CS106ES: ELEMENTS OF COMPUTER SCIENCE AND ENGINEERING

B.Tech. I Year I Sem.

UNIT – I

Fundamentals of Computer

Q.1 Define a computer. Explain the Characteristics of a computer.

WHAT IS COMPUTER?

• The word "computer" is comes from the word "TO COMPUTE" means to calculate.

• A computer is normally considered to be a calculation device which can perform the arithmetic operations very speedily.

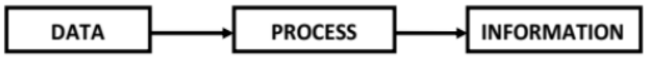
• A computer may be defined as a device which operates upon the data.

• Data can be in the form of numbers, letters, symbols, size etc. And it comes in various shapes & sizes depending upon the type of computer application.

• A computer can store, process & retrieve data as and when we desired.

• The fact that computer process data is so fundamental that many people have started calling as "Data Processor".

• A computer first it gets the Data, does Process on it and then produces Information.



Definition of Computer

o A computer is an electronic device which takes input from the user, processes it and gives the output as per user's requirement.

o So the main tasks of performed by the computer are:

▪ Input ▪ Process ▪ Output

The Characteristics of Computer

Some important characteristics of the computer are as follow:

• **Automatic**:

o Computers are automatic machines because it works by itself without human intervention.

o Once it started on a job they carry on until the job is finished.

o Computer cannot start themselves.

o They can works from the instructions which are stored inside the system in the form of programs which specify how a particular job is to be done.

• **Accuracy**:

o The accuracy of a computer is very high.

o The degree of accuracy of a particular computer depends upon its design.

o Errors can occur by the computer. But these are due to human weakness, due to incorrect data, but not due to the technological weakness.

• **Speed**:

o Computer is a very fact device. It can perform the amount of work in few seconds for which a human can take an entire year.

o While talking about computer speed we do not talk in terms of seconds and milliseconds but in microseconds.

o A powerful computer is capable of performing several billion 10^9 simple arithmetic operations per second.

• **Diligence**:

o Unlike human beings, a computer is free from monotony, tiredness & lack of concentration.

o It can continuously work for hours without creating any error & without grumbling.

o If you give ten million calculations to performed, it will perform with exactly the same accuracy & speed as the first one.

• **Versatility**:

o It is one of the most wonderful features about the computer.

o One moment it is preparing the results of a particular examination, the next moment it is busy with preparing electricity bills and in between it may be helping an office secretary to trace an important letter in seconds.

• **Power of remembering**:

o Computer can store and recall any amount of data because of its high storage capacity of its storage devices.

o Every piece of information can be retained as long as desired by the user and can be recalled as and when required.

o Even after several years, if the information recalled, it will be as accurate as on the day when it was filled to the computers.

• **No I.Q.**

o A computer is not a magical device; it processes no intelligence of its own.

o Its I.Q. is zero.

o It has to be told what to do & in what sequence.

o It cannot take its own decision.

• **No Feelings:**

o A Computer has no feelings because they are machines.

o Based on our feelings, task, knowledge and experience we often make certain judgments in our day today life.

o But Computer goes exactly the way which we have given the instructions

Q.2 Explain the generations of the computer.

Generations of Computers

In Computer language, “Generation” is a set of Technology. It provides a framework for the growth of the computer technology. There are totally Five Computer Generations till today. Discussed as following.

First Generation: • Duration: 1942-1955 • Technology: vacuum tube

o Used as a calculating device.
o Performed calculations in milliseconds.
o To bulky in size & complex design.
o Required large room to place it.
o Generates too much heat & burnt.
o Required continuously hardware maintenance.
o Generates much heat so must air-conditioner rooms are required.
o Commercial production is difficult & costly.
o Difficult to configure.
o Limited commercial use.
o ENIAC, EDVAC, EDSAC are example of 1st generation computer.

Second Generation: • Duration: 1955-1964 • Technology: transistor

o 10 times Smaller in size than 1st generation system.
o Less heat than 1st generation computers.
o Consumed less power than 1st generation system.
o Computers were done calculations in microseconds.
o Air-conditioner is also required.
o Easy to configure than 1st generation computers
o More reliable in information.
o Wider commercial use.
o Large & fast primary/secondary storage than 1st generation computers.

Third Generation: • Duration: 1965-1975 • Technology: IC chip

o Smaller in size than 1st & 2nd generation computers.
o Perform more fast calculations than 2nd generation systems.
o Large & fast primary/secondary storage than 2nd generation computers.
o Air –conditioner is required.
o Widely used for commercial applications.
o General purpose computers.
o High level languages like COBOL & FORTAN are allowed to write programs.
o Generate less heat & consumed less power than 2nd generation computer.

Fourth Generation: • Duration: 1975-1989 • Technology: Microprocessor chip

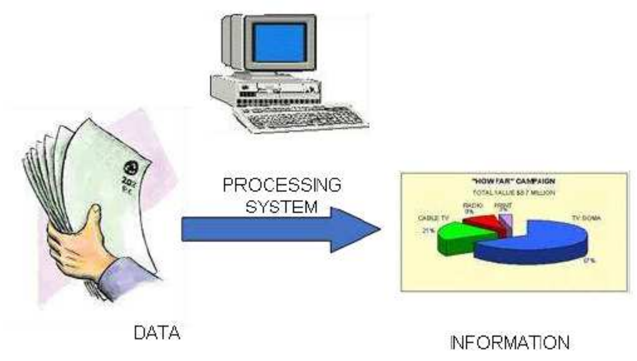
o Based on LSI & VLSI microprocessor chip.
o Smaller in size.
o Much faster than previous generations.
o Minimum hardware maintenance is required.
o Very reliable as computer to previous generation computers.
o Totally general purpose computer.

o Easy to configure.
o Possible to use network concept to connect the computer together.
o NO requirement of air-conditioners.
o Cheapest in price.
Fifth Generation: • Duration: 1989 to Present • Technology: ULSI microprocessor chip
o Much smaller & handy.
o Based on the ULSI chip which contains 100 million electronic components.
o The speed of the operations is increased.
o Consumed less power.
o Air-conditioner is not required.
o More user-friendly interface with multi-media features.
o High level languages are allowed to write programs.
o Larger & faster primary/secondary storage than previous generations.
o Notebook computers are the example of 5th generation computers.

Q.3 Explain “Computer is an Information Processing System”.

Information Processing System

- DATA is a collection of independent and unorganized facts.
- INFORMATION is the processed and organized data presented in a meaningful form.
- DATA PROCESSING is the course of doing things in a sequence of steps.
- COMPUTER is an electronic machine that follows a set of instructions in order that it may be able to accept and gather data and transform these into information.



Functions of an Information Processing System

1. It accepts and gather data. (INPUT)
2. It processes data to become information. (PROCESSING)
3. It stores data and information. (STORE)
4. It presents information. (OUTPUT)

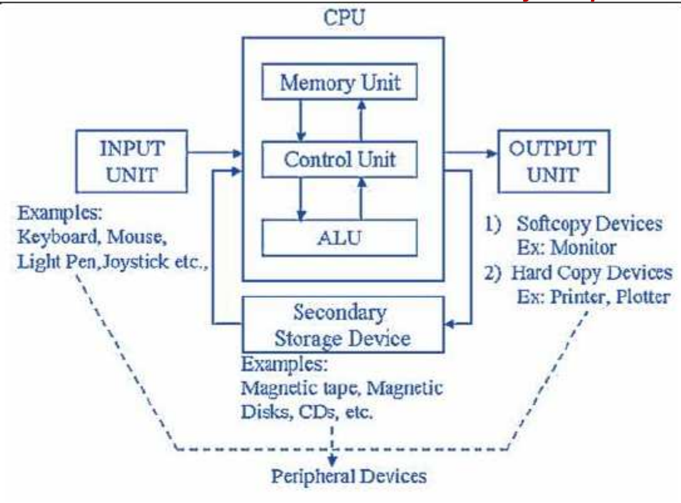
Q.4 Explain the components and functions using block diagram of a computer.

Basic hardware of a PC system

- Central Processing Unit (CPU)
- Memory Unit
- Input Devices
- Output Devices
- Secondary Storage Devices

Central Processing Unit

- Brain of the computer.
- It directs and controls the entire computer system and performs all arithmetic and logical operations.
- This unit controls the operations of all parts of computer but does not carry out any actual data processing operations.


CPU
Memory Unit
Control Unit
ALU
INPUT UNIT
Examples: Keyboard, Mouse, Light Pen,Joystick etc.,
OUTPUT UNIT
1) Softcopy Devices Ex: Monitor
2) Hard Copy Devices Ex: Printer, Plotter
Secondary Storage Device
Examples: Magnetic tape, Magnetic Disks, CDs, etc.
Peripheral Devices

Block Diagram of a Computer

Functions:

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

ALU(Arithmetic Logic Unit)

This unit consists of two subsections namely

- Arithmetic section
- Logic Section

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

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

Memory Unit

■ This unit can store instructions, data and intermediate results. This unit supplies information to the other units of the computer when needed. It is also known as internal storage unit or main memory or primary storage or Random-access memory (RAM).

■ Its size affects speed, power and capability.

■ Primary memory and secondary memory are two types of memories in the computer.

Functions:

- It stores all the data and the instructions required for processing.
- It stores intermediate results of processing.
- It stores final results of processing before these results are released to an output device.
- All inputs and outputs are transmitted through main memory.

■ Where the programs and data are stored.

• READ ONLY MEMORY (ROM) contains the pre-programmed computer instructions such as the Basic Input Output System (BIOS).

• RANDOM ACCESS MEMORY (RAM) is used to store the programs and data that you will run. Exists only when there is power.

Three Major Components of an Information Processing System

■ HARDWARE is the tangible part of a computer system.

■ SOFTWARE is the non-tangible part that tells the computer how to do its job.

- PEOPLEWARE refer to people who use and operate the computer system, write computer programs, and analyze and design the information system.

Basic Units of Measurement

- BIT is a unit of information equivalent to the result of a choice between only 2 possible alternatives in the binary number system.
- BYTE is a sequence of 8 bits (enough to represent one character of alphanumeric data) processed as a single unit for information.
- A byte can be used to represent a single character, which can be:

  A letter A number A special character or symbol, or A space

Input Devices

Allows data and programs to be sent to the CPU.

- Keyboard, Mouse, Joystick, Microphone, Webcam, Scanner, Monitor

Output Devices

Media used by the computer in displaying its responses to our requests and instructions.

- Monitor, Audio Speakers, Printer

Q.6 Briefly explain the various secondary storage units.

Secondary Storage Devices

Attached to the computer system to allow you to store programs and data permanently for the purpose of retrieving them for future use.

- Floppy disk, Hard disk, CD Rom, DVD

Optical Drives

CD-ROM	read CDs
CD-Writer	read/write CDs
DVD-Combo	read/write CDs, read DVD
DVD Writer	read/write CDs, read/write DVDs

Parts that Build Up A System Unit

- Casing or cover
- Power Supply
- Motherboard
- Microprocessor
- Memory
- Video Card
- Sound card
- Floppy disk drive
- Hard disk drive
- CD-ROM drive
- MODEM

Software

- Instructions that tell the computer how to process data into the form you want.
- Software and programs are interchangeable.
- Two major types: System and Applications

Q.7 Differentiate between hardware and software.

Computer Hardware & Software:

Computer Hardware:

Hardware refers to the physical components of a computer. Computer Hardware is any part of the computer that we can touch these parts. These are the primary electronic devices used to build up the computer. Examples of hardware in a computer are the Processor, Memory Devices, Monitor, Printer, Keyboard, Mouse, and the Central Processing Unit.

Computer Software:

Software is a collection of instructions, procedures, and documentation that performs different tasks on a computer system. we can say also Computer Software is a programming code executed on a computer processor. The code can be machine-level code or the code written for an operating system. Examples of software are Ms Word, Excel, PowerPoint, Google Chrome, Photoshop, MySQL, etc.

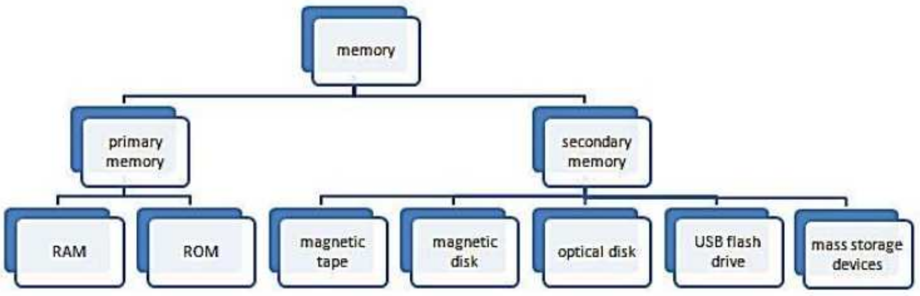
Difference Between Hardware and Software:

S. No.	Parameters	Hardware	Software
1	**Basic Definition**	Hardware is a physical part of the computer that causes the processing of data.	Software is a set of instructions that tells a computer exactly what to do.
2	**Development**	It is manufactured.	It is developed and engineered.
3	**Dependency**	Hardware cannot perform any task without software.	The software can not be executed without hardware.
4	**Process of creating**	Electronic and other materials are used to create hardware.	Created by utilizing a computer language to write instructions.
5	**Tangible**	Hardware is tangible as hardware is a physical electronic device, that can be touched.	Software is intangible as we can see and also use the software but can't touch them.
6	**Durability**	Hardware typically wears out over time.	The software does not wear out with time. However, it may contain flaws and glitches.
7	**Types**	It has **four** main categories: input devices, output devices, storage, and internal components.	It is mainly divided into System software and Application software.
8	**Virus effect**	Hardware is not affected by computer viruses.	Software is affected by computer viruses.
9	**Transfer**	It cannot be transferred from one place to another electrically through the network.	It can be transferred via a network means.
10	**Machine-Level language**	Only machine-level language is known to be understood by hardware.	The program accepts human-readable input, interprets it in machine-level language, and sends it to hardware for additional processing.
11	**Replacement**	If hardware is damaged, it is replaced with a new one.	If the software is damaged, its backup copy can be reinstalled.
12	**Failures**	Dust, overheating, humidity and other factors are commonly responsible for hardware failures.	Overloading, systematic error, major-minor version error, and other factors are commonly responsible for software failures.
13	**Examples**	Ex: Keyboard, Mouse, Monitor, Printer, CPU, Hard disk, RAM, ROM, etc.	Ex: MS Word, Excel, PowerPoint, Photoshop, MySQL, etc.

Q.8 Categorize the computer memories and explain in detail.

Computer Memory

A computer is a device that is electronic and that accepts data, processes that data, and gives the desired output. It performs programmed computation with great accuracy & higher speed. Or in other words, the computer takes data as input and stores the data/instructions in the memory (use them when required). It then processes the data and converts it into useful information. Finally, it gives the output. Here, *input* refers to the raw data that we want the machine to process and return to us as a result, *output* refers to the response that the machine provides in response to the raw data entered and the processing of data may involve analyzing, searching, distributing, storing data, etc. Thus, we can also call a computer data processing system.



What is Memory?

Computer memory is just like the human brain. It is used to store data/information and instructions. It is a data storage unit or a data storage device where data is to be processed and instructions required for processing are stored. It can store both the input and output can be stored here.

Characteristics of Main Memory:

- It is faster computer memory as compared to secondary memory.
- It is semiconductor memories.
- It is usually a volatile memory.
- It is the main memory of the computer.
- A computer system cannot run without primary memory.

In general, memory is of three types:

- Primary memory
- Secondary memory
- Cache memory

1. Primary Memory: It is also known as the main memory of the computer system. It is used to store data and programs or instructions during computer operations. It uses semiconductor technology and hence is commonly called semiconductor memory. Primary memory is of two types:

(i) RAM (Random Access Memory): It is a volatile memory. Volatile memory stores information based on the power supply. If the power supply fails/ interrupted/stopped, all the data & information on this memory will be lost. RAM is used for booting up or start the computer. It temporarily stores programs/ data which has to be executed by the processor. RAM is of two types:

- **SRAM (Static RAM):** It uses transistors, and the circuits of this memory are capable of retaining their state as long as the power is applied. This memory consists of the number of flip flops with each flip flop storing 1 bit. It has less access time and hence, it is faster.
- **DRAM (Dynamic RAM):** It uses capacitors and transistors and stores the data as a charge on the capacitors. They contain thousands of memory cells. It needs refreshing of charge on capacitor after a few milliseconds. This memory is slower than SRAM.

(ii) ROM (Read Only Memory): It is a non-volatile memory. Non-volatile memory stores information even when there is a power supply failed/ interrupted/stopped. ROM is used to store information that is used to operate the system. As its name refers to read-only memory, we can only read the programs and data that is stored on it. It contains some electronic fuses that can be programmed for a piece of specific information. The information stored in the ROM in binary format. It is also known as permanent memory. ROM is of four types:

- **MROM (Masked ROM):** Hard-wired devices with a pre-programmed collection of data or instructions were the first ROMs. Masked ROMs are a type of low-cost ROM that works in this way.
- **PROM (Programmable Read Only Memory):** This read-only memory is modifiable once by the user. The user purchases a blank PROM and uses a PROM program to put the required contents into the PROM. Its content can’t be erased once written.
- **EPROM (Erasable Programmable Read Only Memory):** It is an extension to PROM where you can erase the content of ROM by exposing it to Ultraviolet rays for nearly 40 minutes.

- **EEPROM (Electrically Erasable Programmable Read Only Memory):** Here the written contents can be erased electrically. You can delete and reprogramme EEPROM up to 10,000 times. Erasing and programming take very little time, i.e., nearly 4-10 ms(milliseconds). Any area in an EEPROM can be wiped and programmed selectively.

2. Secondary Memory: It is also known as auxiliary memory and backup memory. It is a non-volatile memory and used to store a large amount of data or information. The data or information stored in secondary memory is permanent, and it is slower than primary memory. A CPU cannot access secondary memory directly. The data/information from the auxiliary memory is first transferred to the main memory, and then the CPU can access it.

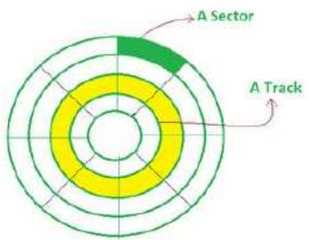
Characteristics of Secondary Memory:

- It is a slow memory but reusable.
- It is a reliable and non-volatile memory.
- It is cheaper than primary memory.
- The storage capacity of secondary memory is large.
- A computer system can run without secondary memory.
- In secondary memory, data is stored permanently even when the power is off.

Types of secondary memory:

(i) Magnetic Tapes: Magnetic tape is a long, narrow strip of plastic film with a thin, magnetic coating on it that is used for magnetic recording. Bits are recorded on tape as magnetic patches called **RECORDS** that run along many tracks. Typically, 7 or 9 bits are recorded concurrently. Each track has one read/write head, which allows data to be recorded and read as a sequence of characters. It can be stopped, started moving forward or backward, or rewound.

(ii) Magnetic Disks: A magnetic disc is a circular metal, or a plastic plate and these plates are coated with magnetic material. The disc is used on both sides. Bits are stored in magnetized surfaces in locations called tracks that run in concentric rings. Sectors are typically used to break tracks into pieces.



Hard discs are discs that are permanently attached and cannot be removed by a single user.

(iii) Optical Disks: It's a laser-based storage medium that can be written to and read. It is reasonably priced and has a long lifespan. The optical disc can be taken out of the computer by occasional users. Types of Optical Disks:

(a) CD – ROM:

- It's called Compact Disk. Only read from memory.
- Information is written to the disc by using a controlled laser beam to burn pits on the disc surface.
- It has a highly reflecting surface, which is usually aluminum.
- The diameter of the disc is 5.25 inches.
- 16000 tracks per inch is the track density.
- The capacity of a CD-ROM is 600 MB, with each sector storing 2048 bytes of data.
- The data transfer rate is about 4800KB/sec. & the new access time is around 80 milliseconds.

(b) WORM (WRITE ONCE READ MANY):

- A user can only write data once.
- The information is written on the disc using a laser beam.
- It is possible to read the written data as many times as desired.
- They keep lasting records of information, but access time is high.
- It is possible to rewrite updated or new data to another part of the disc.
- Data that has already been written cannot be changed.

- Usual size – 5.25 inch or 3.5 inch diameter.
- The usual capacity of 5.25 inch disk is 650 MB,5.2GB etc.

(c) DVDs:

- The term "DVD" stands for "Digital Versatile/Video Disc," and there are two sorts of DVDs: (i)DVDR (writable) and (ii) DVDRW (Re-Writable)
- *DVD-ROMS (Digital Versatile Discs):* These are read-only memory (ROM) discs that can be used in a variety of ways. When compared to CD-ROMs, they can store a lot more data. It has a thick polycarbonate plastic layer that serves as a foundation for the other layers. It's an optical memory that can read and write data.
- *DVD-R:* It is a writable optical disc that can be used just once. It's a DVD that can be recorded. It's a lot like WORM. DVD-ROMs have capacities ranging from 4.7 to 17 GB. The capacity of 3.5 inch disk is 1.3 GB.

3. Cache Memory: It is a type of high-speed semiconductor memory that can help the CPU run faster. Between the CPU and the main memory, it serves as a buffer. It is used to store the data and programs that the CPU uses the most frequently.

Advantages of cache memory:

- It is faster than the main memory.
- When compared to the main memory, it takes less time to access it.
- It keeps the programs that can be run in a short amount of time.
- It stores data in temporary use.

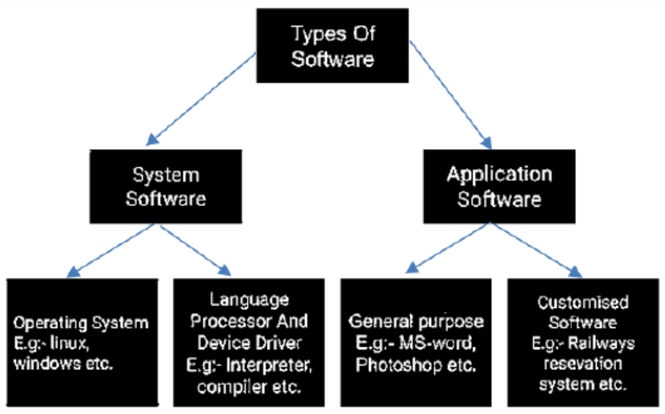
Disadvantages of cache memory:

- Because of the semiconductors used, it is very expensive.
- The size of the cache (amount of data it can store) is usually small.

Software and its Types

Q.9 Explain the different types of software.

In a computer system, the software is basically a set of instructions or commands that tells a computer what to do. Or in other words, the software is a computer program that provides a set of instructions to execute a user's commands and tell the computer what to do. For example, like MS-Word, MS-Excel, PowerPoint, etc. The chart below describes the types of software:


Types Of Software
System Software
Application Software
Operating System E.g:- linux, windows etc.
Language Processor And Device Driver E.g:- Interpreter, compiler etc.
General purpose E.g:- MS-word, Photoshop etc.
Customised Software E.g:- Railways resevation system etc.

Above is the diagram of types of software.

System Software

System software is software that directly operates the computer hardware and provides the basic functionality to the users as well as to the other software to operate smoothly. Or in other words, system software basically controls a computer's internal functioning and also controls hardware devices such as monitors, printers, and storage devices, etc. It is like an interface between hardware and user applications, it helps them to communicate with each other because hardware understands machine language (i.e. 1 or 0) whereas user applications are work in human-readable languages like English, Hindi, German, etc. so system software converts the human-readable language into machine language and vice versa.

Features Of System Software:

Let us discuss some of the features of System Software:

1. System Software is closer to the computer system.
2. System Software is written in a low-level language in general.
3. System software is difficult to design and understand.
4. System software is fast in speed (working speed).
5. System software is less interactive for the users in comparison to application software.

Types Of System Software:

It has two subtypes which are:

1. **Operating System**: It is the main program of a computer system. When the computer system ON it is the first software that loads into the computer's memory. Basically, it manages all the resources such as memory, CPU, printer, hard disk, etc., and provides an interface to the user, which helps the user to interact with the computer system. It also provides various services to other computer software. Examples of operating systems are Linux, Apple macOS, Microsoft Windows, etc

2. **Language Processor**: As we know that system software converts the human-readable language into a machine language and vice versa. So, the conversion is done by the language processor. It converts programs written in high-level programming languages like Java, C, C++, Python, etc(known as source code), into sets of instructions that are easily readable by machines(known as object code or machine code).

3. **Device Driver**: A device driver is a program or software that controls a device and helps that device to perform its functions. Every device like a printer, mouse, modem, etc. needs a driver to connect with the computer system eternally. So, when you connect a new device with your computer system, first you need to install the driver of that device so that your operating system knows how to control or manage that device.

Q.10

Application Software

Software that performs special functions or provides functions that are much more than the basic operation of the computer is known as application software. Or in other words, application software is designed to perform a specific task for end-users. It is a product or a program that is designed only to fulfill end-users' requirements. It includes word processors, spreadsheets, database management, inventory, payroll programs, etc.

Features Of Application Software:

Let us discuss some of the features of Application Software:

1. An important feature of application software is it performs more specialized tasks like word processing, spreadsheets, email, etc.
2. Mostly, the size of the software is big, so it requires more storage space.
3. Application software is more interactive for the users, so it is easy to use and design.
4. The application software is easy to design and understand.
5. Application software is written in a high-level language in general.

Types Of Application Software:

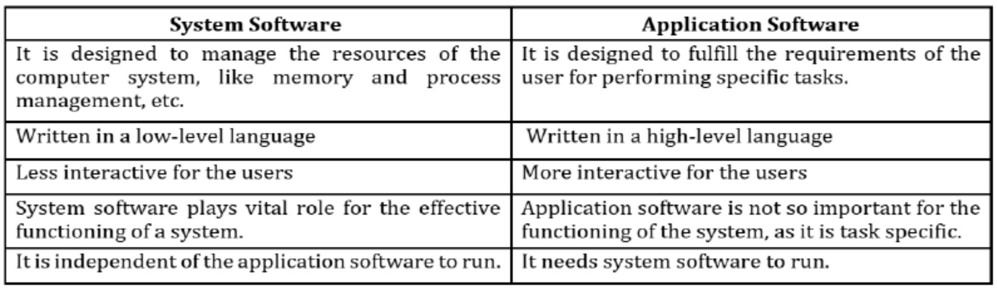
There are different types of application software and those are:

1. **General Purpose Software**: This type of application software is used for a variety of tasks and it is not limited to performing a specific task only. For example, MS-Word, MS-Excel, PowerPoint, etc.

2. **Customized Software**: This type of application software is used or designed to perform specific tasks or functions or designed for specific organizations. For example, railway reservation system, airline reservation system, invoice management system, etc.

3. **Utility Software**: This type of application software is used to support the computer infrastructure. It is designed to analyze, configure, optimize, and maintains the system, and take care of its requirements as well. For example, antivirus, disk fragmenter, memory tester, disk repair, disk cleaners, registry cleaners, disk space analyzer, etc.

Difference between system software and application software



System Software	Application Software
It is designed to manage the resources of the computer system, like memory and process management, etc.	It is designed to fulfill the requirements of the user for performing specific tasks.
Written in a low-level language	Written in a high-level language
Less interactive for the users	More interactive for the users
System software plays vital role for the effective functioning of a system.	Application software is not so important for the functioning of the system, as it is task specific.
It is independent of the application software to run.	It needs system software to run.

4. **Packages**: A package may refer to any of the following:
    - With software, a package is a module that can be added to any program to add additional options, features, or functionality. A package can often be added to program using an "include" or "import" type of statement, as in the below Java code.
    - With hardware, a package is a single container containing one or more items that help make up the overall package. For example, a processor package is the overall computer processor, which contains all the necessary parts that allow it to work in your computer.

    Example: The **FileReader** package is used for reading character streams, useful for obtaining user input from a console.
5. **Frameworks**: A programming framework is a prepackaged set of solutions that solves common development problems. A programming framework as a tool or a set of tools, that's use to make life easier when performing common programming tasks. A software framework is built on top of a programming language. Example: Rails, also known as Ruby on Rails, is a web framework built on top of the Ruby programming language. Django and Flask are two different web frameworks built on top of the Python programming language. Hence, they are also known as Python frameworks.
6. **IDEs**: IDE is the **Integrated Development Environment** that provides the user interface for code development, testing and debugging features. It helps to organize the project artifacts that are relevant to the source code of the software application. It provides several tools and features to make the development easy and standardize based upon the programming language the developer writes the code. The IDEs also have the functionalities to compile and interpret the program. It is a software application that defines the visual representation of the location of the files easily and makes it more understandable for the user. It contains development tools such as text editors, code libraries, compilers, and test platforms and consists of at least build automation tools and a debugger.

    Example: Some of the widely used IDs are Eclipse for Java programming development, Microsoft Visual Studio, Android Studio for Mobile Apps development, RStudio for R programs and PyCharm for Python programming.

UNIT – II

Software Development

Software Development is the development of software for distinct purposes. For software development, there is a specific programming language like Java, Python, C/C++, etc. The entire process of software development isn't as simple as its definition, it's a complicated process. Accordingly, it requires an efficient approach from the developer in the form of the Software Development Life Cycle (SDLC).

Software Development Life Cycle (SDLC)

The software development life cycle (SDLC) explains the different stages of software development. This framework is important because it covers the planning, building, deployment, and maintenance of the software. The SDLC delivers high-quality software by creating it in a systematic manner.

Proper planning is an essential aspect of the software development life cycle. From there, team members develop and execute plans into the software.

In this guide, you'll learn the basic components of the SDLC. We'll show you how these phases can be implemented to manage any development effort successfully. Lastly, we'll review some of the most popular software development life cycle methodologies.

Stages of the software development life cycle

The need for software development methodologies dates back to the 1950s. At that time, words like "framework" and "approach" really didn't exist in the context of software development.

Since then, software engineers have sought to create and implement development methods to accelerate software development. Now, the SDLC is used to reduce time-to-market while building an intuitive software for clients.

All software development life cycle models involve various stages. Although these strategies can vary from model to model, we'll look at the following SDLC sequence:

Stage 1: Requirement gathering and analysis
Stage 2: Design
Stage 3: Implementation or coding
Stage 4: Testing
Stage 5: Deployment
Stage 6: Maintenance

Stage 1: Requirement gathering and analysis

In this stage, teams should collect all relevant information from the client. They use this information to develop the product, ensuring that they meet client expectations. Typically, the business analysts and project managers meet with the client to gather information.

This information includes:

A description of what they want the software to be

- The end-user
- The purpose

Once they've gathered and understood the information, they should produce the software requirement specification (SDS) document.

From there, the software development team should receive this document and ask any questions. They will then pass on the document to the client. This way, the client can verify that the project is well-understood by the team and can hold on to the document for future reference.

Stage 2: Design

At this point, the requirements from the SRS document are referenced to create the software architecture. The project manager will decide on the approach that the team will take and outline a pricing model.

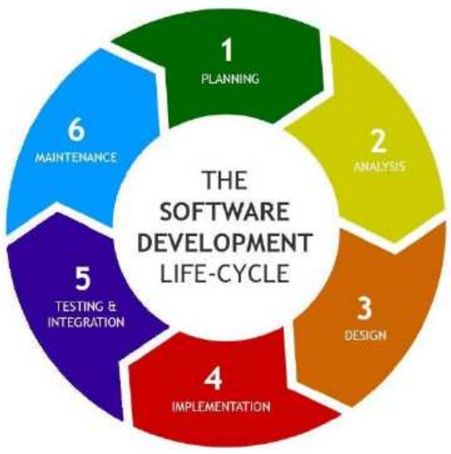
Stage 3: Implementation or coding

This phase begins after the developer receives the design document. At this point, the design is translated into source code. This is when software developers go in and implement the code.

Stage 4: Testing

Once the development team has started coding, they release modules. These modules are then rigorously tested. Issues and bugs are detected, and software developers are assigned areas to test. Testers reference the SRS document to confirm that the software matches client expectations.

This process continues until the software is perfected.



Stage 5: Deployment

At this point, the software is deployed into production. In some cases, the client may request that the software goes through user acceptance testing (UAT). Regardless of whether the client elects UAT, they will decide whether the software meets their expectations in this step.

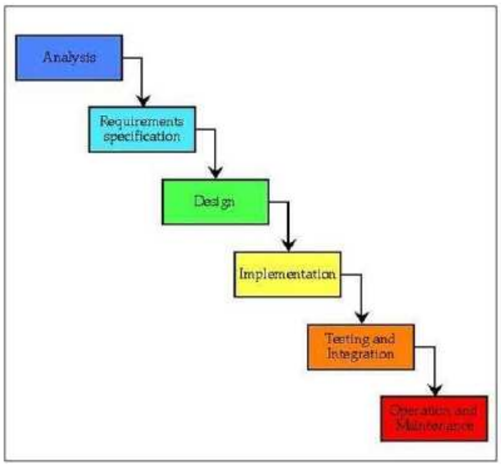
Stage 6: Maintenance

Following production, the development team will maintain the product. Sometimes, issues might arise during testing. At this point, the software developers can fix these issues. In some cases, a client might request additional features, which can be added as enhancements during this phase.

Waterfall Model

In the 1970s, the Waterfall model, also known as the linear sequential model, was created. This model focuses on an organized approach to project management.

This methodology focuses on receiving clear requirements from clients and stakeholders so that the development team can best accommodate their requests. It requires clear milestones and explanations for team members from clients. Before the development team can move on to another phase, they must complete the phase ahead of it.



Phases of the Waterfall method are as follows:

- ➢ Requirement analysis
- ➢ System design
- ➢ Implementation
- ➢ Testing
- ➢ Deployment
- ➢ Maintenance

Pros

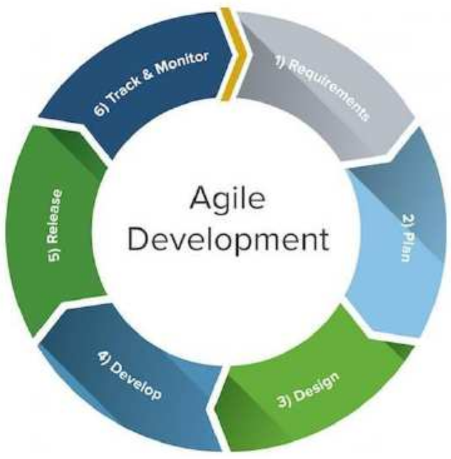
- ➢ Simple: This model is easily understood.
- ➢ Phases: The Waterfall method has clear phases for development teams to follow.
- ➢ Manageable: Because each phase is so clearly defined, the project is easily manageable.

Cons

- ➢ Time-consuming: Teams cannot move to another step until the previous phase has been completed.
- ➢ Not adaptable: This project cannot be used for projects with nonspecific requirements. Clients must be incredibly clear with their requirements for this model to work.
- ➢ Not for short-term projects: Because of the nature of the phases, this model may not work well for projects that are short in duration.

Agile

The Agile model, which formally began in the 1990s, focuses on adaptability instead of strict requirements. An Agile approach to software development empowers teams to meet requirements in a flexible manner.
Agile is particularly helpful for remote teams, as it can be used to combat issues with time zones, communication problems, availability, and more.
Projects following the Agile method are broken into smaller incremental builds. These periods are referred to as sprints. Each sprint usually lasts somewhere between two to four weeks. At the beginning of each sprint, the development team meets with the client to outline the goals of that sprint, and then they develop and test code. Finally, they review the features with the client.



This method focuses on:

- Incremental improvements
- Customer feedback
- Sprints between two to four weeks
- Constant testing

Pros

- Adaptability: Agile is incredibly flexible, allowing teams to adjust to changes in requirements.
- Improved client satisfaction: Because there is near-constant communication, and clients are given a chance to provide feedback every step of the way, this method may produce higher degrees of client satisfaction.
- Quickly add features: Since sprints generally last two to four weeks, features can be quickly added.

Cons

- Requires experience: The Agile method requires very experienced team members.
- Need for clarity: Clients need to be extremely clear with their expectations because there is no SDS documentation.
- No documentation: The Agile method focuses on software quality over documentation.

Types of Computer Languages

Computer Languages

A programming language is an artificial language designed to express computations that can be performed by a machine, particularly a computer. Programming languages can be used to create programs that control the behavior of a machine, to express algorithms precisely, or as a mode of human communication.
Many programming languages have some form of written specification of their syntax (form) and semantics (meaning). Some languages are defined by a specification document. For example, the C programming language is specified by an ISO Standard. Other languages, such as Perl, have a dominant implementation that is used as a reference.
The earliest programming languages predate the invention of the computer and were used to direct the behavior of machines. Thousands of different programming languages have been created, mainly in the computer field, with many more being created every year. Most programming languages describe computation in an imperative style, i.e., as a sequence of commands, although some languages, such as those that support functional programming or logic programming, use alternative forms of description.

Machine Language:

Machine code or machine language is a system of instructions and data executed directly by a computer's central processing unit (CPU). Machine code may be regarded as a primitive programming language or as the lowest-level representation of a compiled and/or assembled computer program. Programs in interpreted languages are not represented by machine code however, although their interpreter often is. Machine code is sometimes called native code when referring to platform-dependent parts of language features or libraries. Machine code should not be confused with so called **byte code**, which is executed by an interpreter.

Symbolic Language:

Most organisms communicate, but humans are unique in communicating via symbolic language. Humans are thus a symbolic species: symbols have literally changed the kind of biological organism we are. We think and behave in ways that are quite odd compared to other species because of the way that language has defined us. Symbolic language has become the dominant feature of the cultural environment to which we must adapt in order to grow. The programming language that uses symbols for expressing operations and operands is called as Symbolic Language. All modern programming languages are symbolic languages. The symbolic language manipulates symbols rather than numbers.

High-Level Language:

A high-level programming language is a programming language with strong abstraction from the details of the computer. In comparison to low-level programming languages, it may use natural language elements, be easier to use, or be more portable across platforms. Such languages hide the details of CPU operations such as memory access models and management of scope.

This greater abstraction and hiding of details is generally intended to make the language user-friendly, as it includes concepts from the problem domain instead of those of the machine used. A high-level language isolates the execution semantics of computer architecture from the specification of the program, making the process of developing a program simpler and more understandable with respect to a low-level language. The amount of abstraction provided defines how **high-level** a programming language is. The examples for the high level languages are C, C++, Java, FORTRAN, COBOL etc.

There are 6 types of High-Level Programming Languages:

1. Procedural-Oriented Programming Language

Procedural-Oriented Programming is also known as Third Generation Programming Language (**3GL**). It is one of the primitive and important paradigms in the Programming Industry. In Procedural-Oriented Programming, instead of focusing on data, we majorly focus on the procedure of the program. The main goal of Procedural-Oriented Programming is to solve a problem. So, data is the second priority in Procedural-Oriented Programming, as a result, this Programming Paradigm is comparatively less secure. In Procedural Oriented Programming, we create a number of statements in order to solve any problem. It uses a Top-Down approach to solve any problem. However, it is very important to maintain the order of every step or statement. Therefore, we make use of functions in Procedural-Oriented Programming. Procedural-Oriented Programming is a much unrealistic approach to solve a problem as compared to other Programming Paradigm.

Examples: Basic, Fortran, C, Pascal, and COBOL.

2. Object-Oriented Programming Language (OOP)

It is one of the most important Programming Paradigm in which we make use of Classes and Objects for creating a program. Object-Oriented Programming is the most realistic programming approach for solving Real-World problems. Here, every problem is viewed as an entity or object, by which the designing of the program becomes simpler and easier. Some of the basic concepts of Object-Oriented Programming are Object, Class, Inheritance, Abstraction, Polymorphism, and Encapsulation.

Examples: Java, Python, C++, C#, JavaScript, and PHP.

3. Functional Programming Language

Functional Programming is a style of programming, where functions are treated and used just like variables. Therefore, Functional Programming Language is also known as the First-Class Function. In Functional Programming, the data is immutable, which means that once the data is created, it cannot be changed, and we have to create a separate variable instead of changing the old one. Another goal of Functional Programming is to keep the Data separate from the Function.

Examples: Haskell, Scala, Python, Clojure, and Swift.

4. Problem-Oriented Programming Language

Problem-Oriented Programming Language is also known as Fourth Generation Programming Language (**4GL**) or Result Oriented Programming Language. Here, you can directly insert or retrieve the result or data without caring about the procedural or getting into the actual complexity of the program. It is usually used for managing the Databases. Here, the priority is given to the data only.

Examples: Fortran, COBOL, Pascal, and GPSS.

5. Scripting Programming Language

Scripting Language is the Programming Language which is used for performing automation or repetitive task with the help of scripts. Unlike, other Programming Languages, Scripting Languages are Run-Time Programming Languages. Web Automation or Web Scripting is one of the applications of Scripting Language. We can also automate our daily task on a computer, with the help of Shell Script or Bash Script, which is another most popular example of Scripting Language.

Examples: Bash, JavaScript, Shell, Python, and Perl.

6. Artificial Intelligence Programming Language

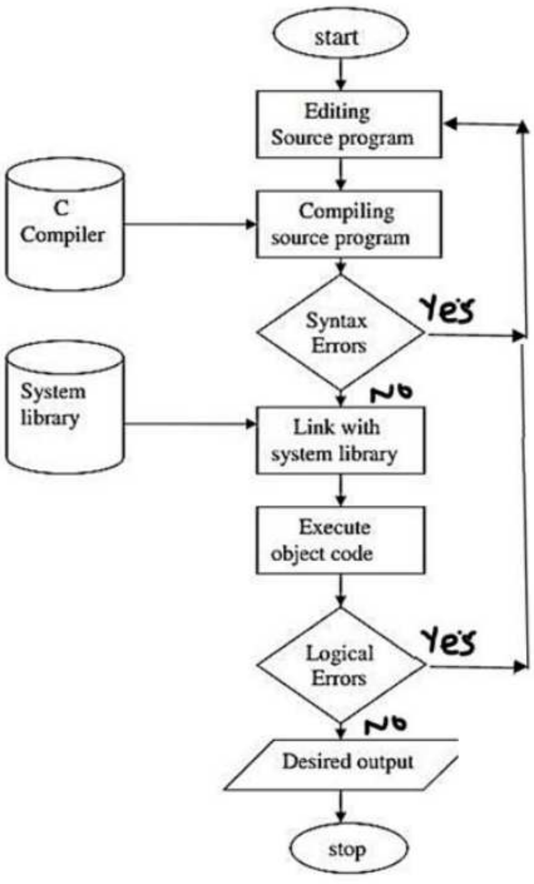
Artificial Intelligence Programming Language is also known as Fifth Generation Programming Language (**5GL**) or Natural Language. In Fifth Generation Programming Languages, the code will be written in the form of normal sentences, as we use in normal communication with others. So, unlike other Programming Languages, here we don’t need any logic or algorithm to create a program. Even a non-programmer will also be able to tell a computer what to do and the computer will perform all the tasks on its own. However, Artificial Intelligence Language is still in its development phase and a lot of research is still going on in its development.

Steps in Program Development

A program has to pass through many phases for its successful execution and to achieve the desired output. The various steps involved in the program development are as follows:

Editing Phase

Consider, in the editing phase, the C program is entered into a file through a ***text editor***. UNIX/LINUX operating system provides a text editor, which can be accessed through the command '**Vi'**. Modern compiler vendors provide Integrated Development Environment (**IDE**). The file is saved on the disk with an extension of 'C'. Corrections in the program at later stages are done through these editors. Once the program has been written, it requires to be translated into machine language.



Compilation Phase

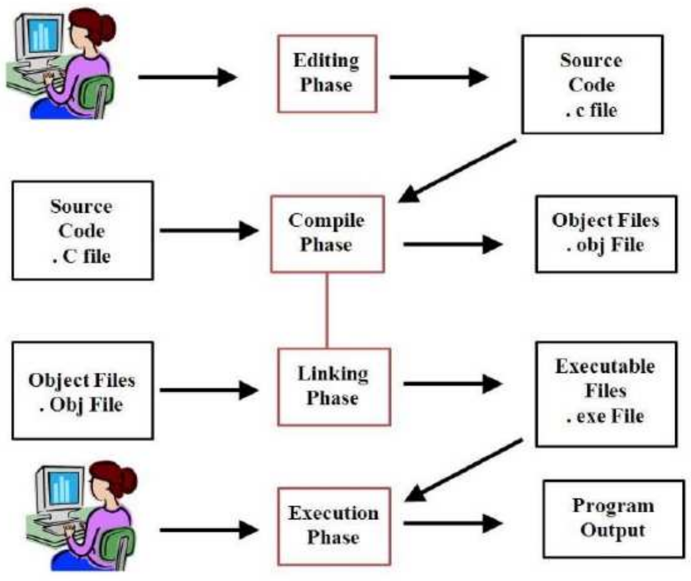
This phase is carried out by a program called as ***compiler***. Compiler translates the source code into the object code. The compilation phase cannot proceed successfully until and unless the source code is error-free. The compiler generates messages if it encounters syntactic errors in the source code. The error-free source code is translated into object code and stored in a separate file with an extension '*.obj'*.

Linking Phase

In this phase, the ***linker*** links all the files and functions with the object code under execution. For example, if a user uses a 'printf' function in his/her program, the linker links the user programs, object code with the object code of the printf function.

Execution Phase

In this phase, the executable object code is ***loaded*** into the memory and the program execution begins. We may encounter errors during the execution phase even though compilation phase is successful. The errors may be run-time errors and logical errors.



Algorithms

What is an Algorithm?

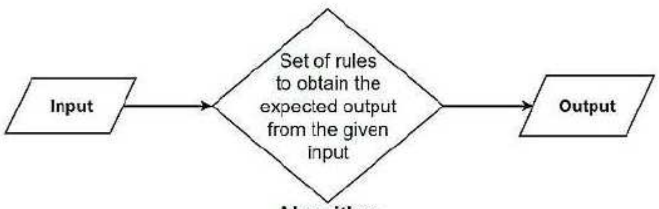
The word Algorithm means

"A set of rules to be followed in calculations or other problem-solving operations"

Or *"A procedure for solving a mathematical problem in a finite number of steps that frequently by recursive operations"*.

Therefore, Algorithm refers to a sequence of finite steps to solve a particular problem.

Algorithms can be simple and complex depending on what you want to achieve.


Input
Set of rules to obtain the expected output from the given input
Output

Algorithm

It can be understood by taking the example of cooking a new recipe. To cook a new recipe, one reads the instructions and steps and executes them one by one, in the given sequence. The result thus obtained is the new dish cooked perfectly. Every time you use your phone, computer, laptop, or calculator you are using Algorithms. Similarly, algorithms help to do a task in programming to get the expected output.

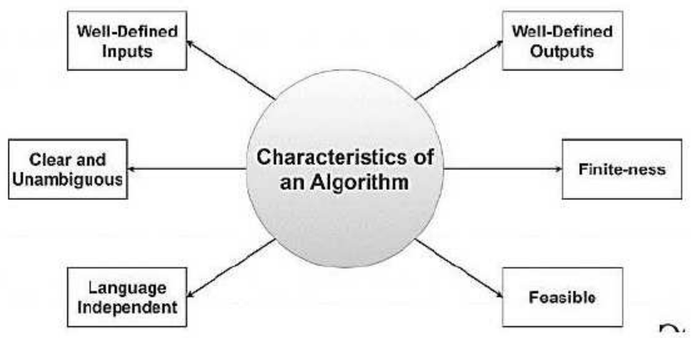
The Algorithm designed are language-independent, i.e. they are just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.

What are the 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 are an **algorithm**. For some instructions to be an algorithm, it must have the following characteristics:

- **Clear and Unambiguous**: The 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.
- **Finiteness:** The algorithm must be finite, i.e. it should terminate after a finite time.
- **Feasible:** The algorithm must be simple, generic, and practical, such that it can be executed with 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 the same, as expected.

Characteristics of an Algorithm



Properties of Algorithm:

- It should terminate after a finite time.
- It should produce at least one output.

- It should take zero or more input.
- It should be deterministic means giving the same output for the same input case.
- Every step in the algorithm must be effective i.e. every step should do some work.

Advantages of Algorithms:

- It is easy to understand.
- An algorithm is a step-wise representation of a solution to a given problem.
- In Algorithm the problem is broken down into smaller pieces or steps hence, it is easier for the programmer to convert it into an actual program.

Disadvantages of Algorithms:

- Writing an algorithm takes a long time so it is time-consuming.
- Understanding complex logic through algorithms can be very difficult.
- Branching and Looping statements are difficult to show in Algorithms**(imp)**.

How to Design an Algorithm?

To write an algorithm, the following things are needed as a pre-requisite:

1. The **problem** that is to be solved by this algorithm i.e. clear problem definition.
2. The **constraints** of the problem must be considered while solving the problem.
3. The **input** to be taken to solve the problem.
4. The **output** to be expected when the problem is solved.
5. The **solution** to this problem, is within the given constraints.

Then the algorithm is written with the help of the above parameters such that it solves the problem.

Example: Consider the example to add three numbers and print the sum.

Step 1: Fulfilling the pre-requisites

As discussed above, in order to write an algorithm, its pre-requisites must be fulfilled.

1. **The problem that is to be solved by this algorithm**: Add 3 numbers and print their sum.
2. **The constraints of the problem that must be considered while solving the problem**: The numbers must contain only digits and no other characters.
3. **The input to be taken to solve the problem:** The three numbers to be added.
4. **The output to be expected when the problem is solved:** The sum of the three numbers taken as the input i.e., a single integer value.
5. **The solution to this problem, in the given constraints:** The solution consists of adding the 3 numbers. It can be done with the help of '+' operator, or bitwise, or any other method.

Step 2: Designing the algorithm

Now let's design the algorithm with the help of the above pre-requisites:

Algorithm to add 3 numbers and print their sum:

```
1. START
2. Declare 3 integer variables num1, num2 and num3.
3. Take the three numbers, to be added, as inputs in variables num1, num2, and num3 respectively.
4. Declare an integer variable sum to store the resultant sum of the 3 numbers.
5. Add the 3 numbers and store the result in the variable sum.
6. Print the value of the variable sum
7. END
```

Step 3: Testing the algorithm by implementing it. In order to test the algorithm, let's implement it in C language.

FLOWCHARTS

Flowcharts are nothing but the graphical representation of the data or the algorithm for a better understanding of the code visually. It displays step-by-step solutions to a problem, algorithm, or process.
It is a pictorial way of representing steps that are preferred by most beginner-level programmers to understand algorithms of computer science, thus it contributes to troubleshooting the issues in the algorithm. A flowchart is a picture of boxes that indicates the process flow in a sequential manner. It is easy to interpret and understand the process.

Following are the uses of a flowchart:

- It is a pictorial representation of an algorithm that increases the readability of the program.
- Complex programs can be drawn in a simple way using a flowchart.
- It helps team members get an insight into the process and use this knowledge to collect data, detect problems, develop software, etc.
- A flowchart is a basic step for designing a new process or add extra features.
- Communication with other people becomes easy by drawing flowcharts and sharing them.

When to use flowchart

Flowcharts are mainly used in the below scenarios:

- It is most importantly used when the programmers make projects. As a flowchart is a basic step to make the design of projects pictorially, it is preferred by many.
- When the flowcharts of a process are drawn, the programmer understands the non-useful parts of the process. So flowcharts are used to separate useful logic from the unwanted parts.
- Since the rules and procedure of drawing a flowchart are universal, flowchart serves as a communication channel to the people who are working on the same project for better understanding.
- Optimizing a process becomes easier with flowcharts. The efficiency of code is improved with the flowchart drawing.

Advantages of Flowchart

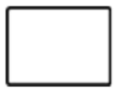
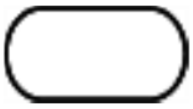
- It is the most efficient way of communicating the logic of system.
- It act like a guide for blueprint during program designed.
- It also helps in debugging process.
- Using flowchart we can easily analyze the programs.
- flowcharts are good for documentation.

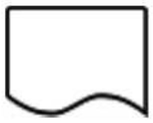
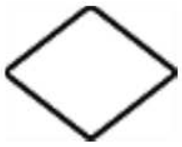
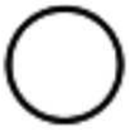
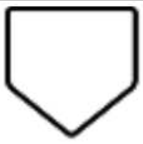
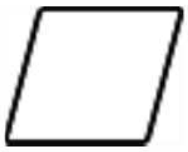
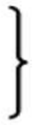
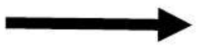
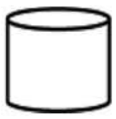
Disadvantages of Flowchart

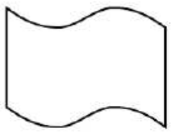
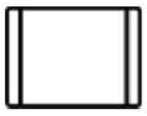
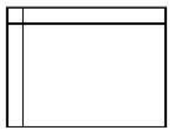
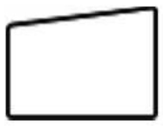
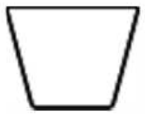
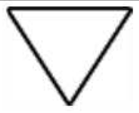
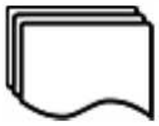
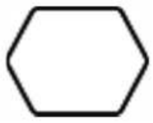
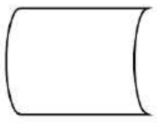
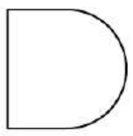
- Flowcharts are difficult to draw for large and complex programs.
- It does not contain the proper amount of details.
- Flowcharts are very difficult to reproduce.
- Flowcharts are very difficult to modify.

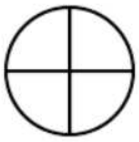
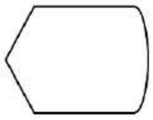
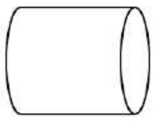
Common flowchart symbols

These flowchart shapes and symbols are some of the most common types you'll find in most flowchart diagrams.

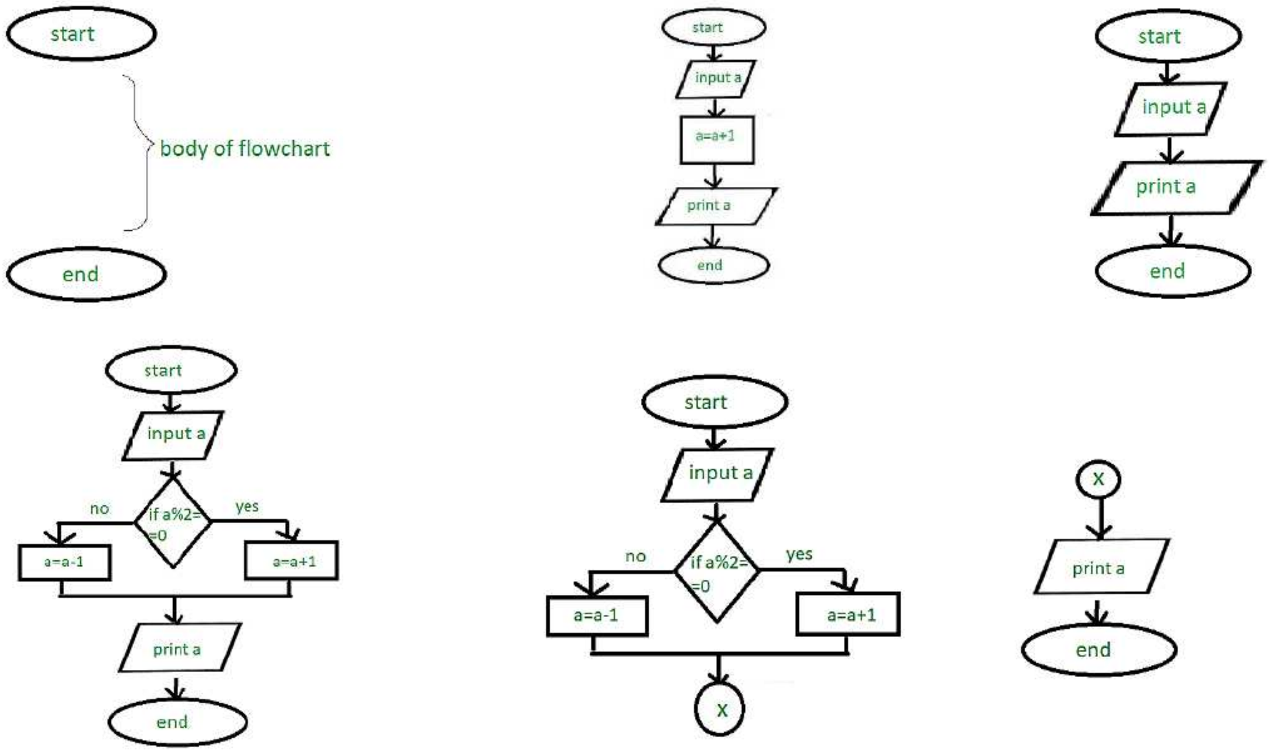
Flowchart Symbol	Name	Description
	Process symbol	Also known as an "Action Symbol," this shape represents a process, action, or function. It's the most widely-used symbol in flowcharting.
	Start/End symbol	Also known as the "Terminator Symbol," this symbol represents the start points, end points, and potential outcomes of a path. Often contains "Start" or "End" within the shape.

Flowchart Symbol	Name	Description
	Document symbol	Represents the input or output of a document, specifically. Examples of and input are receiving a report, email, or order. Examples of an output using a document symbol include generating a presentation, memo, or letter.
	Decision symbol	Indicates a question to be answered — usually yes/no or true/false. The flowchart path may then split off into different branches depending on the answer or consequences thereafter.
	Connector symbol	Usually used within more complex charts, this symbol connects separate elements across one page.
	Off-Page Connector/Link symbol	Frequently used within complex charts, this symbol connects separate elements across multiple pages with the page number usually placed on or within the shape for easy reference.
	Input/Output symbol	Also referred to as the "Data Symbol," this shape represents dat that is available for input or output as well as representing resources used or generated. While the paper tape symbol also represents input/output, it is outdated and no longer in commor use for flowchart diagramming.
	Comment/ Note symbol	Placed along with context, this symbol adds needed explanation or comments within the specified range. It may be connected by a dashed line to the relevant section of the flowchart as well.
	Flow	This arrow line represents the flow of the algorithm or process. I represents the direction of the process flow. in all the previous examples, we included arrows in every step to display the flow o the program. arrow increases the readability of the program.
	Database symbol	Represents data housed on a storage service that will likely allow for searching and filtering by users.

Flowchart Symbol	Name	Description
	Paper tape symbol	An outdated symbol rarely ever used in modern practices or process flows, but this shape could be used if you're mapping ou processes or input methods on much older computers and CNC machines.
	Predefined process symbol	Indicates a complicated process or operation that is well-known or defined elsewhere.
	Internal storage symbol	Commonly used to map out software designs, this shape indicates data that is stored within internal memory.
	Manual input symbol	Represents the manual input of data into a field or step in a process, usually through a keyboard or device. Example scenario includes the step in a login process where a user is prompted to enter data manually.
	Manual operation symbol	Indicates a step that must be done manually, not automatically.
	Merge symbol	Combines multiple paths to become one.
	Multiple documents symbol	Represents multiple documents or reports.
	Preparation symbol	Differentiates between steps that prepare for work and steps that actually do work. It helps introduce the setup to another step within the same process.
	Stored data symbol	Also known as "Data Storage" symbol, this shape represents where data gets stored within a process.
	Delay symbol	Represents a segment of delay in a process. It can be helpful to indicate the exact length of delay within the shape.

Flowchart Symbol	Name	Description
	Or symbol/ Summing junction symbol	this shape indicates that the process flow continues two paths o more/ Sums the input of several converging paths.
	Display symbol	This shape is useful to indicate where information will get displayed within a process flow.
	Hard disk symbol	Indicates where data is stored within a hard drive, also known as direct access storage.

Example of Flowcharts:



Q1. Draw a flowchart to find the greatest number among the 2 numbers.

Algorithm:

1. Start

2. Input 2 variables from user

3. Now check the condition If a > b, goto step 4, else goto step 5.

4. Print a is greater, goto step 6

5. Print b is greater

6. Stop

Q2. Draw a flowchart to check whether the input number is odd or even

Algorithm:

1. Start

2. Put input a

3. Now check the condition if (a % 2 == 0), goto step 5. else goto step 4

4. Now print ("number is odd") and goto step 6

5. print ("number is even")

6. Stop

Q3. Draw a flowchart to print numbers from 1 to 10.

Algorithm:

1. Start

2. Now initialize c = 1

3. Now we check the condition if c < 11, then goto step 4 otherwise goto step 6.

4. Print c

5. c = c + 1 then goto step 3

6. Stop

Q4. Draw a flowchart to print the first 5 multiples of 3.

Algorithm:

1. Start

2. Now initialize c = 1

3. Now check the condition if c <=5, then goto step 4. otherwise goto step 6

*4. Print 3 * c*

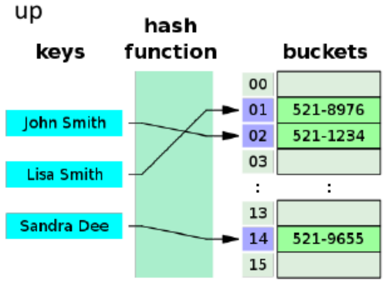
5. c += 1. Then goto step 3.

6. Stop

Data Structures

Definition

In computer science, a **data structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, B-trees are particularly well-suited for implementation of databases, while compiler implementations usually use hash tables to look up identifiers.



Data structures provide a means to manage huge amounts of data efficiently, such as large databases and internet indexing services. Usually, efficient data structures are a key to designing efficient algorithms. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design.

So, the way in which the various data elements are organized in memory with respect to each other is called a **data structure**. Data structures are the most convenient way to handle data of different types including abstract data type for a known problem. Again, problem solving is an essential part of every scientific discipline. To solve a given problem by using a computer, you need to write a program for it.

A program consists of two components: **algorithm and data structure**.

Many different algorithms can be used to solve the same problem. Similarly, various types of data structures can be used to represent a problem in a computer.

To solve the problem in an efficient manner, you need to select a combination of algorithms and data structures that provide maximum efficiency. Here, efficiency means that the algorithm should work in minimal time and use minimal memory. In addition to improving the efficiency of an algorithm, the use of appropriate data structures also allows you to overcome some other programming challenges, such as :

– simplifying complex problems

– creating standard, reusable code components

– creating programs that are easy to understand and maintain

Types of Data Structures

Data can be organized in many different ways; therefore, you can create as many data structures as you want. However, data structures have been classified in several ways.

Basically, data structures are of two types

: linear data structure and nonlinear data structure.

Linear data structure: a data structure is said to be linear if the elements form a sequence i.e., while traversing sequentially, we can reach only one element directly from another.

For example: Array, linked list, Queue etc.

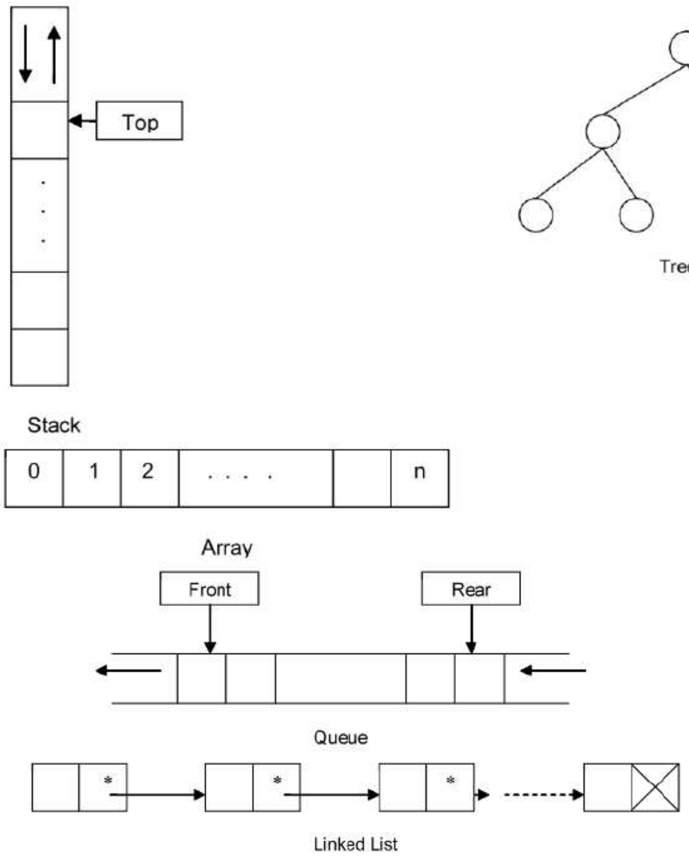
Nonlinear data structure: elements in a nonlinear data structure do not form a sequence i.e., each item or element may be connected with two or more other items or elements in a non-linear arrangement. Moreover, removing one of the links could divide the data structure into two disjoint pieces.

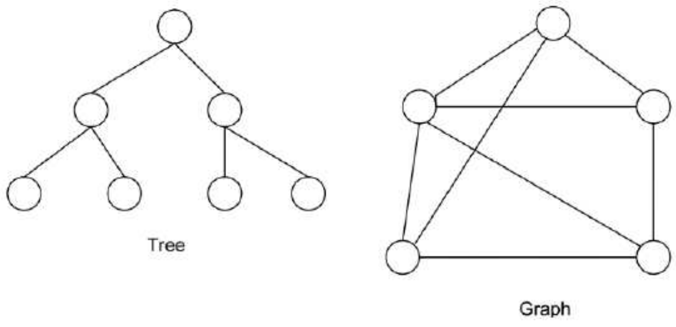
For example: Trees and Graphs etc.

The following figures shows the linear and nonlinear data structures. All these data structures are designed to hold a collection of data items.

Linear data structure

Nonlinear data structure





Data structures are generally based on the ability of a computer to fetch and store data at any place in its memory, specified by an address—a bit string that can be itself stored in memory and manipulated by the program. Thus the record and array data structures are based on computing the addresses of data items with arithmetic operations, while the linked data structures are based on storing addresses of data items within the structure itself. Many data structures use both principles, sometimes combined in non-trivial ways