

Study Material
on
Computer Application
of
1st and 2nd semester Engineering Branch



Study Material on Computer Application of

1ST and 2nd semester of all Engineering Branches

Contents Written by

Mr. Laxmidhar Sethy, Sr. Lecturer Computer Science and Engineering,
Govt. Polytechnic Bhadrak, Odisha

Syllabus

Th.1b. COMPUTER APPLICATION (2nd sem)

1. COMPUTER ORGANISATION

Introduction to Computer Evolution of Computers Generation of Computers
Classification of Computers Basic Organisation of Computer (Functional Block diagram) Input Devices, CPU & Output Devices.
Computer Memory and Classification of Memory

2. COMPUTER SOFTWARE

Software concept, System software, Application software
Overview of Operating System Objectives and Functions of O.S ,
Types of Operating System: Batch Processing, Multiprogramming, Time Sharing OS Features of DOS,
Windows and UNIX
Programming Languages Compiler, interpreter
Computer Virus Different Types of computer virus Detection and prevention of Virus
Application of computers in different Domain

3. COMPUTER NETWORK AND INTERNET

Networking concept, Protocol, Connecting Media, Data Transmission mode Network Topologies, Types of Network Networking Devices like Hub, Repeater, Switch, Bridge, Router, Gateway & NIC Internet Services like E-Mail, WWW, FTP, Chatting, Internet Conferencing, Electronic Newspaper & Online Shopping
Different types of Internet connectivity and ISP

4. FILE MANAGEMENT AND DATA PROCESSING

Concept of File and Folder
File Access and Storage methods. Sequential, Direct, ISAM
Data Capture, Data storage Data Processing and Retrieval

5. PROBLEM SOLVING METHODOLOGY

Algorithm, Pseudo code and Flowchart
Generation of Programming Languages Structured Programming Language
Examples of Problem solving through Flowchart

6. OVERVIEW OF C PROGRAMMING LANGUAGE

Constants, Variables and Data types in C Managing Input and Output operations.
Operators, Expressions, Type conversion & Typecasting
Decision Control and Looping Statements (If, If-else, If-else-if, Switch, While, Do-while, For, Break, Continue & Goto) Programming Assignments using the above features.

7. ADVANCED FEATURES OF C

Functions and Passing Parameters to the Function (Call by Value and Call by Reference) Scope of Variables and Storage Classes Recursion Function and Types of Recursion One Dimensional Array and Multidimensional Array String Operations and Pointer, Pointer Expression and Pointer Arithmetic Programming Assignments using the above features.
Structure and Union (Only concepts, No Programming).

Contents

<u>Sl. No</u>	<u>Chapter Name</u>	<u>Page No.</u>
1	COMPUTER ORGANISATION	5-31
2	COMPUTER SOFTWARE	32-58
3	COMPUTER NETWORK AND INTERNET	59-87
4	FILE MANAGEMENT AND DATA PROCESSING	88-96
5	PROBLEM SOLVING METHODOLOGY	97-112
6	OVERVIEW OF C PROGRAMMING LANGUAGE	113-159
7	ADVANCED FEATURES OF C	160-180

Chapter-1

Computer Organisation

1.1 Introduction to Computer

1.2 Evolution of Computers

1.3. Generation of Computers

1.3.i 1st Generation of computer

1.3.ii. 2nd Generation of computer

1.3.iii. 3rd Generation of Computer

1.3 .iv. 4th Generation of Computer

1.3.v. 5th Generation of Computer

1.4 Classification of Computers

4.i. Super Computer

4.ii. Mainframe Computer

4.iii. Mini Computer

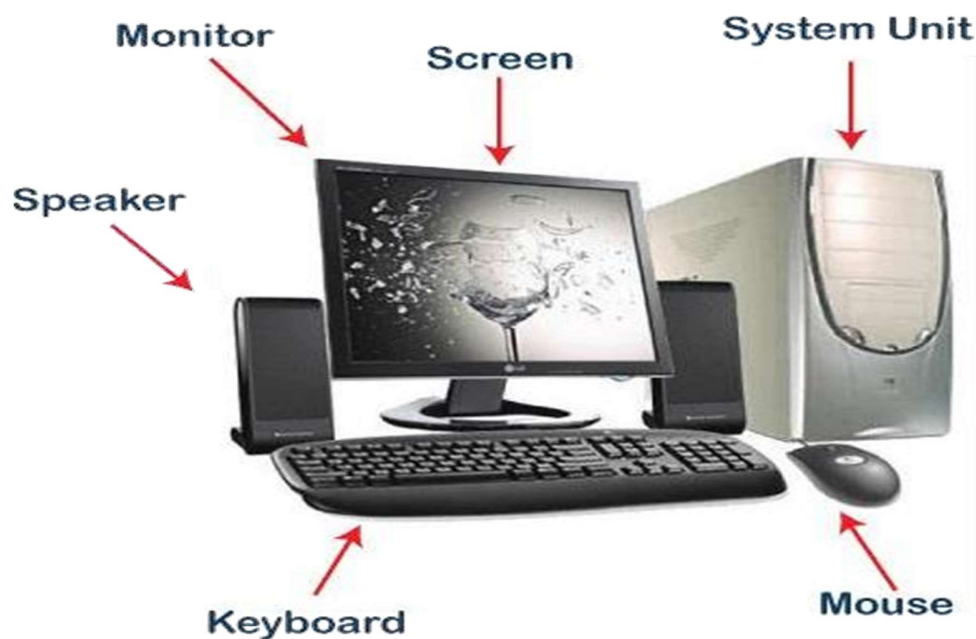
5.iv. Micro Computer

1.4 Basic Organisation of Computer (Functional Block diagram) Input Devices, CPU & Output Devices.

1.5 Computer Memory and Classification of Memory

1. INTRODUCTION TO COMPUTER

- Computer is an electronic device that operates (works) under the control of programs stored in its own memory unit.
- A computer is an electronic machine that processes raw data to give information as output.
- An electronic device that accepts data as input, and transforms it under the influence of a set of special instructions called Programs, to produce the desired output (referred to as Information).



Explanations:

- A computer is described as an electronic device because; it is made up of electronic components and uses electric energy (such as electricity) to operate.
- A computer has an internal memory, which stores data & instructions temporarily awaiting processing, and even holds the intermediate result (information) before it is communicated to the recipients through the Output devices.
- It works on the data using the instructions issued, means that, the computer cannot do any useful job on its own. It can only work as per the set of instructions issued.
- A computer will accept data in one form and produce it in another form. The data is normally held within the computer as it is being processed.

Program:

- A computer Program is a set of related instructions written in the language of the computer & is used to make the computer perform a specific task (or, to direct the computer on what to do).
- A set of related instructions which specify how the data is to be processed.
- A set of instructions used to guide a computer through a process.

Data:

- Data is a collection of raw facts, figures or instructions that do not have much meaning to the user.
- Data may be in form of numbers, alphabets/letters or symbols, and can be processed to produce information.

Types of Data

There are two types/forms of data:

a). Digital (discrete) data:

Digital data is discrete in nature. It must be represented in form of numbers, alphabets or symbols for it to be processed by a computer. Digital data is obtained by counting. E.g. 1, 2, 3...

b). Analogue (continuous) data:

- Analogue data is continuous in nature. It must be represented in physical nature in order to be processed by the computer.
- Analogue data is obtained by measurement. E.g. Pressure, Temperature, Humidity, Lengths or currents, etc. The output is in form of smooth graphs from which the data can be read.

Data Processing:

- It is the process of collecting all items of data together & converting them into information.
- Processing refers to the way the data is manipulated (or handled) to turn it into information.
- The processing may involve calculation, comparison or any other logic to produce the required result. The processing of the data usually results in some meaningful information being produced.

Information:

- Information is the data which has been refined, summarized & manipulated in the way you want it, or into a more meaningful form for decision-making.
- The information must be accurate, timely, complete and relevant.

2. EVOLUTION OF COMPUTERS

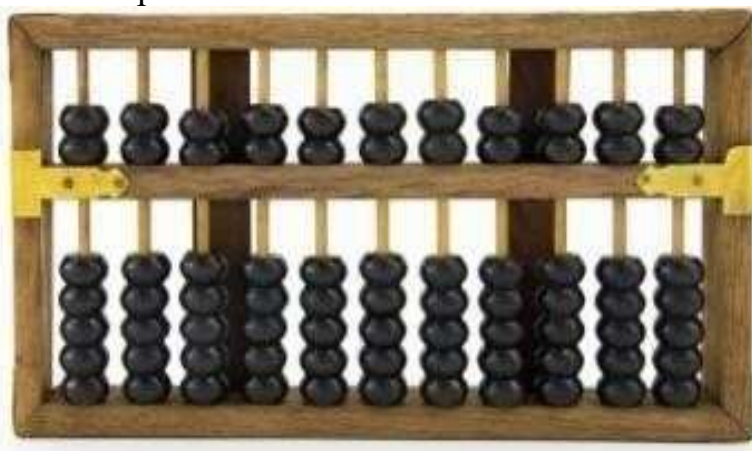
Computer evolution refers to the change in computer technology right from the time computers were first used to the present. As computing evolves to higher system levels, so its design also changes, from technical to socio-technical design.

The series of the evolution of computers are given below.

- Abacus
- Napier's Bones
- Slide Rule
- Pascaline
- Difference engine
- Jacquard Loom
- ABC
- UNIVAC – I

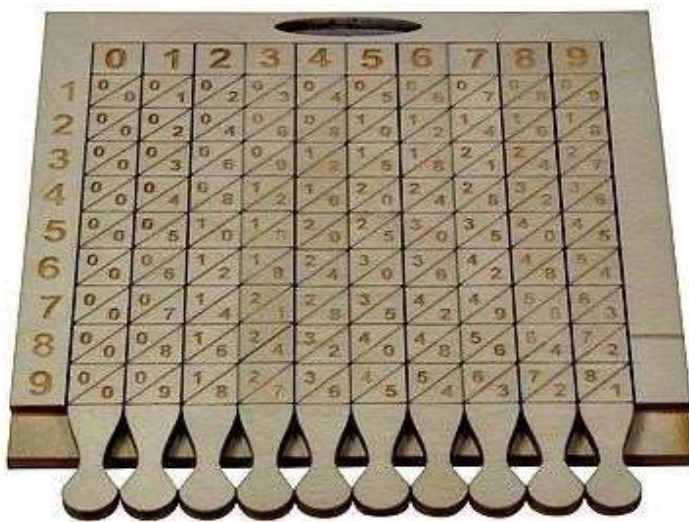
Abacus

- The present day computers are a result of an evolutionary process which started way back in 200 B.C. when Egyptian used a machine which is an early form of Abacus.
- However the present form of Abacus is attributed to the Chinese and Japanese.
- This is a machine, which was used for addition, subtraction, multiplication and division operation.



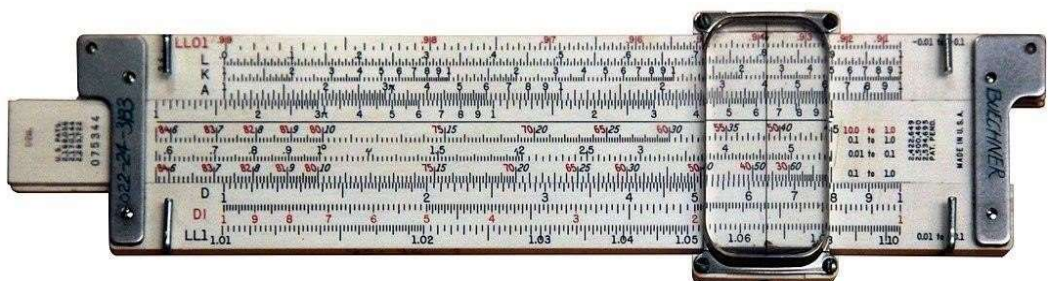
Napier's bone

- Napier's bones is a manually-operated calculating device created by John Napier in 1614.
- The rods of this device were sometimes made of ivory and resembled animal bones.
- The user was able to multiply number by manipulating the rods.



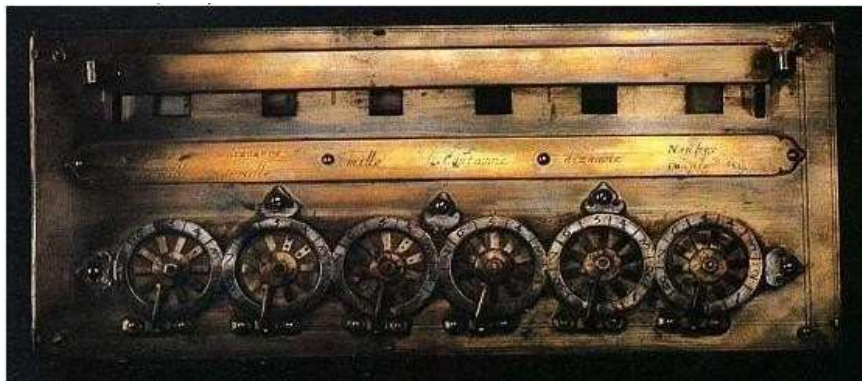
Slide Rule

- The **slide rule** is a mechanical computer developed by William Oughtred in 1622.
- The slide rule is used primarily for multiplication and division, and also for functions such as exponents, roots, logarithms, and trigonometry, and typically not for addition or subtraction.



Pascaline

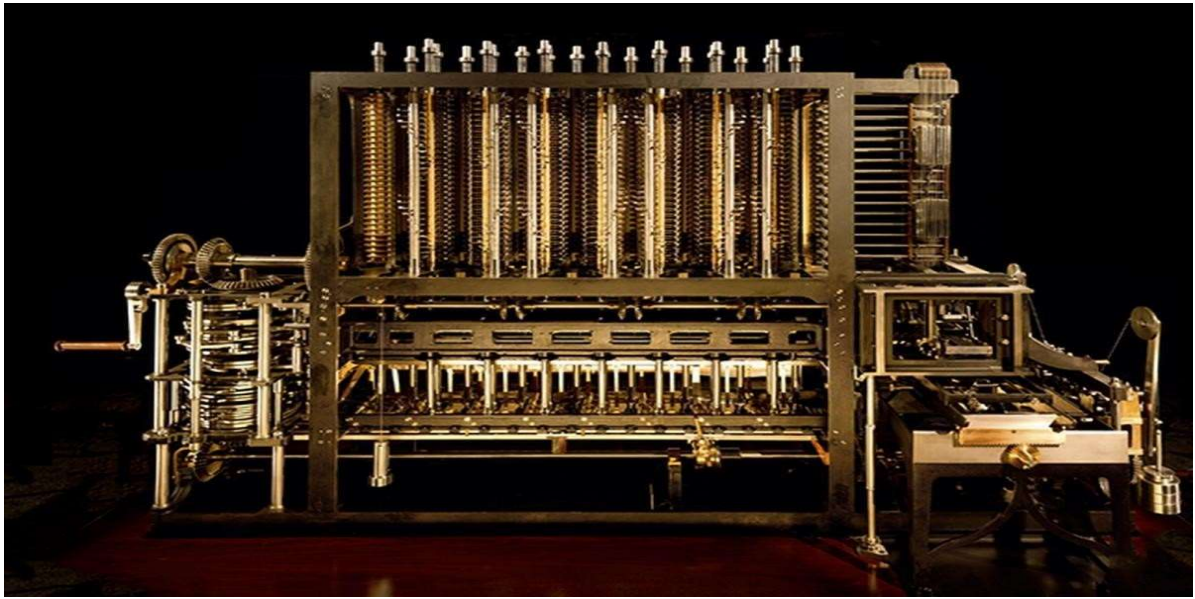
- In 1645 a device known as Pascaline was invented by French mathematician Blaise Pascal.
- The machine was also used per addition and subtraction purpose.
- The device was operated by dialing a set of wheels.
- In 1671 Leibniz improved on Pascal's adding machine and invented the Leibniz's Calculator.



Pascaline

Difference engine

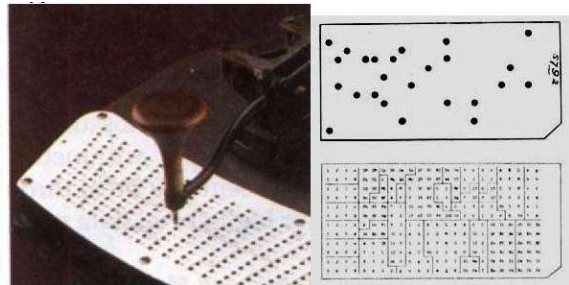
- In 1822 Charles Babbage invented a Difference Engine.
- The purpose of this device was to calculate the roots of polynomial equations and prepare astronomy table for the British Navy.
- He upgraded this to, invent an Analytical engine, which could store program instructions initially coded on punched cards and subsequently shared internally.
- Therefore Charles Babbage is known as the father of computers.



Difference Engine

Punched card equipment

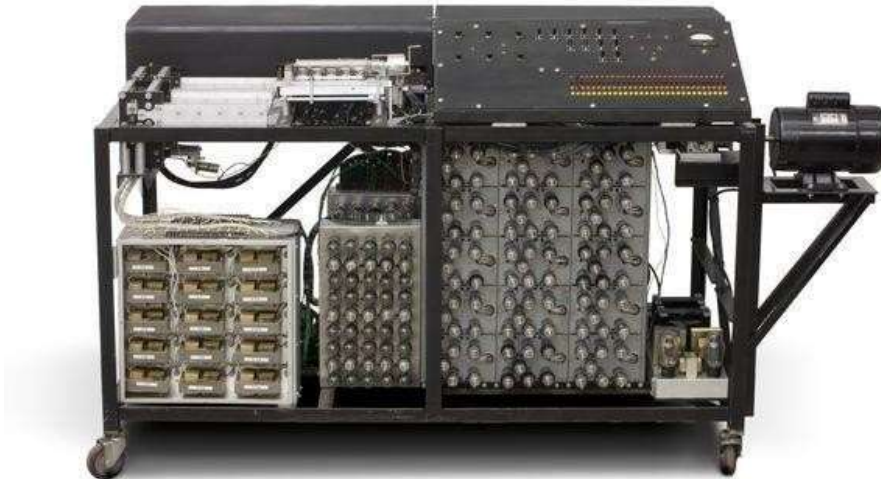
- In 1890 Dr. H. Hollerith developed punched card equipment.
- This equipment read the holes punched in the card and mechanically performed the statistical analysis.



ABC (Atanasoff-Berry Computer)

- The first pure electronic computer was invented by J. V. Atanasoff and C. Berry which is known as Atanasoff-Berry Computer or ABC.
- It used vacuum tubes for both data storage and data computation.

- Subsequently Electronic Numerical Integrator and Calculator (ENIAC) was designed and accepted as the general purpose computer



UNIVAC:

- In 1945 John Von Neumann first gave the idea of sharing the same internal memory for storing both data and instruction, which was subsequently adopted in every computer organization.
- On this principle subsequently Universal Automatic Computer (UNIVAC-1) was invented



3. GENERATION OF COMPUTERS

Computers are devices that accomplish tasks or calculations in accordance to a set of directions, or programs. The first fully electronic computers, introduced in the 1940s, were voluminous devices that required teams of people to handle. In comparison to those new machines, today's computers are astounding. They are not only thousands of times more expeditious, but also they can fit on your desk, on your lap, or even in your pocket. Computers are such an integral part of our everyday life now most people take them for granted.

Computers work through an interaction of hardware and software. The whole picture of the computer goes back to decades. However there are five apparent generations of computers. Each generation is defined by a paramount technological development that changes necessarily how computers operate – leading to more compressed, inexpensive, but more dynamic, efficient and booming machines.

There are five generations of computers.

FIRST GENERATION

- ***Introduction:***

1940-1956 is the period of first generation computer.

First generation computers were based on vacuum tubes.

- ***Advantages:***

1. It made use of vacuum tubes which are the only electronic component available during those days.

- ***Disadvantages:***

1. These were very big in size, weight was about 30 tones.
2. These computers were based on vacuum tubes.
3. These computers were very costly.
4. It could store only a small amount of information due to the presence of magnetic drums.
5. As the invention of first generation computers involves vacuum tubes, so another disadvantage of these computers was, vacuum tubes require a large cooling system.
6. Very less work efficiency.
7. Limited programming capabilities and punch cards were used to take inputs.
8. Large amount of energy consumption.

9. Not reliable and constant maintenance is required.

- ***Few Examples are:***

1. ENIAC(Electronic Numerical Integrated and Computer)
2. EDVAC(Electronic Discrete Variable Automatic Computer)
3. UNIVAC(Universal Automatic Computer)
4. IBM-701(International Business Machine)
5. IBM-650

SECOND GENERATION

- ***Introduction:***

1. 1956-1963 is the period of second-generation computer.
2. Second generation computers were based on Transistor instead of vacuum tubes.

- ***Advantages:***

1. Due to the presence of transistors instead of vacuum tubes, the size of electron component decreased. This resulted in reducing the size of a computer as compared to first generation computers.
2. Less energy and not produce as much heat as the first generation.
3. Assembly language and punch cards were used for input.
4. Low cost than first generation computers.
5. Better speed than first generation of computers.
6. Better portability as compared to first generation

- ***Disadvantages:***

1. A cooling system was required.
2. Constant maintenance was required.
3. Only used for specific purposes.

- ***Few Examples are:***

1. Honeywell 400
2. IBM 7094
3. CDC 1604(Control Data Corporation)
4. CDC 3600
5. UNIVAC 1108

THIRD GENERATION

- ***Introduction:***

1. 1964-1971 is the period of third generation computer.
2. These computers were based on Integrated circuits.
3. IC was a single component containing number of transistors.

- **Advantages:**

1. These computers were cheaper as compared to second-generation computers.
2. They were fast and reliable.
3. Use of IC in the computer provides the small size of the computer.
4. IC not only reduce the size of the computer but it also improves the performance of the computer as compared to previous computers.
5. This generation of computers has big storage capacity.
6. Instead of punch cards, mouse and keyboard are used for input.
7. They used an operating system for better resource management and used the concept of time-sharing and multiple programming.

- **Disadvantages:**

1. IC chips are difficult to maintain.
2. The highly sophisticated technology required for the manufacturing of IC chips.
3. Air conditioning is required.

- **Few Examples are:**

1. PDP-8(programmed data Processor)
2. PDP-11
3. IBM 360
4. IBM 370

FOURTH GENERATION

- **Introduction:**

1. 1971-1990 is the period of fourth generation computer.
2. This technology is based on Microprocessor.
3. A microprocessor is used in a computer for any logical and arithmetic function to be performed in any program.
4. Graphics User Interface (GUI) technology was exploited to offer more comfort to users.

- **Advantages:**

1. Fastest in computation and size get reduced as compared to the previous generation of computer.
2. Heat generated is negligible.
3. Small in size as compared to previous generation computers.
4. Less maintenance is required.
5. All types of high-level language can be used in this type of computers.

- **Disadvantages:**

1. The Microprocessor design and fabrication are very complex.
2. Air conditioning is required in many cases due to the presence of ICs.
3. Advance technology is required to make the ICs.

- ***Few Examples are:***

1. IBM 4341
2. DEC 10(Digital Equipment Corporation)
3. STAR 1000(String Array)
4. CRAY(Seymour Cray)

FIFTH GENERATION

- ***Introduction:***

1. The period of the fifth generation in 1990-onwards.
2. The aim of the fifth generation is to make a device which could respond to natural language input and are capable of learning and self-organization.
3. This generation is based on ULSI(Ultra Large Scale Integration) technology resulting in the production of microprocessor chips having ten million electronic component.
4. This generation is based on artificial intelligence.

- ***Advantages:***

5. It is more reliable and works faster.
6. It is available in different sizes and unique features.
7. It provides computers with more user-friendly interfaces with multimedia features.

- ***Disadvantages:***

8. These computers are having complex tool.
9. They may make the human brains dull and doomed.

- ***Few Examples are:***

Desktop
Laptop
Tablet
Mobile

4. CLASSIFICATION OF COMPUTERS

The computer systems can be classified on the following basis:

1. On the basis of data handling.
2. On the basis of size.

Classification on the basis of data handling

1. **Analog** : An analog computer is a form of computer that uses the continuously-changeable aspects of physical fact such as variation in temperature, pressure, speed, voltage etc. Mostly used in industries.
Ex-Speedometer, Thermometer, voltmeter.
2. **Digital** : A computer that performs calculations and logical operations with quantities represented as digits, usually in the binary number system of “0” and “1”, “Computer capable of solving problems by processing information expressed in discrete form.
Ex-Calculators, Digital watches, desktop, smartphone.
3. **Hybrid** : A computer that processes both analog and digital data, Hybrid computer is a digital computer that accepts analog signals, converts them to digital and processes them in digital form.
Ex-ECG(Electrocardiography), Ultrasound Machine, fuel dispenser.

Classification on the basis of size

1. Super computers :

- The super computers are the most high performing system.
- A supercomputer is a computer with a high level of performance compared to a general-purpose computer.
- The actual Performance of a supercomputer is measured in FLOPS instead of MIPS.
- All of the world’s fastest 500 supercomputers run Linux-based operating systems.
- Additional research is being conducted in China, the US, the EU, Taiwan and Japan to build even faster, more high performing and more technologically superior supercomputers.
- Supercomputers actually play an important role in the field of computation, and are used for intensive computation tasks in various fields, including weather forecasting, climate research, oil and gas exploration, nuclear research, military agencies and physical simulations

Ex- PARAM, jaguar, roadrunner, Blue gene.

2. Mainframe computers :

- These are commonly called as big iron, they are usually used by big organisations for bulk data processing such as statistics, census data processing, transaction

processing and are widely used as the servers as these systems has a higher processing capability as compared to the other classes of computers,

- The processing speed is very fast.
 - Can handle multiple inputs at same time.
 - Redundancy, Can withstand failure of a part without affecting the function of rest of the computer.
 - Always available, as once started they will remain on for rest of the time.
 - Reliability.
 - Mainframes cannot be used as a normal computer, because they are made for specific task only.
 - It requires a special OS to run.
 - Are very expensive.
 - Mainly used for commercial purposes like transaction processing.
- Ex: IBM z Series, System z9 and System z10 servers.

3. Mini computers :

- Mini Computer, as name suggests, is a type of computer that offers most features and capabilities that large computer generally offers.
- It generally supports multiple users at a time so one can say that it is a multiprocessing system.
- It is a smaller computer designed for business applications and services, and also can do time-sharing, batch processing, online processing, etc

Ex-IBM 800, PDP11, Honeywell200.

4. Micro computers :

- Micro Computer, as name suggests, is a personal computer that is specially designed for personal use and generally consists of single chip that is CPU, data memory, I/O buses, etc.
- It can be used by one person at a time. Its type includes tablet and smartphone microcomputers, desktop micro computers, workstation and server microcomputers, etc.

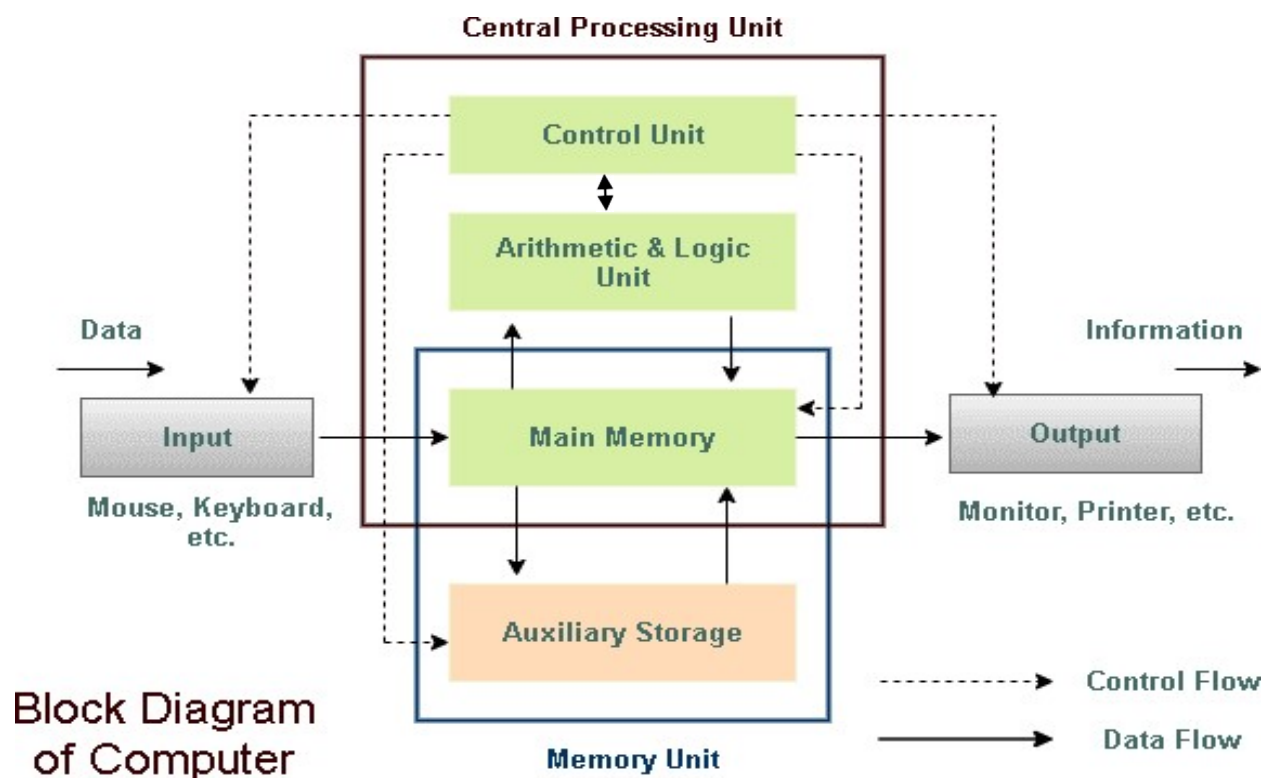
Example : Laptop, desktop, IBM-PC, etc

5. BASIC ORGANISATION OF COMPUTER(FUNCTIONAL BLOCK DIAGRAM)

A digital computer is considered to be a calculating device that can perform arithmetic operations at enormous speed. It is defined as a device that operates upon information/data. To be able to process data the computer is made of various functional units to perform its specified task.

Functionally a digital computer can be divided into four units.

1. Input unit
2. Central Processing Unit(CPU)
3. Memory Unit and
4. Output Unit



Input Devices:

Computers need to receive data and instruction in order to solve any problem. Therefore, we need to input the data and instructions into the computers. The input unit consists of one or more input devices. Keyboard is the one of the most commonly used input device. All the input devices perform the following functions.

- Accept the data and instructions from the outside world
- Convert it to a form that the computer can understand
- Supply the converted data to the computer system for further processing.

Examples of input devices:

Mouse, scanner, Microphone, OMR, Scanner, Joystick etc.

Output Devices:

The output unit of a computer provides the information and results of a computation to outside world.

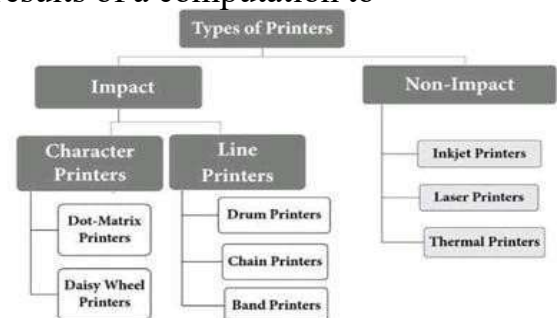
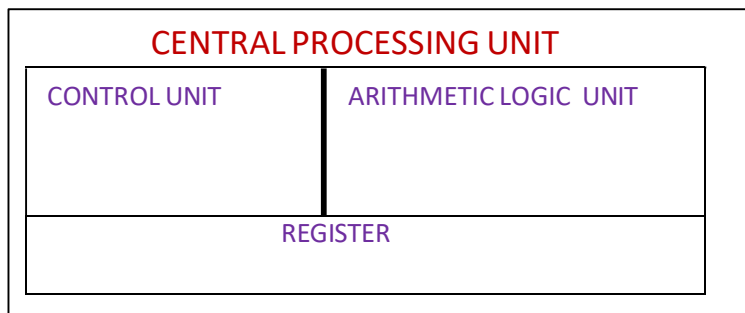
Examples of Output Devices:

Monitor, Printer, Speaker, Plotter, Projector Headphone etc.

Central processing unit:

The Control Unit(CU) and Arithmetic Logic Unit(ALU) of the computer are together known as the central Processing Unit(CPU).

- It is the brain of computer, it is also known as system unit.
- It performs all calculations.
- It takes all decisions.
- It controls all units of the computer.



CU-

- It is responsible for coordinating the activities like transfer of data, instruction etc. Between various sub units.
- It issues control signal to various units to carry out the coordination task.
- The control unit is generally referred as the central nervous system of the computer that control and synchronizes it's working.

ALU-

It is responsible for performing all arithmetic and logic operation that takes place inside a computer.

Register-

These are very small and high speed memory.

While performing these operations the ALU takes data from the temporary storage area inside the CPU named register.

Memory Unit(Storage Unit):

The storage unit of the computer holds data and instructions. It preserves the intermediate and final results before these are sent to the output devices. The various storage devices of a computer system are divided into two categories.

1. Primary Memory or Main Memory
2. Secondary Memory or Auxiliary Memory

Primary Memory:

- It is also known as main memory of a computer.
- It is a temporary or volatile memory used by the computer to store data and instruction during processing.
- This memory is generally used to hold the program being currently executed in the computer.

- The cost of primary memory is more compared to the secondary storage therefore most computers have limited primary storage capacity.

Secondary Memory:

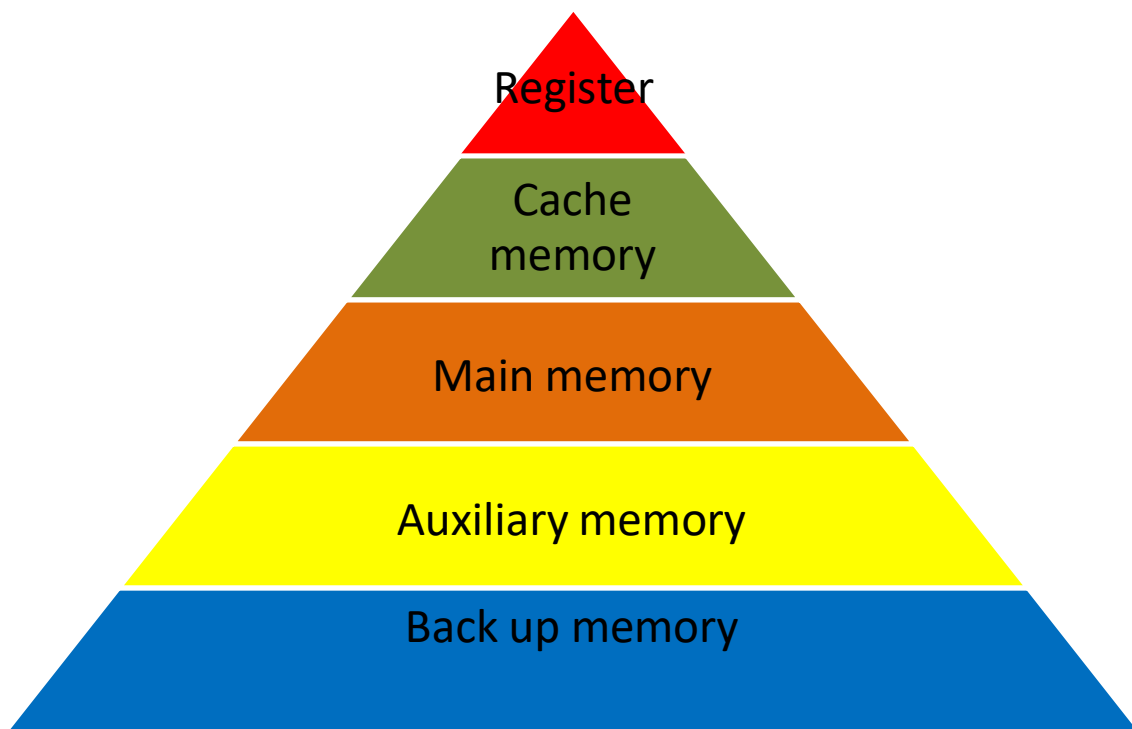
- It is also known as auxiliary memory.
- It is a permanent memory where we store data or instruction permanently.
- The programs that you run on the computer are first transferred to the primary memory before it is actually run.
- The secondary memory is slower and cheaper than the primary memory.
- The size of secondary memory is very large as compare to primary memory.

6. COMPUTER MEMORY AND CLASSIFICATION OF MEMORY

- Computer Memory is the part of the system used to store data and programs temporarily or permanently depending on the requirement and type of memory.
- The different classes of memory used in a computer-
 - *Register
 - *Cache memory
 - *Main Memory or Primary memory
 - *Secondary memory or Auxiliary memory
 - *Backup Memory

Memory hierarchy:

- The memory hierarchy design in a computer system mainly includes different storage devices.
- The following memory hierarchy diagram is a hierarchical pyramid for computer memory.



1. Register Memory:

- It is the high speed memory integrated inside CPU.
- It consist of a number of flip flops arranged in certain manner.
- This is a small capacity memory used for storing data or instruction temporarily during execution of an instruction.
- Some special purpose register such as MAR,MDR,IR,PC,accumulator

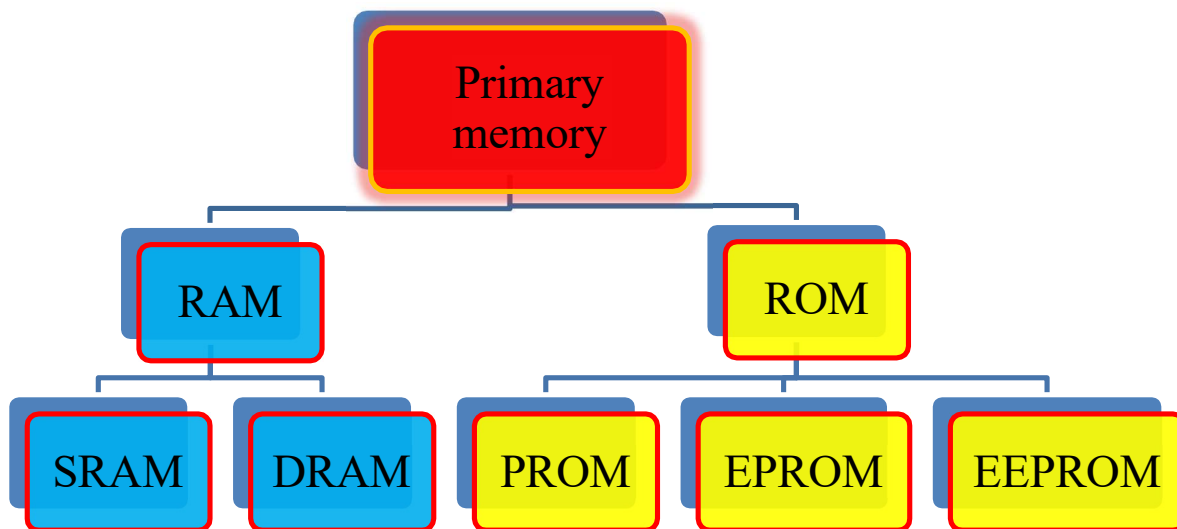
2. Cache Memory:

- The next memory in the memory hierarchy is a cache memory.
- This is a small memory situated between the CPU and main memory.
- The purpose of this memory is to hold or store frequently needed instruction or data from the main memory location during the execution process.
- When processor needs to read from or write to allocation in main memory it first check whether a copy that data is in the cache.



3. Primary memory or main memory:

- This is a memory which is used to store data and instruction during the execution of program.
- This memory directly interacts with the CPU during the execution process.



RAM:

- RAM(Random Access Memory) is a part of computer's Main Memory which is directly accessible by CPU.
- RAM is used to Read and Write data into it which is accessed by CPU randomly.
- RAM is volatile in nature, it means if the power goes off, the stored information is lost.
- RAM is used to store the data that is currently processed by the CPU.

RAM are available in two form:

1. SRAM(Static RAM)
2. DRAM(Dynamic RAM)

SRAM:(Static Random access memory)

- Data is stored in transistors and requires a constant power flow. Because of the continuous power, SRAM doesn't need to be refreshed to remember the data being stored.
- SRAM is called static as no change or action i.e. refreshing is not needed to keep the data intact. It is used in cache memories.

DRAM: (Dynamic random access memory)

- DRAM is slower than SRAM.
- Data is stored in capacitors. Capacitors that store data in DRAM gradually discharge energy, no energy means the data has been lost. So, a periodic refresh of power is required in order to function.
- DRAM is called dynamic as constant change or action i.e. refreshing is needed to keep the data intact. It is used to implement main memory.

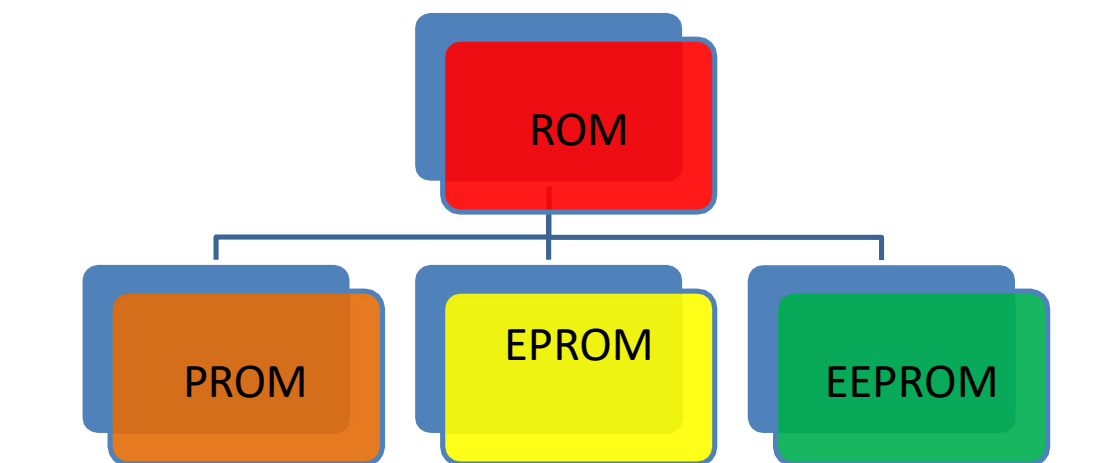
Difference between SRAM and DRAM:

SL NO.	SRAM	DRAM
1	Transistors are used to store information in SRAM.	Capacitors are used to store data in DRAM.
2	Capacitors are not used hence no refreshing is required.	To store information for a longer time, contents of the capacitor needs to be refreshed periodically.
3	SRAM is faster as compared to DRAM.	DRAM provides slow access speeds.
4	These are expensive.	These are cheaper.
5	These are used in cache	These are used in main memories.

ROM:

Read Only Memory (ROM) is a type of memory where the data has been pre recorded. Data stored in ROM is retained even after the computer is turned off (non-volatile).

Types of ROM:



1. **Programmable ROM**, where the data is written after the memory chip has been created. It is non-volatile.
2. **Erasable Programmable ROM**, where the data on this non-volatile memory chip can be erased by exposing it to high-intensity UV light.
3. **Electrically Erasable Programmable ROM**, where the data on this non-volatile memory chip can be electrically erased and reprogrammed using pulsed voltage.

Difference between RAM and ROM

Sl No	RAM	ROM
1	RAM is a volatile memory which could store the data as long as the power is supplied.	ROM is a non-volatile memory which could retain the data even when power is turned off.
2	Used to store the data that has to be currently processed by CPU temporarily.	It stores the instructions required during bootstrap of the computer.
3	Data stored in RAM can be read and write.	Data stored in ROM can only be read.
4	The CPU can access the data stored on it.	The CPU cannot access the data stored on it unless the data is stored in RAM.
5	Used in CPU Cache, Primary memory.	Firmware, Micro-controllers
6	It is a high speed memory	It is much slower than the RAM
7	costlier	Cheaper than RAM

4. Auxiliary memory:

- It is also known as **external memory** and **secondary memory**.
- It is a non-volatile device that holds data until it is deleted or overwritten.
- Secondary memory is not accessed directly by the Central Processing Unit(CPU). Instead, data accessed from a secondary memory is first loaded into Random Access Memory(RAM) and is then sent to the Processing Unit.

Some popular auxiliary memory devices:

*Floppy disk

*Hard disk

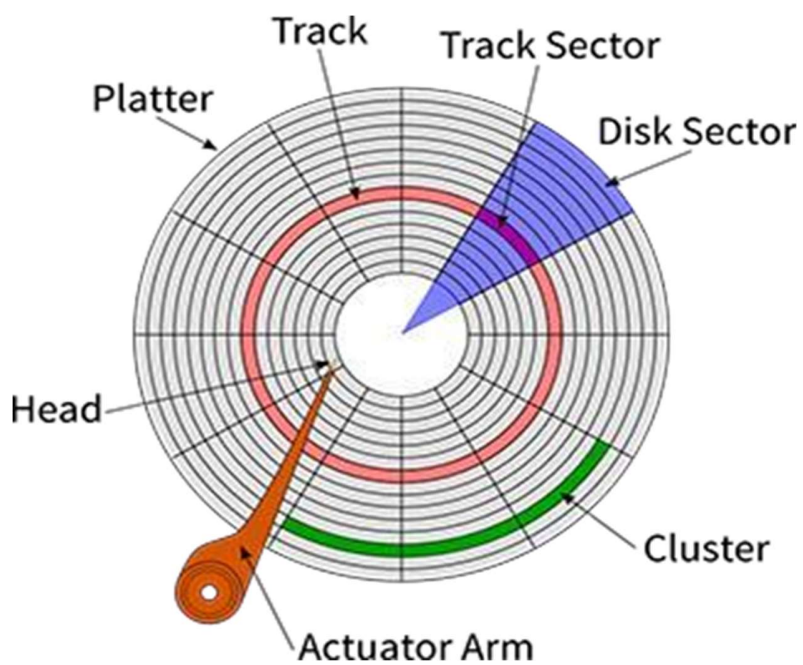
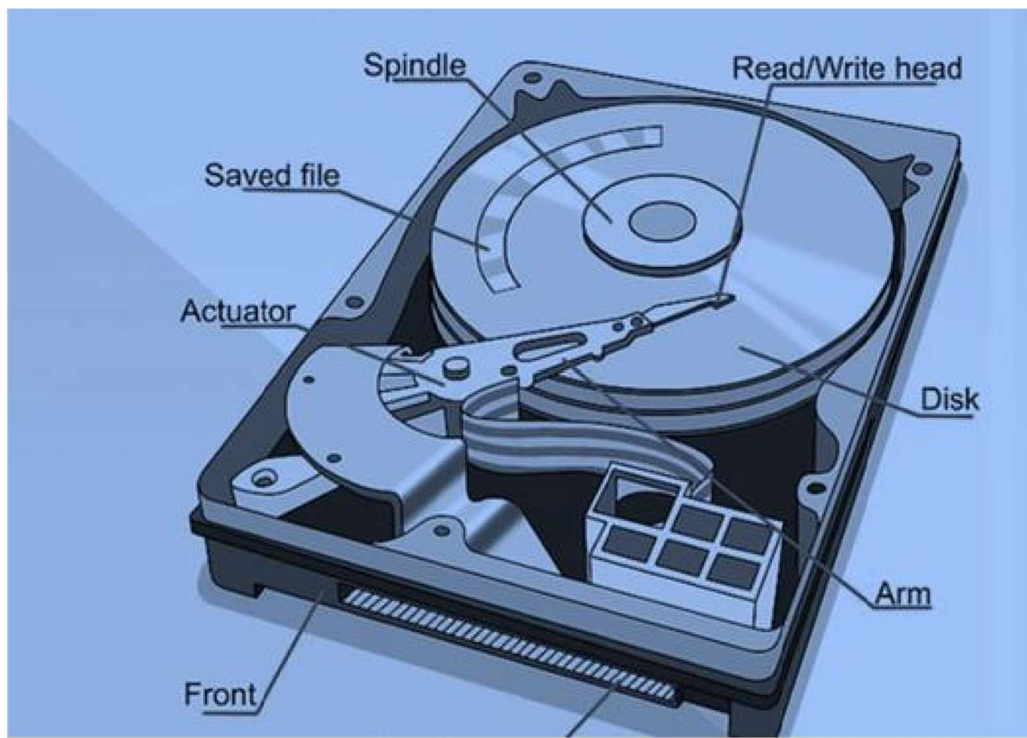
*CD ROM

Floppy Disk:

- It is the first auxiliary storage device to be fitted to a computer.
- A floppy disk is a flexible disk with a magnetic coating on it. It is packaged inside a protective plastic envelope.
- These are one of the oldest type of portable storage devices that could store up to 1.44 MB of data but now they are not used due to very less memory storage.

Hard disk:

- A hard disk consists of one or more circular disks called platters which are mounted on a common spindle.
- Each surface of a platter is coated with a magnetic material. Both surfaces of each disk are capable of storing data except the top and bottom disk where only the inner surface is used.
- The information is recorded on the surface of the rotating disk by magnetic read/write heads. These heads are joined to a common arm known as access arm.
- Storage capacity of hard disk is in TB.



Seek Time:

A disk is divided into many circular tracks. Seek Time is defined as the time required by the read/write head to move from one track to another.

Latency –

It is the time required by the read/write head to move from the current sector to the requested sector.

Data transfer rate:

The rate at which data is written to disk or read from the disk is called data transfer rate.

Access of disk:

The sum of seek time ,latency time and data transfer rate is called access of disk.

(Access of disk=seek time + latency time + data transfer rate)

CD Drive:

- CD stands for compact Disk.
- CDs are circular disks that use optical rays,usually lasers,to read and write data.
- Storage capacity of CDs are in hundreds of MB.CDs are inserted in CD drives built into CPU cabinet.
- They are portable as you can eject the drive ,remove the CD and carry it with you.

There are three types of CDs-

- **CD-ROM(Compact Disk Read Only Memory):** The data on these CDs are recorded by the manufacturer.
- Proprietary Software, audio or video are released on CD-ROMs.
- **CD-R(Compact Disk-Recordable):** Data can be written by the user once on the CD-R.It cannot be deleted or modified later.

- **CD-RW(Compact Disk-Rewritable):** Data can be written and deleted on these optical disks again and again.

DVD Drive:

- DVD stands for Digital Video Display.
- DVD are optical devices that can store 15times the data held by CDs.
- They are usually used to store rich multimedia files that need high storage capacity.
- DVDs also come in three varieties-read only, recordable and rewritable.

Pen Drive:

- Pen drive is a portable memory device that uses solid state memory rather than magnetic fields or lasers to record data.
- It uses a technology similar to RAM,except that it is non volatile.It is also called USB drive,key drive or flash memory.

5. Backup memory:

- Backup storage refers to a storage device, medium or facility that is used for storing copies and instances of back up data.
- Backup storage is primarily an additional storage device used for keeping backup data.Typically, it is external to the system,server or device for which the backup data is created,such as a local/remote storage server.
- The backup storage itself can be a hard disk drive, tape drive, compact disk drive or any mass storage medium installed within a computer or storage server.

Chapter-2

Computer Software

2.1. Concept of Software

- a. System Software
- b. Application Software

2.2. Operating System

- a. Objectives of OS and its Application
- b. Types of OS
 - i. Batch Operating system
 - ii. Multiprogramming Operating system
 - iii. Multiprocessing Operating system
 - iv. Multitasking Operating system

2.3. Features of DOS, WINDOWS and UNIX

2.4. Programming languages

- a. Machine level language
- b. Assembly level language
- c. High level language

2.5. Compiler and Interpreter

2.6. Computer virus of computer viruses

- a. Types of computer viruses
- b. Detection and prevention of viruses

Computer System is mainly divided into two categories:

- Computer Hardware
- Computer Software

Computer Hardware:

It refers to all the physical components presents in a computer and related devices which we can touch i.e. all the tangible components.

Computer Software:

It is the set of program used to operate computer and related devices that tells a computer what to do or how to perform a task.

Computer Software can be classified into two categories

- a. System Software
- b. Application Software

1.a. System Software:

- It is a software program that is designed to run a computer hardware and application programs.

System Software perform the following activities:

- It provide a platform for installation and development of application software.
- It facilitate error free communication between the main computer system and attached peripheral devices.
- It facilitates the execution of a program written in high level language.
- It monitors the effective utilization of the various hardware resources.

Some common system software:

- ❖ Operating system
- ❖ Language Processor
- ❖ Device Drivers
- ❖ Utility Programs

Operating System:

- It is a system software which is responsible for managing the various computer resources.
- It also provides a platform to the user for loading application software.

Language Processor:

- It is responsible for translating and interpreting program written by using programming languages.
- Example-Compiler, interpreter, assembler.

Device Drivers:

- It comes along with a peripheral devices which is used to establish an error free and easy communication between the device and computer.
- Example-Computer printer, graphics card, Network card etc.

Utility Program:

- It is capable of directly interacting with the computer hardware for various purpose.
- These programs are generally used for system maintenance activity.
- Example-disk format and partition utility, disk backup utility.

1.b. Application Software:

It is a computer software designed to perform a group of coordinated functions, task or activities for the benefit of the user.

There are mainly two classes of application software:

- High Level Language
- Application Package

Application program written by using 01's called as MLLP.

Application program written by using pneumatic codes called ALLP.

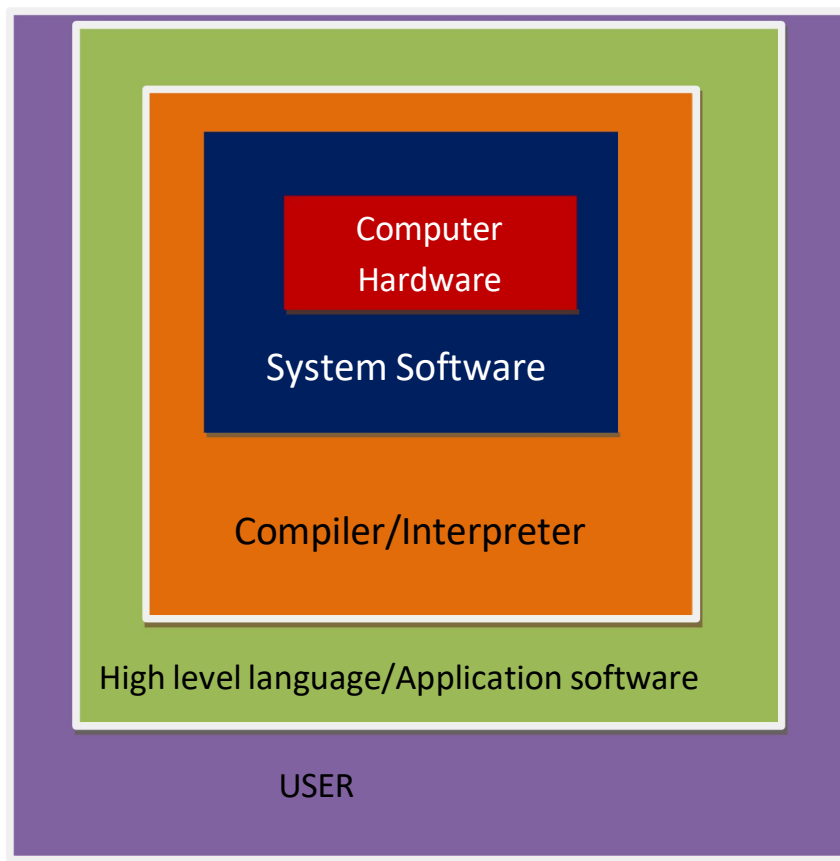
Application program written by using HLL or application package called HLLP.

Examples of HLL:

BASIC(beginners' all purpose symbolic instruction code),FORTRAN(formula translator),COBOL(common business oriented language),C,C++,JAVA etc.

2. OPERATING SYSTEM

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.



Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security

- Job accounting
- Coordination between other software and users

Memory Management:

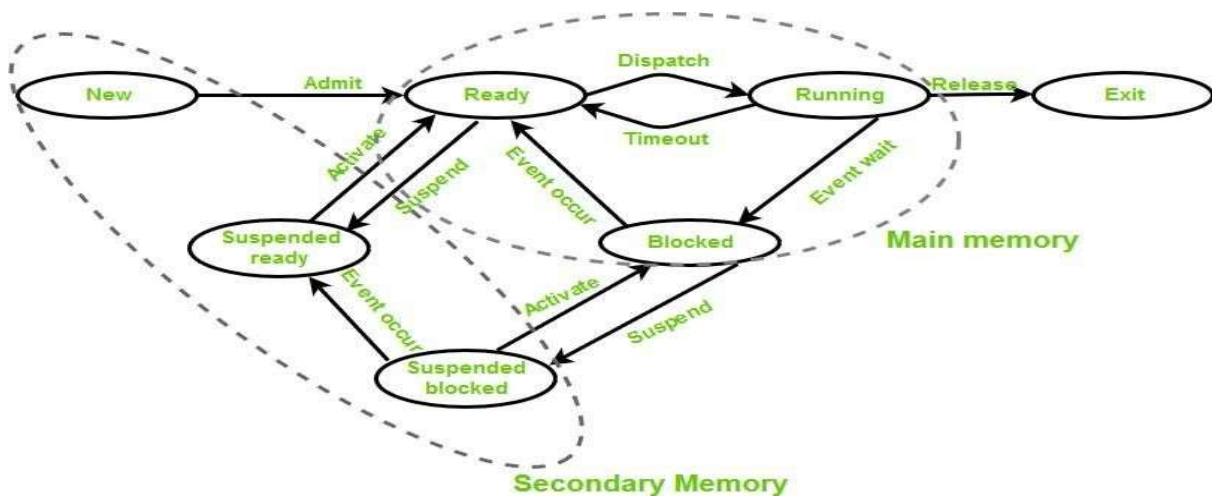
Memory management refers to management of Primary Memory or Main Memory. Main memory provides a fast storage that can be accessed directly by the CPU.

For a program to be executed, it must be in the main memory. An Operating System does the following activities for memory management –

- Keeps tracks of primary memory, i.e., what part of it is in use by whom, what part is not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Process Management:

In a multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called **process scheduling**. An Operating System does the following activities for process management –



- Keeps tracks of processor and status of process. The program responsible for this task is known as **traffic controller**.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management :

An Operating System manages device communication via their respective drivers. It does the following activities for device management –

- Keeps tracks of all devices. Program responsible for this task is known as the **I/O controller**.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management:

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

An Operating System does the following activities for file management –

- Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Security

- By means of password and similar other techniques, it prevents unauthorized access to programs and data.

Job accounting

- Keeping track of time and resources used by various jobs and users.

Coordination between other softwares and users

Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

2.A.Objective And Functions Of Operating System:

1. It acts as an interface between the human user and the computer hardware .
2. It provides a command interpreter, which provides some command either in the form of text or in the form of icon(GUI) to the user.
3. It provides some data management facility to the user to organize the data stored in the computer.
4. It either provides or supports programming development tools, which helps the user to write and execute program.
5. It offers a user interface.
6. Loads program into computer's memory.
7. Coordinates how program works with hardware and other software.
8. Manages how information is stored and retrieved from the disk.
9. Saves contents of file on to disk.
10. Reads contents of file from disk to memory.
11. Sends document to the printer and activates the printer.
12. Provides resources that copy or move data from one document to another, or from one program to another.
13. Allocates RAM among the running programs.
14. Recognizes keystrokes or mouse clicks and displays characters or graphics on the screen.

2.b.Types of Operating System:

There are four types of operating systems based on **types of computers they control and application they support** –

- Single-User/Single-Tasking operating system
- Single-User/Multitasking operating system
- Multi-User/Multitasking operating system
- Real-time operating system

Single-User/Single-Tasking OS:

An operating system that allows a single user to perform only one task at a time is called a Single-User Single-Tasking Operating System. Functions like printing a document, downloading images, etc., can be performed only one at a time. Examples include MS-DOS, Palm OS, etc.



Advantages

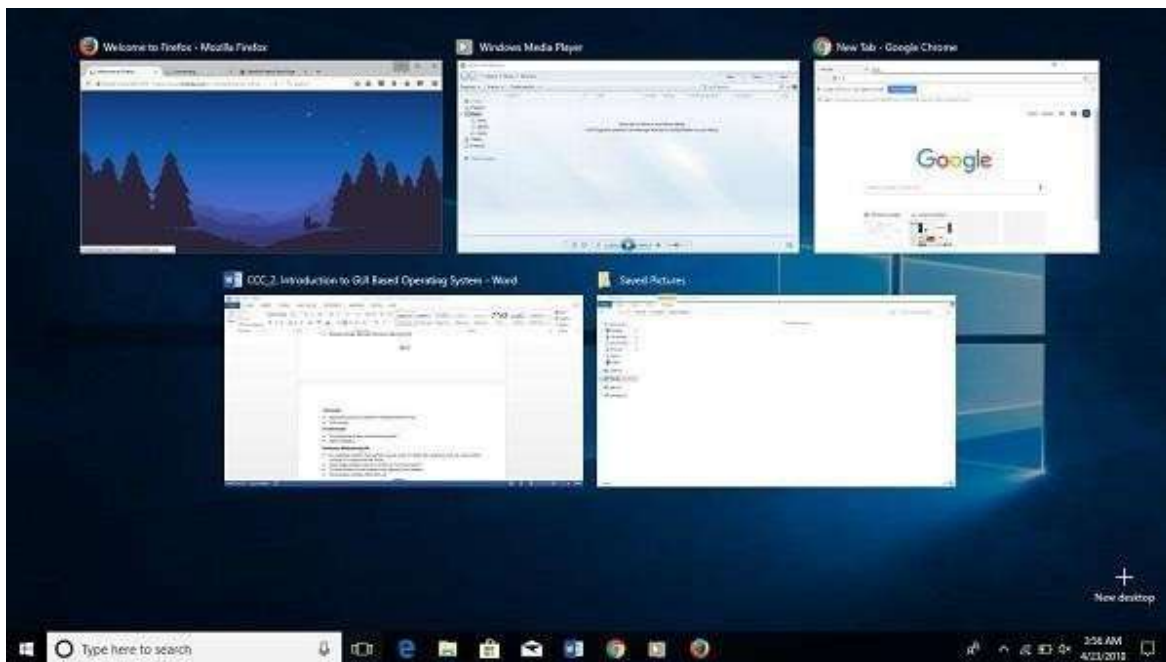
- This operating system occupies less space in memory.

Disadvantages

- It can perform only a single task at a time.

Single-User/Multitasking OS:

An operating system that allows a single user to perform more than one task at a time is called Single-User Multitasking Operating System. Examples include Microsoft Windows and Macintosh OS.



Advantages

- It is time saving as it performs multiple tasks at a time yielding high productivity.

Disadvantages

- This operating system is highly complex and occupies more space.

Multitasking OS:

It is an operating system that permits several users to utilize the programs that are concurrently running on a single network server. The single network server is termed as "Terminal server". "Terminal client" is a software that supports user sessions. Examples include UNIX, MVS, etc.

Advantages:

- It is highly productive as it performs multiple tasks at a time.
- It is time saving as we don't have to make changes in many desktops, instead can make changes only to the server.

Disadvantages:

- If the connection to the server is broken, user cannot perform any task on the client as it is connected to that server.

Real-time operating system

Real-time operating system is designed to run real-time applications. It can be both single- and multi-tasking. Examples include Abbasi, AMX RTOS, etc.



Advantages

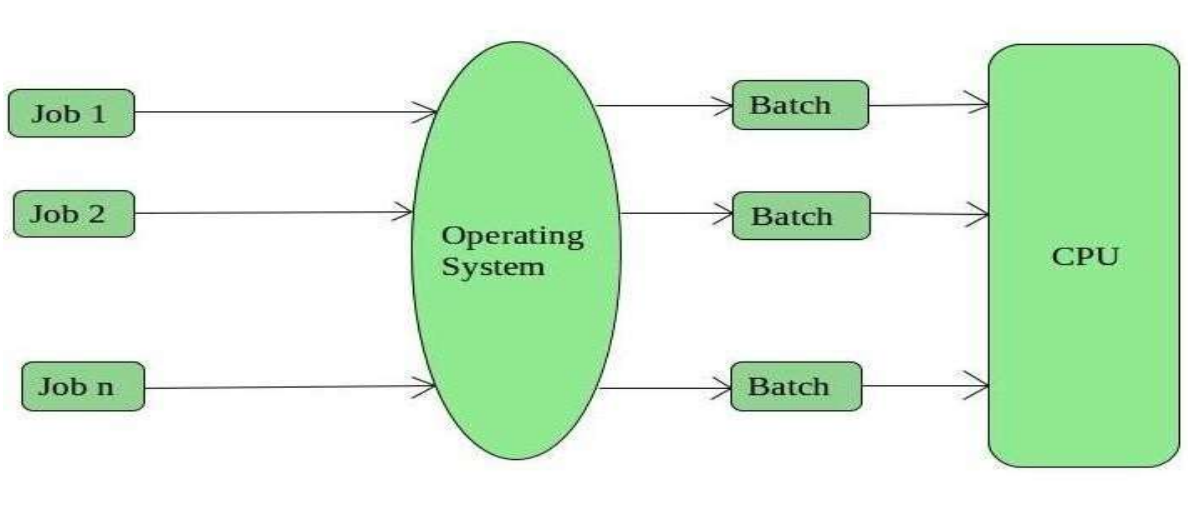
- It works very fast.
- It is time saving, as it need not be loaded from memory.
- Since it is very small, it occupies less space in memory.

Following are the popular types of Operating System:

- Batch Operating System
- Multitasking/Time Sharing OS
- Multiprocessing OS
- Real Time OS
- Distributed OS
- Network OS
- Mobile OS

Batch Operating System –

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirement and group them into batches. It is the responsibility of the operator to sort jobs with similar needs.



Advantages of Batch Operating System:

- It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for the batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometimes costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System: Payroll System, Bank Statements, etc.

Multiprogramming OS:

Sharing the processor, when two or more programs reside in memory at the same time, is referred as **multiprogramming**. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

The following figure shows the memory layout for a multiprogramming system.



An OS does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the jobs in the memory.
- Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensure that the CPU is never idle, unless there are no jobs to process.

Advantages

- High and efficient CPU utilization.
- User feels that many programs are allotted CPU almost simultaneously.

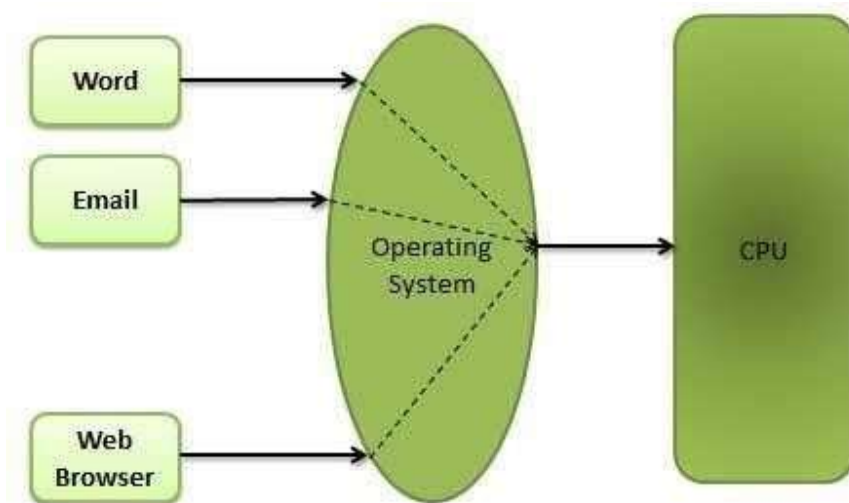
Disadvantages

- CPU scheduling is required.
- To accommodate many jobs in memory, memory management is required.

Multitasking OS

Multitasking is when multiple jobs are executed by the CPU simultaneously by switching between them. Switches occur so frequently that the users may interact with each program while it is running. An OS does the following activities related to multitasking –

- The user gives instructions to the operating system or to a program directly, and receives an immediate response.
- The OS handles multitasking in the way that it can handle multiple operations/executes multiple programs at a time.
- Multitasking Operating Systems are also known as Time-sharing systems.
- These Operating Systems were developed to provide interactive use of a computer system at a reasonable cost.
- A time-shared operating system uses the concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU.
- Each user has at least one separate program in memory.



- A program that is loaded into memory and is executing is commonly referred to as a **process**.
- When a process executes, it typically executes for only a very short time before it either finishes or needs to perform I/O.
- Since interactive I/O typically runs at slower speeds, it may take a long time to complete. During this time, a CPU can be utilized by another process.

- The operating system allows the users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user.
- As the system switches CPU rapidly from one user/program to the next, each user is given the impression that he/she has his/her own CPU, whereas actually one CPU is being shared among many users.

Real time OS

A real time operating system time interval to process and respond to inputs is very small. Examples: Military Software Systems, Space Software Systems are the Real time OS example.

Distributed Operating System

Distributed systems use many processors located in different machines to provide very fast computation to its users.

Network Operating System

Network Operating System runs on a server. It provides the capability to serve to manage data, user, groups, security, application, and other networking functions.

Mobile OS

Mobile operating systems are those OS which is especially that are designed to power smartphones, tablets, and wearables devices.

Some most famous mobile operating systems are Android and iOS, but others include BlackBerry, Web, and watchOS.

3. Features of DOS, Windows and Unix:

1. Disk Operating System (DOS) :

DOS stands for Disk Operating System. It is a single-user (no security), a single-process system that gives complete control of the computer to the user program. It consumes less memory and power than Unix.

DOS commands can be typed in either upper case or lower case.

2. UNIX :

UNIX is a powerful, multi-user, and multitasking operating system that was originally developed at AT & T Bell Laboratories. It is a very popular operating system among the scientific, engineering, and academic due to its most appreciating features like flexibility, portability, network capabilities, etc.

3. Windows :

Microsoft window is a most demanding Graphical user Interface (GUI) based operating system that replaces all the command line based functions to the user friendly screens. Microsoft introduced a series of versions with the latest functions.

Features of DOS

Following are the significant features of DOS –

- It is a single user system.
- It controls program.
- It is machine independence.
- It manages (computer) files.
- It manages input and output system.
- It manages (computer) memory.
- It provides command processing facilities.
- It operates with Assembler.

Following are the major types of DOS Command –

1. CLS – It is a command that allows you to clear the complete contents of the screen and leave only a prompt.
2. CD- To displays the name of or changes the current directory
3. REN – It is used to rename files and directories from the original name to a new name.
4. EXIT – The exit command is used to withdrawal from the currently running application and the MS-DOS session.
5. DEL- Del is a command used to delete files from the computer.
6. COPY – Allows you to copy one or more files to an alternate location.
7. VOL – Displays the volume of information about the designated drive.
8. TYPE- Display the contents of a text file.

9. DATE – The date command can be used to look at the current date of the computer as well as change the date to an alternate date.
10. TIME – Allows the user to view and edit the computer's time.
11. DIR – The dir command displays information about files and directories, and how many spaces available.
12. MKDIR/MD – Allows you to create directories in MS-DOS.
13. PATH – Path is used to specify the location where MS-DOS looks when using a command.
14. DISKCOMP- Comparing two diskettes.
15. DISKCOPY- Copying a diskette.

Features of Windows OS

Following are the significant element of **Windows Operating System (WOS)** –

- Graphical User Interface
- Icons (pictures, documents, application, program icons, etc.)
- Taskbar
- Start button
- Windows explorer
- Mouse button
- Hardware compatibility
- Software compatibility
- Help, etc.

The following are the advantages of Unix Features.

- **Portability:**
- The system is written in high-level language making it easier to read, understand, change and, therefore move to other machines. The code can be changed and compiled on a new machine. Customers can then choose from a wide variety of hardware vendors without being locked in with a particular vendor.
- **Machine-independence:**
- The System hides the machine architecture from the user, making it easier to write applications that can run on micros, mins and mainframes.
- **Multi-Tasking:**

- Unix is a powerful multi-tasking operating system; it means when a active task in in process, there can be a simultaneous background process working too. Unix handles these active and background threads efficiently and manages the system resources in a fair-share manner.
- **Multi-User Operations:**
- UNIX is a multi-user system designed to support a group of users simultaneously. The system allows for the sharing of processing power and peripheral resources, while at the same time providing excellent security features.
- **Hierarchical File System:**
- UNIX uses a hierarchile file structure to store information. This structure has the maximum flexibility in grouping information in a way that reflects its natural state. It allows for easy maintenance and efficient implementation.
- **UNIX shell:**
- UNIX has a simple user interface called the shell that has the power to provide the services that the user wants. It protects the user from having to know the intricate hardware details.
- **Pipes and Filters:**
- UNIX has facilities called Pipes and Filters which permit the user to create complex programs from simple programs.
- **Utilities:**
- UNIX has over 200 utility programs for various functions. New utilities can be built effortlessly by combining existing utilities.
- **Software Development Tools:**
- UNIX offers an excellent variety of tools for software development for all phases, from program editing to maintenance of software,

Differences between DOS and Windows:

Sl. No.	DOS	WINDOWS
1	DOS stands for Disk Operating System.	Windows stands for Windows, no specific form.
2	DOS is single tasking OS.	Windows is multi-tasking OS.
3	DOS consumes quite low power	Windows consumes high power.
4	DOS memory	Windows memory

	requirements are quite low.	requirements are quite high as compared to DOS.
5	DOS has no support for networking	Windows supports networking.
6	DOS is complex in usage. You need to remember commands to use DOS properly.	Windows usages is user-friendly and is quite simple to use.
7	DOS is command line based OS.	Windows is GUI based OS
8	Multimedia is not supported in DOS.	Windows supports multimedia likes games, videos, audios etc.
9	DOS command execution is faster than Windows.	Windows operations are slower as compared to DOS.
10	DOS supports single window at a time.	Windows supports multiple window at a time.

Difference between DOS and Linux :

Sl.No	DOS	UNIX
1	DOS is single tasking operating system.	UNIX are multitasking operating systems.
2	It is monouser (no concept of more than one user)	UNIX are multiuser (with multiple simultaneous users);
3	It consumes low power.	It consumes high power.
4	It has no native support for IP networks.	It come with built-in support for IP networks.
5	It has 3 proprietary implementations (MS-DOS, IBM DOS and DR-DOS) and one free implementation (FreeDOS).	There are many proprietary and free/open source implementations.

6	It is not case sensitive.	It is case sensitive.
7	It uses backslashes.	It uses forward slashes.
8	It has no virtual memory nor protected memory.	It usually have virtual memory and protected memory.

Difference between UNIX and Windows Operating System :

1	UNIX	Windows
2	It is an open source code.	It is not an open source code.
3	It is more stable.	As, multiple programs are running. So it is unstable.
4	It is case sensitive.	It is not a case sensitive.
5	In this, extensions have not affect on the type of file.	In this, certain file extensions are used to identify the type of file.
6	In this, several ways are used to manage the device drivers.	After the installation of Windows, the various device-drivers packages provide interactive GUI for the configuration of devices.
7	We cannot recover our data.	In this, we can recover our data from the recycle-bin, because in this after the deletion of data it will store in the recycle-bin.
8	It has greater processing power.	It has less processing power.
9	It has greater in-built security.	It has less in-built security.
10	It has interface between user and kernel known as shell, so it does not face any virus attack.	In this, we don't have shell interface. User direct interact with hardware and it enhances the attack of virus.
11	In this, file system is represented as a hierarchical tree under the same root. There is no drive system like drive C, drive D etc.	In this, file system can have many hierarchies; for example, different file system for each partitions and these partitions are represented as drive alphabet letters like as C: D: etc.

4. PROGRAMMING LANGUAGES :

A programming language is a formal language comprising a set of strings that produce various kinds of machine code output. Programming languages are used in computer programming to implement algorithms. Most programming languages consist of instructions for computers.

Types of Programming Languages:

1. Machine level language
2. Assembly level language
3. High level language

Machine level language:

Machine language is the lowest and most elementary level of programming language and was the first type of programming language to be developed. Machine language is basically the only language that a computer can understand and it is usually written in hex.

In fact, a manufacturer designs a computer to obey just one language, its machine code, which is represented inside the computer by a string of binary digits (bits) 0 and 1. The symbol 0 stands for the absence of an electric pulse and the 1 stands for the presence of an electric pulse. Since a computer is capable of recognizing electric signals, it understands machine language.

Advantages:

- a. Machine language makes fast and efficient use of the computer
- b. It requires no translator to translate the code. It is directly understood by the computer.

Disadvantages:

- a. All operation codes have to be remembered
- b. All memory addresses have to be remembered.
- c. It is hard to amend or find errors in a program written in the machine language.

Assembly level language:

Assembly Language program can be defined as that category of program which are written by using mnemonic codes.

It overcomes the problem of writing a program by using binary digit and hence minimises the probability of error.

It helps in understanding the programming language to machine code.

In computers, there is an assembler that helps in converting the assembly code into machine code executable. Assembly language is designed to understand the instruction and provide it to machine language for further processing.

Advantages

1. It allows complex jobs to run in a simpler way.
2. It is memory efficient, as it requires less memory.
3. It is faster in speed, as its execution time is less.
4. It is mainly hardware-oriented.
5. It requires less instruction to get the result.
6. It is used for critical jobs.
7. It is not required to keep track of memory locations.
8. It is a low-level embedded system.

Disadvantages

1. It takes a lot of time and effort to write the code for the same.
2. It is very complex and difficult to understand.
3. The syntax is difficult to remember.
4. It has a lack of portability of program between different computer architectures.
5. It needs more size or memory of the computer to run the long programs written in Assembly Language.

High level language:

High level language is abbreviated as **HLL**. High level languages are similar to the human language. Unlike low level languages, high level languages are programmers friendly, easy to code, debug and maintain.

High level language provides higher level of abstraction from machine language. They do not interact directly with the hardware. Rather, they focus more on the complex arithmetic operations, optimal program efficiency and easiness in coding.

Low level programming uses machine friendly language. Programmers writes code either in binary or assembly language. Writing programs in binary is complex and cumbersome process. Hence, to make programming more programmers friendly. Programs in high level language is written using English statements.

High level programs require compilers/interpreters to translate source code to machine language. We can compile the source code written in high level language to multiple machine languages. Thus, they are machine independent language.

Advantages

1. High level languages are programmer friendly. They are easy to write, debug and maintain.
2. It provide higher level of abstraction from machine languages.
3. It is machine independent language.
4. Easy to learn.
5. Less error prone, easy to find and debug errors.
6. High level programming results in better programming productivity.

Disadvantages

1. It takes additional translation times to translate the source to machine code.
2. High level programs are comparatively slower than low level programs.
3. Compared to low level programs, they are generally less memory efficient.
4. Cannot communicate directly with the hardware.

5.a. What is Compiler?

A compiler is a computer program that transforms code written in a high-level programming language into the machine code. It is a program which translates the human-readable code to a language a computer processor understands (binary 1 and 0 bits). The computer processes the machine code to perform the corresponding tasks.

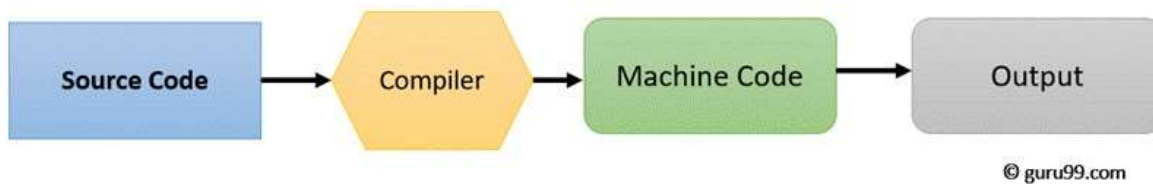
A compiler should comply with the syntax rule of that programming language in which it is written. However, the compiler is only a program and cannot fix errors found in that program. So, if you make a mistake, you need to make changes in the syntax of your program. Otherwise, it will not compile.

5.b. What is Interpreter?

An interpreter is a computer program, which coverts each high-level program statement into the machine code. This includes source code, pre-compiled code, and scripts. Both compiler and interpreters do the same job which is converting higher level programming language to machine code. However, a compiler will convert the code into machine code

(create an exe) before program run. Interpreters convert code into machine code when the program is run.

How Compiler Works



How Interpreter Works



Difference between compiler and interpreter

Sl no.	Compiler	Interpreter
1	Translate the entire program at a time	Translate the program line by line
2	Requires more main memory	Requires less main memory
3	If error is detected nothing is executed	If error is encountered stops but previous statements are already executed
4	Execution time is less	Execution time is more
5	Neither source program nor compiler is required for execution	Source program along with interpreter are required for execution.

6.Computer Virus and Different types of computer virus

What is a Computer Virus?

A computer virus is a malicious software written intentionally to enter a computer without the users permission or knowledge,with the ability to self-replicate infecting other programs on a computer. Just like how humans catch a cold or flu, it can remain dormant inside the system and gets activated when you least expect it.

A computer virus is developed to spread from one host to another and there are numerous ways on how your computer catches it. It can be through email attachments, file downloads, software installations, or unsecured links.

Common Types Of Computer Viruses

- ❖ **Boot Sector Virus**
- ❖ **program Virus**
- ❖ **Macro Virus**
- ❖ **Multipartile Virus**
- ❖ **Polymorphic Virus**
- ❖ **Stealth Virus**

Boot Sector virus:

It infects the master boot record in the hard disk.They replace the boot record program which is responsible for loading the operating system in memory,by copying it elsewhere on the disk or overwriting it.

Boot sector virus load into memory when the computer boots.

Examples:

- ❖ Disk killer
- ❖ Store virus
- ❖ Michael Angelo
- ❖ From

Program Virus:

It infects executable files having extension like. Exe,.com,.bin,.drv,.sys etc.

These virus are loaded in memory during execution of the files.

The virus program remain active in the memory and multiply itself making the memory full.

Examples:

- ❖ Sunday
- ❖ Cascade

Macro Virus:

It is new type of computer virus that infects the macros within a document or template.

When we open a word processing or spread sheet document, the macro virus is activated and it infects the normal template.

These virus propagates from one computer to another through the infected document file.

Examples:

- ❖ Nuclear
- ❖ Word concept
- ❖ Document macro virus

Multipartile virus:

It is a hybrid of boot and program virus.

They first infect the program files and when the infected program is executed, these virus infect the boot record.

When the system is booted next time, the virus from the boot record loads itself to the memory and infects other programs.

Examples:

- ❖ Invader
- ❖ Flip
- ❖ Tequila

Polymorphic virus:

It is capable of encrypting its code in different manner so that each appears different in each infection.

These virus are difficult to detect .

Examples:

- ❖ Cascade
- ❖ Evil
- ❖ Proud
- ❖ Virus 101

Stealth virus:

It use certain technique to avoid detection.

They usually direct the disk head to read a wrong sector instead of the one in which they reside, or they change the reading of the infected files size shown in the directory listing.

Examples:

- ❖ Joshi
- ❖ Whale
- ❖ Frodo

Virus life cycle:

1. Virus creation
2. Virus infection and replication
3. Virus activation
4. Virus detection
5. Virus eradiction

Computer may be infected if shows following symptoms:

- Computer is giving problem during booting or takes a lot of time for booting
- Computer is resetting automatically.
- Computer is hanging when the user tries to execute a particular program
- Computer is displaying some unusual figures/signs on the screen
- The computer is giving some message such as "Diskfull", "Insufficient" memory" etc.
- Through installation of illegal or pirates softwares specially games.

Protection Against These Types Computer Viruses:

A virus left untreated can damage on your device but if detected early, and appropriate measures are done, then the recovery would be quick. Just like how we protect ourselves from catching a virus, here are a few notes to remember to help keep your devices safe.

1. Avoid clicking on suspicious links.
2. Don't open unknown emails received in your mailbox ,Scan email attachments before opening it.
3. Avoid clicking on pop-up advertisements and get a pop-up blocker for your web browser.
4. Don't visit websites which are not reputed.
5. Install a reliable anti-virus program and always keep it up to date.

What is an Anti-Virus?

An anti-virus is a software which comprises programs or set of programs which can detect and remove all the harmful and malicious software from your device. This anti-virus software is designed in a manner that they can search through the files in a computer and determine the files which are heavy or mildly infected by a virus.

Given below is a list of few of the major antivirus software which is most commonly used:

Norton Antivirus
Kaspersky Antivirus
AVAST Antivirus

Comodo Antivirus
McAfee Antivirus

Application of Computer

- Education
- Business
- Scientific Research
- Government Offices
- Health and Medicine
- Training
- Real Time System
- Manufacturing Industries
- Home
- Agriculture
- Transportation
- Communication with the world
- Bank

Chapter-3

Computer Network and Internet

3.1. Network Concept

- i. Protocol
- ii. .Data Transmission Mode

3.2. Network Topology

- i. Bus Topology
- ii. Star Topology
- iii. Ring Topology
- iv. Hybrid Topology

3.4. Types of Network

- i. PAN
- ii. LAN
- iii. MAN
- iv. WAN

3.5. Network Devices

- iii. Hub

- iv. Switch
- iii.Repeater
- iv.Router
- v.Bridge
- vi.Gateway

3.6. Internet Services

- v. WWW
- vi. FTP
- iii.Chatting
- iv.HTTP

3.7. Different Types of Internet Connectivity and ISPs.

1. Computer Network

A computer network is a group of computers and other devices connected in some ways so as to be able to exchange data. Networks are built with a combination of computer hardware and software that support data communication across these devices.

Network Goals:

1. Resource sharing
2. Job sharing
3. Achieving reliability.

1.i. Protocol:

Protocol is a set of rules which are used in digital communication to connect network devices and exchange information between them.

These rules include guidelines that regulate the following characteristics of a network:

1. Access Methods
2. Allowed Physical Topology
3. Types of Cabling
4. Speed of data transfer

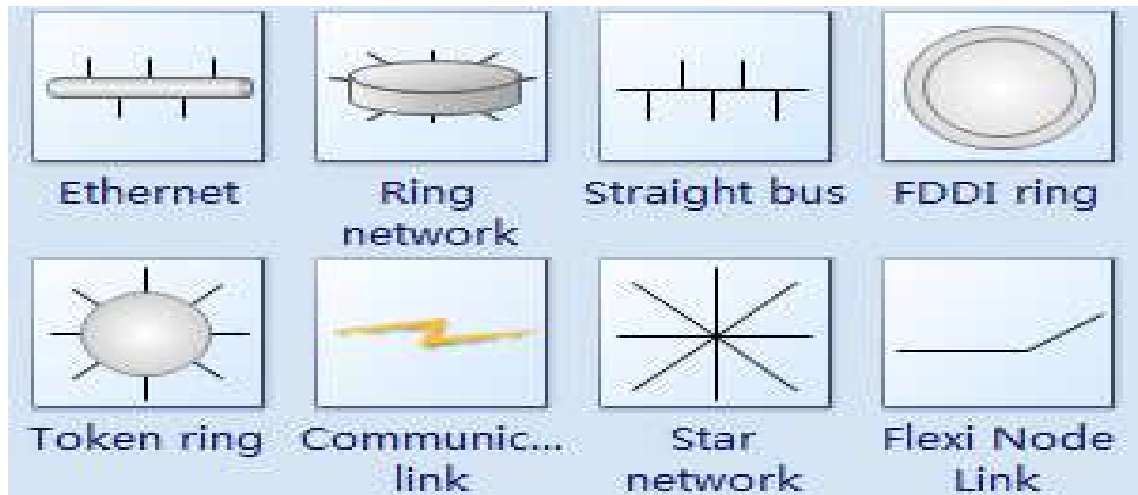
Types of Network Protocols

The most common network protocols are:

1. Ethernet
2. Local Talk
3. Token Ring
4. FDDI

5. ATM

The followings are some commonly used network symbols to draw different kinds of network protocols.



1-Ethernet

The Ethernet protocol is by far the most widely used one.

Ethernet uses an access method called CSMA/CD (Carrier Sense Multiple Access/Collision Detection). This is a system where each computer listens to the cable before sending anything through the network. If the network is clear, the computer will transmit. If some other nodes have already transmitted on the cable, the computer will wait and try again when the line is clear. Sometimes, two computers attempt to transmit at the same instant.

A collision occurs when this happens. Each computer then backs off and waits a random amount of time before attempting to retransmit. With this access method, it is normal to have collisions. However, the delay caused by collisions and retransmitting is very small and does not normally effect the speed of transmission on the network.

The Ethernet protocol allows for linear bus, star, or tree topologies. Data can be transmitted over wireless access points, twisted pair, coaxial, or fiber optic cable at a speed of 10 Mbps up to 1000 Mbps.

2-Local Talk

The method used by Local Talk is called CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance).

It is similar to CSMA/CD except that a computer signals its intent to transmit before it actually does so. Local Talk adapters and special twisted pair cable can be used to connect a series of computers through the serial port.

The Local Talk protocol allows for linear bus, star, or tree topologies using twisted pair cable. A primary disadvantage of Local Talk is low speed. Its speed of transmission is only 230 Kbps.

3-Token Ring

The Token Ring protocol uses the access method called token-passing.

In Token Ring, the computers are connected so that the signal travels around the network from one computer to another in a logical ring. A single electronic token moves around the ring from one computer to the next. If a computer does not have information to transmit, it simply passes the token on to the next workstation. If a computer wishes to transmit and receives an empty token, it attaches data to the token. The token then proceeds around the ring until it comes to the computer for which the data is meant. At this point, the data is captured by the receiving computer.

The Token Ring protocol requires a star-wired ring using twisted pair or fiber optic cable. It can operate at transmission speeds of 4 Mbps or 16 Mbps. Due to the increasing popularity of Ethernet, the use of Token Ring in school environments has decreased.

4-FDDI

Fiber Distributed Data Interface (FDDI) is a network protocol that is used primarily to interconnect two or more local area networks, often over large distances.

The access method used by FDDI involves token-passing. FDDI uses a dual ring physical topology. Transmission normally occurs on one of the rings; however, if a break occurs, the system keeps information moving by automatically using portions of the second ring to create a new complete ring. A major advantage of FDDI is high speed. It operates over fiber optic cable at 100 Mbps.

Protocol	Cable	Speed	Topology
Ethernet	Twisted Pair, Coaxial, Fiber	10 Mbps	Linear Bus, Star, Tree
Fast Ethernet	Twisted Pair, Fiber	100 Mbps	Star
LocalTalk	Twisted Pair	23 Mbps	Linear Bus or Star
Token Ring	Twisted Pair	4 Mbps-16 Mbps	Star-Wired Ring
FDDI	Fiber	100 Mbps	Dual ring
ATM	Twisted Pair, Fiber	155-2488 Mbps	Linear Bus, Star, Tree

5-ATM

Asynchronous Transfer Mode (ATM) is a network protocol that transmits data at a speed of 155 Mbps and higher. ATM works by transmitting all data in small packets of a fixed size; whereas, other protocols transfer variable length packets. ATM supports a variety of media such as video, CD-quality audio, and imaging. ATM employs a star topology, which can work with fiber optic as well as twisted pair cable.

ATM is most often used to interconnect two or more local area networks. It is also frequently used by Internet Service Providers to utilize high-speed access to the Internet for their clients. As ATM technology becomes more cost-effective, it will provide another solution for constructing faster local area networks.

Connecting Media

- Connecting media of a network refer to the transmission media used in the network.
- It refers to the physical media through which communication signals can be transmitted from one point to another.
- It can be divided into two broad categories

1) Guided Media

2) Unguided Media

Guided Media:

- The data signal in guided medium is bound by the cabling system that guide the data signal along a specific path.
- It consists of a cable composed of materials like copper, tin or silver.
- Basically, they are divided into three categories:

(a) Ethernet cable or Twisted Pair:

- In this pair, wires are twisted together, which are surrounded by an insulating material and an outer layer called jacket.
- A twisted pair consists of two conductors (copper)

- Ex- LAN Cable

(b) Coaxial Cable:

- It carries the signal of higher frequency data communication through the network.
- It is commonly used in transferring multi-channel television signals in cities.
- Ex- cable TV network

(c) Fiber – Optics Cable:

- It is made up of glass or plastic and transmits signals in the form of light from a source at one end to another end.
- The speed of Optical fiber is hundreds of times of faster than coaxial cables.
- Ex- Wavelength Division Multiplexing, SONET network

Unguided Media:

- It is the transfer of information over a distance without the use of enhanced electrical conductors or wires.
- When the computers in a network interconnected and data is transmitted through waves, then they are said to be connected through unguided media.
- Some commonly used unguided media of transmission are –
 1. Radio wave transmission
 2. Micro wave transmission
 3. Satellite communication
 4. Infrared wave transmission
 5. Bluetooth.

Data Transmission Mode

Data Transmission mode defines the direction of the flow of information between two communication devices. It is also called Data Communication or Directional Mode. It specifies the direction of the flow of information from one place to another in a computer network.

The data transmission modes can be characterized in the following three types based on the direction of exchange of information:

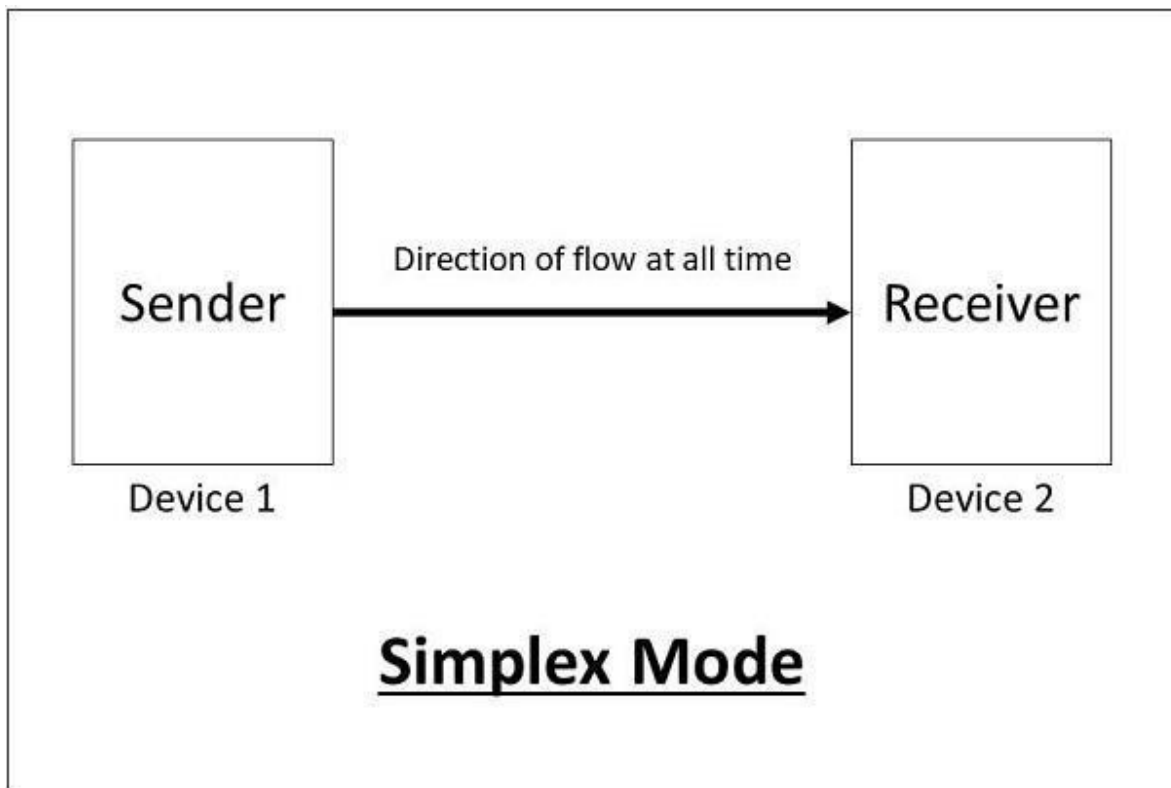
1. Simplex
2. Half-Duplex
3. Full Duplex

Now, let us study these various data transmission modes in the computer network one by one.

According to the Direction of Exchange of Information:

1. Simplex

Simplex is the data transmission mode in which the data can flow only in one direction, i.e., the communication is unidirectional. In this mode, a sender can only send data but can not receive it. Similarly, a receiver can only receive data but can not send it.



This transmission mode is not so popular because we cannot perform two-way communication between the sender and receiver in this mode. It is mainly used in the business field as in sales that do not require any corresponding reply. It is similar to a one-way street.

For Example, Radio and TV transmission, keyboard, mouse, etc.

Following are the advantages of using a Simplex transmission mode:

1. It utilizes the full capacity of the communication channel during data transmission.
2. It has the least or no data traffic issues as data flows only in one direction.

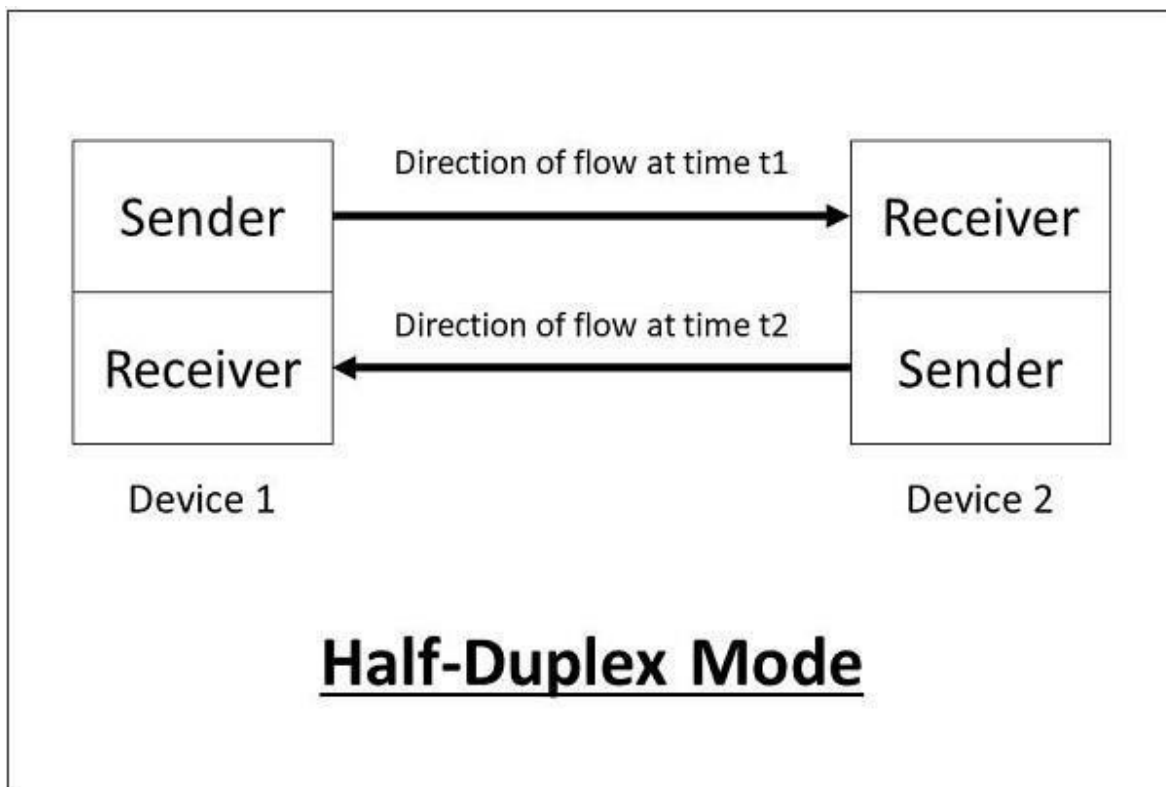
Following are the disadvantages of using a Simplex transmission mode:

1. It is unidirectional in nature having no inter-communication between devices.

2. There is no mechanism for information to be transmitted back to the sender (No mechanism for acknowledgement).

2. Half-Duplex

Half-Duplex is the data transmission mode in which the data can flow in both directions but in one direction at a time. It is also referred to as Semi-Duplex. In other words, each station can both transmit and receive the data but not at the same time. When one device is sending the other can only receive and vice-versa.



In this type of transmission mode, the entire capacity of the channel can be utilized for each direction. Transmission lines can carry data in both directions, but the data can be sent only in one direction at a time.

This type of data transmission mode can be used in cases where there is no need for communication in both directions at the same time. It can be used for error detection

when the sender does not send or the receiver does not receive the data properly. In such cases, the data needs to be transmitted again by the receiver.

For Example, Walkie-Talkie, Internet Browsers, etc.

Following are the advantages of using a half-duplex transmission mode:

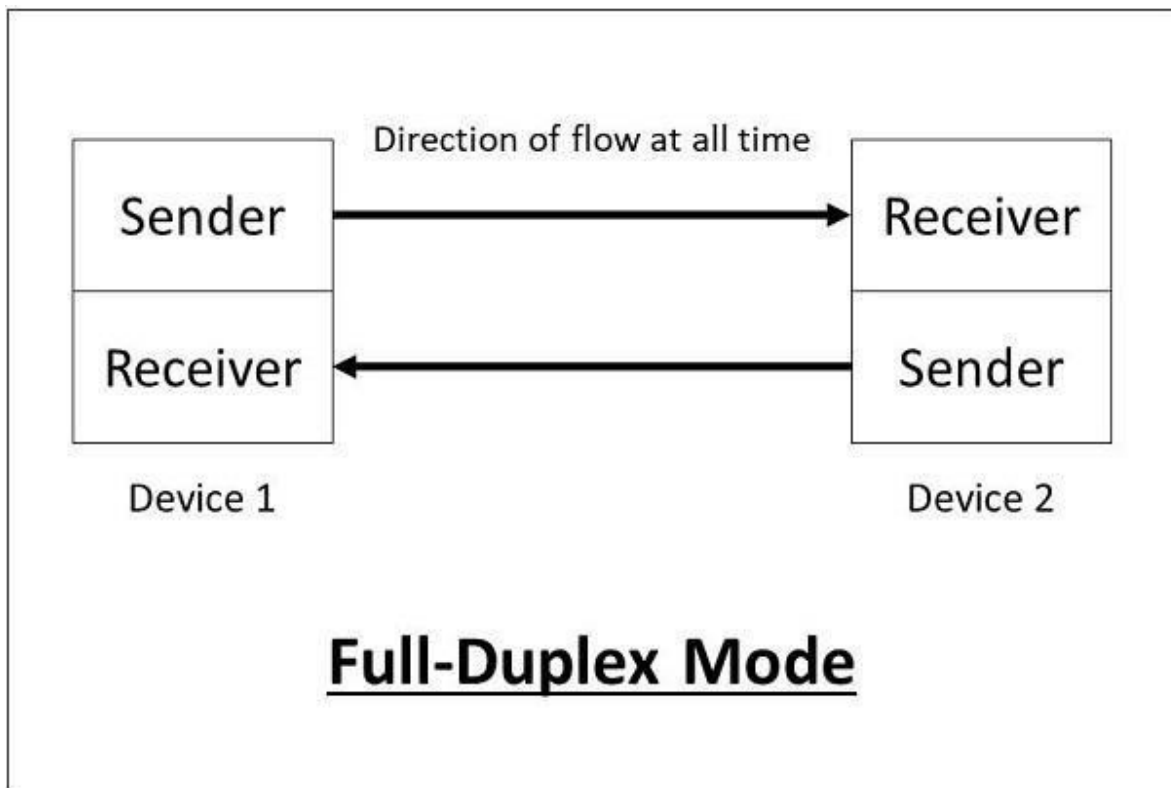
1. It facilitates the optimum use of the communication channel.
2. It provides two-way communication.

Following are the disadvantages of using a half-duplex transmission mode:

1. The two-way communication can not be established simultaneously at the same time.
2. Delay in transmission may occur as only one way communication can be possible at a time.

3. Full-Duplex

Full-Duplex is the data transmission mode in which the data can flow in both directions at the same time. It is bi-directional in nature. It is two-way communication in which both the stations can transmit and receive the data simultaneously.



Full-Duplex mode has double bandwidth as compared to the half-duplex. The capacity of the channel is divided between the two directions of communication. This mode is used when communication in both directions is required simultaneously.

For Example, a Telephone Network, in which both the persons can talk and listen to each other simultaneously.

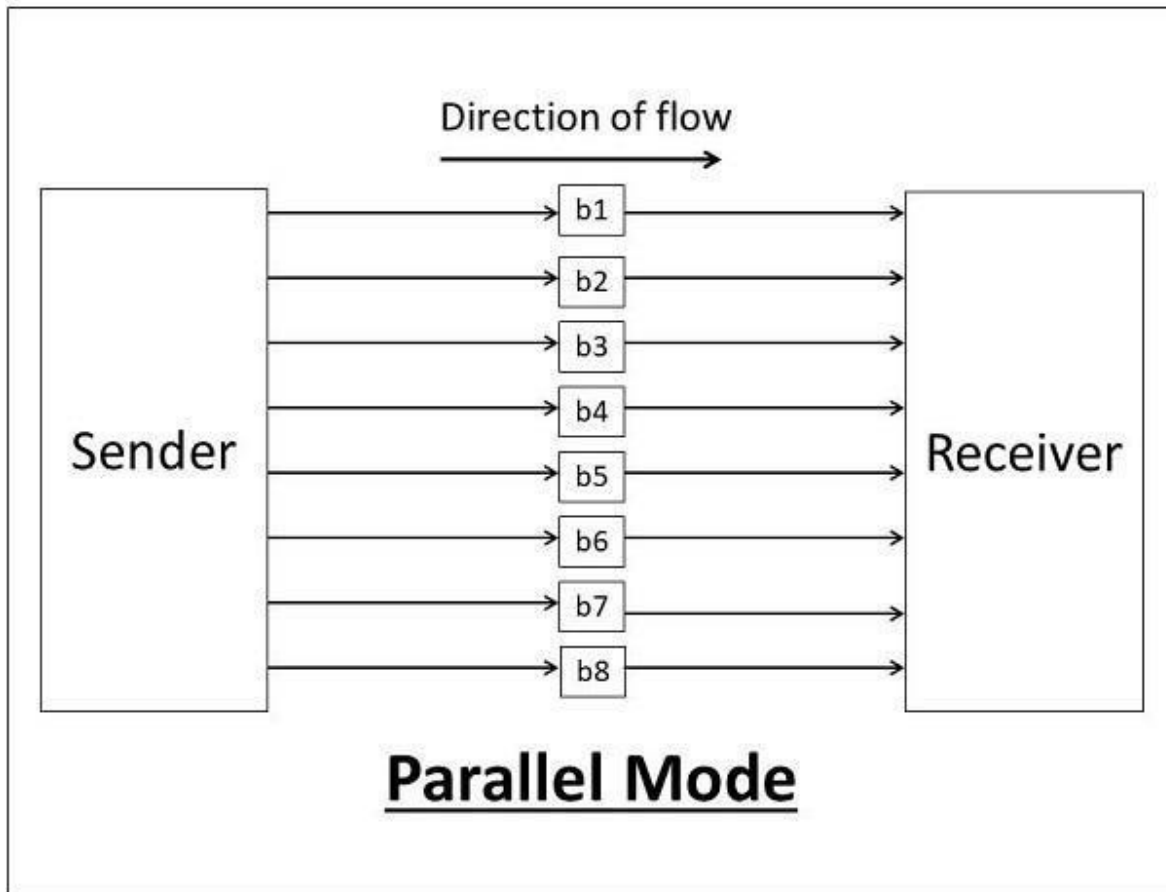
Following are the advantages of using a full-duplex transmission mode:

1. The two-way communication can be carried out simultaneously in both directions.
2. It is the fastest mode of communication between devices.

Following are the disadvantages of using a half-duplex transmission mode:

1. The capacity of the communication channel is divided into two parts. Also, no dedicated path exists for data transfer.

2. It has improper channel bandwidth utilization as there exist two separate paths for two communicating devices.



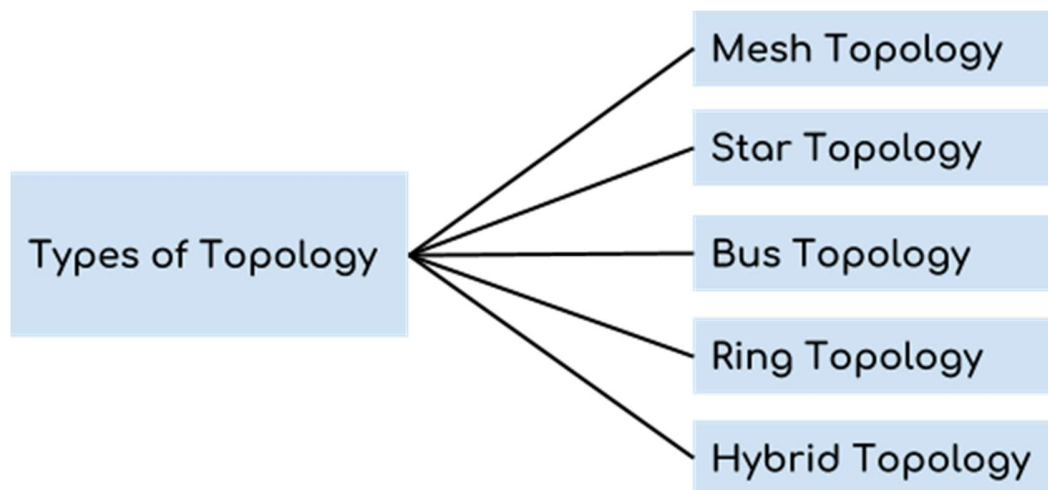
2. Network Topologies, Types of Network:

Network Topologies Network

- Topology is determined only by the configuration of connections between nodes.
- In a fully connected network with n nodes, there are $n(n-1)/2$ direct links.

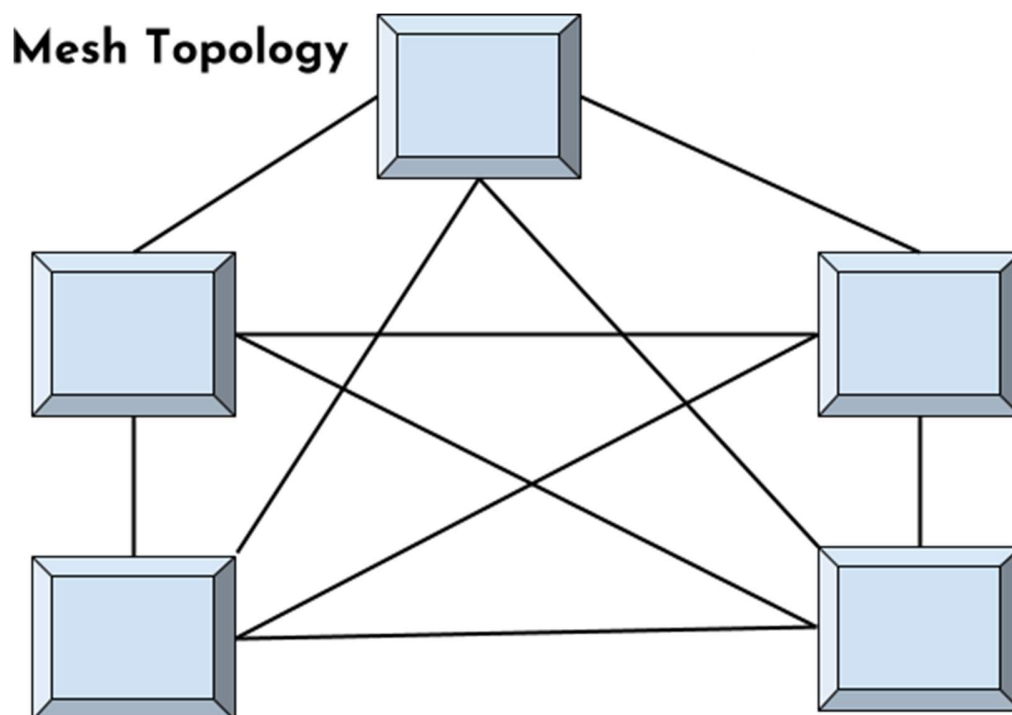
Types of Topology

There are five types of topology in computer networks:



1. Mesh Topology
2. Star Topology
3. Bus Topology
4. Ring Topology
5. Hybrid Topology

Mesh Topology



In mesh topology each device is connected to every other device on the network through a dedicated point-to-point link. When we say dedicated it means that the

link only carries data for the two connected devices only. Lets say we have n devices in the network then each device must be connected with $(n-1)$ devices of the network. Number of links in a mesh topology of n devices would be $n(n-1)/2$.

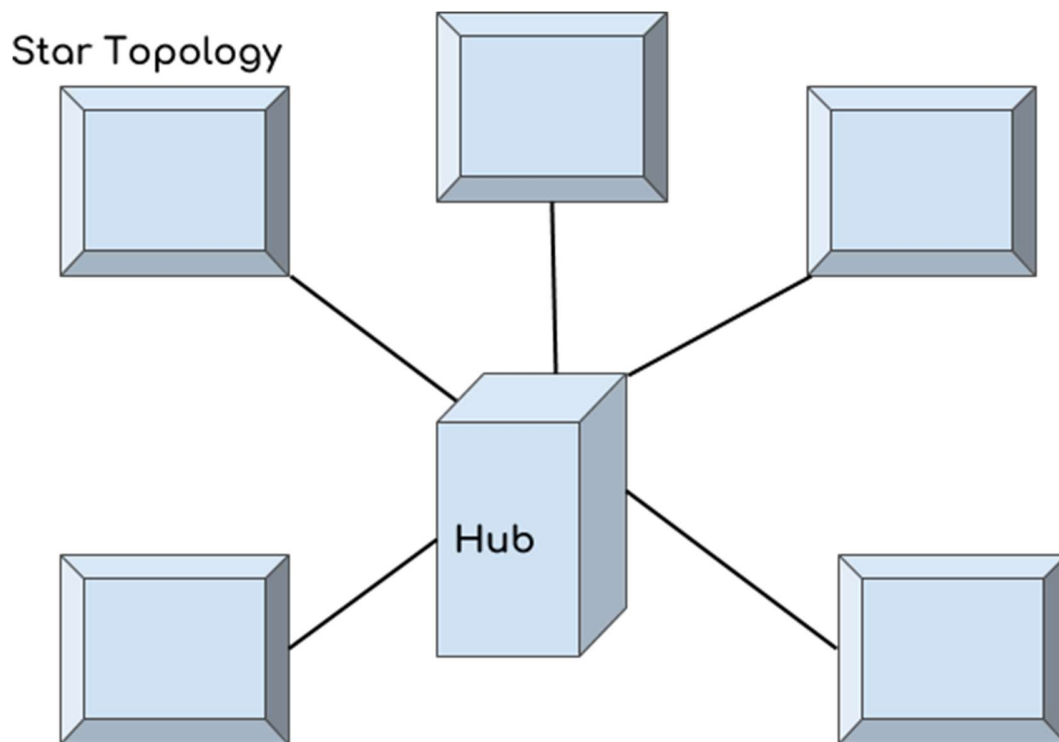
Advantages of Mesh topology

1. No data traffic issues as there is a dedicated link between two devices which means the link is only available for those two devices.
2. Mesh topology is reliable and robust as failure of one link doesn't affect other links and the communication between other devices on the network.
3. Mesh topology is secure because there is a point to point link thus unauthorized access is not possible.
4. Fault detection is easy.

Disadvantages of Mesh topology

1. Amount of wires required to connected each system is tedious and headache.
2. Since each device needs to be connected with other devices, number of I/O ports required must be huge.
3. Scalability issues because a device cannot be connected with large number of devices with a dedicated point to point link.

Star Topology



In star topology each device in the network is connected to a central device called hub. Unlike Mesh topology, star topology doesn't allow direct communication between devices, a device must have to communicate through hub. If one device wants to send data to other device, it has to first send the data to hub and then the hub transmit that data to the designated device.

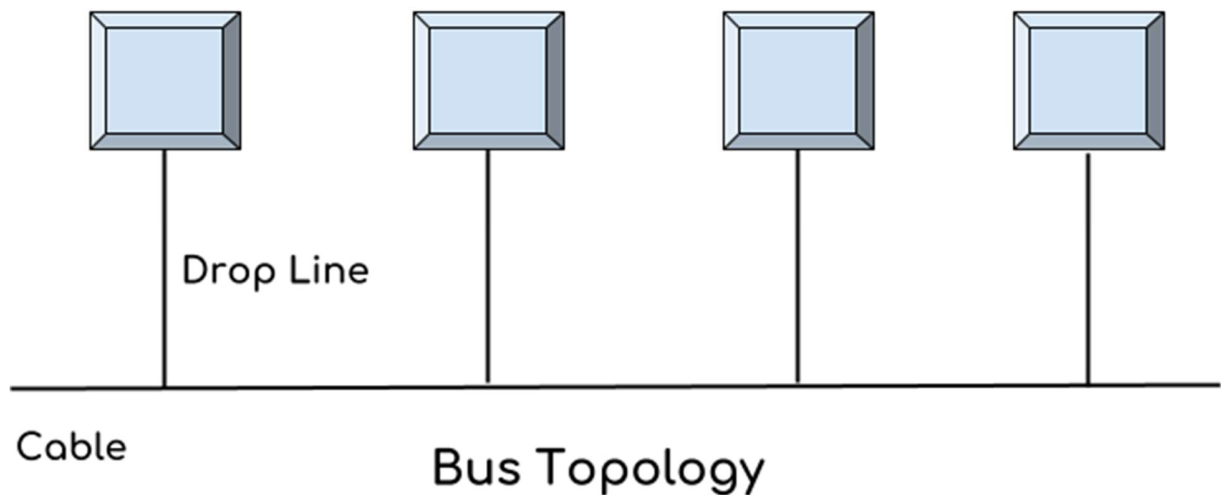
Advantages of Star topology

1. Less expensive because each device only need one I/O port and needs to be connected with hub with one link.
2. Easier to install
3. Less amount of cables required because each device needs to be connected with the hub only.
4. Robust, if one link fails, other links will work just fine.
5. Easy fault detection because the link can be easily identified.

Disadvantages of Star topology

1. If hub goes down everything goes down, none of the devices can work without hub.
2. Hub requires more resources and regular maintenance because it is the central system of star topology.

Bus Topology



In bus topology there is a main cable and all the devices are connected to this main cable through drop lines. There is a device called tap that connects the drop line to the main cable. Since all the data is transmitted over the main cable, there is a limit of drop lines and the distance a main cable can have.

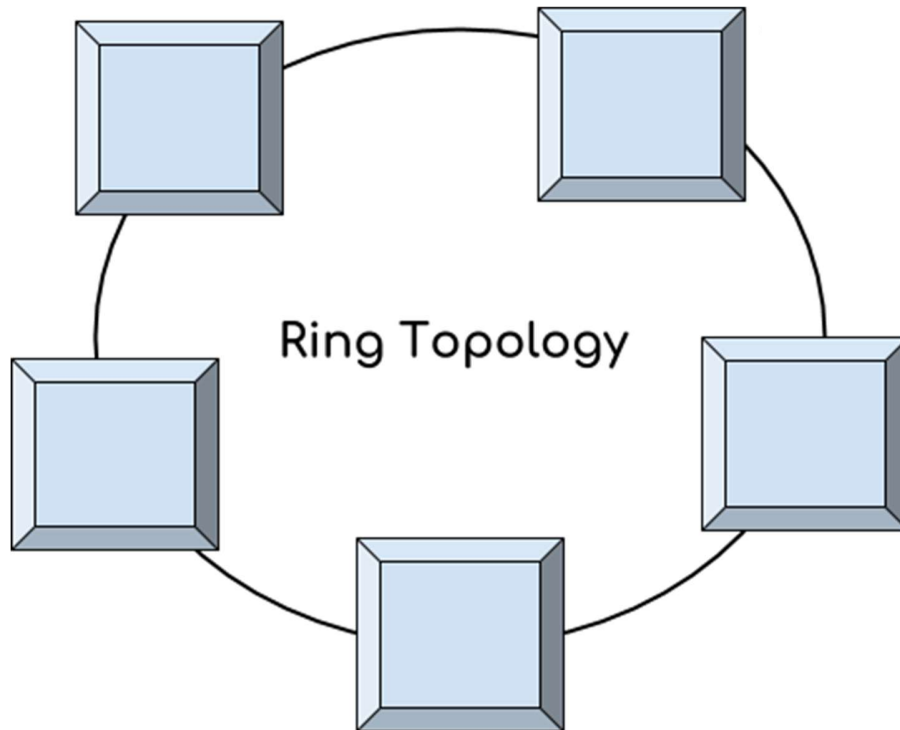
Advantages of bus topology

1. Easy installation, each cable needs to be connected with backbone cable.
2. Less cables required than Mesh and star topology

Disadvantages of bus topology

1. Difficultly in fault detection.
2. Not scalable as there is a limit of how many nodes you can connect with backbone cable.

Ring Topology



In ring topology each device is connected with the two devices on either side of it. There are two dedicated point to point links a device has with the devices on the either side of it. This structure forms a ring thus it is known as ring topology. If a device wants to send data to another device then it sends the data in one direction, each device in ring topology has a repeater, if the received data is intended for other device then repeater forwards this data until the intended device receives it.

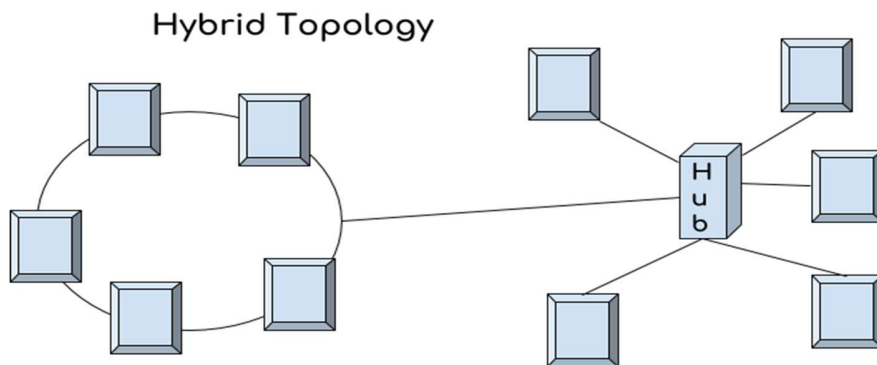
Advantages of Ring Topology

1. Easy to install.
2. Managing is easier as to add or remove a device from the topology only two links are required to be changed.

Disadvantages of Ring Topology

1. A link failure can fail the entire network as the signal will not travel forward due to failure.
2. Data traffic issues, since all the data is circulating in a ring.

Hybrid topology



A combination of two or more topology is known as hybrid topology. For example a combination of star and mesh topology is known as hybrid topology.

Advantages of Hybrid topology

1. We can choose the topology based on the requirement for example, scalability is our concern then we can use star topology instead of bus technology.
2. Scalable as we can further connect other computer networks with the existing networks with different topologies.

Disadvantages of Hybrid topology

1. Fault detection is difficult.
2. Installation is difficult.
3. Design is complex so maintenance is high thus expensive.

3.Types of network

There are many types of computer networks, the common types of area networks including those five:

LAN - Local Area Network,

WAN - Wide Area Network,

WLAN - Wireless Local Area Network,

MAN - Metropolitan Area Network and

CAN - Campus Area Network.

PAN-Personal Area Network

LAN (Local Area Network) - Can go up to 1 KM radius. A local area network (LAN) is a group of computers and associated devices that share a common communications line or wireless link to a server. Typically, a LAN encompasses computers and peripherals connected to a server within a distinct geographic area such as an office or a commercial establishment.

WAN (Wide Area Network) - No Limit. A wide area network (WAN) is a network that exists over a large-scale geographical area. A WAN connects different smaller networks, including local area networks (LANs) and metro area networks (MANs). This ensures that computers and users in one location can communicate with computers and users in other locations. WAN implementation can be done either with the help of the public transmission system or a private network.

WLAN(Wireless Local Area Network) - A wireless local area network (WLAN) is a wireless computer network that links two or more devices using wireless communication within a limited area such as a home, school, computer laboratory, or office building. This gives users the ability to move around within a local coverage area and yet still be connected to the network. Through a gateway, a WLAN can also provide a connection to the wider Internet.

Most modern WLANs are based on IEEE 802.11 standards and are marketed under the Wi-Fi brand name.

MAN(Metropolitan Area Network) - A metropolitan area network is a computer network that interconnects users with computer resources in a geographic area or region larger than that covered by even a large local area network (LAN) but smaller than the area covered by a wide area network (WAN). The term is applied to the interconnection of networks in a city into a single larger network (which may then also offer efficient connection to a wide area network). It is also used to mean the interconnection of several local area networks by bridging them with backbone lines. The latter usage is also sometimes referred to as a campus network.

CAN (Campus Area Network) - A campus area network is a computer network made up of an interconnection of local area networks (LANs) within a limited geographical area. The networking equipments (switches, routers) and transmission media (optical fiber, copper plant, Cat5 cabling etc) are almost entirely owned by the campus tenant / owner: an enterprise, university, government etc.

PAN (Personal Area Network) - The smallest and most basic type of network, a PAN is made up of a wireless modem, a computer or two, phones, printers, tablets, etc., and revolves around one person in one building. These types of networks are typically found in small offices or residences, and are managed by one person or organization from a single device.

VPN(Virtual Private Network) - A virtual private network extends a private network across a public network, and enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network. Applications running across the VPN may therefore benefit from the functionality, security, and management of the private network.

4. What are network devices?

Network devices, or networking hardware, are physical devices that are required for communication and interaction between hardware on a computer network.

Types of network devices

Here is the common network device list:

- Hub
- Switch
- Router
- Bridge

- Gateway
- Repeater
- Access Point

4.i. Bridge

Bridges are used to connect two or more hosts or network segments together. The basic role of bridges in network architecture is storing and forwarding frames between the different segments that the bridge connects. They use hardware Media Access Control (MAC) addresses for transferring frames. By looking at the MAC address of the devices connected to each segment, bridges can forward the data or block it from crossing. Bridges can also be used to connect two physical LANs into a larger logical LAN.

4.ii. Repeater – A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.

4.iii. Hub – A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have the intelligence to find out best path for data packets which leads to inefficiencies and wastage.

4.iv. Bridge – A bridge operates at data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.

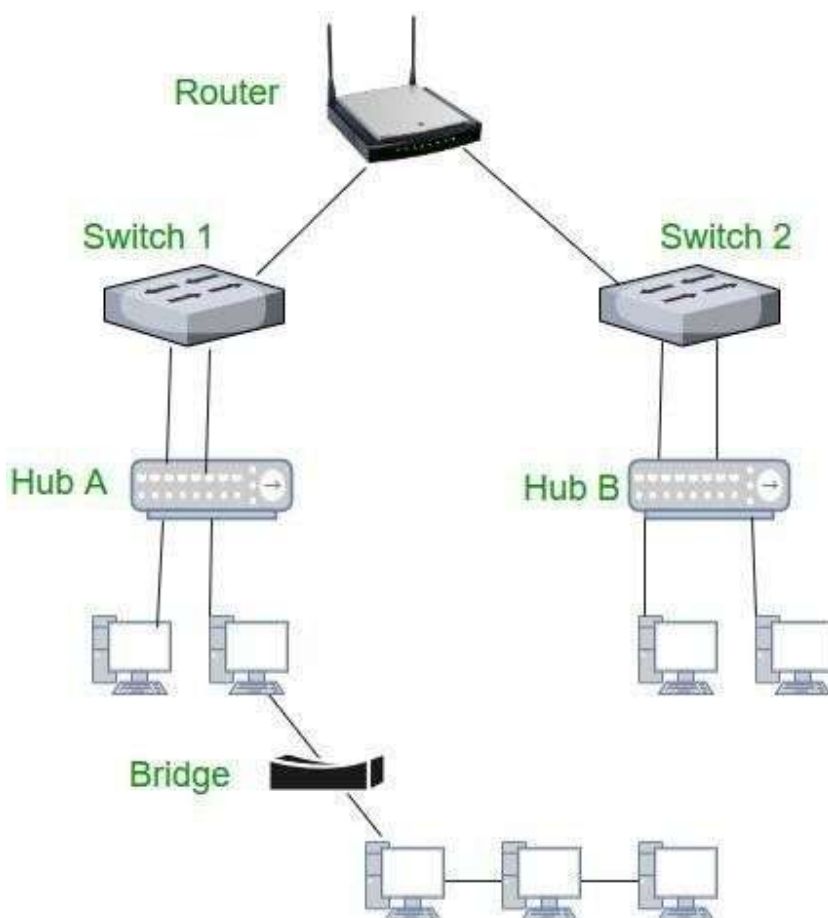
Types of Bridges

- **Transparent Bridges:-** These are the bridge in which the stations are completely unaware of the bridge's existence i.e. whether or not a bridge is added or deleted from the network, reconfiguration of the stations is unnecessary. These bridges make use of two processes i.e. bridge forwarding and bridge learning.

- **Source Routing Bridges:-** In these bridges, routing operation is performed by source station and the frame specifies which route to follow. The host can discover frame by sending a special frame called discovery frame, which spreads through the entire network using all possible paths to destination.

4.vSwitch – A switch is a multiport bridge with a buffer and a design that can boost its efficiency(a large number of ports imply less traffic) and performance. A switch is a data link layer device. The switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.

4.vi. Routers – A router is a device like a switch that routes data packets based on their IP addresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.



4.vii. Gateway – A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically work as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.

4.viii. Access Point :

While an access point (AP) can technically involve either a wired or wireless connection, it commonly means a wireless device. An AP works at the second OSI layer, the Data Link layer, and it can operate either as a bridge connecting a standard wired network to wireless devices or as a router passing data transmissions from one access point to another.

5. Internet Services like E-Mail, WWW, FTP, Chatting, Internet Conferencing, Electronic Newspaper & Online Shopping

- Internet is a network of networks that consists millions of private and public network of local to global scope.
- An internet user can access to a wide variety of services such as

E-MAIL (Electronic Mail)

- E-mail is an electronic version of sending and receiving letter.
- To use E-mail, a user must have an Email address.
- The Email address contains all information required to send or receive a message is called mail box.
- Email address consists of two parts separated by @ symbol
 - 1st part is user name
 - 2nd part is host name (domain name)
 - Example: smartclassroomgpb@gmail.com

WWW (World Wide Web)

- The World Wide Web is a system of internet servers that supports hypertext and multimedia to access several internet protocols on a single interface.

- It is a way of exchanging information between computers on the internet.
Example:

<https://www.google.com/>

<http://www.bosecuttack.in/>

FTP (File Transfer Protocol)

- FTP is the internet file transfer between any computers that have an internet connection and also works between computers using totally different operating systems.
- It is a protocol through which internet users can upload files from their computers to a website or download files from a website to their PC.
- It is the easiest way to transfer files between computers via the internet and utilities TCP/IP systems to perform uploading and downloading tasks.

Chatting

- Chatting is the online textual or multimedia conversation.
- Chatting i.e., a virtual means of communication that involves the sending and receiving of messages, share audio and video between users located in any part of the world.

Internet Conferencing

- Internet conferencing allows users to carry on business meetings and seminars make presentation, provide online education and offer direct customer support.
- Internet conferencing solutions require high speed internet connection at all user site.

Electronic Newspaper

- An electronic newspaper is a self-contained, reusable and refreshable version of a traditional newspaper that acquires and holds information electronically.
- Information to be displayed will be downloaded through some wireless internet connections.

Online Shopping

- It is the process of buying goods and services from merchants who sell on the internet.
- The main components of online shopping are product, selling price, accessibility to people, placement of orders, mode of payments, delivery mechanism.

6. Different types of Internet Connectivity

WIRELESS

Radio frequency bands are used in place of telephone or cable networks. One of the greatest advantages of wireless Internet connections is the "always-on" connection that can be accessed from any location that falls within network coverage. Wireless connections are made possible through the use of a modem, which picks up Internet signals and sends them to other devices.

MOBILE

Many cell phone and smartphone providers offer voice plans with Internet access. Mobile Internet connections provide good speeds and allow you to access the Internet.

HOTSPOTS

Hotspots are sites that offer Internet access over a wireless local area network (WLAN) by way of a router that then connects to an Internet service provider. Hotspots utilize WiFi technology, which allows electronic devices to connect to the Internet or exchange data wirelessly through radio waves. Hotspots can be phone-based or free-standing, commercial or free to the public.

DIAL-UP

Dial-up connections require users to link their phone line to a computer in order to access the Internet. This particular type of connection—also referred to as analog—does not permit users to make or receive phone calls through their home phone service while using the Internet.

BROADBAND

This high-speed Internet connection is provided through either cable or telephone companies. One of the fastest options available, broadband Internet uses multiple data channels to send large quantities of information. The term broadband is shorthand for broad bandwidth. Broadband Internet connections such as DSL and cable are considered high-bandwidth connections. Although many DSL connections can be considered broadband, not all broadband connections are DSL.

DSL

DSL, which stands for Digital Subscriber Line, uses existing 2-wire copper telephone line connected to one's home so service is delivered at the same time as landline telephone service. Customers can still place calls while surfing the Internet.

CABLE

Cable Internet connection is a form of broadband access. Through use of a cable modem, users can access the Internet over cable TV lines. Cable modems can provide extremely fast access to the Internet.

SATELLITE

In certain areas where broadband connection is not yet offered, a satellite Internet option may be available. Similar to wireless access, satellite connection utilizes a modem.

ISDN

ISDN (Integrated Services Digital Network) allows users to send data, voice and video content over digital telephone lines or standard telephone wires. The installation of an ISDN adapter is required at both ends of the transmission—on the part of the user as well as the Internet access provider.

Chapter-4

File Management and data processing

4.1. Concept of File and Folder

4.2. file Access and Storage

methods

4.3. Sequential, Direct, ISAM

4.4 Data Processing and Retrieval

File :

File is nothing but an Electronic document. The contents can be ordinary Text or it can be an executable program. Each file is given a file name to identify it. The File name is in the form File Name . Extension Filename can consist of Alphabets or combinations of alphabets, numerals and special characters. Extension indicates the type of file.

Example : XY.Doc

Here File name is XY

Extension name is .DOC which indicates Document file.

Folder:

Folder contains a group of files. Folder is otherwise called as directory. Folder may have a set of files under it. It may have other folders under it also. This files and folders can be arranged in hierarchical manner or a tree like structure.

Computer - Memory Units:

Memory unit is the amount of data that can be stored in the storage unit.

This storage capacity is expressed in terms of Bytes.

The following explains the main memory storage units –

- 1. Bit (Binary Digit):** A binary digit is logical 0 and 1 representing a passive or an active state of a component in an electric circuit.
- 2. Nibble :** A group of 4 bits is called nibble.

3 .Byte : A group of 8 bits is called byte. A byte is the smallest unit, which can represent a data item or a character.

4 .Word : A computer word, like a byte, is a group of fixed number of bits processed as a unit, which varies from computer to computer but is fixed for each computer. The length of a computer word is called word-size or word length. It may be as small as 8 bits or may be as long as 96 bits. A computer stores the information in the form of computer words.

The following table lists some higher storage units-

1. Kilobyte (KB)

1 KB = 1024 Bytes

2. Megabyte (MB)

1 MB = 1024 KB

3 .GigaByte (GB)

1 GB = 1024 MB

4 .TeraByte (TB)

1 TB = 1024 GB

5 .PetaByte (PB)

1 PB = 1024 TB

File organization :

The arrangement of records in a file is known as File Organisation. File Organisation deals with the arrangement of data items in the secondary storage devices like magnetic disk. That is, the file organisation deals with how the logical tuples (rows) of tables (relations) are organised on the physical storage medium.

For organising records efficiently in the form of a computer file, following three things are important:

(a) A logical method should be selected to organise records in a file.

(b) File structure should be so designed that it would allow quick access to needed data items.

(c) Means of adding or deleting data items or records from files must be present.

Depending on the above considerations, a file may be organised as:

(a) Sequential file

(b) Direct or random access file

(c) Indexed-sequential file

Sequential Access –

It is the simplest access method. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion.

Read and write make up the bulk of the operation on a file. A read operation *-read next-* read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the write *write next* append to the end of the file and advance to the newly written material.

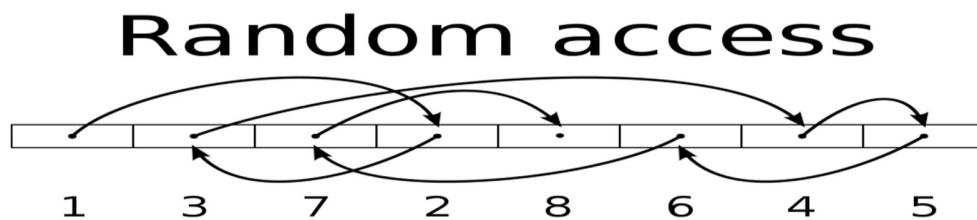


Key points

- Data is accessed one record right after another record in an order.
- When we use read command, it move ahead pointer by one
- When we use write command, it will allocate memory and move the pointer to the end of the file
- Such a method is reasonable for tape.

Direct Access or Random Access –

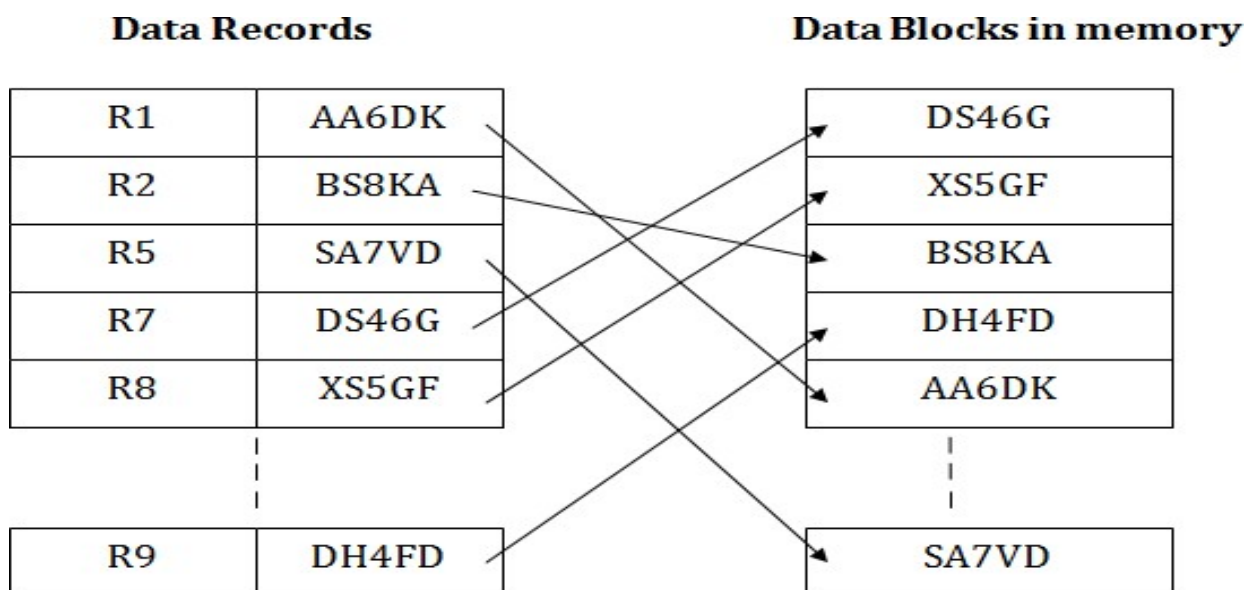
Another method is *direct access method* also known as *relative access method*. A file-length logical record that allows the program to read and write record rapidly. in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 4 then block 15 and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file.



A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.

Index sequential method –

It is the other method of accessing a file which is built on the top of the sequential access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index and then by the help of pointer we access the file directly.



Key points:

- It is built on top of Sequential access.
- It control the pointer by using index.

Data capture:

Data capture, or electronic data capture, is **the process of extracting information from a document and converting it into data readable by a computer**. More generally, data capturing can also refer to collecting relevant information whether sourced from paper or electronic documents.

Methods of capture from documents in electronic format are identified below:

1. OCR:

OCR technology revolutionised data capture and underpinned the digitisation and automation of back office operations that involved the processing of paper or digital documents such as PDF invoices, contracts, ID etc.

As a technology it provides the ability to recognise machine produced characters as part a data capture and extraction process. OCR systems can recognise many different OCR fonts, as well as typewriter and computer-generated characters.

2. ICR:

ICR is the computer translation of hand printed and written characters. A scanned image of a handwritten document is analysed and recognised by sophisticated ICR software. ICR is similar to optical character recognition (OCR) but is a more difficult process since OCR is from printed text, as opposed to handwritten characters which are more variable.

3. Bar code Recognition:

Dependent upon the type of barcode that is used, the amount of metadata that can be included or marked up can be high, as is the level of recognition. QR codes for example can contain webpage links ultimately linking a webpage of almost anything and any amount of information. Barcodes can be applied to documents, webpages or almost any objects for a range of purposes including inventory management, location or task tracking, webpage opening or authentication via authenticated app, production batch tracking, delivery notes, digital form locating and more. Smartphone with barcode

applications have removed the need for dedicated barcode scanning tools making barcode use even more affordable.

4.IDR:

Intelligent document recognition also interprets and indexes different documents based on the document type, its meta data and elements of the document identified. For example, invoices, letters, contracts, post codes, logos, key words, VAT registration numbers. Data that has been identified through OCR can be validated and verified through look-up tables and database as well configure or “taught” rules associated with such documents and data. and even databases to maximise accuracy.

5.Artificial Intelligence and data capture:

Artificial Intelligence is ultimately an umbrella terms for different artificial intelligence techniques. AI is best viewed in context of the use case and application. All of the methods described here can be augmented to some degree or another by Artificial intelligence such as

- Computer vision, Image or pattern recognition to improve the recognition of any type image.
- Neural Networks & Machine learning to assist with accurate recognition training based on large data sets and assisted learning.
- Natural Language Processing for interpreting sentences and their meaning.

Data Storage:

Data storage is the holding of data in an electromagnetic form for access by a computer processor.

There are two kinds of storage:

- a) Primary storage is data that is held in RAM and other memory devices that are built into computers.
- b) Secondary storage is data that is stored on external storage devices such as hard disks, tapes, CD, Pen drive etc.

Following are some main devices for data storage:

- Hard disks
- Floppy disks
- Optical disks

- CD
- Pen drives
- Flat memory card/memory card

Data Processing:

- Data processing must be processed in order to convert it into information.
 - For this purpose, different operations may be performed on data.
 - Data processing is defined as a sequence of operations on data to convert it into useful information.
 - The data processing can be accomplished through following methods:
- 1. Manual Data Processing:**
 - In this method, data is processed manually without using any machine or tool to get required results.
 - In manual data processing, all the calculations and logical operations are performed manually on the data.
 - Ex: Mark sheets, fee receipts
 - 2. Mechanical Data Processing:**
 - In this method, data is processed by using different devices like typewriters, mechanical printers or other mechanical devices.
 - Examination board and printing press use mechanical data processing devices frequently.
 - 3. Electronic Data Processing:**
 - It is the modern technique to process data.
 - The data is processed through computer; data and set of instructions are given to the computer as input and the data according to the given set of instructions.
 - The computer is also known as electronic data processing machine.
 - Ex: results of students are prepared through computers.

Data Retrieval:

- Data is one of the most important assets of any business.
 - Data recovery refers to the whole process of salvaging this lost data that is corrupted, failed, damaged or inaccessible.
 - Lost files can occur because of any of the below possibilities.
 - 1) File was mistakenly deleted.
 - 2) File was corrupted or deleted by scandisk.
 - 3) Another program deleted the file
 - 4) File is password protected.
- Following are some different methods of data recovery:
- 1) Physical damage to storage devices:

- Different failure can cause physical damage to your storage media.
- 2) Media errors and corrupt partitions and file systems:
 - In some cases, media errors or damage to the file system or partition table can make the data on a hard drive to be unreadable.
- 3) Online data recovery:
 - This is another popular method of data recovery Sydney business use to restore deleted or lost files.
 - It is a method of data recovery that is performed over the internet without necessarily having the computer or the drive in possession.

Chapter-5

PROBLEM SOLVING METHODOLOGY

- 5.1. Algorithm,Pseudocode and Flowchart
- 5.2. Generation of Programming languages
- 5.3. Structured Programming Language
- 5.4. Examples of Problems solving through Algorithm
- 5.5.Examples of problem solving through flowchart

Problem solving

Solving problems is the core of computer science. Programmers must first understand how a human solves a problem, then understand how to translate this "algorithm" into something a computer can do, and finally how to "write" the specific syntax (required by a computer) to get the job done. It is sometimes the case that a machine will solve a problem in a completely different way than a human.

Computer Programmers are problem solvers. In order to solve a problem on a computer you must:

1. Know how to represent the information (data) describing the problem.
2. Determine the steps to transform the information from one representation into another

Algorithm, Pseudo code and Flowchart

Algorithm

An algorithm is a set of specific steps to solve a problem. Think of it this way: if you were to tell your 3 year old niece to play your favorite song on the piano (assuming the niece has never played a piano), you would have to tell her where the piano was, and how to sit on the bench, and how to open the cover, and which keys to press, and which order to press them in, etc, etc, etc.

Definition:

- An algorithm is a well-defined procedure that allows a computer to solve a problem.
- Algorithm is defined as the step-by-step solution of problem in user's language.
- It is considered as an effective procedure for solving a problem in finite number of steps.
- Another way to describe an algorithm is a sequence of unambiguous instructions.
- In fact, it is difficult to think of a task performed by your computer that does not use algorithms.

The characteristics of Algorithm are

- Precise
- Unambiguous
- Finite termination

- Unique solution

Once algorithm is written, it can be coded into a program using any programming language. Algorithm uses 3 different constructs

- Sequence
- Branching or Decision making
- Repetition

Sequence says that instructions are to be executed in what order or sequence. Branching involves testing of condition and based on the outcome of the condition testing different instructions are executed. Repetition means one or more instructions shall be repeated for a number of times. This is otherwise called as loop. There are different types of loops such as While- do , do-while for.

Example:

1. Write an algorithm to find the sum of two numbers.

Step 1 : Start
Step 2 : Accept First
Number
Step 3 : Accept Second
Number
Step 4 : Add These Two
Numbers
Step 5 : Display Result
Step 6 : Stop

2. Write an algorithm to find the sum of three numbers.

Step 1 : Start
Step 2 : Accept All Three
Numbers
Step 3 : Add All Three

Numbers And Store In One
Variable

Step 4 : Display The
Result

Step 5 : Stop

**3. Write an algorithm to find the area of
rectangle**

Step 1 : Start

Step 2 : Accept The W Of
Rectangle

Step 3 : Accept The H Of
Rectangle

Step 4 : $\text{Area} = W \times H$

Step 5: Display

Step 6 : Stop

**4. Write an algorithm to find the
largest among three different
numbers entered by user.**

Step 1: Start

Step 2: Declare variables a, b
and c.

Step 3: If $a > b$ If
 $a > c$ Display a is
the largest Else

Display c is the
largest Else

If $b > c$

Display b is the
largest Else

Display c
greatest

Step 4: Stop

5. Algorithm to find out sum of first 10 natural numbers.

Step 1: $i=1$, $Sum=0$

Step2: Repeat step 3 and 4 while
 $i \leq 10$

Step 3: $Sum = Sum + i$

Step 4: $i = i + 1$

Step5: PrintSum






Pseudocode

It is a concise description algorithm in English language that uses programming language constructs. It contains outlines of the program that can be easily converted to program. It focuses on the logic of the algorithm without giving stress on the syntax of programming language. This is meant for understanding the logic of the program easily. Flowchart can be considered as an alternative to pseudo code. Several constructs/key words of programming language can be used in the algorithm to write the pseudo code.

Some of them
are If ... Endif
Do while ... end
do While ... end
while Repeat ...
until
For ... end for
Case end case
Call
Return

Flowchart

Flowchart is a graphical or symbolic representation of the process of solution to a problem or algorithm. It helps to visualize the complex logic of the solution of the problem in a simplified manner through diagrammatic representation. Each step of the algorithm is presented using a symbol and a short description. The different symbols used for the flowchart are

Flow Chart Symbol	Meaning	Explanation
	Start and end	The symbol denoting the beginning and end of the flow chart.
	Step	This symbol shows that the user performs a task. (Note: In many flow charts steps and actions are interchangeable.)
	Decision	This symbol represents a point where a decision is made.
	Action	This symbol means that the user performs an action. (Note: In many flow charts steps and actions are interchangeable.)
	Flow line	A line that connects the various symbols in an ordered way.

Generation of Programming Languages

Programming Language Programming language is a tool to express the logic or instructions for understanding of the computer. Any programming language has two components:

- Syntax
- Semantics

Syntax refers to the rules to be followed for writing valid program statements. Compiler can detect errors in syntax while compiling the program. Semantics is associated with logic of the program.

Compiler cannot detect the semantic error. The user or programmer can diagnose semantic error. There are good numbers of High level languages, each meant for specific area of data processing.

Commonly known languages are BASIC, FORTRAN, COBOL, Pascal, C, C++ etc.

While FORTRAN is good for Numerical and scientific calculation, COBOL is good for Business applications involving large amount of data handling.

Generations of Programming Language

The Programming languages can be classified into 4 generations:

1stGeneration: Machine Language

2ndGeneration: Assembly Language

3rdGeneration: High Level Language

4thGeneration: Very High Level Language

Machine Level language contains instructions in binary form i.e. in 0s and 1s. Thus writing instructions was very difficult and needs heavy expertise. This was used in early days computers.

Assembly level language instructions were written using symbolic codes known as mnemonics. In comparison to Machine language, it is relatively easier to write program, but still it requires lot of expertise. A translator called assembler is used to translate assembly language program to machine level language.

High level language contains instructions in English like words so that user will feel easier to formulate and write the logical statements of the program. Here the logic may spread over multiple statements as against a single statement in assembly language. It uses a translator called compiler for translation of High level language program to machine level language program. There are many High level languages used for programming such as BASIC, FORTRAN, COBOL, PASCAL, C, C++ etc.

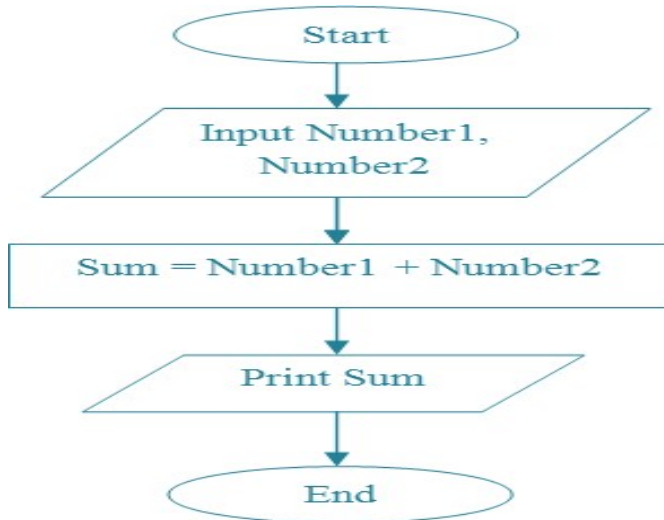
Very High Level language otherwise called as 4GL uses nonprocedural logical statements. A typical example of 4GL is the query language such as SQL.

Structured Programming Language

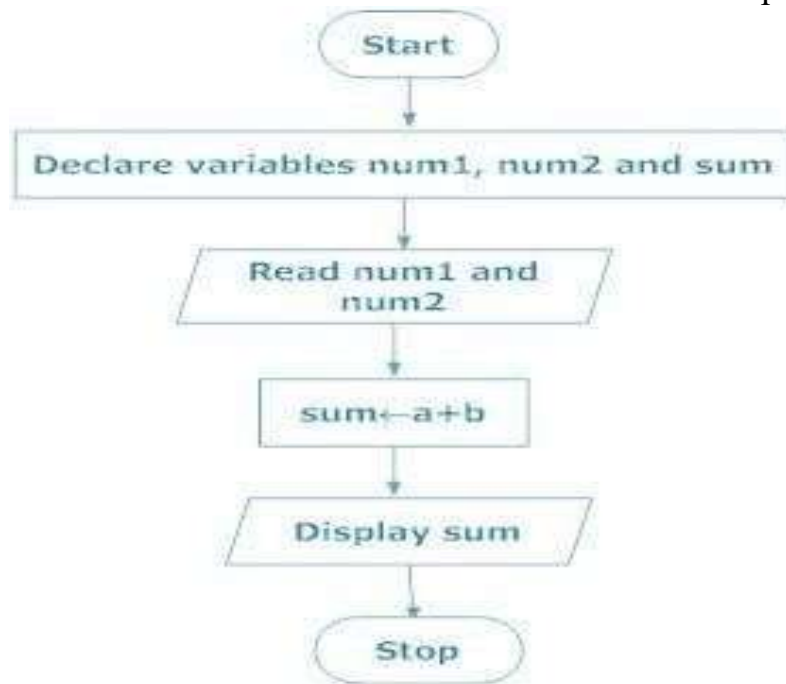
Structured Programming is also known as Modular Programming. In this type of programming technique, the program shall be broken into several modules. This helps in managing memory efficiently as the required module of the program will be loaded into the memory only and not the entire program. This will also enhance code reuse. Writing, understanding, debugging and modifying the individual module of the program is also easier.

Examples of Problem solving through Flowchart

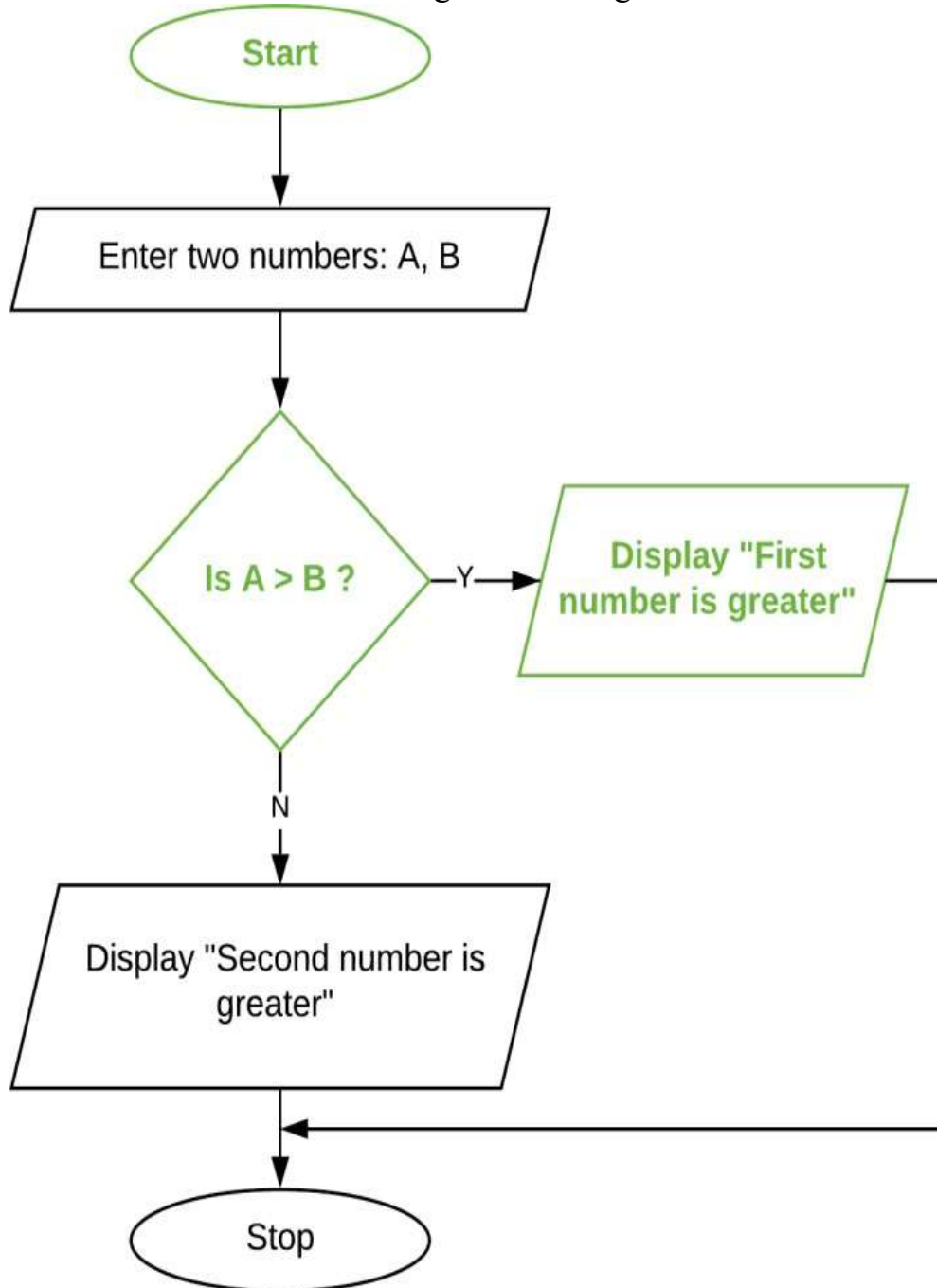
1. Draw a flow chart to add two number



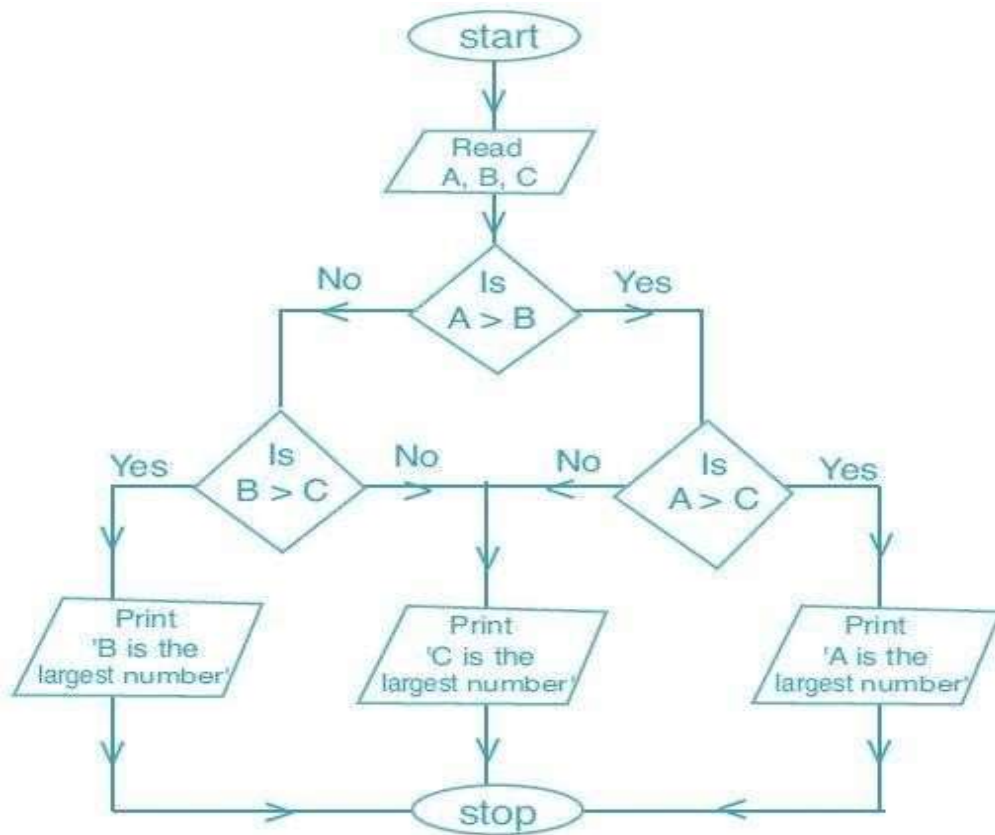
2. Draw a flowchart to add two number where input are given by user.



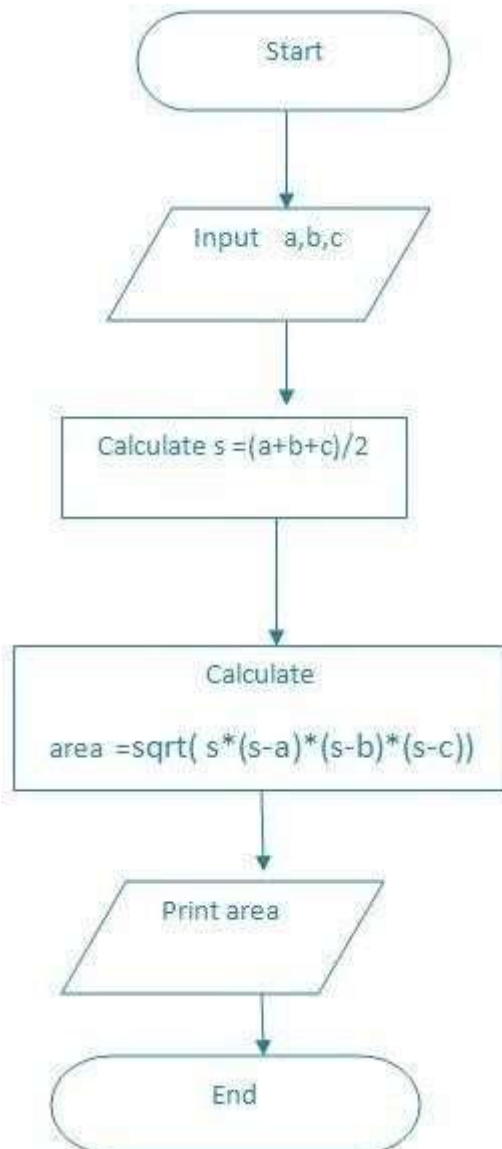
3. Draw a flowchart to find greater among two number.



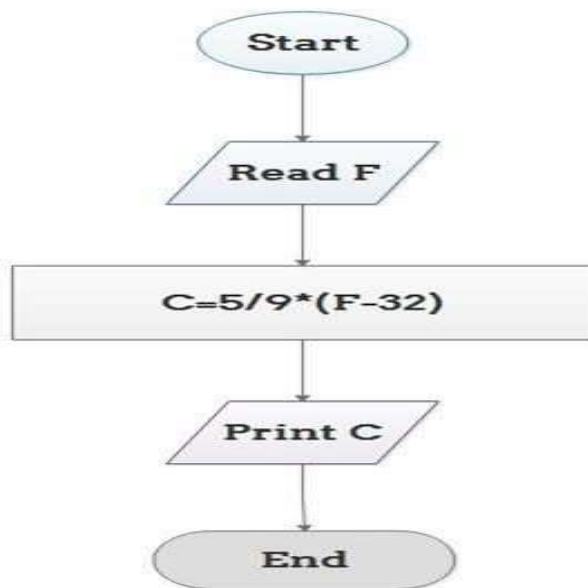
4. Draw a flowchart to find geater among three number.



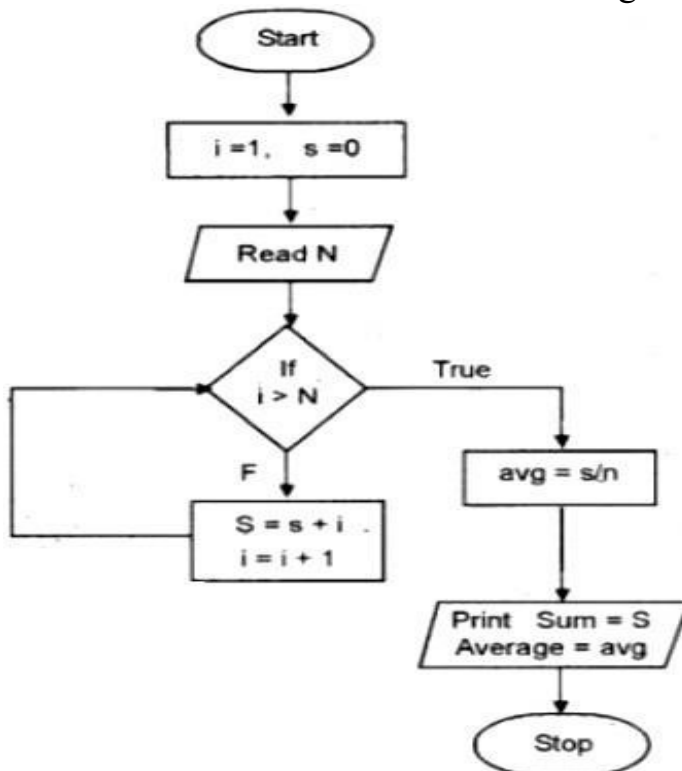
5. Draw a flowchart to find area of a triangle.



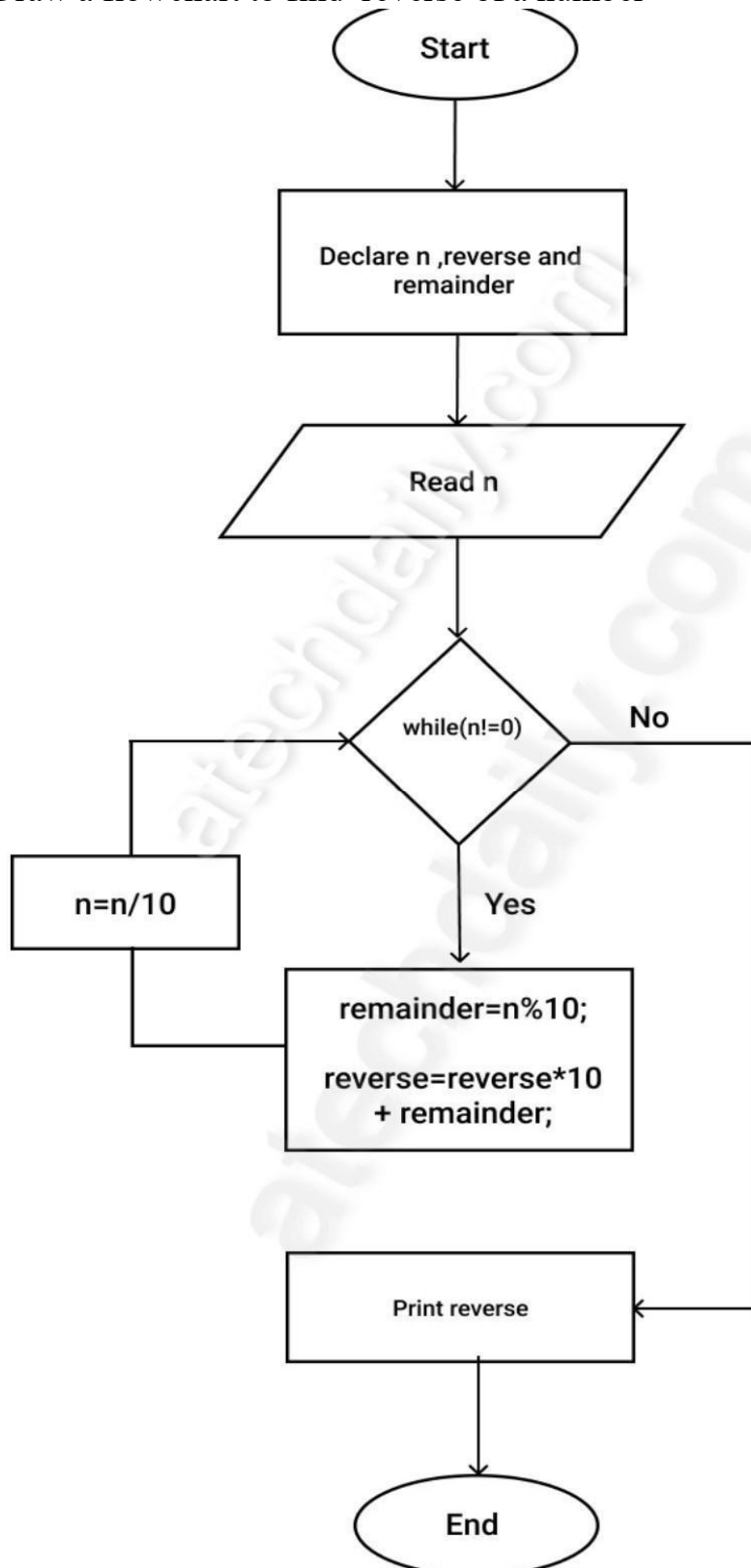
6. Draw a flowchart to convert temperature in Fehrenheit to Celsius



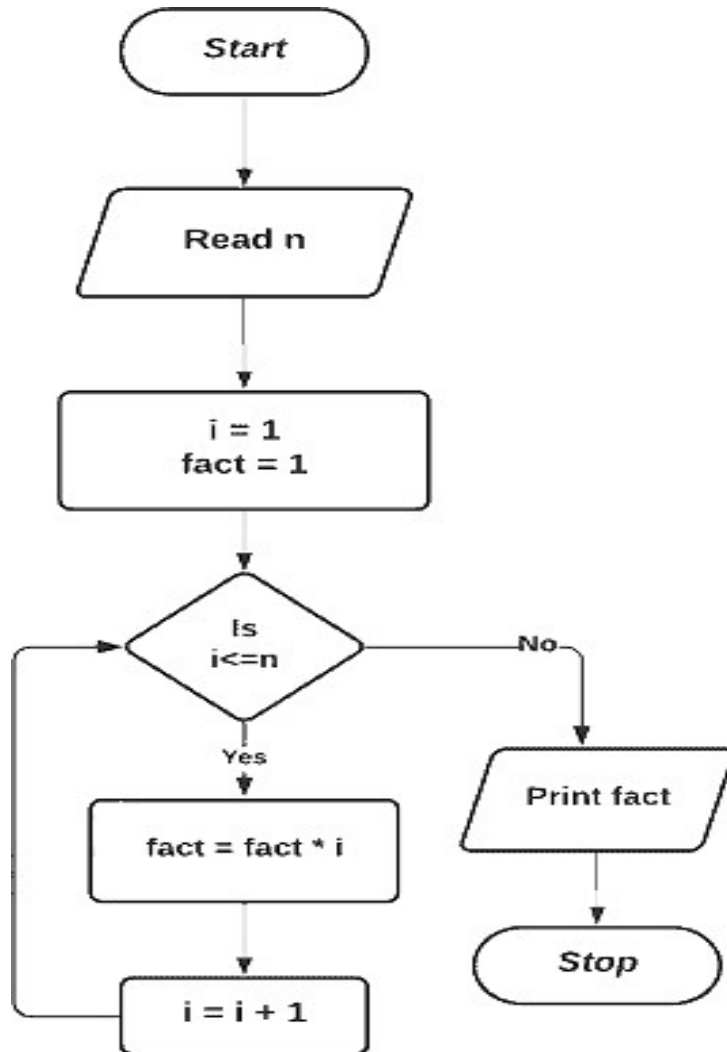
7. Draw a flowchart to find sum and average of first n integer



8. Draw a flowchart to find reverse of a number



9. Draw a flowchart to Check whether a number is prime or not.



Chapter-6

OVERVIEW OF C PROGRAMMING LANGUAGE

- 6.1. Introduction to C
- 6.2. C program structure
- 6.3. Preprocessor
- 6.4. Character set
- 6.5. Constants, Variable and Data Types
- 6.6. Managing Input and output operations
- 6.7. Operators, Expressions, Type conversion and typecasting
- 6.8. Decision control and looping statement (If, If-else, If-else-if, switch, while, Do-while, for, break, continue and goto).
- 6.9. Programming assignment using the above features

The C Language C is a professional programmer's language. It was designed to get in one's way as little as possible. Kernighan and Ritchie wrote the original language definition in their book, *The C Programming Language* (below), as part of their research at AT&T. Unix and C++ emerged from the same labs. For several years I used AT&T as my long distance carrier in appreciation of all that CS research, but hearing "thank you for using AT&T" for the millionth time has used up that goodwill.

Important Points

➤ The C Language is developed by Dennis Ritchie for creating system applications that directly interact with the hardware devices such as drivers, kernels, etc.

➤ C programming is considered as the base for other programming languages, that is why it is known as mother language.

➤ It can be defined by the following ways:

1. Mother language
2. System programming language
3. Procedure-oriented programming language
4. Structured programming language
5. Mid-level programming language

1) C as a mother language C language is considered as the mother language of all the modern programming languages because most of the compilers, JVMs, Kernels, etc. are written in C language, and most of the programming languages follow C syntax, for example, C++, Java, C#, etc. It provides the core concepts like the array, strings, functions, file handling, etc. that are being used in many languages like C++, Java, C#, etc.

2) C as a system programming language A system programming language is used to create system software. C language is a system programming language because it can be used to do low-level programming (for example driver and kernel). It is generally used to create hardware devices, OS, drivers, kernels, etc. For example, Linux kernel is written in C. It can't be used for internet programming like Java, .Net, PHP, etc.

3) C as a procedural language A procedure is known as a function, method, routine, subroutine, etc. A procedural language specifies a series of steps for the program to solve the problem. A procedural language breaks the program into functions, data structures, etc. C is a procedural language. In C, variables and function prototypes must be declared before being used.

4) C as a structured programming language A structured programming language is a subset of the procedural language. Structure means to break a program into parts or blocks so that it may be easy to understand. In the C language, we break the program into parts using functions. It makes the program easier to understand and modify.

5) **C as a mid-level programming language** C is considered as a middle-level language because it supports the feature of both lowlevel and high-level languages. C language program is converted into assembly code, it supports pointer arithmetic (low-level), but it is machine independent (a feature of high-level). A Low-level language is specific to one machine, i.e., machine dependent. It is machine dependent, fast to run. But it is not easy to understand. A High-Level language is not specific to one machine, i.e., machine independent. It is easy to understand.

Quick History of C

- Developed at Bell Laboratories in the early seventies by Dennis Ritchie.
- Born out of two other languages – BCPL (Basic Control Programming Language) and B.
- C introduced such things as character types, floating point arithmetic, structures, unions and the preprocessor.
- The principal objective was to devise a language that was easy enough to understand to be "high-level" – i.e. understood by general programmers, but low-level enough to be applicable to the writing of systems-level software.
- The language should abstract the details of how the computer achieves its tasks in such a way as to ensure that C could be portable across different types of computers, thus allowing the UNIX operating system to be compiled on other computers with a minimum of re-writing.
- C as a language was in use by 1973, although extra functionality, such as new types, was introduced up until 1980.
- In 1978, Brian Kernighan and Dennis M. Ritchie wrote the seminal work The C Programming Language, which is now the standard reference book for C.
- A formal ANSI standard for C was produced in 1989.
- In 1986, a descendant of C, called C++ was developed by Bjarne Stroustrup, which is in wide use today. Many modern languages such as C#, Java and Perl are based on C and C++.
- Using C language scientific, business and system-level applications can be developed easily.

6.1.1 Constants In C programming language, a constant is similar to the variable but the constant hold only one value during the program execution. That means, once a value is assigned to the constant, that value can't be changed during the program execution. Once the value is assigned to the constant, it is fixed throughout the program. A constant can be defined as follows... **A constant is a named memory location which holds only one value throughout the program execution.**

In C programming language, a constant can be of any data type like integer, floatingpoint, character, string and double, etc.,

Integer constants An integer constant can be a decimal integer or octal integer or hexadecimal integer. A decimal integer value is specified as direct integer value whereas octal integer value is prefixed with 'O' and hexadecimal value is prefixed with 'OX'. An integer constant can also be unsigned type of integer constant or long type of integer constant. Unsigned integer constant value is suffixed with 'u' and long integer constant value is suffixed with 'l' whereas unsigned long integer constant value is suffixed with 'ul'.

Example

125-----> Decimal Integer Constant
076 -----> Octal Integer Constant
0X3A -----> Hexa Decimal Integer Constant
50u-----> Unsigned Integer Constant
30l-----> Long Integer Constant
100ul ----> Unsigned Long Integer Constant

Floating Point constants

A floating-point constant must contain both integer and decimal parts. Sometimes it may also contain the exponent part. When a floating-point constant is represented in exponent form, the value must be suffixed with 'e' or 'E'.

Example The floating-point value 3.14 is represented as 3E-14 in exponent form.

Character Constants

A character constant is a symbol enclosed in single quotation. A character constant has a maximum length of one character.

Example

'A'
'2'
'+'

In the C programming language, there are some predefined character constants called escape sequences. Every escape sequence has its own special functionality and every escape sequence is prefixed with '\ ' symbol. These escape sequences are used in output function called 'printf()'.

String Constants

- A string constant is a collection of characters, digits, special symbols and escape sequences that are enclosed in double quotations.
- We define string constant in a single line as follows.... "This is Diploma smart class"
- We can define string constant using multiple lines as follows...
" This\
is\
Diploma smart class "
- We can also define string constant by separating it with white space as follows...
"This" "is" "Diploma smart class"
All the above three defines the same string constant.

Creating constants in C

□ In a c programming language, constants can be created using two concepts...

Using the —**const** keyword

○ Using —**#define** preprocessor

Using the “const” keyword

□ We create a constant of any data type using 'const' keyword. To create a constant, we prefix the variable declaration with 'const' keyword.

□ The general syntax for creating constant using 'const' keyword is as follows... const datatype constantName ;

OR

const data type constant Name = value ;

Example

```
const int x = 10 ;
```

Here, 'x' is a integer constant with fixed value 10.

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i = 9 ;
const int x = 10 ;
i = 15 ;
x = 100 ;
// creates an error
printf("i = %d\nx = %d", i, x ) ;
}
```

The above program gives an error because we are trying to change the constant variable value (x = 100).

Using '#define' preprocessor

We can also create constants using '#define' preprocessor directive. When we create constant using this preprocessor directive it must be defined at the beginning of the program (because all the preprocessor directives must be written before the global declaration).

We use the following syntax to create constant using '#define'

preprocessor directive... #define CONSTANTNAME value

Example

```
#define PI 3.14
```

Here, PI is a constant with value 3.14

Example Program

```
#include<stdio.h>
#include<conio.h>
#define PI 3.14
void main()
{
int r, area ;
printf("Please enter the radius of circle : ") ;
scanf("%d", &r) ;
area = PI * (r * r) ;
printf("Area of the circle = %d", area) ;
}
```

6.1.2 Variables:

Variables in a c programming language are the named memory locations where the user can store different values of the same datatype during the

program execution. That means a variable is a name given to a memory location in which we can store different values of the same data type. In other words, a variable can be defined as a storage container to hold values of the same data type during the program execution. The formal definition of a data type is as follows.

Variable is a name given to a memory location where we can store different values of the same datatype during the program execution.

Every variable in c programming language must be declared in the declaration section before it is used. Every variable must have a data type that determines the range and type of values be stored and the size of the memory to be allocated.

A variable name may contain letters, digits and underscore symbol. The following are the rules to specify a variable name...

- ☐ Variable name should not start with a digit.
- ☐ Keywords should not be used as variable names.
- ☐ A variable name should not contain any special symbols except underscore (_).
- ☐ A variable name can be of any length but compiler considers only the first 31 characters of the variable name.

Declaration of Variable

Declaration of a variable tells the compiler to allocate the required amount of memory with the specified variable name and allows only specified datatype values into that memory location. In C programming language, the declaration can be performed either before the function as global variables or inside any block or function. But it must be at the beginning of block or function.

Declaration Syntax:

datatype variableName;

Example

```
int number;
```

The above declaration tells to the compiler that allocates 2 bytes of memory with the name —

number and allows only integer values into that memory location. **6.1.3**

Data types

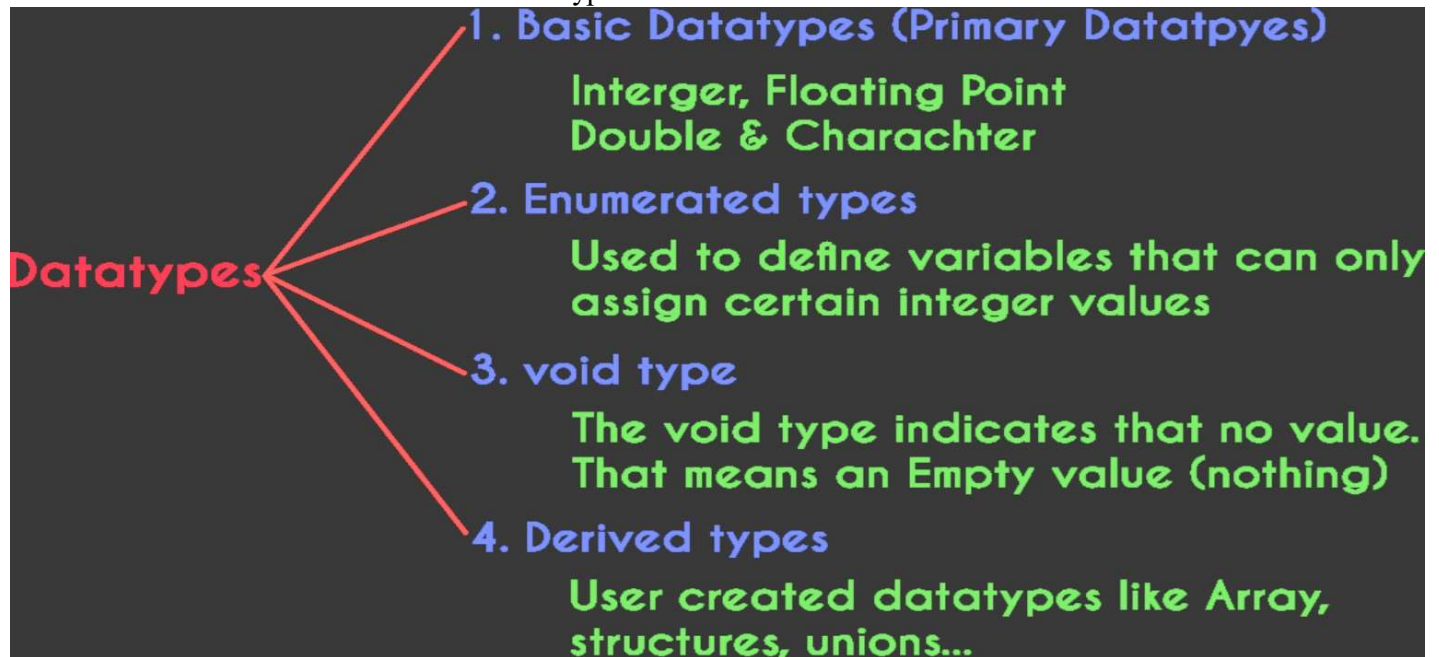
Data used in c program is classified into different types based on its properties. In the C programming language, a data type can be defined as a set of values with similar characteristics. All the values in a data type have the same properties. Data types in the c programming language are used to specify what kind of value can be stored in a variable. The memory size and type of the value of a variable are determined by the variable data type. In a c program, each variable or constant or array must have a data type and this data type specifies how much memory is to be allocated and what type of values are to be stored in that variable or constant or array. The formal definition of a data type is as follows...

The Data type is a set of value with predefined characteristics. Data types are used to declare variable, constants, arrays, pointers, and functions. In the c programming language, data types are classified as follows...

- Primary data types (Basic data types OR Predefined data types)
- Derived data types (Secondary data types OR User-defined data

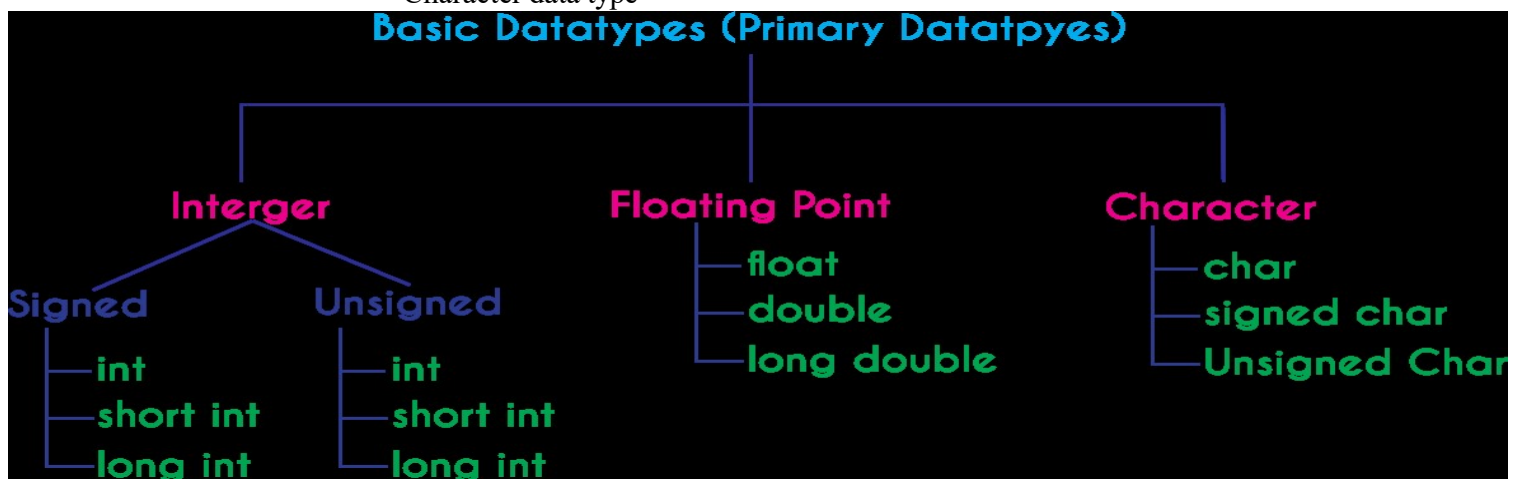
types)

- Enumeration data types
- Void data type



Primary data types The primary data types in the C programming language are the basic data types. All the primary data types are already defined in the system. Primary data types are also called as Built-In data types. The following are the primary data types in c programming language.

- ☐ Integer data type
- ☐ Floating Point data type
- ☐ Double data type
- ☐ Character data type



Integer Data type (int) The integer data type is a set of whole numbers. Every integer value does not have the decimal value. We use the keyword "int" to represent integer data type in c. We use the keyword int to declare the variables and to specify the return type of a function. The integer data type is used with different type modifiers like short, long, signed and unsigned. The following table provides complete details about the integer data type.

Type	Size (bytes)	Range	Specifier
int (signed short int)	2	-32768 to +32767	%d
short int (signed short int)	2	-32768 to +32767	%d
long int (signed long int)	4	-2,147,483,648 to +2,147,483,647	%d
unsigned int (unsigned short int)	2	0 to 65535	%u
unsigned long int	4	0 to 4,294,967,295	%u

Floating

Point data types

Floating-point data types are a set of numbers with the decimal value. Every floatingpoint value must contain the decimal value. The floating-point data type has two variants...

- ☐ float
- ☐ double

We use the keyword "float" to represent floating-point data type and "double" to represent double data type in c. Both float and double are similar but they differ in the number of decimal places. The float value contains 6 decimal places whereas double value contains 15 or 19 decimal places. The following table provides complete details about floating-point data type.

Character data type

The character data type is a set of characters enclosed in single quotations.

Type	Size (bytes)	Range	Specifier
char (signed char)	1	-128 to +127	%c
unsigned char	1	0 to 255	%c

The

following table provides complete details about the character data type. The following table provides complete information about all the data types in c programming language...

	Integer	Floating Point	Double	Character
What is it?	Numbers without decimal value	Numbers with decimal value	Numbers with decimal value	Any symbol enclosed in single quotation
Keyword	int	float	double	char
Memory Size	2 or 4 Bytes	4 Bytes	8 or 10 Bytes	1 Byte
Range	-32768 to +32767 (or) 0 to 65535 (Incase of 2 bytes only)	1.2E - 38 to 3.4E + 38	2.3E-308 to 1.7E+308	-128 to + 127 (or) 0 to 255
Type Specifier	%d or %i or %u	%f	%ld	%c or %s
Type Modifier	short, long signed, unsigned	No modifiers	long	signed, unsigned
Type Qualifier	const, volatile	const, volatile	const, volatil	const, volatile

Void data type

The void data type means nothing or no value. Generally, the void is used to specify a function which does not return any value. We also use the void data type to specify empty parameters of a function.

Type	Size (Bytes)	Range	Specifier
float	4	1.2E - 38 to 3.4E + 38	%f
double	8	2.3E-308 to 1.7E+308	%ld
long double	10	3.4E-4932 to 1.1E+4932	%ld

Enumerated data type

An enumerated data type is a user-defined data type that consists of integer constants and each integer constant is given a name. The keyword "enum" is used to define the enumerated data type.

Derived data types

Derived data types are user-defined data types. The derived data types are also called as user-defined data types or secondary data types. In the c programming language, the derived data types are created using the following concepts...

- ☐ Arrays
- ☐ Structures
- ☐ Unions

6.2 Managing Input and Output operations.

Output Functions

C programming language provides built-in functions to perform output operation. The output operations are used to display data on user screen (output screen) or printer or any file. The c programming language provides the following built-in output functions...

- ☐ printf()
- ☐ putchar()
- ☐ puts()
- ☐ fprintf()

printf() function

The printf() function is used to print string or data values or a combination of string and data values on the output screen (User screen). The printf() function is built-in function defined in a header file called "stdio.h". When we want to use printf() function in our program we need to include the respective header file (stdio.h) using the #include statement. The printf() function has the following syntax... **Syntax:**
printf("message to be display!!!");

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
printf("Hello! Welcome to Diploma smart class!!!");
}
```

In the above example program, we used the printf() function to print a string on to the output screen.

Output of the program is **Hello! Welcome to Diploma smart class!!!**

The printf() function is also used to display data values. When we want to display data values we use format string of the data value to be displayed.

Syntax:

```
printf("format string",variableName);
```

Example Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i = 10;
float x = 5.5;
printf("%d %f",i, x);
}
```

In the above example program, we used the printf() function to print data values of variables i and x on to the output screen. Here i is an integer variable so we have used format string %d and x is a float variable so we have used format string %f.

The printf() function can also be used to display string along with data values. To display the output in different lines or as we wish, we use some special characters called escape sequences. Escape sequences are special characters with special functionality used in printf() function to format the output according to the user requirement. In the C programming language, we have the following escape sequences...

Escape sequence	Meaning
\n	Moves the cursor to New Line
\t	Inserts Horizontal Tab (5 characters space)
\v	Inserts Vertical Tab (5 lines space)
\a	Beep sound
\b	Backspace (removes the previous character from its current position)
\\	Inserts Backward slash symbol

\? Inserts Question mark symbol
\' Inserts Single quotation mark symbol
\\" Inserts Double quotation mark symbol

Consider the following example program...

Example Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
printf("Welcome to\n");
printf("Diploma smart class\n");
printf("the perfect website for learning");
}
```

Output:-

Welcome to
Diploma smart class
the perfect website for learning

putchar() function

The putchar() function is used to display a single character on the output screen. The putchar() functions prints the character which is passed as a parameter to it and returns the same character as a return value. This function is used to print only a single character. To print multiple characters we need to write multiple times or use a looping statement. Consider the following example program...

```
#include<stdio.h>
#include<conio.h>
void main()
{
char ch = 'A';
putchar(ch);
}
```

puts() function

The puts() function is used to display a string on the output screen. The puts() functions prints a string or sequence of characters till the newline. Consider the following example program...

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
char name[30];
printf("\nEnter your favourite website: ");
gets(name);
puts(name);
}
```

fprintf() function

The fprintf() function is used with the concept of files. The fprintf() function is used to print a line into the file. When you want to use fprintf() function the file must be opened in writing mode.

Input Functions

C programming language provides built-in functions to perform input operations. The input operations are used to read user values (input) from the keyboard. The c programming language provides the following built-in input functions.

- ☐ scanf()
- ☐ getchar()
- ☐ getch()
- ☐ gets()
- ☐ fscanf()

scanf() function

The scanf() function is used to read multiple data values of different data types from the keyboard. The scanf() function is built-in function defined in a header file called "stdio.h". When we want to use scanf() function in our program, we need to include the respective header file (stdio.h) using #include statement. The scanf() function has the following syntax...

Syntax:

```
scanf("format strings",&variableNames);
```

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i;
    printf("\nEnter any integer value: ");
    scanf("%d",&i);
    printf("\nYou have entered %d number",i);
}
```

In the above example program, we used the scanf() function to read an integer value from the keyboard and store it into variable 'i'.

Output:-

```
Enter any integer value: 53
You have entered 53 number
```

The scanf() function is also used to read multiple data values of different or the same data types. Consider the following example program...

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i;
    float x;
    printf("\nEnter one integer followed by one float value : ");
    scanf("%d%f",&i, &x);
    printf("\ninteger value = %d, float value = %f",i, x);
}
```

Output:-

Enter one integer followed by one float value : 59 32.8

integer value = 59, float value =32.8

In the above example program, we used the scanf() function to read one integer value and one float value from the keyboard. Here 'i' is an integer variable so we have used format string

%d, and 'x' is a float variable so we have used format string %f.

The scanf() function returns an integer value equal to the total number of input values read using scanf function.

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i,a,b;
    float x;
    printf("\nEnter two integers and one float : ");
    i = scanf("%d%d%f",&a, &b, &x);
    printf("\nTotal inputs read : %d",i);
}
```

getchar() function:

The getchar() function is used to read a character from the keyboard and return it to the program. This function is used to read a single character. To read multiple characters we need to write multiple times or use a looping statement. Consider the following example program...

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char ch;
    printf("\nEnter any character : ");
    ch = getchar();
    printf("\nYou have entered : %c\n",ch);
}
```

Output:-

Enter any character :H

You have entered :H

getch() function

The getch() function is similar to getchar function. The getch() function is used to read a character from the keyboard and return it to the program. This function is used to read a single character. To read multiple characters we need to write multiple times or use a looping statement. Consider the following example program...

Example Program

```
#include<stdio.h>
#include<conio.h>
```

```

void main()
{
char ch;
printf("\nEnter any character : ");
ch = getch();
printf("\nYou have entered : %c",ch);
}

```

gets() function

The gets() function is used to read a line of string and stores it into a character array. The gets() function reads a line of string or sequence of characters till a newline symbol enters. Consider the following example program...

Example Program

```

#include<stdio.h>
#include<conio.h>
void main()
{
char name[30];
printf("\nEnter your favourite website: ");
gets(name); printf("%s",name);
}

```

fscanf() function

The fscanf() function is used with the concept of files. The fscanf() function is used to read data values from a file. When you want to use fscanf() function the file must be opened in reading mode.

6.3 Operators, Expressions, Type conversion & Typecasting

6.3.1 Operators

An operator is a symbol used to perform arithmetic and logical operations in a program. That means an operator is a special symbol that tells the compiler to perform mathematical or logical operations. C programming language supports a rich set of operators that are classified as follows.

- ☐ Arithmetic Operators
- ☐ Relational Operators
- ☐ Logical Operators
- ☐ Increment & Decrement Operators
- ☐ Assignment Operators
- ☐ Bitwise Operators
- ☐ Conditional Operator
- ☐ Special Operators

Arithmetic Operators (+, -, *, /, %)

The arithmetic operators are the symbols that are used to perform basic mathematical operations like addition, subtraction, multiplication, division and percentage modulo. The following table provides information about arithmetic operators.

Operator	Meaning	Example
+	Addition	10 + 5 = 15
-	Subtraction	10 - 5 = 5
*	Multiplication	10 * 5 = 50
/	Division	10 / 5 = 2

% Remainder of the Division 5 % 2 = 1

- The addition operator can be used with numerical data types and character data type. When it is used with numerical values, it performs mathematical addition and when it is used with character data type values, it performs concatenation (appending).
- The remainder of the division operator is used with integer data type only.

Relational Operators (<, >, <=, >=, ==, !=)

The relational operators are the symbols that are used to compare two values. That means the relational operators are used to check the relationship between two values. Every relational operator has two results TRUE or FALSE. In simple words, the relational operators are used to define conditions in a program. The following table provides information about relational operators.

Operator	Meaning	Example
<	Returns TRUE if the first value is smaller than second value otherwise returns FALSE	10 < 5 is FALSE
>	Returns TRUE if the first value is larger than second value otherwise returns FALSE	10 > 5 is TRUE
<=	Returns TRUE if the first value is smaller than or equal to second value otherwise returns	
FALSE		10 <= 5 is FALSE
>=	Returns TRUE if the first value is larger than or equal to second value otherwise returns	
FALSE		10 >= 5 is TRUE
==	Returns TRUE if both values are equal otherwise returns FALSE	
		10 == 5 is FALSE
!=	Returns TRUE if both values are not equal otherwise returns FALSE	
		10 != 5 is TRUE

Logical Operators (&&, ||, !)

The logical operators are the symbols that are used to combine multiple conditions into one condition. The following table provides information about logical operators.

Operator	Meaning	Example
&&	Logical AND - Returns TRUE if all conditions are TRUE otherwise returns FALSE	10 < 5 && 12 > 10 is FALSE
	Logical OR - Returns FALSE if all conditions are FALSE otherwise returns TRUE	10 < 5 12 > 10 is TRUE
!	Logical NOT - Returns TRUE if condition is FLASE and returns FALSE if it is TRUE	!(10 < 5 && 12 > 10) is TRUE

- Logical AND - Returns TRUE only if all conditions are TRUE, if any of the conditions is FALSE then complete condition becomes FALSE.
- Logical OR - Returns FALSE only if all conditions are FALSE, if any of the conditions is TRUE then

complete condition becomes TRUE.

Increment & Decrement Operators (++ & --)

The increment and decrement operators are called unary operators because both need only one operand. The increment operators add one to the existing value of the operand and the decrement operator subtracts one from the existing value of the operand. The following table provides information about increment and decrement operators. The increment and decrement operators are used in front of the operand (++a) or after the operand (a++). If it is used in front of the operand, we call it as pre-increment or predecrement and if it is used after the operand, we call it as post-increment or post-decrement.

Pre-Increment or Pre-Decrement

In the case of pre-increment, the value of the variable is increased by one before the expression evaluation. In the case of pre-decrement, the value of the variable is decreased by one before the expression evaluation. That means, when we use pre-increment or predecrement, first the value of the variable is incremented or decremented by one, then the modified value is used in the expression evaluation.

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i = 5,j;
    j = ++i; // Pre-Increment
    printf("i = %d, j = %d",i,j);
}
```

Post-Increment or Post-Decrement

In the case of post-increment, the value of the variable is increased by one after the expression evaluation. In the case of post-decrement, the value of the variable is decreased by one after the expression evaluation. That means, when we use post-increment or postdecrement, first the expression is evaluated with existing value, then the value of the variable is incremented or decremented by one.

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i = 5,j;
    j = i++; // Post-Increment
    printf("i = %d, j = %d",i,j);
}
```

Increment/Decrement Operators		Let us assume X is a variable
Operator	Expression	Description
++	++X	Pre-increment
	X++	Post-increment
--	--X	Pre-decrement
	X--	Post-decrement

Assignment Operators (=, +=, -=, *=, /=, %=)

The assignment operators are used to assign right-hand side value (Rvalue) to the lefthand side variable (Lvalue). The assignment operator is used in different variants along with arithmetic operators. The following table describes all the assignment operators in the C programming language.

Operator	Example	Equivalent Arithmetic Operators	Resulting x
=	x=5	x=5	5
+=	x+=5	x=x+5	15
-=	x-=5	x=x-5	5
=	x=5	x=x*5	50
/=	x/=5	x=x/5	2
%=	x%=5	x=x%5	0

Bitwise

Operators (&, |, ^, ~, >>, <<)

The bitwise operators are used to perform bit-level operations in the c programming language. When we use the bitwise operators, the operations are performed based on the binary values. The following table describes all the bitwise operators in the C programming language. Let us consider two variables A and B as A = 25 (11001) and B = 20 (10100).

Operator	Example	Explanation
<< left shift	X = X<<2;	Before 0000 1111 X is 15 (8+4+2+1) After 0011 1100 X is 60 (32+16+8+4)
>> right shift	X = X>>2;	Before 0000 1111 X is 15 (8+4+2+1) After 0000 0011 X is 3 (2+1)
& bit-wise AND	X = X&28;	0000 1111 & 0001 1010 = 0000 1010 15 28 10
bit-wise OR	X = X 28;	0000 1111 0001 1010 = 0001 1111 15 28 31
^ bit-wise XOR	X = X^28;	0000 1111 ^ 0001 1010 = 0001 0101 15 28 21
~ bit inversion	X = ~X;	Before 0000 1111 X is 15 (8+4+2+1) After 1111 0000 X is -2,147,483,633

Conditional Operator (?:)

The conditional operator is also called a ternary operator because it requires three operands. This operator is used for decision making. In this operator, first we verify a condition, then we perform one operation out of the two operations based on the condition result. If the condition is TRUE the first option is performed, if the condition is FALSE the second option is performed. The conditional operator is used with the following syntax.

Condition ? TRUE Part : FALSE Part;

Example

A = (10<15)?100:200; ⇒ A value is 100

Special Operators (sizeof, pointer, comma, dot, etc.)

The following are the special operators in c programming language.

sizeof operator

This operator is used to find the size of the memory (in bytes) allocated for a variable. This operator is used with the following syntax.

sizeof(variable Name);

Example

sizeof(A); ⇒ the result is 2 if A is an integer

Pointer operator (*)

This operator is used to define pointer variables in c programming language.

Comma operator (,)

This operator is used to separate variables while they are declaring, separate the expressions in function calls, etc.

Dot operator (.)

This operator is used to access members of structure or union.

Operator Precedence and Associativity

What is Operator Precedence?

Operator precedence is used to determine the order of operators evaluated in an expression. In c programming language every operator has precedence (priority). When there is more than one operator in an expression the operator with higher precedence is evaluated first and the operator with the least precedence is evaluated last.

What is Operator Associativity?

Operator associativity is used to determine the order of operators with equal precedence evaluated in an expression. In the c programming language, when an expression contains multiple operators with equal precedence, we use associativity to determine the order of evaluation of those operators. In c programming language the operator precedence and associativity are as shown in the following table

<i>Operator</i>	<i>Description</i>	<i>Associativity</i>
() [] . -> ++ --	Parentheses or function call Brackets or array subscript Dot or Member selection operator Arrow operator Postfix increment/decrement	left to right
++ -- + - ! ~ (type) * & sizeof	Prefix increment/decrement Unary plus and minus not operator and bitwise complement type cast Indirection or dereference operator Address of operator Determine size in bytes	right to left
* / %	Multiplication, division and modulus	left to right
+ -	Addition and subtraction	left to right
<< >>	Bitwise left shift and right shift	left to right
< <= > >=	relational less than/less than equal to relational greater than/greater than or equal to	left to right
== !=	Relational equal to and not equal to	left to right
&	Bitwise AND	left to right
^	Bitwise exclusive OR	left to right
 	Bitwise inclusive OR	left to right
&&	Logical AND	left to right
 	Logical OR	left to right
? :	Ternary operator	right to left
= += -= *= /= %= &= ^= = <<= >>=	Assignment operator Addition/subtraction assignment Multiplication/division assignment Modulus and bitwise assignment Bitwise exclusive/inclusive OR assignment	right to left
,	Comma operator	left to right

6.3.2

Expressions

What is an expression?

In any programming language, if we want to perform any calculation or to frame any condition etc., we use a set of symbols to perform the task. These set of symbols makes an expression. In the C programming

language, an expression is defined as follows.

An expression is a collection of operators and operands that represents a specific value. In the above definition, an operator is a symbol that performs tasks like arithmetic operations, logical operations, and conditional operations, etc. Operands are the values on which the operators perform the task. Here operand can be a direct value or variable or address of memory location.

Expression Types in C In the C programming language, expressions are divided into THREE types. They are as follows...

- ☐ Infix Expression
- ☐ Postfix Expression
- ☐ Prefix Expression The above classification is based on the operator position in the expression.

Infix Expression

- ☐ The expression in which the operator is used between operands is called infix expression.
- ☐ The infix expression has the following general structure.

Operand1 Operator Operand2

Postfix Expression

- ☐ The expression in which the operator is used after operands is called postfix expression.
- ☐ The postfix expression has the following general structure.

Operand1 Operand2 Operator

Prefix Expression

- ☐ The expression in which the operator is used before operands is called a prefix expression.
- ☐ The prefix expression has the following general structure.

Operator Operand1 Operand2

6.3.3 Type Conversion and Type Casting In a programming language, the expression contains data values of the same datatype or different data types. When the expression contains similar datatype values then it is evaluated without any problem. But if the expression contains two or more different datatype values then they must be converted to the single datatype of destination datatype. Here, the destination is the location where the final result of that expression is stored. For example, the multiplication of an integer data value with the float data value and storing the result into a float variable. In this case, the integer value must be converted to float value so that the final result is a float datatype value. In a c programming language, the data conversion is performed in two different methods as follows...

- ☐ Type Conversion
- ☐ Type Casting

Type Conversion The type conversion is the process of converting a data value from one data type to another data type automatically by the compiler. Sometimes type conversion is also called implicit type conversion. The implicit type conversion is automatically performed by the compiler. For example, in c programming language, when we assign an integer value to a float variable the integer value automatically gets converted to float value by adding decimal value 0. And when a float value is assigned to an integer variable the float value automatically gets converted to an integer value by removing the decimal value. To understand more about type conversion observe the following...

```
int i = 10 ;  
float x = 15.5 ;  
char ch = 'A' ;  
i = x ; =====> x value 15.5 is converted as 15 and assigned to variable i  
x = i ; =====> Here i value 10 is converted as 10.000000 and assigned to variable x  
i = ch ; =====> Here the ASCII value of A (65) is assigned to i
```

Example Program

```
#include<stdio.h>  
#include<conio.h>
```

```

void main()
{
int i = 95 ;
float x = 90.99 ;
char ch = 'A' ;
i = x ;
printf("i value is %d\n",i);
x = i ;
printf("x value is %f\n",x);
i = ch ;
printf("i value is %d\n",i);
}

```

In the above program, we assign $i = x$, i.e., float variable value is assigned to the integer variable. Here, the compiler automatically converts the float value (90.99) into integer value (90) by removing the decimal part of the float value (90.99) and then it is assigned to variable i . Similarly, when we assign $x = i$, the integer value (90) gets converted to float value (90.000000) by adding zero as the decimal part.

Typecasting

Typecasting is also called an explicit type conversion. Compiler converts data from one data type to another data type implicitly. When compiler converts implicitly, there may be a data loss. In such a case, we convert the data from one data type to another data type using explicit type conversion. To perform this we use the unary cast operator. To convert data from one type to another type we specify the target data type in parenthesis as a prefix to the data value that has to be converted.

The general syntax of typecasting is as follows.

(TargetDatatype) DataValue

Example

```

int totalMarks = 450, maxMarks = 600 ;
float average ;
average = (float) totalMarks / maxMarks * 100 ;

```

In the above example code, both totalMarks and maxMarks are integer data values. When we perform totalMarks / maxMarks the result is a float value, but the destination (average) datatype is a float. So we use type casting to convert totalMarks and maxMarks into float data type.

Example Program

```

#include<stdio.h>
#include<conio.h>
int main()
{
int a, b, c ;
float avg ;
printf( "Enter any three integer values : ");
scanf(—%d%d%dll,a,b, c);
avg = (a + b + c) / 3 ;
printf("avg before casting = &f", avg \n);
avg = (float)(a + b + c) / 3 ;
printf("avg after casting = %f",avg\n);
return 0;
}

```

Comments

Comments in C are enclosed by slash/star pairs: `/* .. comments .. */` which may cross multiple lines. C++ introduced a form of comment started by two slashes and extending to the end of the line:

`// comment until the line end`

The `//` comment form is so handy that many C compilers now also support it, although it is not technically part of the C language. Along with well-chosen function names, comments are an important part of well written code. Comments should not just repeat what the code says. Comments should describe what the code accomplishes which much more is interesting than a translation of what each statement does. Comments should also narrate what is tricky or non-obvious about a section of code.

6.4 Decision Control and Looping Statements (If, If-else, If-else-if, Switch, While, Do- while, For, Break, Continue & Goto)

Control Structures

C uses curly braces (`{}`) to group multiple statements together. The statements execute in order. Some languages let you declare variables on any line (C++). Other languages insist that variables are declared only at the beginning of functions (Pascal). C takes the middle road -- variables may be declared within the body of a function, but they must follow a `{`. More modern languages like Java and C++ allow you to declare variables on any line, which is handy

. What is Decision Making Statement?

In the C programming language, the program execution flow is line by line from top to bottom. That means the c program is executed line by line from the main method. But this type of execution flow may not be suitable for all the program solutions. Sometimes, we make some decisions or we may skip the execution of one or more lines of code. Consider a situation, where we write a program to check whether a student has passed or failed in a particular subject. Here, we need to check whether the marks are greater than the pass marks or not. If marks are greater, then we decide that the student has passed otherwise failed. To solve such kind of problems in c we use the statements called decision making statements.

Decision-making statements are the statements that are used to verify a given condition and decide whether a block of statements gets executed or not based on the condition result.

In the c programming language, there are two decision-making statements they are as follows.

1. if statement
2. switch statement

if statement in c

In c, if statement is used to make decisions based on a condition. The if statement verifies the given condition and decides whether a block of statements are executed or not based on the condition result. In c, if statement is classified into four types as follows...

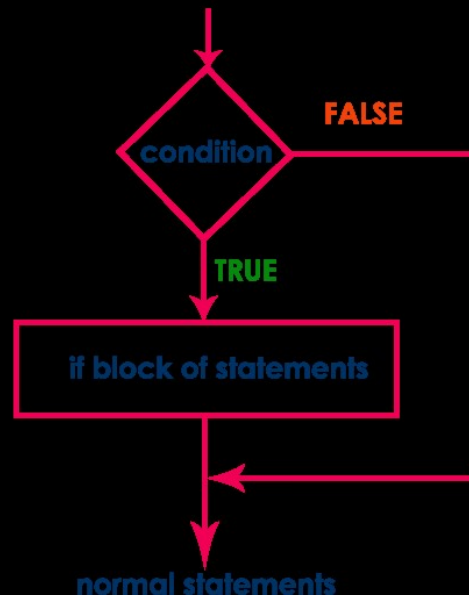
1. Simple if statement
2. if-else statement
3. Nested if statement
4. if-else-if statement (if-else ladder)

Simple if statement

Syntax

```
if ( condition )  
{  
    ....  
    block of statements;  
    ....  
}
```

Execution flow diagram



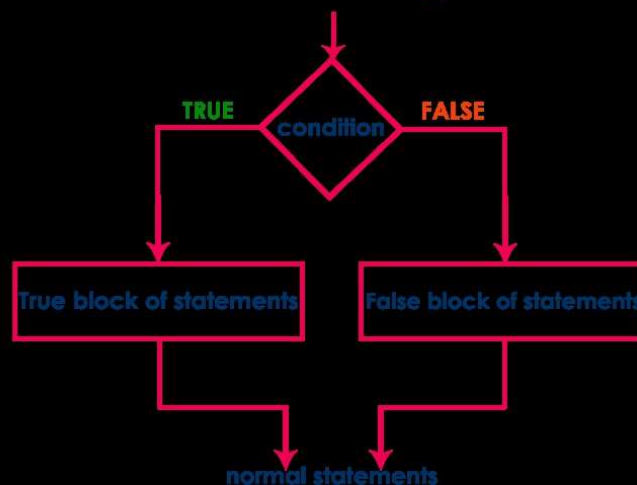
Simple if statement is used to verify the given condition and executes the block of statements based on the condition result. The simple if statement evaluates specified condition. If it is TRUE, it executes the next statement or block of statements. If the condition is FALSE, it skips the execution of the next statement or block of statements. The general syntax and execution flow of the simple if statement is as follows. Simple if statement is used when we have only one option that is executed or skipped based on a condition.

if-else statement

Syntax

```
if ( condition )  
{  
    ....  
    True block of statements;  
    ....  
}  
else  
{  
    ....  
    False block of statements;  
    ....  
}
```

Execution flow diagram



The if-else statement is used to verify the given condition and executes only one out of the two blocks of statements based on the condition result. The if-else statement evaluates the specified condition. If it is TRUE, it executes a block of statements (True block). If the condition is FALSE, it executes another block of statements (False block). The general syntax and execution flow of the if-else statement is as follows. The if-else statement is used when we have two options and only one option has to be executed based on a condition result (TRUE or FALSE).

Syntax

```
if ( condition1 )  
{  
    if ( condition2 )  
    {  
        ....  
        True block of statements 1;  
    }  
    ....  
}  
else  
{  
    False block of condition1;  
}
```

Nested if statement

Writing a if statement inside another if statement is called nested if statement. The general syntax of the nested if statement is as follows. The nested if statement can be defined using any combination of simple if & if-else statements.

Syntax

```
if ( condition1 )
{
    ....
    True block of statements1;
    ....
}
else if ( condition2 )
{
    False block of condition1;
    &
    True block of condition2
}
```

if-else-if statement (if-else ladder)

Writing a if statement inside else of an if statement is called if-else-if statement. The general syntax of the if-else-if statement is as follows... The if-else-if statement can be defined using any combination of simple if & if-else statements.

Examples:

Program 1:(Simple if statement)

```
#include <stdio.h>
int main()
{
int x = 20;
int y = 22;
if (x<y)
{
printf("Variable x is less than y");
} return 0;
}
```

Output:

Variable x is less than y

Program 2:(Simple if statement)

```
#include <stdio.h>
int main()
{
```

```

int i = 10;
if (i > 15)
{
printf("10 is less than 15");
} printf("I am Not in if");
}

```

Output:

I am Not in if

Program 3:(if-else statement)

```

#include <stdio.h>
int main()
{
int i = 20;
if (i < 15)
printf("i is smaller than 15");
else
printf("i is greater than 15");
return 0;
}

```

Output:

i is greater than 15

Program 4:(if-else statement)

```

#include <stdio.h>
int main()
{
int age;
printf("Enter your age:");
scanf("%d",&age);
if(age >=18)
{
/* This statement will only execute if the * above condition (age>=18) returns true */
printf("You are eligible for voting");
}
else
{
/* This statement will only execute if the * condition specified in the "if" returns false. */
printf("You are not eligible for voting");
}
return 0;
}

```

Output:

Enter your age:14

You are not eligible for voting

Program 5:(Nested-if statement)

```

#include <stdio.h>
int main()

```

```

{
int var1, var2;
printf("Input the value of var1:");
scanf("%d", &var1);
printf("Input the value of var2:");
scanf("%d",&var2);
if (var1 != var2)
{
printf("var1 is not equal to var2\n");
//Nested if else if (var1 > var2)
{
printf("var1 is greater than var2\n");
}
}
Else
{
printf("var2 is greater than var1\n");
}
}

```

Output:

```

Input the value of var1:12
Input the value of var2:21
var1 is not equal to var2
var2 is greater than var1

```

Program 6:(Nested-if statement)

```

#include <stdio.h>
int main()
{
int i = 10;
if (i == 10)
{
// First if statement
if (i < 15)
printf("i is smaller than 15\n");
// Nested - if statement will only be executed if statement above is true
if (i < 12)
printf("i is smaller than 12 too\n");
else printf("i is greater than 15");
}
return 0;
}

```

Output:

```

i is smaller than 15
i is smaller than 12 too

```

Program 7:(if-else-if statement)

```

#include <stdio.h>
void main()

```

```

{
int i = 20;
if (i == 10)
printf("i is 10");
else if (i == 15)
printf("i is 15");
else if (i == 20) printf("i is 20");
else
printf("i is not present");
}

```

Output:

i is 20

Program 8:(if-else-if statement)

```

#include <stdio.h>
int main()
{
int var1, var2;
printf("Input the value of var1:");
scanf("%d", &var1);
printf("Input the value of var2:");
scanf("%d",&var2);
if (var1 !=var2)
{
printf("var1 is not equal to var2\n");
}
Else
if (var1 > var2)
{
printf("var1 is greater than var2\n");
}
else
if (var2 > var1)
{
printf("var2 is greater than var1\n");
}
Else
{
printf("var1 is equal to var2\n");
}
return 0;
}

```

Output:

Input the value of var1:12

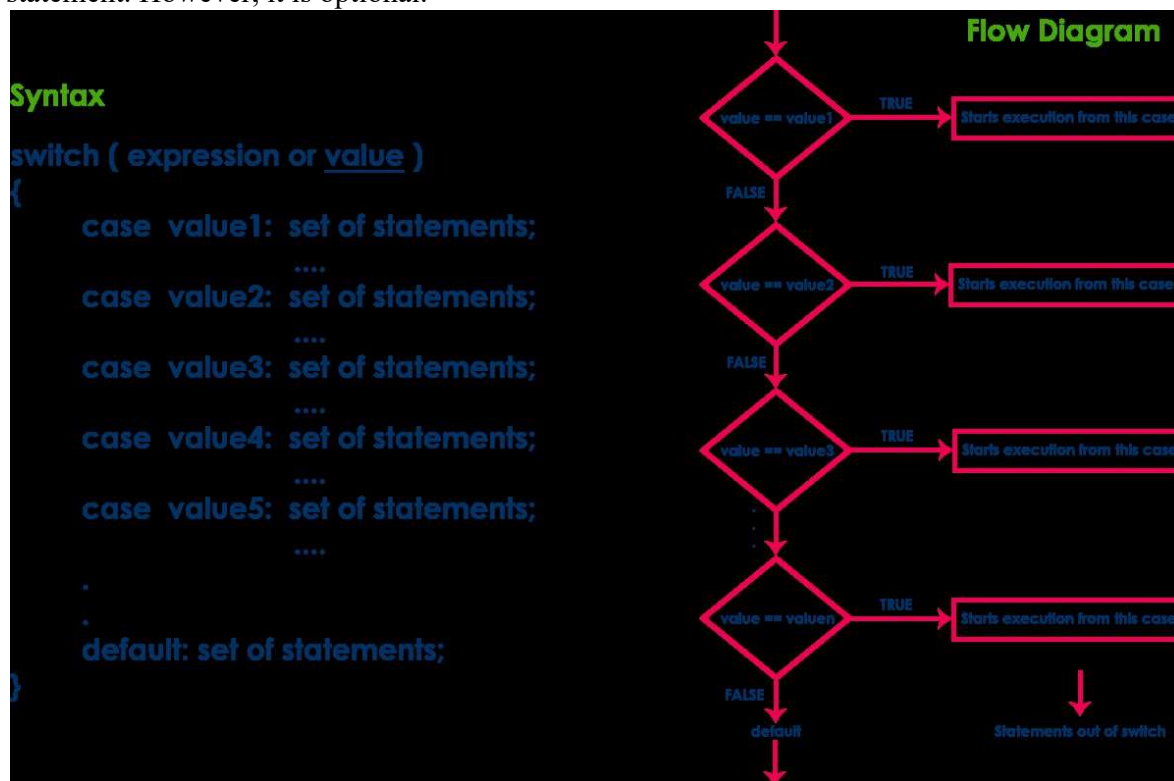
Input the value of var2:21

var1 is not equal to var2

'switch' statement in C

Consider a situation in which we have many options out of which we need to select only one option that is

to be executed. Such kind of problems can be solved using nested if statement. But as the number of options increases, the complexity of the program also gets increased. This type of problem can be solved very easily using a switch statement. Using the switch statement, one can select only one option from more number of options very easily. In the switch statement, we provide a value that is to be compared with a value associated with each option. Whenever the given value matches the value associated with an option, the execution starts from that option. In the switch statement, every option is defined as a case. The switch statement has the following syntax and execution flow diagram. The switch statement contains one or more cases and each case has a value associated with it. At first switch statement compares the first case value with the switchValue, if it gets matched the execution starts from the first case. If it doesn't match the switch statement compares the second case value with the switch Value and if it is matched the execution starts from the second case. This process continues until it finds a match. If no case value matches with the switchValue specified in the switch statement, then a special case called default is executed. When a case value matches with the switch Value, the execution starts from that particular case. This execution flow continues with the next case statements also. To avoid this, we use the "break" statement at the end of each case. That means the break statement is used to terminate the switch statement. However, it is optional.



Examples:

Program 1:

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int n ;
    clrscr() ;
    printf("Enter any digit: ") ;
    scanf("%d", &n) ;
    switch( n )
    {
        case 0: printf("ZERO") ;

```

```

break ;
case 1:printf("ONE");
break ;
case 2:printf("TWO");
break ;
case 3: printf("THREE");
break ;
case 4: printf("FOUR");
break ;
case 5: printf("FIVE");
break ;
case 6: printf("SIX");
break ;
case 7: printf("SEVEN");
break ;
case 8: printf("EIGHT");
break ;
case 9: printf("NINE");
break ;
default: printf("Not a Digit") ;
}
getch() ;
}

```

Output:

Enter any digit: 5
FIVE

Program 2:

```

#include <stdio.h>
int main()
{
int num=2;
switch(num+2)
{
case 1:printf("Case1: Value is: %d", num);
case 2:printf("Case2: Value is: %d", num);
case 3:printf("Case3: Value is: %d", num);
default:printf("Default: Value is: %d", num);
}
return 0;
}

```

Output: Default: value is: 2

Looping statements

Consider a situation in which we execute a single statement or block of statements repeatedly for the required number of times. Such kind of problems can be solved using looping statements in C. For example, assume a situation where we print a message 100 times. If we want to perform that task without using looping statements, we have to either write 100 printf statements or we have to write the same

message 100 times in a single printf statement. Both are complex methods. The same task can be performed very easily using looping statements.

The looping statements are used to execute a single statement or block of statements repeatedly until the given condition is FALSE.

C language provides three looping statements...

- ☐ while statement
- ☐ do-while statement
- ☐ for statement

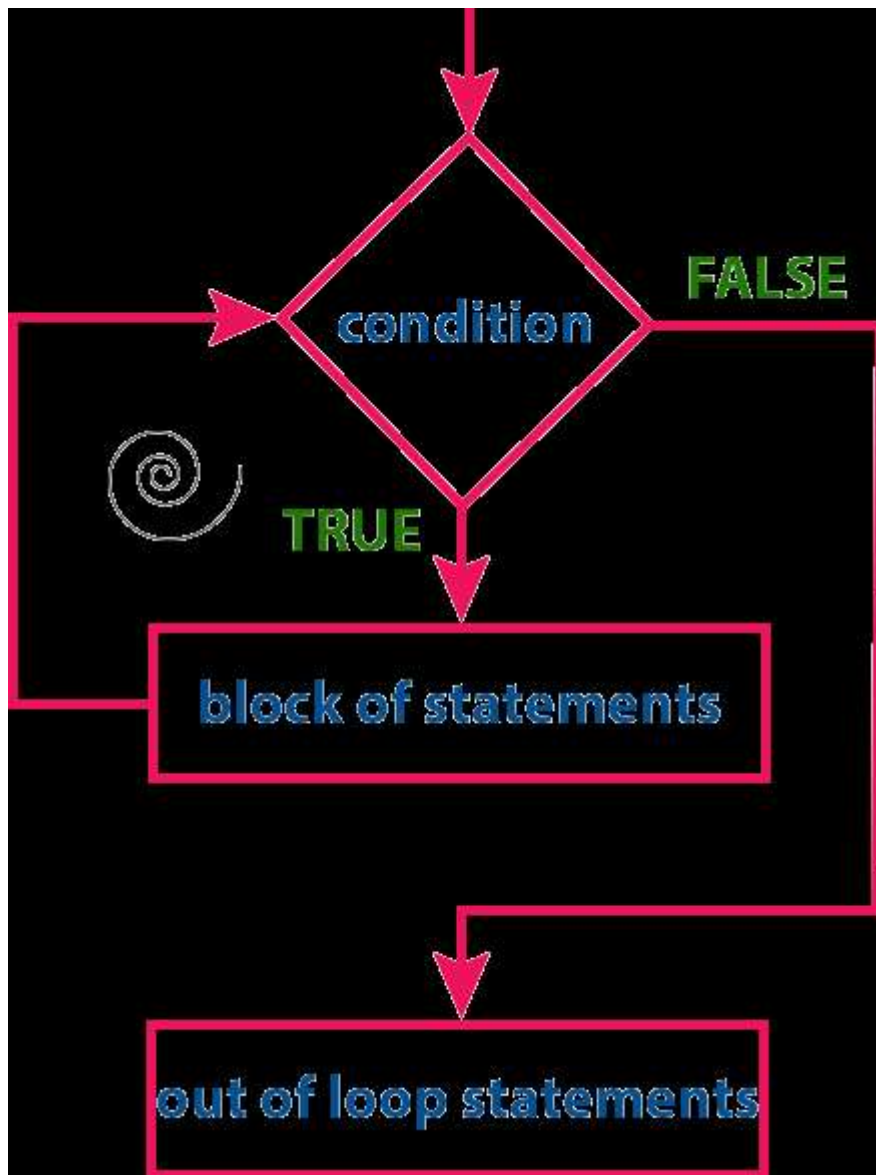
while Statement

The while statement is used to execute a single statement or block of statements repeatedly as long as the given condition is TRUE. The while statement is also known as Entry control looping statement. The while statement has the following syntax... The while statement has the following execution flow diagram... At first, the given condition is evaluated. If the condition is TRUE, the single statement or block of statements gets executed. Once the execution gets completed the condition is evaluated again. If it is TRUE, again the same statements get executed. The same process is repeated until the condition is evaluated to FALSE.

Whenever the condition is evaluated to FALSE, the execution control moves out of the while block.

Syntax:

```
while( condition )  
{  
    ...  
    block of statements;  
    ...  
}
```



Examples:

Program 1:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n = 0;
    clrscr() ;
    printf("Even numbers upto 10 are");
    while( n<= 10 )
    {
        if( n%2 == 0)
        printf("%d\t", n) ;
        n++ ;
    }
    getch() ;
}
```


Output:

Even numbers upto 10 are 0 2 4 6 8 10

Program 2:

```
#include <stdio.h>
int main()
{
    int count=1;
    while (count <= 4)
    {
        printf("%d ", count); count++;
    }
    return 0;
}
```

Output:

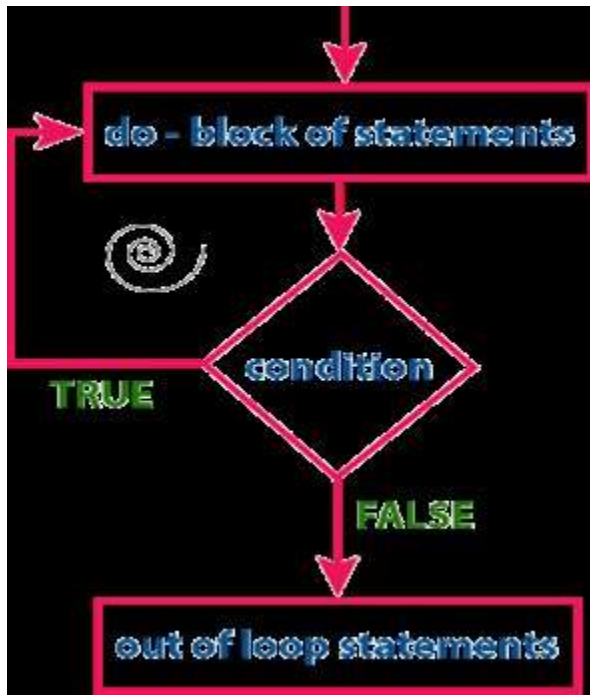
1 2 3 4

'do-while' statement

The do-while statement is used to execute a single statement or block of statements repeatedly as long as given the condition is TRUE. The do-while statement is also known as the Exit control looping statement. The do-while statement has the following syntax... The do-while statement has the following execution flow diagram... At first, the single statement or block of statements which are defined in do block are executed. After the execution of the do block, the given condition gets evaluated. If the condition is evaluated to TRUE, the single statement or block of statements of do block are executed again. Once the execution gets completed again the condition is evaluated. If it is TRUE, again the same statements are executed. The same process is repeated until the condition is evaluated to FALSE. Whenever the condition is evaluated to FALSE, the execution control moves out of the while block.



```
Syntax:
do
{
    ...
    block of statements;
    ...
} while( condition ) ;
```



Examples:

Program 1:

```
#include <stdio.h>
int main()
{
    int j=0;
do
{
    printf("Value of variable j is: %d\n", j);
    j++;
}
while (j<=3);
return 0;
}
```

Output:

Value of variable j is: 0
Value of variable j is: 1
Value of variable j is: 2
Value of variable j is: 3

Program 2:

```
#include <stdio.h>
int main()
{
    int i=0;
do
{
    printf("while vs do-while\n");
}
while(i==1);
```

```
printf("Out of loop");  
}
```

Output:

while vs do-while

Out of loop

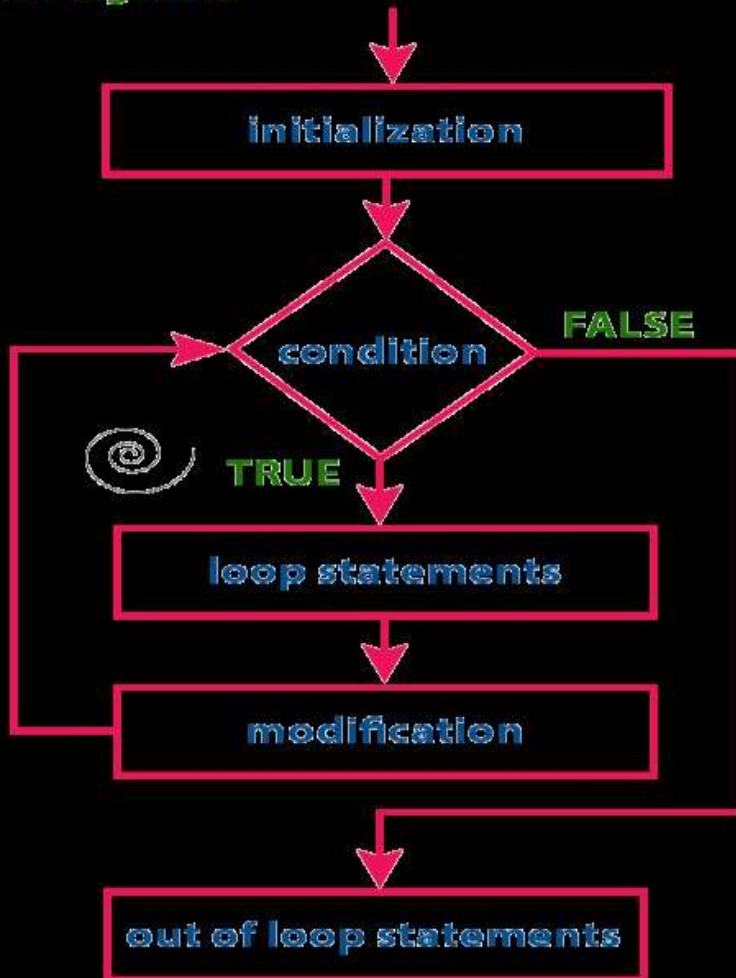
'for' statement

The for statement is used to execute a single statement or a block of statements repeatedly as long as the given condition is TRUE. The for statement has the following syntax and execution flow diagram... At first, the for statement executes initialization followed by condition evaluation. If the condition is evaluated to TRUE, the single statement or block of statements of for statement are executed. Once the execution gets completed, the modification statement is executed and again the condition is evaluated. If it is TRUE, again the same statements are executed. The same process is repeated until the condition is evaluated to FALSE. Whenever the condition is evaluated to FALSE, the execution control moves out of the for block.

Syntax:

```
for( initialization ; condition ; modification )  
{  
    ...  
    block of statements;  
    ...  
}
```

Execution flow diagram:



Examples:

Program 1: // Print the natural numbers from 1 to 10

```
#include <stdio.h>  
int main()  
{  
    int i;  
    for (i = 1; i < 11; ++i)  
    {  
        printf("%d ", i);  
    }  
    return 0;  
}
```

Output:

1 2 3 4 5 6 7 8 9 10

Program 2:

```
#include <stdio.h>
int main()
{
    for (int i=0; i<2; i++)
    {
        for (int j=0; j<4; j++)
        {
            printf("%d, %d\n", i, j);
        }
    }
    return 0;
}
```

Output:

0, 0 0, 1 0, 2 0, 3 1, 0 1, 1 1, 2 1, 3

break, continue and goto in C

In c, there are control statements that do not need any condition to control the program execution flow. These control statements are called as unconditional control statements. C programming language provides the following unconditional control statements...

- ☐ break
- ☐ continue
- ☐ goto

The above three statements do not need any condition to control the program execution flow.

“break” statement

In C, the break statement is used to perform the following two things...

- ☐ break statement is used to terminate the switch case statement
- ☐ break statement is also used to terminate looping statements like while, do-while and for.

When a break statement is encountered inside the switch case statement, the execution control moves out of the switch statement directly. For example, consider the following program.

Examples:**Program 1:**

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
    int i;
    for(i = 0; i<10; i++)
    {
        printf("%d ",i);
        if(i == 5) break;
    }
    printf("\ncame outside of loop i = %d",i);
}
```

```
}
```

Output:

0 1 2 3 4 5

came outside of loop i = 5

Program 2:

```
#include <stdio.h>
int main ()
{
    /* local variable definition */
    int a = 10;
    /* while loop execution */
    while( a< 20 )
    {
        printf("value of a: %d\n", a);
        a++;
        if( a> 15)
        {
            break;
        }
        /* terminate the loop using break statement */
    }
    return 0;
}
```

Output:

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

continue statement

The continue statement is used to move the program execution control to the beginning of the looping statement. When the continue statement is encountered in a looping statement, the execution control skips the rest of the statements in the looping block and directly jumps to the beginning of the loop. The continue statement can be used with looping statements like while, do-while and for. When we use continue statement with while and do-while statements the execution control directly jumps to the condition. When we use continue statement with for statement the execution control directly jumps to the modification portion (increment/decrement/any modification) of the for loop.

Examples:**Program 1:**

```
#include<stdio.h>
void main ()
{
    int i = 0;
    while(i!=10)
    {
```

```
printf("%d", i); continue; i++;  
}  
}
```

Output:

Infinite loop

Program 2:

```
#include<stdio.h>  
int main()  
{  
    int i=1;//initializing a local variable  
    //starting a loop from 1 to 10  
    for(i=1;i<=10;i++)  
    {  
        if(i==5)  
        {  
            //if value of i is equal to 5, it will continue the loop  
            continue;  
        }  
        printf("%d \n",i);  
    }  
    //end of for loop  
    return 0;  
}
```

Output:

1
2
3
4
6
7
8
9
10

goto statement

The goto statement is used to jump from one line to another line in the program. Using goto statement we can jump from top to bottom or bottom to top. To jump from one line to another line, the goto statement requires a label. Label is a name given to the instruction or line in the program. When we use a goto statement in the program, the execution control directly jumps to the line with the specified label.

Examples:**Program 1:**

```
#include <stdio.h>  
void main()  
{  
    int num,i=1;  
    printf("Enter the number whose table you want to print:");
```

```
scanf("%d",&num);
table: printf("%d x %d = %d\n",num,i,num*i);
i++;
if(i<=10) goto table;
}
```

Output:

Enter the number whose table you want to print: 10

```
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

Program 2:

```
#include <stdio.h>
void main()
{
    int i, j, k;
    for(i=0;i<10;i++)
    {
        for(j=0;j<5;j++)
        {
            for(k=0;k<3;k++)
            {
                printf("%d %d %d\n",i,j,k);
                if(j == 3)
                {
                    goto out;
                }
            }
        }
    }
    out:
    printf("came out of the loop");
}
```

Output:

```
0 0 0
0 0 1
0 0 2
0 1 0
0 1 1
0 1 2
0 2 0
0 2 1
0 2 2
```


0 3 0

came out of the loop

6.5 Programming Assignments using the above features.

Some more Examples:

Program to Check Even or Odd

```
#include <stdio.h>
int main()
{
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0) // True if num is perfectly divisible by 2
        printf("%d is even.", num);
    else printf("%d is odd.", num);
    return 0;
}
```

Output

Enter an integer: 7
7 is odd.

Program to Check Vowel or consonant

```
#include <stdio.h>
int main()
{
    char c;
    int lowercase_vowel, uppercase_vowel;
    printf("Enter an alphabet: ");
    scanf("%c", &c);
    // evaluates to 1 if variable c is a lowercase vowel lowercase_vowel = (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');
    // evaluates to 1 if variable c is a uppercase vowel uppercase_vowel = (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');
    // evaluates to 1 (true) if c is a vowel if (lowercase_vowel || uppercase_vowel)
    printf("%c is a vowel.", c);
    else
    printf("%c is a consonant.", c);
    return 0;
}
```

Output

Enter an alphabet: G
G is a consonant.

Program to Check Leap Year

```
#include <stdio.h>
int main()
```

```

{
int year;
printf("Enter a year: ");
scanf("%d", &year);
// leap year if perfectly visible by 400
if (year % 400 == 0)
{
printf("%d is a leap year.", year);
}
// not a leap year if visible by 100
// but not divisible by 400
else if (year % 100 == 0)
{
printf("%d is not a leap year.", year);
}
// leap year if not divisible by 100
// but divisible by 4
else if (year % 4 == 0)
{
printf("%d is a leap year.", year);
}
// all other years are not leap year else
{
printf("%d is not a leap year.", year);
}
return 0;
}

```

Output

Enter a year: 1900
1900 is not a leap year

Program to Check Alphabet

```

#include <stdio.h>
int main()
{
char c;
printf("Enter a character: ");
scanf("%c", &c);
if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'))
printf("%c is an alphabet.", c);
else
printf("%c is not an alphabet.", c);
return 0;
}

```

Output

Enter a character: *
* is not an alphabet

To find the Factorial of a Number

```
#include <stdio.h>
int main()
{
    int n, i;
    unsigned long long fact = 1;
    printf("Enter an integer: ");
    scanf("%d", &n);
    // shows error if the user enters a negative integer
    if (n < 0)
        printf("Error! Factorial of a negative number doesn't exist.");
    else
    {
        for (i = 1; i <= n; ++i)
        {
            fact *= i;
        }
        printf("Factorial of %d = %llu", n, fact);
    }
    return 0;
}
```

Output

Enter an integer: 5
Factorial of 5 = 120

Multiplication Table Up to 10

```
#include <stdio.h>
int main()
{
    int n, i;
    printf("Enter an integer: ");
    scanf("%d", &n);
    for (i = 1; i <= 10; ++i)
    {
        printf("%d * %d = %d \n", n, i, n * i);
    }
    return 0;
}
```

Output

Enter an integer: 9
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72

9 * 9 = 81
9 * 10 = 90

To Reverse an Integer number

```
#include <stdio.h>

int main()
{
    int n, rev = 0, remainder;
    printf("Enter an integer: ");
    scanf("%d", &n);
    while (n != 0)
    {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n /= 10;
    }
    printf("Reversed number = %d", rev);
    return 0;
}
```

Output

Enter an integer: 2345
Reversed number = 5432

Program to Check Palindrome

```
#include <stdio.h>

int main()
{
    int n, reversedN = 0, remainder, originalN;
    printf("Enter an integer: ");
    scanf("%d", &n);
    originalN = n;
    // reversed integer is stored in reversedN
    while (n != 0)
    {
        remainder = n % 10;
        reversedN = reversedN * 10 + remainder;
        n /= 10;
    }
    // palindrome if originalN and reversedN are equal
    if (originalN == reversedN)
        printf("%d is a palindrome.", originalN);
    else
        printf("%d is not a palindrome.", originalN);
    return 0;
}
```

Output

Enter an integer: 1001
1001 is a palindrome.

Simple Calculator using switch Statement

```
#include <stdio.h>
int main()
{
    char operator;
    double first, second;
    printf("Enter an operator (+, -, *, /): ");
    scanf("%c", &operator);
    printf("Enter two operands: ");
    scanf("%lf%lf", &first, &second);
    switch (operator)
    {
        case '+':printf("%.1lf + %.1lf = %.1lf", first, second, first + second);
            break;
        case '-':printf("%.1lf - %.1lf = %.1lf", first, second, first - second);
            break;
        case '*':printf("%.1lf * %.1lf = %.1lf", first, second, first * second);
            break;
        case '/':printf("%.1lf / %.1lf = %.1lf", first, second, first / second);
            break;
        default:printf("Error! operator is not correct");
    }
    return 0;
}
```

Output

```
Enter an operator (+, -, *, /): *
Enter two operands: 1.54.5
1.5 * 4.5 = 6.75
```

Solved Questions

Short Answer Type Questions.

Q.1 Name of the first developer of C programming languages.

Ans:- Dennis Ritchie

Q.2 What are the various types of statement available in C program?

Ans:- C has three types of statement.

- (i) Assignment =
- (ii) Selection (branching) if (expression) else switch.
- (iii) Iteration (looping) while (expression)for(expression;expression;expression) do {block}

Q.3 What is the purpose of i/o statement in „C“?

Ans:- ☐ It is used to display a string inputted by gets() function. It isalso used to display an text (message) on the screen for program simplicity.

☐ Input Output Statement.

Q.4 Explain the importance of C-language?

Ans:- C is highly portable and is used for scripting system applications which form a major part of Windows, UNIX, and Linux operating system. C is a general-purpose programming language and can efficiently work on enterprise applications, games, graphics, and applications requiring calculations, etc.

Long Answer Type Questions

Q.1 Give the general structure of a „C“ program, and discuss about each of the lines. (2017-Winter)

Ans:- Programming in C is a difficult task for someone who is completely oblivious to the basic structure of a C program. After completing this tutorial, you would learn how the Structure of C Program looks like and soon you would be comfortable writing your own programs with ease! Part of C program

1. **# include <stdio.h>** – This command is a preprocessor directive in C that includes all standard input-output files before compiling any C program so as to make use of all those functions in our C program.
2. **int main()** – This is the line from where the execution of the program starts. The main() function starts the execution of any Program.
3. **{ (Opening bracket)** – This indicates the beginning of any function in the program (Here it indicates the beginning of the main function).
4. **/* some comments */** – Whatever is inside **/*——*/** are not compiled and executed; they are only written for user understanding or for making the program interactive by inserting a comment line. These are known as multiline comments. Single line comments are represented with the help of 2 forward slashes **//——**.
5. **printf(“Hello World”)** –The printf() command is included in the C stdio.h library, which helps to display the message on the output screen.
6. **getch()** – This command helps to hold the screen.
7. **return 0** – This command terminates the C program and returns a null value, that is, 0.
8. **} (Closing brackets)**- This indicates the end of the function. (Here it indicates the end of the main function)

Q.2 WAP in C to find the real roots of a quadratic equation. (2017-Summer)

Ans:- #include <stdio.h> #include <math.h>

```
int main() {
    int a, b, c, d;
    double root1, root2;
    printf("Enter a, b and c where a*x*x + b*x + c = 0\n");
    scanf("%d%d%d", &a, &b, &c);
    d = b*b - 4*a*c; if (d < 0)
    {
        // complex roots, i is for iota (√-1, square root of -1)
        printf("First root = %.2lf + i%.2lf\n", -b/(double)(2*a), sqrt(-d)/(2*a)); printf("Second root = %.2lf - i%.2lf\n", -b/(double)(2*a), sqrt(-d)/(2*a));
    }
    else
    {
        // real roots root1 = (-b + sqrt(d))/(2*a);
        root2 = (-b - sqrt(d))/(2*a);
        printf("First root = %.2lf\n", root1);
        printf("Second root = %.2lf\n", root2);
    }
    return 0;
}
```

EXERCISE

Short Answer Type Questions.

- Q. 1 Differentiate between Numeric and Character Constant?
- Q. 2 How can you use a symbolic statement?
- Q. 3 What do you mean by operator and operand?
- Q. 4 What are the various types of Operator used in the C programming? (2016-Summer)
- Q. 5 Differentiate between logical and bitwise operator?
- Q. 6 Differentiate between increment and decrement operator?
- Q. 7 Differentiate between pre and post increment/decrement operator? (2013- Summer)

- Q. 8 Differentiate between unary plus and unary minus operator?
- Q. 9 Why conditional operator is called ternary operator?
- Q. 10 Define Operator?
- Q. 11 What is the relation of arithmetic operator with relational operator? Explain with an appropriate expression?
- Q. 12 How can expression with increment and decrement operators will be solved?
- Q. 13 Differentiate between if-else and else-if statement?
- Q. 14 Differentiate between ladder if and switch statement? (2017-Summer)
- Q. 15 What do you mean by conditional control statement?
- Q. 16 Define an iterative statement in a C program.
- Q. 17 Give the general syntax of switch..... case statement in C.
- Q. 18 Differentiate between do-while and while..... do statement in C.

Long Answer Type Questions

- Q.1 WAP in C to print all 2-Digit Odd Numbers.
- Q.2 WAP in C to Calculate and print the factorial of a given number.
- Q.3 WAP in C to Compute and print the sum of the following series. **(2017- Winter)** $S = 1 + 1/x + 1/x^2 + 1/x^3 + 1/x^4 + \dots + 1/x^n$
- Q.4 WAP in C to calculate the sum of the digits of a given number.
- Q.5 WAP in C to Compute and print the simple interest and compound interest.
- Q.6 WAP in C to compute $(a+b)^2$.
- Q.7 WAP in C to interchange value of two variables without using third variable.
- Q.8 WAP in to find the sum of the given series. **(2015-Winter)** $1_n + 2_n + 3_n + 4_n + \dots + m_n$.
- Q.9 WAP in C to print.
- *
- **
- ***
- Q. 10 WAP in C to find the prime number. (2017-Summer)
- Q. 11 WAP in C to find greatest number among three integer numbers. **(2015- Winter)**
- Q. 12 WAP in C to find whether a number is Armstrong number or not. *****

CHAPTER –7: ADVANCED FEATURES OF C

7.1 Functions and Passing Parameters to the Function (Call by Value and Call by Reference)

Functions

- A function is a group of statements that together perform a task. Every C program has at least one function, which is main (), and all the most trivial programs can define additional functions.
- You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division is such that each function performs a specific task.
- A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.
- A function can also be referred as a method or a sub-routine or a procedure, etc.

Defining a Function

- The general form of a function definition in C programming language is as follows –
 - return type function name(parameter list) { ○ body of the function ○ }
- A function definition in C programming consists of a function header and a function body. Here are all the parts of a function –
 - **Return Type** – A function may return a value. The return type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return type is the keyword void.
 - **Function Name** – This is the actual name of the function. The function name and the parameter list together constitute the function signature.
 - **Parameters** – A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
 - **Function Body** – The function body contains a collection of statements that define what the function does.

Parameters in C functions

- A Parameter is the symbolic name for "data" that goes into a function. There are two ways to pass parameters in C: Pass by Value, Pass by Reference.

Call by Value

- Pass by Value, means that a copy of the data is made and stored by way of the name of the parameter. Any changes to the parameter have NO effect on data in the calling function.
- In call by value method, the value of the actual parameters is copied into the formal parameters. In other words, we can say that the value of the variable is used in the function call in the call by value method.
- In call by value method, we cannot modify the value of the actual parameter by the formal parameter. ○ In call by value, different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter.
- The actual parameter is the argument which is used in the function call whereas formal parameter is the argument which is used in the function definition.

Call by Reference

- A reference parameter "refers" to the original data in the calling function. Thus, any changes made to the parameter are also made to the original variable.
- In call by reference, the address of the variable is passed into the function call as the actual parameter.

- The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.
- In call by reference, the memory allocation is similar for both formal parameters and actual parameters. All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address.
- There are two ways to make a pass by reference parameter:

ARRAYS

- Arrays are always pass by reference in C. Any change made to the parameter containing the array will change the value of the original array.

The ampersand (&) used in the function prototype. Function (& parameter name)

- To make a normal parameter into a pass by reference parameter, we use the "& param" notation. The ampersand (&) is the syntax to tell C that any changes made to the parameter also modify the original variable containing the data.

Call by Value Example:

Swapping the values of the two variables (Swapping not Possible)

```
#include <stdio.h> void swap(int , int);
//prototype of the function
int main()
{
    int a = 10; int b = 20;
    printf("Before swapping the values in main a = %d, b = %d\n",a,b);
    swap(a,b);
    printf("After swapping values in main a = %d, b = %d\n",a,b);
    // The value of actual parameters do not change by changing the formal parameters in call by value, a = 10, b = 20
}
void swap (int a1, int b1)
{
    int temp;
    temp = a;
    a=b;
    b=temp;
    printf("After swapping values in function a = %d, b = %d\n",a,b);
    // Formal parameters, a = 20, b = 10
}
```

Output

Before swapping the values in main a = 10, b = 20

After swapping values in function a = 20, b = 10

After swapping values in main a = 10, b = 20

Call by reference Example:

Swapping the values of the two variables

```
#include <stdio.h>
void swap(int *, int *);
//prototype of the function
int main()
{
    int a = 10;
    int b = 20;
    printf("Before swapping the values in main a = %d, b = %d\n",a,b);
    // printing the value of a and b in main swap(&a,&b);
}
```

```
printf("After swapping values in main a = %d, b = %d\n",a,b);
// The values of actual parameters do change in call by reference, a = 10, b = 20
}
void swap (int *a, int *b)
{
    int temp;
    temp = *a;
    *a=*b; *b=temp;
    printf("After swapping values in function a = %d, b = %d\n",*a,*b);
// Formal parameters, a = 20, b = 10
}
```

Output:

Before swapping the values in main a = 10, b = 20

After swapping values in function a = 20, b = 10

After swapping values in main a = 20, b = 10

7.2 Scope of Variables and Storage Classes, Recursion, Function and Types of Recursion

7.2.1 Scope of variables

□ When we declare a variable in a program, it cannot be accessed against the scope rules. Variables can be accessed based on their scope. The scope of a variable decides the portion of a program in which the variable can be accessed. The scope of the variable is defined as follows...

- Scope of a variable is the portion of the program where a defined variable can be accessed.
- The variable scope defines the visibility of variable in the program. Scope of a variable depends on the position of variable declaration.
- In C programming language, a variable can be declared in three different positions and they are as follows...

- Before the function definition (Global Declaration)
- Inside the function or block (Local Declaration)
- In the function definition parameters (Formal Parameters)

Before the function definition (Global Declaration)

Example Program

```
#include<stdio.h>
#include<conio.h>
int num1, num2 ;
void main()
{
    void addition() ; void subtraction() ; void multiplication() ;
    clrscr() ;
    num1 = 10 ;
    num2 = 20 ;
    printf("num1 = %d, num2 = %d", num1, num2) ;
    addition() ;
    subtraction() ;
    multiplication() ;
    getch() ;
}
void addition()
{
    int result ;
```

```

result = num1 + num2 ;
printf("\naddition = %d", result) ;
}
void subtraction()
{
int result ;
result = num1 - num2 ;
printf("\nsubtraction = %d", result) ;
}
void multiplication()
{
int result ; result = num1 * num2 ;
printf("\nmultiplication = %d", result) ;
}

```

Output:

Inside the function or block (Local Declaration)

Example Program

```

#include<stdio.h>
#include<conio.h>
void main()
{
void addition() ;
int num1, num2 ;
clrscr() ; num1 = 10 ; num2 = 20 ;
printf("num1 = %d, num2 = %d", num1, num2) ;
addition() ;
getch() ;
}
void addition() { int sumResult ;
sumResult = num1 + num2 ;
printf("\naddition = %d", sumResult) ;
}

```

Output:

In the function definition parameters (Formal Parameters)

Example Program

```

#include<stdio.h>
#include<conio.h>
void main()
{
void addition(int, int) ;
int num1, num2 ;
clrscr() ;
num1 = 10 ;
num2 = 20 ;
addition(num1, num2) ;
getch() ;
}
void addition(int a, int b)
{

```

```
int sumResult ;
sumResult = a + b ;
printf("\naddition = %d", sumResult) ;
}
```

7.2.2 Storage Classes

□ Storage classes in C are used to determine the lifetime, visibility, memory location, and initial value of a variable. There are four types of storage classes in C

- Automatic
- External
- Static
- Register

Automatic

- Automatic variables are allocated memory automatically at runtime.
- The visibility of the automatic variables is limited to the block in which they are defined.
 - The scope of the automatic variables is limited to the block in which they are defined.
- The automatic variables are initialized to garbage by default.
 - The memory assigned to automatic variables gets freed upon exiting from the block.
 - The keyword used for defining automatic variables is auto.
- Every local variable is automatic in C by default.

Example

```
#include <stdio.h>
int main()
{
int a;
//auto char b;
float c;
printf("%d %c %f",a,b,c);
// printing initial default value of automatic variables a, b, and c.
return 0;
}
```

Output:

garbage garbagegarbage

Static

- The variables defined as static specifier can hold their value between the multiple function calls.
- Static local variables are visible only to the function or the block in which they are defined.
- A same static variable can be declared many times but can be assigned at only one time.
- Default initial value of the static integral variable is 0 otherwise null.
 - The visibility of the static global variable is limited to the file in which it has declared.
- The keyword used to define static variable is static.

Example

```
#include<stdio.h>
static char c;
static int i;
static float f;
static char s[100];
void main ()
{
```

```
printf("%d %d %f %s",c,i,f);
// the initial default value of c, i, and f will be printed.
}
```

Output: 0 0 0.000000 (null)

Register

- ☐ The variables defined as the register is allocated the memory into the CPU registers depending upon the size of the memory remaining in the CPU.
- ☐ We cannot dereference the register variables, i.e., we cannot use &operator for the register variable.
- ☐ The access time of the register variables is faster than the automatic variables.
- ☐ The initial default value of the register local variables is 0.
- ☐ The register keyword is used for the variable which should be stored in the CPU register. However, it is compiler's choice whether or not; the variables can be stored in the register.
- ☐ We can store pointers into the register, i.e., a register can store the address of a variable.
- ☐ Static variables cannot be stored into the register since we cannot use more than one storage specifier for the same variable.

Example

```
#include <stdio.h>
int main()
{
    register int a;
    // variable a is allocated memory in the CPU register. The initial default value of a is 0.
    printf("%d",a);
}
```

Output:

0

External

- ☐ The external storage class is used to tell the compiler that the variable defined as extern is declared with an external linkage elsewhere in the program.
- ☐ The variables declared as extern are not allocated any memory. It is only declaration and intended to specify that the variable is declared elsewhere in the program.
- ☐ The default initial value of external integral type is 0 otherwise null.
- ☐ We can only initialize the extern variable globally, i.e., we can not initialize the external variable within any block or method.
- ☐ An external variable can be declared many times but can be initialized at only once.
- ☐ If a variable is declared as external then the compiler searches for that variable to be initialized somewhere in the program which may be extern or static. If it is not, then the compiler will show an error.

Example

```
#include <stdio.h>
int main()
{
    extern int a;
    printf("%d",a);
}
```

Output

main.c:(.text+0x6): undefined reference to `a'
collect2: error: ld returned 1 exit status

7.2.3 Recursion Function

- ☐ Recursion is the process of repeating items in a self-similar way. In programming languages, if a

program allows you to call a function inside the same function, then it is called a recursive call of the function.

```
void recursion()
{
    recursion ();
/* function calls itself */
}
int main()
{
    recursion ();
}
```

□ The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.

□ Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.

7.2.4 Types of Recursion

Recursion are mainly of two types depending on whether a function calls itself from within itself whether two function call one another mutually. Thus, the two types of recursion are:

- Direct recursion
- Indirect recursion

Recursion may be further categorized as:

- Linear recursion
- Binary recursion
- Multiple recursion

7.3 One Dimensional Array and Multidimensional Array, String Operations and Pointers

7.3.1 One-dimensional array

□ Conceptually you can think of a one-dimensional array as a row, where elements are stored one after another.

□ Syntax: data type array_name[size];

□ datatype: It denotes the type of the elements in the array.

□ array_name: Name of the array. It must be a valid identifier.

□ size: Number of elements an array can hold. here are some example of array declarations:

```
int num[100];
float temp[20];
char ch[50];
```

7.3.2 Multidimensional Arrays

□ The simplest form of multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size [x][y], you would write something as follows –

```
type array Name [ x ][ y ];
```

- Where type can be any valid C data type and array Name will be a valid C identifier.

- A two-dimensional array can be considered as a table which will have x number of rows and y number

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

of columns.

- A two-dimensional array a, which contains three rows and four columns can be shown as follows –
- Thus, every element in the array a is identified by an element name of the form a[i][j], where 'a' is the name of the array, and 'i' and 'j' are the subscripts that uniquely identify each element in 'a'.

7.3.3 String operations

- Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.
- The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

- If you follow the rule of array initialization then you can write the above statement as follows –

```
char greeting[] = "Hello";
```

- Following is the memory presentation of the above defined string in C/C++ –

- Actually, you do not place the null character at the end of a string constant. The C compiler automatically places the '\0' at the end of the string when it initializes the array. Let us try to print the above mentioned string – Live Demo

```
#include <stdio.h>
int main ()
{
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
printf("Greeting message: %s\n", greeting );
return 0;
}
```

7.3.4 Pointers

- A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address. The general form of a pointer variable declaration is –

```
type *var-name;
```

- Here, type is the pointer's base type; it must be a valid C data type and var-name is the name of the pointer variable. The asterisk * used to declare a pointer is the same asterisk used for multiplication.

How to Use Pointers?

- There are a few important operations, which we will do with the help of pointers very frequently.
- We define a pointer variable,
- assign the address of a variable to a pointer and
- Finally access the value at the address available in the pointer variable.
- This is done by using unary operator * that returns the value of the variable located at the address specified by its operand. The following example makes use of these operations –

```
#include <stdio.h>
int main ()
```

```

{
int var = 20;
/* actual variable declaration */
int *ip;
/* pointer variable declaration */
ip = &var; /* store address of var in pointer variable*/
printf("Address of var variable: %x\n", &var );
/* address stored in pointer variable */
printf("Address stored in ip variable: %x\n", ip );
/* access the value using the pointer */
printf("Value of *ip variable: %d\n", *ip );
return 0;
}

```

7.4 Pointer Expression and Pointer Arithmetic Programming, Assignments using the above features.

□ Pointers are used to point to address the location of a variable. A pointer is declared by preceding the name of the pointer by an asterisk (*).

□ Syntax:

```
datatype *pointer_name;
```

□ When we need to initialize a pointer with variable's location, we use ampersand sign(&) before the variable name.

// Declaration of integer variable

```
int var=10;
```

// Initialization of pointer variable

```
int *pointer=&var;
```

➤ The ampersand (&) is used to get the address of a variable. We can directly find the location of any identifier by just preceding it with an ampersand (&) sign.

Example:

```

// This code prints the address of x
#include <stdio.h>
int main()
{
int x = 10;
// Prints address of x
printf("Address of variable x = %p", &x);
return 0;
}

```

Pointer Arithmetic Programming

Pointer in c is an address, which is a numeric value. Therefore, you can perform arithmetic operations on a pointer just as you can on a numeric value. There are four arithmetic operators that can be used on pointers: ++, --, +, and –

To understand pointer arithmetic, let us consider that ptr is an integer pointer which points to the address 1000. Assuming 32-bit integers, let us perform the following arithmetic operation on the pointer –ptr++

After the above operation, the ptr will point to the location 1004 because each time ptr is incremented, it

will point to the next integer location which is 4 bytes next to the current location. This operation will move the pointer to the next memory location without impacting the actual value at the memory location. If ptr points to a character whose address is 1000, then the above operation will point to the location 1001 because the next character will be available at 1001.

Incrementing a Pointer

We prefer using a pointer in our program instead of an array because the variable pointer can be incremented, unlike the array name which cannot be incremented because it is a constant pointer. The following program increments the variable pointer to access each succeeding element of the array –

```
#include <stdio.h>
const int MAX = 3;
void main () {
    int var[] = {10, 100, 200};
    int i, *ptr;
    ptr = var;
    /* let us have array address in pointer */
    for ( i = 0; i < MAX; i++)
    {
        printf("Address of var[%d] = %x\n", i, ptr );
        printf("Value of var[%d] = %d\n", i, *ptr );
        ptr++;
    }
    /* move to the next location */
}
```

Output

```
Address of var[0] = bf882b30
Value of var[0] = 10
Address of var[1] = bf882b34
Value of var[1] = 100
Address of var[2] = bf882b38
Value of var[2] = 200
```

Decrementing a Pointer

The same considerations apply to decrementing a pointer, which decreases its value by the number of bytes of its data type as shown below –

```
#include <stdio.h>
const int MAX = 3;
int main ()
{
    int var[] = {10, 100, 200};
    int i, *ptr;
    ptr = &var[MAX-1];
    /* let us have array address in pointer */
    for ( i = MAX; i > 0; i--)
    {
        printf("Address of var[%d] = %x\n", i-1, ptr );
        printf("Value of var[%d] = %d\n", i-1, *ptr ); ptr--;
    }
    /* move to the previous location */
}
```

```
return 0;
}
Output Address of var[2] = bfeedbcd8
Value of var[2] = 200
Address of var[1] = bfeedbcd4
Value of var[1] = 100
Address of var[0] = bfeedbcd0
Value of var[0] = 10
```

Pointer Comparisons

Pointers may be compared by using relational operators, such as ==, <, and >. If p1 and p2 point to variables that are related to each other, such as elements of the same array, then p1 and p2 can be meaningfully compared.

The following program modifies the previous example – one by incrementing the variable pointer so long as the address to which it points is either less than or equal to the address of the last element of the array, which is &var[MAX - 1] –

Example

```
#include <stdio.h> const int MAX = 3;
int main ()
{
    int var[] = {10, 100, 200};
    int i, *ptr;
    /* let us have address of the first element in pointer */
    ptr = var; i = 0;
    while ( ptr<= &var[MAX - 1] )
    {
        printf("Address of var[%d] = %x\n", i, ptr );
        printf("Value of var[%d] = %d\n", i, *ptr );
        /* point to the next location */ ptr++; i++;
    }
    return 0;
}
```

Output

```
Address of var[0] = bfeedbcb20
Value of var[0] = 10
Address of var[1] = bfeedbcb24
Value of var[1] = 100
Address of var[2] = bfeedbcb28
Value of var[2] = 200
```

7.5 Structure and Union (Only concepts, No Programming)

What is a structure?

A structure is a user defined data type in C. A structure creates a data type that can be used to group items of possibly different types into a single type.

How to create a structure?

—**struct** keyword is used to create a structure.

Following is an example

```

struct address
{
char name[50];
char street[100];
char city[50];
char state[20];
int pin;
};

```

How to declare structure variables?

☐ A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

☐ A variable declaration with structure declaration.

```

struct Point {
int x, y;
} p1;

```

The variable p1 is declared with 'Point'.

☐ A variable declaration like basic data types

```

struct Point

```

```

{
int x, y;
};

```

```

int main()

```

```

{ struct Point p1; // The variable p1 is declared like a normal variable

```

```

}

```

Union

☐ A union is a special data type available in C that allows storing different data types in the same memory location.

☐ You can define a union with many members, but only one member can contain a value at any given time.

☐ Unions provide an efficient way of using the same memory location for multiple purposes.

Defining a Union:

☐ To define a union, you must use the union statement in the same way as you did while defining a structure.

☐ The union statement defines a new data type with more than one member for your program.

☐ The format of the union statement is as follows:

```

union [union name]
{
member definition;
member definition;
... member definition; };

```

	STRUCTURE	UNION
Keyword	The keyword struct is used to define a structure	The keyword union is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members.	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of union is equal to the size of largest member.
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

Standard Library Functions

Many basic housekeeping functions are available to a C program in form of standard library functions. To call these, a program must `#include` the appropriate `.h` file. Most compilers link in the standard library code by default. The functions listed in the next section are the most commonly used ones, but there are many more which are not listed here.

<code>stdio.h</code>	file input and output
<code>ctype.h</code>	character tests
<code>string.h</code>	string operations
<code>math.h</code>	mathematical functions such as <code>sin()</code> and <code>cos()</code>
<code>stdlib.h</code>	utility functions such as <code>malloc()</code> and <code>rand()</code>
<code>assert.h</code>	the <code>assert()</code> debugging macro
<code>stdarg.h</code>	support for functions with variable numbers of arguments
<code>setjmp.h</code>	support for non-local flow control jumps
<code>signal.h</code>	support for exceptional conditions signals
<code>time.h</code>	date and time
<code>limits.h</code>	float.h constants which define type range values such as <code>INT_MAX</code>

Examples of advance programming in C

Example of Array In C programming to find out the average of 4 integers

```
#include <stdio.h> int main()
{
    int avg = 0;
    int sum = 0;
    int x = 0;
    int num[4];
    /* Array- declaration – length 4 */
    for (x = 0; x < 4; x++)
    /* We are using for loop to traverse through the */
    {
        printf("Enter number %d \n", (x+1));
        scanf("%d", &num[x]);
    }
    for (x = 0; x < 4; x++)
    {
        sum = sum + num[x];
    }
}
```

```
avg = sum/4; printf("Average of entered number is: %d", avg);  
return 0;  
}
```

Output:

```
Enter number 1 10  
Enter number 2 10  
Enter number 3 20  
Enter number 4 40  
Average of entered number is: 20
```

Example to print the address of array elements

```
#include <stdio.h> int main( )  
{  
int val[7] = { 11, 22, 33, 44, 55, 66, 77 } ;  
/* for loop to print value and address of each element of array*/  
for ( inti = 0 ; i< 7 ; i++ )  
{  
printf("val[%d]: value is %d and address is %d\n", i, val[i], &val[i]);  
}  
return 0;  
}
```

Output:

```
val[0]: value is 11 and address is 1423453232  
val[1]: value is 22 and address is 1423453236  
val[2]: value is 33 and address is 1423453240  
val[3]: value is 44 and address is 1423453244  
val[4]: value is 55 and address is 1423453248  
val[5]: value is 66 and address is 1423453252  
val[6]: value is 77 and address is 1423453256
```

Example: Passing Pointer to a Function in C

```
#include <stdio.h>  
void salaryhike(int *var, int b)  
{  
*var = *var+b;  
}  
int main()  
{  
int salary=0, bonus=0; printf("Enter the employee current salary:");  
scanf("%d", &salary);  
printf("Enter bonus:");  
scanf("%d", &bonus);  
salaryhike(&salary, bonus);  
printf("Final salary: %d", salary);  
return 0;  
}
```

Output:

```
Enter the employee current salary:10000  
Enter bonus:2000
```

Final salary: 12000

Example for Swapping two numbers using Pointers

```
#include <stdio.h>
void swapnum(int *num1, int *num2)
{
    int tempnum; tempnum = *num1;
    *num1 = *num2;
    *num2 = tempnum;
}
int main( )
{
    int v1 = 11, v2 = 77 ;
    printf("Before swapping:"); printf("\nValue of v1 is: %d", v1);
    printf("\nValue of v2 is: %d", v2);
    /*calling swap function*/
    swapnum( &v1, &v2 );
    printf("\nAfter swapping:");
    printf("\nValue of v1 is: %d", v1);
    printf("\nValue of v2 is: %d", v2);
}
```

Output:

Before swapping:

Value of v1 is: 11

Value of v2 is: 77

After swapping:

Value of v1 is: 77

Value of v2 is: 11

Example:

Program to find the size of an array

```
#include <stdio.h>
int main()
{
    double arr[] = {11, 22, 33, 44, 55, 66};
    int n;
    /* Calculating the size of the array with this formula.
    * n = sizeof(array_name) / sizeof(array_name[0])
    * This is a universal formula to find number of elements in
    * an array, which means it will work for arrays of all data
    * types such as int, char, float etc.
    */
    n = sizeof(arr) / sizeof(arr[0]);
    printf("Size of the array is: %d\n", n);
    return 0;
}
```

Output:

Size of the array is: 6

Example:

Program to Calculate Standard Deviation

```

#include <math.h>
#include <stdio.h>
float calculateSD(float data[]);
int main()
{
int i; float data[10]; printf("Enter 10 elements: ");
for (i = 0; i < 10; ++i)
scanf("%f", &data[i]);
printf("\nStandard Deviation = %.6f", calculateSD(data));
return 0;
}
float calculateSD(float data[])
{
float sum = 0.0, mean, SD = 0.0; int i;
for (i = 0; i < 10; ++i) { sum += data[i]; }
mean = sum / 10;
for (i = 0; i < 10; ++i) SD += pow(data[i] - mean, 2);
return sqrt(SD / 10);
}

```

Output

Enter 10 elements:

1
2
3
4
5
6
7
8
9
10

Standard Deviation = 2.872281

Example:

Program to Add Two Matrices

```

#include <stdio.h>
int main()
{
int r, c, a[100][100], b[100][100], sum[100][100], i, j;
printf("Enter the number of rows (between 1 and 100): ");
scanf("%d", &r);
printf("Enter the number of columns (between 1 and 100): ");
scanf("%d", &c);
printf("\nEnter elements of 1st matrix:\n");
for (i = 0; i < r; ++i) for (j = 0; j < c; ++j)
{
printf("Enter element a%d%d: ", i + 1, j + 1);
scanf("%d", &a[i][j]); }
printf("Enter elements of 2nd matrix:\n");
for (i = 0; i < r; ++i) for (j = 0; j < c; ++j)
{

```

```

printf("Enter element a%d%d: ", i + 1, j + 1);
scanf("%d", &b[i][j]);
}
for (i = 0; i < r; ++i)
// adding two matrices
for (j = 0; j < c; ++j)
sum[i][j] = a[i][j] + b[i][j];
printf("\nSum of two matrices: \n");
// printing the result
for (i = 0; i < r; ++i)
for (j = 0; j < c; ++j) {
printf("%d ", sum[i][j]);
if (j == c - 1)
{
printf("\n\n");
}
}
return 0;
}

```

Output

```

Enter the number of rows (between 1 and 100): 2
Enter the number of columns (between 1 and 100): 3
Enter elements of 1st matrix:
Enter element a11: 2
Enter element a12: 3
Enter element a13: 4
Enter element a21: 5
Enter element a22: 2
Enter element a23: 3
Enter elements of 2nd matrix:
Enter element a11: -4
Enter element a12: 5
Enter element a13: 3
Enter element a21: 5
Enter element a22: 6
Enter element a23: 3
Sum of two matrices:
-2 8 7
10 8 6

```

Example:

Multiply Matrices by Passing it to a Function

```

#include <stdio.h>
// function to get matrix elements entered by the user
void getMatrixElements(int matrix[][10], int row, int column)
{
printf("\nEnter elements: \n");
for (int i = 0; i < row; ++i)
{
for (int j = 0; j < column; ++j)
{

```



```

printf("Enter a%d%d: ", i + 1, j + 1);
scanf("%d", &matrix[i][j]); } } }
// function to multiply two matrices
void multiplyMatrices(int first[][10], int second[][10], int result[][10], int r1, int c1, int r2, int c2)
{
    // Initializing elements of matrix mult to 0.
    for (int i = 0; i < r1; ++i)
    {
        for (int j = 0; j < c2; ++j)
        {
            result[i][j] = 0;
        }
    }
    // Multiplying first and second matrices and storing it in result
    for (int i = 0; i < r1; ++i)
    {
        for (int j = 0; j < c2; ++j)
        {
            for (int k = 0; k < c1; ++k)
            {
                result[i][j] += first[i][k] * second[k][j];
            }
        }
    }
}
// function to display the matrix
void display(int result[][10], int row, int column)
{
    printf("\nOutput Matrix:\n");
    for (int i = 0; i < row; ++i)
    {
        for (int j = 0; j < column; ++j)
        {
            printf("%d ", result[i][j]); if (j == column - 1)
            printf("\n");
        }
    }
}
int main()
{
    int first[10][10], second[10][10], result[10][10], r1, c1, r2, c2; printf("Enter rows and column for the first
matrix: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter rows and column for the second matrix: ");
    scanf("%d %d", &r2, &c2);
    // Taking input until
    // 1st matrix columns is not equal to 2nd matrix row
    while (c1 != r2)
    {
        printf("Error! Enter rows and columns again.\n"); printf("Enter rows and columns for the first matrix: ");
        scanf("%d %d", &r1, &c1);
    }
}

```

```

printf("Enter rows and columns for the second matrix: ");
scanf("%d%d", &r2, &c2); }
getMatrixElements(first, r1, c1); // get elements of the first matrix
getMatrixElements(second, r2, c2); // get elements of the second matrix
multiplyMatrices(first, second, result, r1, c1, r2, c2); // multiply two matrices.
display(result, r1, c2); // display the result
return 0;
}

```

Output

Enter rows and column for the first matrix:

2 3

Enter rows and column for the second matrix:

3 2

Enter elements:

Enter a11: 2

Enter a12: -3

Enter a13: 4

Enter a21: 5

Enter a22: 3

Enter a23: 5

Enter elements:

Enter a11: 3

Enter a12: 3

Enter a21: 5

Enter a22: 0

Enter a31: -3

Enter a32: 4

Output Matrix:

-21 22

159 179

Example:

Program to Find the Transpose of a Matrix

```
#include <stdio.h>
```

```
int main()
```

```
{
    int a[10][10], transpose[10][10], r, c, i, j;
```

```
    printf("Enter rows and columns: ");
```

```
    scanf("%d %d", &r, &c);
```

```
    // Assigning elements to the matrix
```

```
    printf("\nEnter matrix elements:\n");
```

```
    for (i = 0; i < r; ++i)
```

```
    for (j = 0; j < c; ++j)
```

```
    {
        printf("Enter element a%d%d: ", i + 1, j + 1);
```

```
        scanf("%d", &a[i][j]);
```

```
    }
```

```
    // Displaying the matrix a[][]
```

```
    printf("\nEnter matrix: \n");
```

```

for (i = 0; i < r; ++i)
for (j = 0; j < c; ++j)
{
printf("%d ", a[i][j]);
if (j == c - 1)
printf("\n");
}
// Finding the transpose of matrix A
for (i = 0; i < r; ++i)
for (j = 0; j < c; ++j)
{
transpose[j][i] = a[i][j];
}
// Displaying the transpose of matrix a
printf("\nTranspose of the matrix:\n");
for (i = 0; i < c; ++i)
for (j = 0; j < r; ++j)
{
printf("%d ", transpose[i][j]);
if (j == r - 1) printf("\n");
}
return 0;
}

```

Output Enter rows and columns: 2 3

Enter matrix elements:

Enter element a11: 1

Enter element a12: 4

Enter element a13: 0

Enter element a21: -5

Enter element a22: 2

Enter element a23: 7

Entered matrix:

1 4 0

-5 2 7

Transpose of the matrix:

1 -5

4 2

0 7

Using Call by Reference

```

#include <stdio.h>
void cyclicSwap(int *a, int *b, int *c);
int main() { int a, b, c;
printf("Enter a, b and c respectively: ");
scanf("%d %d %d", &a, &b, &c);
printf("Value before swapping:\n");
printf("a = %d \nb = %d \nc = %d\n", a, b, c);
cyclicSwap(&a, &b, &c);
printf("Value after swapping:\n");

```

```
printf("a = %d \nb = %d \nc = %d", a, b, c);  
return 0;  
}  
void cyclicSwap(int *n1, int *n2, int *n3)  
{  
    int temp; // swapping in cyclic order  
    temp = *n2;  
    *n2 = *n1;  
    *n1 = *n3;  
    *n3 = temp;  
}
```

Output Enter a, b and c respectively:

1 2 3

Value before swapping: a = 1 b = 2 c = 3

Value after swapping: a = 3 b = 1 c = 2