IDEAL SCHOOL OF ENGINEERING

RETANG, BHUBANESWAR

COMPUTER APPLICATION
(TH.1.b)

1st and 2nd Semester (Diploma Course)
(As per the syllabus prepared by the SCTE&VT, Bhubaneswar, Odisha)

shutterstock.com - 295595840

COMPUTER APPLICATION (TH.1.b)

Topic Wise Distribution of Marks

1. Ch1 Computer Organization-------------10marks

2. Ch2 Computer Software ---- 15 marks

3. CH-3- Computer Network and Internet—15 mark

4. Ch-4 File Management and data processing-10marks

5. Ch-5 ---Problem Solving Methodology—10marks

6. Ch-6 –Overview of C programming language 25marks

7. Ch-7 -Advance Features of C. - 25marks

COMPUTER APPLICATION

(1st / 2nd sem Common)

Theory: 4 Periods per Week	I.A : 20 Marks
Total Periods: 60 Periods	End Sem Exam : 80 Marks
Examination: 3 Hours	TOTAL MARKS : 100 Marks

Objective:
The students will get to know about the fundamentals of computer. They will get acquainted with various components of computer hardware, software etc. Idea on Role of operating system and its usability will also be known. Knowledge on word processing, electronic spreadsheet, presentation software and Internet will also be acquired. The students will be given brief knowledge about Programming methodology and C programming.

Topic wise distribution of periods

Sl. No.	Topics	Periods
1	Computer Organisation	05
2	Computer Software	07
3	Computer Network and Internet	08
4	File Management and Data Processing	05
5	Problem Solving Methodology	05
6	Overview of C Programming language	15
7	Advanced features of C	15
	TOTAL	**60**

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, Date 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 Pointers
Pointer Expression and Pointer Arithmetic Programming Assignments using the above features. Structure and Union (Only concepts, No Programming)

Syllabus coverage upto I.A

Chapter- 1,2 3,4

Books Recommended

1. Computer Fundamentals and Programming in C by Reema Thareja, Oxford Unversity Press
2. Programming in ANSI C by A.N Kamthane, Pearson Education
3. Computer Application by Kalyani Publisher
4. Let us C by Y. Kanetkar, BPB
5. Computer Fundamentals, by E. Balaguruswamy, TMH

CH1 -COMPUTER ORGNISATION

1.1/1.2EVOLUTION 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. 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.

1.3 First Generation – Vacuum Tubes (1940 – 1956)

- These ancient computers utilized vacuum tubes as circuitry and magnetic drums for recollection. As a result, they were huge, actually taking up entire rooms and costing resources to run.
- These were ineffective materials which produce a huge amount of heat, sucked enormous electricity and subsequently engendered an abundance of heat which caused perpetual breakdowns.
- These first-generation computers relied on 'machine language' (which is the most fundamental programming language that can be understood by computers).
- These computers were limited to solving one problem at a time. Input was predicated on punched cards and paper tape. Output emerged on print-outs. The two eminent machines of this era were the UNIVAC and ENIAC machines – the UNIVAC is the first ever commercial computer which was purchased in 1951 by a business named as the US Census Bureau.

Second Generation – Transistors (1956 – 1963)

- Transistor first invented in 1947, transistors weren't used considerably in computers until the cessation of the 1950s.
- They were a huge development over the vacuum tube.
- They were extremely superior to the vacuum tubes, making computers smaller, more expeditious, inexpensive and less burdensome on electricity use.
- The language emerged from strange binary language to symbolic ('assembly') languages. This meant programmers could discover instructions in words. Meanwhile during the same time high calibre programming

languages were being developed (early versions of COBOL and FORTRAN).

Third Generation – Integrated Circuits (1964 – 1971)

- These were the first computers where users interacted utilizing keyboards and monitors which interfaced with an operating system, a consequential leap up from the punch cards and printouts.
- This facilitated these machines to run various applications at once utilizing a central program which functioned to monitor memory. As a result of these advances which again made machines more reasonable and tinier, a brand-new group of users emerged during the '60s.

Fourth Generation – Microprocessors (1972 – 2010)

- This innovation can be defined in one word: Intel. The chip-maker accomplished the Intel 4004 chip in 1971, which located all components of computer such as CPU, recollection, input/output controls onto a single chip.
- The Intel chip contained thousands of unified circuits.
- The year 1981 saw the first ever computer (IBM) categorically designed for home use and 1984 saw the Macintosh introduced by Apple.
- The incremented power of these small computers denoted they could be linked, establishing networks. Other primary advances during this period have been the Graphical user interface (GUI), the mouse and more of late the startling advances in laptop capability and hand-held contrivances.

Fifth Generation – Artificial Intelligence (2010 Onwards)

- Computer devices with artificial potentiality are still in development, but some of these technologies are commencing to emerge and be used such as voice recognition.
- AI is an authenticity, made possible by adopting parallel processing and superconductors.
- Inclining to the future, computers will be thoroughly revolutionized again by quantum computation, molecular and nano technology. The essence of fifth generation will be utilizing these technologies to ultimately engender machines which can proceed and acknowledge natural language, and have efficiency to determine and organise themselves.

1.4. CLASSIFICATION OF COMPUTERS

The computer systems can be classified on the following basis.

Classification on the basis of data handling:

Analog:

- An analog computer is a form of computer that uses the continuously-changeable aspects of physical fact such as electrical, mechanical, or hydraulic quantities to model the problem being solved.
- Anything that is variable with respect to time and continuous can be claimed as analog just like an analog clock measures time by means of the distance travelled for the spokes of the clock around the circular dial.

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.
- From manipulation of the combinations of the binary digits, it can perform mathematical calculations, organize and analyze data, control industrial and other processes, and simulate dynamic systems such as global weather patterns.

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.

Classification on the basis of size:

Super Computers:

- The super computers are the highest 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, higher 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 quantum mechanics, weather forecasting, climate research, oil and gas exploration, molecular modelling, and physical simulations.

Eg: PARAM, jaguar, roadrunner.

Mainframe Computers:

- These are commonly called as big iron, they are usually used by big organisations for bulk data processing such as statics, census data processing, transaction processing and are widely used as the severs as these systems has a higher processing capability as compared to the other classes of computers, most of these mainframe architectures were established in 1960s, the research and development worked continuously over the years and the mainframes of today are far more better than the earlier ones, in size, capacity and efficiency.

Eg: IBM z Series, System z9 and System z10 servers.

Mini computers:

- These computers came into the market in mid 1960s and were sold at a much cheaper price than the main frames, they were actually designed for control, instrumentation, human interaction, and communication switching as distinct from calculation and record keeping, later they became very popular for personal uses with evolution.
- In the 60s to describe the smaller computers that became possible with the use of transistors and core memory technologies, minimal instruction sets and less expensive peripherals such as the ubiquitous Teletype Model 33 ASR.
- They usually took up one or a few inch rack cabinets, compared with the large mainframes that could fill a room, there was a new term "MINICOMPUTERS" coined

Eg: Personal Laptop, PC etc.

Micro Computers:

- A microcomputer is a small, relatively inexpensive computer with a microprocessor as its CPU.
- It includes a microprocessor, memory, and minimal I/O circuitry mounted on a single printed circuit board.
- The previous to these computers, mainframes and minicomputers, were comparatively much larger, hard to maintain and more expensive.
- They actually formed the foundation for present day microcomputers and smart gadgets that we use in day to day life.

Eg: Tablets, Smartwatches.

Classification on the basis of functionality:

Servers: Servers are nothing but dedicated computers which are set-up to offer some services to the clients. They are named depending on the type of service they offered.

Eg: security server, database server.

Workstation: Those are the computers designed to primarily to be used by single user at a time. They run multi-user operating systems. They are the ones which we use for our day to day personal / commercial work.

Information Appliances:

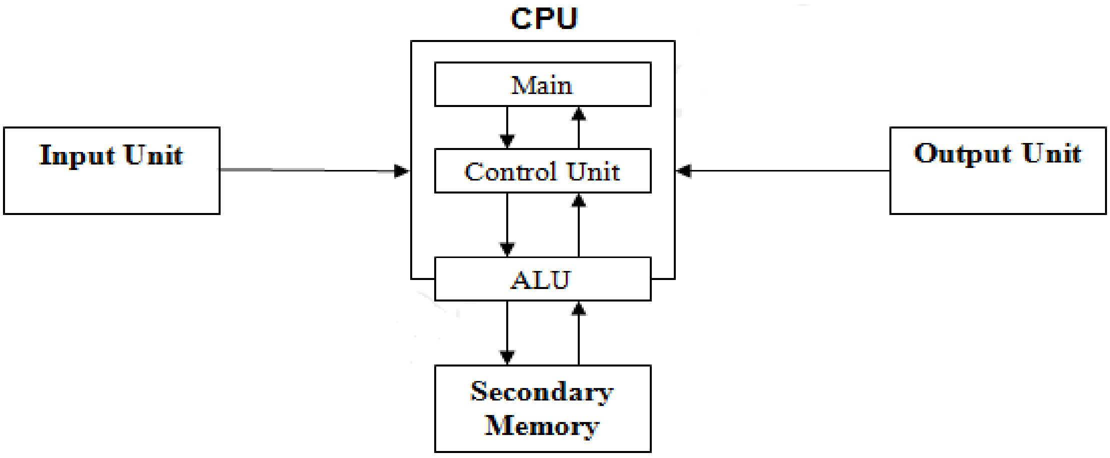
- They are the portable devices which are designed to perform a limited set of tasks like basic calculations, playing multimedia, browsing internet etc.
- They are generally referred as the mobile devices.
- They have very limited memory and flexibility and generally run on “as-is” basis.

Embedded Computers:

- They are the computing devices which are used in other machines to serve limited set of requirements. They follow instructions from the non-volatile memory and they are not required to execute reboot or reset.
- The processing units used in such device work to those basic requirements only and are different from the ones that are used in personal computers- better known as workstations.

1.5 BASIC ORGANIZATION OF A COMPUTER SYSTEM

- Any computer can perform the four basic operations of Input, Processing, Output, and Storage (IPOS).
- These operations constitute the IPOS cycle.
- The internal design or structure of a computer may differ from one system to another though the basic operations remain the same.
- The figure displays all the functional units of a computer which carry out the basic computer operations.
- The lines in the figure indicate the flow of instructions and data, while the Control Unit and the Arithmetic/Logical Unit together direct the flow of control in the central processing unit.



Input Unit:

- Data and instructions are entered into the computer through the input unit to get processed into information.
- Input devices like the keyboard, the mouse, or the microphone are used to enter the data.
- The data is entered in various forms depending on the type of input devices. For instance, a keyboard can be used to input characters, numbers, and certain symbols; a mouse is a device that has an on-screen pointer that enables the users to select items and choose options; a microphone can be used if the user wishes to enter instructions by making a voice entry.
- Regardless of the ways in which the input devices receive the inputs, the input interfaces convert them into binary codes, i.e., 0s and 1s, as the primary memory of the computer is designed to accept data only in this format. Several advancements can be seen in input devices with devices like cordless keyboards, optical mouse, laser mouse, cordless mouse, etc., being introduced in the market.

Central Processing Unit:

The actual processing of the data is carried out in the Central Processing Unit (CPU), which is the brain of computer. The CPU stores the data and instructions in the primary memory of the computer, called the Random Access Memory (RAM) and processes them from this location. The Arithmetic Logic Unit (ALU) and the Control Unit (CU) are the two subcomponents of the CPU. The ALU carries out the arithmetic and logical operations while the CU retrieves the information from the storage unit and interprets this information. The CPU also consists of circuitry devices called cache and registers.

Arithmetic Logic Unit:

The data and instructions stored in the RAM are transferred to the ALU for processing. The ALU performs the logical and the arithmetic operations on the data and the results are temporarily stored in the RAM. After the processing, the final results are stored in the secondary memory, i.e., the storage unit, and are released through an output device.

Control Unit:

The CU obtains the program instructions stored in the primary memory of the computer, interprets them, and issues signals that result in their execution. It helps in maintaining order and directs the operations of the entire system. It selects, interprets, and ensures the proper execution of the program instructions.

Processors:

Some computers use more than one processor for processing in order to reduce the load on a single processor.

Output Unit:

The output unit passes on the final results of computation to the users through the output devices like the monitor, printer, etc. A monitor displays the final results of the processed data on the screen while a printer can be used for obtaining the output in a printed format. These output devices link the computer with the users. The output interfaces convert the binary code produced by the computer into the human-readable form.

Storage Unit:

Before the actual processing takes place, the data and instructions that enter the computer system have to be stored internally. Also, the final results generated by the computer after processing has to be stored before being sent to the output unit. The storage unit of a computer system is designed to store the data generated at various stages of processing. Storage media like hard disks, floppy disks, etc., aid in storing the data in various forms. The hard disk is an integral part of the computer system. It is also referred to as hard drive, disk drive, or hard disk drive. The hard disk provides a large amount of storage space for the programs and data. Computers these days feature a hard disk that has several gigabytes of storage capacity. The floppy disk drives, CD-ROM/CD-RW drives, DVD drives, and USB ports enable the user to store and exchange data with others using storage media like floppy disks, compact discs (CDs), digital video discs (DVDs), and pen drives.

1.6 Input and output devices

- Input—keyboard,mouse joystick, touchpad , touchscreen, scanner MICR,OMR etc
- Output- Monitor, Printers, projectoretc

1.7 CLASSIFICATION OF COMPUTER MEMORY

- Computer memory is a generic term for all of the different types of data storage technology that a computer may use, including RAM, ROM, and flash memory.
- Some types of computer memory are designed to be very fast, meaning that the central processing unit (CPU) can access data stored there very quickly.
- Other types are designed to be very low cost, so that large amounts of data can be stored there economically.Another way that computer memory can vary is that some types are non-volatile, which means they can store data on a long-term basis even when there is no power. And some types are volatile,

which are often faster, but which lose all the data stored on them as soon as the power is switched off.

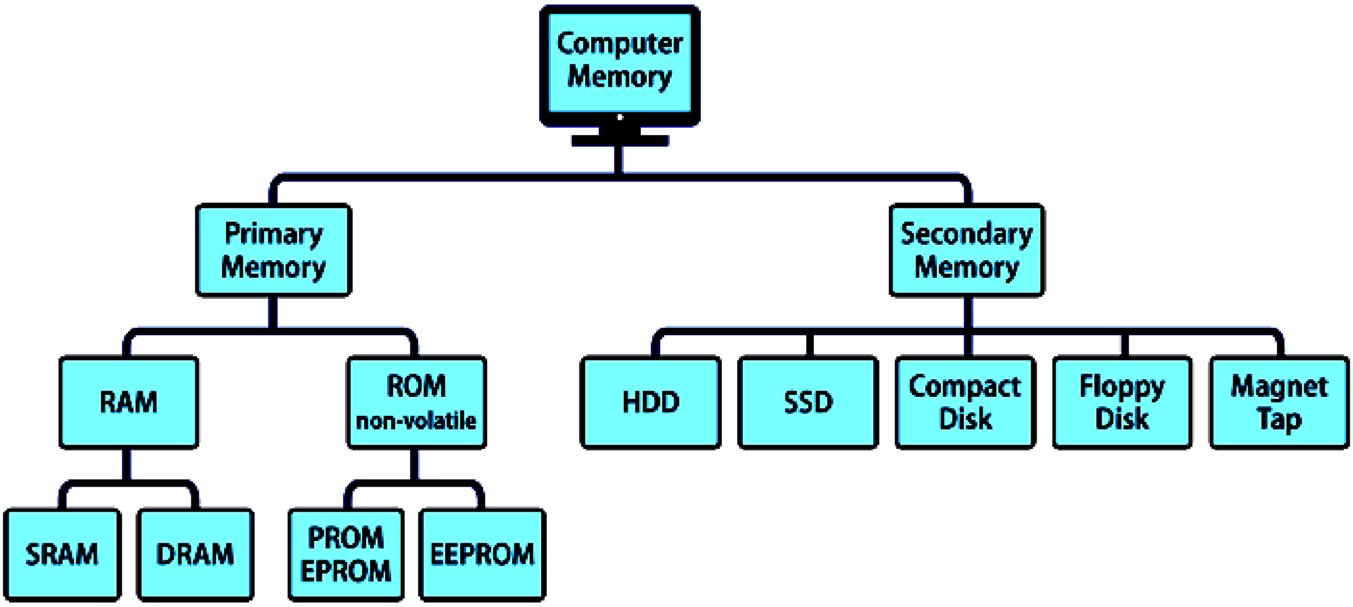
- A computer system is built using a combination of these types of computer memory, and the exact configuration can be optimized to produce the maximum data processing speed or the minimum cost, or some compromise between the two.

Types of Computer Memory: Primary and Secondary

Although many types of memory in a computer exist, the most basic distinction is between primary memory, often called system memory, and secondary memory, which is more commonly called storage.The key difference between primary and secondary memory is speed of access.

Primary memory includes ROM and RAM, and is located close to the CPU on the computer motherboard, enabling the CPU to read data from primary memory very quickly indeed. It is used to store data that the CPU needs imminently so that it does not have to wait for it to be delivered.

Secondary memory by contrast, is usually physically located within a separate storage device, such as a Hard Disk Drive (HDD)or Solid State Drive (SSD), which is connected to the computer system either directly or over a network. The cost per gigabyte of secondary memory is much lower, but the read and write speeds are significantly slower.



Primary Memory Types: RAM and ROM

There are two key types of primary memory:

RAM or Random Access Memory, ROM or ReadOnly Memory

1) RAM Computer Memory

- The acronym RAM stems from the fact that data stored in random access memory can be accessed – as the name suggests – in any random order. Or, put another way, any random bit of data can be accessed just as quickly as any other bit.

- The most important things to understand about RAM are that RAM memory is very fast, it can be written to as well as read, it is volatile (so all data stored in RAM memory is lost when it loses power) and, finally, it is very expensive compared to all types of secondary memory in terms of cost per gigabyte.
- Data that is required for imminent processing is moved to RAM where it can be accessed and modified very quickly, so that the CPU is not kept waiting.

Types of RAM

DRAM: DRAM stands for Dynamic RAM, and it is the most common type of RAM used in computers. The oldest type is known as single data rate (SDR) DRAM, but newer computers use faster dual data rate (DDR) DRAM. DDR comes in several versions including DDR2, DDR3, and DDR4, which offer better performance and are more energy efficient than DDR.

SRAM: SRAM stands for Static RAM, and it is a particular type of RAM which is faster than DRAM, but more expensive and bulker, having six transistors in each cell. For those reasons SRAM is generally only used as a data cache within a CPU itself or as RAM in very high-end server systems.

2) ROM Computer Memory

- ROM stands for read-only memory, and the name stems from the fact that while data can be read from this type of computer memory, data cannot normally be written to it.
- It is a very fast type of computer memory which is usually installed close to the CPU on the motherboard.
- ROM is a type of non-volatile memory, which means that the data stored in ROM persists in the memory even when it receives no power – for example when the computer is turned off.
- In that sense it is similar to secondary memory, which is used for long term storage.
- When a computer is turned on, the CPU can begin reading information stored in ROM without the need for drivers or other complex software to help it communicate.
- The ROM usually contains "bootstrap code" which is the basic set of instructions a computer needs to carry out to become aware of the operating system stored in secondary memory, and to load parts of the operating system into primary memory so that it can start up and become ready to use.
- ROM is also used in simpler electronic devices to store firmware which runs as soon as the device is switched on.

Types of ROM

ROM is available in several different types, including PROM, EPROM, and EEPROM.

- **PROM:** PROM stands for Programmable Read-Only Memory, and it is different from true ROM in that while a ROM is programmed (i.e. has data written to it) during the manufacturing process, a PROM is manufactured in an empty state and then programmed later using a PROM programmer or burner.
- **EPROM:** EPROM stands for Erasable Programmable Read-Only Memory, and as the name suggests, data stored in an EPROM can be erased and the EPROM reprogrammed. Erasing an EPROM involves removing it from the computer and exposing it to ultraviolet light before re-burning it.
- **EEPROM:** EEPROM stands for Electrically Erasable Programmable Read-Only Memory, and the distinction between EPROM and EEPROM is that the latter can be erased and written to by the computer system it is installed in. In that sense EEPROM is not strictly read-only. However, in many cases the write process is slow, so it is normally only done to update program code such as firmware or BIOS code on an occasional basis.

Secondary Memory Types

Secondary memory comprises many different storage media which can be directly attached to a computer system. These include:

Hard Disk Drives (HDD), Solid State Drives (SSD),
Optical (CD or DVD), Drives, Tape drives

Differences between RAM and ROM

ROM	RAM
• Non-volatile. • Fast to read. • Usually used in small quantities. • Cannot be written too quickly. • Used to store boot instructions or firmware. • Relatively expensive per megabyte stored compared to RAM.	• Volatile. • Fast to read and write. • Used as system memory to store data (including program code) that the CPU needs to process imminently. • Relatively cheap per megabyte stored compared to ROM, but relatively expensive compared to secondary memory.

Short Questions

1. What is computer?
2. Write short note
   - RAM
   - ROM
   - Hybrid computer
   - CPU
   - ALU
   - Cache Memory
3. What is primary memory?
4. Define Digital Computer.
5. Different between RAM and ROM.
6. What is secondary memory?

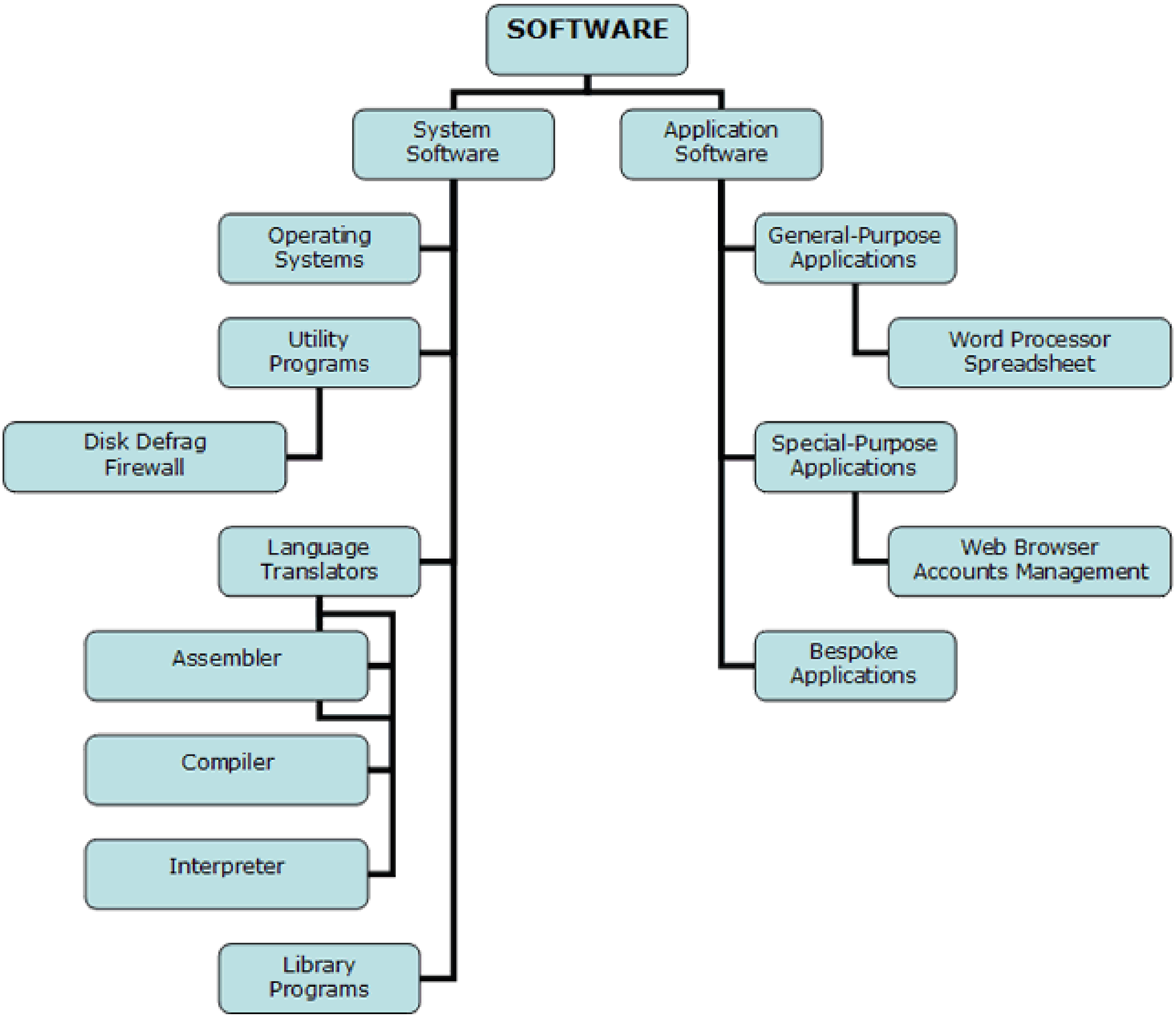
Long Questions

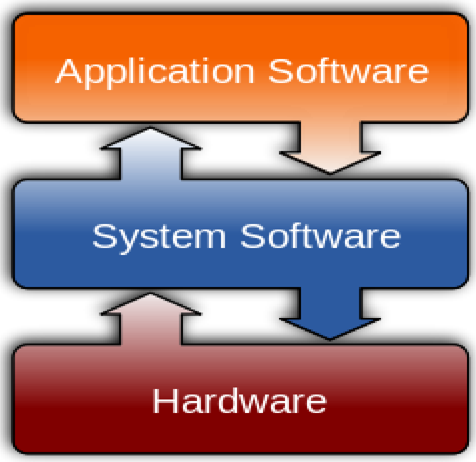
1. What is computer?what are it's characteristics.
2. Explain the classification of computer.
3. What is computer? Explain any five major area of application of computer.
4. Discuss the basic organization of the computer with diagram.
5. Discuss about the input devices and out put devices of the computer.
6. Explain all the generation and key features about the generation.

CH-2 COMPUTER SOFTWARE

2.1 CLASSIFICATION OF SOFTWARE

The following diagram shows the way we classify software.





2.2 System Software

We use the term **System Software** for software which is primarily used to operate the hardware.

Operating Systems:

- The operating system is the software that allows you to operate the hardware. The programs that we want to execute, the applications that we want to use all require a platform on which to execute. That platform is provided by the operating system.

Utility Programs:

- Some utility programs are bundled in with operating system software these days.
- Utility programs tend to perform specific tasks related to the management of hardware.
- Examples of utility programs include compression programs, formatters, defragmenters and other disk management tools.

Library Programs:

- Library programs are compiled libraries of commonly-used routines**Language Translators:**

There are 3 main categories of translator used.

- **Assembler:** An assembler is a program that translates the mnemonic codes used in assembly language into the bit patterns that represent machine operations.
- Assembly language has a one-to-one equivalence with machine code, each assembly statement can be converted into a single machine operation.
- **Compiler:** A compiler turns the source code that you write in a high-level language into object code (machine code) that can be executed by the computer.
- The compiler is a more complex beast than the assembler. It may require several machine operations to represent a single high-level language statement. As a result, compiling may well be a lengthy process with very large programs.
- **Interpreter:** Interpreters translate the source code at run-time. The interpreter translates statements one-at-a-time as the program is executed.Interpreters are often used to execute high-level language programs whilst they are being developed since this can be quicker than compiling the entire program.
- The program would be compiled when it is complete and ready to be released.
- Interpreters are also used with high-level scripting languages like PHP, JavaScript and many more.

- These instructions are not compiled and have to be interpreted either by the browser (in the case of JavaScript) or by interpreters on the server (in the case of PHP).

2.3 Application Software

- Application software tends be used for the tasks that have some relationship to the world outside of the computer.
- For example, you might use a word processor to write a letter or an essay.

General-Purpose Software:

Software is general-purpose if it can be used for lots of different tasks. You can use a word processor to write letters, memos, essays, instructions, notes, faxes, invoices and lots more.

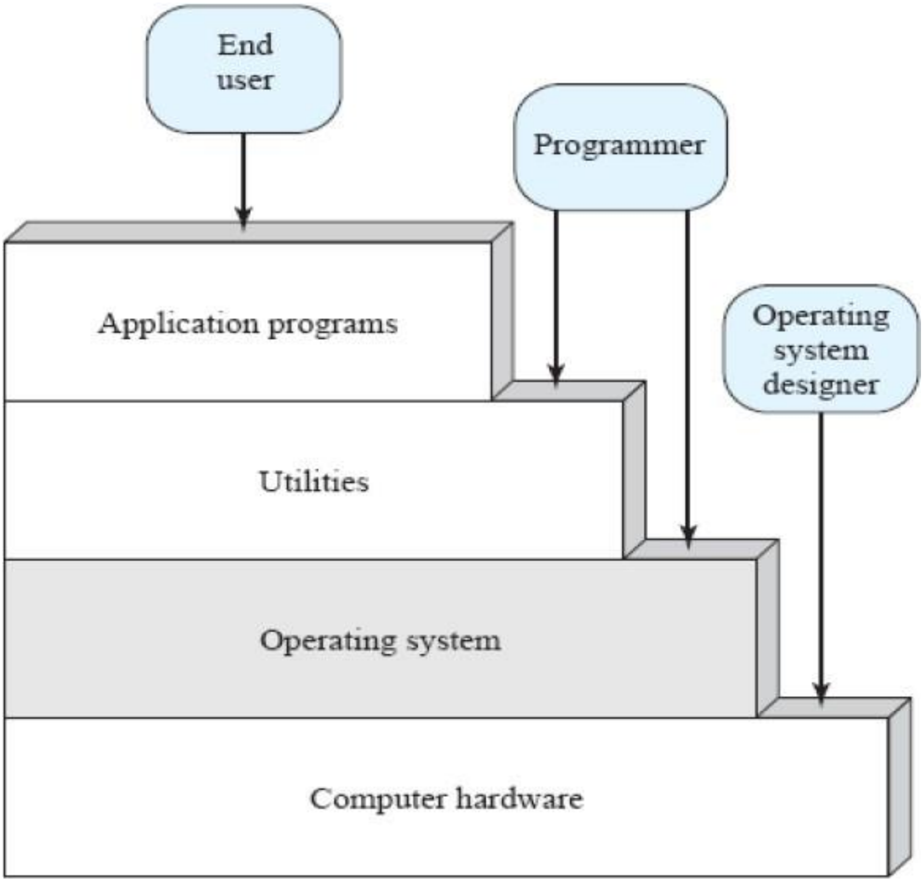
Special-Purpose Software:

- This software performs a single specific task.
- This task might be complex like payroll calculation, stock control etc. but will be based on a single task.
- These days, web browsers can contain a lot of features.
- They are still primarily focused on a single task, rendering web pages and so the web browser is special-purpose.

2.4 OPERATING SYSTEM OBJECTIVES AND FUNCTIONS:

- An Operating System exploits the hardware resources of one or moreprocessors to provide a set of services to system users.
- The OS also managessecondary memory and I/O devices on behalf of its users. So, it is necessary to havesome understanding some of computer system hardware.
- An OS is a program that controls the execution of application programs andacts as an interface between applications and the computer hardware. It can bethought of as having three objectives:

- Convenience: An OS makes a computer more convenient to use.
- Efficiency: An OS allows the computer system resources to be used in anefficient manner.
- Ability to evolve: An OS should be constructed in such a way as to permit theeffective development, testing, and introduction of new system functions withoutinterfering with service.

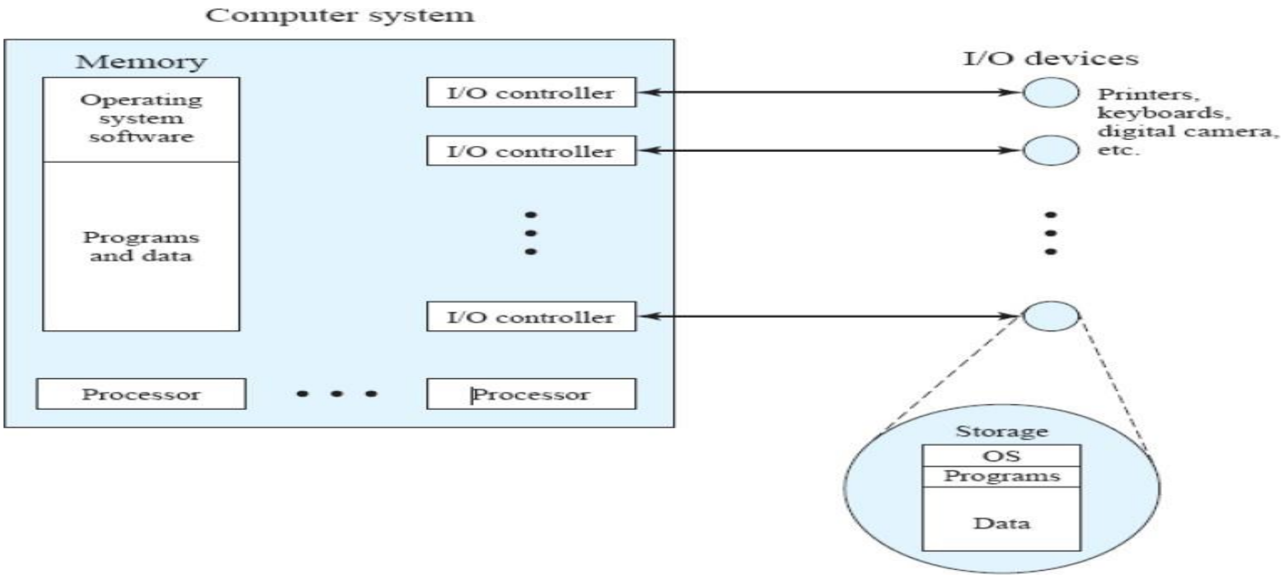


<u>OS typically provides services in the following areas:</u>

- Program development: The OS provides a variety of facilities and services,such as editors and debuggers, to assist the programmer in creating programs.
- Program execution: A number of steps need to be performed to execute aprogram.Instructions and data must be loaded into main memory, I/O devicesand files must be initialized, and other resources must be prepared.
- Access to I/O devices: Each I/O device requires its own peculiar set ofinstructions or control signals for operation.
- Controlled access to files: For file access, the OS must reflect a detailedunderstanding of not only the nature of the I/O device (disk drive, tape drive) butalso the structure of the data contained in the files on the storage medium.
- System access: For shared or public systems, the OS controls access to the system as a whole and to specific system resources. The access function must provide protection of resources and data from unauthorized users and must resolve conflicts for resource contention.
- Error detection and response: A variety of errors can occur while a computer system is running.
- Accounting: A good OS will collect usage statistics for various resources andmonitor performance parameters such as response time.

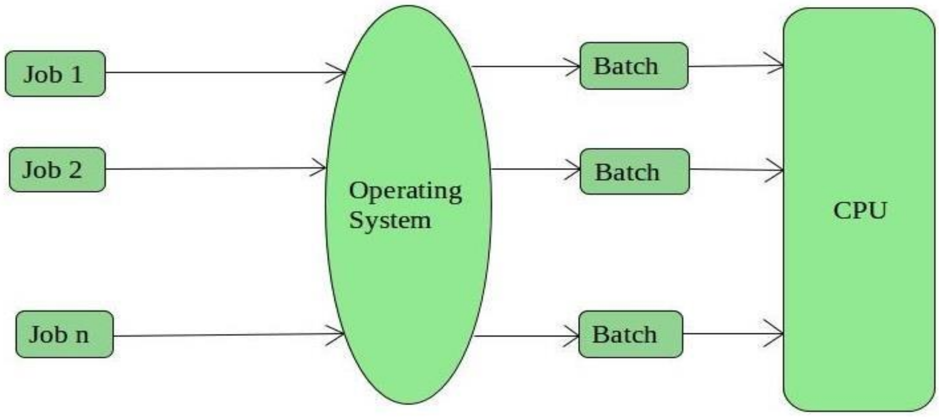
Operating System as a Resource Manager:

- A computer is a set of resources for the movement, storage, and processingof data and for the control of these functions.
- The OS is responsible for managingthese resourcesThe OS functions in the same way as ordinary computer software; that is, it is aprogram or suite of programs executed by the processor.
- The OS frequently relinquishes control and must depend on the processor toallow it to regain control.



2.6 TYPES OF OPERATING SYSTEMS

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



Advantages of Batch Operating System:

- It is very difficult to guess or know the time required by any job to complete. Processors of the batch systems know how long the job would be when it is in queue.

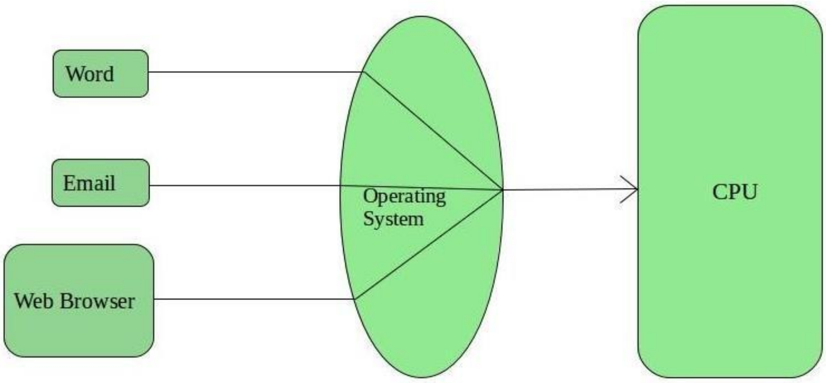
- Multiple users can share the batch systems.
- The idle time for batch system is very less.
- It is easy to manage large work repeatedly in batch systems.

Disadvantages of Batch Operating System:

- The computer operators should be well known with batch systems.
- Batch systems are hard to debug.
- It is sometime 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.

2. Time-Sharing Operating Systems: Each task is given some time to execute, so that all the tasks work smoothly. Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems. The task can be from single user or from different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.



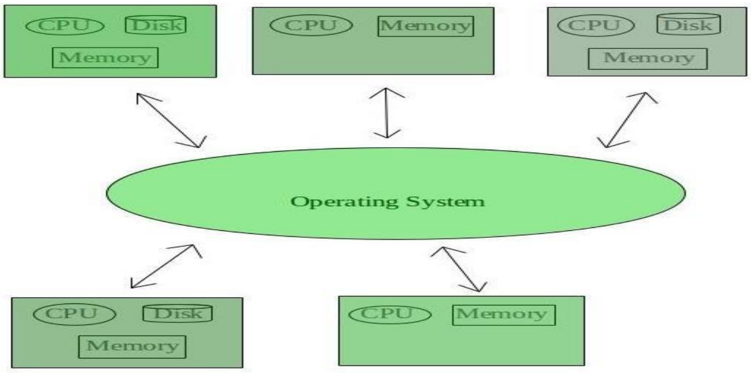
Advantages of Time-Sharing OS:

- Each task gets an equal opportunity.
- Less chances of duplication of software.
- CPU idle time can be reduced.

Disadvantages of Time-Sharing OS:

- Reliability problem.
- One must have to take care of security and integrity of user programs and data.
- Data communication problem.

Examples of Time-Sharing OSs are: Multics, Unix etc.



3. Distributed Operating System: These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all-over the world and, that too, with a great pace. Various autonomous interconnected computers communicate each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred as loosely coupled systems or distributed systems. These system's processors differ in size and function. The major benefit of working with these types of operating system is that it is always possible that one user can access the files or software which are not actually present on his system but on some other system connected within this network i.e., remote access is enabled within the devices connected in that network.

Advantages of Distributed Operating System:

- Failure of one will not affect the other network communication, as all systems are independent from each other.
- Electronic mail increases the data exchange speed.
- Since resources are being shared, computation is highly fast and durable.
- Load on host computer reduces.
- These systems are easily scalable as many systems can be easily added to the network.
- Delay in data processing reduces.

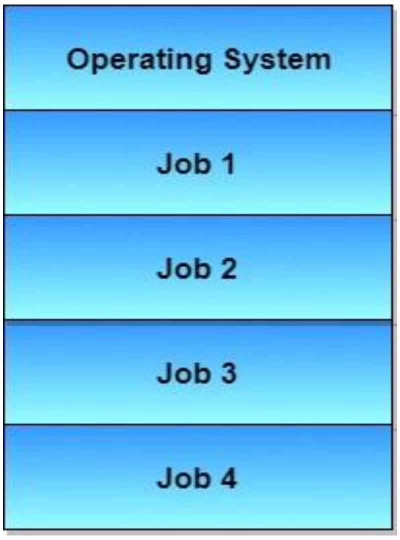
Disadvantages of Distributed Operating System:

- Failure of the main network will stop the entire communication.
- To establish distributed systems the language which are used are not well defined yet.
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet.

Examples of Distributed Operating System are: LOCUS etc.

4. Multiprogramming Operating Systems:

- In this the operating system picks up and begins to execute one of the jobs from memory.Once this job needs an I/O operation operating system switches to another job (CPU and OS always busy).Jobs in the memory are always less than the number of jobs on disk(Job Pool).If several jobs are ready to run at the same time, then the system chooses which one to run through the process of CPU Scheduling.



5. Multiprocessor Systems:A Multiprocessor system consists of several processors that share a common physical memory. Multiprocessor system provides higher computing power and speed. In multiprocessor system all processors operate under single operating system. Multiplicity of the processors and how they do act together are transparent to the others.

Advantages of Multiprocessor Systems:

- Enhanced performance.
- Execution of several tasks by different processors concurrently, increases the system's throughput without speeding up the execution of a single task.
- If possible, system divides task into many subtasks and then these subtasks can be executed in parallel in different processors. Thereby speeding up the execution of single tasks.

6. Real Time operating System:

- A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment.
- The time taken by the system to respond to an input and display of required updated information is termed as the response time. So, in this method, the response time is very less as compared to online processing.
- Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application.
- A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

OPERATING SYSTEM COMPARISON

- An operating system or OS, is a fundamental component of a computer system that manages activities and resources on the machine.
- As a host application, it handles the operations of hardware devices which makes it easy for the associated programs to function.
- Nearly every computer including desktops, laptops, supercomputers, hand-held and even video game consoles use some type of operating system.

Comparison Chart:

BASIS FOR COMPARISON	LINUX	WINDOWS
Cost	Free of cost	Expensive
Open source	Yes	No
Customizable	Yes	No
Security	More secure	Vulnerable to viruses and malware attacks.
Booting	Either primary or logical	Only primary partition.

	partition.	
Separation of the directories using	Forward slash	Back slash
File names	Case sensitive	Case insensitive
File system	EXT2, EXT3, EXT4, Reisers FS, XFS and JFS	FAT, FAT32, NTFS and ReFS
Type of kernel used	Monolithic kernel	Microkernel
Efficiency	Effective running efficiency	Lower than Linux

COMPUTER VIRUSES

A computer virus is a malware program that is written intentionally to gain access to a computer without its owner's permission.

These kinds of programs are primarily written to steal or destroy computer data. Most systems catch viruses due to program bugs, the vulnerability of operating systems, and poor security practices.

Types of Computer Viruses:

1. Boot Sector Virus: Boot Sector virus infects the storage device's master boot record (MBR). Any media, whether it is bootable or not can trigger this virus. These viruses inject their code to hard disk's partition table. It then gets into the main memory once the computer restarts.Booting problems, unstable system performance and inability to locate hard disk are common issues that may arise after getting infected. However, it has become rare since the decline in floppy disks. Modern operating systems come with an inbuilt boot sector safeguard which makes it difficult to find the MBR.

Can affect: Any file after getting into the main memory

Examples: Form, Disk Killer, Stone virus, Polyboot.B

Protection: Make sure that the disk you are using is write-protected. Do not start/restart the computer with unknown external disks connected.

2. Direct Action Virus: This virus quickly gets into the main memory, infects all programs/files/folder defined in Autoexec.bat path and then deletes itself. It can also destroy the data present in harddisk or USB attached to the computer.

While these viruses are found in hard disk's root directory, they are capable of changing location on every execution. In most cases, they don't delete system files but alter the system's overall performance.

Can affect: All .exe and .com file extension

Example: VCL.428, created by the Virus Construction Laboratory

Protection: Use antivirus scanner. Direct action virus is easy to detect and all infected files can be restored completely.

3. **Overwrite Virus:**Overwrite viruses are very dangerous. They have affected a wide range of operating system including Windows, DOS, Macintosh, and Linux. They simply delete the data (partially or completely) and replace the old code with their own.They replace the file content without changing its size. It is easy to detect as the original program stops working. Once the file gets infected, it can't be restored and you will end up losing all data.
Can affect: Any file
Examples: Grog.377, Grog.202/456, Way, Loveletter
Protection: The only way to get rid of this virus is to delete all the infected files, so it's better to keep your antivirus program updated, especially if you are using Windows.
4. **Web Scripting Virus:**A web scripting virus breaches web browser security and allows attackers to inject client-side scripting into the web page. They propagate quite faster than other conventional viruses.It is used to attack large sites like social networking, user review or email. It has the potential to send a large amount of spam, fraud activity, and damage files on sever.
Can affect: Any web page by injecting hidden code in header, footer or root access file.
Examples:DDos, JS.fornight
Protection: Use malicious software removal tool in Windows, disable scripts, use cookie security or install real-time protection software for the web browser.
5. **Directory Virus:**Directory Virus (also known as Cluster virus) infects the file by changing the DOS directory information. In this case, DOS points to the virus code rather than pointing to the original program.When you run a program, DOS first loads and executes the virus code before running the actual program code. It becomes very difficult to locate the original file after getting infected.
Can affect: The entire program in the directory
Example: Dir-2
Protection: Install the antivirus to relocate the misplaced files.

6. **Polymorphic Virus:**The polymorphic virus encodes themselves using different encryption keys and algorithms each time they infect a program or create a copy of itself. Because of different encryption keys, it becomes very difficult for the antivirus software to find them. In other words, it is a self-encrypted virus which is designed to avoid detection by scanners.
Can affect: Any file
Examples: Whale, Simile, SMEG engine, 1260
Protection: Install advanced, high-end antivirus
7. **Memory Resident Virus:**These viruses live in primary memory (RAM) and get activated whenever you switch on the computer. They affect all files currently

running on the desktop. Basically, it allocates memory, blocks original scripts, and runs its own code when any program is executed.

Can affect: Any file running on PC and files that are being copied or renamed.

Examples: Randex, Meve, CMJ

Protection: Install strong antivirus software

8. Macro Virus: There are a few software such as a word processor that allows a macro program to embed in documents. This virus is written in the macro language, so it may run automatically when the document is opened and it can easily spread to other files too.It depends on the application rather than the operating system. They are generally hidden in documents that are more likely shared via email.

Can affect: .mdb, .PPS, .Doc, .XLs files

Examples: Bablas, Concept and Melissa virus

Protection: Disable macros and don't open emails from unknown sources. Alternatively, you can install modern antivirus software that can detect macro virus easily.

9. Companion Virus: Companion Viruses were more popular during the MS-DOS era. Unlike traditional viruses, they do not modify the existing file. It creates a copy of a file with a different extension (usually .com) which runs in parallel with the actual program.For example, if there is a file named abc.exe, this virus will create another hidden file named abc.com. And when the system calls a file 'abc', the .com (higher priority extension) runs before the .exe extension. It can perform malicious steps such as deleting the original files.

Can affect: All .exe files

Examples: Stator, Terrax.1096

Protection: Can be easily detected because of the presence of additional .com file. Install reliable antivirus software and avoid downloading attachments of unsolicited emails.

10. Multipartite virus: The Multipartite virus infects and spreads in multiple ways depending on the operating system. They usually stay in memory and infect the hard disk.Once it gets into the system, it infects all drives by altering applications' content. You will soon start noticing performance lag and low virtual memory available for user applications.

Can affect: Files and boot sector

Examples: Ghostball, Invader

Protection: Clean boot sector and entire disk before reloading the data. Do not open attachments from a non-trusted internet source and install quality antivirus software.

11. FAT Virus:FAT stands for file allocation table which is a section of storage disk that is used to store information, such as the location of all files, total storage capacity, available space, used space etc. A FAT virus alters the index and makes it impossible for the computer to allocate the file. It is powerful enough to force you to format the whole disk.

Can affect: Any file

Example: The link virus

Protection: Avoid downloading files from non-trusted sources, especially those identified as "attack site" by browser or search engine. Use robust antivirus software.

12. Trojan Horse: Trojan Horse (or Trojan) is a non-replicating type of malware that looks legitimate. Users are typically tricked into loading and executing it on the system. It can destroy/modify all the files, crash the computer, modify the registry, and is strong enough to give hackers remote access to your PC.

Examples: ProRat, ZeroAccess, Beast, Netbus, Zeus

Protection: Use reliable high-end antivirus software and update it regularly.

13. Worm: Worm is a standalone malware program that replicates itself in order to spread to other computers. It relies on networks (mostly emails) and security holes to travel from one system to another. Unlike viruses, it overloads the network by replicating or sending too much data (overusing bandwidth), forcing the hosts to shut down the server.

Example: Code red, ILOVEYOU, Morris, Nimda, Sober, WANK

Protection: Use antivirus and anti-spyware software.

14. Logic Bombs: They are not a virus but inherently malicious like worms and viruses. It is a piece of code intentionally inserted (hidden) into a software tool. This code is executed after certain criteria are met.For example, a cracker can insert a Keylogger code inside any web browser extension. The code gets activated whenever you visit a login page and then captures the keystrokes that you entered while filling your username and passwords. These malicious codes are known as Logic Bombs.

APPLICATIONS OF DIFFERENT DOMAIN

Business:A computer has high speed of calculation, diligence, accuracy, reliability, or versatility which has made it an integrated part in all business organizations.

Computer is used in business organizations for −

- Payroll calculations
- Budgeting
- Sales analysis
- Financial forecasting
- Managing employee database
- Maintenance of stocks, etc.

Banking: Today, banking is almost totally dependent on computers.

Banks provide the following facilities −

- Online accounting facility, which includes checking current balance, making deposits and overdrafts, checking interest charges, shares, and trustee records.
- ATM machines which are completely automated are making it even easier for customers to deal with banks.

Insurance: Insurance companies are keeping all records up-to-date with the help of computers. Insurance companies, finance houses, and stock broking firms are widely using computers for their concerns.Insurance companies are maintaining a database of all clients with information showing −

- Procedure to continue with policies
- Starting date of the policies
- Next due instalment of a policy
- Maturity date
- Interests due
- Survival benefits
- Bonus

Education: The computer helps in providing a lot of facilities in the education system.

- The computer provides a tool in the education system known as CBE (Computer Based Education).
- CBE involves control, delivery, and evaluation of learning.
- Computer education is rapidly increasing the graph of number of computer students.
- There are a number of methods in which educational institutions can use a computer to educate the students.
- It is used to prepare a database about performance of a student and analysis is carried out on this basis.

Marketing: In marketing, uses of the computer are following −

- Advertising − With computers, advertising professionals create art and graphics, write and revise copy, and print and disseminate ads with the goal of selling more products.
- Home Shopping − Home shopping has been made possible through the use of computerized catalogues that provide access to product information and permit direct entry of orders to be filled by the customers.

Healthcare: Computers have become an important part in hospitals, labs, and dispensaries. They are being used in hospitals to keep the record of patients and medicines. It is also used in scanning and diagnosing different diseases. ECG, EEG, ultrasounds and CT scans, etc. are also done by computerized machines.

Following are some major fields of health care in which computers are used-

- Diagnostic System − Computers are used to collect data and identify the cause of illness.
- Lab-diagnostic System − All tests can be done and the reports are prepared by computer.

- Patient Monitoring System − These are used to check the patient's signs for abnormality such as in Cardiac Arrest, ECG, etc.
- Pharma Information System − Computer is used to check drug labels, expiry dates, harmful side effects, etc.
- Surgery − Nowadays, computers are also used in performing surgery.

Engineering Design: Computers are widely used for Engineering purpose.

One of the major areas is CAD (Computer Aided Design) that provides creation and modification of images. Some of the fields are −

- Structural Engineering − Requires stress and strain analysis for design of ships, buildings, budgets, airplanes, etc.
- Industrial Engineering − Computers deal with design, implementation, and improvement of integrated systems of people, materials, and equipment.
- Architectural Engineering − Computers help in planning towns, designing buildings, determining a range of buildings on a site using both 2D and 3D drawings.

Military: Computers are largely used in defence. Modern tanks, missiles, weapons, etc. Military also employs computerized control systems. Some military areas where a computer has been used are −

- Missile Control
- Military Communication
- Military Operation and Planning
- Smart Weapons

Communication: Communication is a way to convey a message, an idea, a picture, or speech that is received and understood clearly and correctly by the person for whom it is meant. Some main areas in this category are −

- E-mail
- Chatting
- Usenet
- FTP
- Telnet
- Video-conferencing

Government: Computers play an important role in government services. Some major fields in this category are −

- Budgets
- Sales tax department
- Income tax department
- Computation of male/female ratio
- Computerization of voters' lists
- Weather forecasting

Short Questions

1. What is FTP?
2. What is Software?
3. What is viruses?
4. What is system software?
5. Define OS.
6. Define DOS.

Write Short note

- Compiler
- Interpreter
- Programming language
- Application software
- Windows

Long Questions

1. Explain different types of OS, and discuss about the batch process Operating system.
2. Difference between compiler and interpreter.
3. What is Computer virus ? Discuss the different computer viruses.
4. Discuss the prevention and protectioncomputer virus.

CH-3 COMPUTER NETWORK & INTERNET

NETWORKING CONCEPTS

A computer networkconsists of two or more computers that are linked in order to share resources such as printers, exchange files and allow communication.The size of computer networks may vary. The Internet is an example of a computer network that spreads all across the world. The Internet is also referred to as the worldwide network of computers and it is growing at a rapid rate.

Need for computer networks: Nowadays, computer networks are a vital part of any organisation. Some of the advantages of computer networks are:

Resource Sharing: All computers in a network can share resources such as printers, fax machines, modems and scanners.

File Sharing and Remote Database Access: A computer network allows sharing of files and access to remote database. We can easily access the files stored on various computers on a network. Also, networking allows many people to work simultaneously on the data stored in a database.

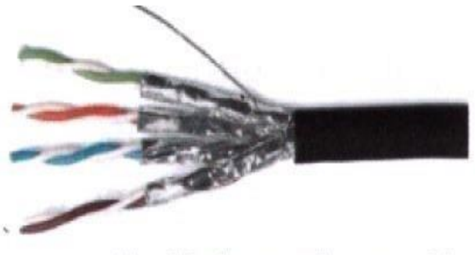
Ease of Communication: Computer networks allow people to communicate through emails and instant messaging facilities. This makes the transmission of information easier, more efficient and less expensive.

CONNECTING MEDIA

Computers must be connected to each other to form a network. Computers can be connected using wires/cables or they can be connected in a wireless manner.

Wired Transmission Media: There are various types of cables that can be used for setting up a network. Some of them are discussedhere.

- Twisted Pair Cable: It consists of a pair of insulated wires twisted together. The use of two wirestwisted around each other helps to reduce disturbances in the signals.The twisted pair cable is often used in two or more pairs, all within a single cable. Twisted pair cablingcomes in two varieties—shielded (Shielded Twisted Pair or STP) and unshielded (Unshielded TwistedPair or UTP). UTP cable is the most commonly used cable in computer networking.The twisted pair cable is often used in two or more pairs, all within a single cable. Twisted pair cablingcomes in two varieties—shielded (Shielded Twisted Pair or STP) and unshielded (Unshielded TwistedPair or UTP). UTP cable is the most commonly used cable in

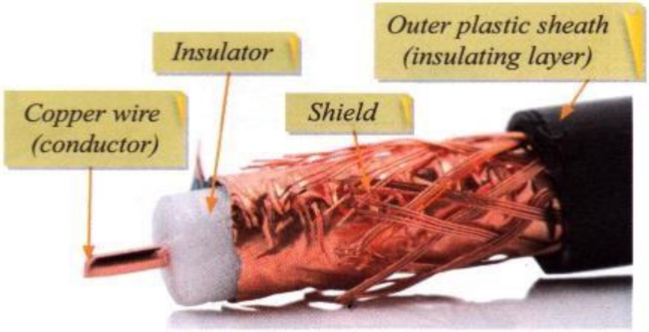


Shielded twisted pair cable

Unshielded twisted pair cable

computer networking.

- **Coaxial Cable (coax):** Coaxial cable is an electrical cable with a conductor at its centre. The inner conductor is surrounded by a tubular insulating layer. The insulating layer is surrounded by a conductive layer called the shield, which is finally covered with a thin insulating layer on the outside.



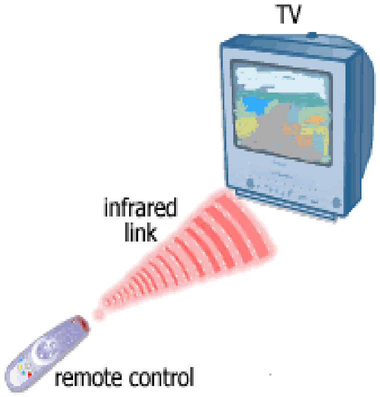
Coaxial wire

- **Optical Fibre Cable:** Optical fibre cable consists of a central glass core surrounded by several layers of protective material. It transmits data in the form of light rather than electronic signals, thus eliminating the problem of electrical interference. Fibre optic cable is expensive as compared to coaxial and twisted pair cables but can transmit signals over much longer distances. It also has the capability to carry data at a very high speed.

Optical fibre cable

Wireless Transmission Media: In wireless networks, data is transmitted without wires. Some of the ways in which wireless networks may be set up are as follows.

- **Infrared:** The infrared communication range of the devices communicating through infrared waves is very limited. Infrared waves cannot penetrate walls or other obstructions and so there should be no physical barrier between the communicating devices. The communication between a TV set and a remote control happens through infrared waves. Infrared mouse and keyboard are other examples of devices that make use of infrared waves for data transmission.



- **Microwave Transmission:** Microwave communications are unidirectional. They can be used for terrestrial communication (on ‘the surface of the earth) or for satellite communication. Microwave propagation is line-of-sight communication. So, when used for terrestrial communication, the towers with antennas mounted on them need to be in direct sight of each other. The antennas are usually located at substantial heights above the ground level to extend the range between antennas and to be able to

Microwave antenna

transmit over obstacles. You must have noticed high towers with microwave antennas in your city.Microwaves can pass through the earth's atmosphere easily and can be used to transmit information between satellites and the earth's base station.

- **Radio-wave Transmission:** Radio-wave communications are omnidirectional, which means that they travel in all directions from the source, so that the transmitter and receiver do not have to be carefully aligned physically. Radio waves are easy to generate, can travel long distances and penetrate through buildings easily. So, they are widely used for communication both indoors and outdoors. However, at all frequencies, radio waves are subject to interference from motors and other electrical equipment.

Radiowave antenna

- **Bluetooth technology:** is used for exchanging data over short distances using radio waves. This technology uses low power, has a short range (30 feetapprox) and medium transmission speed. Bluetooth technology can be used to transfer songs or pictures between two mobile phones or a Bluetooth headset can be used with a mobile phone.

Bluetooth technology

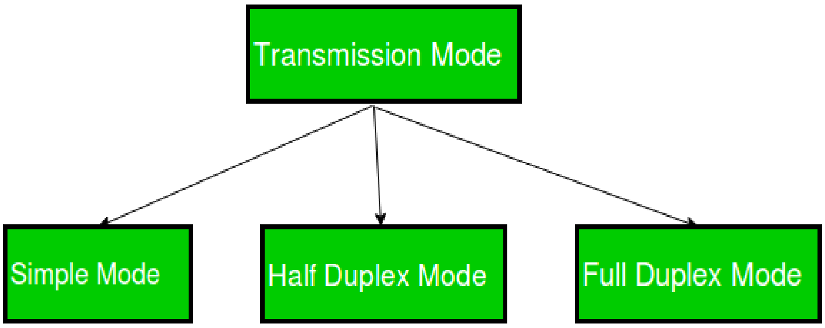
- **Wi-Fi technology:** also makes use of radio waves to transmit and receive data. This technology requires more energy but enables the signal to go farther (300 feet approx.) with a faster rate of transmission. This technology is used to set up networks in which a computer's wireless adapter translates the data into a radio signal and transmits it. A wireless router receives the signal, decodes it and sends it to the Internet using a wired connection.

WiFi technology

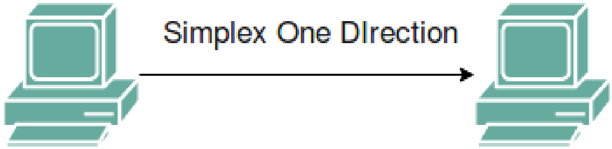
TRANSMISSION MODES IN COMPUTER NETWORKS

Transmission mode means transferring of data between two devices. It is also known as communication mode. Buses and networks are designed to allow communication to occur between individual devices that are interconnected. There are three types of transmission mode:-

- Simplex Mode
- Half-Duplex Mode
- Full-Duplex Mode

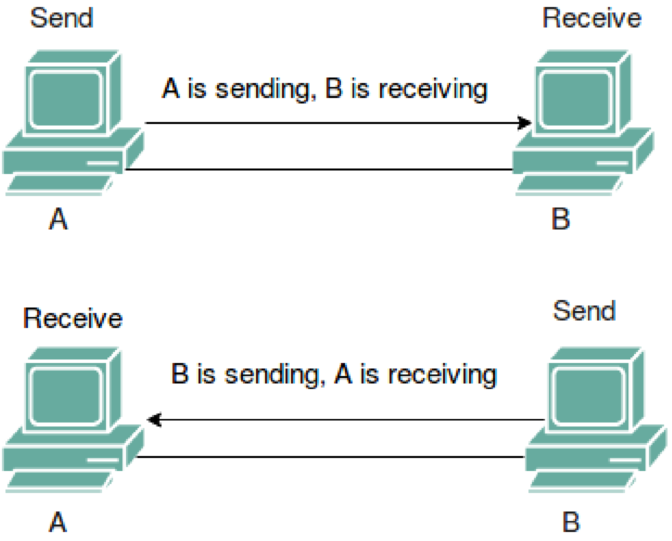


Simplex Mode: In Simplex mode, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit, the other can only receive. The simplex mode can use the entire capacity of the channel to send data in one direction.



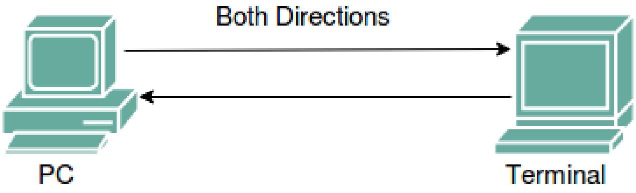
Example: Keyboard and traditional monitors. The keyboard can only introduce input, the monitor can only give the output.

Half-Duplex Mode: In half-duplex mode, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa. The half-duplex mode is used in cases where there is no need for communication in both direction at the same time. The entire capacity of the channel can be utilized for each direction.



Example: Walkie- talkie in which message is sent one at a time and messages are sent in both the directions.

Full-Duplex Mode: In full-duplex mode, both stations can transmit and receive simultaneously. In full-duplex mode, signals going in one direction share the capacity of the link with signals going in other direction, this sharing can occur in two ways:



- Either the link must contain two physically separate transmission paths, one for sending and other for receiving.
- Or the capacity is divided between signals travelling in both directions.

Full-duplex mode is used when communication in both direction is required all the time. The capacity of the channel, however must be divided between the two directions.

Example: Telephone Network in which there is communication between two persons by a telephone line, through which both can talk and listen at the same time.

PROTOCOLS

Just the way we follow certain rules while communicating or travelling on the road, similarly rules or protocols have to be followed for effective network communication. Protocol is a set of rules used by computers on a network to communicate with each other. Some examples of protocols are:

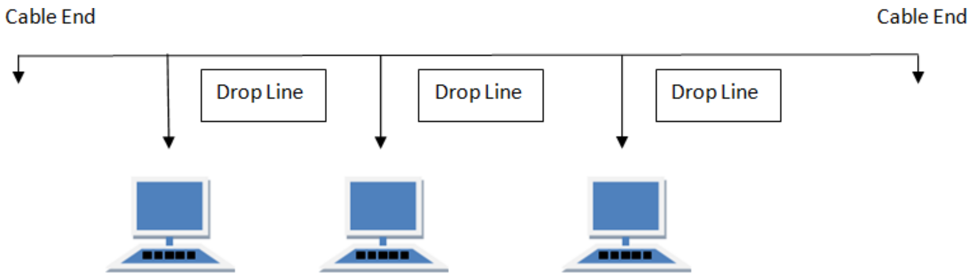
HTTP (Hyper Text Transfer Protocol): It is a protocol used between a web server and a web browser for transferring HTML pages.

TCP/IP (Transmission Control Protocol/Internet Protocol): TCP is a protocol that is used along with the IP to send data over the Internet. The information is transmitted across the Internet in the form of bundles called TCP is responsible for dividing the data into packets before they are sent and for reassembling the packets when they arrive at the destination. IP is a set of specifications that determines the best route for the packets across the Internet so that the packets reach their destination address.

NETWORK TOPOLOGY

Network Topology is the schematic description of a network arrangement, connecting various nodes(sender and receiver) through lines of connection.

1. BUS Topology: Bus topology is a network type in which every computer and network device is connected to a single cable. When it has exactly two endpoints, then it is called Linear Bus topology.



Features of Bus Topology:

- It transmits data only in one direction.
- Every device is connected to a single cable

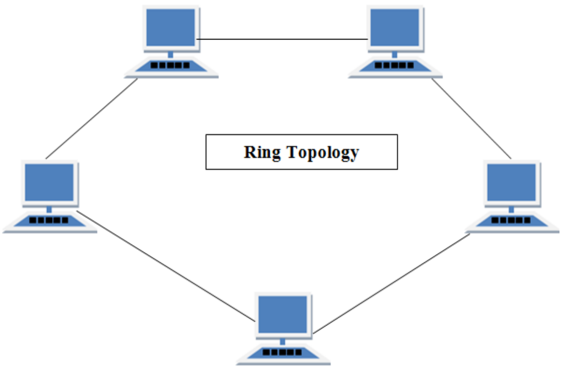
Advantages of Bus Topology:

- It is cost effective.
- Cable required is least compared to other network topology.
- Used in small networks.
- It is easy to understand.
- Easy to expand joining two cables together.

Disadvantages of Bus Topology:

- Cables fails then whole network fails.
- If network traffic is heavy or nodes are more the performance of the network decreases.
- Cable has a limited length.
- It is slower than the ring topology.

2. RING Topology: It is called ring topology because it forms a ring as each computer is connected to another computer, with the last one connected to the first. Exactly two neighbours for each device.



Features of Ring Topology:

- A number of repeaters are used for Ring topology with large number of nodes, because if someone wants to send some data to the last node in the ring topology with 100 nodes, then the data will have to pass through 99 nodes to reach the 100th node. Hence to prevent data loss repeaters are used in the network.

- The transmission is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called Dual Ring Topology.
- In Dual Ring Topology, two ring networks are formed, and data flow is in opposite direction in them. Also, if one ring fails, the second ring can act as a backup, to keep the network up.
- Data is transferred in a sequential manner that is bit by bit. Data transmitted, has to pass through each node of the network, till the destination node.

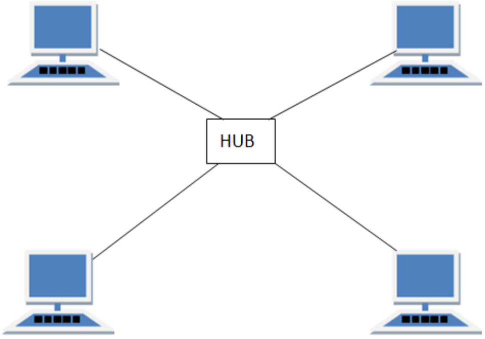
Advantages of Ring Topology:

- Transmitting network is not affected by high traffic or by adding more nodes, as only the nodes having tokens can transmit data.
- Cheap to install and expand

Disadvantages of Ring Topology:

- Troubleshooting is difficult in ring topology.
- Adding or deleting the computers disturbs the network activity.
- Failure of one computer disturbs the whole network.

3. STAR Topology: In this type of topology all the computers are connected to a single hub through a cable. This hub is the central node and all others nodes are connected to the central node.



Features of Star Topology:

- Every node has its own dedicated connection to the hub.
- Hub acts as a repeater for data flow.
- Can be used with twisted pair, Optical Fibre or coaxial cable.

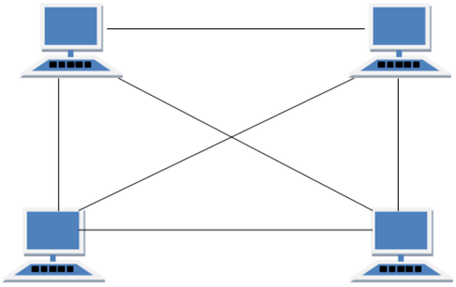
Advantages of Star Topology:

- Fast performance with few nodes and low network traffic.
- Hub can be upgraded easily.
- Easy to troubleshoot.
- Easy to setup and modify.
- Only that node is affected which has failed, rest of the nodes can work smoothly.

Disadvantages of Star Topology:

- Cost of installation is high.
- Expensive to use.
- If the hub fails then the whole network is stopped because all the nodes depend on the hub.
- Performance is based on the hub that is it depends on its capacity

4. MESH Topology: It is a point-to-point connection to other nodes or devices. All the network nodes are connected to each other. Mesh has n(n-1)/2 physical channels to link n devices.There are two techniques to transmit data over the Mesh topology, they are: Routing and Flooding



MESH Topology (Routing): In routing, the nodes have a routing logic, as per the network requirements. Like routing logic to direct the data to reach the destination using the shortest distance. Or, routing logic which has information about the broken links, and it avoids those node etc. We can even have routing logic, to re-configure the failed nodes.

MESH Topology (Flooding): In flooding, the same data is transmitted to all the network nodes, hence no routing logic is required. The network is robust, and the its very unlikely to lose the data. But it leads to unwanted load over the network.

Types of Mesh Topology:

- Partial Mesh Topology: In this topology some of the systems are connected in the same fashion as mesh topology but some devices are only connected to two or three devices.
- Full Mesh Topology: Each and every nodes or devices are connected to each other.

Features of Mesh Topology:

- Fully connected.
- Robust.
- Not flexible.

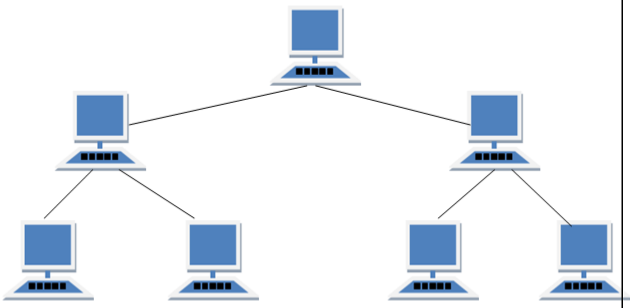
Advantages of Mesh Topology:

- Each connection can carry its own data load.
- It is robust.
- Fault is diagnosed easily.
- Provides security and privacy.

Disadvantages of Mesh Topology:

- Installation and configuration is difficult.
- Cabling cost is more.
- Bulk wiring is required.

5. TREE Topology: It has a root node and all other nodes are connected to it forming a hierarchy. It is also called hierarchical topology. It should at least have three levels to the hierarchy.



Features of Tree Topology:

- Ideal if workstations are located in groups.
- Used in Wide Area Network.

Advantages of Tree Topology:

- Extension of bus and star topologies.
- Expansion of nodes is possible and easy.

- Easily managed and maintained.
- Error detection is easily done.

Disadvantages of Tree Topology:

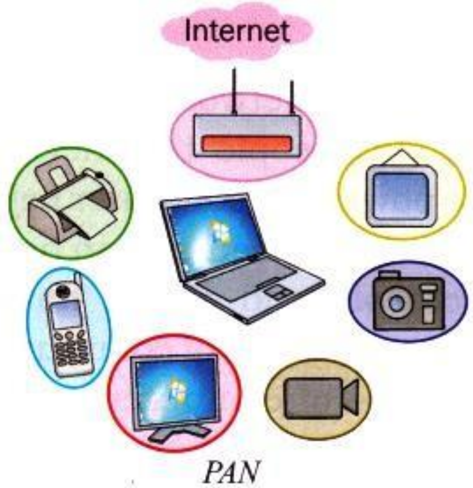
- Heavily cabled.
- Costly.
- If more nodes are added maintenance is difficult.
- Central hub fails, network fails.

6. HYBRID Topology: It is two different types of topologies which is a mixture of two or more topologies. For example, if in an office in one department ring topology is used and in another star topology is used, connecting these topologies will result in Hybrid Topology (ring topology and star topology).

TYPES OF COMPUTER NETWORKS

The following are the types of networks based on the geographical area covered or scale of the network.

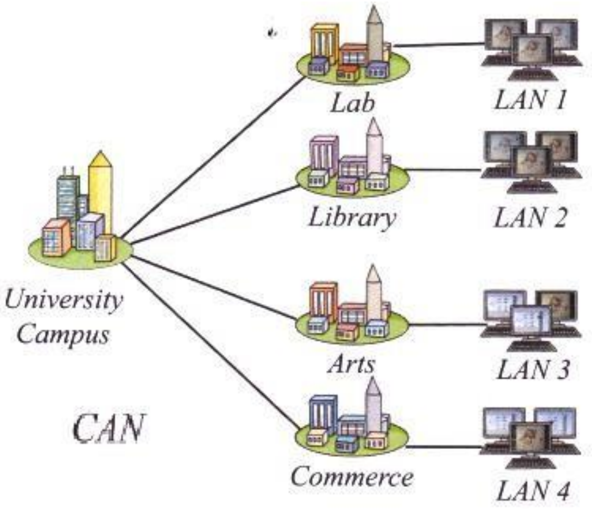
Personal Area Network (PAN): A PAN is a computer network organised around a person. It is used for communication between devices such as phones, personal digital assistants, printers and laptops that are in close proximity. We can use these networks to transfer files and photos between the various devices.



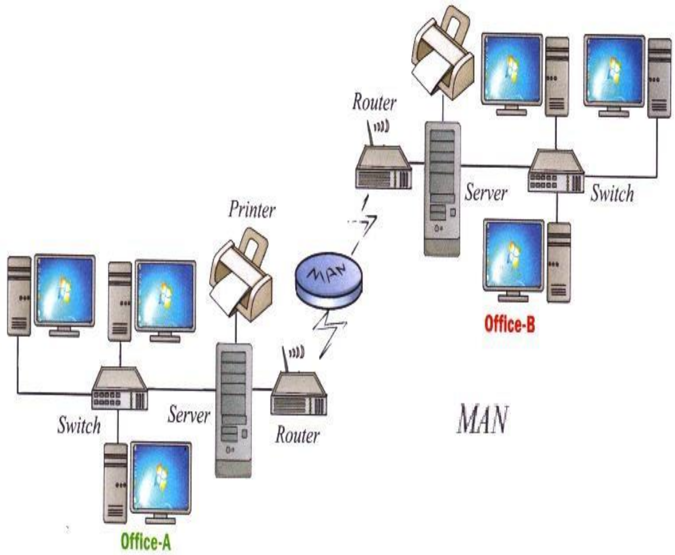
Campus Area Network (CAN): A CAN is a computer network that connects multiple local area networks (LAN) in a limited geographical area. A CAN is smaller than a wide area network (WAN) or metropolitan area network (MAN). It can be set up by a

Local Area Network (LAN): A LAN is a computer network that is limited to a local area such as a laboratory, a school or an office building. Cables (wires) or low-power radio-waves (wireless) are used for the connections in a LAN. A wireless LAN (or WLAN) is also sometimes called LAWN (Local Area Wireless Network).

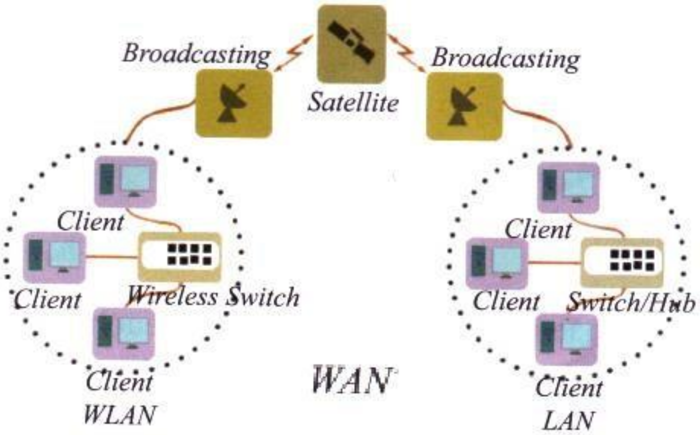
college, company and so on.



Metropolitan Area Network (MAN):A MAN is a computer network that usually covers a larger area than a LAN. For example, a network that connects two offices in a city, a neighbourhood area and so on.



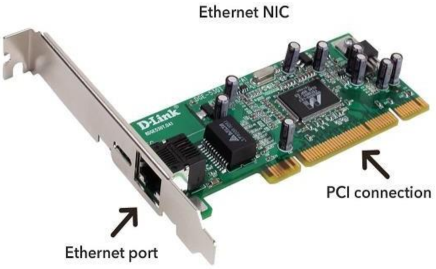
Wide Area Network (WAN):A WAN is a computer network that spans a wide geographical area. A WAN may be spread across cities, countries and continents. A WAN is formed by connecting LANs and MANs. Computers or networks across long distances are usually connected with optical fibre cables, satellite radio links or microwave radio links.



Global Area Network (GAN):A global network, such as the internet, is referred to as the Globe Area Network (GAN). The internet is, however, not the only computer network of its kind. Internationally operating companies also support local networks that comprise of several WANs and connect company computers across the world. GANs use the fibre optic infrastructure from wide area networks and combine these with international undersea cables or satellite transmissions.

NETWORK DEVICES

NIC (Network Interface Card):



It is a hardware device that is attached to a computer to enable it to communicate over the network. The NIC has a ROM chip that contains a unique number, which is the hardware address or the Media Access Control

(MAC) This hardware address uniquely identifies a computer on the network.

Modem (Modulator-demodulator):

It is an electronic device that converts the digital signals of a computer into an analog form so that they can travel over a telephone line. At the destination, the receiving modem converts the analog signals back into their digital form so that the destination computer understands them.Modems are used for connecting computers to the Internet. Modems are connected to a computer and a telephone line.

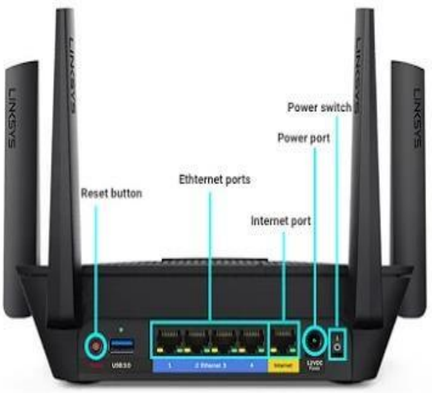
Hub:

A hub is a device that is used to connect computers in a network. In a hub, when one computer sends data on the network, the hub simply forwards the packets to all the other computers connected to it. Each computer is responsible for determining which packets are destined for it and which are to be ignored.

Switch:

A switch is a device that is also used to connect computers in a network. However, a switch is a more intelligent device than a hub. Unlike a hub, the switch sends the incoming data to the desired destination only. It records the addresses of all the computers connected to it. So, when a packet is received, the switch reads the information about the destination address to determine if the destination device is connected to it or not. If the destination device is connected, the switch forwards the packet only to that destination device. In this way, the other computers do not have to read and deal with data that is not meant for them.

Router:



A router is a network device that connects two or more networks. It is commonly used to connect a computer or a network to the Internet. Lines from different networks are connected to a router. Wireless routers are also available. A router examines the address of the packet coming on the line, uses the routing information stored in it and forwards the packet to the next network. In this way, a packet after going through multiple routers reaches its destination.

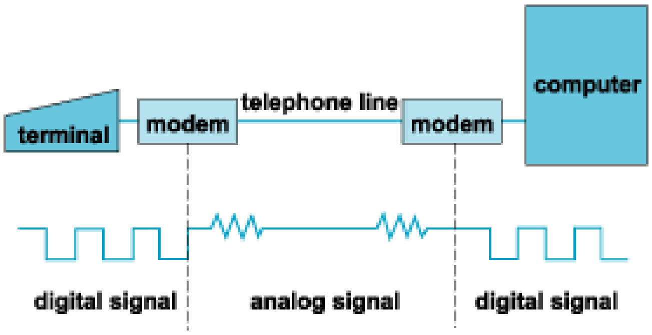
Bridge:

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.

Gateway:

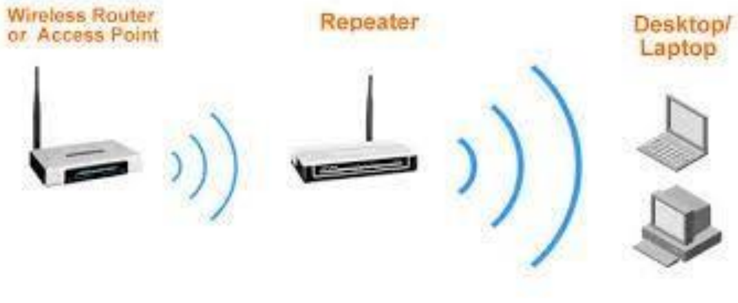
Gateways normally work at the Transport and Session layers of the OSI model. At the Transport layer and above, there are numerous protocols and standards from different vendors; gateways are used to deal with them. Gateways provide translation between networking technologies such as Open System Interconnection (OSI) and Transmission Control Protocol/Internet Protocol (TCP/IP). Because of this, gateways connect two or more autonomous networks, each with its own routing algorithms, protocols, topology, domain name service, and network administration procedures and policies.

Modem (Modulator-demodulator):



It is an electronic device that converts the digital signals of a computer into an analog form so that they can travel over a telephone line. At the destination, the receiving modem converts the analog signals back into their digital form so that the destination computer understands them.

Repeater:



A repeater is an electronic device that amplifies the signal it receives. You can think of repeater as a device which receives a signal and retransmits it at a higher level or higher power so that the signal can cover longer distances, more than 100 meters for standard LAN cables. Repeaters work on the Physical layer.

USES OF INTERNET:

Internet is a virtual networking medium that can be connected and used on a variety of devices these days. It enables the users to send, receive, collect, store, update, delete, and many other operations of the data across the world. Internet usage is expanding its boundaries every day, as the technological growth is huge. A few of the major uses of Internet are e-commerce, e-learning, knowledge sharing, social connectivity, variety of media, file transfer, communication, etc.

1. Electronic Mail (email): The first major use of the internet is Email. People thronged to Email for sharing information, data files, Photos, Videos, Business communications, and any other files instantaneously with others. This had enabled faster communication between people and improve business efficiency. Email has reduced the usage of paper considerably and reduced load on physical mail systems.Though other latest collaboration tools provide many rich features, they are not able to de-popularize Email and it still rules the official and personal communication. There are many free Email websites offering mail services and practically every individual has an Email address and connected by Email.

2. FTP File Transfer: This is the second major use case for the internet in the early days. FTP is the file transfer protocol that enables data exchange between two stakeholders over internet media in a secure way. The data exchange may occur between two business entities or customers with business and vice versa. Normally E-mail restricts the size of a file that can be shared and also it is not secured to share sensitive and confidential data across public networks. FTP concept is still in use even today in mobile apps for files downloading.

3. Search Engines: These engines locate the information one seeks, available in whichever server across the globe (world wide web). Google, Yahoo, and MSN are the renowned search engines in use today. One can search on anything in this site and the search question can be in any format. In fact, People have started using the word Google as a generic verb synonymous to search.

4. E-Commerce: The Internet enables the selling of goods and services in online mode. There are many e-commerce platform vendors like Amazon, Ola who aggregate several products/services available in the market and sell them through their portal to customers. Products are procured by platform vendors, stored in their warehouses, packed and distributed by them in their own brand. Customers get a good discount and they don’t have to visit physical stores.

5. Online Banking: Called as Net banking, it allows doing banking transactions at ease sitting at home or while on mobile. Footfalls in the bank branches have come down appreciably with almost all the services are available in net banking 24×7. Any amount of money can be transferred instantaneously through this facility. E-Banking supports Electricity bills, Telephone bills, and other services payment.

6. Cashless Transactions: Bill Payment at merchandise outlets through debit cards, credit cards, UPI gateway are on the increase. Cash circulation gets reduced in the system to the extent of the growth of these transactions. It’s growing by more than 50% every year and it is expected to grow by 10 times over the next 5 years.

7. Education: The Internet offers a wealth of educational material on any subject with structured navigation and search facilities. One can seek any reading material and the internet will get it for them from any server in any part of the world and people need not have to go to libraries to go through books. Those who cannot attend physical (face to face) class can take an online course where they get connected to the teacher, in the other part of the world, in video mode and get taught on the subject backed up other audio-visual tools.

8. Internet Conferencing: Online chat tools like messenger, Skype, and other video conferencing tools help people to get connected 24x7 and have a hassle-free business and personal discussion. This avoids unwanted travel by people and save their time for productive use. The Internet has also facilitated work from home with seamless connectivity to the office and avoid daily commuting.

9. Social Networking: Internet connects people online and enables them to form social groups. Information, Ideas, views, and opinions on any social/political issues are exchanged. The political and social organization makes use of this platform in promoting their interest among the public.

10. Chatting: On the Internet, chatting is talking to other people who are using the Internet at the same time you are. Usually, this "talking" is the exchange of typed-in messages requiring one site as the repository for the messages (or "chat site") and a group of users who take part from anywhere on the Internet. In some cases, a private chat can be arranged between two parties who meet initially in a group chat. Chats can be ongoing or scheduled for a particular time and duration. Most chats are focused on a particular topic of interest and some involve guest experts or famous people who "talk" to anyone joining the chat.

11. WWW:WWW stands for World Wide Web. A technical definition of the World Wide Web is: all the resources and users on the Internet that are using the Hypertext Transfer Protocol (HTTP).The World Wide Web is the universe of network-accessible information, an embodiment of human knowledge.In simple terms, The World Wide Web is a way of exchanging information between computers on the Internet, tying them together into a vast collection of interactive multimedia resources.

12. E-Newspapers: E-Newspapers are newspapers which are published electronically. They can take the form of normal print publications published on the Internet; additional or complimentary content to print publications published on-line; or original publications published exclusively on the World Wide Web. Many news organizations require subscription to e-newspapers, just like regular print newspapers. E-newspapers run the gauntlet of newspapers, from serious hard news, to features, to arts and entertainment, to sports, and everything in between.

DIFFERENT TYPES OF INTERNET CONNECTIONS

There are many ways a personal electronic device can connect to the internet. They all use different hardware and each has a range of connection speeds. As technology changes, faster internet connections are needed to handle those changes. I thought it would be interesting to list some of the different types of internet connections that are available for home and personal use, paired with their average speeds.

Dial-Up (Analog 56K): Dial-up access is cheap but slow. A modem (internal or external) connects to the Internet after the computer dials a phone number. This analog signal is converted to digital via the modem and sent over a land-line serviced by a public telephone network. Telephone lines are variable in quality and the connection can be poor at times. The lines regularly experience interference and this affects the speed, anywhere from 28K to 56K. Since a computer or other device shares the same line as the telephone, they can't be active at the same time.

DSL: DSL stands for Digital Subscriber Line. It is an internet connection that is always "on". This uses 2 lines so your phone is not tied up when your computer is connected. There is also no need to dial a phone number to connect. DSL uses a router to transport data and the range of connection speed, depending on the service offered, is between 128K to 8 Mbps.

Cable: Cable provides an internet connection through a cable modem and operates over cable TV lines. There are different speeds depending on if you are uploading data transmissions or downloading. Since the coax cable provides a much greater bandwidth over dial-up or DSL telephone lines, you can get faster access. Cable speeds range from 512K to 20 Mbps.

Wireless: Wireless, or Wi-Fi, as the name suggests, does not use telephone lines or cables to connect to the internet. Instead, it uses radio frequency. Wireless is also an always on connection and it can be accessed from just about anywhere. Wireless networks are growing in coverage areas by the minute so when I mean access from just about anywhere, I really mean it. Speeds will vary, and the range is between 5 Mbps to 20 Mbps.

Satellite: Satellite accesses the internet via a satellite in Earth's orbit. The enormous distance that a signal travels from earth to satellite and back again, provides a delayed connection compared to cable and DSL. Satellite connection speeds are around 512K to 2.0 Mbps.

Cellular: Cellular technology provides wireless Internet access through cell phones. The speeds vary depending on the provider, but the most common are 3G and 4G speeds. A 3G is a term that describes a 3rd generation cellular network obtaining mobile speeds of around 2.0 Mbps. 4G is the fourth generation of cellular wireless standards. The goal of 4G is to achieve peak mobile speeds of 100 Mbps but the reality is about 21 Mbps currently.

ISP(INTERNET SERVICE PROVIDER)

An ISP is a company that provides individuals and other companies access to the Internet and other related services such as Web site building and virtual hosting. An ISP has the equipment and the telecommunication line access required to have a point-of-presence on the Internet for the geographic area served. The larger ISPs have their own high-speed leased lines so that they are less dependent on the telecommunication providers and can provide better service to their customers.

Short Question

1. **Define Network.**
2. **What is Internet?**
3. **Define E-MAIL.**
4. **Define switch.**
5. **Define Hub.**
6. **Define Repeater.**
7. **Define ISP.**
8. **Define WWW.**

Long Question

1. **Discuss the different types of computer network.**
2. **Explain the different services used in Internet.**
3. **Explain the different network devices used in computer network.**

CH4-FILE MANAGEMENT & DATA PROCESSING

FILE AND FOLDER

File: A File is defined as a set of related data or information that is being stored in secondary storage. A file is data file or a program file where former contains data and information in the form of alphanumeric, numeric or binary and latter containing the program code and can also be executed, is a program file.

Folder: It is used to contain many other folders and files. We can have any number of folders, and each folder can have different/numerous entries depending on the files created where each file has a position in a parent folder.

Difference between File and Folder:

SL. NO.	COMPARISON	FILE	FOLDER
1	Extensions	Files can have extensions.	Folders does not have any extensions.
2	Organizations	Serial, sequential, indexed sequential and direct file organizations.	Single directory per user and multiple directories per user organization.
3	Contain other same entity	No.	Yes.
4	Basic	Collection of data.	A place to store a group of related files and folders.
5	Space consumption	There is a specific size of a file.	Folder does not consume space in the memory.
6	Properties	It has Name, Extension, Date, Time, Length and Protection attributes.	It has Name, Date, Time and Protection attributes.
7	After Creation	After creation, we can open, save, rename, print, email and modify file content.	After creation, we can move, rename and delete folders.
8	Share on Network	We can't share file on network.	We can share folder on network.

FILE ACCESS METHODS

When a file is used, information is read and accessed into computer memory and there are several ways to access this information of the file. Some systems provide only one access method for files. Other systems, such as those of IBM, support many access methods, and choosing the right one for a particular application is a major design problem.There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

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: Another method is direct access method also known as relative access method. A filed-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 14 then block 59 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 SequentialAccessMethod (ISAM): It is the other method of accessing a file which is built on the top of the direct 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 controls the pointer by using index.

DATA PROCESSING

Data can be handled in 4 ways.

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. Optical character recognition can also be a component of data capture involving the extraction of text from scanned or digital documents (receipts, contracts, books, etc.,) and the conversion of the results into data for editing and processing.

Data storage: Data storage is a general term for archiving data in electromagnetic or other forms for use by a computer or device. Different types of data storage play different roles in a computing environment. In addition to forms of hard data storage, there are now new options for remote data storage, such as cloud computing, that can revolutionize the ways that users access data.

Data process: Data processing is the conversion of data into usable and desired form. This conversion or "processing" is carried out using a predefined sequence of operations either manually or automatically. Most of the processing is done by using computers and thus done automatically. The output or "processed" data can be obtained in various forms. Example of these forms include image, graph, table, vector file, audio, charts or any other desired format. The form obtained depends on the software or method of data processing used. When done itself it is referred to as automatic data processing.

Data Retrieval: Data retrieval is the process of identifying and extracting data from a database, based on a query provided by the user or application.It enables the fetching of data from a database in order to display it on a monitor and/or use within an application.Data retrieval typically requires writing and executing data retrieval or extraction commands or queries on a database. Based on the query provided, the database looks for and retrieves the data requested. Applications and software generally use various queries to retrieve data in different formats. In addition to simple or smaller data, data retrieval can also include retrieving large amounts of data, usually in the form of reports.

Short Question

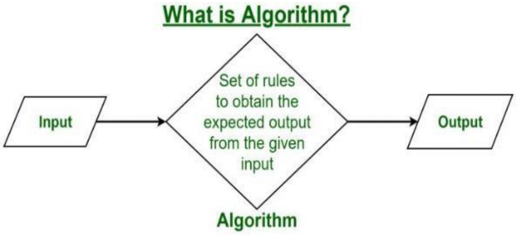
1. What is file?
2. What is folder?
3. What is sequence access?
4. Define data capture.

Long Questions

What is file ? Discuss the different types of file access method.

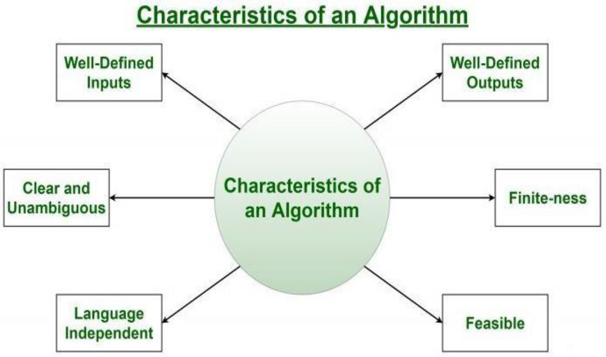
CH-5 PROBLEM SOLVING METHODOLOGY

ALGORITHMS


What is Algorithm?
Input
Set of rules to obtain the expected output from the given input
Output
Algorithm

Introduction: The word Algorithm means “a process or set of rules to be followed in calculations or other problem-solving operations”. Therefore, Algorithm refers to a set of rules/instructions that step-by-step define how a work is to be executed upon in order to get the expected results.It can be understood by taking an example of cooking a new recipe. To cook a new recipe, one reads the instructions and steps and execute them one by one, in the given sequence. The result thus obtained is the new dish cooked perfectly. Similarly, algorithms help to do a task in programming to get the expected output.The Algorithm designed are language-independent, i.e. they are just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.

Characteristics of an Algorithm: As one would not follow any written instructions to cook the recipe, but only the standard one. Similarly, not all written instructions for programming is an algorithm. In order for some instructions to be an algorithm, it must have the following characteristics:



- **Clear and Unambiguous:** Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-Defined Inputs:** If an algorithm says to take inputs, it should be well-defined inputs.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well.
- **Finite-ness:** The algorithm must be finite, i.e. it should not end up in an infinite loop or similar.
- **Feasible:** The algorithm must be simple, generic and practical, such that it can be executed upon will the available resources. It must not contain some future technology, or anything.
- **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

Pseudocode: It’s simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.

Examples of Algorithms in Programming:

1. Write an algorithm to add two numbers entered by the user.

```
Step 1:    Start
Step 2:    Declare variables num1, num2 and sum.
Step 3:    Read values num1 and num2.
Step 4:    Add num1 and num2 and assign the result to sum.
           sum←num1+num2
Step 5:    Display sum
Step 6:    Stop
```

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

```
Step 1:    Start
Step 2:    Declare variables a,b and c.
Step 3:    Read variables a,b and c.
Step 4:    If a > b&&If a > c
               Display 'a' is the largest number.
           Else
               Display 'c' is the largest number.
           Else
               If b > c
                   Display 'b' is the largest number.
               Else
                   Display 'c' is the greatest number.
Step 5:    Stop
```

3. Write an algorithm to check whether a number is prime or not.

```
Step 1:    Start
Step 2:    Declare variables n, i, flag.
Step 3:    Initialize variables
           flag ← 1
           i← 2
Step 4:    Read n from the user.
Step 5:    Repeat the steps until i<(n/2)
    5.1    If remainder of n÷i equals 0
               flag ← 0
               Go to step 6
    5.2    i← i+1
Step 6:    If flag = 0
               Display n is not prime
           else
```

```
                Display n is prime
Step 7:     Stop
```

4. Write an algorithm to find the factorial of a number entered by the user.

```
Step 1:     Start
Step 2:     Declare variables n, factorial and i.
Step 3:     Initialize variables
            factorial ← 1
            i← 1
Step 4:     Read value of n
Step 5:     Repeat the steps until i = n
    5.1:    factorial ← factorial*i
    5.2:    i← i+1
Step 6:     Display factorial
Step 7:     Stop
```

5. Write an algorithm to find the Fibonacci series till term≤1000.

```
Step 1:     Start
Step 2:     Declare variables first_term,second_term and temp.
Step 3:     Initialize variables first_term← 0 second_term← 1
Step 4:     Display first_term and second_term
Step 5:     Repeat the steps until second_term ≤ 1000
    5.1:    temp ←second_term
    5.2:    second_term←second_term + first_term
    5.3:    first_term← temp
    5.4:    Display second_term
Step 6:     Stop
```

6. Write an algorithm to find all roots of a quadratic equation ax2+bx+c=0.

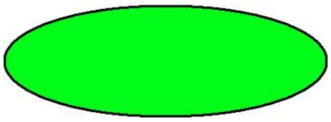
```
Step 1:     Start
Step 2:     Declare variables a, b, c, D, x1, x2, rp and ip;
Step 3:     Calculate discriminant
            D ← b2-4ac
Step 4:     If D ≥ 0
                r1 ← $(-b+\sqrt{D})/2a$
                r2 ← $(-b-\sqrt{D})/2a$
                Display r1 and r2 as roots.
            Else
                Calculate real part and imaginary part
                rp← b/2a
                ip←$\sqrt{(-D)}/2a$
                Display rp+j(ip) and rp-j(ip) as roots
Step 5:     Stop
```

FLOWCHART

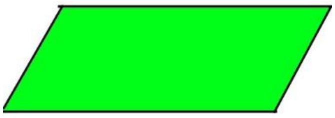
Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing.

Symbols used in Flowchart:

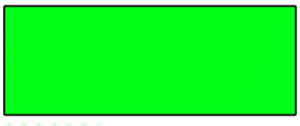
Terminal: The oval symbol indicates Start, Stop and Halt in a program's logic flow. A pause/halt is generally used in a program logic under some error conditions. Terminal is the first and last symbols in the flowchart.



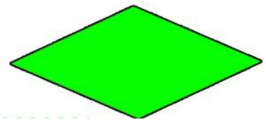
Input/Output: A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



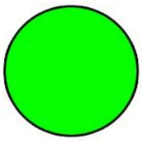
Processing: A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



Decision: Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.

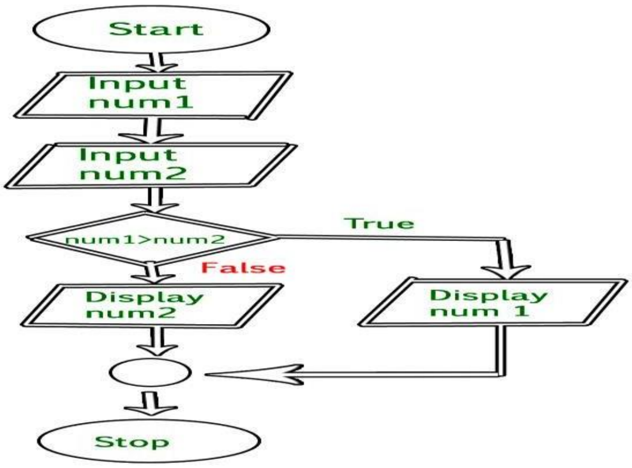


Connectors: Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.



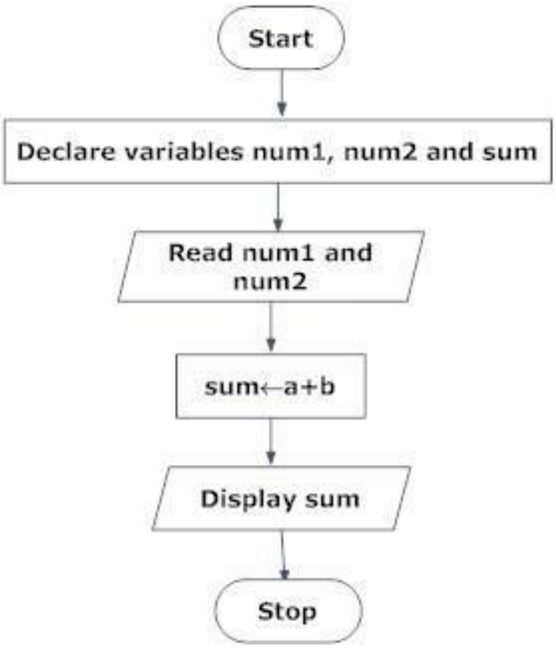
Flow lines: Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.

Example: **Draw a flowchart to input two numbers from user and display the largest of two numbers.**

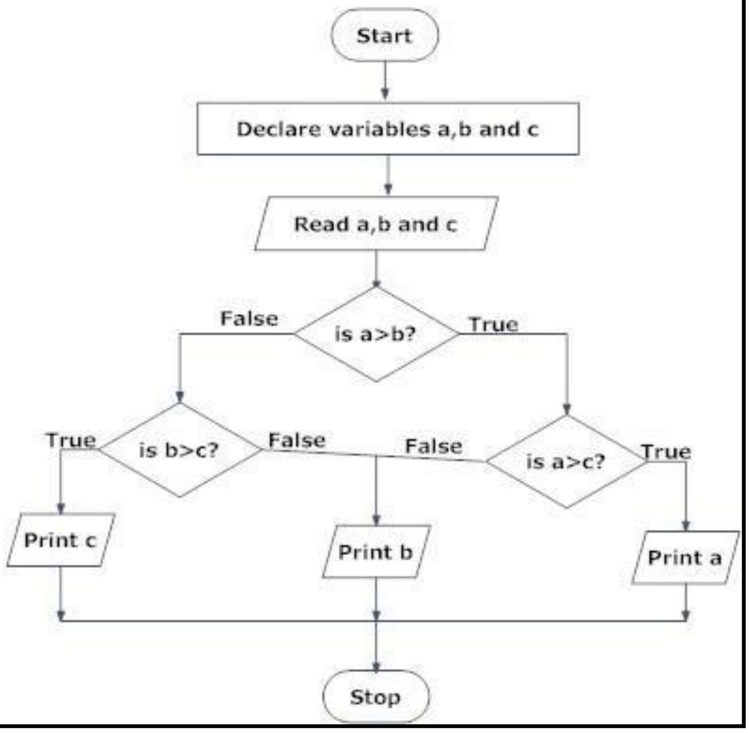


Examples of flowcharts in programming:

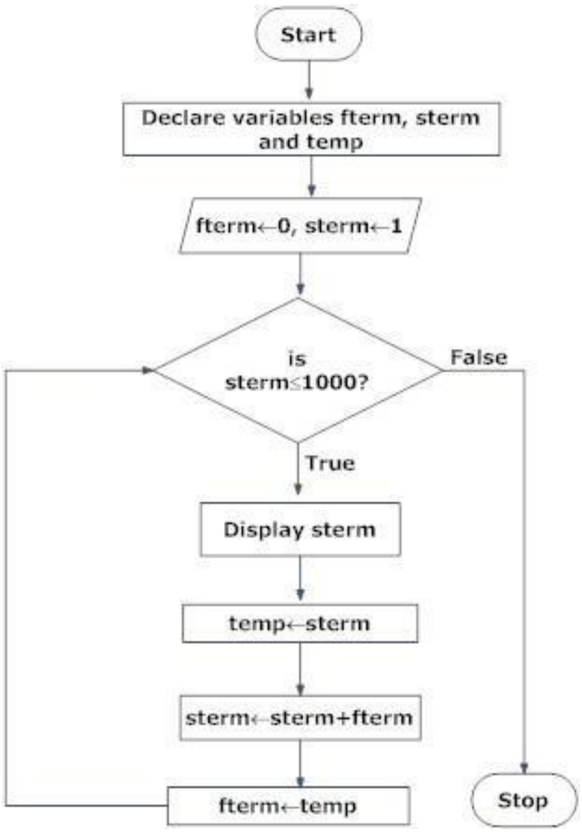
1. Add two numbers entered by the user.



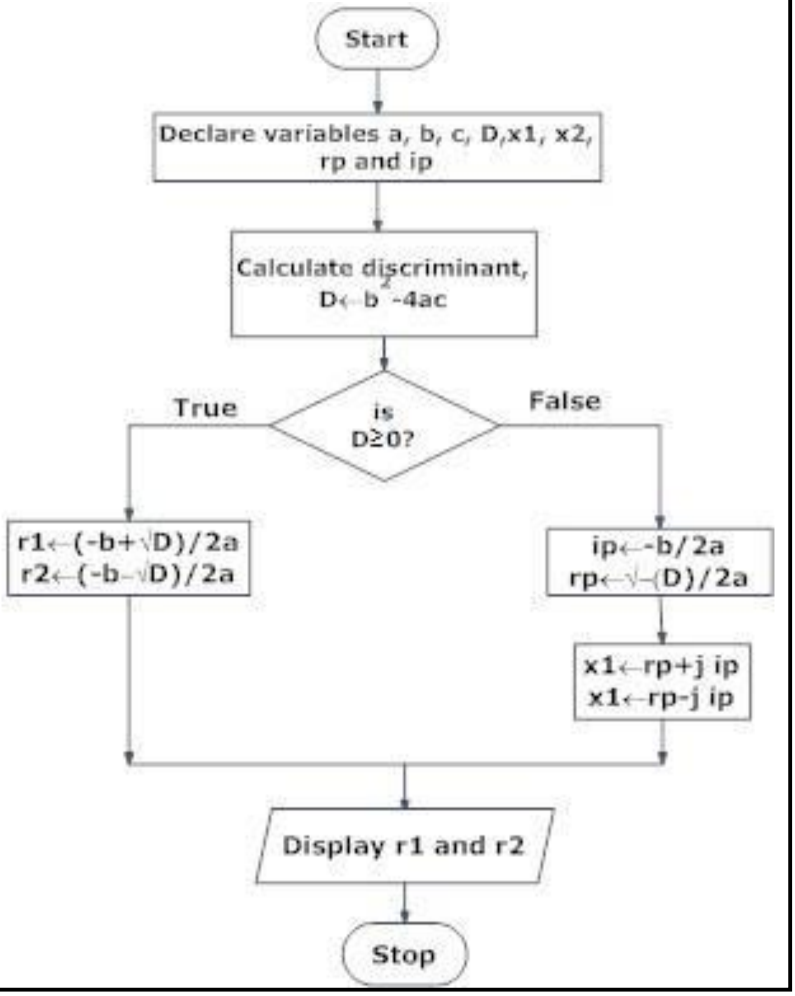
2. Find the largest among three different numbers entered by the user.



3. Find the Fibonacci series till term≤1000.



4. Find all the roots of a quadratic equation $ax^2+bx+c=0$.



Generations of Programming Language

First Generation Languages: Here we are talking about machine code. This is the only form of code that can be executed by the computer directly.

Second Generation Languages: Assembly language was developed to make it easier for programmers to write instructions than it would be using machine code. Mnemonics are used instead of bit patterns (which are harder to remember). First and second-generation languages are low level and machine-oriented. This refers to the way that they are based on the machine operations that are available for a given processor.

Third Generation Languages: Third generation languages are **high level**, **platform-independent** and **problem oriented**. When source code is compiled, there is a one-to-many equivalence of high-level language statements to machine code statements. Third generation programs can be run on any platform for which an appropriate compiler or interpreter exists.High level languages are developed to help solve particular types of problem. The FORTRAN language was designed with Mathematics, Science and Engineering in mind, it contains lots of scientific functions that the average programmer may not need. The COBOL language was developed with business logic in mind, PHP was developed for server-side scripting and so on.

All of the languages in the first 3 generations are called **imperative** languages because the program's statements are executed in the order specified by the programmer.

Fourth Generation Languages: Fourth generation languages are **declarative**. This means that the programmer will write facts or rules rather than statements. The interpreter for the language produces the result using whichever standard algorithms it has been given for doing so.SQL and Prolog are both examples of declarative languages. Both are described in the programming section of this site and are relatively easy to try out. A quick half-hour blast at each would give you a feel for how they work and help you to understand how they differ from the other types of language.

STRUCTURED PROGRAMMING LANGUAGE

Structured Programming Approach, as the word suggests, can be defined as a programming approach in which the program is made as a single structure. It means that the code will execute the instruction by instruction one after the other. It doesn't support the possibility of jumping from one instruction to some other with the help of any statement like GOTO, etc. Therefore, the instructions in this approach will be executed in a serial and structured manner. The structured program consists of well-structured and separated modules. But the entry and exit in a Structured program is a single-time event. It means that the program uses single-entry and single-exit elements. The languages that support Structured programming approach are:C, C++, Java, C#.

Short Questions

1. What is algorithm.?
2. What is Flowchart?
3. What is pseudo code?
4. Define Structure programming language.

CH 6-OVERVIEW OF C PROGRAMMING LANGUAGE

CONSTANTS

As the name suggests the name constants is given to such variables or values in C/C++ programming language which cannot be modified once they are defined. They are fixed values in a program. There can be any types of constants like integer, float, octal, hexadecimal, character constants etc. Every constant has some range. The integers that are too big to fit into an int will be taken as long. Now there are various ranges that differ from unsigned to signed bits. Under the signed bit, the range of an int varies from -128 to +127 and under the unsigned bit, int varies from 0 to 255.

Defining Constants:

In C program we can define constants in two ways as shown below.

Using #define pre-processor directive:
#include<stdio.h>
#define val 10
#define floatVal 4.5
#define charVal 'G'
int main()
{
printf("Integer Constant: %d\n",val);
printf("Floating point Constant: %.1f\n",floatVal);
printf("Character Constant: %c\n",charVal);
return 0;
}
	Output:
	Integer Constant: 10
	Floating point Constant: 4.5
	Character Constant: G

Using a const keyword:
#include <stdio.h>
int main()
{
constintintVal = 10;
const float floatVal = 4.14;
const char charVal = 'A';
const char stringVal[10] = "ABC";
printf("Integer constant:%d \n", intVal);
printf("Floating point constant: %.2f\n", floatVal);
printf("Character constant: %c\n", charVal);
printf("String constant: %s\n", stringVal);
 return 0;
}
	Output:
	Integer constant: 10
	Floating point constant: 4.14
	Character constant: A
	String constant: ABC

VARIABLE IN ‘C’

A variable definition tells the compiler where and how much storage to create for the variable. A variable definition specifies a data type and contains a list of one or more variables of that type as follows −

type variable_list;

Here, type must be a valid C data type including char, w_char, int, float, double, bool, or any user-defined object; and variable_list may consist of one or more identifier names separated by commas. Some valid declarations are shown here −

inti, j, k;
char   c, ch;
float f, salary;
double d;

The line inti, j, k; declares and defines the variables i, j, and k; which instruct the compiler to create variables named i, j and k of type int.Variables can be initialized (assigned an initial value) in their declaration. The initializer consists of an equal sign followed by a constant expression as follows –

type variable_name = value;

Example

Try the following example, where variables have been declared at the top, but they have been defined and initialized inside the main function –

```
#include <stdio.h>
// Variable declaration:
extern int a, b;
extern int c;
extern float f;
int main ()
{
   /* variable definition: */
   int a, b;
   int c;
   float f;
    /* actual initialization */
   a = 10;
   b = 20;
   c = a + b;
printf("value of c : %d \n", c);
   f = 70.0/3.0;
printf("value of f : %f \n", f);
   return 0;
}
```

Output:

```
value of c : 30
value of f : 23.333334
```

DIFFERENCE BETWEEN VARIABLE AND CONSTANT

CONSTANTS	VARIABLE
A value that cannot be altered throughout the program	A storage location paired with an associated symbolic name which has a value
It is similar to a variable but it cannot be modified by the program once defined	A storage area holds data
Cannot be changed	Can be changed according to the need of the programmer
Value is fixed	Value is varying

C PROGRAMMING – MANAGING INPUT AND OUTPUT OPERATIONS

Input means to provide the program with some data to be used in the program and Output means to display data on screen or write the data to a printer or a file.C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.All these built-in functions are present in C header files, we will also specify the name of header files in which a particular function is defined while discussing about it.

scanf() and printf() functions

The standard input-output header file, named stdio.h contains the definition of the functions printf() and scanf(), which are used to display output on screen and to take input from user respectively.

Example:

```
#include<stdio.h>
void main()
{
inta,b,c;
printf("Please enter any two numbers: \n");
scanf("%d %d", &a, &b);
c =  a + b;
printf("The addition of two number is: %d", c);
}
```

Output:

```
Please enter any two numbers:
12
3
The addition of two number is:15
```

You must be wondering what is the purpose of %d inside the scanf() or printf() functions. It is known as format string and this informs the scanf() function, what type of input to expect and in printf() it is used to give a heads up to the compiler, what type of output to expect.

Format String	**Meaning**
%d	Scan or print an integer as signed decimal number
%f	Scan or print a floating-point number
%c	To scan or print a character

C - Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators −

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Misc Operators

Arithmetic Operators-

The following table shows all the arithmetic operators supported by the C language. Assume variable A holds 10 and variable B holds 20 then –

Operator	Description	Example
+	Adds two operands.	A + B = 30
−	Subtracts second operand from the first.	A − B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

Relational Operators-

The following table shows all the relational operators supported by C. Assume variable A holds 10 and variable B holds 20 then –

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A==B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A!=B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A>B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A<B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A>=B) is not true.
<=	Checks if the value of left operand is less than or equal	(A<=B) is

	to the value of right operand. If yes, then the condition becomes true.	true.

Logical Operators-

Following table shows all the logical operators supported by C language. Assume variable A holds 1 and variable B holds 0, then –

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A&&B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A||B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A&&B) is true.

Bitwise Operators-

Bitwise operator works on bits and perform bit-by-bit operation. The truth tables for &, |, and ^ is as follows –

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

The following table lists the bitwise operators supported by C. Assume variable 'A' holds 60 and variable 'B' holds 13, then –

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e,. -0111101
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right	A >> 2 = 15 i.e., 0000 1111

	operand.	

Assignment Operators-

The following table lists the assignment operators supported by the C language –

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C=A+B will assign the value of A+B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C+=A is equivalent to C=C+A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C-=A is equivalent to C=C-A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C*=A is equivalent to C=C*A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C/=A is equivalent to C=C/A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	C%=A is equivalent to C=C%A
<<=	Left shift AND assignment operator.	C<<=2 is same as C=C<<2
>>=	Right shift AND assignment operator.	C>>=2 is same as C=C>>2
&=	Bitwise AND assignment operator.	C&=2 is same as C=C&2
^=	Bitwise exclusive OR and assignment operator.	C^=2 is same as C=C^2
|=	Bitwise inclusive OR and assignment operator.	C|=2 is same as C=C|2

Misc Operators ↦sizeof& ternary-

Besides the operators discussed above, there are a few other important operators including sizeof and ? : supported by the C Language.

Operator	Description	Example
sizeof()	Returns the size of a variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of a variable.	&a; returns the actual address of the variable.
*	Pointer to a variable.	*a;

? :	Conditional Expression.	If Condition is true? then value X : otherwise value Y

Operators Precedence in C-

Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator.For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has a higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* &sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<<>>	Left to right
Relational	<<= >>=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

C EXPRESSIONS

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

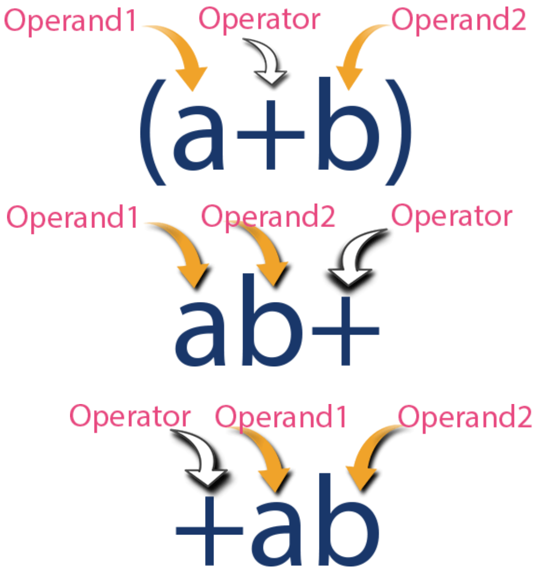
In the C programming language, expressions are divided into THREE types. They are as follows based on the operator position in the expression.

1. Infix Expression
2. Postfix Expression
3. Prefix 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.

Postfix Expression: The expression in which the operator is used after operands is called postfix expression.The postfix expression has the following general structure.

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.



C - TYPE CASTING OR TYPE CONVERSION

Converting one datatype into another is known as type casting or, type-conversion. For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int'. You can convert the values from one type to another explicitly using the cast operator as follows –

(type_name) expression

Consider the following example where the cast operator causes the division of one integer variable by another to be performed as a floating-point operation –

Example:

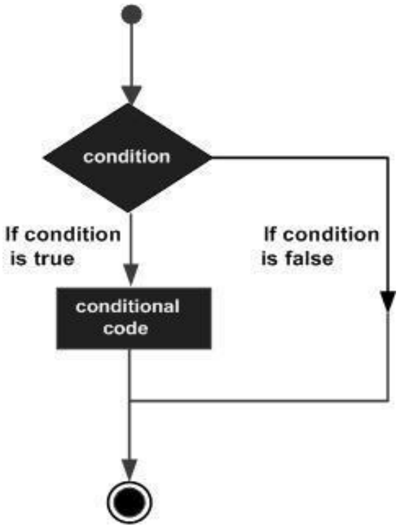
```
#include <stdio.h>
main()
{
   int sum = 17, count = 5;
   double mean;

   mean = (double) sum / count;
printf("Value of mean : %f\n", mean );
}
```

Output:

```
Value of mean : 3.400000
```

C - DECISION MAKING

Decision making structures require that the programmer specifies one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.C programming language assumes any non-zero and non-null values as true, and if it is either zero or null, then it is assumed as false value.C programming language provides the following types of decision-making statements.



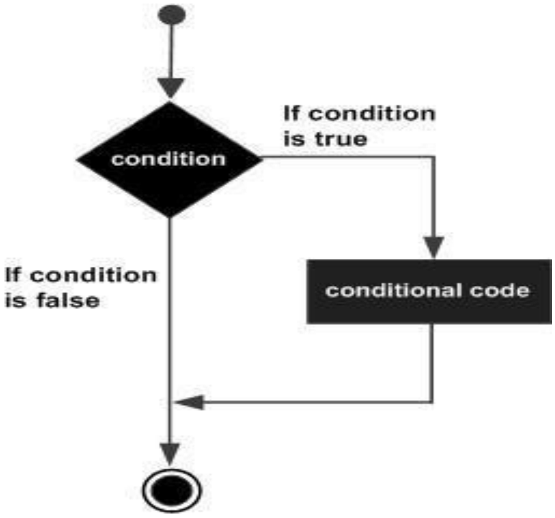
Sl.No.	Statement & Description
1	**if statement:** An **if statement** consists of a boolean expression followed by one or more statements.
2	**if...else statement:** An **if statement** can be followed by an optional **else statement**, which executes when the Boolean expression is false.
3	**nested if statements:** You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).
4	**switch statement:** A **switch** statement allows a variable to be tested for equality against a list of values.
5	**nested switch statements:** You can use one **switch** statement inside another **switch** statement(s).

if statement

An if statement consists of a Boolean expression followed by one or more statements.

Syntax *if(boolean_expression)*

```
if(boolean_expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
```



If the Boolean expression evaluates to true, then the block of code inside the 'if' statement will be executed. If the Boolean expression evaluates to false, then the first set of code after the end of the 'if' statement (after the closing curly brace) will be executed.C programming language assumes any non-zero and non-null values as true and if it is either zero or null, then it is assumed as false value.

Example:

```
#include <stdio.h>
int main ()
{
  int a = 10;
  if( a < 20 )
{
printf("a is less than 20\n"
);
  }
printf("value of a is :
%d\n", a);
  return 0;
}
```

Output:

```
a is less than 20;
value of a is : 10
```

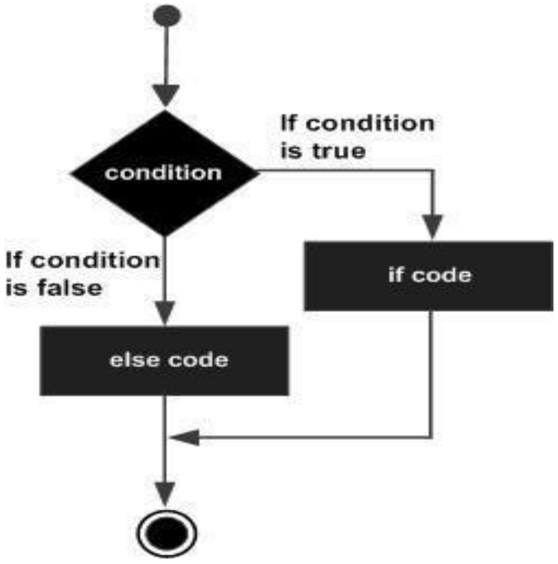
if...else statement

An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.

Syntax

```
if(boolean_expression) {
   /* statement(s) will execute if the boolean expression is true */
} else {
   /* statement(s) will execute if the boolean expression is false */
}
```

If the Boolean expression evaluates to true, then the if block will be executed, otherwise, the else block will be executed.C programming language assumes any non-zero and non-null values as true, and if it is either zero or null, then it is assumed as false value.



Example:	Output:

#include <stdio.h>
int main ()
{
  int a = 100;
  if(a < 20)
{
printf("a is less than 20\n");
  }
else
{
printf("a is not less than 20\n");
  }
printf("value of a is : %d\n", a);
  return 0;
}
	a is not less than 20;
	value of a is : 100

if...else if...else Statement

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.When using if...else if..else statements, there are few points to keep in mind −

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of the remaining else if's or else's will be tested.

Syntax

```
if(boolean_expression 1) {
   /* Executes when the boolean expression 1 is true */
} else if( boolean_expression 2) {
   /* Executes when the boolean expression 2 is true */
} else if( boolean_expression 3) {
   /* Executes when the boolean expression 3 is true */
} else {
   /* executes when the none of the above condition is true */
}
```

Example:

```
#include <stdio.h>
int main ()
{
int a = 100;
   if( a == 10 )
{
printf("Value of a is 10\n" );
   }
else if( a == 20 )
{
printf("Value of a is 20\n" );
   }
else if( a == 30 )
{
printf("Value of a is 30\n" );
   }
else
{
printf("None of the values is matching\n" );
   }
printf("Exact value of a is: %d\n", a );
   return 0;
```

Output:

```
None of the values is matching
Exact value of a is: 100
```

```
}
```

nested if statements

It is always legal in C programming to nest if-else statements, which means you can use one if or else if statement inside another if or else if statement(s).

Syntax

```
if( boolean_expression 1) {
   /* Executes when the boolean expression 1 is true */
   if(boolean_expression 2) {
      /* Executes when the boolean expression 2 is true */
   }
}
```

You can nest else if...else in the similar way as you have nested if statements.

Example:

```
#include <stdio.h>
int main ()
{
   int a = 100;
   int b = 200;
   if( a == 100 )
{
      if( b == 200 )
{
printf("Value of a is 100 and b is 200\n" );
      }
   }
printf("Exact value of a is : %d\n", a );
printf("Exact value of b is : %d\n", b );
   return 0;
}
```

Output:

```
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

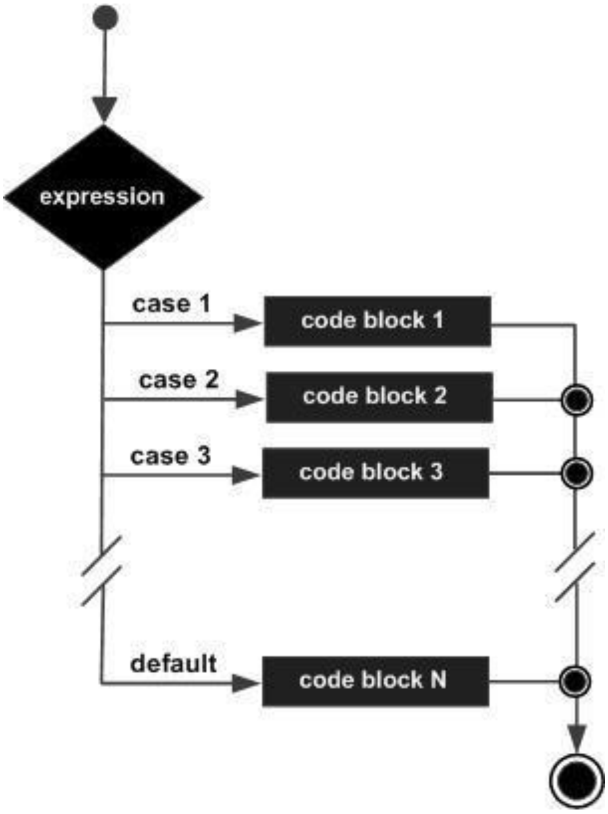
switch statement

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.

Syntax

```
switch(expression) {
    case constant-expression :
      statement(s);
      break;   /*  optional   */
    case constant-expression  :
      statement(s);
      break; /* optional */
    /* you can have any number of case statements */
    default : /* Optional */
    statement(s);
 }
```

The following rules apply to a switch statement −



- The expression used in a switch statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The constant-expression for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.
- When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.

- A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

Example:

```
#include <stdio.h>
int main ()
{
  char grade = 'B';
  switch(grade)
{
    case 'A' :
printf("Excellent!\n" );
      break;
    case 'B' :
    case 'C' :
printf("Well done\n" );
      break;
    case 'D' :
printf("You passed\n" );
      break;
    case 'F' :
printf("Better try again\n" );
      break;
    default :
printf("Invalid grade\n" );
  }
printf("Your grade is %c\n", grade );
  return 0;
}
```

Output:

```
Well done
Your grade is B
```

nested switch statements

It is possible to have a switch as a part of the statement sequence of an outer switch. Even if the case constants of the inner and outer switch contain common values, no conflicts will arise.

Syntax

```
switch(ch1)
{
  case 'A':
printf("This A is part of outer switch" );
    switch(ch2)
    {
      case 'A':
printf("This A is part of inner switch" );
        break;
      case 'B': /* case code */
    }
    break;
  case 'B': /* case code */
}
```

Example:

```
#include <stdio.h>
int main ()
{
  int a = 100;
  int b = 200;
  switch(a)
{
      case 100:
printf("This is part of outer switch\n", a );
      switch(b)
{
        case 200:
printf("This is part of inner switch\n", a );
      }
  }
printf("Exact value of a is : %d\n", a );
printf("Exact value of b is : %d\n", b );
  return 0;
```

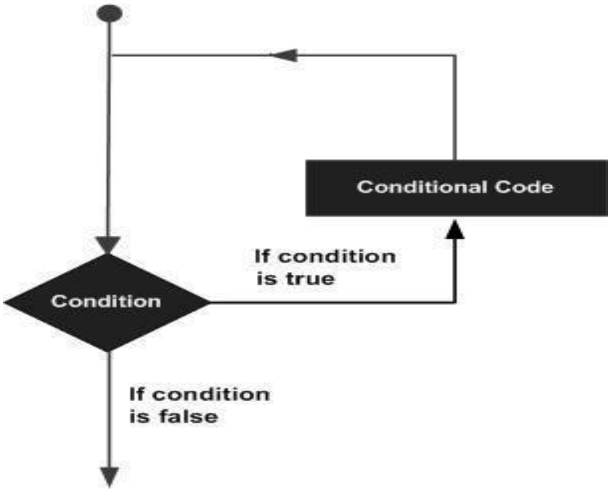
Output:

```
This is part of outer switch
This is part of inner switch
Exact value of a is : 100
Exact value of b is : 200
```

}	

C - LOOPS

You may encounter situations, when a block of code needs to be executed several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.Programming languages provide various control structures that allow for more complicated execution paths.A loop statement allows us to execute a statement or group of statements multiple times.



Sl.No.	Loop Type & Description
1	**while loop:** Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	**for loop:** Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	**do...while loop:** It is more like a while statement, except that it tests the condition at the end of the loop body.
4	**nested loops:** You can use one or more loops inside any other while, for, or do..while loop.

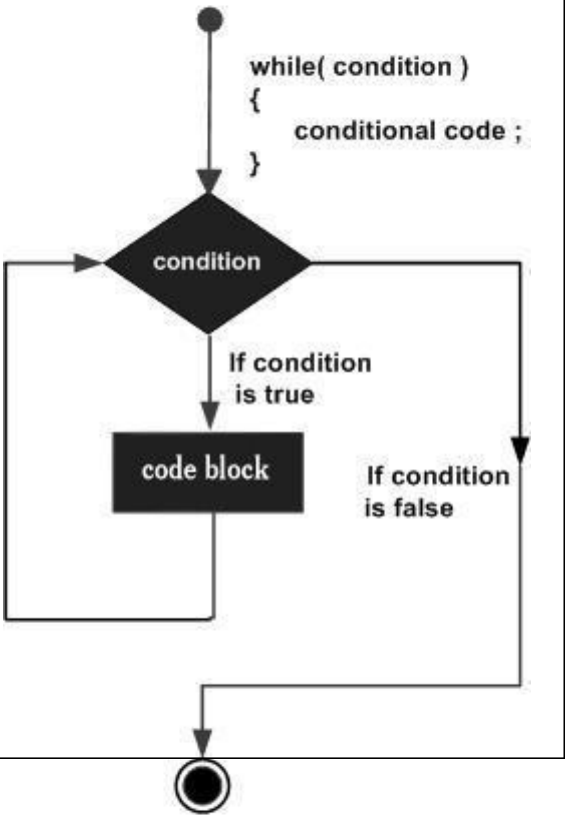
while loop

A while loop in C programming repeatedly executes a target statement as long as a given condition is true.

Syntax

```
while(condition) {
   statement(s);
}
```

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any nonzero value. The loop iterates while the condition is true.When the condition becomes false, the program control passes to the line immediately following the loop.Here, the key point to note is that a while loop might not execute at all. When the condition is tested and the result is false, the loop body



will be skipped and the first statement after the while loop will be executed.

Example:

```
#include <stdio.h>
int main ()
{
  int a = 10;
  while( a < 20 )
{
printf("value of a: %d\n", a);
    a++;
  }
  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
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

for loop

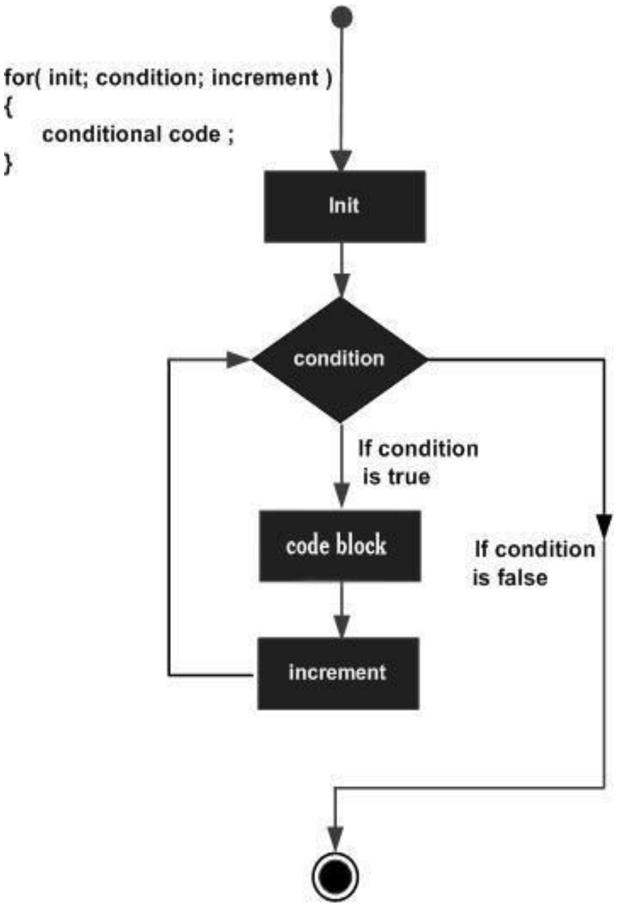
A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax

```
for ( init; condition; increment ) {
   statement(s);
}
```

Here is the flow of control in a 'for' loop −

- The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
- Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
- After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This



statement can be left blank, as long as a semicolon appears after the condition.

- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

Example:

```
#include <stdio.h>
int main ()
{
   int a;
   for( a = 10; a < 20; a = a + 1 )
{
printf("value of a: %d\n", a);
   }
   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
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

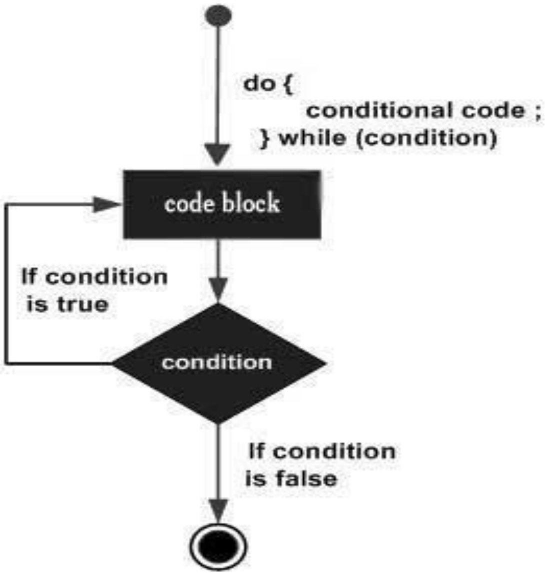
do...while loop

Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop in C programming checks its condition at the bottom of the loop.A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

Syntax

```
do   {
   statement(s);
} while( condition );
```

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested.If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.



Example:

```
#include <stdio.h>
int main ()
{
   int a = 10;
   do
{
printf("value of a: %d\n", a);
```

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
value of a: 16
```

```
      a = a + 1;
   }while( a < 20 );
   return 0;
}
```

```
value of a: 17
value of a: 18
value of a: 19
```

nested loops

C programming allows to use one loop inside another loop.

The syntax for a nested for loop statement in C is as follows –

```
for ( init; condition; increment ) {
   for ( init; condition; increment ) {
      statement(s);
   }
   statement(s);
}
```

The syntax for a nested while loop statement in C is as follows −

```
while(condition) {
   while(condition) {
      statement(s);
   }
   statement(s);
}
```

The syntax for a nested do...while loop statement in C is as follows −

```
do {
   statement(s);
   do {
      statement(s);
   }while( condition );
}while( condition );
```

A final note on loop nesting is that you can put any type of loop inside any other type of loop. For example, a 'for' loop can be inside a 'while' loop or vice versa.

Example:

```
#include <stdio.h>
int main ()
{
inti, j;
   for(i = 2; i<30; i++)
{
      for(j = 2; j <= (i/j); j++)
      if(!(i%j)) break;
```

Output:

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
```

```
    if(j > (i/j)) printf("%d is prime\n", i);
  }
  return 0;
}
```

```
23 is prime
29 is prime
```

LOOP CONTROL STATEMENTS

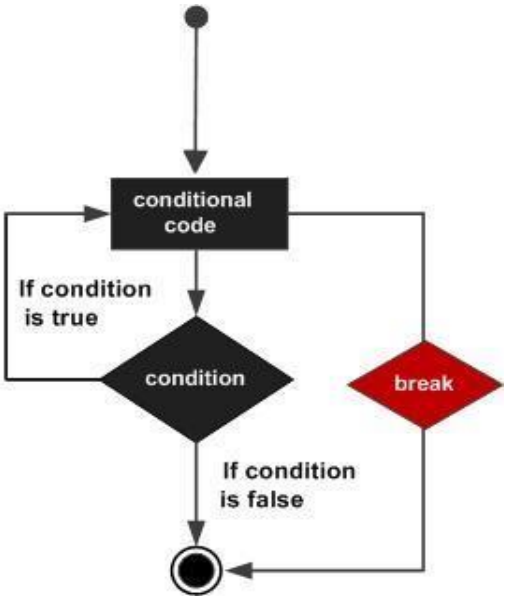
Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.C supports the following control statements.

Sl.No.	Control Statement & Description
1	**break statement:**Terminates the **loop** or **switch** statement and transfers execution to the statement immediately following the loop or switch.
2	**continue statement:**Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
3	**goto statement:**Transfers control to the labelled statement.

break statement

The break statement in C programming has the following two usages −

- When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.
- It can be used to terminate a case in the switch statement (covered in the next chapter).



If you are using nested loops, the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Syntax *break;*

Example:

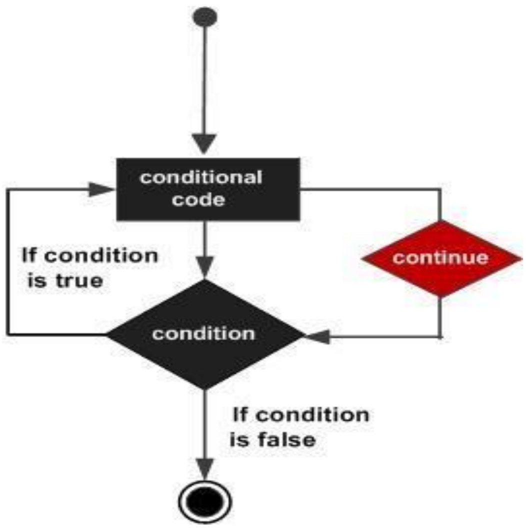
```
#include <stdio.h>
int main ()
{
  int a = 10;
  while( a < 20 )
{
printf("value of a: %d\n", a);
```

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
```

```
     a++;
     if( a > 15)
{
        break;
     }
  }
return 0;
}
```

continue statement



The continue statement in C programming works somewhat like the break statement. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between.For the for loop, continue statement causes the conditional test and increment portions of the loop to execute. For the while and do...while loops, continue statement causes the program control to pass to the conditional tests.

Syntax continue;

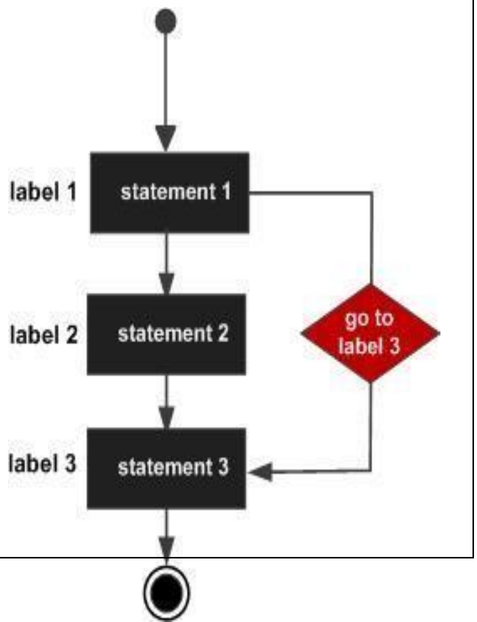
Example:

```
#include <stdio.h>
int main ()
{
  int a = 10;
  do
{
     if( a == 15)
{
        a = a + 1;
        continue;
     }
printf("value of a: %d\n", a);
     a++;
  } while( a < 20 );
  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: 16
value of a: 17
value of a: 18
value of a: 19
```

goto statement



A goto statement in C programming provides an unconditional jump from the 'goto' to a labelled statement in the same function.

NOTE – Use of goto statement is highly discouraged in any programming language because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. Any program that uses a goto can be rewritten to avoid them.

Syntax *goto label;*

```
goto label;
..
label: statement;
```

Example:

```
#include <stdio.h>
int main ()
{
  int a = 10;
LOOP:do
{
    if( a == 15)
{
      a = a + 1;
goto LOOP;
    }
printf("value of a: %d\n", a);
    a++;
  }while( a < 20 );
  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: 16
value of a: 17
value of a: 18
value of a: 19
```

The Infinite Loop

A loop becomes an infinite loop if a condition never becomes false. The for loop is traditionally used for this purpose. Since none of the three expressions that form the 'for' loop are required, you can make an endless loop by leaving the conditional expression empty.

Example:

```
#include <stdio.h>
int main ()
{
  for( ; ; )
  {
printf("This loop will run forever.\n");
  }
  return 0;
}
```

When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but C programmers more commonly use the for(;;) construct to signify an infinite loop.
NOTE − You can terminate an infinite loop by pressing Ctrl + C keys.

PRACTICE

In this program user is asked to enter the age and based on the input, the if..else statement checks whether the entered age is greater than or equal to 18. If this condition meet then display message “You are eligible for voting”, however if the condition doesn’t meet then display a different message “You are not eligible for voting”.

```
#include  <stdio.h>
#include <conio.h>
void main()
{
  int age;
clrscr();
printf("Enter your age:");
scanf("%d",&age);
  if(age >=18)
  {
      printf("You are eligible for voting");
  }
  else
  {
      printf("You are not eligible for
voting");
  }
getch();
}
```

Output:

```
Enter your age:14
You are not eligible for voting
```

Program to calculate the sum of first n natural numbers.
Positive integers 1,2,3...n are known as natural numbers.

```
#include <stdio.h>
#include <conio.h>
void main()
{
intnum, count, sum = 0;
```

Output:

```
Enter a positive integer: 10
Sum = 55
```

```
clrscr();
printf("Enter a positive integer: ");
scanf("%d", &num);
    for(count = 1; count <= num; ++count)
    {
        sum += count;
    }
printf("Sum = %d", sum);
getch();
}
```

Print numbers from 1 to 5

```
#include  <stdio.h>
#include <conio.h>
void main()
{
int i = 1;
clrscr();
    while (i<= 5)
    {
printf("%d\n", i);
        ++i;
    }
getch();
}
```

Output:

```
1
2
3
4
5
```

Print numbers from 1 to 5

```
#include  <stdio.h>
#include <conio.h>
void main()
{
int i = 1;
 clrscr();
     do
    {
printf("%d\n", i);
        ++i;
    }
    while (i<= 5)
getch();
}
```

Output:

```
1
2
3
4
5
```

Use of break in a for loop

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

Output:

```
#include <conio.h>
void main()
{
int num = 5;
clrscr();
while (num > 0)
{
  if (num == 3)
  {
break;
  }
printf("%d\n", num);
  num--;
}
getch();
}
```

```
5
4
```

Example: continue statement inside for loop- Value 4 is missing in the output, why? When the value of variable j is 4, the program encountered a continue statement, which makes the control to jump at the beginning of the for loop for next iteration, skipping the statements for current iteration (that's the reason printf didn't execute when j is equal to 4).

```
#include  <stdio.h>
#include <conio.h>
void main()
{
   for (int j=0; j<=8; j++)
   {
      if (j==4)
      {
            continue;
       }
printf("%d ", j);
   }
getch();
}
```

Output:

```
0 1 2 3 5 6 7 8
```

Example of goto statement- In this example, we have a label addition and when the value of i (inside loop) is equal to 5 then we are jumping to this label using goto. This is reason the sum is displaying the sum of numbers till 5 even though the loop is set to run from 0 to 10.

```
#include <stdio.h>
#include <conio.h>
void main()
{
int sum=0;
clrscr();
for(inti = 0; i<=10; i++)
{
```

Output:

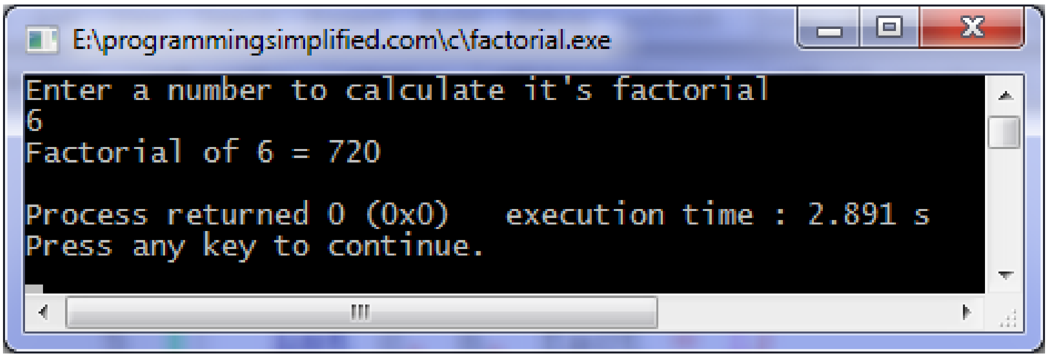
```
15
```

sum = sum+i;
if(i==5)
{
goto addition;
}
}
addition:
printf("%d", sum);
getch();
}	

Factorial in C using a for loop

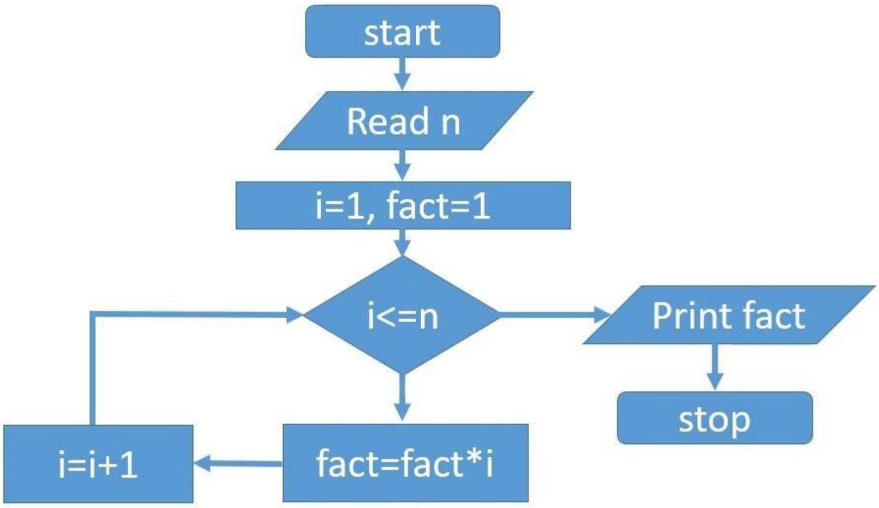
```
#include  <stdio.h>
#include <conio.h>
void main()
{
  int c, n, f = 1;
clrscr();
printf("Enter a number to calculate its factorial\n");
scanf("%d", &n);
  for (c = 1; c <= n; c++)
        {
        f = f * c;
        }
printf("Factorial of %d = %d\n", n, f);
getch();
}
```

Output:



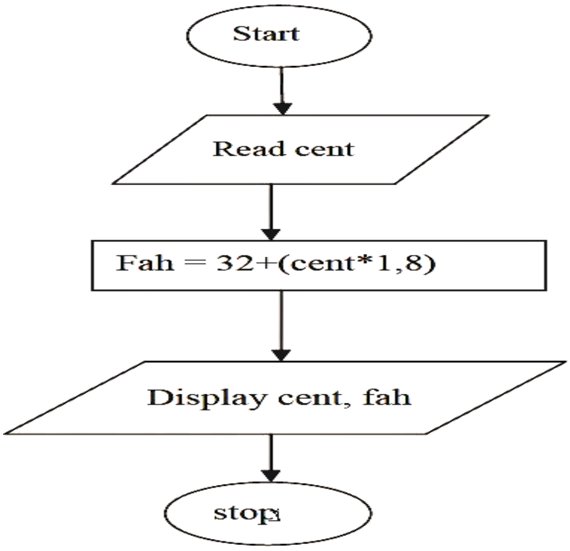
```
Enter a number to calculate it's factorial
6
Factorial of 6 = 720

Process returned 0 (0x0)   execution time : 2.891 s
Press any key to continue.
```

FLOWCHART FOR FACTORIAL OF A NUMBER



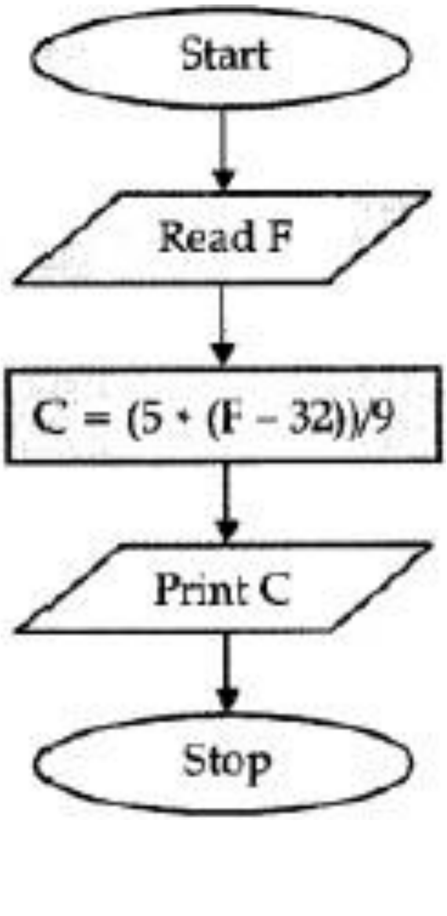
Convert temperature from Centigrade to Fahrenheit

```
#include<stdio.h>
#include <conio.h>
void main ()
{
        float temp_c, temp_f;
clrscr();
printf (“Enter the value of Temperature in Celcius: “);
scanf (“%f”, &temp_c);
temp_f = (1.8 * temp_c) + 32;
printf (“The Temperature in Fahreinheit is: %f”, temp_f);
getch();
}
```



Convert temperature from Fahrenheit to Celsius.

```
#include  <stdio.h>
#include <conio.h>
void main ()
{
   float celsius, fahrenheit;
clrscr();
printf("Enter temperature in Fahrenheit: ");
scanf("%f", &fahrenheit);
celsius = (fahrenheit - 32) * 5 / 9;
printf("%.2f Fahrenheit = %.2f Celsius", fahrenheit, celsius);
getch();
}
```



C Program to Check whether a number is Prime or not

```
#include  <stdio.h>
#include <conio.h>
void main ()
 {
int n, i, c = 0;
clrscr();
printf("Enter any number n:");
scanf("%d", &n);
  for (i = 1; i<= n; i++)
{
if (n % i == 0)
{
c++;
}
         }
  if (c == 2)
{
printf("n is a Prime number");
}
  else
{
printf("n is not a Prime number");
}
getch();
}
```

Output:

```
Enter any number n: 7
n is Prime
```

Calculate roots of a quadratic equation.

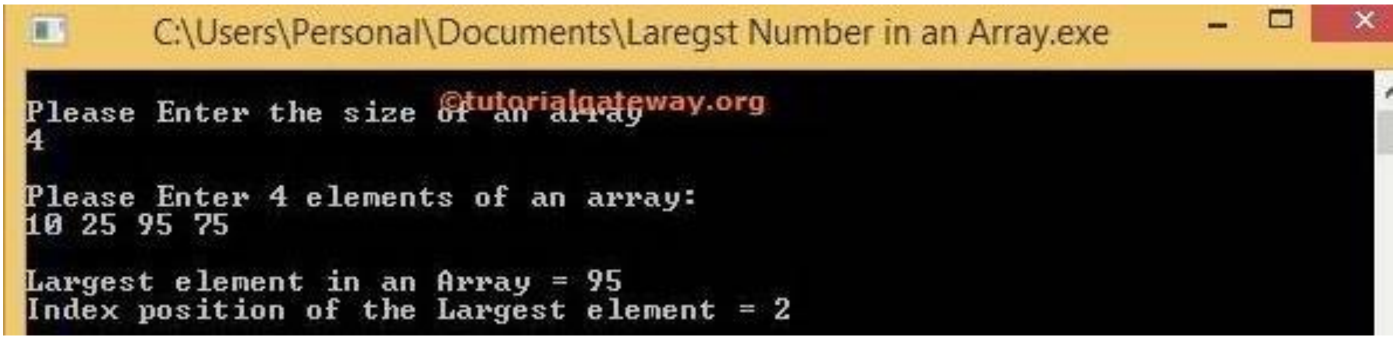
```
#include<stdio.h>
#include<math.h>
#include <conio.h>
void main ()
{
  float a,b,c;
  float d,root1,root2;
clrscr();
printf("Enter a, b and c of quadratic equation: ");
scanf("%f%f%f",&a,&b,&c);
  d = b * b - 4 * a * c;
  if(d < 0)
{
printf("Roots are complex number.\n");
printf("Roots of quadratic equation are: ");
printf("%.3f%+.3fi",-b/(2*a),sqrt(-d)/(2*a));
printf(", %.3f%+.3fi",-b/(2*a),-sqrt(-d)/(2*a));
}
  else if(d==0)
{
printf("Both roots are equal.\n");
        root1 = -b /(2* a);
printf("Root of quadratic equation is: %.3f ",root1);
}
  else
{
printf("Roots are real numbers.\n");
root1 = ( -b + sqrt(d)) / (2* a);
root2 = ( -b - sqrt(d)) / (2* a);
printf("Roots of quadratic equation are: %.3f , %.3f",root1,root2);
}
getch();
}
```

Output:

```
Enter a, b and c of quadratic equation: 2 4 1
Roots are real numbers.
Roots of quadratic equation are: -0.293, -1.707
```

C program to find the largest number in a given list and its location in the list.

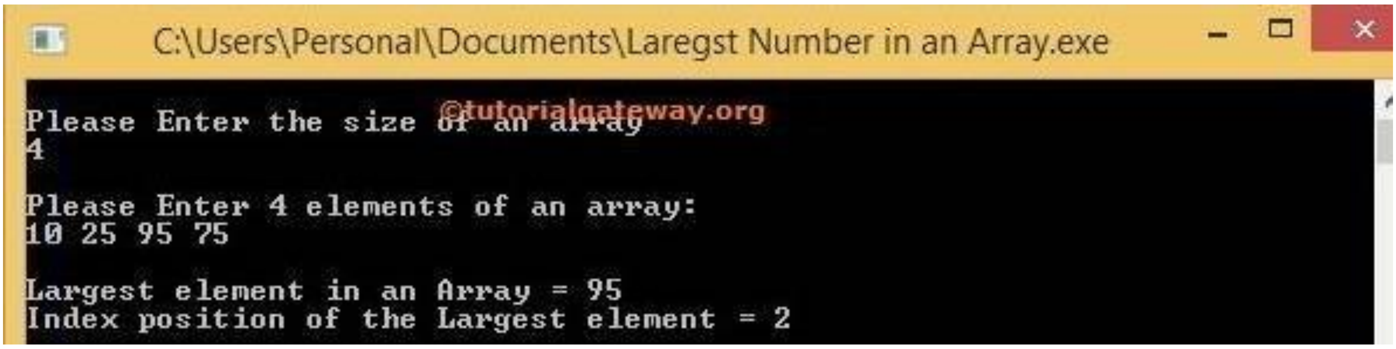
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10], Size, i, Largest, Position;
clrscr();
printf("\n Please Enter the size of an array \n");
scanf("%d",&Size);
printf("\n Please Enter %d elements of an array: \n", Size);
  for(i=0; i<Size; i++)
   {
scanf("%d",&a[i]);
   }
  Largest = a[0];
  for(i=1; i<Size; i++)
   {
    if(Largest<a[i])
     {
       Largest = a[i];
       Position = i;
     }
   }
printf("\n Largest element in an Array = %d", Largest);
printf("\n Index position of the Largest element = %d", Position);
getch();
}
```



```
Please Enter the size of an array
4

Please Enter 4 elements of an array:
10 25 95 75

Largest element in an Array = 95
Index position of the Largest element = 2
```

C program to find the smallest number in a given list and its location in the list.

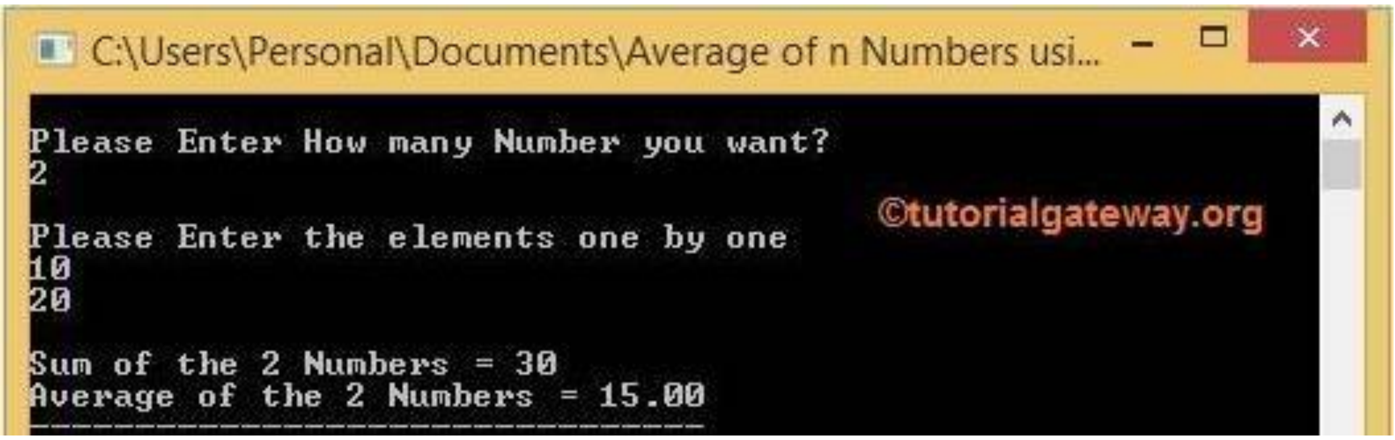
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10], Size, i, Smallest, Position;
clrscr();
printf("\n Please Enter the size of an array \n");
scanf("%d",&Size);
printf("\n Please Enter %d elements of an array: \n", Size);
  for(i=0; i<Size; i++)
   {
scanf("%d",&a[i]);
   }
  Smallest = a[0];
  for(i=1; i<Size; i++)
   {
    if(Smallest>a[i])
     {
       Smallest = a[i];
       Position = i;
     }
   }
printf("\n Smallest element in an Array = %d", Smallest);
printf("\n Index position of the smallest element = %d", Position);
getch();
}
```



```
Please Enter the size of an array
4

Please Enter 4 elements of an array:
10 25 95 75

Largest element in an Array = 95
Index position of the Largest element = 2
```

C program to find the sum and average of n numbers.

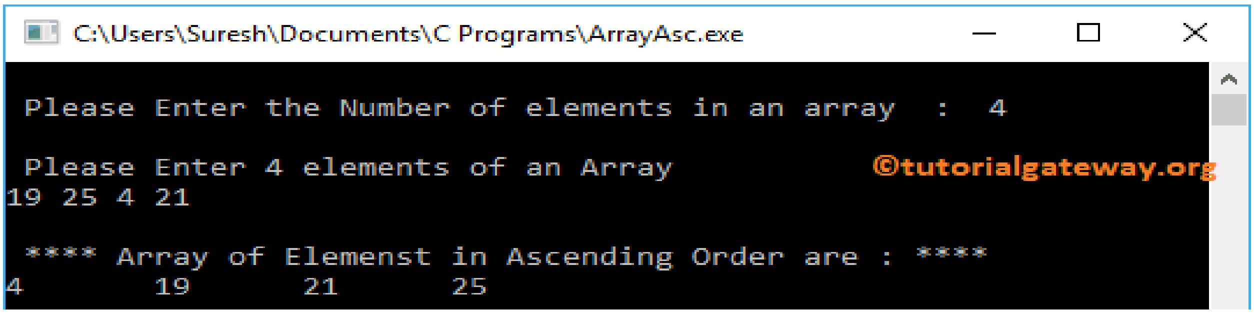
```
#include<stdio.h>
#include<conio.h>
void main()
{
inti, n, Sum=0, numbers;
  float Average;
clrscr();
printf("\n Please Enter How many Number you want? \n");
scanf("%d",&n);
printf("\n Please Enter the elements one by one \n");
  for(i=0; i<n; ++i)
   {
scanf("%d",&numbers);
     Sum = Sum+numbers;
   }
  Average = Sum/n;
printf("\n Sum of the %d Numbers = %d",n, Sum);
printf("\n Average of the %d Numbers = %.2f",n, Average);
getch();
}
```



```
Please Enter How many Number you want?
2

Please Enter the elements one by one
10
20

Sum of the 2 Numbers = 30
Average of the 2 Numbers = 15.00
```

C program to arrange numbers in ascending order.

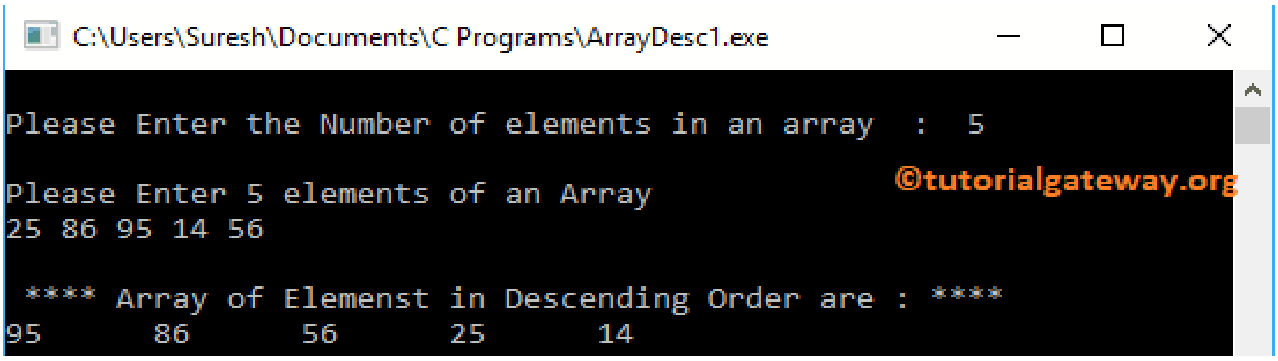
```
#include<stdio.h>
#include<conio.h>
void main()
{
        int Array[50], i, j, temp, Size;
        clrscr();
        printf("\n Please Enter the Number of elements in an array : ");
        scanf("%d", &Size);
        printf("\n Please Enter %d elements of an Array \n", Size);
        for (i=0; i<Size; i++)
        {
                scanf("%d", &Array[i]);
        }
        for (i=0; i<Size; i++)
        {
                for (j=i+1; j<Size; j++)
                {
                        if(Array[i] > Array[j])
                        {
                                temp = Array[i];
                                Array[i] = Array[j];
                                Array[j] = temp;
                        }
                }
        }
        printf("\n **** Array of Elemenst in Ascending Order are : **** \n");
        for (i=0; i<Size; i++)
        {
                printf("%d\t", Array[i]);
        }
        getch();
}
```



```
 Please Enter the Number of elements in an array  :  4

 Please Enter 4 elements of an Array
19 25 4 21

 **** Array of Elemenst in Ascending Order are : ****
4       19      21      25
```

C program to arrange numbers in descending order.

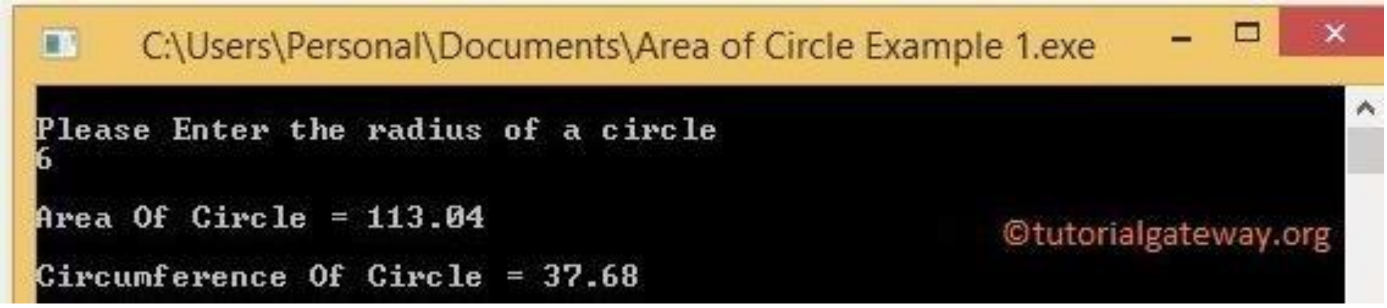
```
#include<stdio.h>
#include<conio.h>
void main()
{
        int Array[50], i, j, temp, Size;
        clrscr();
        printf("\n Please Enter the Number of elements in an array : ");
        scanf("%d", &Size);
        printf("\n Please Enter %d elements of an Array \n", Size);
        for (i=0; i<Size; i++)
        {
                scanf("%d", &Array[i]);
        }
        for (i=0; i<Size; i++)
        {
                for (j=i+1; j<Size; j++)
                {
                        if(Array[i] < Array[j])
                        {
                                temp = Array[i];
                                Array[i] = Array[j];
                                Array[j] = temp;
                        }
                }
        }
        printf("\n **** Array of Elemenst in Descending Order are : **** \n");
        for (i=0; i<Size; i++)
        {
                printf("%d\t", Array[i]);
        }
        getch();
}
```



```
Please Enter the Number of elements in an array  :  5

Please Enter 5 elements of an Array
25 86 95 14 56

 **** Array of Elemenst in Descending Order are : ****
95      86      56      25      14
```

C program to find area of circle.

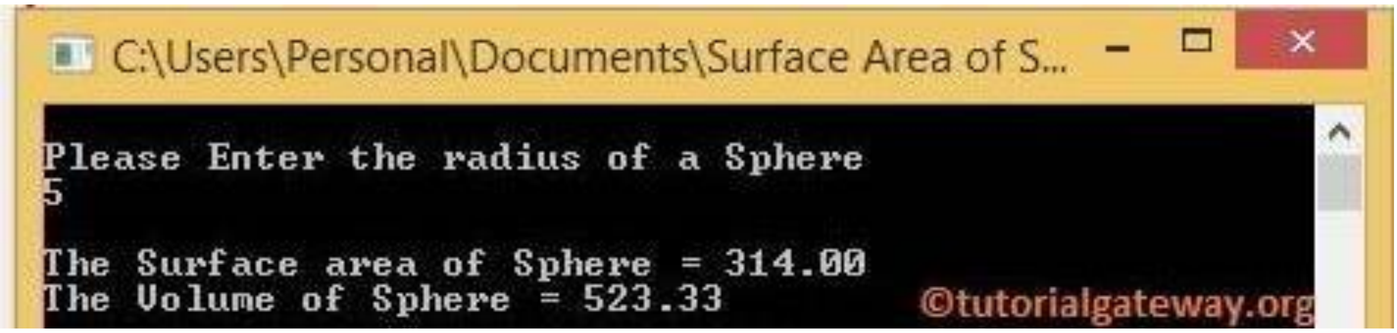
```
#include<stdio.h>
#include<conio.h>
#define PI 3.14
void main()
{
  float radius, area, circumference;
clrscr();
printf("\n Please Enter the radius of a circle\n");
scanf("%f",&radius);
  area = PI*radius*radius;
  circumference = 2*PI*radius;
printf("\n Area Of a Circle = %.2f\n", area);
printf("\n Circumference Of a Circle = %.2f\n", circumference);
getch();
}
```



```
Please Enter the radius of a circle
6

Area Of Circle = 113.04

Circumference Of Circle = 37.68
```

C program to find surface area and volume of sphere.

```
#include<stdio.h>
#include<conio.h>
#define PI 3.14
void main()
{
  float radius, sa, Volume;
clrscr();
printf("\n Please Enter the radius of a Sphere \n");
scanf("%f", &radius);
sa = 4*PI*radius*radius;
  Volume = (4.0/3)*PI*radius*radius*radius;
printf("\n The Surface area of a Sphere = %.2f", sa);
printf("\n The Volume of a Sphere = %.2f", Volume);
getch();
}
```



```
Please Enter the radius of a Sphere
5

The Surface area of Sphere = 314.00
The Volume of Sphere = 523.33
```

Short Question

1. What’s the constant?
2. What is variable?
3. What is operator?
4. Define Expression.
5. Define while loop.
6. Define For loop.

Long Questions

1. Explain the decision control and looping statement.
2. Discuss the different types of operator used in programming language.

CH-7 ADVANCED FEATURES OF C

PARAMETER PASSING TECHNIQUES IN C

In C we can pass parameters in two different ways. These are call by value, and call by reference.

Function call by Value in C

The call by value method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.By default, C programming uses call by value to pass arguments. In general, it means the code within a function cannot alter the arguments used to call the function. Consider the function swap() definition as follows.

```
void swap(int x, int y)
{
   int temp;
   temp = x; /* save the value of x */
   x = y;    /* put y into x */
   y = temp; /* put temp into y */
   return;
}
```

Now, let us call the function swap() by passing actual values as in the following example –

```
#include <stdio.h>
void swap(int x, int y);
int main ()
{
   int a = 100;
   int b = 200;
printf("Before swap, value of a : %d\n", a );
printf("Before swap, value of b : %d\n", b );
   swap(a, b);
printf("After swap, value of a : %d\n", a );
printf("After swap, value of b : %d\n", b );
   return 0;
}
```

Output:

```
Before swap, value of a :100
Before swap, value of b :200
After swap, value of a :100
After swap, value of b :200
```

It shows that there are no changes in the values, though they had been changed inside the function.

Function call by reference in C

The call by reference method of passing arguments to a function copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. It means the changes made to the parameter affect the passed argument.To pass a value by reference, argument pointers are passed to the functions just like any other value. So accordingly you need to declare the function parameters as pointer types as in the following function swap(), which exchanges the values of the two integer variables pointed to, by their arguments.

```
void swap(int *x, int *y)
{
   int temp;
   temp = *x;    /* save the value at address x */
   *x = *y;      /* put y into x */
   *y = temp;    /* put temp into y */
   return;
}
```

Let us now call the function swap() by passing values by reference as in the following example –

```
#include <stdio.h>
void swap(int *x, int *y);
int main ()
{
   int a = 100;
   int b = 200;
printf("Before swap, value of a : %d\n", a );
printf("Before swap, value of b : %d\n", b );
   swap(&a, &b);
printf("After swap, value of a : %d\n", a );
printf("After swap, value of b : %d\n", b );
   return 0;
}
```

Output:

```
Before swap, value of a :100
Before swap, value of b :200
After swap, value of a :200
After swap, value of b :100
```

It shows that the change has reflected outside the function as well, unlike call by value where the changes do not reflect outside the function.

C - SCOPE OF VARIABLES

A scope in any programming is a region of the program where a defined variable can have its existence and beyond that variable it cannot be accessed. There are three places where variables can be declared in C programming language −

- Inside a function or a block which is called local variables.
- Outside of all functions which is called global variables.
- In the definition of function parameters which are called formal parameters.

Local Variables

Variables that are declared inside a function or block are called local variables. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own. The following example shows how local variables are used. Here all the variables a, b, and c are local to main() function.

```
#include <stdio.h>
int main ()
{
  int a, b;
  int c;
  a = 10;
  b = 20;
  c = a + b;
printf ("value of a = %d, b = %d and c = %d\n", a, b, c);
  return 0;
}
```

Global Variables

Global variables are defined outside a function, usually on top of the program. Global variables hold their values throughout the lifetime of your program and they can be accessed inside any of the functions defined for the program.A global variable can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration. The following program show how global variables are used in a program.

```
#include <stdio.h>
int g;
int main ()
{
  int a, b;
  a = 10;
  b = 20;
  g = a + b;
printf ("value of a = %d, b = %d and g = %d\n", a, b, g);
  return 0;
}
```

A program can have same name for local and global variables but the value of local variable inside a function will take preference. Here is an example –

```
#include <stdio.h>
int g = 20;
int main ()
{
  int g = 10;
printf ("value of g = %d\n", g);
  return 0;
}
```

Output:

```
value of g = 10
```

Formal Parameters

Formal parameters, are treated as local variables with-in a function and they take precedence over global variables. Following is an example –

```
#include <stdio.h>
int a = 20;
int main ()
{
  int a = 10;
  int b = 20;
  int c = 0;
printf ("value of a in main() = %d\n", a);
  c = sum( a, b);
printf ("value of c in main() = %d\n", c);
  return 0;
}
int sum(int a, int b)
{
printf ("value of a in sum() = %d\n", a);
printf ("value of b in sum() = %d\n",  b);
   return a + b;
}
```

Output:

```
value of a in main() = 10
value of a in sum() = 10
value of b in sum() = 20
value of c in main() = 30
```

Initializing Local and Global Variables

When a local variable is defined, it is not initialized by the system, you must initialize it yourself. Global variables are initialized automatically by the system when you define them as follows –

Data Type	Initial Default Value
int	0
char	'\0'
float	0
double	0
pointer	NULL

It is a good programming practice to initialize variables properly, otherwise your program may produce unexpected results, because uninitialized variables will take some garbage value already available at their memory location.

C - STORAGE CLASSES

A storage class defines the scope (visibility) and life-time of variables and/or functions within a C Program. They precede the type that they modify. We have four different storage classes in a C program –

1. auto 2. register 3. static 4. extern

The auto Storage Class

The auto storage class is the default storage class for all local variables.

```
{
  int  mount;
  auto int month;
}
```

The example above defines two variables with in the same storage class. 'auto' can only be used within functions, i.e., local variables.

The register Storage Class

The register storage class is used to define local variables that should be stored in a register instead of RAM. This means that the variable has a maximum size equal to the register size (usually one word) and can't have the unary '&' operator applied to it (as it does not have a memory location).

```
{
  register int  miles;
}
```

The register should only be used for variables that require quick access such as counters. It should also be noted that defining 'register' does not mean that the variable will be stored in a register. It means that it MIGHT be stored in a register depending on hardware and implementation restrictions.

The static Storage Class

The static storage class instructs the compiler to keep a local variable in existence during the life-time of the program instead of creating and destroying it each time it comes into and goes out of scope. Therefore, making local variables static allows them to maintain their values between function calls.The static modifier may also be applied to global variables. When this is done, it causes that variable's scope to be restricted to the file in which it is declared.In C programming, when static is used on a global variable, it causes only one copy of that member to be shared by all the objects of its class.

Example:

```
#include <stdio.h>
void func(void);
static int count = 5;
main()
{
  while(count--)
{
func();
  }
return 0;
}
void func( void )
{
  static int i = 5;
i++;
printf("i is %d and count is %d\n", i, count);
}
```

Output:

```
i is 6 and count is 4
i is 7 and count is 3
i is 8 and count is 2
i is 9 and count is 1
i is 10 and count is 0
```

The extern Storage Class

The extern storage class is used to give a reference of a global variable that is visible to ALL the program files. When you use 'extern', the variable cannot be initialized however, it points the variable name at a storage location that has been previously defined.When you have multiple files and you define a global variable or function, which will also be used in other files, then extern will be used in another file to provide the reference of defined variable or function. Just for understanding, extern is used to declare a global variable or function in another file.The extern modifier is most commonly used when there are two or more files sharing the same global variables or functions as explained below.

First File: main.c

```
#include <stdio.h>
int count ;
extern void write_extern();
main()
{
   count = 5;
write_extern();
}
```

Second File: support.c

```
#include <stdio.h>
extern int count;
void write_extern(void)
{
printf("count is %d\n", count);
}
```

Here, extern is being used to declare count in the second file, where as it has its definition in the first file, main.c. Now, compile these two files as follows –

```
$gccmain.csupport.c
```

It will produce the executable program a.out. When this program is executed, it produces the following result −

```
count is 5
```

C - RECURSION

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();
}
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.

Types of Recursions:

Recursion are mainly of two types depending on whether a function calls itself from within itself or more than one function call one another mutually. The first one is called direct recursion and another one is called indirect recursion. Thus, the two types of recursion are:

1. Direct Recursion: These can be further categorized into four types:

a) Tail Recursion: If a recursive function calling itself and that recursive call is the last statement in the function then it’s known as Tail Recursion.After that call the recursive function performs nothing. The function has to process or perform any operation at the time of calling and it does nothing at returning time.

Example:
#include <stdio.h>
void fun(int n)
{
   if (n > 0)
{
printf("%d ", n);
       fun(n - 1);
   }
}
	int main()
	{
	   int x = 3;
	   fun(x);
	   return 0;
	}
		Output:
		3 2 1

b) Head Recursion: If a recursive function calling itself and that recursive call is the first statement in the function then it’s known as Head Recursion.There’s no statement, no operation before the call. The function doesn’t have to process or perform any operation at the time of calling and all operations are done at returning time.

Example:
#include <stdio.h>
void fun(int n)
{
   if (n > 0)
{
     fun(n - 1);
printf("%d ", n);
   }
}
	int main()
	{
	   int x = 3;
	   fun(x);
	   return 0;
	}
		Output:
		1 2 3

c) Tree Recursion: To understand Tree Recursion let’s first understand Linear Recursion. If a recursive function calling itself for one time then it’s known as Linear Recursion. Otherwise if a recursive function calling itself for more than one time then it’s known as Tree Recursion.

Example:

```
#include <stdio.h>
void fun(int n)
{
   if (n > 0)
{
printf("%d ", n);
      fun(n - 1);
      fun(n - 1);
   }
}
```

```
int main()
{
   fun(3);
   return 0;
}
```

Output:

```
3 2 1 1 2 1 1
```

d) Nested Recursion: In this recursion, a recursive function will pass the parameter as a recursive call.That means “recursion inside recursion”. Let see the example to understand this recursion.

Example:

```
#include <stdio.h>
int fun(int n)
{
   if (n > 100)
   return n - 10;
   return fun(fun(n + 11));
}
```

```
int main()
{
   int r;
   r = fun(95);
printf("%d\n", r);
   return 0;
}
```

Output:

```
91
```

2.Indirect Recursion: In this recursion, there may be more than one functions and they are calling one another in a circular manner.

Example:

```
#include <stdio.h>
void funB(int n);
void funA(int n)
{
   if (n > 0)
{
printf("%d ", n);
funB(n - 1);
   }
}
void funB(int n)
{
   if (n > 1)
{
printf("%d ", n);
funA(n / 2);
   }
}
```

```
int main()
{
funA(20);
   return 0;
}
```

Output:

```
20 19 9 8 4 3 1
```

Examples:

Number Factorial:

```
#include <stdio.h>
unsigned long longint factorial(unsigned inti)
{
  if(i<= 1)
  {
    return 1;
  }
  return i * factorial(i - 1);
}
int  main()
{
int i = 12;
printf("Factorial of %d is %d\n", i, factorial(i));
  return 0;
}
```

Output:

```
Factorial of 12 is 479001600
```

Fibonacci Series:

```
#include <stdio.h>
intfibonacci(inti)
{
   if(i == 0)
{
      return 0;
   }
   if(i == 1)
{
      return 1;
   }
   return fibonacci(i-1) + fibonacci(i-2);
}
int main()
{
inti;
   for (i = 0; i< 10; i++)
{
printf("%d\t\n", fibonacci(i));
   }
   return 0;
}
```

Output:

```
0
1
1
2
3
5
8
13
21
34
```

C - ARRAYS

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

One Dimensional Array

An one dimensional array consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

Declaring One Dimensional Array:

```
double balance[10];
```

Initializing One Dimensional Array:

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

Accessing One Dimensional Array Elements:

```
double salary = balance[9];
```

Example:

```
#include <stdio.h>
int main ()
{
  int n[ 10 ];
inti,j;
  for ( i = 0; i< 10; i++ )
{
    n[ i ] = i + 100;
  }
  for (j = 0; j < 10; j++ )
{
printf("Element[%d] = %d\n", j, n[j] );
  }
  return 0;
}
```

Output:

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

Multi-dimensional Array

C programming language allows 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.

Declaring Two-Dimensional Array:

```
type arrayName [ x ][ y ];
```

Initializing Two-Dimensional Array:

```
int a[3][4] ={{0, 1, 2, 3} , {4, 5, 6, 7} , {8, 9, 10, 11}};
```

Accessing Two-Dimensional Array Elements:

```
intval = a[2][3];
```

Example:

```
#include <stdio.h>
int main ()
{
  int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
inti, j;
  for ( i = 0; i< 5; i++ )
  {
    for ( j = 0; j < 2; j++ )
    {
printf("a[%d][%d] = %d\n", i,j, a[i][j] );
    }
  }
  return 0;
}
```

Output:

```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

C – STRINGS

Strings are actually one-dimensional array of characters terminated by a null character '\0'. 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";

Example:

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

Output:

```
Greeting message: Hello
```

C - 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. However, in this statement the asterisk is being used to designate a variable as a pointer. Take a look at some of the valid pointer declarations −

```
int    *ip;
double *dp;
float *fp;
char   *ch;
```

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

Example:

```
#include <stdio.h>
int main ()
{
   int  var = 20;
int  *ip;
ip = &var;
printf("Adrs of var variable: %x\n", &var);
printf("Adrs stored in ip variable: %x\n", ip);
printf("Value of *ip variable: %d\n", *ip);
   return 0;
}
```

Output:

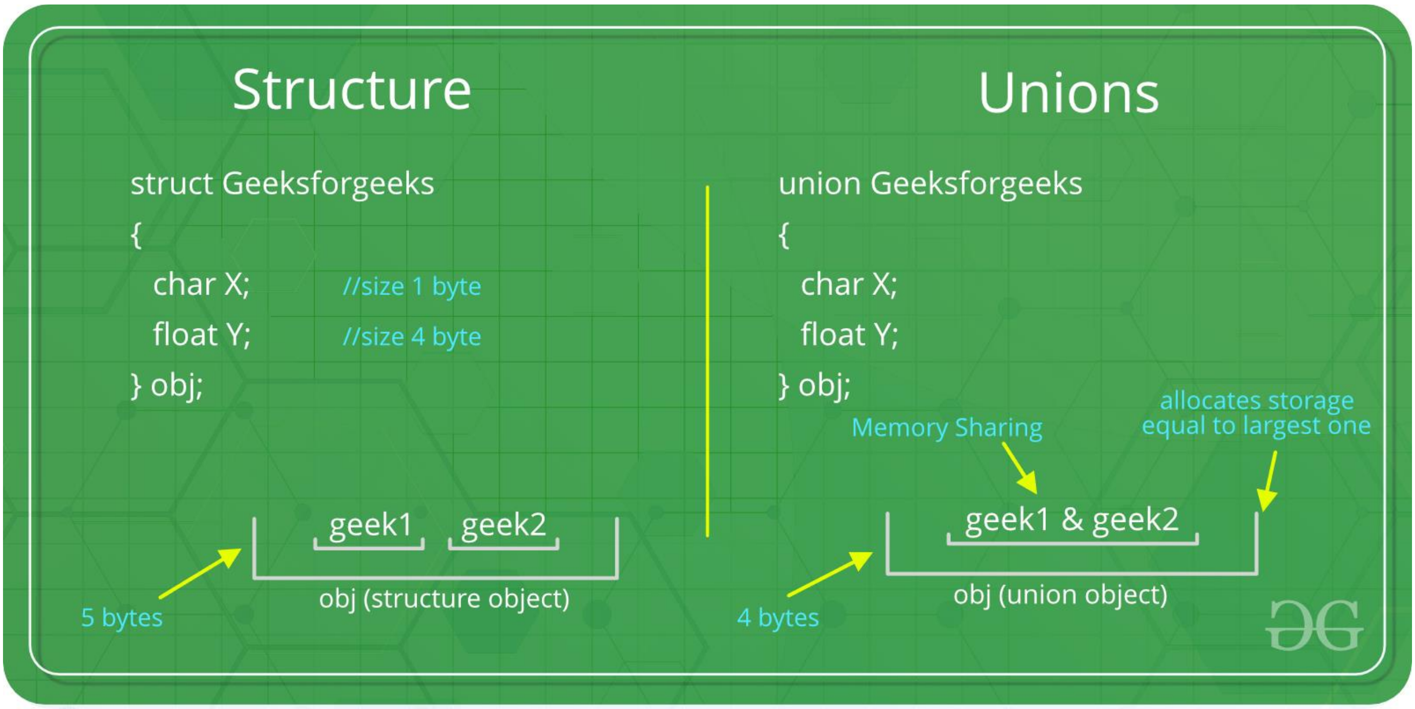
```
Adrs of var variable: bffd8b3c
Adrs stored in ip variable: bffd8b3c
Value of *ip variable: 20
```

STRUCTURES IN C

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

UNION IN C

Like Structures, union is a user defined data type. In union, all members share the same memory location.



```
struct Geeksforgeeks
{
   char X;         //size 1 byte
   float Y;        //size 4 byte
} obj;
```

```
union Geeksforgeeks
{
   char X;
   float Y;
} obj;
```

Short Questions

1. What is string?
2. What is Array?
3. Define call by value.
4. What is pointer?

Long Question

1. Differentiate between structure and union.
2. Differentiate between the call by value and call by reference.
3. What’s Array ? Discuss it’s types.

BEST OF

LUCK