

COMPUTATIONAL THINKING & PROGRAMMING

23ESCC111

Contents

- **Introduction to Problem solving**
- **Computational Thinking**
- **Key techniques of computational thinking**
- **Problem analysis**
- **Logic building for solutions**
- **Algorithm, Pseudo Code, Flow Chart**
- **Debugging & Testing the codes.**
- **Introduction to digital computers**
- **Overview of operating systems**
- **Assemblers, Compilers, Interpreters**
- **Programming languages.**

What is Computational Thinking?

Computational thinking is a problem-solving approach that involves thinking in a way that draws upon concepts and techniques from computer science.

It is a mindset or set of skills that enables individuals to approach complex problems and tasks systematically, breaking them down into smaller, more manageable parts.

What is Computational Thinking?

(Contd....)

Computational thinking is not limited to computer-related problems; it can be applied to a wide range of domains and everyday life.

**Examples which demonstrate how
computational techniques can be used
in various fields**

Examples on CT(Contd....)

Route Optimization: GPS navigation systems use computational thinking to find the best route from one location to another. They decompose the problem into smaller steps, such as identifying roads, calculating distances, and factoring in traffic conditions.

Language Translation: Machine translation systems like Google Translate employ computational thinking to translate text from one language to another. They use algorithms to recognize patterns in language and generate corresponding translations

Examples on CT(Contd....)

Language Translation: Machine translation systems like Google Translate employ computational thinking to translate text from one language to another. They use algorithms to recognize patterns in language and generate corresponding translations.

Weather Forecasting: Meteorologists use computational thinking to process vast amounts of weather data, simulate weather patterns, and make predictions.

Examples on CT(Contd....)

Robotics: In robotics, computational thinking is used to program robots to perform tasks. Algorithms help robots navigate their environments, make decisions, and manipulate objects.

Natural Language Processing (NLP): NLP applications like chatbots and sentiment analysis rely on computational thinking to process and understand human language. Algorithms break down language into structured data for analysis.

Key Techniques of CT

Computational thinking involves several key techniques that help individuals approach and solve complex problems in a systematic and efficient manner.

Key Techniques of CT (Contd..)

Decomposition: Break down a complex problem or task into smaller, more manageable sub problems. This technique simplifies the problem, making it easier to understand and solve. Decomposition helps identify the main components and dependencies within the problem.

Pattern Recognition: Recognize patterns, regularities, or similarities within data or a problem. This technique involves identifying recurring elements, relationships, or trends. Pattern recognition can lead to insights and more efficient problem-solving approaches.

Key Techniques of CT (Contd..)

Abstraction: Focus on the essential details and hide unnecessary complexities. Abstraction simplifies complex problems by creating a high-level representation that captures the most important aspects. It helps in understanding the problem's structure and finding solutions.

Algorithm Design: Create step-by-step plans or algorithms to solve a problem. Algorithms provide a clear and systematic approach to accomplishing tasks or addressing challenges. Effective algorithm design is crucial for achieving efficient and reliable solutions.

Key Techniques of CT (Contd..)

Generalization: Apply solutions or strategies from one problem to similar problems. Generalization allows you to leverage previous knowledge and experiences to solve new and related challenges. It promotes efficiency and consistency in problem-solving.

Evaluation: Assess the quality and effectiveness of a solution. Evaluation involves critical thinking and the refinement of solutions to make them more efficient, accurate, and reliable. It helps in determining whether a solution meets the desired goals and criteria.

Key Techniques of CT (Contd..)

Logical Reasoning: Apply logic and reasoning to analyze problems and make informed decisions. Logical thinking helps in determining the validity of arguments, identifying cause-and-effect relationships, and drawing conclusions based on evidence.

Optimization: Find the most efficient or optimal solution within a set of constraints. Optimization techniques aim to maximize or minimize certain objectives while considering limitations or requirements. It is often used in resource allocation and decision-making.

Key Techniques of CT (Contd..)

Simulation: Create models or simulations of real-world systems to understand their behavior and make predictions. Simulation techniques help in testing hypotheses and scenarios without affecting the actual system.

Debugging: Identify and correct errors or issues in code or solutions. Debugging involves systematic problem-solving to locate and fix problems in software or algorithms.

What is problem analysis?

- ✓ Problem analysis is a systematic process of examining a problem or a challenge to gain a deeper understanding of its nature, causes, and potential solutions.
- ✓ It is an essential step in problem-solving and decision-making in various domains, including business, engineering, science, and everyday life.

What is problem analysis?

- ✓ The goal of problem analysis is to break down a complex problem into its constituent parts, identify the underlying causes or factors contributing to the problem, and generate insights that can guide effective solutions.

Examples that illustrate how problem analysis can be applied in different contexts:

Examples on Problem Analysis

Business Operations:

Problem: A manufacturing company is experiencing a significant increase in product defects.

Analysis: Conduct a root cause analysis to identify factors contributing to the defects, such as machine malfunctions, operator errors, or material quality issues.

Solutions: Implement corrective actions based on the analysis, which may include machine maintenance, operator training, and supplier quality checks.

Examples on Problem Analysis

Healthcare:

Problem: A hospital is facing long patient wait times in the emergency department.

Analysis: Analyse patient flow, staffing levels, and triage processes to identify bottlenecks and delays.

Solutions: Optimize staff scheduling, streamline triage procedures(prioritizing), and consider expanding capacity to reduce wait times.

Examples on Problem Analysis

Environmental Conservation:

Problem: A city is experiencing increased pollution levels and environmental degradation.

Analysis: Investigate the sources of pollution, including industrial emissions, transportation, and waste management practices.

Solutions: Develop and implement policies and initiatives to reduce pollution, such as stricter emissions standards, public transportation improvements, and recycling programs.

Examples on Problem Analysis

Public Policy:

Problem: A city experiences traffic congestion and jam during rush hours.

Analysis: Examine traffic flow patterns, road infrastructure, and public transportation options.

Solutions: Implement traffic management strategies, invest in public transportation expansion, and consider congestion pricing policies.

Logic Building for solutions

- Logic building for solutions involves developing a systematic and structured approach to solving problems and making decisions.
- It is a critical skill in problem-solving and is applicable across various domains.

Steps and strategies to build logical solutions

Steps and strategies to build logical solutions

Understand the Problem:

- Clearly define the problem and its scope(boundaries).
- Gather all relevant information and data related to the problem.
- Identify any constraints or limitations that must be considered.

Steps and strategies to build logical solutions(Contd..)

Decompose the Problem:

- Break the problem down into smaller, more manageable sub problems or components.
- Identify the relationships and dependencies between these components.

Steps and strategies to build logical solutions(Contd..)

Analyse the Causes:

- Conduct a root cause analysis to determine the underlying factors contributing to the problem.
- Investigate the sequence of events that lead to the problem's occurrence.

Steps and strategies to build logical solutions(Contd..)

Identify Goals and Objectives:

- Clearly state the desired outcomes and goals of solving the problem.
- Define specific criteria for success and what constitutes a solution.

Steps and strategies to build logical solutions(Contd..)

Generate Ideas and Solutions:

- Brainstorm potential solutions without judgment.
- Prioritize solutions based on feasibility, effectiveness, and potential impact.

Steps and strategies to build logical solutions(Contd..)

Evaluate Alternatives:

- Assess the advantages and disadvantages of each proposed solution.
- Consider potential risks, costs, and benefits associated with each option.

Steps and strategies to build logical solutions(Contd..)

Select the Best Solution:

- Choose the solution that aligns best with the problem's goals and objectives.
- Ensure that the selected solution is practical, feasible, and has a high likelihood of success.

Steps and strategies to build logical solutions(Contd..)

Plan Implementation:

- Develop a detailed plan for implementing the chosen solution.
- Specify the steps, resources, and timeline required for execution.
- Assign responsibilities to individuals or teams involved in the implementation.

Steps and strategies to build logical solutions(Contd..)

Execute and Monitor:

- Implement the solution according to the plan.
- Continuously monitor progress and gather data to assess the solution's effectiveness.
- Be prepared to make adjustments or refinements as necessary.

Program Designing Tools

What is an Algorithm ?

An algorithm is a step-by-step, precise set of instructions or a well-defined computational procedure used to solve a specific problem, perform a particular task, or accomplish a desired outcome.

Key characteristics of algorithms

Step-by-Step Instructions: Algorithms consist of a sequence of well-defined and ordered steps that guide the execution of a task. Each step represents an operation or action that should be performed.

Unambiguous: Algorithms are unambiguous and leave no room for interpretation. Each step must be clearly defined and understood, so there is no ambiguity in how to execute them.

Algorithms characteristics(Contd..)

Termination: An algorithm should eventually reach a conclusion or produce an output, indicating the completion of the task. It should not run indefinitely.

Input and Output: Algorithms typically take input data, process it through a series of steps, and produce an output or result based on the input and the algorithm's logic.

Algorithms characteristics(Contd..)

Deterministic: Algorithms are deterministic, meaning that for a given input, they will always produce the same output. There is no randomness or uncertainty in their behaviour.

Efficiency: Algorithms can be evaluated in terms of their efficiency, such as how quickly they produce results and how many computational resources (e.g., time and memory) they consume.

Algorithm to find the largest of 2 numbers

Step 1: Start

Step 2: Input the values of A, B Compare A and B.

Step 3: If $A > B$ then go to step 5

Step 4: Display “B is largest” go to Step 6

Step 5: Display “A is largest”

Step 6: Stop

Algorithm to add two numbers

Step 1: Start

Step 2: Read values for num1, num2.

Step 4: Add num1 and num2 to get the sum

Step 5: Display sum

Step 6: Stop

Algorithm to compute simple interest

$$\text{Simple Interest} = (p * t * r) / 100$$

Step 1: Start

Step 2: Read p, t, r

Step 3: $si \leftarrow (p * t * r) / 100$

Step 4 : Print “ si ”

Step 5: Stop

Algorithm to find area and circumference of circle

Step 1: Start

Step 2: Read r

Step 3: $\text{area} \leftarrow 3.142 * r * r$

Step 4: $\text{circum} \leftarrow 2 * 3.142 * r$

Step 5: Print “area , circumference”

Step 6: Stop

Program Designing Tools

Pseudocode

It is a detailed readable description of what a computer program must do, expressed in a formally-styled natural language rather than in a programming language

Characteristics of Pseudocode

1. Informal Language: Pseudocode is written in plain, human-readable language rather than a specific programming language. This makes it accessible to both programmers and non-programmers.

2. Readability: Pseudocode should be easy to read and understand. It often uses structured English phrases, simple mathematical notations, and indentation to represent program flow and structure.

3. No Fixed Syntax: Pseudocode has no rigid syntax rules or conventions like programming languages do. Instead, it relies on the author's discretion to create a clear and coherent representation of the algorithm.

Characteristics of Pseudocode

4. Modularity: Pseudocode encourages breaking down complex problems into smaller, manageable sub-problems or functions. These sub-problems can be described separately and then integrated into the overall algorithm.

5. No Specific Language Constraints: Pseudocode is not tied to any particular programming language, which allows programmers to express algorithms in a language-agnostic way.

Pseudocode to find Area of a circle and Perimeter of a circle

// Area of a circle

BEGIN

NUMBER r, area

INPUT r

area=3.14*r*r

OUTPUT area

END

// Perimeter of a circle

BEGIN

NUMBER r, perimeter

INPUT r

perimeter=3.14*2*r

OUTPUT perimeter

END

Pseudocode to calculate Simple Interest

// calculate a simple interest

- 1. Start**
- 2. Input the principal amount (P)**
- 3. Input the annual interest rate (R)**
- 4. Input the time period in years (T)**
- 5. Calculate the simple interest (SI) using the formula:**
$$SI = (P * R * T) / 100$$
- 6. Display the simple interest (SI)**
- 7. End**

Pseudocode to add two numbers

// add two numbers

- 1. Start**
- 2. Input the first number (num1)**
- 3. Input the second number (num2)**
- 4. Add num1 and num2 and store the result in a variable (sum)**
- 5. Display the sum**
- 6. End**

Pseudocode to find largest of two numbers

// largest of two numbers

1. Start

2. Input the first number (num1)

3. Input the second number (num2)

4. If num1 is greater than num2, then:

**5. Display num1 as the largest number Else if num2 is greater than num1,
then:**

6. Display num2 as the largest number

Else:

7. Display "Both numbers are equal."

8. End

Comparing Algorithm and Pseudocode

Algorithm:

- An algorithm is a step-by-step set of instructions for solving a specific problem or performing a task.
- It is a high-level description of the solution, independent of any specific programming language.
- Algorithms are usually presented in a natural language or a more formal mathematical notation.

Pseudocode in C:

- Pseudocode is a way to represent an algorithm using a mix of natural language and some programming-like constructs.
- Pseudocode is less formal than actual code and is used to outline the structure and logic of a program without getting into the specifics of syntax.
- Pseudocode often uses C-like syntax for readability and to make it easier to translate into actual code later.

Comparing Algorithm and Pseudocode

Algorithm:

- They focus on the logic and overall strategy for solving a problem.
- Algorithms can be implemented in various programming languages

Pseudocode in C:

- It is a helpful tool for planning and designing a program before you start writing actual code.
- Pseudocode is not a strict programming language; it's a bridge between a high-level algorithm and low-level code.

Algorithm and Pseudocode to find largest of 3 numbers

Algorithm:

Step 1. Start

Step 2. Read three numbers A, B, and C

Step 3. If A is greater than B and A is greater than C, then

Step 4. Display A as the largest number

Step 5. otherwise if B is greater than A and B is greater than C, then

Step 6. Display B as the largest number

Step 7. otherwise Display C as the largest number

Step 8. stop

Pseudocode:

1. Begin

2. Declare variables A, B, C, and largest as integers

3. Prompt the user to enter three numbers and store them in A, B, and C

4. Set largest to A

5. If B is greater than largest, then Set largest to B

6. If C is greater than largest, then Set largest to C

7. Print largest as the largest number

8. End

Algorithm and Pseudocode to find voting eligibility

Algorithm:

- Step 1. Start
- Step 2. Read the age of the person
- Step 3. If the age is greater than or equal to 18, then
- Step 4: Display "You are eligible to vote."
- Step 5. otherwise
Display "You are not eligible to vote."
- Step 6. Stop

Pseudocode:

- 1. Begin
- 2. Declare a variable age as an integer
- 3. Prompt the user to enter their age and store it in age
- 4. If age is greater than or equal to 18, then
- 5. Print "You are eligible to vote."
- 6. Else Print "You are not eligible to vote."
- 7. End

Algorithm and Pseudocode to Convert a temperature from Celsius to Fahrenheit:

Algorithm:

- Step 1. Start
- Step 2. Read the temperature in Celsius (C)
- Step 3. Calculate the temperature in Fahrenheit (F) using the formula: $F = (C \times 9/5) + 32$
- Step 4. Display the temperature in Fahrenheit (F)
- Step 5. Stop

Pseudocode:

- 1. Begin
- 2. Declare variables C and F as real numbers
- 3. Prompt the user to enter the temperature in Celsius and store it.
- 4. Set F to $(C * 9/5) + 32$
- 5. Print "The temperature in Fahrenheit is:", F
- 6. End

Algorithm and Pseudocode to Swap Two Numbers using temporary variable

Algorithm:

Step 1: Start

Step 2: Initialize two variables, a and b, with the numbers you want to swap.

Step 3: Create a temporary variable, temp, to hold one of the numbers.

Step 4: Store the value of a in temp.

Step 5: Assign the value of b to a.

Step 6: Assign the value of temp (which is the original value of a) to b.

Step 7: Now, a and b have been swapped.

Step 8: Stop

Pseudocode:

1. Begin

2. Input the values of 'a' and 'b'.

3. Print the original values of 'a' and 'b'.

4. Set temp=a

5. Set a=b

6. Set b=temp.

7. Print the swapped values of 'a' and 'b'.

8. End

Algorithm and Pseudocode to Swap Two Numbers without using temporary variable

Algorithm:

Step 1: Start

Step 2: Initialize two variables, num1 and num2, with the numbers you want to swap.

Step 3:

Step 4: add num1 and num2 store it in num1.

Step 5: sub num2 from num1 store it in num2.

Step 6: sub num2 from num1 store it in num1.

Step 7: Now, num1 and num2 have been swapped.

Step 8: Stop

Pseudocode:

procedure swap (num1, num2)

Begin

Input num1, num2

$\text{num1} \leftarrow \text{num1} + \text{num2}$ // num1 holds the sum of both

$\text{num2} \leftarrow \text{num1} - \text{num2}$ // num2 now holds the value of num1

$\text{num1} \leftarrow \text{num1} - \text{num2}$ // num1 now holds value of num2

Output num1, num2

End

Algorithm and Pseudocode to convert Kilometer to meter

Algorithm:

Step 1: Start

Step 2: read kilometer

Step 3: $\text{meter} = \text{kilometer} * 1000$

Step 4: print meter

Step 5: Stop

Pseudocode:

procedure kilometre to meter

Begin

Input kilometre

$\text{meters} = \text{kilometre} * 1000$

Output meters

End

Algorithm to find Sum of First n Numbers

Algorithm:

Input : input number n

Step 1: Start

Step 2: Read number n

Step 3: assign sum to 0 and i to 1

Step 4: Repeat steps 5 to 7 until $i \leq n$

Step 5: update sum as $\text{sum} = \text{sum} + i$

Step 6: increment i by 1

Step 7: Print sum

Step 8: Stop

Output: sum

Input : input number n \rightarrow 5

$n=5$

$\text{Sum}=0, i=1$

$\text{sum}=\text{sum} + i$

$1=0+1$ increment i by 1(till $i=n$)

$3=1+2$ increment i by 1(till $i=n$)

$6=3+3$ increment i by 1(till $i=n$)

$10=6+4$ increment i by 1(till $i=n$)

$15=10+5$ increment i by 1(till $i=n$)

Output: $\text{sum} = 15$

Algorithm to find factorial of given number

Algorithm:

Step 1 -> START

Step 2 → Take integer variable A

Step 3 → Assign value to the variable

Step 4 → From value A upto 1 multiply each digit and store

Step 5 → the final stored value is factorial of A

Step 6 -> STOP

Pseudocode:

1. Begin
2. Declare N and F as integer variable.
3. assign $F=1$.
4. Enter the value of N.
5. Check whether $N>0$, if not then $F=1$.
6. If yes then, $F=F * N$
7. Decrease the value of N by 1 .
8. Repeat step 6 and 7 until $N=0$.
9. print the value of F.
10. End

Program Designing Tools

Flowchart

Flowchart in C is a **diagrammatic representation of a sequence of logical steps of a program.**

Flowcharts use standardized symbols and conventions to represent different elements of a process, making it easier for people to understand and communicate complex procedures or algorithms.




Characteristics of Flowchart

- **Visualization:** They provide a visual representation of a process, making it easier to understand and analyze.
- **Documentation:** Flowcharts document complex procedures, making it easier for others to follow, implement, or troubleshoot.
- **Communication:** They facilitate communication between team members, stakeholders, or collaborators, as they offer a common visual language for discussing processes.

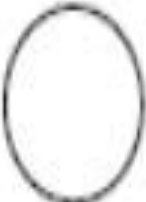


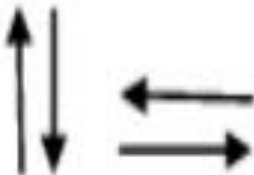
Characteristics of Flowchart

- **Analysis:** Flowcharts helps to identify or errors in a process and can be used for process improvement.
- **Problem Solving:** Flowcharts are useful for breaking down complex problems into smaller, more manageable steps, aiding in problem-solving and decision-making.

Basic flowchart symbols

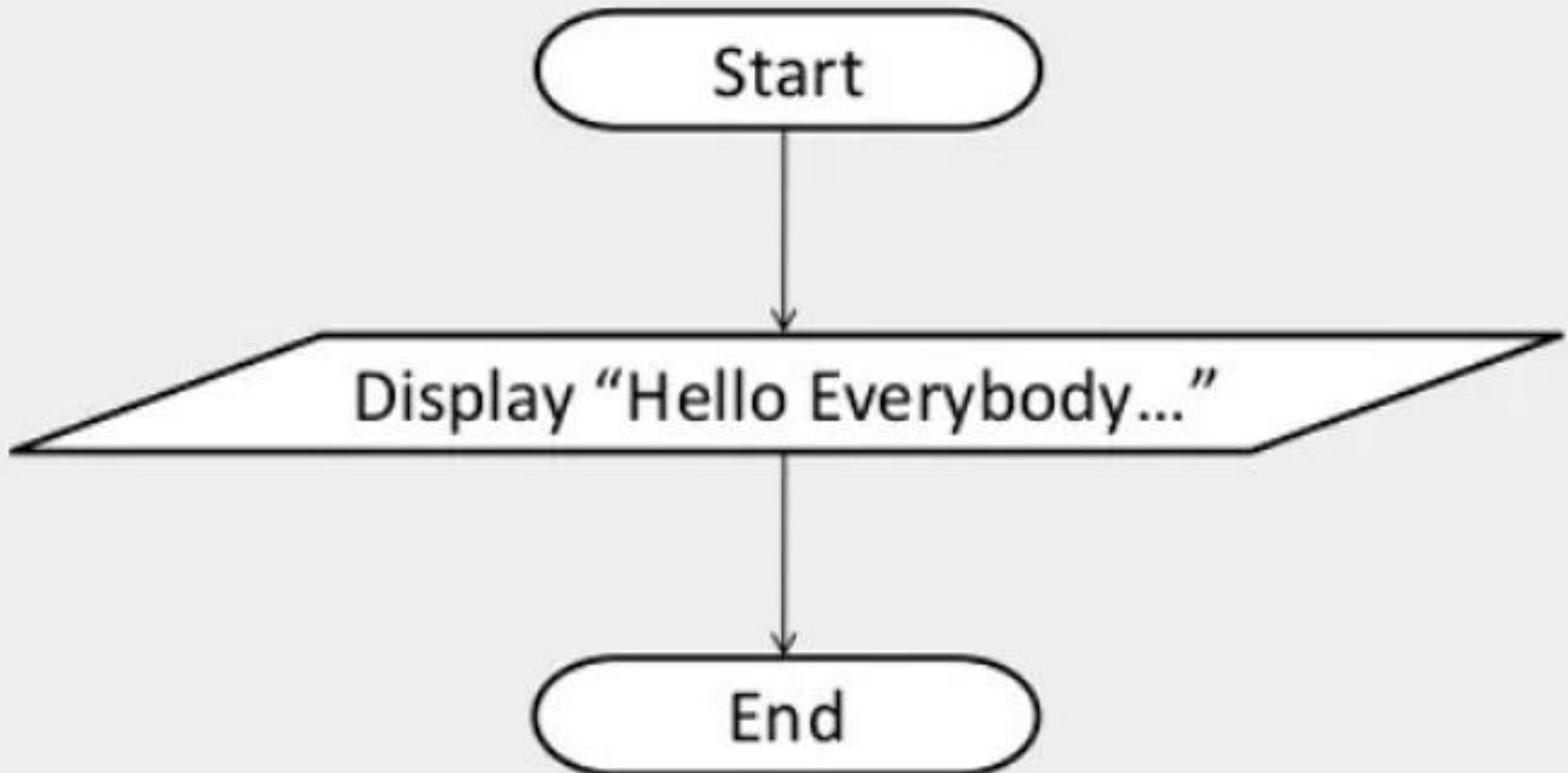
Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)

Basic flowchart symbols

	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

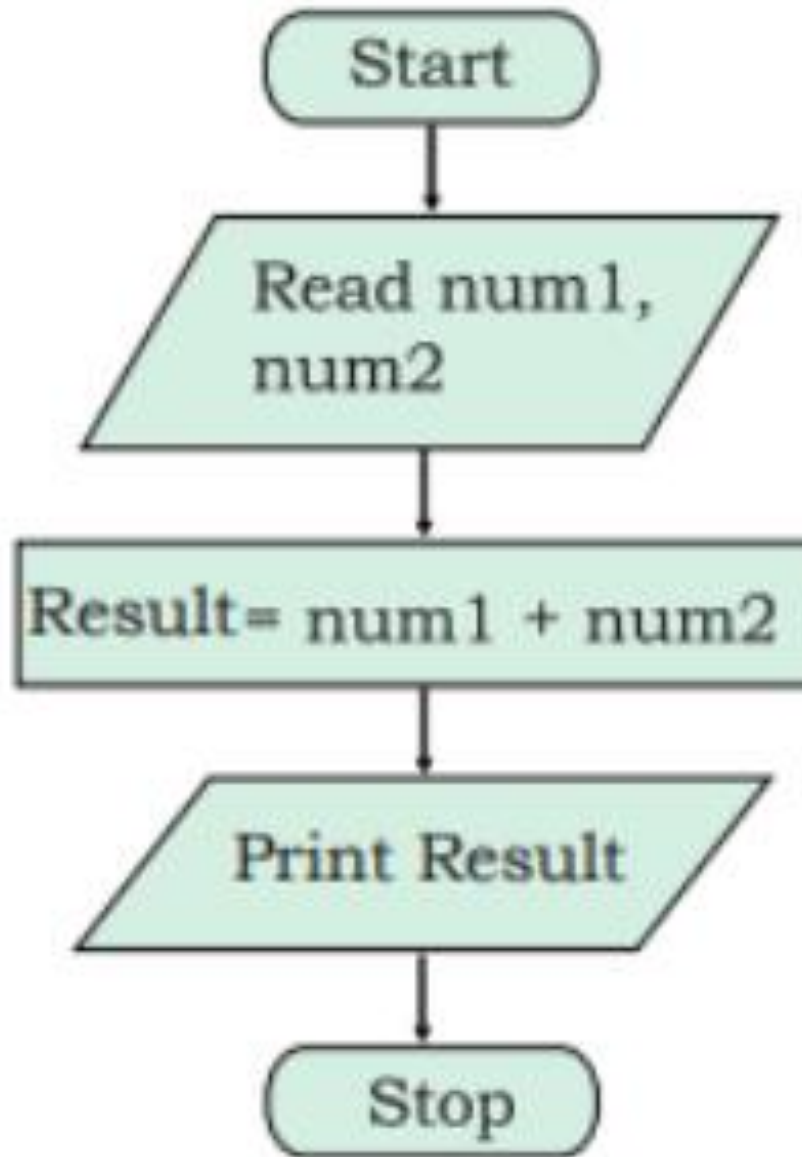
Flowchart

Flowchart to Print message "Hello Everybody"



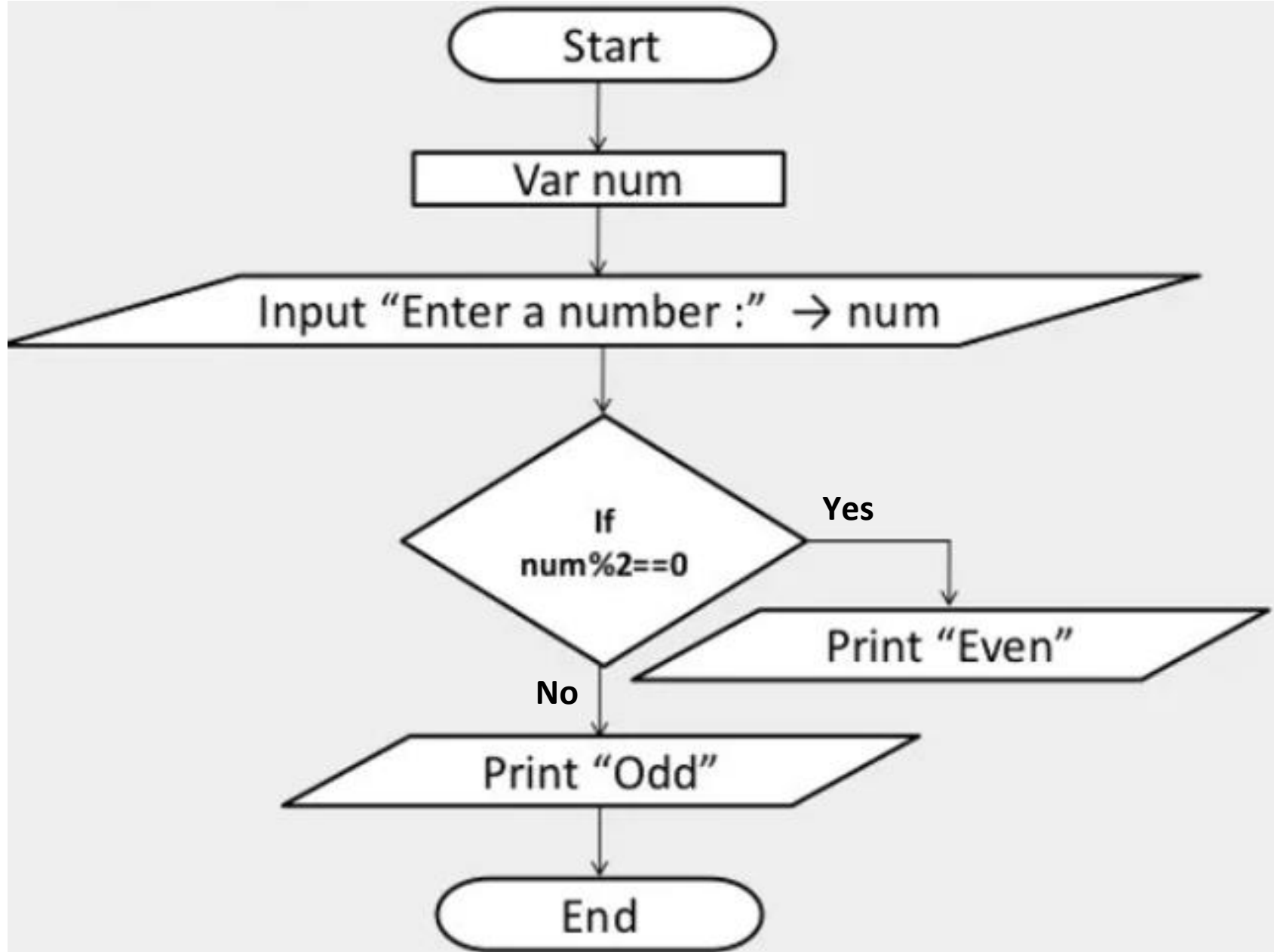
Flowchart

Flowchart to Enter two numbers and their sum to be printed



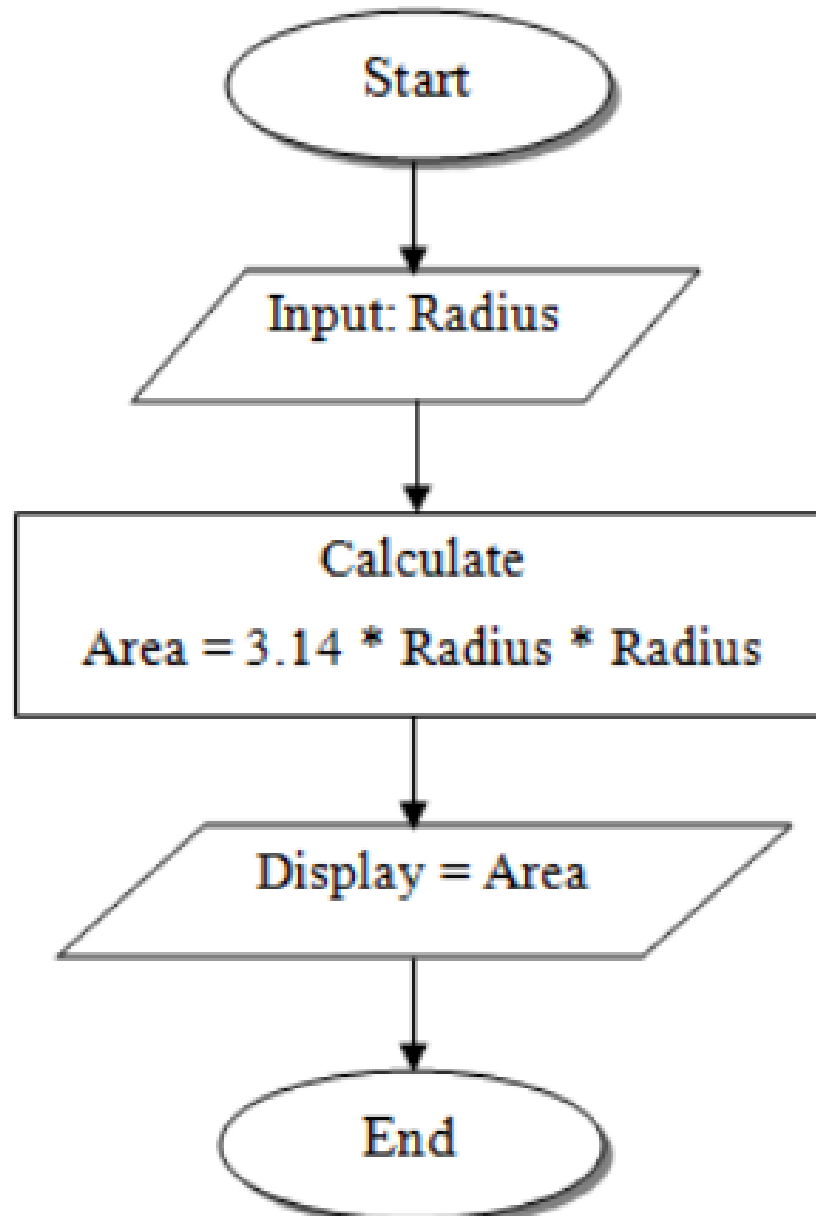
Flowchart

Flowchart to Check whether the number is even or odd



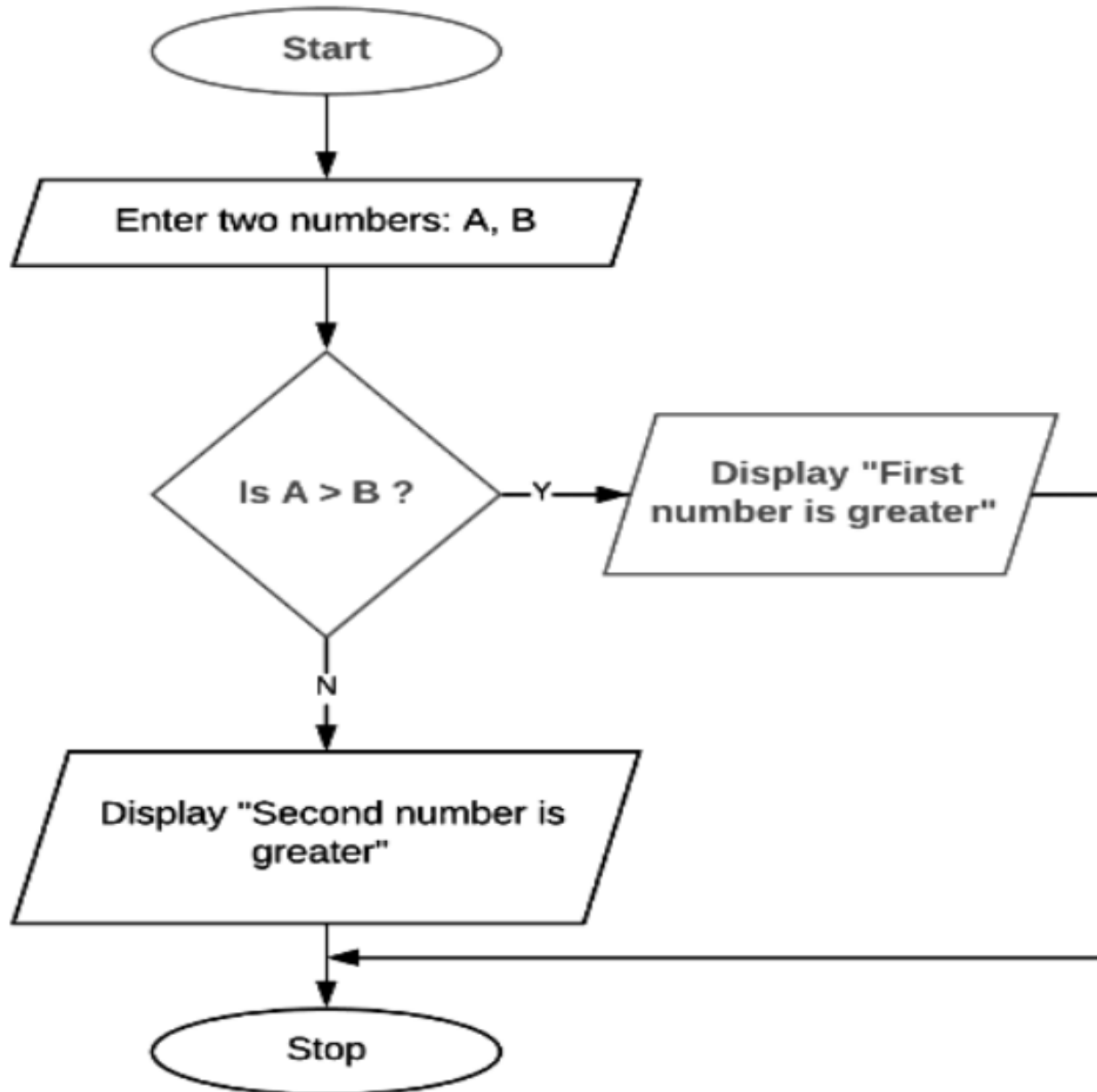
Flowchart

Flowchart to find area of circle



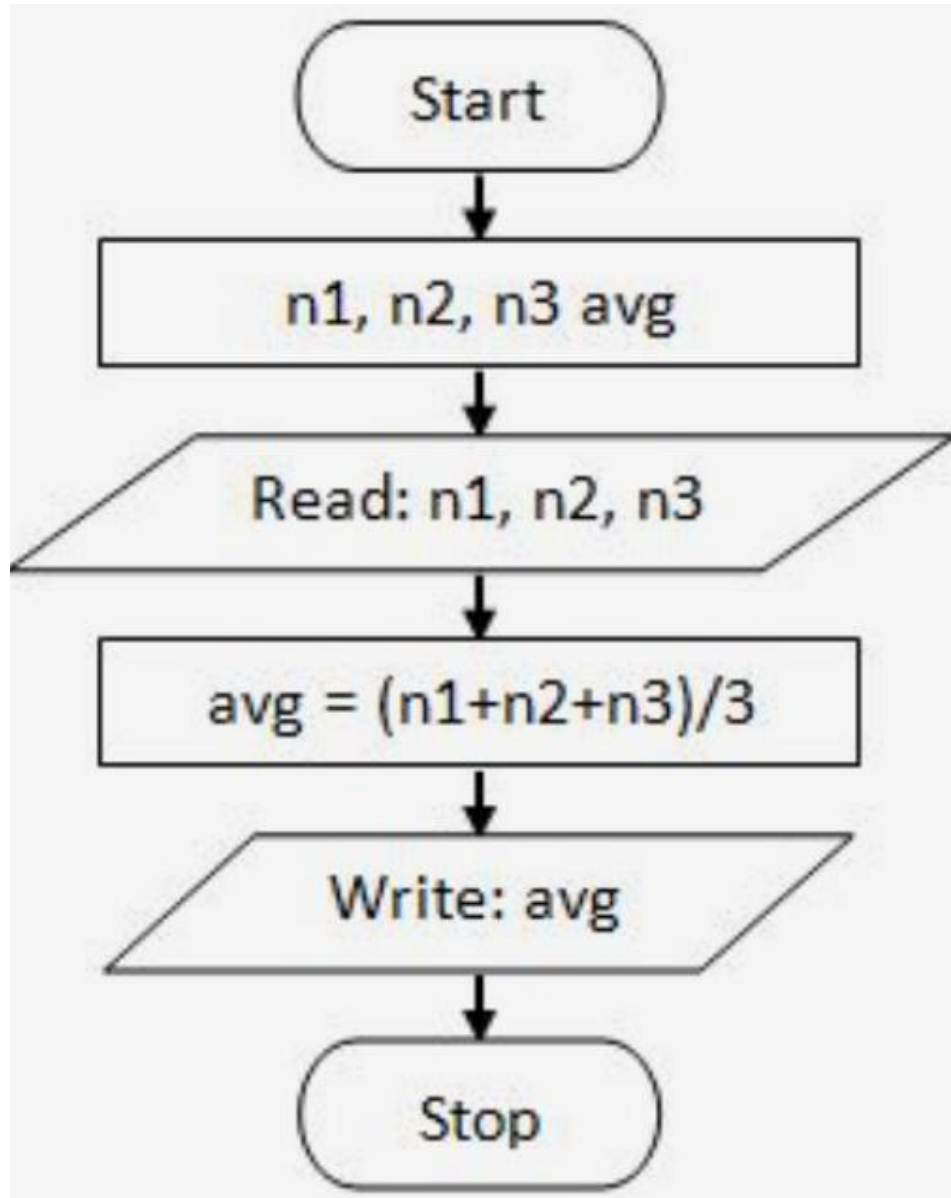
Flowchart

Flowchart to find largest of two numbers

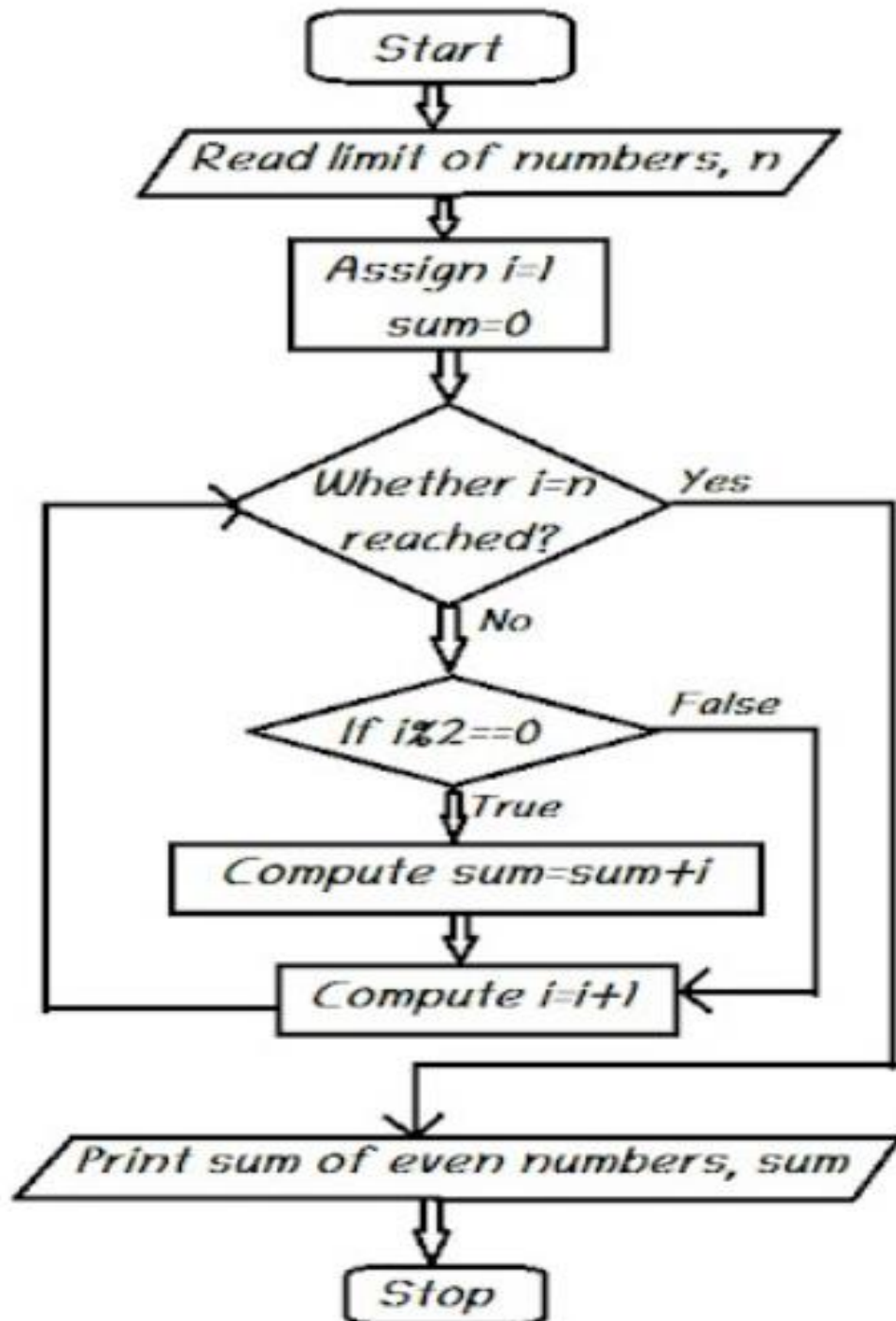


Flowchart

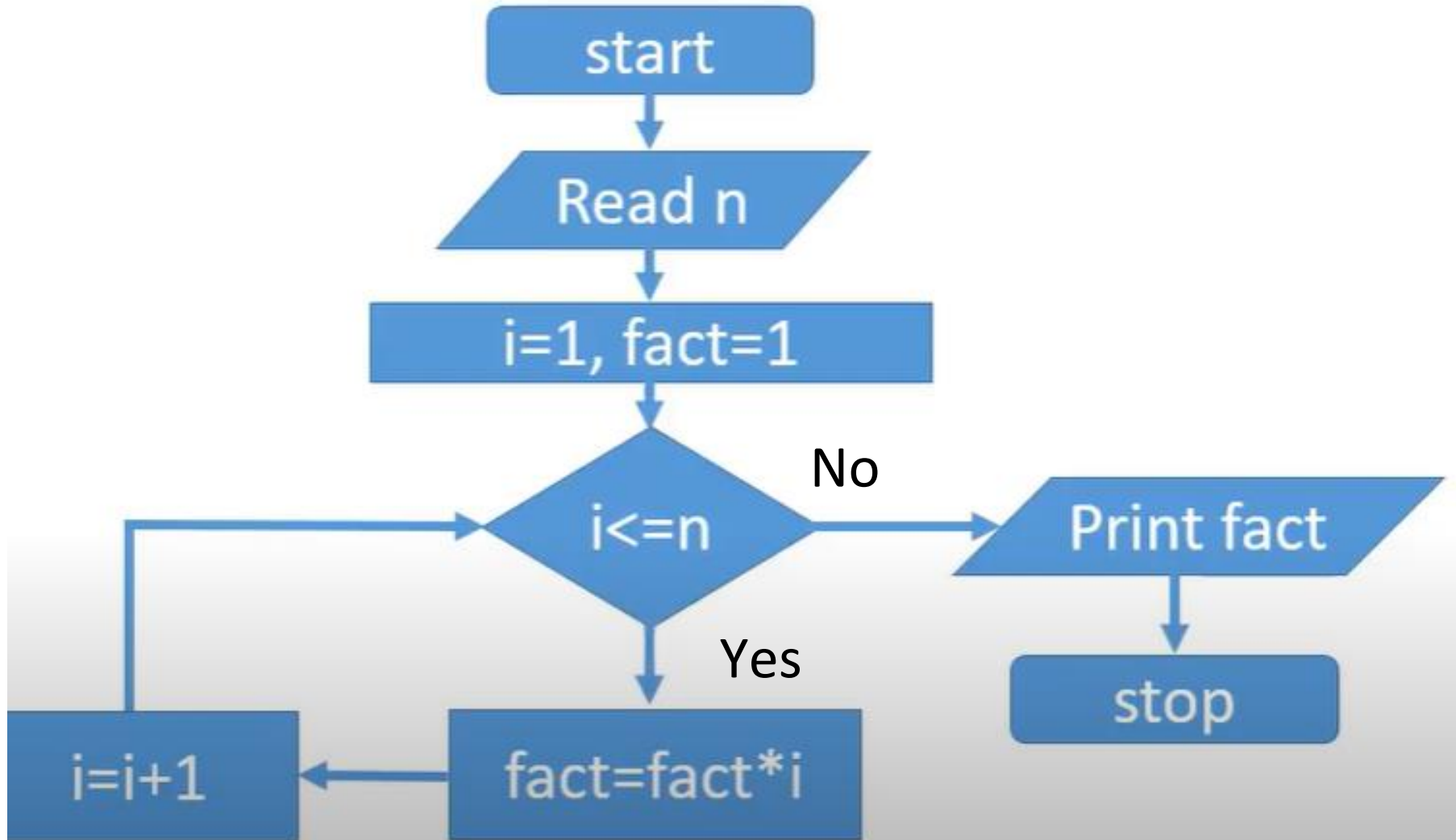
flowchart to find average of 3 numbers.



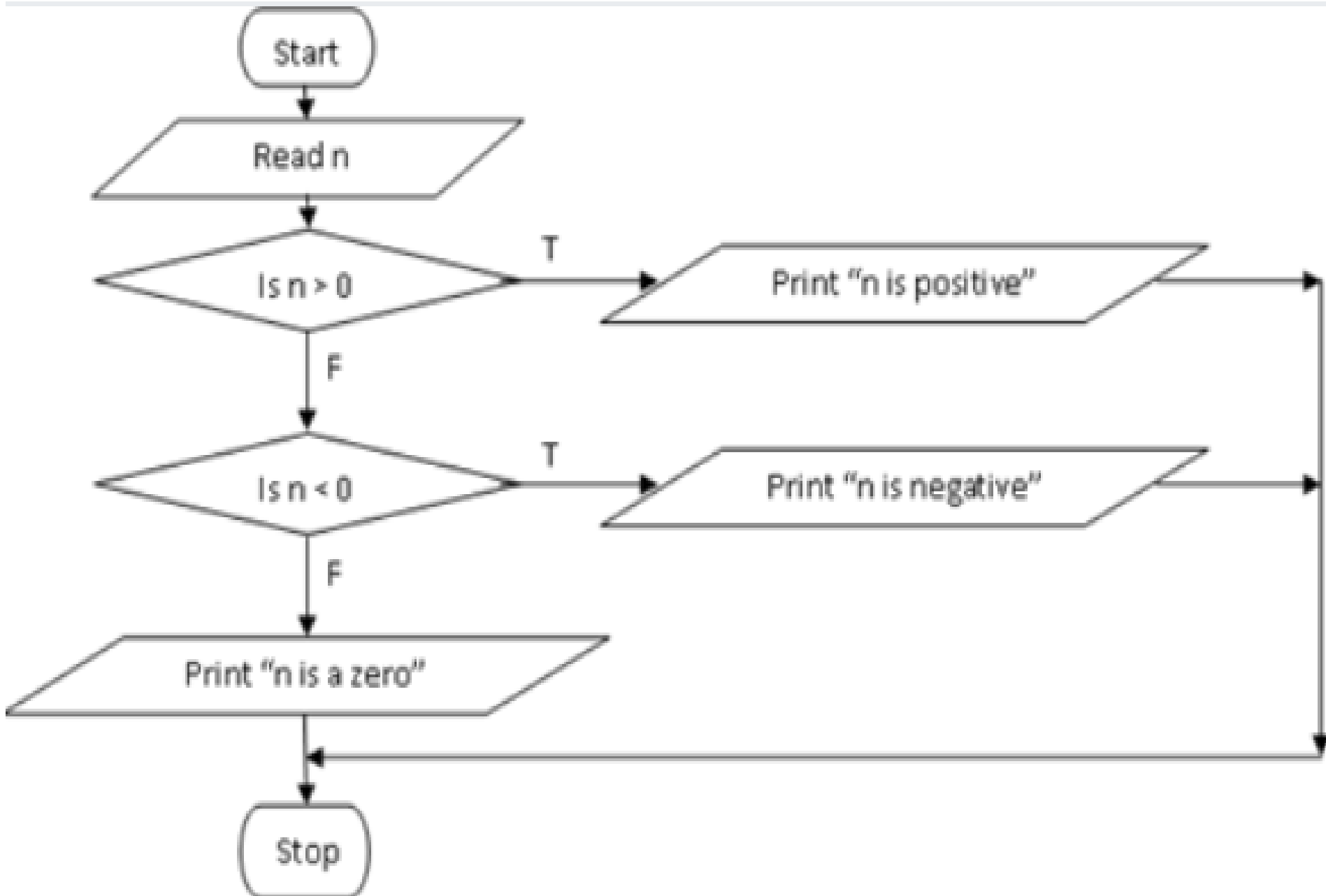
flowchart to find sum
of even numbers



flowchart to find factorial of given number



flowchart to find whether a number is positive or negative



Testing and Debugging Approaches in C

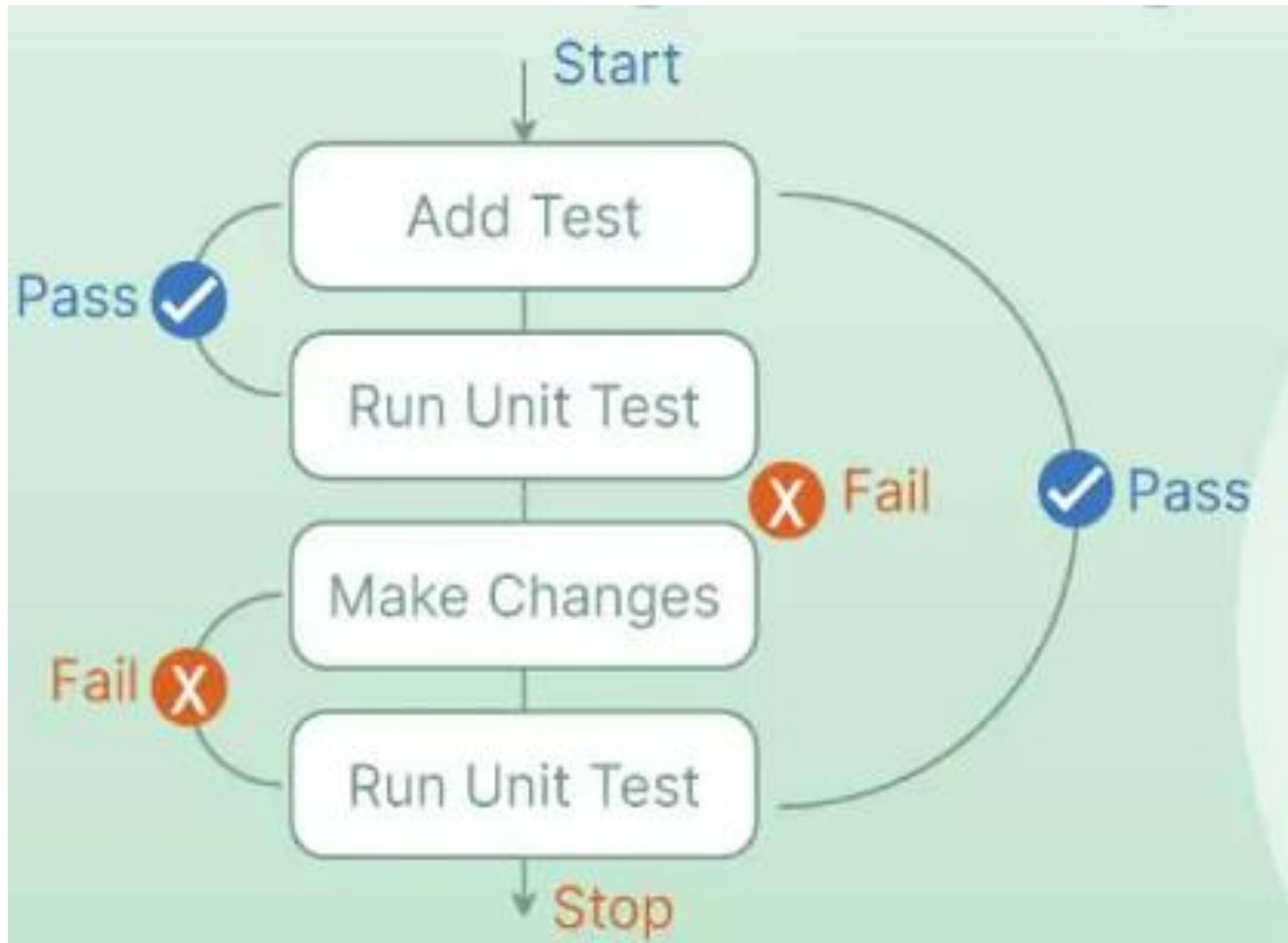
Testing: Testing in C involves various techniques and practices to ensure that your C programs are reliable, free of bugs, and meet their intended functionality

1. Unit Testing

- Unit testing is a method of **testing individual units or components of a software application**. It is typically **done by developers** and is used to ensure that the individual units of the software are working as intended.

Testing and Debugging Approaches in C

1. Unit Testing

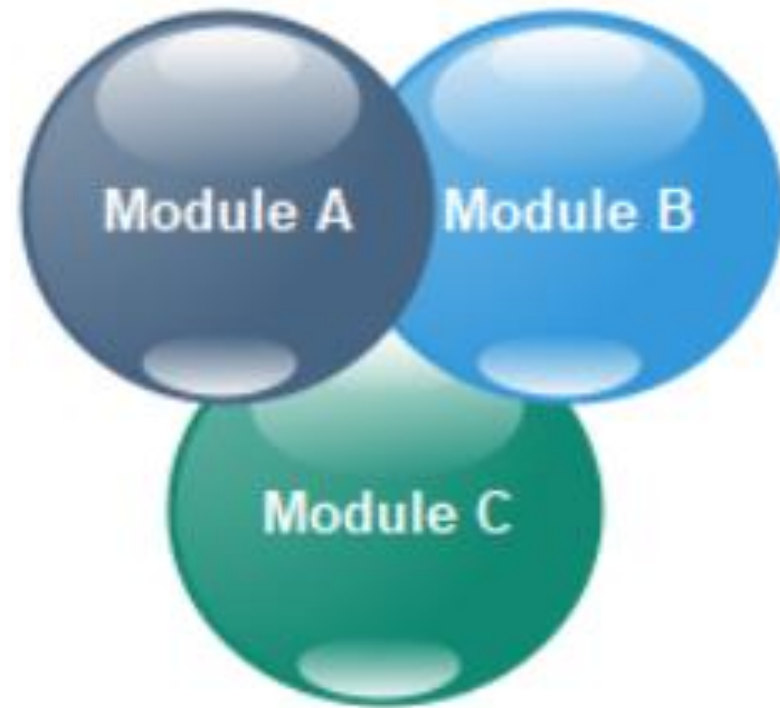


Testing and Debugging Approaches in C

Testing:

2. Integration Testing

Integration testing is a method of testing how different units or components of a software application interact with each other.



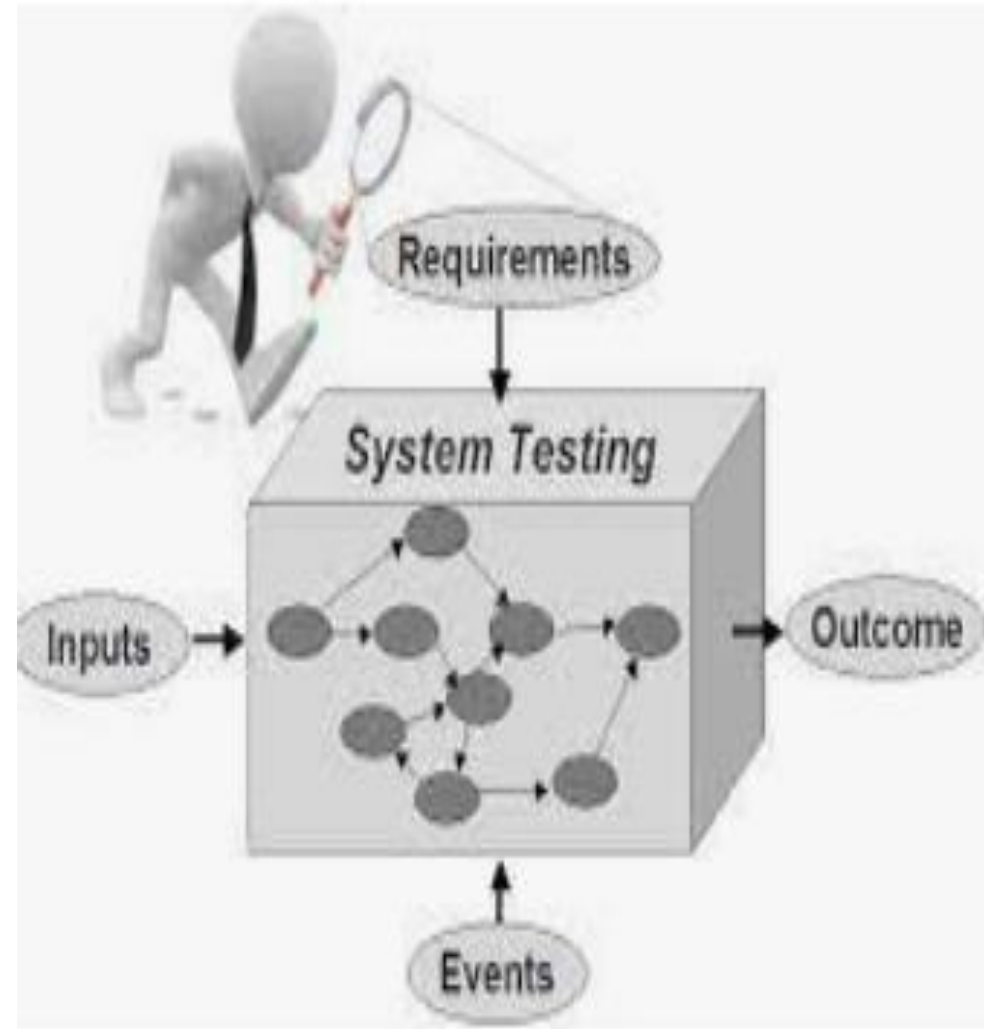
Under Integration Testing

Testing and Debugging Approaches in C

Testing:

3. System testing

It is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application.



Testing and Debugging Approaches in C

Debugging: Debugging is the process of **finding and resolving defects or problems within a computer program** that prevent the correct operation of computer software or a system.

a. Brute Force Method: The brute force approach is an approach that finds all the possible solutions to find a satisfactory solution to a given problem.

b. Back tracking: This can be defined as a general algorithmic technique that considers **searching every possible combination** in order to solve a computational problem.

Introduction to digital computers

What is computer?

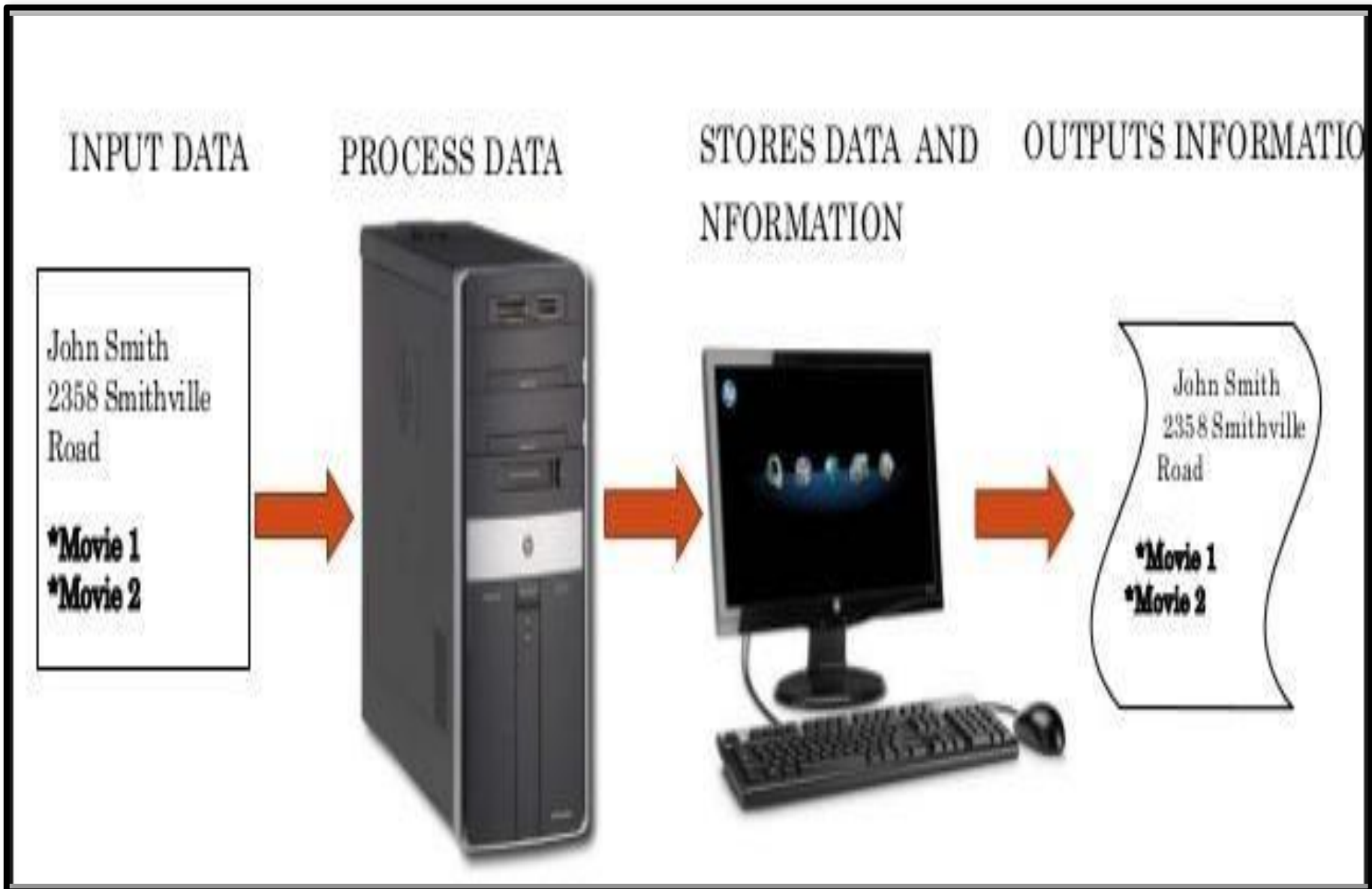
A computer is a machine or device that performs processes, calculations and operations based on instructions provided by a software or hardware program.

It has the ability to accept data (input), process it, and then produce outputs.

What is computer?

- A Computer is device that can automatically performs a set of instructions.
- The computer takes as input these instructions as a single unit, uses them to manipulate the data, and outputs the results in user-specified ways.
- The processing is fast, accurate and consistent, and is generally achieved without significant human intervention.

What is computer?



Characteristics of a Computer

1. Speed: – As you know computer can work very fast. It takes only few seconds for calculations that we take hours to complete.

- perform millions (1,000,000) of instructions and even more per second.

2. Accuracy: – The degree of accuracy of computer is very high and every calculation is performed with the same accuracy.

Characteristic of a Computer

3. Diligence: – A computer is free from tiredness, lack of concentration, low energy, etc. It can work for hours without creating any error.

4. Versatility: – It means the capacity to perform completely different type of work.

5. Automation:- Can perform task without any human intervention.

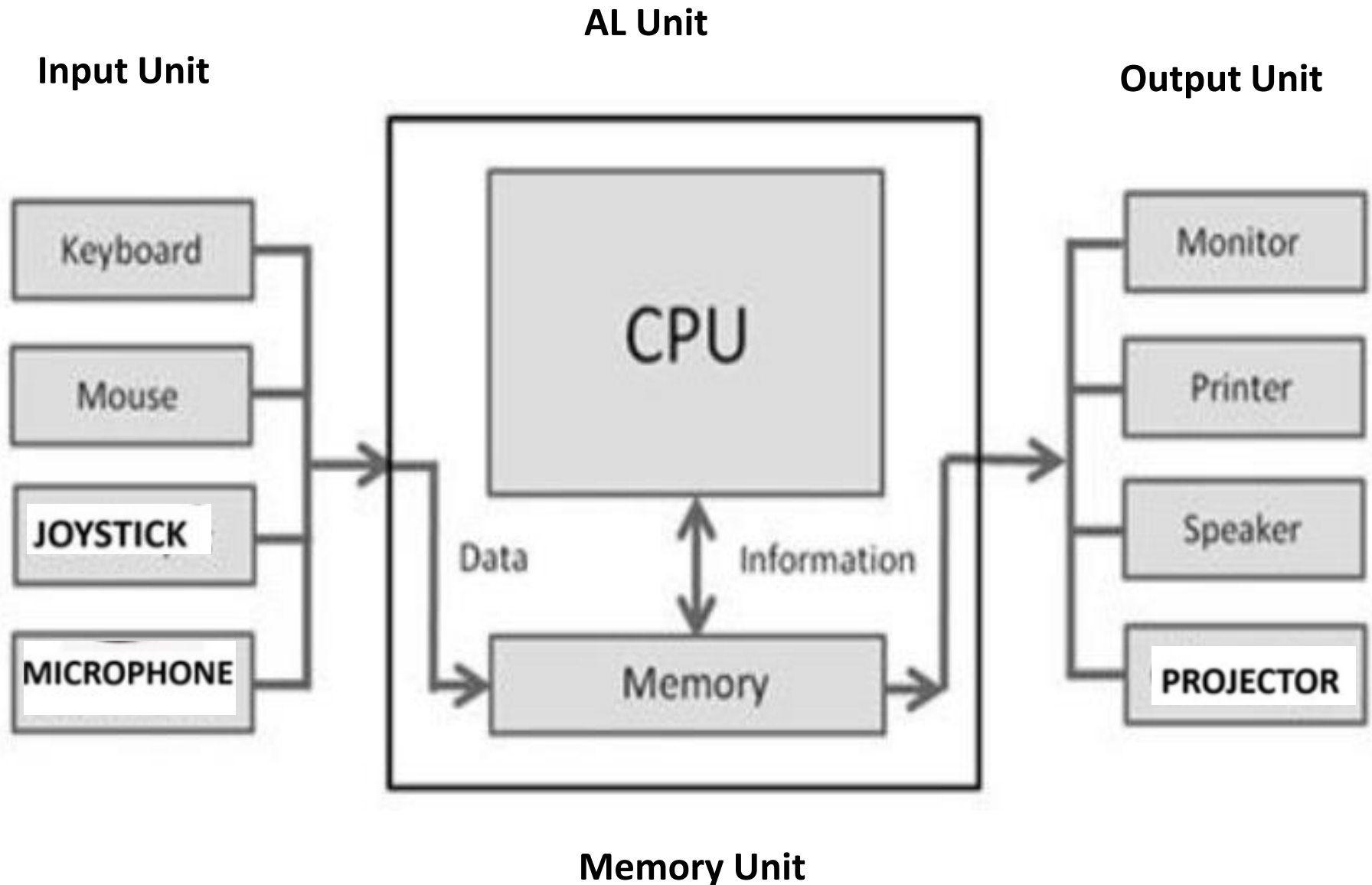
Characteristic of a Computer

6. Memory: – Computer has inbuilt memory where it can store large amount of data.

7. No IQ: – Computer is a dumb machine and it cannot do any work without instruction from the user.

8. Economical:- Since computers reduces manpower requirements and leads to efficient way of performing task.

Parts of Computer



Parts of Computer

- **Input Unit**

A computer will only respond when a command is given to the device. These commands can be given using the input unit or the input devices.

For example: Keyboard, Mouse

- The data entered can be in the form of numbers, alphabet, images, etc. We enter the information using an input device, the processing units convert it into computer understandable languages and then the final output is received by a human-understandable language.

Parts of Computer

- **Output Unit**

When we command a computer to perform a task, it reverts for the action performed and gives us a result. This result is called output. There are various output devices connected to the computer.

- The most basic of which is a monitor. Whatever we write using a keyboard or click using a mouse, is all displayed on the monitor.

For example: Monitor, Printer

Parts of Computer

- **Memory Unit**

- When we enter the data into the computer using an input device, the entered information immediately gets saved in the memory unit of the Central Processing Unit (CPU).
- The output of our command is processed by the computer, it is saved in the memory unit before giving the output to the user.

Parts of Computer

- **Control Unit**

This is the core unit which manages the entire functioning of the computer device.

The Control Unit collects the data entered using the input unit, leads it on for processing and once that is done, receives the output and presents it to the user.

Parts of Computer

- **Arithmetic & Logical Unit**

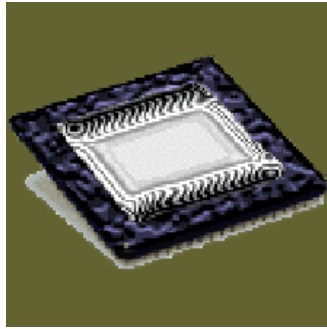
As the name suggests, all the mathematical calculations or arithmetic operations are performed in the Arithmetic and Logical Unit of the CPU.

- It can also perform actions like a comparison of data and decision-making actions.
- The ALU comprises circuits using which addition, subtraction, multiplication, division and other numerical based calculations can be performed.

Parts of a Computer

Two basic parts

Hardware



Computer **hardware** includes the physical parts of a computer

Software



Computer **software**, instructions tell a computer what to do.

Hardware

Hardware is basically that you can touch with your fingers.

- Computer Case
- CPU (central processing unit...Pentium chip)
- Monitor
- Keyboard & Mouse
- Disk Drive- CD-ROM, DVD,
- Hard Drive
- Memory (RAM)
- Speakers
- Printer

THE CENTRAL PROCESSING UNIT (CPU)

- The CPU has evolved from a bulky vacuum tube based unit of the 1940s to a modern 5cm square chip that is commonly called the microprocessor, or simple processor. It comprises the following components
 - Arithmetic and Logic Unit (ALU)
 - Special purpose registers(hold the status of a program)
 - Control Unit (CU)
 - A clock



PRIMARY MEMORY

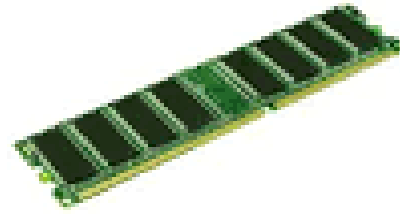
Primary memory is **computer memory that a processor or computer accesses first.**

It allows a processor to access running execution applications and services that are temporarily stored in a specific memory location.

It is a volatile memory as the data loses when power is turned off.

PRIMARY MEMORY

RAM
Random Access Memory



ROM
Read Only Memory



- **Random Access Memory (RAM-SRAM and DRAM):** RAM is essentially short term memory where data is stored as the processor needs it.
- **Read Only Memory (ROM, PROM, EPROM, EEPROM):** ROM, is a type of computer storage containing non-volatile, permanent data that, normally, can only be read, not written to.

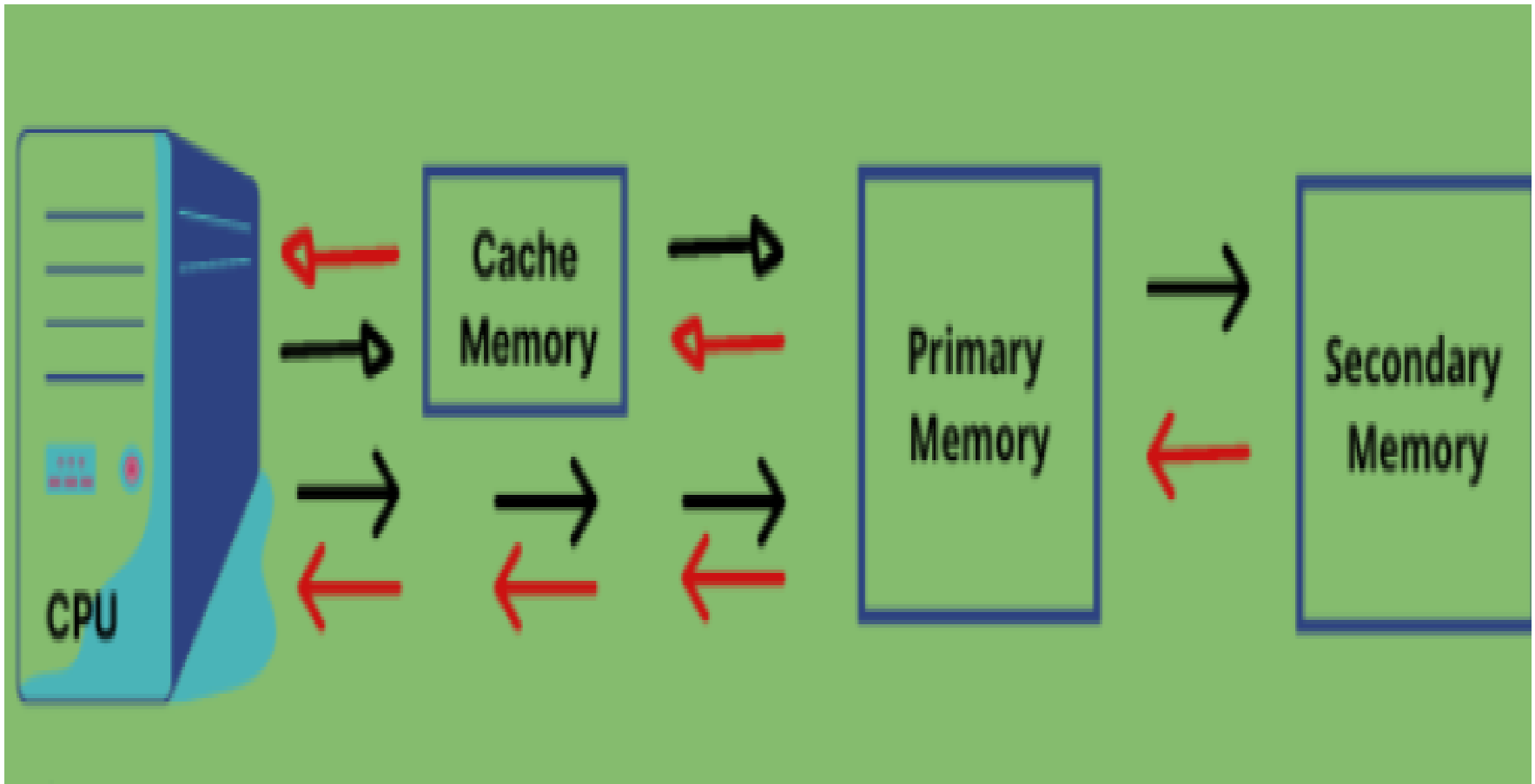
PRIMARY MEMORY

The primary memory which includes the following types:

- **Cache Memory (L1, L2 and L3):** It acts as a temporary storage area that the computer's processor can retrieve data from easily.
- **CPU Registers:** is the smallest and fastest memory in a computer. It is not a part of the main memory and is located in the CPU in the form of registers, which are the smallest data holding elements.

PRIMARY MEMORY

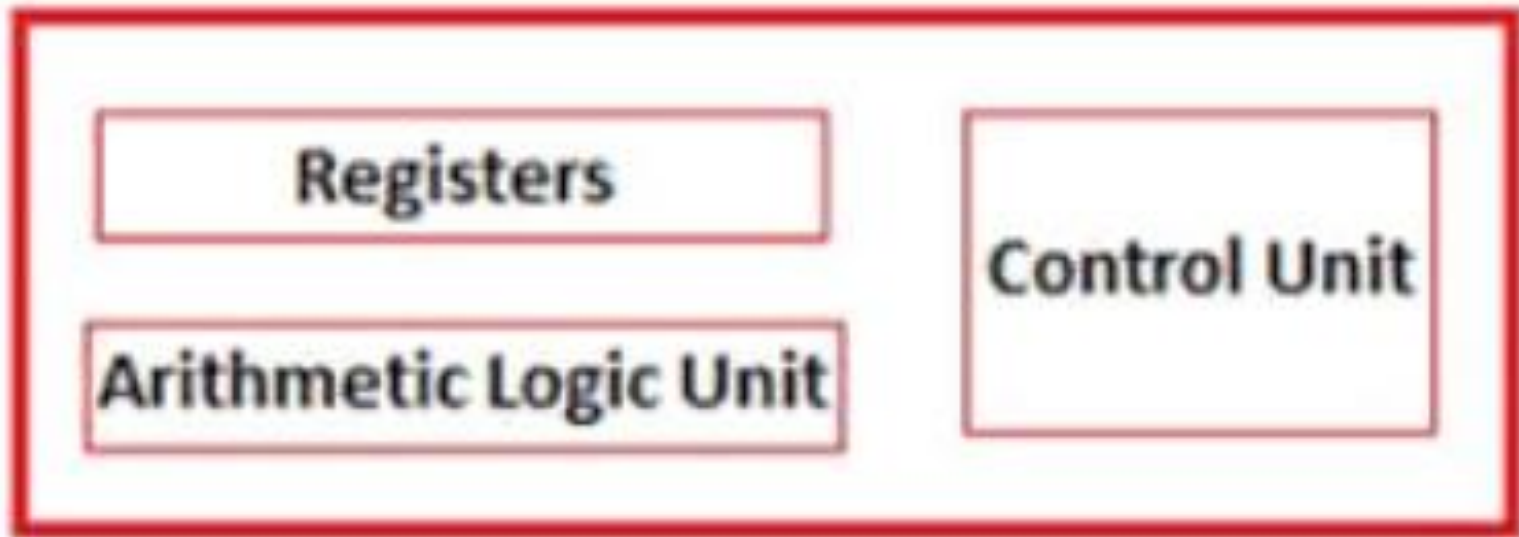
The **Cache Memory (L1, L2 and L3)**:



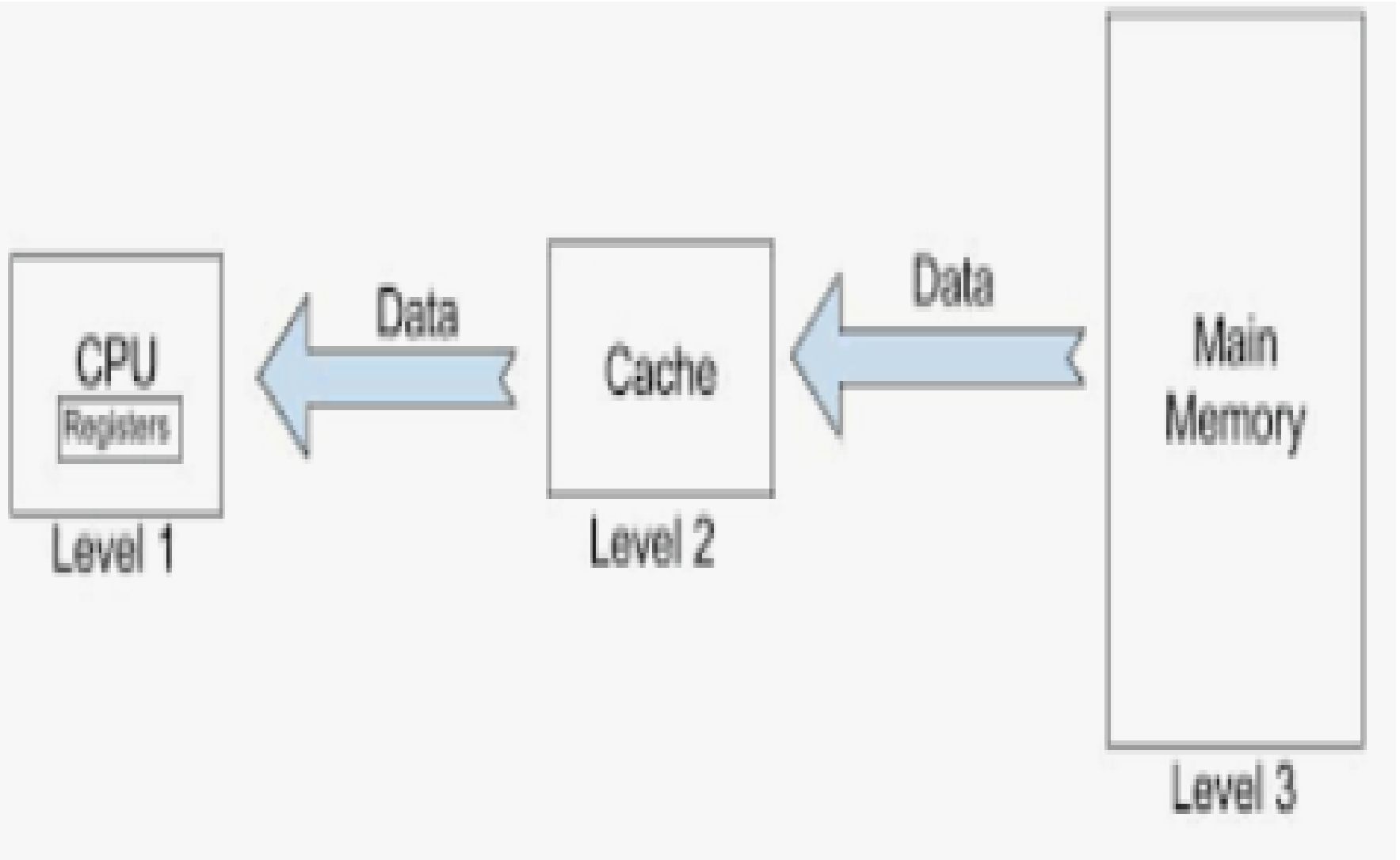
PRIMARY MEMORY

The CPU Registers

CPU



MEMORY



SECONDARY MEMORY

- Secondary memory is **computer memory that is non-volatile and persistent in nature and is not directly accessed by a computer/processor.**
- It allows a user to store data that may be instantly and easily retrieved, transported and used by applications and services.

SECONDARY MEMORY

- Hard disk including the portable disk (500 GB to 4 TB).
- Magnetic tape (20 TB).
- CD-ROM (700 MB-less than 1 GB).
- DVD-ROM (4.7 GB and 8.5 GB).
- Blu-ray disk (27 GB and 50 GB).
- Flash memory (1 GB to 128 GB).
- The obsoleted floppy disk (1.2 MB and 1.44 MB).

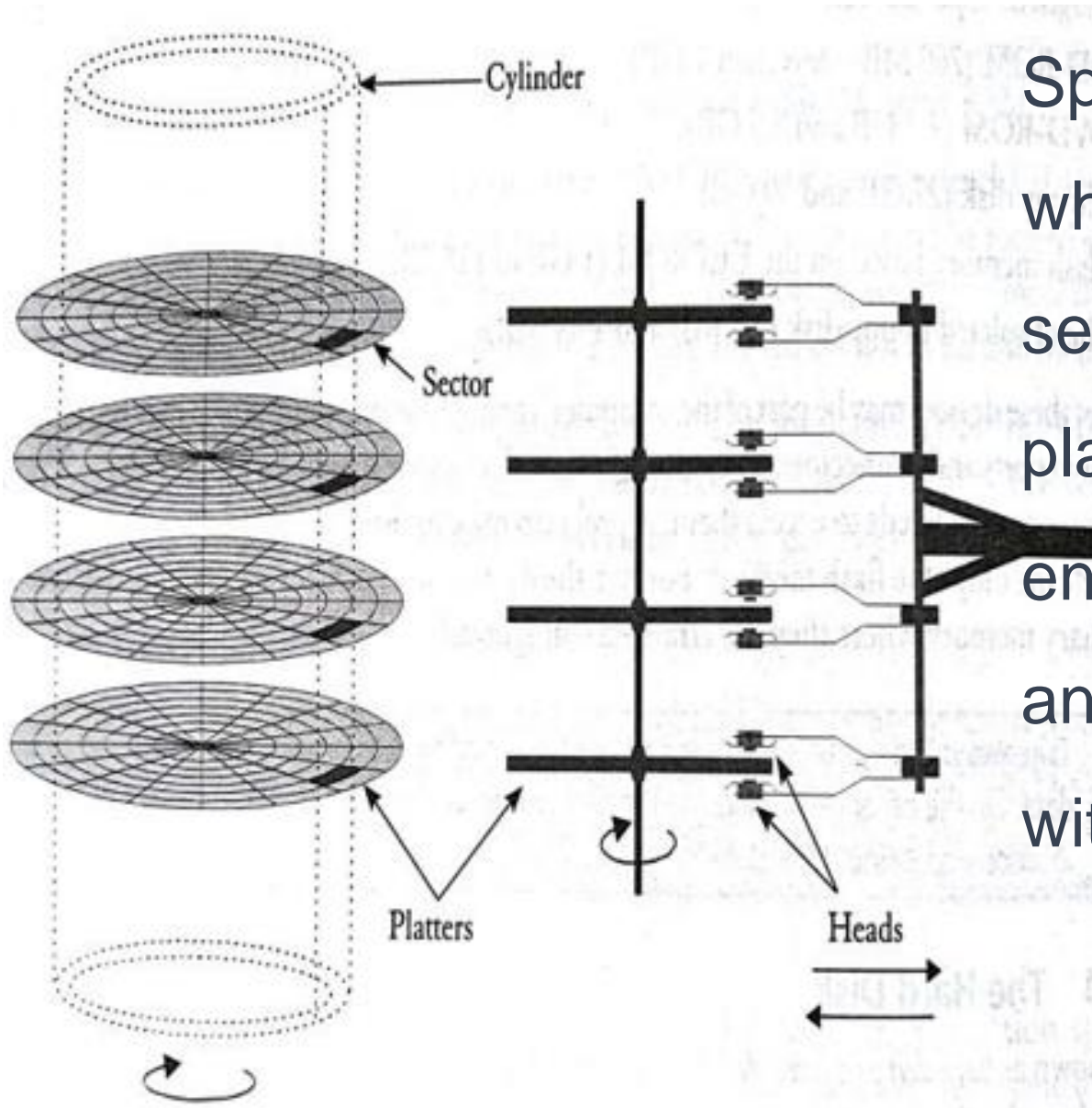
The Hard Disk

- Hard disk, also called hard disk drive or hard drive, **magnetic storage medium for a computer.**
- Hard disks are flat circular plates made of aluminum or glass and coated with a magnetic material.
- Hard disks for personal computers can store terabytes (trillions of bytes) of information.

The Hard Disk

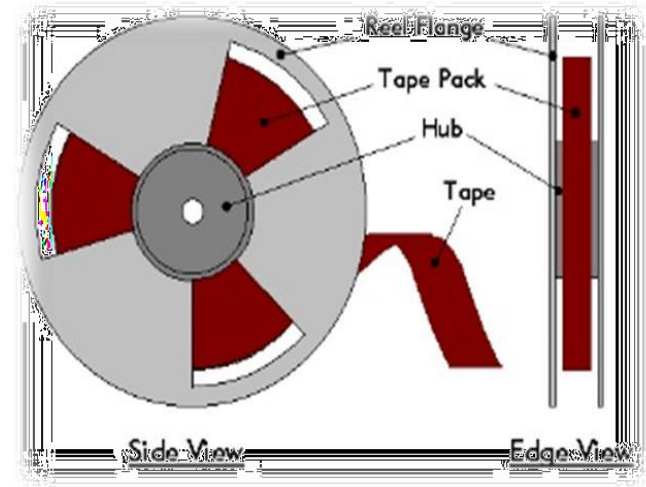
- They can **store operating systems, software programs and other files** using magnetic disks.
- Hard disk drives manage the process of both reading from and writing to the hard disk, which serves as the storage medium for data.

The Hard Disk



Spinning platters
where data is stored in
sectors, and these
platters rotate to
enable the reading
and writing of data
within specific sectors.

Magnetic Tape



Optical Disks: The CD-ROM, DVD-ROM and Blu-Ray

Flash Memory

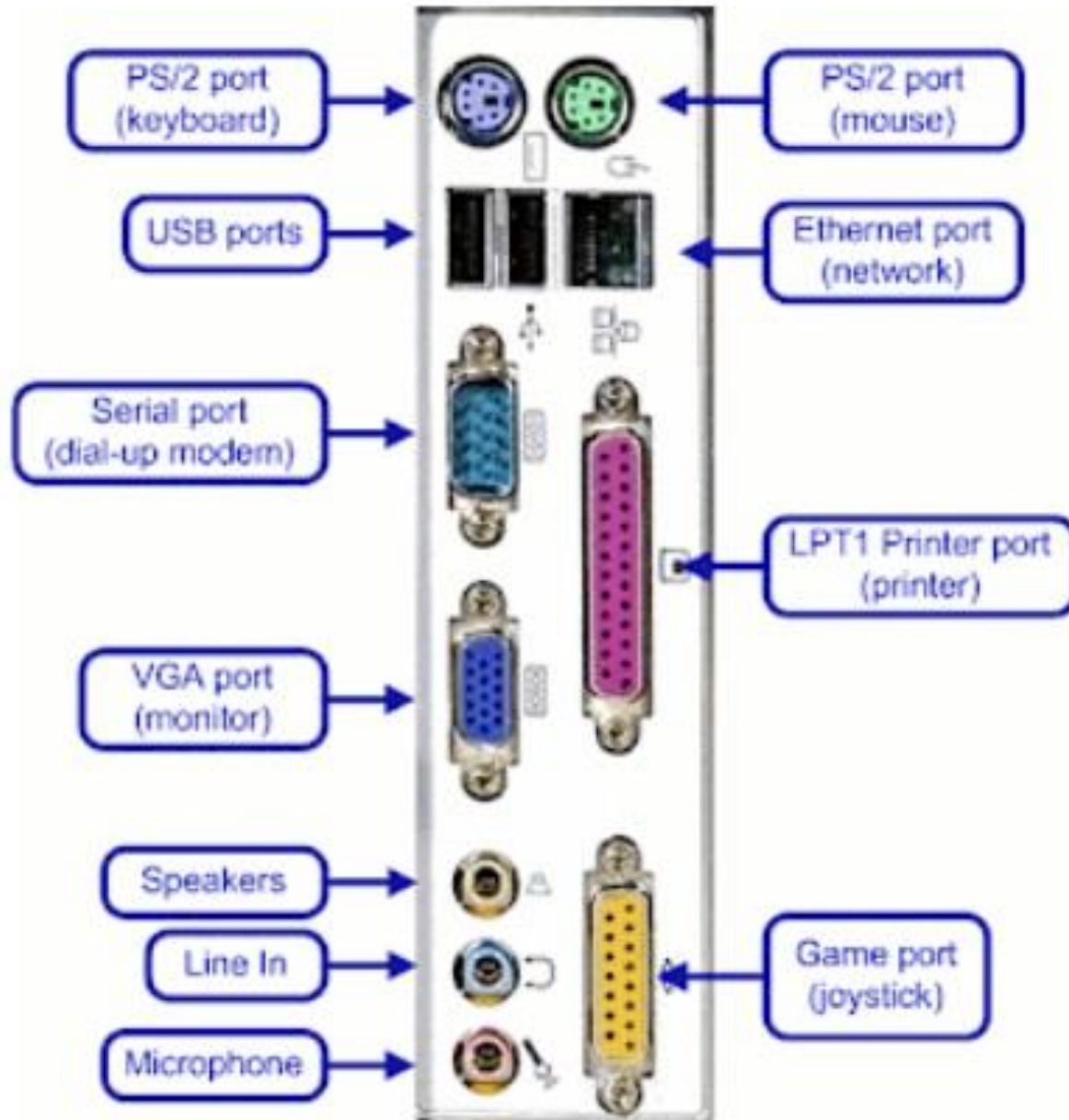


Floppy Diskette

PORTS AND CONNECTORS

1. *Universal Serial Bits (USB)*
2. *Serial port*
3. *Parallel port*
4. *Video Graphics Array (VGA) port*
5. *digital video interface (DVI)*
6. *RJ45(Registered Jack) port*
7. *PS(Personal System)/2 port*
8. *High Definition Multimedia Interface (HDMI)*

PORTS AND CONNECTORS



INPUT DEVICES

1. The Keyboard
2. Pointing Devices
3. The Scanner



INPUT DEVICES

The Keyboard: A keyboard is defined as the set of typewriter-like keys that enables you to enter data into a computer or other devices.

Pointing Devices: A pointing device, or sometimes called a pointing tool, is a hardware input device that allows the user to move the mouse cursor in a computer program or GUI operating system.

The Scanner: A scanner is an input device that scans documents such as photographs and pages of text. When a document is scanned, it is converted into a digital format.

OUTPUT DEVICES

1. The Monitor

2. Impact Printers

- *Dot-matrix Printer*
- *Daisy-wheel Printer*
- *Line Printer*

3. Non-Impact Printers

Laser Printer

Ink-jet Printer

4. Plotters

Types of Monitor



OUTPUT DEVICES

The Monitor: A computer monitor is an output device that displays information in pictorial form.

Impact Printers: An impact printer is a type of printer that operates by striking a metal or plastic head against an ink ribbon. The ink ribbon is pressed against the paper, marking the page with the appropriate character, dot, line, or symbol.

Plotters: A plotter is a printer designed for printing vector graphics. Instead of printing individual dots on the paper, plotters draw continuous lines.

Overview of Operating systems

- An **Operating System** (OS) is an interface between a computer user and computer hardware.
- An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling

Software

- **Software is a set of instructions, data or programs used to operate computers and execute specific tasks.**
- **Programs that tell the computer what to do.**
- **Provides instructions that the CPU will need to carry out.**

It is a set of instructions or programs that are used to execute any particular task. The user cannot touch the software but can see through the GUI.

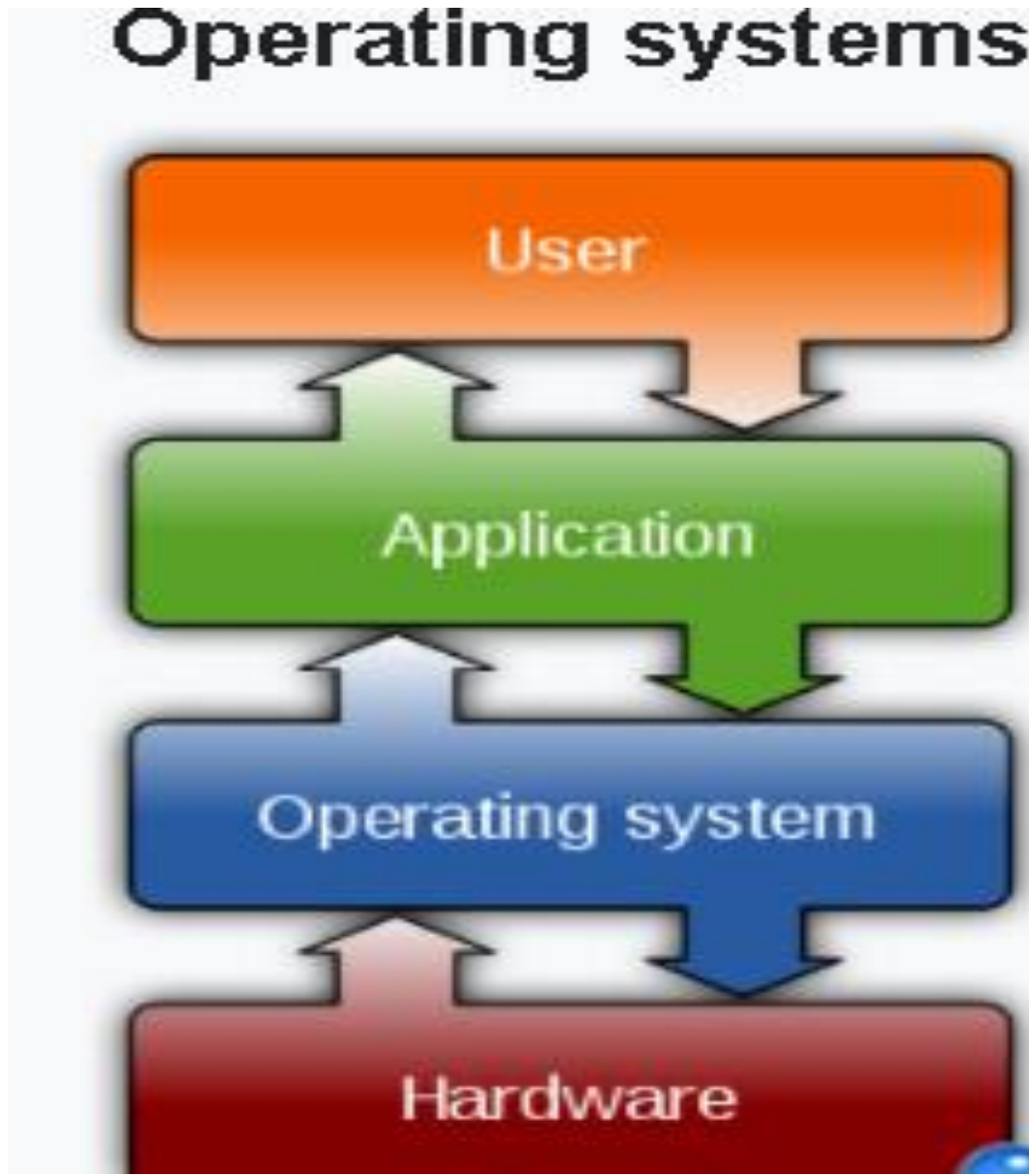
Software

1. System Software

- The **system software** is a type of computer software that is designed for running the **computer hardware parts and the application programs**. It is the platform provided to the computer system where other computer programs can execute.
- The system software act as a **middle layer between the user applications and hardware**. The operating system is the type of system software. The operating system is used to manage all other programs installed on the computer.

Software

1. System Software



Software

2. Application Software

- The **application software** that is **designed for the users to perform some specific tasks** like writing a letter, listening to music or seeing any video.
- The **operating software** runs the **application software** in the computer system.

Software

2. Application Software

- The **difference between system software and application software** is the difference in the **user interface**.
- In **system software**, there is **no user interface present** whereas in **application software** the **user interface is present** for each software.

Software

2. Application Software



AOL Desktop



Apple Mail 10



IBM Notes 9



Outlook 2000-03



Postbox



Thunderbird



Windows 10 Mail



Windows Live Mail



AOL Alto
iOS app



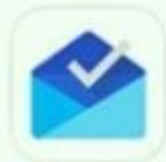
BlackBerry



Gmail Android app



Gmail Android
app IMAP



Google Inbox
iOS app



iOS 10 Mail



Outlook
Android app



Outlook
iOS app

Software

3. Programming Languages

- The programming language is the **third category of computer software which is used by the programmers to write their programs, scripts, and instructions** which can be executed by a computer.
- The other name of the programming language is a computer language that can be used to create some common standards.
- The examples of programming languages are C, JAVA, C++,Python and other languages.

Software

3. Programming Languages



Assemblers, Compilers, Interpreters And Programming languages

Assembler

The Assembler is a Software that **converts an assembly language code to machine code**. It takes basic **Computer commands and converts them into Binary Code** that Computer's Processor can use to perform its Basic Operations.

These instructions are assembler language or assembly language.

Assembler

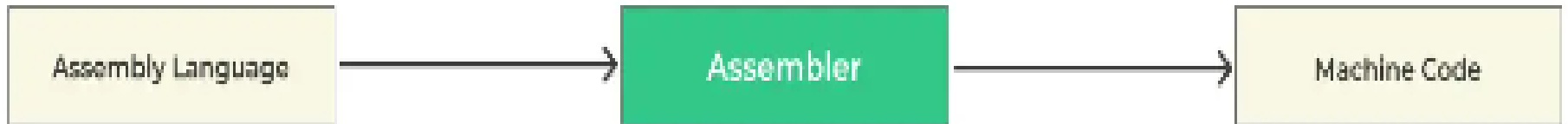
Assembly Language

An **assembly language is a low-level language**. It gives instructions to the processors for different tasks. **It is specific for any processor**. The machine language only **consists of 0s and 1s** therefore, it is difficult to write a program in it.

Machine Code

These are **machine language instructions, which are used to control a computer's central processing unit (CPU)**. Each instruction causes the CPU to perform a very specific task, such as a load, a store an arithmetic logic unit (ALU) operation on one or more units of data in the CPU's registers or memory.

Assembler



```
start  mov  rax, 1
      mov  eax, 3
      mov  ebx, 4
      add  eax, ebx, ecx
```

A screenshot of a code editor window with a green title bar and three colored window control buttons (red, yellow, green) on the left. The editor contains five lines of assembly code. The first line is "start mov rax, 1". The next three lines are "mov eax, 3", "mov ebx, 4", and "add eax, ebx, ecx". Each line has a light gray highlight bar to its left, and the entire code block is highlighted with a light gray background.

Assembly Language (Example)

```
1010010 100101 100101 101001
1010010 100101 100101 101001
1010010 100101 100101 101001
1010010 100101 100101 101001
1010010 100101 100101 101001
```

A screenshot of a code editor window with a green title bar and three colored window control buttons (red, yellow, green) on the left. The editor contains five lines of binary code, each consisting of four groups of eight bits separated by spaces: "1010010 100101 100101 101001". Each line has a light gray highlight bar to its left, and the entire code block is highlighted with a light gray background.

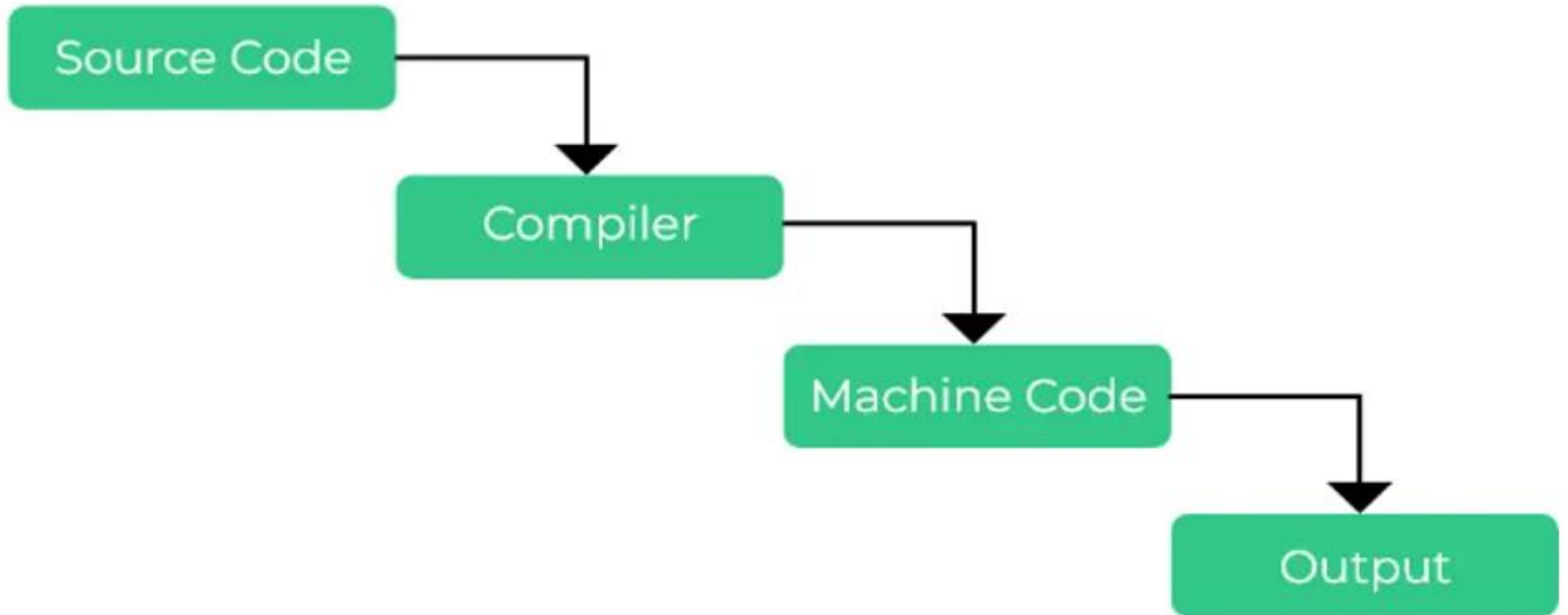
Machine Code (Example)

Compiler

A **Compiler** is a program that **translates source code from a high-level programming language to a lower level language computer understandable language** (e.g. assembly language, object code, or machine code) to create an executable program

- Compiler goes through the **entire code at once**.
- It helps to **detect error** and get displayed after reading the entire code by compiler.
- “Compilers turns the **high level language to binary language or machine code at only time once**”, it is known as Compiler.

Compiler



Interpreter

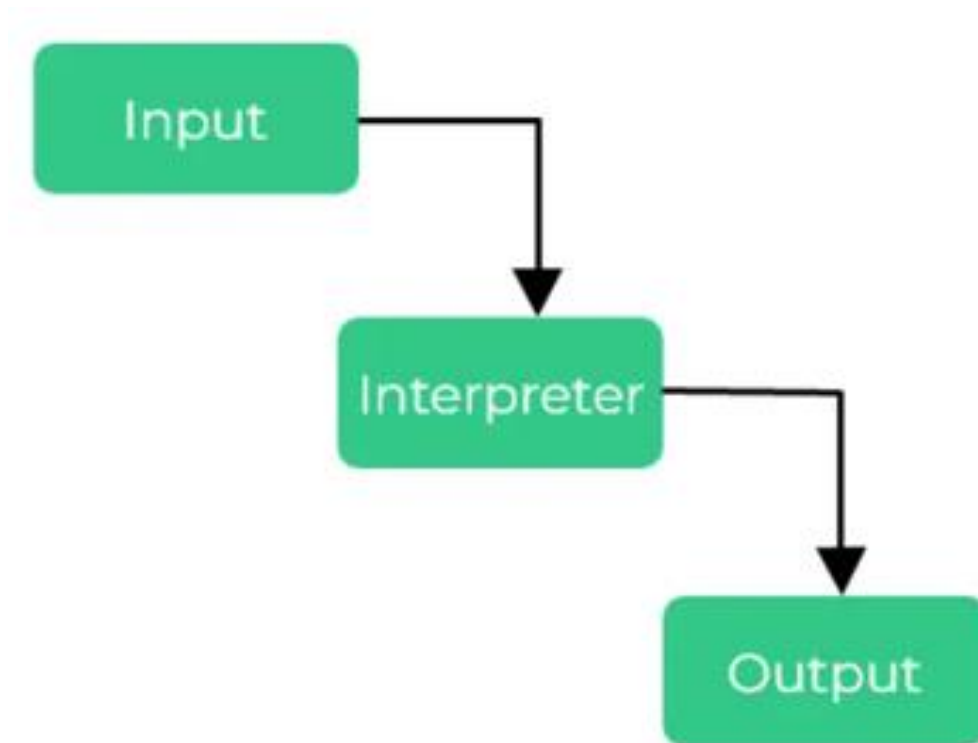
An interpreter is also a program like a compiler that **converts assembly language into Machine Code.**

- Interpreter goes through **one line of code at a time and executes it** and then goes on to the next line of the code and then the next and **keeps going on until there is an error in the line or the code has completed.**
- It is **5 to 25 times faster than a compiler**, but it stops at the line where error occurs and then again if the next line has an error too.

Interpreter

Whereas a **compiler gives all the errors in the code at once.**

Also, a compiler saves the machine codes for future use permanently,
but an interpreter doesn't, but an interpreter occupies less memory.



Interpreter vs Compiler

Interpreter is differing from compiler such as,

- Interpreter is faster than compiler.
- It contains less memory.
- Interpreter executes the instructions in to source programming language.

Assembler vs Compiler

Compiler	Assembler
Compiler converts the high-level language source code into machine level language code.	An assembler converts the assembly level language code into the machine level language code.
The input of a compiler is high-level language source code.	Whereas, its input is low level assembly code.
Compiler converts the whole source code to machine code at once.	Assembler does not convert the code in one go.

Assembler vs Compiler

Compiler	Assembler
It has the following phases: syntax analysis, semantic analysis, intermediate code generation, code optimization, code generation and error handling.	An assembler completes the task in two passes.
It produces a machine code in form of mnemonics.	It produces binary code in form of 0s and 1s.

Programming Language

- A programming language is a computer language that is used by programmers (developers) to communicate with computers.
- It is a set of instructions written in any specific language (C, C++, Java, Python) to perform a specific task.
- A programming language is mainly used to develop desktop applications, websites, and mobile applications.

Types of Programming Language

1. Low-level programming language

Low-level language is machine-dependent (0s and 1s) programming language. The processor runs low-level programs directly without the need of a compiler or interpreter, so the programs written in low-level language can be run very fast.

Low-level language is further divided into two parts -

- i. Machine Language**
- ii. Assembly Language**

Types of Programming Language

Low-level language is further divided into two parts -

i. Machine Language

- Machine language is a type of low-level programming language. It is also called as machine code or object code. Machine language is easier to read because it is normally displayed in binary or hexadecimal form (base 16) form.
- The advantage of machine language is that it helps the programmer to execute the programs faster than the high-level programming language.

Types of Programming Language

Low-level language is further divided into two parts -

ii. Assembly Language

- Assembly language (ASM) is also a type of low-level programming language that is designed for specific processors. It represents the set of instructions in a symbolic and human-understandable form.
- It uses an assembler to convert the assembly language to machine language. The advantage of assembly language is that it requires less memory and less execution time to execute a program.

Types of Programming Language

2. High-level programming language

- High-level programming language (HLL) is designed for developing user-friendly software programs and websites. This programming language requires a compiler or interpreter to translate the program into machine language (execute the program).
- The main advantage of a high-level language is that it is easy to read, write, and maintain.
- High-level programming language includes Python, Java, JavaScript, PHP, C#, C++, Objective C, Cobol, Perl, Pascal, LISP, FORTRAN, and Swift programming language.