



MODULE 1





CANARA ENGINEERING COLLEGE

Benjanapadavu, Bantwal Taluk - 574219

Department of Computer Science & Engineering



VISION

To be recognized as a center of knowledge dissemination in Computer Science and Engineering by imparting value-added education to transform budding minds into competent computer professionals.

MISSION

- M1.** Provide a learning environment enriched with ethics that helps in enhancing problem solving skills of students and, cater to the needs of the society and industry.
- M2.** Expose the students to cutting-edge technologies and state-of-the-art tools in the many areas of Computer Science & Engineering.
- M3.** Create opportunities for all round development of students through co-curricular and extra-curricular activities.
- M4.** Promote research, innovation and development activities among staff and students.

PROGRAMME EDUCATIONAL OBJECTIVES

- A. Graduates will work productively as computer science engineers exhibiting ethical qualities and leadership roles in multidisciplinary teams.
- B. Graduates will adapt to the changing technologies, tools and societal requirements.
- C. Graduates will design and deploy software that meets the needs of individuals and the industries
- D. Graduates will take up higher education and/or be associated with the field so that they can keep themselves abreast of Research & Development

PROGRAMME OUTCOMES

Engineering graduates in Computer Science and Engineering will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specific needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods, including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Select/Create and apply appropriate techniques, resources and modern engineering and IT tools, including prediction and modeling to complex engineering activities, taking comprehensive cognizance of their limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the relevant scientific and/or engineering practices.
9. **Individual and team work:** Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society-at-large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work as a member and leader in a team to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for and above have the preparation and ability to engage in independent and life-long learning in the broadcast context of technological changes.

PROGRAMME SPECIFIC OUTCOMES

1. **Computer System Components:** Apply the principles of computer system architecture and software to design, develop and deploy computer subsystem.
2. **Data Driven and Internet Applications:** Apply the knowledge of data storage, analytics and network architecture in designing Internet based applications.

MODULE	DETAILS	HOURS
I	Module 1: Introduction to computer Hardware and software: Computer generations, computer, types, bits, bytes and words CPU, Primary memory, Secondary memory, ports and connections, input devices, output devices, Computers in a network, Network hardware ,Software basics, software types. Overview of C: Basic structure of C program, executing a C program. Constant, variable and datatypes, Operators and expressions.	8
II	Module 2: Managing Input and output operations, Conditional Branching and Loops, Example programs, Finding roots of a quadratic equation, computation of binomial coefficients, plotting of Pascals triangle	8
III	Module 3: Arrays: Arrays(1-D,2-D),Character arrays and Strings, Basic Algorithms: Searching and Sorting Algorithms(Linear search, Binary search, Bubble sort and Selection sort).	8
IV	Module 4: User Defined Functions and Recursion. Example programs, Finding Factorial of a positive integers and Fibonacci series.	8
V	Module 5: Structures and pointers , preprocessor Directives	8
TOTAL HOURS		40

TEXT, REFERENCE & ADDITIONAL REFERENCE BOOKS:

	BOOK TITLE/AUTHORS/PUBLICATION
T-1.	E. Balaguruswamy, Programming in ANSI C, 7 th Edition, TataMcGraw-Hill
T-2.	Brian W. Kernighan and Dennis M. Ritchie, The 'C' Programming Language, Prentice Hall of India
R-1.	Sumitabha Das, Computer Fundamentals & C Programming, McGraw Hill Education.
R-2.	Gary J Bronson, ANSIC Programming, 4 th Edition, Ceneage Learning.
R-3	Deyand G bosh, Programming in C, 3 rd 'Edition, Oxford University Press.
AR-1	R S Bichkar, Programming in C, University Press,2012
WR-1	https://www.tutorialspoint.com/cprogramming/
WR-2	https://www.programiz.com/c-programming

COURSE OBJECTIVES:

1	To understand the basic concepts of computer, fundamentals of C and techniques of solving problems
2	To understand the different programming constructs like I/O operations, Conditional branching and loops
3	To understand and implement arrays with applications
4	To understand the types of functions and their implementations
5	To understand the use of pointers & structures with simple applications

COURSE OUTCOMES (COs):

SL. NO	DESCRIPTION	PO MAPPING
CO:1	Explain the computer hardware and software concepts of computer and fundamentals of 'C' language	1,2,5
CO:2	Construct programming solution using I/O operations Conditional branching and loops statements	1,2
CO:3	Explain and write programs using arrays and strings	1,2
CO:4	Explain and write programs using functions	1,2
CO:5	Define, describe, and explain a pointers, structures	1,2

Module-1

Table of contents	Page No.
1. The Background	2
2. Computer Generations	3
3. Computer Types	5
4. Bits, Bytes and Words	6
5. Inside the Computer	7
6. Primary Memory	9
7. Secondary Memory	11
8. I-O PORTS Or Communication ports	14
9. Input Devices	15
10. Output Devices	16
11. Computer in a Network	18
12. Computer Software Overview	22
13. Overview of C	27
14. Basic Structure of C	30
15. C Tokens	32
16. Data Types	39
17. Operators and Expressions	43
18. Type Conversions	48
19. Precedence and Associativity	50
Web Resources	52
Video Resources	53
Question Bank	54
University Questions	55

Chapter 1

INTRODUCTION

1.THE BACKGROUND

Today computers can be seen in schools, offices and almost everywhere. Computers have gone through many stages of evolution. Some of the early computing machines are as follows:

- Abacus emerged about 5000 years ago in Asia minor is considered as the first computer.
- In abacus computations are done by using a system of sliding beads arranged on a rack
- First mechanical adding device Pascaline was invented by Blaise Pascal in 1642. It used a base of 10 to perform calculations. Drawback was it could perform only addition.
- In 1694, a German mathematician and philosopher, Gottfried Wilhelm von Leibniz improved Pascaline by creating a machine that could perform both addition and multiplication. Leibniz mechanical multiplier worked by the system of gears and dial.

An English mathematics professor Charles Babbage developed a steam powered machine to solve differential equations, called Difference Engine. It could perform calculations and print results automatically.

- Charles Babbage developed Analytical Engine. In analytical engine punch cards were used to encode machine's instructions.
- Charles Babbage is also known as Father of Computers.
- In 1820 Joseph-Marie Jacquard invented Jacquard loom, which used punched boards to control the patterns to be woven.

- In 1889 an American inventor Herman Hollerith used loom concept of punch cards to store data, that was fed into a machine that compiled results mechanically.
- Herman Hollerith introduced punch cards to market by starting Tabulating Machine Company in 1896, which was later known as international Business Machines(IBM).

2. COMPUTER GENERATIONS

Generation: Computer generation means a step of advancement in technology. It also reflects the growth of computer industry.

First generation of computers(1945-1956):

- Mark-I an electronic relay computer.
- Electronic Numerical Integrator and Computer (ENIAC).
- Electronic Discrete Variable Automatic Computer(EDVAC).
- UNIVAC I (Universal Automatic Computer).

Features of First-Generation computers:

- Vacuum tubes were used for internal operations.
- Magnetic drums were used for memory.
- Punched cards were used for input and output.
- Low level languages were used for programming.
- The system was not powerful, huge, non portable and expensive.

Second Generation Computers(1956-1963):

- Stretch by IBM and LARC by Sperry-Rand were super computers of this generation made by using Transistor Technology.
- High level programming languages like COBAL(Common Business Oriented Language) and FORTRAN(Formula Translator) were used in this generation.

- Transistors replaced vacuum tubes in second generation
- Smaller magnetic cores replaced Magnetic drums
- Programmed using Symbolic or assembly language
- Other things were same as first generation, but less heat generated due to transistors.

Third Generation Computers(1964-1971):

Jack Kilby an engineer with Texas instruments, developed the Integrated Circuit (IC) in 1958. Operating Systems were developed which allowed to run multiple programs at once.

Features of Third Generation Computers:

- Integrated circuits (ICs) were used for computation.
- It consists of transistors, resistors, capacitors, and diodes in single chip.
- Smaller size, cheaper and energy efficient.
- Keyboard and monitors were used for input and output.
- Magnetic hard disk was used as storage.
- Operating systems came into use.
- Programming languages like BASIC, C, C++ were used.

Fourth Generation Computers(1971-present):

- Large Scale Integration(LSI) circuit integrated hundreds of components on a single chip.
- In 1980's Very Large Scale Integration(VLSI) circuit integrated thousands of components on a single chip.

Features of Fourth Generation Computers:

- More circuits on chip LSI, VLSI and ULSI.
- Microprocessors were introduced.
- Microcomputers and personal computers were affordable to common man.
- Use of chips for memory. Storage capacity was increased from MB to several GBs
- Operating systems like windows came into use. Networking technology developed
- Systems were smaller in size and easily portable.
- The systems were faster, more powerful, more reliable, cheaper, smaller in size and had more memory.

Fifth Generation Computers(Present and Beyond):

- Fifth generation computers will have Artificial Intelligence.
- HAL900 from Arthur C.Clarke's novel and 2001: a space Odyssey are examples of fifth generation computers.

Features of Fifth Generation Computers:

- Development of storage technology.
- Advancement in networking technology.
- Systems are more reliable, faster, and cheaper.
- Development of supercomputers.
- Concept of parallel processing in computers.

3. Computer Types

Micro Computers:

Microcomputers are the smallest category of computers fabricated using a microprocessor and other integrated circuits namely Ram and I/O interfaces. It can perform simple tasks like word processing or spreadsheet calculations. It is also known as Personal Computer (PC). The first PC was built by IBM.

Minicomputers:

Minicomputers are relatively fast but small computers with limited Input/output capabilities. These are designed for real time dedicated applications or multiuser applications. Minicomputer is a multiprocessor system. Digital alpha and sun ultra are some of the minicomputers.

Mainframes:

Mainframes are a large multiuser computer system designed to handle massive amounts of input, output, and storage. Applications that require high performance and large amount of data processing use mainframe computers. Example: IBM S/390. Mainframes are commonly used in Banking sector.

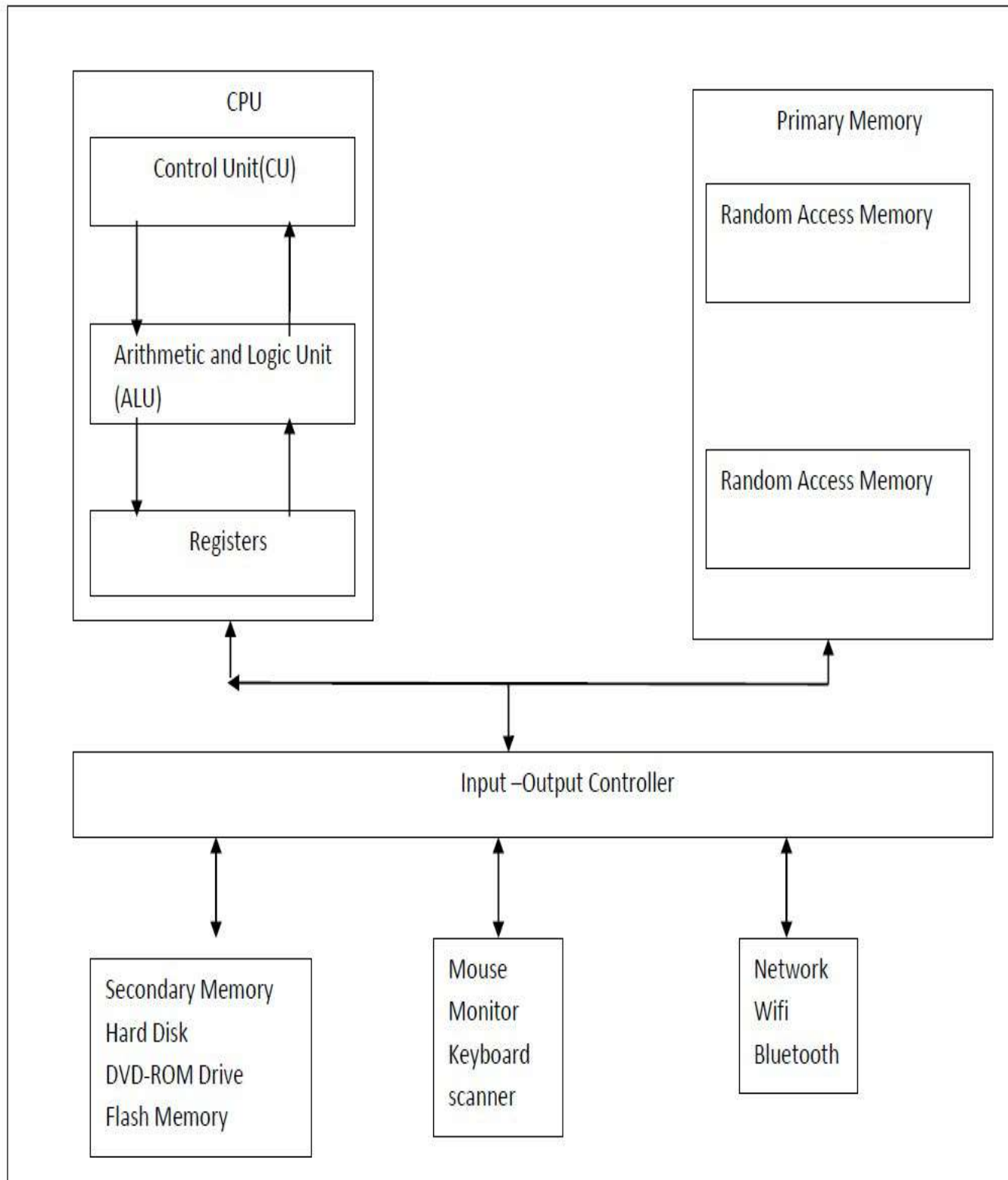
Super computers:

Super computers are the largest, fastest, and most powerful computers. These are expensive and have massive parallel processing capabilities. Example: IBM's deep blue. NASA uses a no. of super computers.

4. BITS, BYTES AND WORDS

Memory is an essential component of a computer. Memory is the only place in the computer where data or information can be stored. The storage capacity of a computer is measured in terms of bytes. One Byte includes a total of 8 individual units called as Bits. One bit can store either a 0 or a 1 in it.

5. INSIDE THE COMPUTER



Central Processing Unit(CPU):

The CPU is known as the brain of the computer. The CPU interprets the instructions in the program and executes them one by one and produces the desired result.

CPU has following three major parts:

- Control Unit.
- Arithmetic and Logic Unit.
- Registers

Control Unit: It controls and directs the transfer of data and program instructions between various units.

It performs four basic operations, they are:

- Fetches an Instruction.
- Decodes the instruction.
- Executes the instruction.
- Stores the result.

Arithmetic and Logic Unit (ALU): ALU performs arithmetic operations, logical operations, Relational operations and controls the speed of these operations.

Registers: They are high speed temporary storage units of CPU.

Memory Unit: Memory unit is the only storage area in a computer. Memory unit stores data, instructions, intermediate results, and final output generated. Secondary storage

devices are additional memory used to store data permanently ex: pen drives, Floppy disk etc.

Input Unit: Input unit of a computer accepts the data and instructions from the outside world or user. The input unit may consist of one or more input devices such as keyboard and mouse. Input unit converts the accepted data into binary form and sends it for processing.

Output Unit: The output unit displays the results in user understandable form. ex: monitor, printers etc.

6. PRIMARY MEMORY

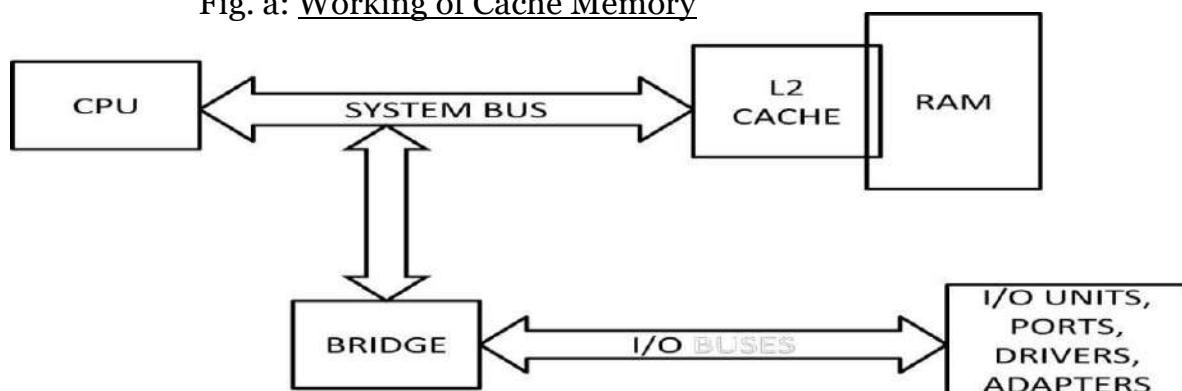
Primary memory performs the following functions:

- Maintains a copy of the operating system.
- Temporarily store a copy of application program that is currently being executed.
- Temporarily stores input data from input devices. Temporarily store results generated from processing.

6.1 Cache Memory

Cache memory is a small fast memory which increases the performance of the system by storing frequently used program code or data.

Fig. a: Working of Cache Memory



- The above fig shows the working of cache memory. When a program is running and CPU needs to read data or program instructions from RAM, it first checks the Cache Memory for the data. If data is not there in cache, the CPU will read the data from RAM into its registers and loads a copy into cache memory for future use.

6.2 Random Access Memory (RAM) is also referred to as main memory of the computer. The user can write Information into as well as read information from RAM. RAM is Volatile, that is, the information in RAM is retained as long as power supply is ON.

Types of RAM:

- **SRAM**: Static Random Access Memory is fast and expensive. It need not be refreshed. It has multiple transistors for each memory cell.
- **DRAM**: Dynamic Random Access Memory is slower than SRAM. It needs to be refreshed. It has single transistor and capacitor for each memory cell.
- **SDRAM**: Synchronous Dynamic Random Access Memory is a special type of DRAM that is synchronized with the system clock. it has better efficiency.
- **DDR-SDRAM**: In Double Data Rate SDRAM the data transfer rate is twice that of SDRAM. It works similar to SDRAM.

6.3 READ ONLY MEMORY(ROM)

- ROM stands for Read Only Memory.
- ROM is non-volatile i.e. The information is not lost even if power supply goes off.
- ROM is used for permanent storage of information.
- It also possesses random access property.

There are 3 different types of ROM they are:

i)PROM:

It stands for Programmable Read Only Memory. A special equipment called PROM programmer is used to store any information in PROM. User can write information into PROM only once, hence its Once programmable.

ii)EPROM:

It stands for Erasable ROM. The contents of EPROM are erased by Exposing it to Ultraviolet light for about 20 minutes. EPROM IC must be removed from the computer to erase contents and all the contents are erased at once. In EPROM we cannot erase selected Memory locations. It is cheaper and reliable.

iii)EEPROM:

It stands for Electrically Erasable PROM. electrical pulse is used to program EEPROM. EEPROM IC is not removed from the computer to erase and reprogram. In EEPROM we can erase selected Memory locations

6.4 Registers

They are high speed temporary storage units of CPU used to store instructions and intermediate data or results temporarily during processing.

7. SECONDARY MEMORY

Secondary Memory or magnetic memory is the memory used to store the information which is not currently used by the CPU. Secondary memory is non-volatile. It is used for bulk storage and has larger capacity than primary memory. Hard disks and floppy disks are commonly used secondary storage devices.

HARD DISK DRIVE:

- Hard disk is a secondary storage device used by the CPU to load programs and store data permanently.
- Faster Data transfer rate between hard disk and CPU.
- Hard Disk drive rotational speed is 7200 rpm.

Types of hard disks:

- SCSI HDD: These drives are of higher speed and are costlier. Used for server systems.
- IDE HDD: The IDE or ATA is commonly used standard in PC's and it uses same connection like CD or DVD drive.
- Serial ATA(SATA): These drives have large capacity, huge buffer, and faster data transfer rates.
- External HDD: These drives are connected to the computer through USB and have large storage capacity, normally used for Backup.

FLOPPY DISK:

Floppy diskette contains a single flat piece of circular plastic coated with metal oxide and enclosed in plastic cover. They are 3.5 ,5 1/4 and 8 inches in size. 3.5 inch floppy disk has a storage capacity of 1.44mb. Floppy disks are smaller in size, portable and cheaper.

OPTICAL DISKS:

Optical disk consists of a rotating disk, coated with a highly reflexive material. Data recording is done by focusing a laser beam on the surface of the spinning disk. The laser beam is turned on and off at varying rates this causes tiny holes to be burnt on the surface. A less powerful beam is focused on the surface to

read the data. The variation in the reflection from normal surface and pits is converted into electronic signals. CD-ROM is one of the commonly used optical disks.

CD-ROM:

- CD-ROM stands for Compact Disk Read Only Memory. It is an optical device.
- It is a read only storage medium capable of storing up to 682 MB of data.
- A device called CD-ROM drive is used to read the data in CD-ROM.
- A device called CD-Writer is used to erase or write information into CD-ROM.
- CD-R and CD-RW are two types of CD-ROM.
- CD-R: It is also called Recordable-CD, data is written once and cannot be erased but read many times.
- CD-RW: Its third generation CD technology introduced by Hp. data can be erased and Rewritten any number of times. It is also called as Rewriteable CD.

FLASH MEMORY

- This class of devices, having no moving parts, is based on the EEPROM. It is available in various forms pen drive, solid state disk (SSD) and the magnetic card (SD card). They are portable, need little power and are quite reliable.
- The memory stick or pen drive is the most common type of flash memory used on the computer. It connects to the USB port of computer where it is detected as yet another drive by the operating system.
- The SSD is a bigger device meant to replace traditional magnetic hard disk.
- The magnetic card is mainly used in cameras using adapters, they can connect to the USB port as well.

8. I-O PORTS or Communication ports

Port is a socket used to connect external devices to the computer.

The following are the types I-O ports:

Serial port: The serial port is an Asynchronous port which transmits data serially one bit at a time. It is used to attach mouse, modem and other small serial devices to the computer.

Parallel Port: Most printers use a special connector called a parallel port. Parallel port carry 8 bits of data at a time on parallel paths, hence it is faster way for computer to communicate with input and output device.

SCSI (Small Computer System Interface) Port: These connectors are costlier as they provide faster access at very high speed. These are typically 25 & 50 pin connectors. High performance hard disks and scanners use SCSI interface.

USB(Universal Serial Bus) port: USB is a Plug and Play interface between computer and add-on devices.

Game Port: It is usually fitted on to the sound cards and display cards. It is used to connect gaming devices like joysticks.

RJ45 port: This port is used by ethernet network. Both computer and the network device have the female form of the port.

PS/2 port: This port has replaced the serial port for connecting the keyboard and mouse. It has 6 pins but occurs in two different colours. The port connects keyboard are purple, green port for mouse.

HDMI (High Definition Multimedia Interface): This is standard for transferring audio and video data between computers and HDTVs, projectors and home theatres and many more.

9. INPUT DEVICES

Input devices are the devices attached to computer which are used to accept the data and instructions from the outside world or user. Keyboard and mouse are most commonly used input devices.

KEYBOARD: Keyboard is the most commonly used, simple to operate and cost-effective input device.

The computer Keyboard has 3 categories of keys:

- i)Alphanumeric keys.
- ii)Special keys.
- iii)Function keys.

Based on the number of function keys, keyboards are classified as:

- Regular with 84 keys or Enhanced with 101 keys.
- Multimedia keyboards.

MOUSE: MOUSE stands for Mechanically Operated User Serial Engine. It is an Input device which is Used as a Pointing Device. It was developed at Stanford Research Institute.

Scanner is a direct entry input device that can be moved over photograph or any document and it converts the data That has been scanned into digital format. It Eliminates duplication of data, reduces human efforts and improves accuracy and performance.

10. OUPUT DEVICES

Monitor or Visual display Unit:- Monitor is an integral part of computer's configuration. Programs use it to display text or graphical output, while users also need a monitor to view the input that they key in. The performance of a monitor is judged mainly by its image quality, resolution, energy consumption and its effect on the eyes.

Two types are:

CRT MONITOR - The CRT monitor uses a rarefied tube containing three electron guns and a screen coated with phosphorescent material. The guns emit electrons to create images on the screen by selectively lighting up the phosphors. They are large and heavy, energy inefficient and generate a lot of heat.

LCD MONITORS - Most of computers today uses LCD monitors (or their variants, LED and TFT). An LCD screen comprises thousands of liquid crystals, which by themselves, do not generate light but may allow or block the passage of light through them. An image is formed by selectively applying a voltage to these crystals and using a separate light source for light to pass through them. The backlight is provided by fluorescent light.

IMPACT PRINTERS

- Printer is hard copy output device that produces text and graphics on a physical medium like paper.
- The printers which use a print head mechanism that strikes an inked ribbon located between the print head and the paper print are called impact printers.

The following are the specifications of two commonly used impact printers:

Dot-matrix printer:

- An impact printer.
- print head has 9-24 pins.
- speed 30-300 characters per second.
- buffer size varies from 1k-64k.

Line printer:

- An impact printer
- Contains chain of characters or pins
- Low quality print
- prints entire line at a time
- High speed printer

NON IMPACT PRINTERS:

Non impact printers include all printers in which print head does not make contact with the paper and no inked ribbon is used to print.

The following are the specifications of different types of non impact printers:

Ink-jet printer:

- A non impact printer
- Prints high quality text, graphics and colour images.
- Speed 1-12 pages per minute
- Buffer size varies from 1Mb-4Mb
- Resolution 300 dots per inch

Laser:

- A non impact printer.
- Prints very high-quality text and graphics.
- Speed 4-24 pages per minute.
- Buffer size varies from 4Mb-32Mb.
- Resolution 600-1200 dots per inch.

Plotters

Plotters are a special kind of device, which are used to print large format images, such as engineering or construction drawings created in CAD systems.

The following are the different types of Plotters:

- Pen plotters.
- Ink jet plotters.
- Electro static plotters.
- Thermal plotters.

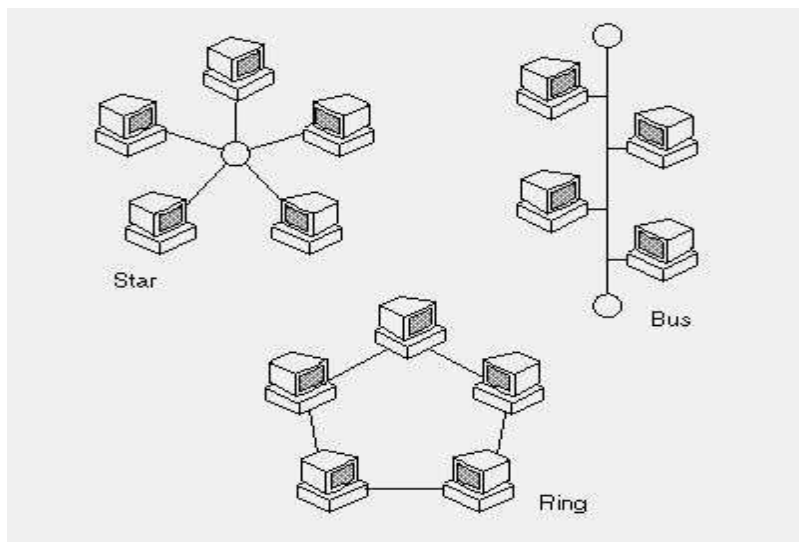
11. COMPUTERS IN A NETWORK

- A computer network, or data network, is a digital telecommunications network which allows nodes to share resources.
- In computer networks, computing devices exchange data with each other using connections (data links) between nodes.
- Apart from sharing files and programs, a network allows the sharing of scarce hardware resources by all connected computers. A printer can be used by several users if the printer itself is connected to one of the computers in the network.

- All the networked node needs a common addressing scheme that would ensure delivery of data to the right node. The scheme is used by most is Ethernet, a hardware standard that defines the addressing, cabling, and signalling requirements of the network.

NETWORK TOPOLOGY

There are number of ways or topologies of connecting computers each having its own merits and demerits.



- **bus topology:** All devices are connected to a central cable, called the bus or backbone. Bus networks are relatively inexpensive and easy to install for small networks. Ethernet systems use a bus topology.
- **star topology:** All devices are connected to a central hub. Star networks are relatively easy to install and manage, but bottlenecks can occur because all data must pass through the hub. This is not much of a problem anymore with the widespread deployment of switches.
- **ring topology:** All devices are connected to one another in the shape of a closed loop, so that each device is connected directly to two other devices, one on either side of it. Ring topologies are relatively expensive and difficult to install, but they offer high bandwidth and can span large distances.

- **Mesh topology:** All devices are connected to one another, offering a choice of multiple routes to travel. When node breaks down, the packet simply changes its route. This is most expensive of all the topologies.

NETWORK TYPES

Networks are classified based on their size. One way to categorize the different types of computer network designs is by their scope or scale. For historical reasons, the networking industry refers to nearly every type of design as some kind of area network.

Common types of area networks are:

- LAN - Local Area Network
- WAN - Wide Area Network
- WLAN - Wireless Local Area Network
- MAN - Metropolitan Area Network
- SAN - Storage Area Network, System Area Network, Server Area Network, or sometimes Small Area Network
- CAN - Campus Area Network, Controller Area Network, or sometimes Cluster Area Network
- PAN - Personal Area Network

LAN and WAN are the two primary and best-known categories of area networks, while the others have emerged with technology advances.

LAN: Local area network, or LAN, consists of a computer network at a single site, typically an individual office building. A LAN is very useful for sharing resources, such as data storage and printers. LANs can be built with relatively inexpensive hardware, such as hubs, network adapters and Ethernet cables.

WAN: A wide area network, or WAN, occupies a very large area, such as an entire country or the entire world. A WAN can contain multiple smaller networks, such as LANs or MANs. The Internet is the best-known example of a public WAN.

OTHER TYPES OF AREA NETWORKS

While LAN and WAN are by far the most popular network types mentioned, you may also commonly see references to these others:

- **Wireless Local Area Network** - A LAN based on Wi-Fi wireless network technology.
- **Metropolitan Area Network** - A network spanning a physical area larger than a LAN but smaller than a WAN, such as a city. A MAN is typically owned and operated by a single entity such as a government body or large corporation.
- **Campus Area Network** - A network spanning multiple LANs but smaller than a MAN, such as on a university or local business campus.
- **Personal Area Network** – This is the smallest network of all and has only recently come of age. A PAN operates within a range of a few meters. It connects small devices like cell phones and laptops with infrared or Bluetooth technology.

NETWORK HARDWARE

1. Network Interface Cards (NICs)

- Built into the motherboard of desktops and laptops
- Puts the data into packets and transmits packet onto the network.
- May be wired or wireless.

2. Hub

- An unintelligent network device that sends one signal to all the stations connected to it.
- All computers/devices are competing for attention because it takes the data that comes into a port and sends it out all the other ports in the hub.

3. Switch

- Split large networks into small segments, decreasing the number of users sharing the same network resources and bandwidth.
- Understands when two devices want to talk to each other and gives them a *switched* connection.

4. Bridge

- Connects two LANs and forwards or filters data packets between them.
- Creates an extended network in which any two workstations on the linked LANs can share data.

5. Router

- A device that connects any number of LANs.
- Uses standardized protocols to move packets efficiently to their destination.
- More sophisticated than bridges, connecting networks of different types.

12. Computer Software Overview

A computer system consists of hardware, the electronic devices capable of computing and manipulating information, and software that carries out predefined instructions to complete a given task. The combination of physical equipment (hardware) and logical instructions (software) gives power and versatility to the modern computing systems.

Software

Software is a collection of computer programs and related data that provide the instruction for telling a computer what to do and how to do it. A software is an interface between user and computer. It is a set of instructions, programs that are used to give command to hardware. It is responsible for controlling, integrating and managing the hardware components of a computer system and for accomplishing specific tasks.

Types of Software

Software can be divided into two major categories.

1. System Software.
2. Application Software.

System Software

- System software consists of several programs, which are directly responsible for controlling, integrating and managing the individual hardware components of a computer system.
- It also provides the interface between the user and component of the computer.
- The purpose of system software is to insulate the applications programmer as much as possible from the detail of the particular complex computer being used.
- Depending on the functionality, the system software can be further divided into two major categories; system management program and developing software.

Operating System:

- It consists of programs, which controls, which controls, coordinates, and supervises the activities of the various components of a computer system. Its function is to provide link between the computer hardware and the user.

- It performs all internal management functions (disk access, memory management, task scheduling and user interfacing) and ensures systematic functioning of a computer system. It provides an environment to run the programs. e.g., MS-DOS, windows XP/2000/98, Unix Linux, etc.

The operating system performs the following functions.

1. It recognizes input from keyboard, sends output to the display screen.
2. It makes sure that programs running at the same time do not interface with each other.
3. It is also responsible for security, ensuring that unauthorized users do not access the system

BIOS

The Basic Input / Output system (BIOS) is commonly known as System Bios. The BIOS controls various electronic components within the main computer system. The initial function of the BIOS is to initialize system devices such as the RAM, hard disk, CD/DVD drive, video display card and other hardware. The BIOS sets the machine hardware into a known state that helps the operating system to configure the hardware components. This process is known as Booting Up.

Device Drivers

A software, which is written with the objective of making a device functional when it is connected to the computer is called device driver. It is a system software that acts like an interface between the device and the user. Every device, whether it is a printer, monitor, mouse or keyboard has a driver program associated with it for its proper functioning.

- Device drivers are a set of instructions that introduce our PC to a hardware device.

- Device drivers are not independent programs, they assist and are assisted by the operating system for the proper functioning.

Programming Languages

A programming language is a primary interface of a programmer with a computer. A programming language is an artificial language to express computation that can be performed by a computer.

Each language has its own syntax i.e., the set of specific rules and expresses the logical steps of an algorithm. programming languages are divided into two categories: Low Level Language (LLL) and High-Level Language (HLL).

Low Level Language (LLL) Low level language is divided into two parts

1. Machine Language It is sometimes, referred to as machine code or object code. It is a collection of binary digits or bits that computer reads and interprets.
2. Assembly Language It is used to interface with computer hardware. It uses instructed commands as substitutions for numbers allowing human to read the code more easily than binary. It uses English –like representation to write a program.

Language Translator:

A language translator helps in converting programming languages to machine language. The translated program is called the object code. There are three different kinds of language translator.

1. *Assembler* It is used to convert the assembly language into machine language (i.e., 0 or 1), This language consists of mnemonic codes which are difficult to learn and is machine dependent.

2. **Compiler** It is used to convert the source code (written in high level language) into machine language. Compiler reads whole source code at a time and trap the errors and inform to programmer. For each high level language, the machine requires a separate compiler.
3. **Interpreter** This language processor converts a high level language program into machine language by converting it line-by-line. If there is any error in any line during execution, it will report it at the same time and cannot resume until the error is rectified.

Linker:

A linker is a system program that links together several object modules and libraries to form a single and coherent program (executable). The main purpose of linker is to resolve references among files. Linker is used to determine the memory locations that code from each module will occupy and relates its instruction by adjusting absolute references.

Loader:

Loader is a kind of system software, which is responsible for loading and relocation of the executable program in the main memory. It is a part of operating system that brings an executable file residing on disk into memory and starts its execution process.

Application Software

Application software is a computer software designed to help the user to perform singular or multiple tasks. It is a set of instructions or program designed for specific uses or applications, that enable the user to interact with a computer. Application software are also called the end-user programs. These programs do the real work for users. Ex: Office Software, Database Software, Entertainment software.

13. Overview of C

C is a programming language developed at AT & T's Bell laboratories of USA in 1972. It was designed and written by a system programmer Dennis Ritchie. The main intention was to develop a language for solve all possible applications.

Characteristics of C language as follows:

- C is a robust language, which consists of number of built-in functions and operators can be used to write any complex program
- Programs written in c are executed fast compared to other languages.
- C language is highly portable
- C language is well suited for structured programming.
- C is a simple language and easy to learn.

1. Algorithm: Algorithm is a step by step procedural description of the programming logic where each step is numbered in a hierarchical order. In algorithm, number of steps must be finite and every step must be complete and error free. The steps must specify the input and output as per the requirement of the user.

2. Characteristics of Algorithm: Finiteness, Definiteness, Inputs, Outputs and Effectiveness

3. Flowchart: Flow chart is a graphical representation of an algorithm. Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them. In flow chart symbols or shapes are used to represent the data flow, operations, process, and recourses used to complete a given task.

4. Pseudocode solution to Problem: Pseudocode is a problem-solving tool which consists of statements written in English and C language. It is a series of steps which describes what must be done to solve a given problem. It is used for developing a program and it is constructed before writing a program. It is an informal high-level description of program mixed with natural descriptions and mathematical notation. It does not have any standard syntax.

History of C



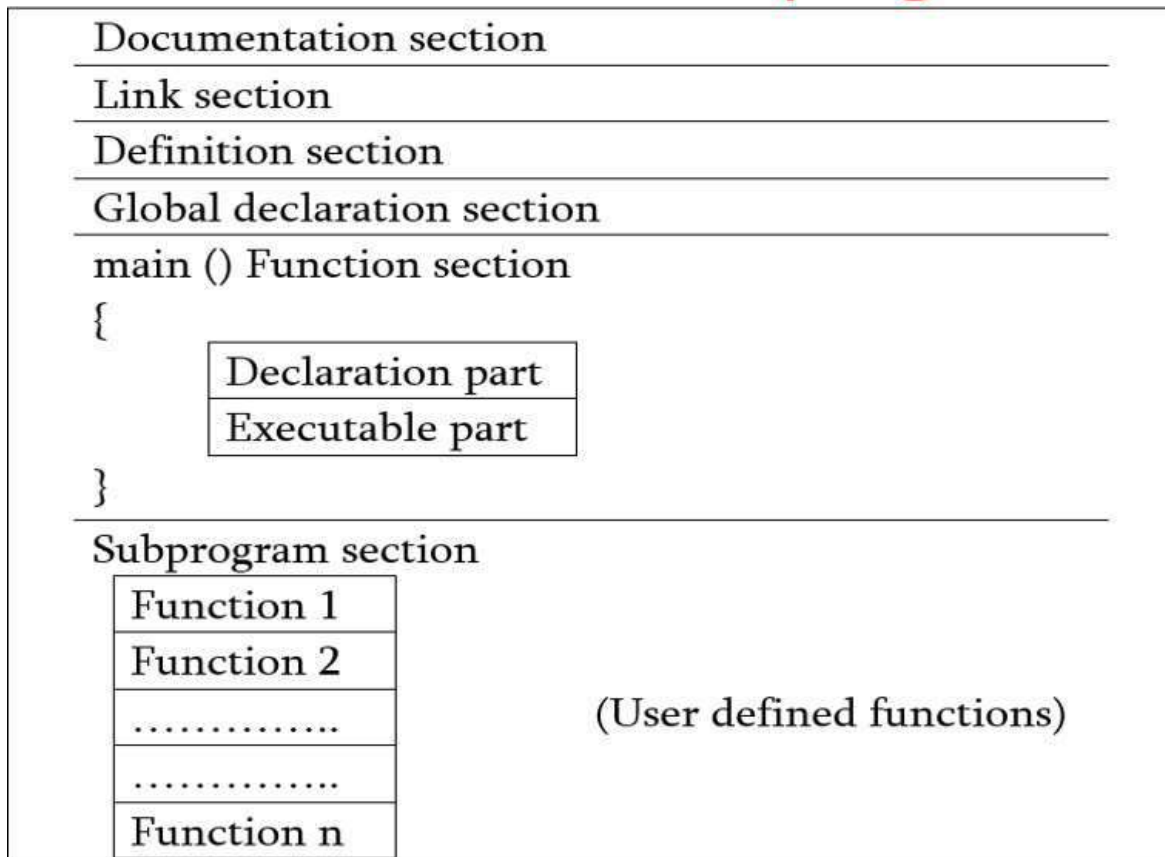
- The root of all modern languages is ALGOL, introduced in the early 1960s. ALGOL(Algorithmic Language) was the first computer language to use a block structure. ALGOL gave the concept of structured programming to the computer science community.

- In 1967, Martin Richards developed a language called BCPL(Basic Combined Programming Language) primarily for writing system software.
- In 1970, Ken Thompson created a language using many features of BCPL and called it simply B. B was used to create early versions of UNIX operating system at Bell Laboratories. Both BCPL and B were “typeless” system programming languages.
- C was evolved from ALGOL,BCPL and B by Dennis Ritchie at the Bell Laboratories in 1972. C uses many concepts from these languages and added concepts of data types and other powerful features. Since it was developed along with the UNIX operating system, it is strongly associated with UNIX. This operating system, which was also developed at Bell Laboratories, was coded almost entirely in C. UNIX is one of the most popular network operating systems in use today and the heart of the Internet data superhighway.
- During 1970s, C had evolved into what is now known as “traditional C”, The language became more popular after publication of the book ‘The C Programming Language’ by Brian Kerningham and Dennis Ritchie in 1978. The book was so popular that the language came to be known as “K&R C” among the programming community.
- To assure that the C language remains standard, in 1983, American National Standards Institute (ANSI) appointed a technical committee to define a standard for C. the committee approved a version of C in December 1989 which is known as ANSI C. It was then approved by the International Standards Organization (ISO) in 1990. C99 is a past work version of C language.

Importance of C

- It is a robust language whose rich set of built – in functions and operators can be used to write any complex program.
- Programs written in C are efficient and fast.
- C is highly portable. This means that C programs written for one computer can be run on another with little or no modification.
- C has its ability to extend itself.

14. Basic Structure of C Program



The **documentation section** consists of a set of comment lines giving the name of the program, the name author and other details which the programmer would like to use later. The **link section** provides instructions to the compiler to link functions from the system library. The **definition section** contains all symbolic constants. There are some variables that are used in more than one function. Such variables are called global variables and are declared in the **global declaration** section.

Every C program must have one **main()** function section. This section contains two parts: **Declaration and Executable part**.

The **declaration part** declares all the variables used in the executable part. There should be at least one statement in the executable part. These two parts must appear between the opening '{' and '}' closing braces. The program execution begins at the opening brace and ends at the closing brace.

The closing brace of the main function section is logical end of the program. All statements in the declaration and executable parts end with a semicolon. **The subprogram section** contains all the user-defined functions that are called in the main function. The main function is very important compared to other sections.

Sample C program:

*/*Program Prog1.c: Program to display a message*/*

```
#include<stdio.h>
void main()
{
    printf("Hello");
}
```

Output:

Hello

Character Set

The characters that can be used to form words, numbers and expressions depend upon the computer on which the program is run. The characters in C are grouped into the following categories.

- Letters- (A-Z a-z)
- Digits- (0-9)
- Special characters
- White Spaces (Blank Space, horizontal tab, carriage return, new line, form feed)

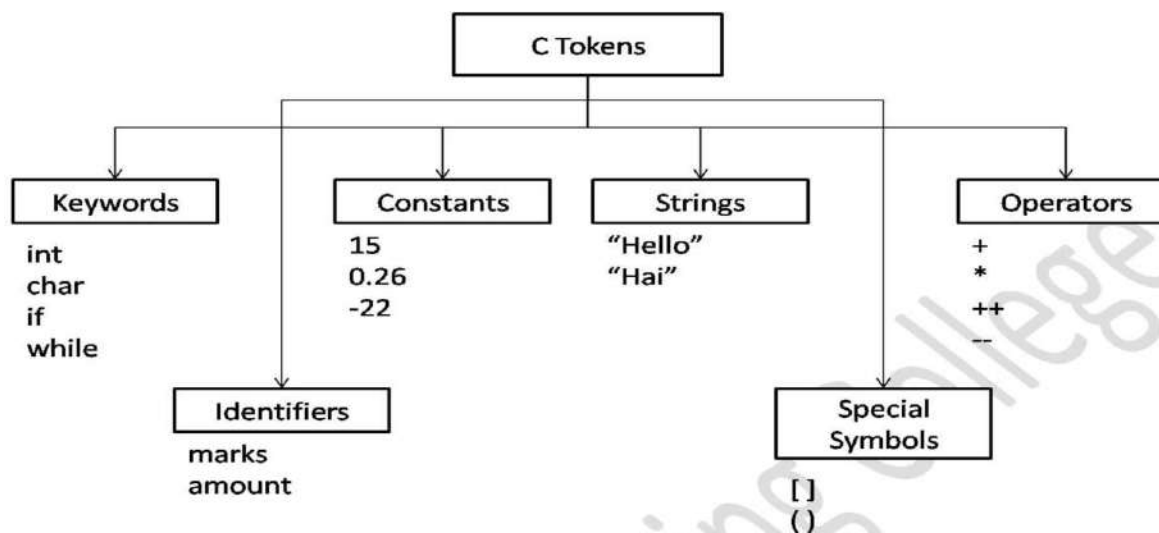
Character	Name	Character	Name
,	Comma	&	Ampersand
.	Period	^	Caret
;	Semicolon	*	Asterisk
:	Colon	-	Minus
?	Question mark	+	Plus
'	Single quote	=	Equal
"	Double quote	<	Less than
!	Exclamation	>	Greater than
	Vertical bar	(Left parenthesis
/	Slash)	Right parenthesis
\	Back slash	[Left square bracket
~	Tilde]	Right square bracket
_	Underscore	{	Left curly bracket
\$	Dollar	}	Right curly bracket
%	Percentage	#	Hash-Number sign

Special character set in C

15. C Tokens

Smallest unit of a program is called as **tokens**.

Types of tokens in C:



Keywords

- Keywords have fixed meanings and these meanings cannot be changed.
- These serve as basic building blocks for program statements. All keywords must be written in lowercase.

The list of all ANSI C keywords is listed below.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Identifiers

- Identifiers are names given to different entities such as variables, structures, functions etc.
- Rules to name an identifier
 - An identifier can only have alphanumeric characters (a-z, A-Z, 0-9) (i.e. letters & digits) and underscore(_) symbol.
 - Identifier names must be unique
 - The first character must be an alphabet or underscore.
 - Cannot use a keyword as identifiers.
 - Only first thirty-one (31) characters are significant.
 - Must not contain white spaces.
 - Identifiers are case-sensitive.

Constants

- **Constants** refer to fixed values that the program may not alter during its execution. These fixed values are also called literals.

Constants can be of any of the basic data types like:

- ▢ Integer constant
- ▢ Real (Floating) constant
- ▢ Character constant
- ▢ String literal

Integer Constants

An integer constant refers to a sequence of digits. There are three types of integers, namely decimal, octal and hexadecimal. Decimal integers consist of a set of digits 0 through 9, preceded by an optional – or + sign.

Some examples of decimal integer constants are:

123
-431
0
34567
+678

Spaces, commas, and non-digit characters are not permitted between digits. For example:

15 750
20,000
Rs 1000

An octal integer constant consists of any combination of digits from the set 0 through 7 with a leading 0. Some examples are:

037
0
0435
0567

A sequence of digits preceded by `0x` is considered as hexadecimal integer. They may also include alphabets A through F or a through F. The letters A through F represent the numbers 10 through 15.

The examples for hexadecimal integers are: `0x2`

`0x9`

`F`

`0xbc`

`d 0x`

Floating-point constants

Integer numbers are inadequate to represent quantities that vary continuously, such as distances, heights, temperatures, prices and so on. These quantities are represented by numbers containing fractional parts like 23.78. Such numbers are called floating-point constants or real constants. Examples for floating-point constants are given below.

`213.`

`.95`

`-.71`

`+.5`

A real number may also be expressed in exponential (or scientific notation). For example the value 213.45 may be written as $2.1354e^2$ in exponential notation. e^2 means multiply by 10^2 . The general form is mantissa e exponent.

Character Constants

A character constant contains a single character enclosed within a pair of single quote marks. Examples of character constants are: `'5'` `'X'` `';` `' '`

Note that the character constant `'5'` is not the same as the number 5. The last constant is a blank space. Character constants have integer values known as ASCII values.

For Example:

```
printf ("%d", 'A'); // prints the number 65(ASCII value of 'A')
```

```
printf ("%c", 65); //prints letter 'A'
```

It is also possible to perform arithmetic operations on character constants.

Backslash Character Constants

C supports some special backslash character constants that are used in output functions. For example, the symbol '\n' stands for new line character. The below table gives a list of backslash constants.

Constant	Meaning
\a	Audible alert(bell)
\b	Back space
\f	Form feed
\n	New line character
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\'	Single quote
\"	Double quote
\?	Question mark
\\	Backslash mark
\0	Null character

Note that each one of them represents one character, although they consist of two characters. These character combinations are called escape sequences.

String constants

A string constant is a sequence of characters enclosed in double quotes. The letters may be numbers, special characters and blank space.

Examples

“THANK YOU” “2345” “?.....” “7+8-9” “X”

Remember that a character constant ‘X’ is not equivalent to the single character string constant (“X”). A single character string constant does not have an equivalent integer value as a single character constant. These types of constants are used in programs to build meaningful programs.

Meaning of variables

A variable is a data name that may be used to store a data value. Unlike the constants that remain unchanged during the execution of a program, a variable may take different values at different times during execution. A variable name can be chosen by the programmer in a meaningful way to reflect its function or nature in the program. Some examples are given below.

Average

Height

Total

Counter

_1

Rules for defining variables

Variable names may consist of letters, digits, and the underscore (_) character, subject to the rules given below:

1. The variables must always begin with a letter. Some systems permit underscore as the first character.
2. ANSI standard recognizes a length of 31 characters. However, the length should not be normally more than eight characters. Since first eight characters are treated as significant by many compilers.
3. Uppercase and lowercase are significant. That is, the variable Rate is not the same as rate or TOTAL.

4. The variable name should not be a keyword.
5. White space is not allowed.

Valid variable Names	Invalid Names
ph_value	345
student1	%
Name	(rate)
Distance	1 student

Variable Definition

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.

Declaration of variables

Declaration should appear at the beginning of a program before the variable names are used.

Syntax:

data_type variable_name;

where **data_type** can be any basic or derived datatypes like int, float, short, char etc. **variable_name** can be any valid identifier.

Ex:

```
int number;
float rate;
char
grade;
```

When a variable name is declared then a memory location is identified and given this name.

Variable Initialization

The process of assigning initial values to a variable during declaration is called as variable initialization.

Syntax:

datatype variable1=value;

Ex: We can initialize a variable called **num1** of type **int** with the value **10** as follows:

int num1=10;

We can initialize two variables with names **length** and **breadth**, both of type **float** with values **15** and **20** respectively as:

float length=15, breadth=20;

16. Data Types

C language has varieties of data types. Storage representations and machine instructions differ from machine to machine. The fundamental or primary data types are:

- Integer (int)
- Character(char)
- Floating point(float)
- Double-precision floating point (double).

Many of them also has extended data types such as long, double, short, unsigned and signed. The range of basic data types are given below:

Data types	Range of values
char	-128 to 127
int	-32,768 to 32,767
float	3.4e-38 to 3.4e+38
double	1.7e-308 to 1.7e+308

char, *int*, *float* and *double* are all keywords and therefore their use is reserved. They may not be used as names of variables. *char* stands for “character” and *int* stands for “integer”. The keywords *short int*, *long int* and *unsigned int* may be and usually are, shortened to just *short*, *long*, and *unsigned*, respectively. Also, *double* and *long float* are equivalent, but *double* is the keyword usually used.

Integer Types

- Integers are whole numbers with a range of values supported by a particular machine.
- Generally integers occupy one word of storage. If we use a 16 bit word length, the size of the integer value is limited to the range -32768 to +32767.
- A signed integer uses one bit for sign and 15 bits for the magnitude of the number.
- Similarly, a 32 bit word length can store an integer ranging from -2,147,483,648 to 2,147,483,647.
- C has three classes of integer storage both *unsigned* and *signed* forms.
 - **short int**
 - **int**
 - **long int**
- For example, *short int* represents fairly small integer. Values and requires half the amount of storage as a regular *int* number uses.
- Unlike signed integers, unsigned integers use all the bits for the magnitude of the number and are always positive.
- Therefore, for a 16 bit machine, the range of unsigned integer numbers will be from 0 to 65,535. We declare long and unsigned integers to increase the range of values.

Floating point types

- Floating point (or real) numbers are stored in 32 bits (on all 16 bit and 32 bit machines), with 6 digits of precision.
- Floating point numbers are defined in C by the keyword **float**.
- The type **double** can be used when the accuracy provided by a *float* number is not sufficient.
- A *double* data type number uses 64 bits giving a precision of 15 digits. These are known as double precision numbers.
- To extend the precision further, we may use **long double** which 80 bits.

Character types

- A single character can be defined as a character (**char**) type data.
- Characters are usually stored in 8 bits (one byte) of internal storage. The qualifier signed or unsigned may be explicitly applied to char.
- While unsigned chars have values between 0 and 255, signed chars have values from -128 to 127.

Size and Range of Data Types on a 16-bit Machine

Type	Size (bits)	Range
char or signed char	8	-128 to 127
unsigned char	8	0 to 255
int or signed int	16	-32,768 to 32,767
unsigned int	16	0 to 65535
short int or signed short int	8	-128 to 127
unsigned short int	8	0 to 255
long int or signed long int	32	-2,147,483,648 to 2,147,483,647
unsigned long int	32	0 to 4,294,967,295
float	32	3.4E - 38 to 3.4E + 38
double	64	1.7E - 308 to 1.7E + 308
long double	80	3.4E - 4932 to 1.1E + 4932

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 – auto, register, static and extern

The auto Storage Class

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

```
void fun()
{
    int mount;
    auto int month;
}
```

The example above defines two variables within 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

register int miles;

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.

static int x;

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.

extern long total;

Defining Symbolic Constants

- **#define** pre-processor directives are used to define symbolic constants.
- The #define directive format is

#define *symbolic_name replacement-value*

When this line appears in a file, all subsequent occurrences of identifier that do not appear in string literals will be replaced by replacement-text automatically before the program is compiled.

Example: `#define PI 3.1415`

The value 3.1415 is replaced in every occurrence of symbolic constant PI in the program.

Declaring a variable as constant

- A constant variable remains constant during the execution of the program.
- *const* data type qualifier is used to declare a variable as constant.

```
const int class_size = 60;
```

- This tells the compiler that the value of *class-size* must not be modified by the program.

Declaring a variable as volatile

ANSI standard defines another qualifier '**volatile**' that tells the compiler that variable's value may be changed at any time by external sources (from outside the program). For example,

```
volatile int date;
```

volatile variable can be modified by the program itself. This can be prevented by using *const* qualifier as follows:

```
volatile const int location = 100;
```

17. Operators and Expressions

Arithmetic Operators

The binary arithmetic operators are +, -, *, / and the modulus operator %. The % operator cannot be applied to float or double. The arithmetic operators *, / and % have the higher precedence than the + and – operators.

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.	$B \% A = 0$

Relational Operators

The relational operators are: `==` `!=` `>` `>=` `<` `<=`. The relational operators have lower precedence than the arithmetic operators. If $a=10$ and $b=20$ then:

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

Logical Operators

The Logical operators are:

&& (Logical AND)

|| (Logical OR)

! (Logical NOT)

- In Logical AND operator, if expressions on both the sides of the logical AND operator is true, then the whole expression is true.
- In Logical OR operator, if expressions on one or both the sides of the logical OR operator is true, then the whole expression is true.
- The Logical NOT operator takes a single expression and negates the value of the expression. That is, logical NOT produces a zero if the expression evaluates to a non-zero value and produces a 1 if the expression produces a zero.
- 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.

Assignment Operators

An assignment operator is used for assigning a value to a variable. The assignment operator `=` is used to assign a value to a variable. Most binary operators have a corresponding assignment operator *op=*.

`+= -= *= /= %= >>= <<= &= ^= |=`

Ex: `a+=b` is equivalent to `a=a+b`

Operator	Example	Same as
<code>=</code>	<code>a = b</code>	<code>a = b</code>
<code>+=</code>	<code>a += b</code>	<code>a = a+b</code>
<code>-=</code>	<code>a -= b</code>	<code>a = a-b</code>
<code>*=</code>	<code>a *= b</code>	<code>a = a*b</code>
<code>/=</code>	<code>a /= b</code>	<code>a = a/b</code>
<code>%=</code>	<code>a %= b</code>	<code>a = a%b</code>

Increment and Decrement Operators

C provides two operators for incrementing and decrementing variables. The increment operator `++` adds 1 to its operand, while the decrement operator `--` subtracts 1. The `++` and `--` may be used either as prefix operators (before the variable, as in `++n`), or postfix (after the variable: `n++`). In both the cases, the effect is to increment `n`. But the expression `++n` increments `n` before its value is used, while `n++` increments `n` after its value has been used.

If `n` is 5, then

`x = n++;` //sets `x` to 5, but

`x = ++n;` //sets `x` to 6.

In both the cases, n becomes 6.

Note: The increment and decrement operators can only be applied to variables. An expression like $(i+j)++$ is illegal.

Bitwise Operators

C provides six operators for bit manipulation.

& bitwise AND
| bitwise inclusive OR
^ bitwise exclusive OR
<< left shift
>> right shift
~ one's complement (unary)

The bitwise AND operator is used to mask off some set of bits.

Ex: $n = n \& 0177$; sets to zero all but low order 7 bits of n

The bitwise OR operator is used to turn bits on:

Ex: $n = n | 0011$

- The bitwise exclusive OR operator ^ sets a one in each bit position where other operands have different bits, and zero where they are the same.
- The shift operators << and >> perform left and right shifts of their left operand by the number of bit positions given by the right operand, which must be positive.
- **Ex:** $x << 2$, shifts the value of x left by two positions, filling the vacated bits with zero. The unary operator ~ yields the one's complement of an integer, that is, it converts each 1-bit into a 0-bit and vice versa.

Conditional Operators

The conditional operator uses the following form:

expr1 ? expr2 :expr3

The expression **expr1** is evaluated first. If it is non-zero (true), then the expression **expr2** is evaluated, and that is the value of the conditional expression. Otherwise **expr3** is evaluated, and that is the value. Only one of **expr2** and **expr3** is evaluated. Thus to set z to the maximum of a and b, we use a conditional operator as:

$z = (a > b) ? a : b ;$

18. Type Conversions

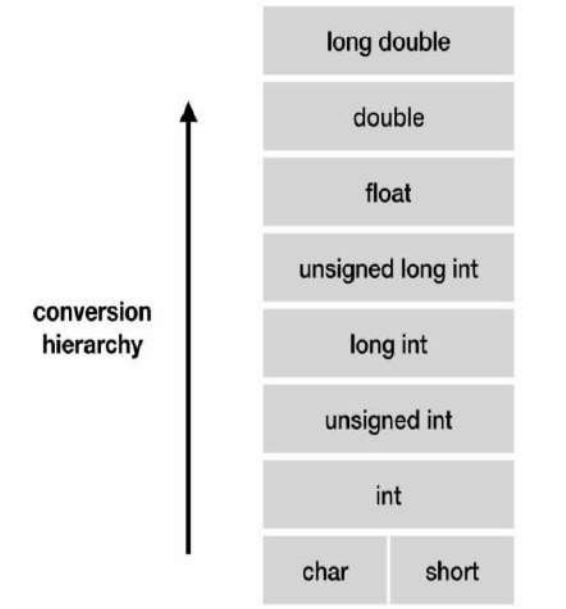
Converting one data type to another data type is called type conversion. Two types of conversions are **Implicit type conversion** and **Explicit type conversion**

Implicit type conversion

Implicitly converting a variable from one data type to another is called as **implicit type conversion**. This is automatically done by the compiler.

When an operator has operands of different types, they are converted to a common type according to a small number of rules:

- If either operand is *long double*, convert the other to *long double*
- If either operand is *double*, convert the other to *double*
- If either operand is *float*, convert the other to *float*
- Convert *char* and short to *int*
- If either operand is long, convert the other to long



Consider the following code in which an integer data type is promoted to float. This is known as promotion (when a lower level data type is promoted to a higher type).

```
float x;  
int y = 3;  
x = y;
```

Now, $x = 3.0$, as the integer value is automatically converted into its equivalent floating point representation.

Now, consider the following group of statements:

```
float f = 3.5;  
int i;  
i = f;
```

The statement $i = f$ results in f to be demoted to type `int`, i.e., the fractional part of f will be lost and i will contain 3 (not 3.5). In this case demotion takes place i.e., a higher level data type is converted into a lower type.

Explicit Type Conversion

Explicit Type Conversion can be forced (“coerced”) in any expression, with a unary operator called a cast.

In the construction, **(type-name) expression**

The expression is converted to a forcefully to the type specified in the type name.

```
int a = 500, b = 70;
```

```
float res;
```

Ex: `res = (float) a / b;`

In the above example, the value obtained by dividing a by b, will be converted to floating number and will be stored in the variable **res**.

19. Precedence and Associativity (Order of evaluation)

- **Precedence** specifies the order in which the different operators in an expression should be evaluated.
- **Associativity** specifies the order in which the operators with the same precedence should be evaluated.

The below table summarizes the rules for precedence and associativity of all types of operators. Operators on the same line have the same precedence; rows are in order of decreasing precedence.

Rules for evaluation of expression

- First, parenthesized sub expression from left to right is evaluated.
- If parentheses are nested, the evaluation begins with the innermost sub-expression.

- The precedence rule is applied in determining the order of application of operators in evaluating sub-expression.
- The associativity rule is applied when two or more operators of the same precedence level appear in a sub-expression.
- Arithmetic expressions are evaluated from left to right using the rules of precedence.
- When parentheses are used, the expressions within parentheses assume highest priority.

TABLE 2-1. PRECEDENCE AND ASSOCIATIVITY OF OPERATORS

OPERATORS	ASSOCIATIVITY
() [] -> .	left to right
! ~ ++ -- + - * & (type) sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
&	left to right
^	left to right
	left to right
&&	left to right
	left to right
? :	right to left
= += -= *= /= %= &= ^= = <<= >>=	right to left
,	left to right

Unary +, -, and * have higher precedence than the binary forms.

Web Resources

1. **Need of computer language:** <https://www.zoom.us/j/757632733>
2. **Basics of Computer:** <https://youtu.be/zkAr7P1S-c8>
3. **Learning to Program in C:**
<https://www.youtube.com/watch?v=rk2fK2IiiQ>
4. **Tokens in C:** <https://www.youtube.com/watch?v=5De6OILHH9A>
5. **Fundamental Data types:**
<https://www.youtube.com/watch?v=vNeOx1rQ25E&t=432s>

Video Resources

1. Introduction-> <https://youtu.be/tMOKg2OfntU>
2. Functional Units of a computer-> <https://youtu.be/keB0Bqtrba4>
3. Computer Memory--> https://youtu.be/_RRDWqlMMXU
4. Secondary Memory--> <https://youtu.be/cRi0ym8hQfw>
5. I/O Devices--> https://youtu.be/g_UrrqJLdpU
6. Ports and Connectors: <https://youtu.be/132Dhf6ANDQ>
7. Software Basics: <https://youtu.be/QcBpRmBcgFI>
8. Computer Networks: <https://youtu.be/KAFkt81Niu4>
9. Overview Of C: <https://youtu.be/XDdyI3W1LPE>
10. C-Tokens: <https://youtu.be/FXprL9eYrho>
11. C-Operators: <https://youtu.be/8GRmmHvkYsk>
12. Type-Conversion: <https://youtu.be/-Omv9KUEc9g>
13. Basic C-Programs: <https://youtu.be/BkZ3ppL9Up0>

Question Bank

1. What is a Computer? List and explain different types of computers.
2. Write a note on various computer generations.
3. With a neat diagram explain the basic structure of a computer.
4. What is Software? Explain different types of software.
5. Explain the steps involved in execution of C program.
6. Explain the basic structure of C program with example.
7. What are the rules to be followed to declare variables with example?
8. Define C tokens. List and explain different C tokens.
9. Define data type. Explain primitive data types supported by C language.
10. Write a note on different types of Type conversion, with example/program for each.
11. List and explain all C operators.
12. Explain ternary operator with suitable example.
13. Write a note on Operator precedence and Associativity.

University Questions

1. With a neat block diagram of computer, explain its components.
2. Describe the types of computer.
3. Classify the following into input and output devices: Monitors, visual display unit, track balls, Bar code reader, digital camera, film recorder, microfiche, OMR, electronic whiteboard, Plotters.
4. Define the terms: Network, LAN, WAN, MAN, and network topology.
5. Write the basic structure of C program. Explain each section briefly with suitable example.
6. Define a token. Explain the different tokens available in C language.
7. Define operator. Explain any 6 operators with suitable example.
8. State whether the following are valid identifiers or not: integer, float, I am, 123_AbC.

Module-1

1. What is a Computer? List and Explain different type of computers?
2. With a neat diagram explain the basic structure of a computer?
3. What is Software? Explain different types of software?
4. Explain basic structure of a C program with an example?
5. What is type conversion? List the different types of type conversion with example?
6. What is operator? Explain Bitwise, Conditional, Increment and Decrement operators with example.
7. Write a program to compute the Simple interest and compound interest.
8. Write a program to convert the temperature in degree Celsius to Fahrenheit.
9. Write a C Program to find the area and perimeter of a circle. Draw the flowchart for the same. Note: Read input from the user.
10. One problem on solving the arithmetic expressions.

Module-2

1. Explain formatted and unformatted input/output statement with syntax and example.
2. Write a C program to check if the entered year is a leap year or not.
3. Explain the various decision making statements in C with syntax and examples.
4. Write a C program that takes three coefficients (a, b, and c) of a quadratic equation: (ax^2+bx+c) as input and compute all possible roots and print them with appropriate messages.
5. Write a C Program to demonstrate the simple calculator operations like addition, subtraction, multiplication & division using 'switch' statement. Print them with appropriate messages.
6. Write a C program to find the largest of three numbers and also draw the flow chart for the same.
7. Explain the syntax of else if ladder statement. Write a 'C' Program to display the grades obtained by student based on the average marks scored using else if ladder.

Average Marks	Grade
80 to 100	Honors
60 to 79	First Division
50 to 59	Second Division
40 to 49	Third Division
0 to 39	Fail