
Go, change the world

# Fundamentals of Programming using C

## Unit -1

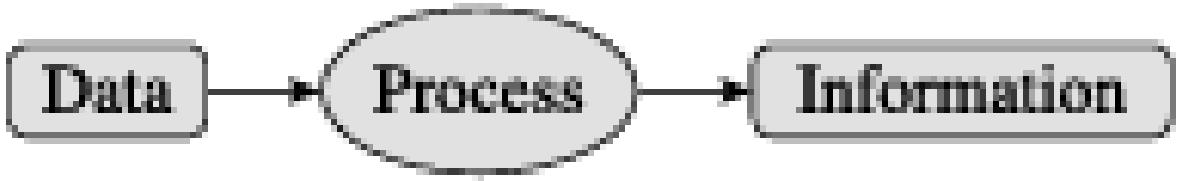
### Introduction to C Programming

# Contents

- Introduction to Programming:
  - Definition of a computer.
  - Components of computer system
  - Programming Languages.
  - Design and implementation of efficient programs.
- Program Design Tools: Algorithms, Flowcharts and Pseudocodes.
- Types of Errors.

# What is a computer?

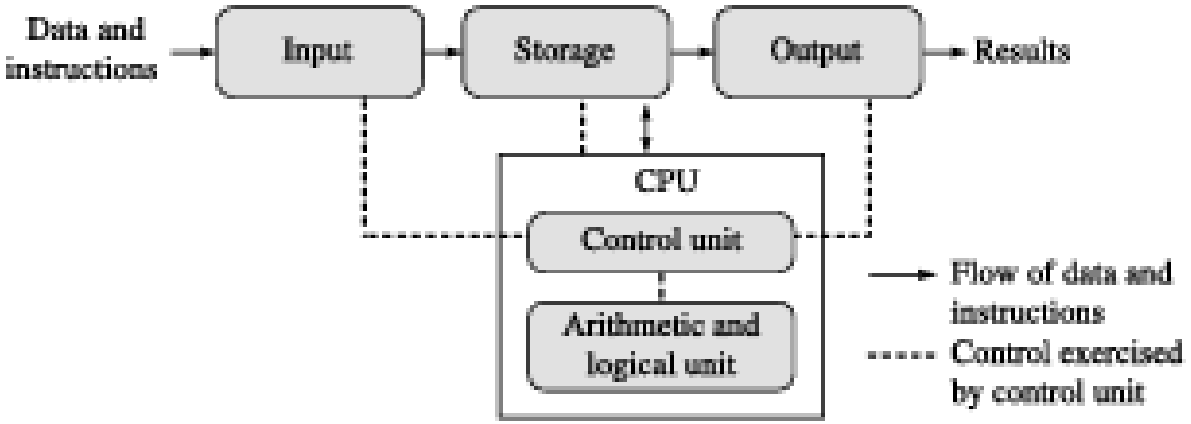
- A computer, in simple terms, can be defined as an electronic device that is designed to accept data, perform the required mathematical and logical operations at high speed, and output the result.
- Data – Unprocessed facts
- Information- organized, interpreted and conceptualized



- An example of **data** vs **information**:
- **Data**: A list of numbers like "85, 90, 75, 95, 88."
- **Information**: "The average test score of the class is 86.6."
- In this case, the individual numbers (data) alone don't provide much meaning until they are processed into something useful.
- Like the average (information), which gives us an understanding of how the class performed overall.

# Basic Computer Organization

- A computer is an electronic device that performs the following operations:
- Accepting data or instructions
- Storing data
- Processing Data
- Displaying Results
- Controlling and Coordinating all the activities within a computer.



- **Input:** This is the process of entering data and instructions (also known as programs) into the computer system.
- The data and instructions can be entered by using different input devices such as keyboard, mouse, scanner, and trackball.
- Note that computers understand binary language, which consists of only two symbols (0 and 1)
- It is the responsibility of the input devices to convert the input data into binary codes.

- **Storage** refers to the process of saving data and instructions permanently in a computer so that they can be used for processing.
- It plays a vital role in a computer system by holding data, programs, intermediate results, and final outcomes of processing.
- There are two main types of storage in a computer: Primary (RAM)

and Secondary(SSDs,Hard drives, USB)

- **Processing** The process of performing operations on the data as per the instructions specified by the user (program) is called processing.
- Data and instructions are taken from the primary memory and transferred to the arithmetic and logical unit (ALU), which performs all sorts of calculations.
- The intermediate results of processing may be stored in the main memory, as they might be required again.
- When the processing completes, the final result is then transferred to the main memory.
- Hence, the data may move from the main memory to the ALU multiple times before the processing is over.

- **Output** Output is the process of giving the result of data processing to the outside world (external to the computer system).
- The results are given through output devices such as monitor and printer.
- Since the computer accepts data only in the binary form and the result of processing is also in the binary form, the result cannot be directly given to the user.
- The output devices, therefore, convert the results available in binary codes into a human-readable language before displaying it to the user

- **Control** The control unit (CU) is the central nervous system of the entire computer system.
- It manages and controls all the components of the computer system. The CU decides the manner in which instructions will be executed and operations performed.
- It takes care of the step-by-step processing of all operations that are performed in the computer.

# Components of a Computer System

- Components of a computer can be categorized on the basis of hardware and software.
- Hardware of a computer system includes:
  - Memory
  - Disks
  - Processor
  - Peripheral Devices/Input and Output Devices

# Memory

- Memory is an internal storage area in the computer, which is used to store data and programs either temporarily or permanently.
- Computer memory can be broadly divided into two groups—
- **primary memory** –data and instructions while processing, volatile
- **secondary memory** – long-term storage, non-volatile
- Secondary memory data access is very low speed as compared with the primary memory.
- Primary Memory Types –**RAM** and **ROM**

# RAM

- RAM is a volatile storage area within the computer that is typically used to store data temporarily, so that it can be promptly accessed by the processor.
- RAM is considered random access because any memory cell can be directly accessed if its address is known.
- **Static RAM** This is a type of RAM that holds data without an external refresh as long as it is powered.
- **Dynamic RAM** This is the most common type of memory used in personal computers, workstations, and servers today.
- A DRAM chip contains millions of tiny memory cells. Each cell is made up of a transistor and a capacitor and can contain 1 bit of information—0 or 1

# ROM

- **ROM** ROM refers to computer memory chips containing permanent or semi-permanent data. Unlike RAM, ROM is non-volatile;
- Most computers contain a small amount of ROM that stores critical programs such as the basic input/output system (BIOS)
- ROMs are used extensively in calculators and peripheral devices such as laser printers, whose fonts are stored in ROM

- **Programmable read-only memory** (PROM) It is also called one-time programmable ROM, and can be written to or programmed using a special device called a PROM programmer.
- • **Erasable programmable read-only memory** (EPROM) It is a type of ROM that can be erased and re-programmed.
- The EPROM can be erased by exposing the chip to strong ultraviolet light, typically for 10 minutes or longer, can then be rewritten with a process that again needs the application of a higher voltage.

**Electrically erasable programmable read-only memory (EEPROM)**

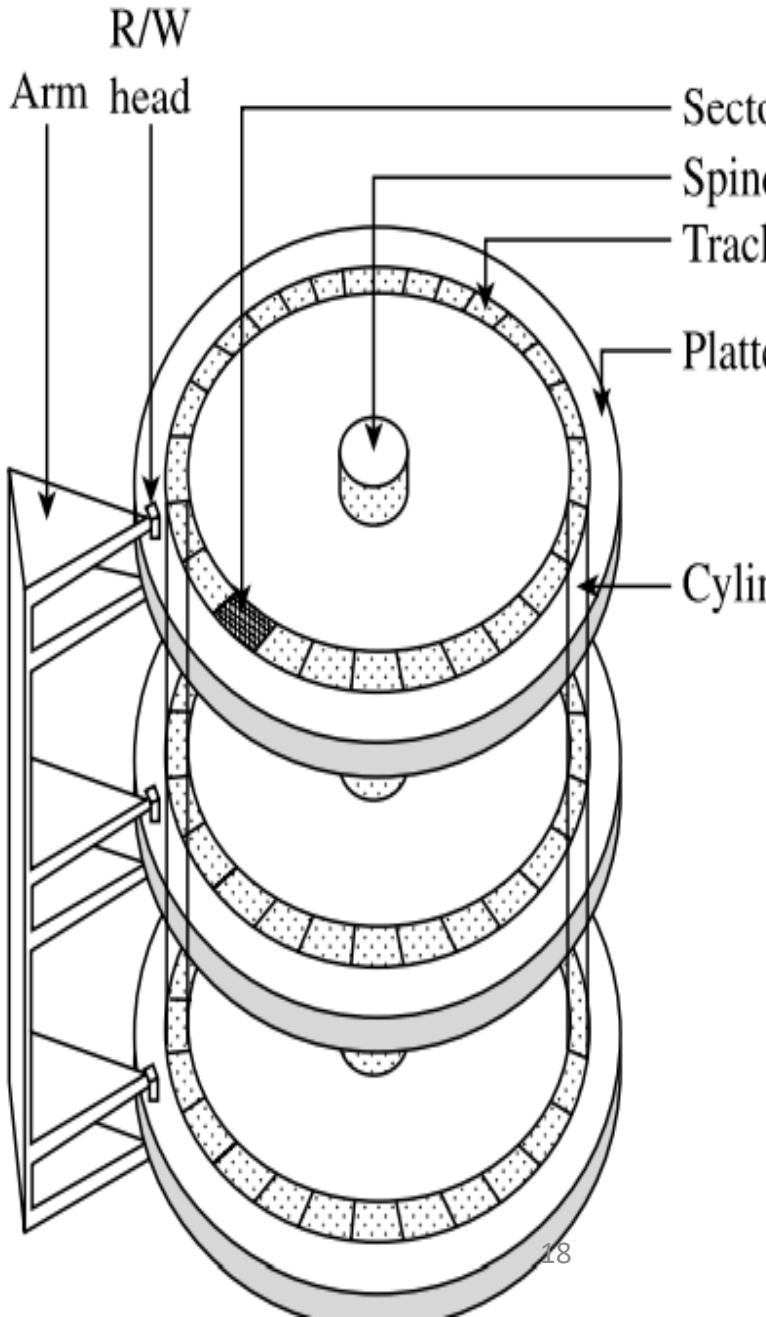
- It is based on a semiconductor structure similar to the EPROM, but allows the entire or selected contents to be electrically erased, then rewritten electrically, so that they need not be removed from the computer (or camera, MP3 player, etc.).
- The process of writing an EEPROM is also known as ***flashing.***

# Secondary Storage Devices

- Also known as *external memory* or *auxiliary storage*
- Differs from main memory –not directly accessible by CPU
- **Non-volatile**:Holds data even when the computer is switched off. Eg:hard disk.
- The computer usually uses its input/output channels to access data from the secondary storage devices to transfer the data to main memory
- CPU can read data stored in main memory in nanoseconds, while it takes to read data from secondary memory in milliseconds.
- The secondary storage devices are basically formatted according to a file system that organizes the data into files and directories.

# Hard Disks

- Part of the computer that stores all programs and files.
- If the drive is damaged, all data is lost
- A hard disk is a set of disks, stacked together like phonograph records, that has data recorded electromagnetically in concentric circles known as ***tracks***
- A single hard disk has several ***platters***(or disks) that are covered with magnetic recording medium.
- Each platter requires two read/write (R/W) heads, one for each side.



- R/W pivots back and forth over the platters to read /write data on them.
- Data is actually stored on the surface of a platter in **sectors** and **tracks.**
- The performance of a hard disk is dependent upon its **access time**
- Access time is a combination of three components:
    - Seek time
    - Rotational delay
    - Transfer time

- **Seek time:** Time taken to position the R/W head over the appropriate cylinder
- Seek time varies depending on the position of the access arm when the R/W command is received.
- 0 if the access arm is positioned over the desired rack, and max if access arm is positioned over the innermost track while the data that has to be accessed is stored on the outermost track.
- Varies from 10 to 100 ms.

- **Rotational delay** This is the time taken to bring the target sector to rotate under the R/W head. Assuming that the hard disk has 7,200 rpm, or 120 rotations per second, a single rotation is done in approximately 8 ms.
- **Transfer time** The time to transfer data or read/write to a disk is called the transfer rate.
- So, the **overall time required to access data = seek time + rotational delay + transfer time**.
- The sum of the seek time and the rotational delay is also known as *disk latency*. **Disk latency** is the time taken to initiate a transfer.

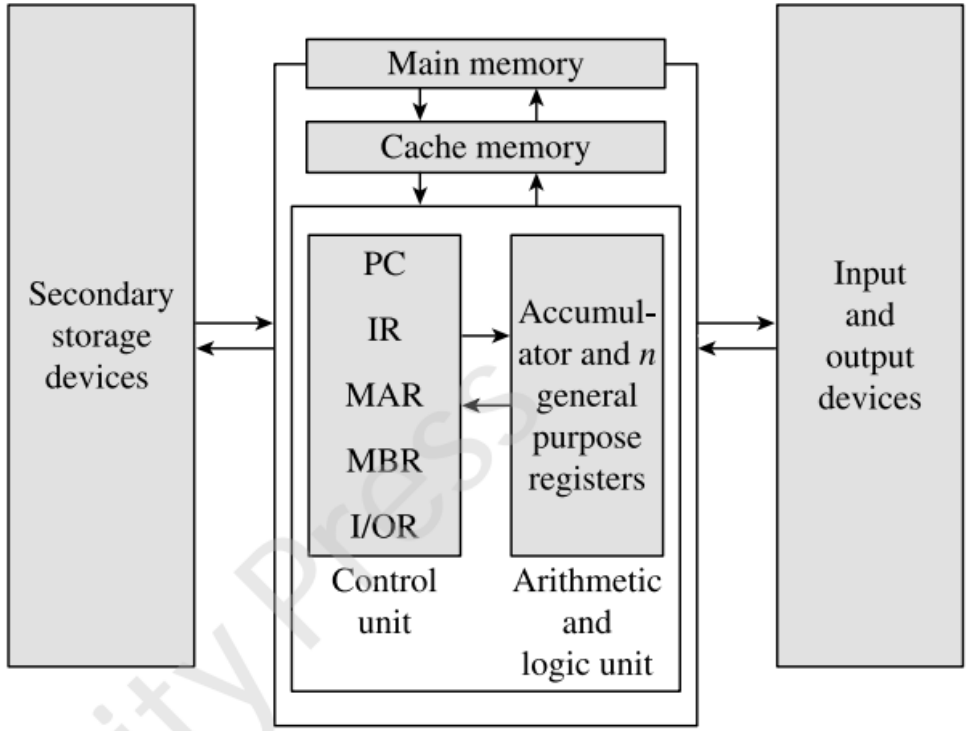
# Processor

- Basic processor consists of 2 parts-

ALU and Control Unit (CU)

- Besides these there are registers, execution unit,

a bus interface unit(BIU)




- **Execution unit** The execution unit mainly consists of the CU, ALU, and registers.
- ***Control unit (Brain of Computer )***The main function of the CU is to direct and coordinate the computer operations.
- It interprets the instructions (program) and initiates action to execute them.
- The CU controls the flow of data through the computer system and directs the ALU, registers, buses, and input/output (I/O) devices.
- In addition, the CU is responsible for fetching, decoding, executing instructions, and storing results

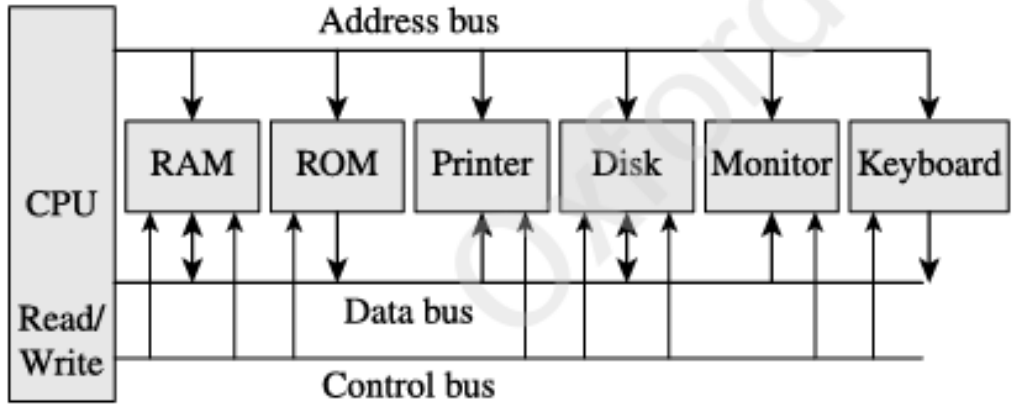
- ***Arithmetic and logic unit*** The ALU performs arithmetic (add, subtract, multiply, divide, etc.), comparison (less than, greater than, or equal to), and other operations.
- **Registers** A processor register is a computer memory that provides quick access to the data currently being used for processing.
- The ALU stores all temporary and final results in the processor registers.
- Registers are also used to store the instructions of the program currently being executed.
- There are different types of registers, each with a specific storage function.

- **Accumulator and general purpose registers** These are frequently used to store the data brought from the main memory and the intermediate results during program execution.
- **Special purpose registers** These include the **memory address register** (MAR) that stores the address of the data or instruction to be fetched from the main memory
- **memory buffer register (MBR)** that stores the data or instruction fetched from the main memory, **the instruction register (IR)** that stores the instructions currently being executed.
- **I/O register** that is used to transfer data or instructions to or from an I/O device, and the program counter that stores the address of the next instruction to be executed.

- **Instruction cycle** To execute an instruction, a processor normally follows a set of basic operations that are together known as an instruction cycle.
- The operations performed in an instruction cycle involve fetch, decode, execute, and store instructions

# Bus Interface Unit

- The BIU provides functions for transferring data between the execution unit of the CPU and other components of the computer system that lie outside the CPU.
- Every computer system has three different types of buses
  - **Address bus** -Carries memory address required by the CPU to read or write
  - **Data bus** –To transfer actual data between cpu,memory and I/O devices
  - **Control bus**-Carries control signals



- The BIU puts the contents of the program counter on the address bus.
- Once the memory receives an address from the BIU, it places the contents at that address on the data bus, which is then transferred to the IR of the processor through the MBR(Memory buffer register).
- The **MBR** temporarily holds the instruction that has been fetched from memory.
- From the MBR, the instruction is then transferred to the **Instruction Register (IR)**, which holds the current instruction that is about to be decoded and executed by the processor.
- At this time, the contents of the program counter are modified (e.g., incremented by 1) so that it now stores the address of the next instruction.

- **Instruction Set**
- The instruction set is a set of commands that instruct the processor to perform specific tasks.
- Tells the processor what it needs to do, from where to find data (register, memory, or I/O device), from where to find the instructions, and so on.
- Each instruction has a specific format, typically consisting of an **opcode**(e.g., ADD, SUB, LOAD). and **operands**(e.g., registers or memory locations).
- Instructions can be categorized based on different operations performed; like data transfer instructions, arithmetic instructions, logical operations, control flow operations, etc.
- Nowadays, computers come with a large set of instructions, and each processor supports its own instruction set.

- The **system clock** is a crucial component in a computer system that controls the timing and synchronization of all operations.
- It is typically implemented using a small **quartz crystal oscillator** that generates regular electrical pulses or "ticks."
- These ticks are used to coordinate the actions of the CPU and other components within the computer.
- Eg:**A single clock cycle** is one tick or pulse of the clock. Each operation in a processor takes a certain number of clock cycles to complete. For example, fetching an instruction from memory might take a few cycles.

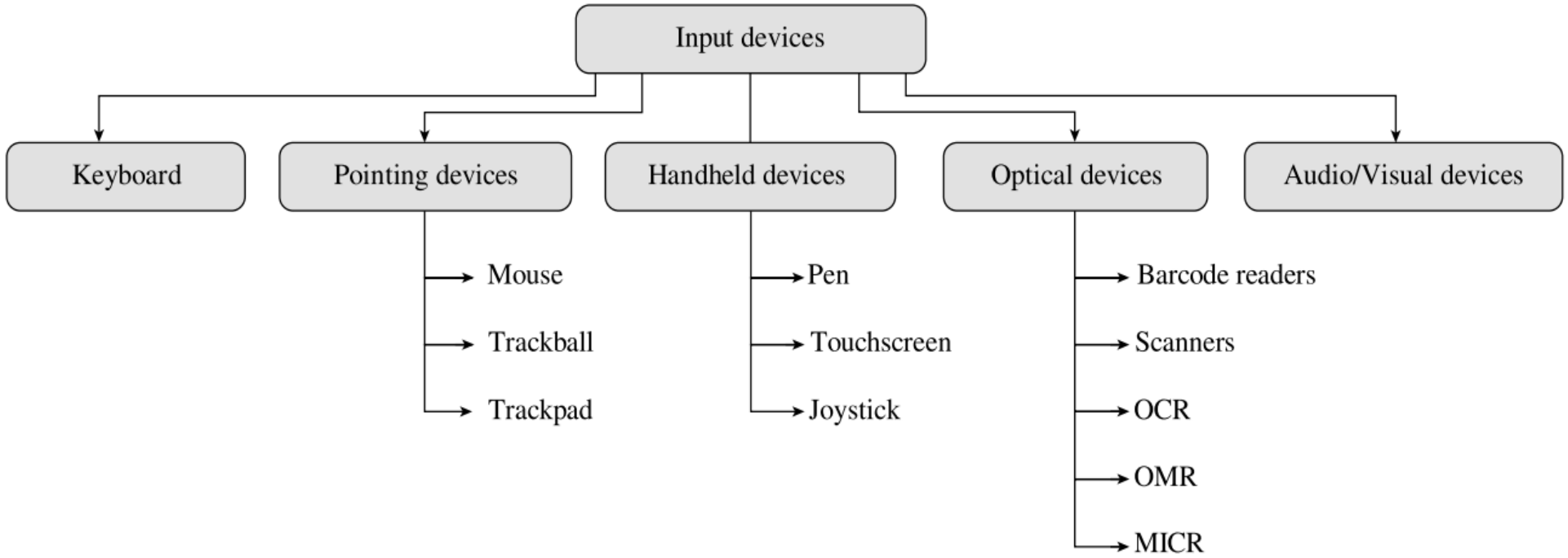
- **Processor Speed** The speed of PCs and minicomputers is usually specified in MHz or GHz.
- The clock frequency, typically measured in **Hertz (Hz)**, determines how many cycles occur per second. For example, a clock speed of 3 GHz means the system clock generates 3 billion cycles per second.
- The speed of a mainframe computer is measured in MIPS (millions of instructions per second) or BIPS (billions of instructions per second)
- Supercomputer is measured in MFLOPS (millions of floating-point operations per second), GFLOPS (giga or billions of floating-point operations per second).
- The reason for the variations in speed is that personal or minicomputers use a single processor to execute instruction
- Mainframes and supercomputers employ multiple processors to speed up their overall performance.

- **Pipeling and Parallel processing:**Most of the modern PCs support pipelining.
- Pipelining is a technique with which the processor can fetch the second instruction before completing the execution of the first instruction.
- This way, multiple instructions are processed simultaneously, though each instruction is at a different stage of execution.(fetch, decode,execute,store)
- In non-pipeline architectures, a processor had to wait for an instruction to complete all stages before it could fetch the next instruction, thereby wasting its time.
- However, with pipelining, processors can operate at a faster pace as they no longer have to wait for one instruction to complete before fetching the next instruction.
- Such processors that can execute more than one instruction per clock cycle are called superscalar processors.

- For example:While one instruction is in the execute stage, the next instruction can be in the decode stage, and a third instruction might be in the fetch stage.
- Analogy: Pipelining can be compared to an assembly line in a factory. Different workers (stages) work on different parts of a product (instructions) at the same time, increasing the overall efficiency.

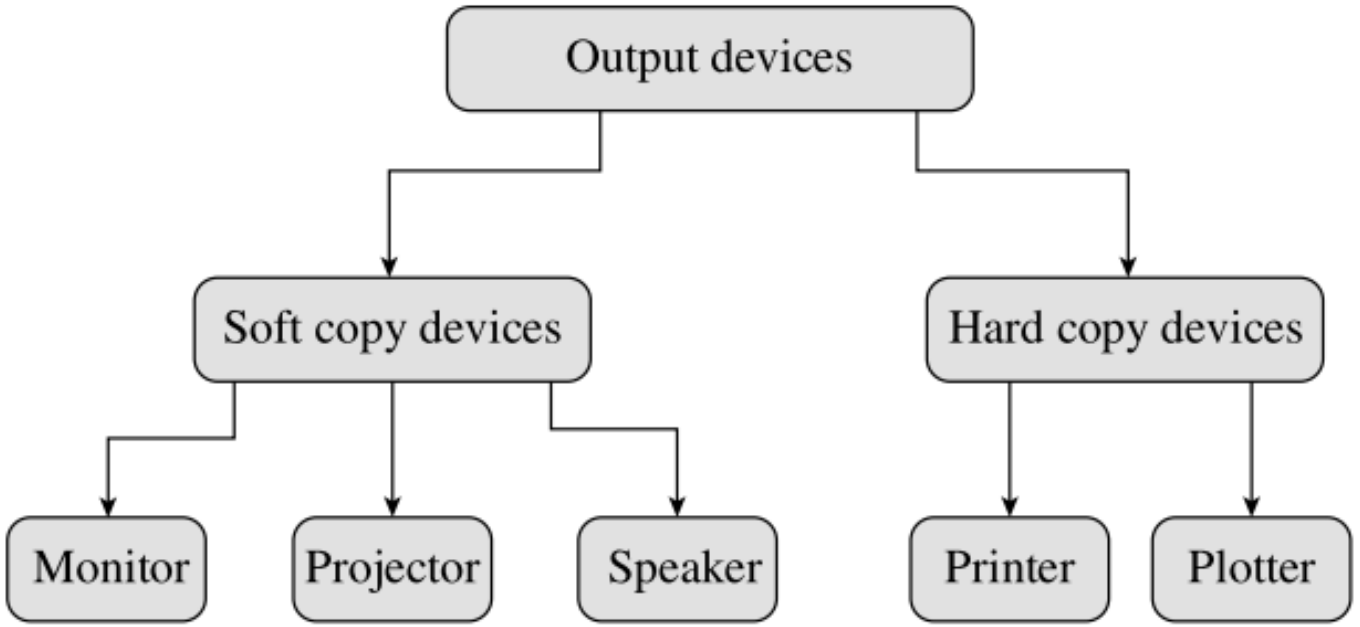
# Peripheral Devices/Input and Output Devices

- we need input and output devices for computers to be able to interact with its users
- There are different types of input/output devices, and each device has capabilities that differentiate it from the others.



**Input Devices** An input device is used to feed data and instructions into the computer.

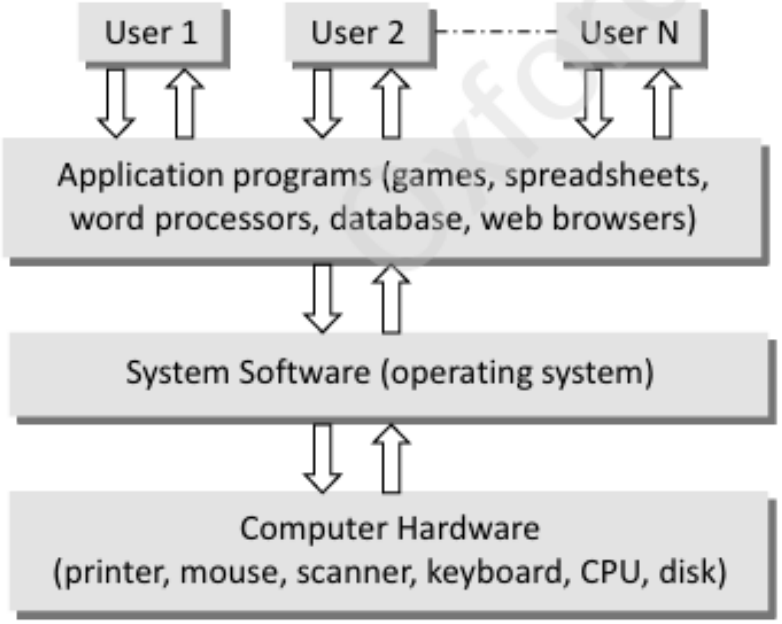
**Output Devices** Any device that outputs/gives information from a computer can be called an output device.



# Computer Software

- Computer software can be broadly classified as
  - System software
  - Application Software
- **Application software** is designed to solve a particular problem for users.
- It is generally what we think of when we say the word ‘computer programs’.
- Examples of application software include spreadsheets, database systems, desktop publishing systems, program development software, games, web browsers, among others.

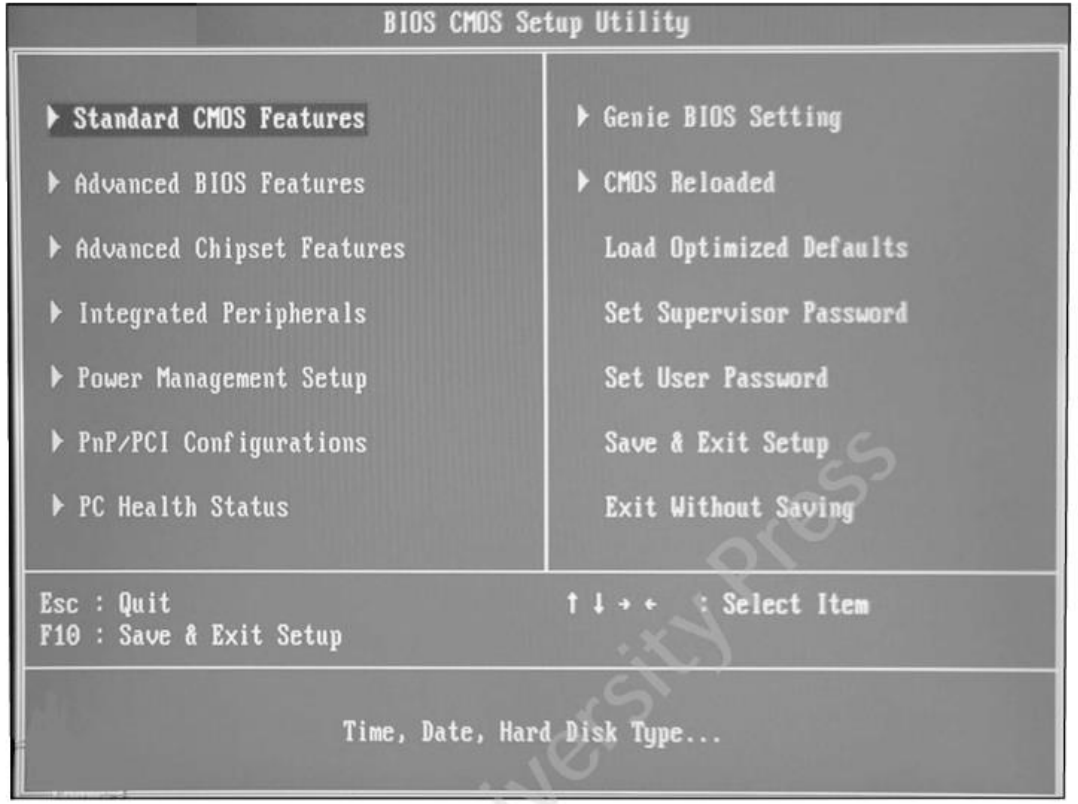
- **System software** refers to a type of software designed to manage and operate computer hardware.
- It serves as interface between the hardware and user applications, allowing users to interact with the system effectively.
- They include operating systems, device drivers, firmwares,utilities etc.



- **Computer BIOS and Device Drivers** The computer BIOS and device drivers provide basic functionality to operate and control the hardware connected to or built into the computer.
- BIOS or Basic Input/Output System is the first code run by the computer when it is switched on.
- The key role of BIOS is to load and start the operating system.
- When the computer starts, the first function that BIOS performs is to initialize and identify system devices such as the video display card, keyboard and mouse, hard disk, CD/DVD drive, and other hardware.

- In other words, the code in the BIOS chip runs a series of tests called POST (Power On Self Test) to ensure that the system devices are working correctly.
- BIOS then locates software held on a peripheral device such as a hard disk or a CD, and loads and executes that software, giving it control of the computer. This process is known as booting.
- BIOS is stored on ROM chip built-in the system and can be accessed by the user
- The BIOS menu can enable the user to configure hardware, set the system clock, enable or disable system components, and most importantly, select which devices are eligible to be a potential boot device and set various password prompts.

# BIOS Functions




- Initializes the system hardware
- Initializes system registers
- Initializes power management system
- Tests RAM
- Tests all the serial and parallel ports
- Initializes CD/DVD drive and hard disk controllers
- Displays system summary information

# Operating System

- The primary goal of an operating system is to make the computer system (or any other device in which it is installed like a cell phone) convenient and efficient to use.
- An operating system offers generic services to support user applications.
- From users' point of view the primary consideration is always the convenience.
- Users should find it easy to launch an application and work on it.

- An operating system ensures that the system resources (such as CPU, memory, I/O devices) are utilized efficiently.
- For example, there may be many service requests on a web server and each user request needs to be serviced.
- Moreover, there may be many programs residing in the main memory.
- Therefore, the system needs to determine which programs are currently being executed and which programs need to wait for some I/O operation.
- This information is necessary because the programs that need to wait can be suspended temporarily from engaging the processor.
- Hence, it is important for an operating system to have a control policy and algorithm to allocate the system resources.

# Utility Software

- Utility software is used to analyse, configure, optimize, and maintain the computer system.
- **Disk defragmenters** can be used to detect computer files whose contents are broken across several locations on the hard disk, and move the fragments to one location in order to increase efficiency.
- **Disk checkers** can be used to scan the contents of a hard disk to find files or areas that are either corrupted in some way, or were not correctly saved, and eliminate them in order to make the hard drive operate more efficiently.
- **Disk cleaners** can be used to locate files that are either not required for computer operation, or take up considerable amounts of space. Disk cleaners help users to decide what to delete when their hard disk is full.
- **Disk space analysers** are used for visualizing the disk space usage by getting the size for each folder (including subfolders) and files in a folder or drive.

- **Backup utilities** can be used to make a copy of all information stored on a disk. In case a disk failure occurs, backup utilities can be used to restore the entire disk. Even if a file gets deleted accidentally, the backup utility can be used to restore the deleted file.
- **Disk compression utilities** can be used to enhance the capacity of the disk by compressing/decompressing the contents of a disk.
- **File managers** can be used to provide a convenient method of performing routine data management tasks such as deleting, renaming, cataloging, moving, copying, merging, generating, and modifying data sets.
- **System profilers** can be used to provide detailed information about the software installed and hardware attached to the computer.

- **Anti-virus** utilities are used to scan for computer viruses.
- **Data compression** utilities can be used to output a file with reduced file size.
- **Cryptographic** utilities can be used to encrypt and decrypt files.
- **Launcher applications** can be used as a convenient access point for application software.
- **Registry cleaners** can be used to clean and optimize the Windows operating system registry by deleting the old
- **Network utilities** can be used to analyse the computer's network connectivity, configure network settings, check data transfer, or log events.
- **Command line interface (CLI)** and **Graphical user interface (GUI)** can be used to make changes to the operating system

# Compiler,Interpreter,Linker and Loader

- **Compiler** It is a special type of program that transforms the source code written in a programming language into machine language comprising just two digits, 1s and 0s
- The resultant code is called as ***object code,*** which is used to create executable program
- Therefore, a compiler is used to translate source code from a high-level programming language to a lower level language (e.g., assembly language or machine code).

- If the source code contains errors then the compiler will not be able to perform its intended task.
- Errors resulting from the code not conforming to the syntax of the programming language are called ***syntax errors.***
- Syntax errors may be spelling mistakes, typing mistakes, etc.
- Another type of error is ***logical error*** which occurs when the program does not function accurately. Logical errors are much harder to locate and correct.
- Until and unless the syntactical errors are rectified the source code cannot be converted into object code.

- **Interpreter: T**he interpreter, on the other hand, translates the instructions into an intermediate form, which it then executes. interpreter interprets the source code line by line
- This is in striking contrast with the compiler which compiles the entire code in one go
- Usually, a compiled program executes faster than an interpreted program.
- However, the big advantage of an interpreted program is that it does not need to go through the compilation stage during which machine instructions are generated.
- This process can be time consuming if the program is long. Moreover, the interpreter can immediately execute high-level programs.

- **Linker**(link editor binder) It is a program that combines object modules to form an executable program. The compiler automatically invokes the linker as the last step in compiling a program, to put together all the submodules object codes together.
- **Loader** It is a special type of program that copies programs from a storage device to main memory, where they can be executed.

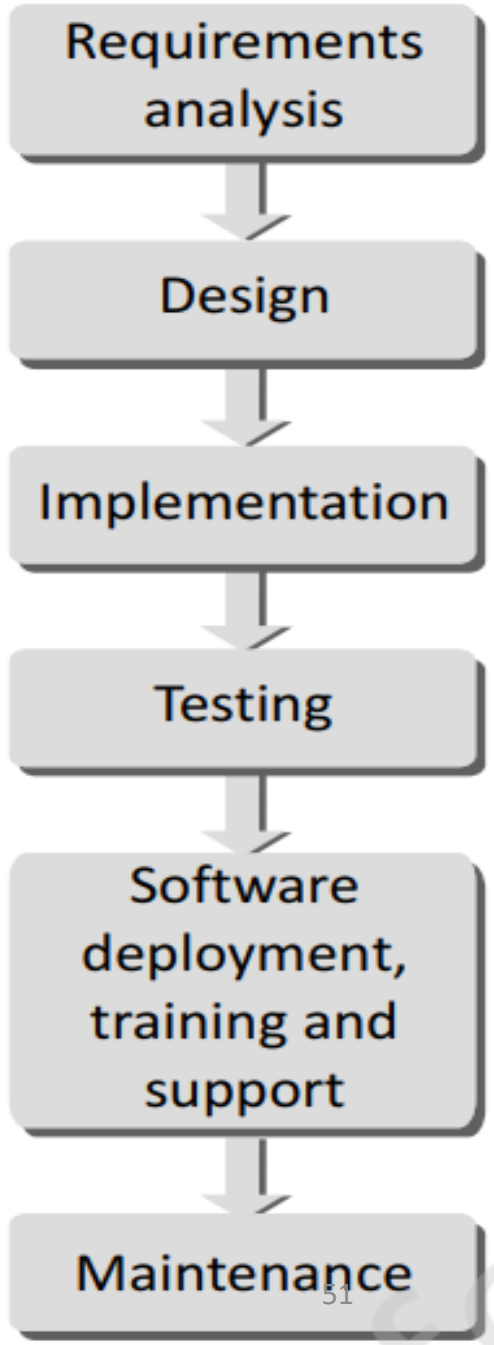
- **Application Software:**
- Application software is a type of computer software that employs the capabilities of a computer directly to perform a user-defined task.
- Typical examples of software applications are word processors, spreadsheets, media players, education software, CAD, CAM, data communication software, and statistical and operational research software

# Programming Languages

- A programming language is a language specifically designed to express computations that can be performed by the computer.
- Programming languages are used to create programs that control the behaviour of a system, to express algorithms, or as a mode of human–computer communication.
- programming languages have a vocabulary of syntax and semantics for instructing a computer to perform specific tasks.
- The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN

# DESIGN AND IMPLEMENTATION OF EFFICIENT PROGRAMS

- The entire program or software (collection of programs) development process is divided into a number of phases where each phase performs a well-defined task.
- Moreover, the output of one phase provides the input for its subsequent phase.
- **Requirements Analysis: A**ll the gathered users' requirements are analysed to pen down the scope or the objective of the overall software product.
- Documenting every identified requirement of the users in order to avoid any doubts or uncertainty regarding the functionality of the programs.
- The functionality, capability, performance, availability of hardware and software components are all analysed in this phase



•**Design:**

- A plan of actions is made before the actual development process could start.
- This plan will be followed throughout the development process.
- In the design phase the core structure of the software/program is broken down into modules.
- The solution of the program is then specified for each module in the form of algorithms, flowcharts, or pseudocodes.
- The design phase, therefore, specifies how the program/software will be built.

- **Implementation:**
- The designed algorithms are converted into program code using any of the high level languages.
- The particular choice of language will depend on the type of program like whether it is a system or an application program.
- The program codes are tested by the programmer to ensure their correctness. This phase is also called construction or code generation phase as the code of the software is generated in this phase.
- While constructing the code, the development team checks whether the software is compatible with the available hardware and other software components that were mentioned in the Requirements Specification Document created in the first phase.

- **Testing**
- In this phase, all the modules are tested together to ensure that the overall system works well as a whole product.
- Integration testing is done in this phase.
- Testing is done to ensure that the software is working as expected by the users' requirements that were identified in the requirements analysis phase.

- **Software Deployment, Training, and Support**
- After the code is tested and the software or the program has been approved by the users, it is then installed or deployed in the production environment.
- Software Training and Support is a crucial phase which is often ignored by most of the developers to use and fix up small errors

- **Maintenance**
- Maintenance and enhancements are ongoing activities which are done to cope with newly discovered problems or new requirements.
- Such activities may take a long time to complete as the requirement may call for addition of new code that does not fit the original design or an extra piece of code required to fix an unforeseen problem.
- As a general rule, if the cost of the maintenance phase exceeds 25% of the prior-phases cost then it clearly indicates that the overall quality of at least one prior phase is poor.
- In such cases, it is better to re-build the software (or some modules) before maintenance cost is out of control.

# Program Design tools, Algorithms,Flowcharts and Pseudocodes

- **Algorithms:**In general terms, an algorithm provides a blueprint to writing a program to solve a particular problem.
- It is considered to be an effective procedure for solving a problem in a finite number of steps.
- Algorithms are mainly used to achieve software re use.
- Once we have an idea or a blueprint of a solution, we can implement it in any high-level language, such as C, C++, Java, and so on.

- In order to qualify as an algorithm, a sequence of instructions must possess the following characteristics:
  - Be precise
  - Be unambiguous
  - Not even a single instruction must be repeated infinitely
  - After the algorithm gets terminated, the desired result must be obtained
- An algorithm has a finite number of steps and some steps may involve decision-making and repetition.
- Broadly speaking, an algorithm can employ any of the three control structures, namely, sequence, decision, and repetition.

# Sequence

- Sequence means that each step of the algorithm is executed in the specified order.
- Example: An algorithm to add two numbers
- Algorithm performs the steps in sequential order.

```
Step 1: Input the first number as A
Step 2: Input the second number as B
Step 3: SET SUM = A + B
Step 4: PRINT SUM
Step 5: END
```

# Decision

- Decision statements are used when the outcome of the process depends on some condition.
- For example, if x=y, then print “EQUAL”.
- *If condition then process*
- *Eg:An algorithm to check the equality of two numbers*

```
Step 1: Input the first number as A
Step 2: Input the second number as B
Step 3: IF A = B
            Then PRINT "EQUAL"
        ELSE
             PRINT "NOT EQUAL"
Step 4: END
```

# Repetitions

- Repetition, which involves executing one or more steps for a number of times, can be implemented using constructs such as while, do-while, and for loops.
- These loops execute one or more steps until some condition is true.
- Eg:An algorithm that prints the first 10 natural numbers.

```
Step 1: [INITIALIZE] SET I = 0, N = 10
Step 2: Repeat Step while I<=N
Step 3:   PRINT I
Step 4: SET I = I + 1
Step 5: END
```

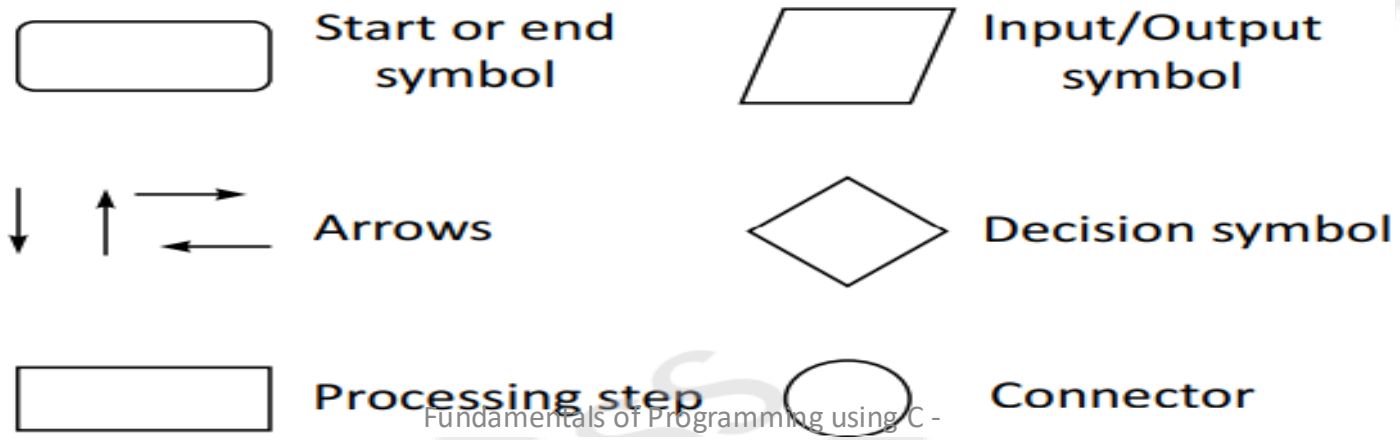
# Selecting the Most Efficient Algorithm

- Many a times, you may formulate more than one algorithm for a problem. In such cases, you must always analyse all the alternatives and try to choose the most efficient algorithm.
- This can be done by time and space complexity of the algorithms designed.
- The **time complexity** of an algorithm is basically the running time of a program as a function of the input size.
- Similarly, the **space complexity** of an algorithm is the amount of computer memory that is required during the program execution as a function of the input size.

- In other words, the number of machine instructions which a program executes is called its time complexity.
- This number is primarily dependent on the size of the program's input and the algorithm used.
- Generally, the space needed by a program depends on the following two parts:
- **Fixed part:** It varies from problem to problem. It includes the space needed for storing instructions, constants, variables, and structured variables
- **Variable part:** It varies from program to program. It includes the space needed for recursion stack, and for structured variables that are allocated space dynamically during the runtime of a program.

# Flowcharts

- A flowchart is a graphical or symbolic representation of a process.
- It is basically used to design and document virtually complex processes to help the viewers to visualize the logic of the process, so that they can gain a better understanding of the process and find flaws bottlenecks, and other less obvious features within it.
- When designing a flowchart, each step in the process is depicted by a different symbol and is associated with a short description.
- The symbols in the flowchart are linked together with arrows to show the flow of logic in the process.

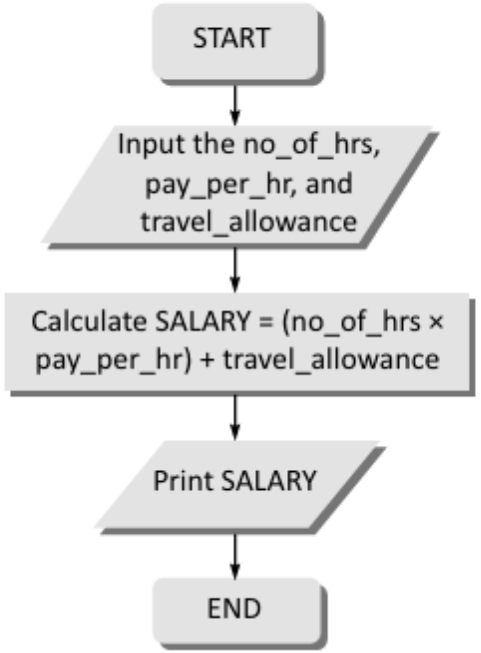


- The symbols of a flowchart include:
- **Start and end symbols** are also known as the terminal symbols and are represented as circles, ovals, or rounded rectangles.
- **Terminal symbols** are always the first and the last symbols in a flowchart.
- **Arrows** depict the flow of control of the program. They illustrate the exact sequence in which the instructions are executed.
- **Generic processing step,** also called as an activity, is represented using a rectangle. Activities include instructions such as add a to b, save the result.
- Therefore, a processing symbol represents arithmetic and data movement instructions.
- When more than one process has to be executed simultaneously, they can be placed in the same processing box, but execution will be carried out in the order of their appearance.

- **Input/output symbols** are represented using a parallelogram and are used to get inputs from the users or display the results to them. •
- **A conditional or decision symbol** is represented using a **diamond**. It is basically used to depict a Yes/No question or a True/False test. The two arrows coming out of it, one from the bottom vertex and the other from the right vertex, correspond to Yes or True, and No or False, respectively.
- **Labelled connectors** are represented by an identifying label inside a circle and are used in complex or multisheet diagrams to substitute for arrows.
- For each label, the ‘outflow’ connector must have one or more ‘inflow’ connectors. A pair of identically labelled connectors issued to indicate a continued flow

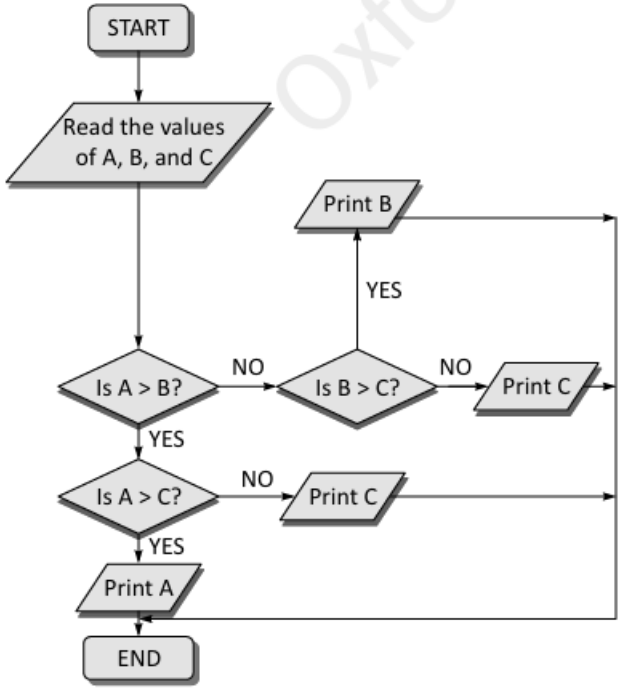
Draw a flowchart to calculate the salary of a daily wager.

*Solution*




Draw a flowchart to determine the largest of three numbers.

*Solution*



# Pseudocode

- **Pseudocode** is a compact and informal high-level description of an algorithm that uses the structural conventions of a programming language.
- Pseudocodes are an outline of a program that can be easily converted into programming statements.
- They consist of short English phrases that explain specific tasks within a program's algorithm. They should not include keywords in any specific computer language.
- The sole purpose of pseudocodes is to enhance human understandability of the solution.
- They are commonly used in textbooks and scientific publications for documenting algorithms, and for sketching out the program structure before the actual coding is done
- There are no standards defined for writing a pseudocode, because a pseudocode is not an executable program.
- Flowcharts can be considered as graphical alternatives to pseudocodes, but require more space on paper.

Write a pseudocode for calculating the price of a product after adding sales tax to its original price.

*Solution*

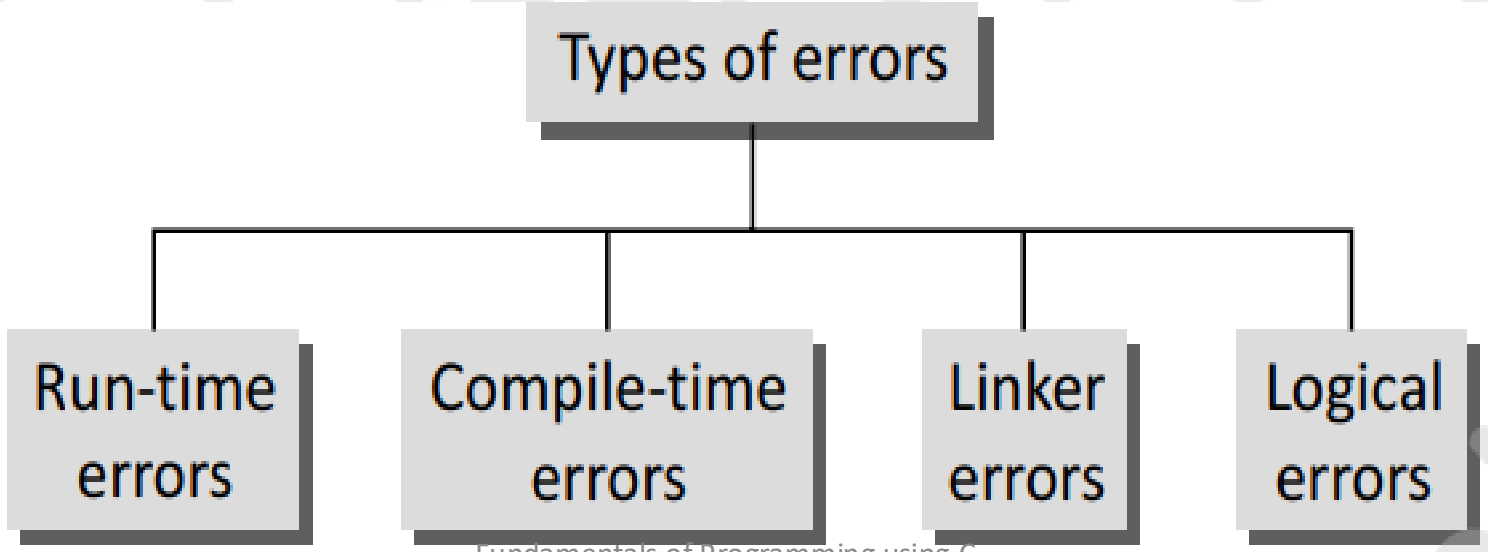
```
1. Read the price of the product
2. Read the sales tax rate
3. Calculate sales tax = price of the item x
   sales tax rate
4. Calculate total price = price of the product
   + sales tax
5. Print total price
6. End
Variables: price of the product, sales tax
   rate, sales tax, total price
```

# Types of Errors

- While writing programs, very often we get errors in our program.

  These errors if not removed will either give erroneous output or will not let the compiler to compile the program.

- These errors are broadly classified under four groups:



- **Run-time Errors**
- As the name suggests, run-time errors occur when the program is being run executed. Such errors occur when the program performs some illegal operations like
    - Dividing a number by zero
    - Opening a file that already exists
    - Lack of free memory space
    - Finding square or logarithm of negative numbers

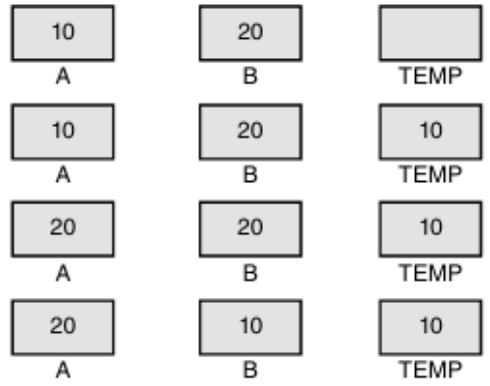
- **Compile-time Errors**
- Again as the name implies, compile errors occur at the time of compilation of the program.
- Such errors can be further classified as follows:

1. **Syntax Errors**:Syntax error is generated when rules of C programming language are violated.
2. **Semantic Errors**: Semantic errors are those errors which may comply with rules of the programming language but are not meaningful to the compiler
3. **Logical Errors**: Logical errors are errors in the program code that result in unexpected and undesirable output which is obviously not correct.
4. **Linker Errors**: These errors occur when the linker is not able to find the function definition for a given prototype.

# Examples of Program Design Tools

## 1. Swap Two Variables

Soln: To swap two variables we will use a temporary variable. First, we copy the value of any one variable in the temporary variable. Then, we copy the value of second variable in first. Finally, the value of the temporary variable is copied in the second variable.

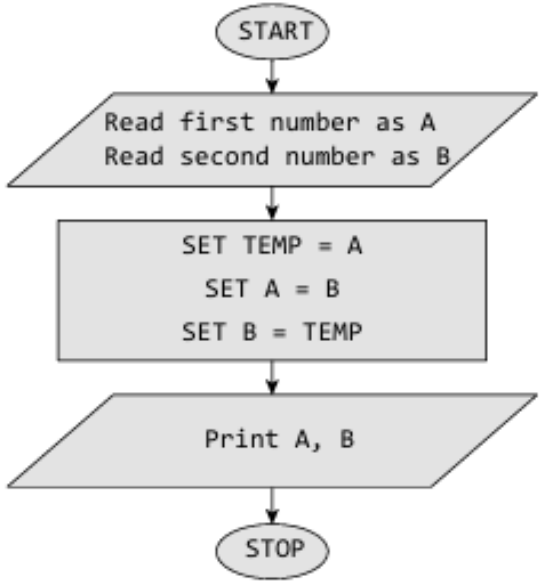



Steps involved in swapping two variables

**Algorithm 1**

```
Step 1: Start
Step 2: Read the first number as A
Step 3: Read the second number as B
Step 4: SET TEMP = A
Step 5: SET A = B
Step 6: SET B = TEMP
Step 7: PRINT A, B
Step 8: End
```

**Flowchart 1**

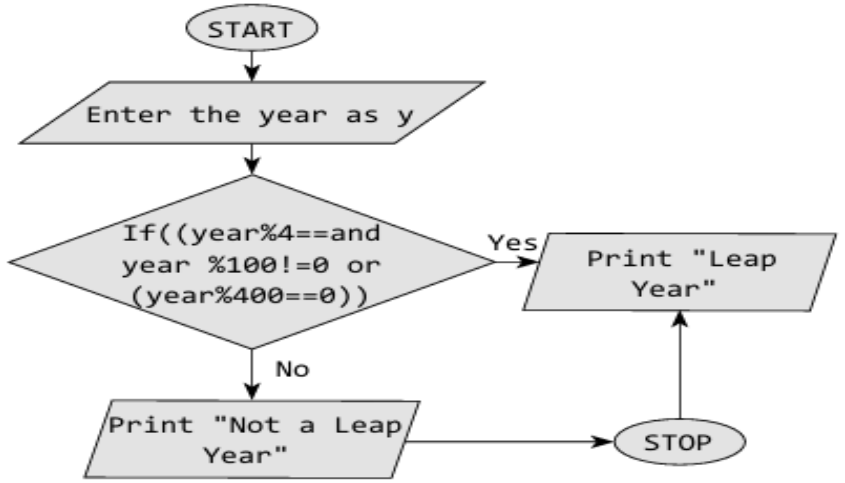


# Test for Leap Year

**Algorithm 3**

```
Step 1: Start
Step 2: Enter a year as year
Step 3: Check IF((year%4==0 and year %100!=0)
or (year%400 == 0)),
           Then PRINT "Leap Year"
         ELSE
            PRINT "Not a Leap Year"
Step 4: End
```

**Flowchart 3**

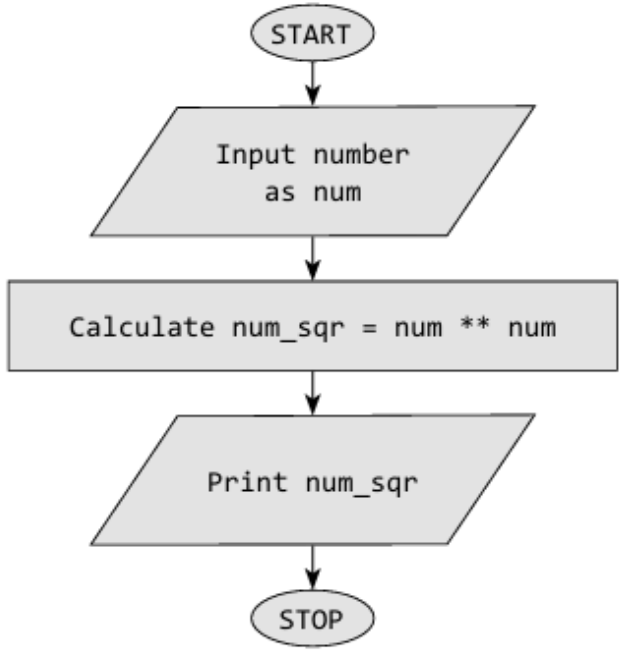


# Square of a number

**Algorithm 4**

```
Step 1: Start
Step 2: Input the number as num
Step 3: Calculate square as num ** num
Step 4: Print square as calculated in Step 3
Step 5: End
```

**Flowchart 4**



# GCD of two numbers

**Algorithm 5**

```
Step 1: Start
Step 2: Enter the two numbers as num1 and num2
Step 3: Set larger of the two numbers as
        dividend
Step 4: Set smaller of the two numbers as
        divisor
Step 5: Repeat Steps 6-8 while divisor != 0
Step 6: Set remainder = dividend % divisor
Step 7: Set dividend = divisor
Step 8: Set divisor = remainder
Step 9: Print dividend
Step 10: End
```

**Flowchart 5**