Unit -1

What is a Computer?

A computer is an electronic device that processes data according to instructions provided by software programs. It takes input (data), processes it using a central processing unit (CPU), stores information, and produces output (results) to perform various tasks.

Types of Computers

There are various types of computers that are used today based on the need of user. Some of the types are:

- **Desktop:** Desktops are mainly used for regular use and they have separate components mounted together like the monitor, keyboard, mouse, CPU etc. Since the system is primarily kept on a desk for better usability it is called a desktop.

- **Laptop:** Laptops are a portable version of desktops, with all the components integrated into a single unit thus providing mobility to the system. They are great for on-the-go work and come with built-in webcams, Bluetooth and Wi-Fi.

- **Servers:** Servers are special types of computers that are used to manage network resources. They provide services to other systems and computers. Some of the primary tasks of servers include creating databases, hosting and providing support to other applications.

- **Tablets:** Tablets are even more portable than laptops. They are smaller than laptops but are larger than smartphones. They come with touchscreens which makes them perfect for browsing the web, consuming content and personal communications.

- **Other devices:** Other devices include smartphones, game consoles, Smart TVs etc.

Types of Computers Based on Functionality-

- **Analog computers:** In analog computers data is stored using continuous physical quantities. A mechanical integrator is an example of an analogue computer.

- **Digital computers:** These are the most common types of computers found in the market today. Data is processed in digital computers using discrete values. Smartphone is a common example of digital computers.
- **Hybrid computers:** These are a combination of both analogue and digital computers. Examples include complex medical equipment.

Types of Computers Based on Size

- **Microcomputers:** Microcomputers are meant for individual use. They are small, compact and very small. For example smartphones and desktops.

- **Minicomputers:** They are used in businesses that are mid-sized and are more powerful than microcomputers. Servers are an example of minicomputers.
- **Mainframe computers:** These are used by large organizations. They help in the processing of bulk data.
- **Supercomputers:** These are extremely powerful computers that help in carrying out complex calculations. They aren't meant for personal use and are often used for research purposes.

Types of Computers Based on Processing Power

- **Personal computers (PCs):** These are the most common type of computer and are designed for personal use. PCs include desktops, laptops, and tablets.
- **Servers:** Servers are designed to manage and distribute resources and data to multiple users or devices. They are often used in businesses or organizations to store and share data and run applications.
- **Mainframes**: Mainframe computers are large, powerful machines that are designed to handle massive amounts of data and perform complex operations. They are often used in large corporations or government agencies.
- **Supercomputers:** Supercomputers are extremely powerful computers that are designed to process data at extremely high speeds. They are often used for scientific research and other specialized applications.
- **Embedded systems:** Embedded systems are small computers that are built into other devices, such as appliances, cars, and medical devices. They are designed to perform specific functions and operate without human intervention.
- **Wearable computers:** Wearable computers are small, portable devices that are worn on the body, such as smartwatches or fitness trackers. They are designed to track data and provide information on the go.

What is Hardware?

Hardware refers to the physical components of a computer that you can touch and see. It includes all the devices and machinery required to make a computer function. Hardware performs tasks like storing data, processing information, and displaying results. Without hardware, there would be no platform for software to run.

Types of Hardware:

- Central Processing Unit (CPU) - Executes instructions and performs calculations.
- Memory (RAM) - Temporarily stores data that the CPU needs during operation.
- Storage Devices (HDD/SSD) - Store data permanently, even when the computer is turned off.
- Input Devices - Allow users to interact with the computer (e.g., keyboard, mouse).
- Output Devices - Display or produce results of the computer’s processing (e.g., monitors, printers).

What is Software?

Software is a set of instructions that tells the computer what to do when to do, it and how to do it. Examples are, the paint that we use in Microsoft, WhatsApp, Web-browsers and games etc, all are types of different software.

How does the Software Work with Hardware?

When you give input (e.g., typing a letter on a keyboard), the hardware (keyboard) sends this input to the software. The software then converts the input into a machine-readable language (binary) that the CPU can process. The output (e.g., the letter ‘A’) is then displayed on the screen as a result of this process.

Example Process:

1. You press the Shift key and the A key on your keyboard.
2. The software translates this into machine code and tells the CPU that the letter ‘A’ should be displayed.
3. The CPU processes the input, and the monitor shows the letter 'A'.

Let's discuss some important component of computer in details-

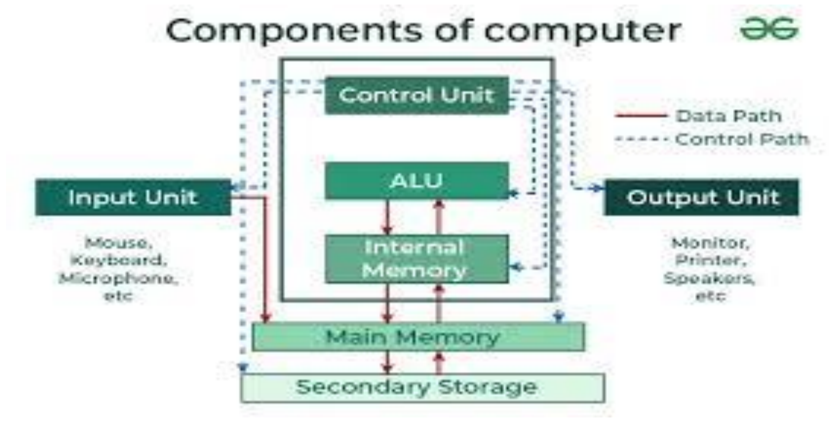
Component	**Description**
Central Processing Unit (CPU)	The CPU often referred to as the "brain" of the computer. It’s responsible for executing instructions, performing calculations, and handling tasks that ensure the system runs efficiently. The CPU processes input data and transforms it into useful information. It consists of the **Arithmetic Logic Unit (ALU)** and **Control Unit (CU)**.
Motherboard	The main circuit board connects and allows communication between all computer components.
Memory (RAM)	Random Access Memory (RAM) stores data temporarily for quick access while the computer is running.
Storage	Includes Hard Disk Drives (HDD) and Solid-State Drives (SSD) that store data permanently.
Input Devices	Devices are used to input data into the computer. Examples: keyboard, mouse, scanner.
Output Devices	Devices that display or output the results of the computer’s processing. Examples: printer, speakers.

Advantage and Disadvantages of Computers-

Speed and Efficiency: Computers process large amounts of data quickly, automating tasks and saving time.	**Health Issues:** Prolonged use can lead to eye strain, back pain, and repetitive strain injuries.
Accuracy: Computers perform calculations and tasks with high precision, reducing human error.	**Cost:** Initial setup, maintenance, and software updates can be expensive.
Storage Capacity: They can store vast amounts of data in a compact space, easily accessible and organized.	**Dependency:** Over-reliance on computers can reduce critical thinking and manual skills
Multitasking: Can handle multiple tasks simultaneously, improving productivity	**Security Risks:** Vulnerable to hacking, viruses, and data breaches if not properly secured
Versatility: Support a wide range of applications, from education to entertainment and business.	**Technical Issues:** Hardware or software failures can disrupt work and require technical expertise to fix.

Components of a Computer-

The functional components of a digital computer include the Input Unit, which takes in data; the CPU, which processes data with its Control Unit (CU), Arithmetic Logic Unit (ALU), and Registers; the Memory Unit, which stores data temporarily (RAM) or permanently (HDD/SSD); the Output Unit, which displays results; and the Bus System, which connects and transfers data between components. These parts work together to execute tasks and provide results.



The **functional components of a computer** are the key parts that work together to process and manage data. These include the **Input Unit** for receiving data, the **CPU** for processing it, the **Memory Unit** for storing information, the **Output Unit** for displaying results, and the **Bus System** that connects all parts. These components help the computer perform tasks efficiently**.**

1. Input Unit

- **Purpose:** Captures data and instructions from users or external sources.
- **Function:** Converts user input into binary signals that the computer can process.
- **Common Devices:**
  - Keyboard, Mouse, Touchscreens
  - Scanners, Sensors, Stylus pens
  - Voice Assistants (e.g., Siri, Alexa)
  - Biometric devices (face/fingerprint recognition)
  - Iot-based inputs from smart devices

2. Central Processing Unit (CPU) – The Brain of the Computer

The **CPU** executes instructions and controls all internal operations. In 2025, CPUs will often have multiple **cores** and **threads** to handle parallel processing efficiently.

Components of CPU:

a. Arithmetic Logic Unit (ALU)

- Performs arithmetic operations (add, subtract, multiply, divide).
- Handles logical operations (comparison, decision-making).
- Supports AI/ML tasks using built-in vector/matrix operations (in modern CPUs).

b. Control Unit (CU)

- Directs the operations of all computer parts.
- Decodes instructions and coordinates data flow.
- Sends control signals to memory and I/O devices.

c. Registers

- **High-speed memory locations** within the CPU.
- Temporarily store instructions, addresses, and intermediate data.
- Examples: Accumulator, Instruction Register, Program Counter, Address Register.
- **Modern CPUs** include 64-bit or even 128-bit registers for faster processing.

3. Memory / Storage Unit

The **memory unit** holds data and instructions before, during, and after processing.

a. Primary Memory (Main Memory):

- **RAM (Random Access Memory):** Temporarily stores data during execution.
  - Types in 2025: DDR5, LPDDR5X, and emerging **MRAM**.
- **ROM (Read-Only Memory):** Stores boot-up instructions and firmware.
- **Cache Memory:** Ultra-fast memory between CPU and RAM (L1, L2, L3 levels).

b. Secondary Storage:

- Used for long-term data storage.
- **Examples**: SSDs (NVMe drives), HDDs, flash drives, and cloud storage.
- **Modern Trend:** Use of **Cloud Integration** and **hybrid storage models**.

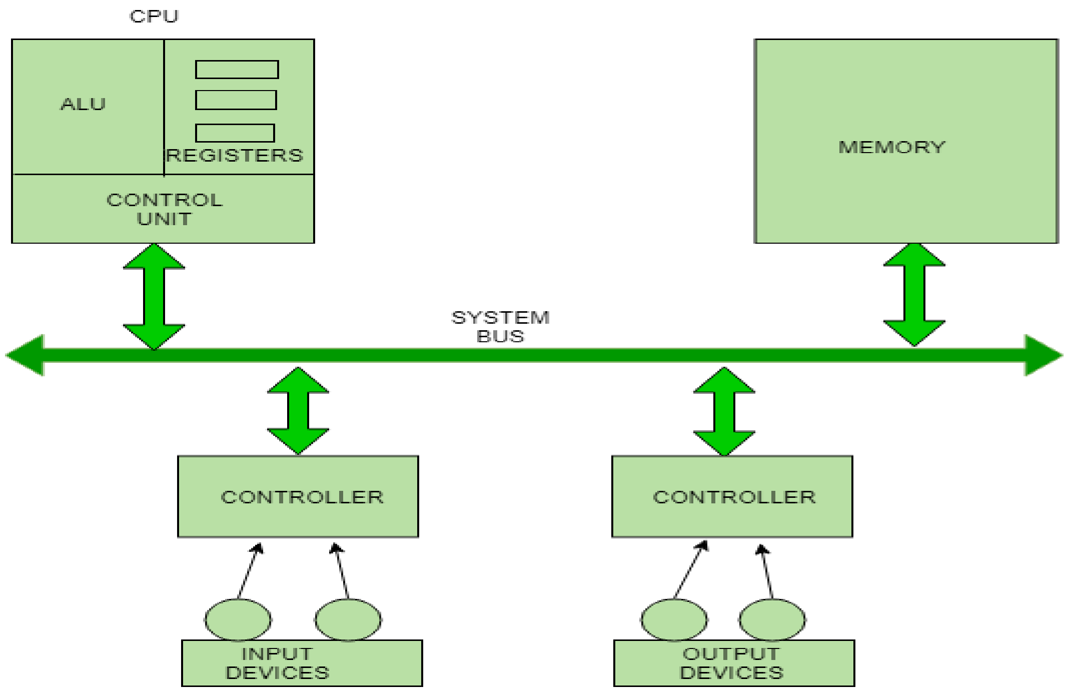
4. Output Unit

- **Purpose:** Converts processed data (binary) into a form users can understand.

- **Examples:**
    - Visual: Monitors (LED, OLED, 4K/8K displays)
    - Print: Printers (Inkjet, Laser, 3D Printers)
    - Audio: Speakers, Headphones

Interconnection between Functional Components-
A computer consists of an input unit that takes input, a CPU that processes the input and an output unit that produces output. All these devices communicate with each other through a common bus. A bus is a transmission path, made of a set of conducting wires over which data or information in the form of electric signals is passed from one component to another in a computer. The bus can be of three types – **Address bus, Data bus and Control Bus.**

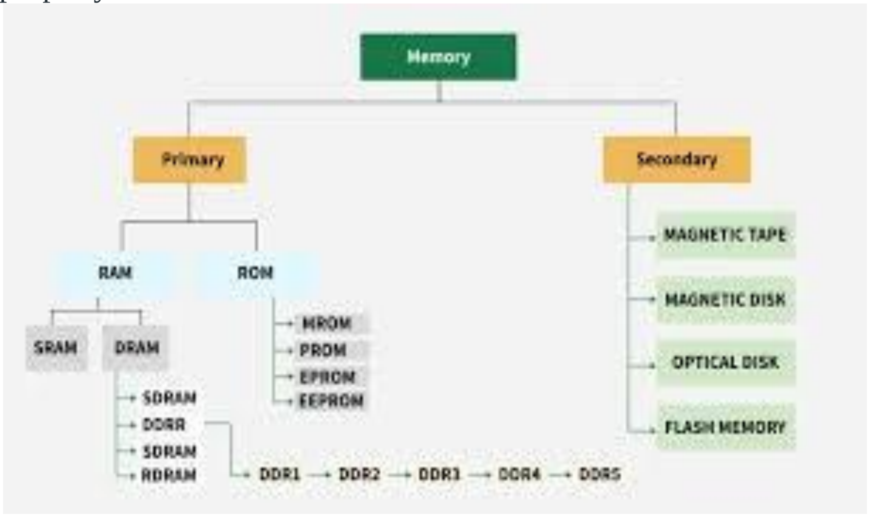
The following figure shows the connection of various functional components:



The address bus carries the address location of the data or instruction. The data bus carries data from one component to another, and the control bus carries the control signals. The system bus is the common communication path that carries signals to/from the CPU, main memory and input/output devices. The input/output devices communicate with the system bus through the controller circuit, which helps in managing various input/output devices attached to the computer.

Computer Memory:
Memory is the electronic storage space where a computer keeps the instructions and data it needs to access quickly. It's the place where information is stored for immediate use. Memory

is an important component of a computer, as without it, the system wouldn't operate correctly. The computer's operating system (OS), hardware, and software all rely on memory to function properly.



Computer memory functions similarly to the human brain, storing data, information, and instructions. It acts as a storage unit or device where data to be processed and the instructions necessary for processing are kept. Both input and output data can be stored in memory.

Types of Computer Memory

In general, computer memory is divided into two types:

- Primary memory
- Secondary memory

Now we discuss each type of memory one by one in detail:

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:

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 and information on this memory will be lost. RAM is used for booting up or starting the computer. It temporarily stores programs/data which has to be executed by the processor. RAM is of two types:

- **S RAM (Static RAM):**S RAM 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.
- **D RAM (Dynamic RAM):**D RAM 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 S RAM.

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 are stored on it. It contains some electronic fuses that can be programmed for a piece of specific information. The information is stored in the ROM in binary format. It is also known as permanent memory. ROM is of four types:

.

- **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):**EPROM 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 reprogram 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.

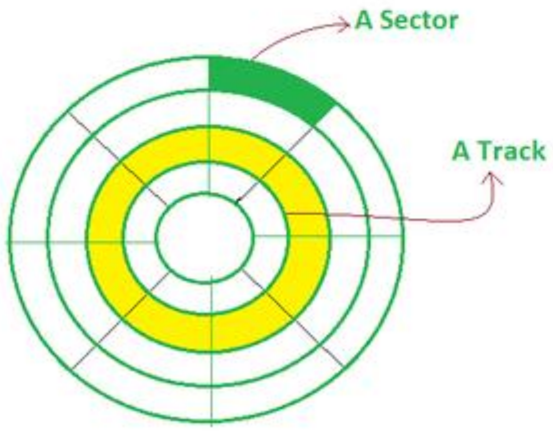
Characteristics of Secondary Memory

- 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

1. 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 backwards or rewound.

2. Magnetic Disks: A magnetic disk 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.

3.**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-
CD - ROM

- It's called a 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 aluminium.
- 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.

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 a 5.25-inch disk is 650 MB,5.2GB etc.

DVDs
The term "DVD" stands for "Digital Versatile/Video Disc," and there are two sorts of DVDs:

- DVDR (writable)
- 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.

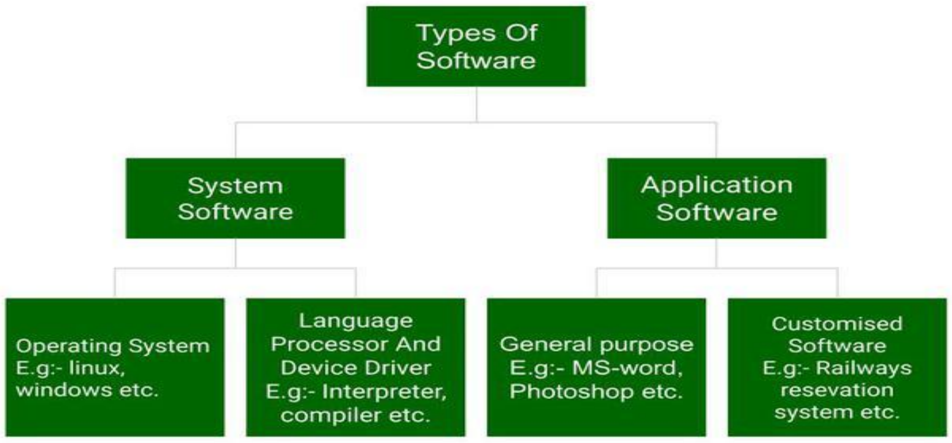
- **DVD-R**: DVD-R is a writable optical disc that can be used just once. It's a DVD that can be recorded. It's a lot like WORM. DVD-ROMs have capacities ranging from 4.7 to 17 GB. The capacity of 3.5 inch disk is 1.3 GB.

  Computer memory is important for storing and processing data in a computer. Data is stored in the form of bits and bytes. As the amount of data grows, it's important to understand how larger units like kilobytes, megabytes, and gigabytes are used to store and manage bigger files.

Software and its Types:
In a computer system, the software is basically a set of instructions or commands that tell a computer to do a specific task that serves its users.

- A software is not a physical thing like hardware, it rather makes the hardware work as per user requirements by giving instructions.
- Examples of software are MS-Word, MS-Excel, PowerPoint and Web Browser

Types of Software-
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. Now we will briefly describe each type and its subtypes:

1. **System Software**
   - Operating System (like Windows and Linux)
   - Language Processor
   - Device Driver
2. **Application Software**
   - General Purpose Software

- Customize Software
- Utility 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.

- 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.

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 computer memory, CPU, printer, hard disk, etc., and provides an interface to the user, which helps the user to interact with the computer system.
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.

Features of System Software

Let us discuss some of the features of System Software:

- Closer to the computer system.
- Written in a low-level language in general.
- Difficult to design and understand.
- Fast in speed(working speed).
- Less interactive for the users in comparison to application software.

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.

- 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.
- Examples include word processors, spreadsheets, database management, inventory, payroll programs, etc.

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.

Features of Application Software

Let us discuss some of the features of Application Software:

- Perform more specialized tasks like word processing, spreadsheets, email, etc.
- Mostly, the size of the software is big, so it requires more storage space.
- More interactive for the users, so it is easy to use and design.
- Easy to design and understand.
- Written in a high-level language in general.

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.

Generations of Computers:
A computer can be simply defined as a machine that can calculate. However, modern computers are no longer just computing devices. They can perform a variety of tasks. When they were first introduced, computers were massive and could fill an entire room. Over time, computers have become more powerful, as well as smaller. Technology changed significantly through the different generations of computers over the years.

There are five generations of computers. These are:

First Generation (1940 to 1956)

The first-generation computer was a fragile device. It was made of glass and the central technology was a vacuum tube. The computers of this generation were heavy and bulky. The main purpose was storage, computing, and control. The machines were huge and occupied entire rooms. They also used a lot of electricity.

Computers of the first generation conducted operations using machine language, a low-level programming language. It sometimes took days or even weeks before the operator could pose a new problem. The input was based on a punch card and paper strip, and output was displayed on a printout. ENIAC, EDVAC, and UNIVAC were the main computers of this generation.

Second Generation (1956 to 1963)

Second-generation computers were developed using transistor technology instead of vacuum tubes. The size of the computers was smaller, and the calculation time was also shorter compared with computers of the first generation.

Bell Labs invented the transistor. The use of transistors helped the computers perform powerfully and quickly. The second-generation computers introduced programming language, CPU, memory, input, and output units. FORTRAN, ALGOL, and COBOL were the languages used for programming in this era.

Third Generation (1964 to 1971)

Third-generation computers were developed using integrated circuit (IC) technology. The size of the third-generation computer was smaller than that of the second-generation computer. Compared with the second-generation computer, the calculation time was also much shorter. Third-generation computers consumed less power and generated less heat. Maintenance costs were also low. In this generation of computers, semiconductors were used. This increased efficiency and speed. For the very first time, computers became accessible to a large audience because they were smaller and cheaper.

Fourth Generation (1971 – Present)

Microprocessor technology was used for the development of the fourth-generation computer. The advantage of this technology is that a single microprocessor chip can hold all the required circuits to perform logic, arithmetic, as well as control functions. The Fourth Generation computers are small and portable. A very low amount of heat is generated. They are also more user-friendly and are capable of multiprocessing, multiprogramming, etc. Computers of this generation have a semiconductor memory. The fourth-generation brought the concept of private computers and computer networks. Some examples of fourth-generation computers are STAR 1000, IBM PC, APPLE II, Alter 8800, etc.

Fifth Generation (Present and Beyond)

The period from 2010 to the present is roughly considered the period of the fifth-generation computers. These computers use AI-based technology, which is commonly used in speech recognition, medical science, and the entertainment industry. Fifth-generation computers give high performance and have a high storage capacity. The computers are fast and can perform multiple tasks at one time. Popular fifth-generation advanced technologies include quantum computing, nanotechnology, and parallel processing. Desktops, laptops, tablets, smartphones, etc, are examples of fifth-generation computers.

Computer languages:

Introduction-

A programming language is a set of instructions and syntax used to create software programs. Some of the key features of programming languages include:

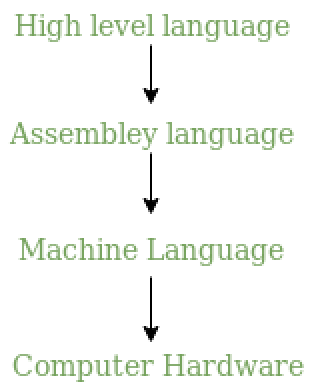
1. **Syntax**: The specific rules and structure used to write code in a programming language.
2. **Data Types**: The type of values that can be stored in a program, such as numbers, strings, and booleans.

3. **Variables**: Named memory locations that can store values.
4. **Operators**: Symbols used to perform operations on values, such as addition, subtraction, and comparison.
5. **Control Structures**: Statements used to control the flow of a program, such as if-else statements, loops, and function calls.
6. **Libraries** and Frameworks: Collections of pre-written code that can be used to perform common tasks and speed up development.
7. **Paradigms**: The programming style or philosophy used in the language, such as procedural, object-oriented, or functional.

Examples of popular programming languages include Python, Java, C++, JavaScript, and Ruby. Each language has its own strengths and weaknesses and is suited for different types of projects.

A programming language is a formal language that specifies a set of instructions for a computer to perform specific tasks. It's used to write software programs and applications, and to control and manipulate computer systems.

Hierarchy of Computer language –



Characteristics of a programming Language -

- A programming language must be simple, easy to learn and use, have good readability, and be human recognizable.
- A portable programming language is always preferred.
- Programming language's efficiency must be high so that it can be easily converted into a machine code and its execution consumes little space in memory.
- A programming language must be consistent in terms of syntax and semantics.

Language Processors: Assembler, Compiler and Interpreter-
Computer programs are generally written in high-level languages (like C++, Python, and Java). A language processor, or language translator, is a computer program that convert source code from one programming language to another language or to machine code (also known as object code). They also find errors during translation.

Compilers, interpreters, translate programs written in high-level languages into machine code that a computer understands and assemblers translate programs written in low-level or assembly language into machine code. In the compilation process, there are several stages.

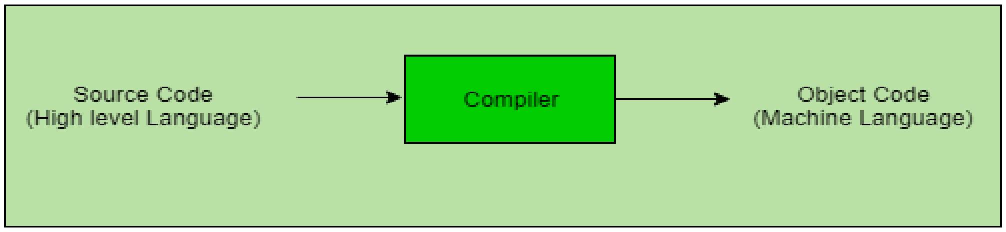
Types of Language Processors

The language processors can be any of the following three types:

1.Compiler

The language processor that reads the complete source program written in high-level language as a whole in one go and translates it into an equivalent program in machine language is called a Compiler. Example: C, C++, JAVA etc.

In a compiler, the source code is translated to object code successfully if it is free of errors. The compiler specifies the errors at the end of the compilation with line numbers when there are any errors in the source code. The errors must be removed before the compiler can successfully recompile the source code again the object program can be executed number of times without translating it again.

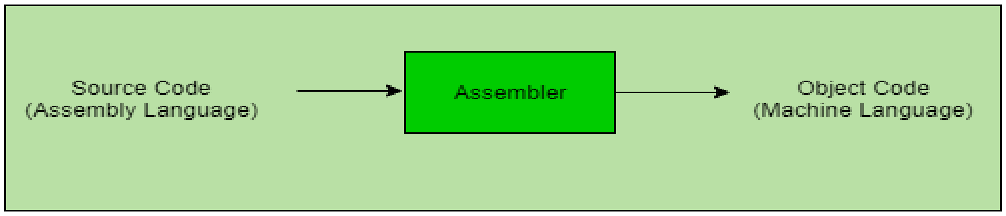


2. Assembler

The Assembler is used to translate the program written in Assembly language into machine code. The source program is an input of an assembler that contains assembly language instructions. The output generated by the assembler is the object code or machine code understandable by the computer.

Code written in assembly language is some sort of mnemonics (instructions) like ADD, MUL, MUX, SUB, DIV, MOV and so on. and the assembler is basically able to convert these mnemonics in binary code. Here, these mnemonics also depend upon the architecture of the machine.

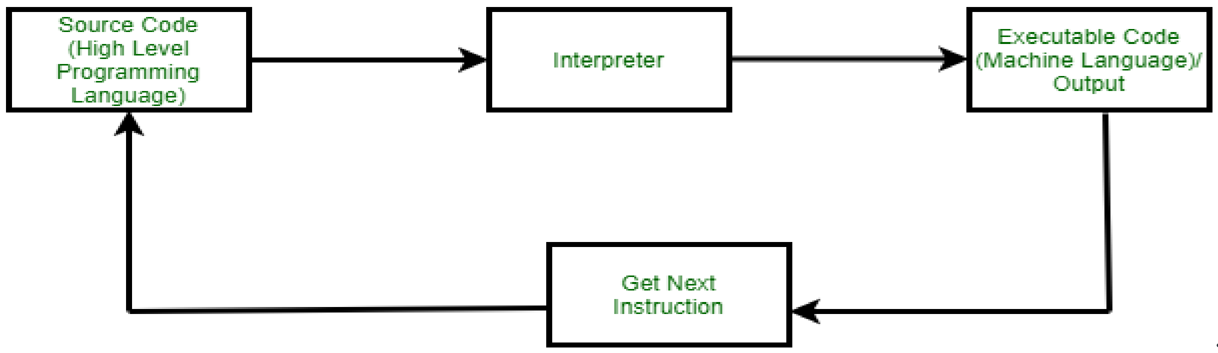
For example, the architecture of intel 8085 and intel 8086 are different.



3. Interpreter

The translation of a single statement of the source program into machine code is done by a language processor and executes immediately before moving on to the next line is called an interpreter. If there is an error in the statement, the interpreter terminates its translating process at that statement and displays an error message. The interpreter moves on to the next line for execution only after the removal of the error. An interpreter translates one line at a time and then executes it.

Example: Perl, Python etc


Source Code (High Level Programming Language)
Interpreter
Executable Code (Machine Language)/ Output
Get Next Instruction

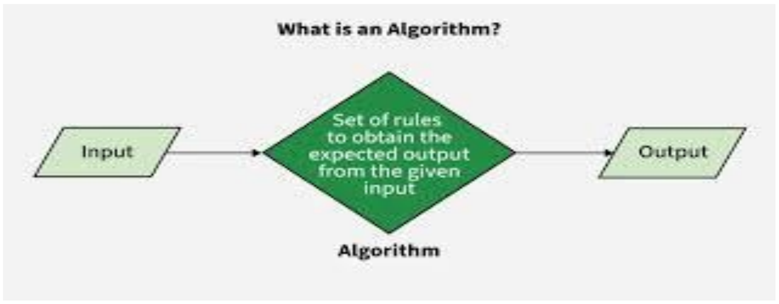
Difference Between Compiler and Interpreter-

Compiler	Interpreter
A compiler is a program that converts the entire source code of a programming language into executable machine code for a CPU.	An interpreter takes a source program and runs it line by line, translating each line as it comes to it.
The compiler takes a large amount of time to analyze the entire source code but the overall execution time of the program is comparatively faster.	An interpreter takes less amount of time to analyze the source code but the overall execution time of the program is slower.
The compiler generates the error message only after scanning the whole program, so debugging is comparatively hard as the error can be present anywhere in the program.	Its Debugging is easier as it continues translating the program until the error is met.
The compiler requires a lot of memory for generating object codes.	It requires less memory than a compiler because no object code is generated.
Generates intermediate object code.	No intermediate object code is generated.

Compiler	Interpreter
For Security purpose compiler is more useful.	The interpreter is a little vulnerable in case of security.
Examples: C, C++, C#	Examples: Python, Perl, JavaScript.

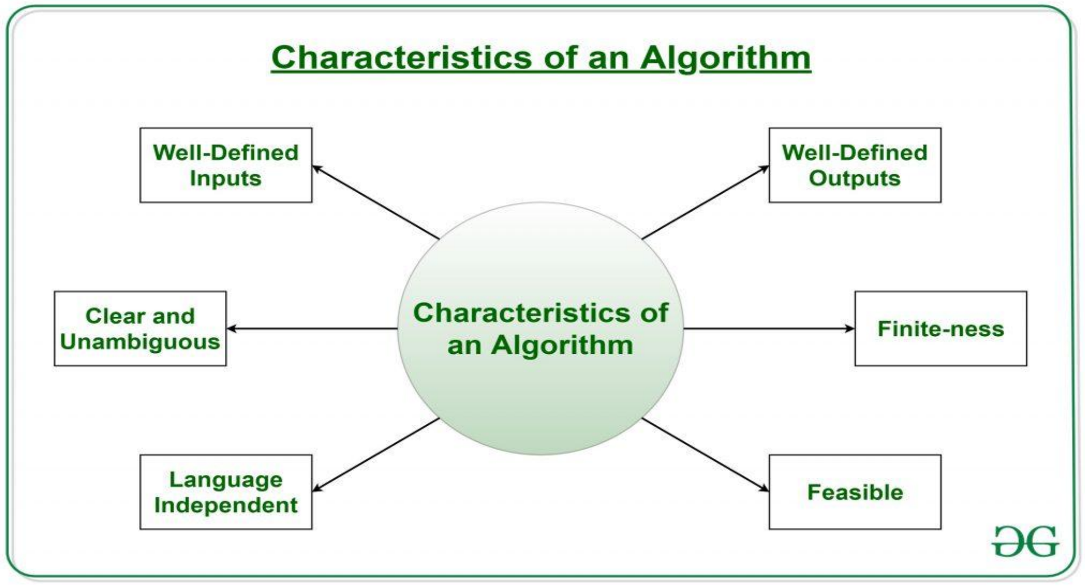
Introduction to Algorithms:
The word **Algorithm** means "*A set of finite rules or instructions to be followed in calculations or other problem-solving operations*" Or "*A procedure for solving a mathematical problem in a finite number of steps that frequently involves recursive operations*".

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


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

What is the need for algorithms?
1. Algorithms are necessary for solving complex problems efficiently and effectively.
2. They help to automate processes and make them more reliable, faster, and easier to perform.
3. Algorithms also enable computers to perform tasks that would be difficult or impossible for humans to do manually.
4. They are used in various fields such as mathematics, computer science, engineering, finance, and many others to optimize processes, analyze data, make predictions, and provide solutions to problems.

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 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. It may or may not take input.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well. It should produce at least 1 output.
- **Finite-ness:** 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.
- **Input**: An algorithm has zero or more inputs. Each that contains a fundamental operator must accept zero or more inputs.
- **Output**: An algorithm produces at least one output. Every instruction that contains a fundamental operator must accept zero or more inputs.
- **Definiteness:** All instructions in an algorithm must be unambiguous, precise, and easy to interpret. By referring to any of the instructions in an algorithm one can clearly understand what is to be done. Every fundamental operator in instruction must be defined without any ambiguity.
- **Finiteness:** An algorithm must terminate after a finite number of steps in all test cases. Every instruction which contains a fundamental operator must be terminated within a finite amount of time. Infinite loops or recursive functions without base conditions do not possess finiteness.

- **Effectiveness:** An algorithm must be developed by using very basic, simple, and feasible operations so that one can trace it out by using just paper and pencil.

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 an 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

Limitations of Algorithms-.

Algorithms are limited because some problems are theoretically undecidable (no algorithm can solve them) or intractable (too inefficient to solve in a reasonable time). Other limitations include computational resources like runtime and memory.

Theoretical Limitations

- **Undecidable Problems:**

  Certain problems are inherently impossible for any algorithm to solve, regardless of its efficiency. A classic example is the halting problem, which asks if an arbitrary program will eventually halt or run forever.

- **Intractable Problems:**

  These problems can be solved algorithmically but require such an immense amount of time or resources as the input size grows that they become impractical to solve in a realistic timeframe.

Practical Limitations

  - **Runtime and Efficiency:**

The speed and resource usage of an algorithm are limited. An algorithm's performance can vary drastically depending on the specific implementation and the nature of the input, with worst-case scenarios sometimes being prohibitively slow.

- **Data and Context:**

Algorithms are built on specific models and frameworks, which may not accurately represent complex real-world scenarios. They are also trained on historical data and can inherit and amplify biases present in that data, leading to skewed or unfair outcomes.

Designing the algorithm-

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

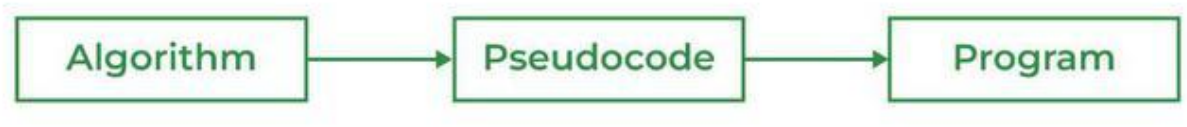
Algorithm: Calculate the square of a number

1. Start.
2. Input the number (N) whose square you want to find.
3. Multiply the number (N) by itself.
4. Store the result of the multiplication in a variable (result).
5. Output the value of the variable (result), which represents the square of the input number.
6. End.

How to express an Algorithm?

1. **Natural Language:-** Here we express the Algorithm in the natural English language. It is too hard to understand the algorithm from it.
2. **Flowchart:-** Here we express the Algorithm by making a graphical/pictorial representation of it. It is easier to understand than Natural Language.
3. **Pseudo Code:-** Here we express the Algorithm in the form of annotations and informative text written in plain English which is very much similar to the real code but as it has no syntax like any of the programming languages, it can't be compiled or interpreted by the computer. It is the best way to express an algorithm because it can be understood by even a layman with some school-level knowledge.

   *Pseudocode is the **intermediate state between an idea and its implementation(code)** in a high-level language.*



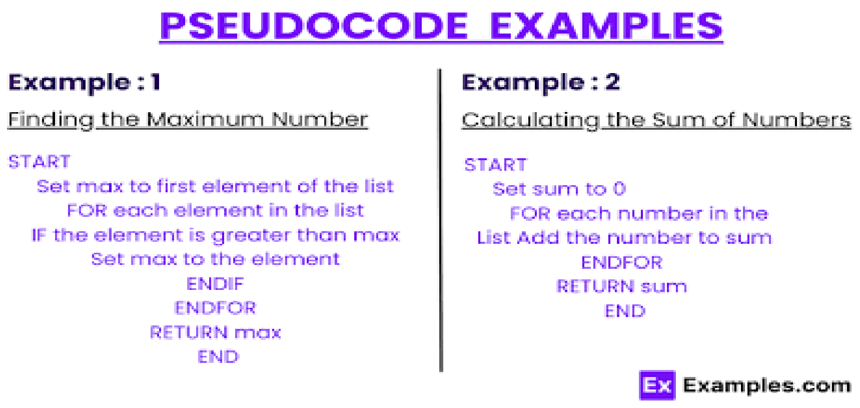
How to write Pseudocode?
Before writing the pseudocode of any algorithm the following points must be kept in mind.

- Organize the sequence of tasks and write the pseudocode accordingly.
- At first, establishes the main goal or the aim. **Example:**
  *This program will print first **N** numbers of Fibonacci series.*
- Use standard programming structures such as **if-else**, **for**, **while**, and **cases** the way we use them in programming. Indent the statements if-else, for, while loops as they are indented in a program, it helps to comprehend the decision control and execution mechanism. It also improves readability to a great extent. **Example:**

```
IF "1"
  print response
     "I AM CASE 1"
IF "2"
  print response
     "I AM CASE 2"
```

- Use appropriate naming conventions. The human tendency follows the approach of following what we see. If a programmer goes through a pseudo code, his approach will be the same as per that, so the naming must be simple and distinct.
- Reserved commands or keywords must be represented in **capital letters**. **Example:** if you are writing IF…ELSE statements then make sure IF and ELSE be in capital letters.
- Check whether all the sections of a pseudo code are complete, finite, and clear to understand and comprehend. Also, explain everything that is going to happen in the actual code.
- Don't write the pseudocode in a programming language. It is necessary that the pseudocode is simple and easy to understand even for a layman or client, minimizing the use of technical terms.

Pseudocode Examples-



PSEUDOCODE EXAMPLES

Example : 1

Finding the Maximum Number

```
START
  Set max to first element of the list
    FOR each element in the list
  IF the element is greater than max
      Set max to the element
          ENDIF
          ENDFOR
        RETURN max
          END
```

Example : 2

Calculating the Sum of Numbers

```
START
  Set sum to 0
    FOR each number in the
 List Add the number to sum
          ENDFOR
        RETURN sum
          END
```

Ex Examples.com

Difference between Algorithm and Pseudocode-

Algorithm	Pseudocode
An Algorithm is used to provide a solution to a particular problem in form of a well-defined step-based form.	A Pseudocode is a step-by-step description of an algorithm in code-like structure using plain English text.
An algorithm only uses simple English words	Pseudocode also uses reserved keywords like if-else, for, while, etc.
These are a sequence of steps of a solution to a problem	These are fake codes as the word pseudo means fake, using code like structure and plain English text
There are no rules to writing algorithms	There are certain rules for writing pseudocode
Algorithms can be considered pseudocode	Pseudocode cannot be considered an algorithm
It is difficult to understand and interpret	It is easy to understand and interpret

Difference between Flowchart and Pseudocode-

Flowchart	Pseudocode
A Flowchart is pictorial representation of flow of an algorithm.	A Pseudocode is a step-by-step description of an algorithm in code like structure using plain English text.
A Flowchart uses standard symbols for input, output decisions and start stop statements. Only uses different shapes like box, circle and arrow.	Pseudocode uses reserved keywords like if-else, for, while, etc.
This is a way of visually representing data, these are nothing but the graphical representation of the algorithm for a better understanding of the code	These are fake codes as the word pseudo means fake, using code like structure but plain English text instead of programming language

Flowchart	Pseudocode
Flowcharts are good for documentation	Pseudocode is better suited for the purpose of understanding

The Main Constructs of Pseudocode-
At its core, pseudocode is the ability to represent six programming constructs (always written in uppercase): SEQUENCE, CASE, WHILE, REPEAT-UNTIL, FOR, and IF-THEN-ELSE. These constructs — also called keywords — are used to describe the control flow of the algorithm.

1. **SEQUENCE** represents linear tasks sequentially performed one after the other.
2. **WHILE** is a loop with a condition at its beginning.
3. **REPEAT-UNTIL** is a loop with a condition at the bottom.
4. **FOR** is another way of looping.
5. **IF-THEN-ELSE** is a conditional statement changing the flow of the algorithm.
6. **CASE** is the generalization form of IF-THEN-ELSE.

IF-THEN-END IF:

This structure executes a block of statements only if a specified condition is true.

Code
```
IF condition THEN
    statement(s) to execute if condition is true
END IF
```

Example:

Code
```
IF age >= 18 THEN
    OUTPUT "You are eligible to vote."
END IF
```

Example:

```
IF score >= 60 THEN
    OUTPUT "You passed the exam."
ELSE
```

```
    OUTPUT "You failed the exam."
END IF
```

CASE (Multiway Branching):

The CASE construct is used when a single variable or expression can have multiple distinct values, each triggering a different set of actions.

Code

```
CASE expression OF
   value1: sequence 1
   value2: sequence 2
   ...
   OTHERS: default sequence (optional)
END CASE
```

Example:

Code

```
CASE dayOfWeek OF
   "Monday": OUTPUT "Start of the work week."
   "Friday": OUTPUT "End of the work week."
   OTHERS: OUTPUT "Mid-week or weekend."
END CASE
```

FOR Loop

Example: Print Numbers from 1 to 5

```
START
   FOR i FROM 1 TO 5
      PRINT i
   END FOR
END
```

WHILE Loop

Example: Print Numbers from 1 to 5

```
START
  SET i TO 1
  WHILE i <= 5 DO
    PRINT i
    INCREMENT i BY 1
  END WHILE
END
```

REPEAT-UNTIL Loop

Example: Keep Asking for a Positive Number

```
START
    REPEAT
        INPUT number
    UNTIL number > 0
    PRINT "You entered a positive number:", number
END
```

CASE/SWITCH Statement

Example: Print the Day of the Week Based on Number Input

```
START
  INPUT dayNumber
  CASE dayNumber OF
```

```
        1: PRINT "Sunday"
        2: PRINT "Monday"
        3: PRINT "Tuesday"
        4: PRINT "Wednesday"
        5: PRINT "Thursday"
        6: PRINT "Friday"
        7: PRINT "Saturday"
        OTHERWISE: PRINT "Invalid day number"
    END CASE
END
```