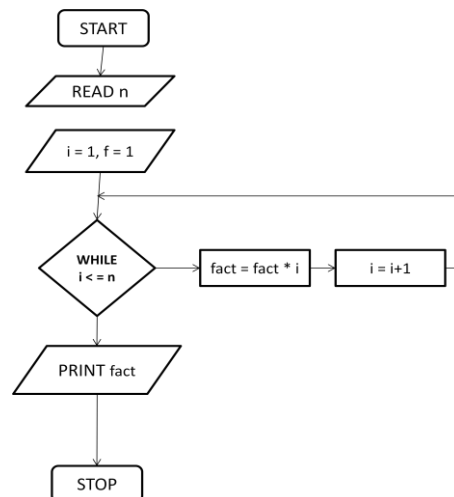
3. Example Flowchart for find Leap Year:



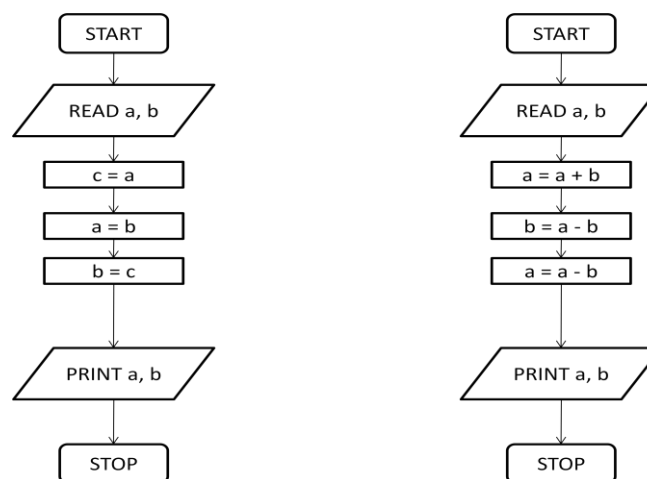
4. Example Flowchart for Increment of Employee salary:



5. Example Flowchart for Factorial of Number:



6. Example Flowchart for Swapping of Two Numbers:



The Software Development Process:

Software development Process is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

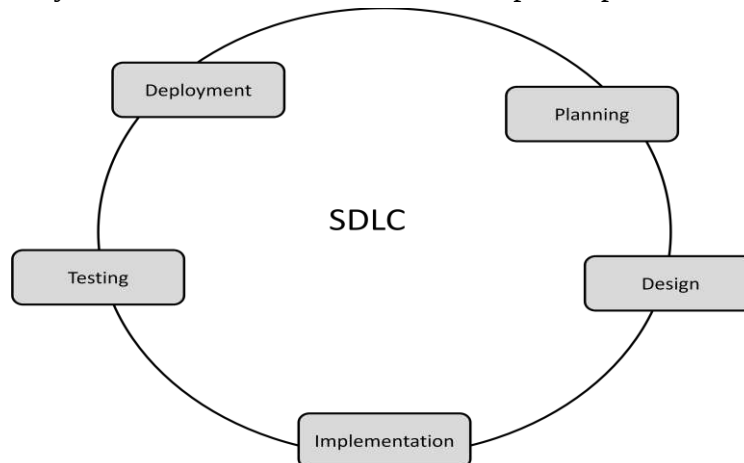


Fig: Software development Life Cycle

A typical Software Development life cycle consists of the following stages:

Stage 1: Planning and Requirement Analysis:

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Stage 2: Designing the product architecture:

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification. This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

Stage 3: Implementing the Product:

In this stage of SDLC the actual development starts and the product is implementing. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle. Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 4: Testing the Product:

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 5: Deployment in the Market and Maintenance:

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.