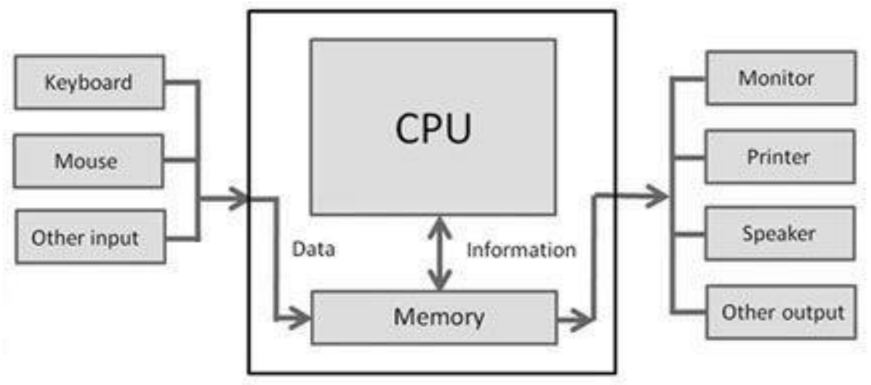
MODULE 1

Basics of Computer Architecture : Processor, Memory, Input& Output devices, Application Software & System software: Compilers, interpreters, High level and low level languages, Introduction to structured approach to programming, Flow chart, Algorithms, Pseudo code (bubble sort, linear search - algorithms and pseudocode)

Basic of Computer Architecture

Computer is an advanced electronic device that takes raw data as input from the user and processes these data under the control of set of instructions (called program) and gives the result (output) and saves output for the future use.

Computer Architecture /Basic Functional units of a Computer



The basic components of a modern digital computer are: Input Device, Output Device, Central Processor Unit (CPU), mass storage device and memory.

1.Central Processing Unit (CPU)

It is the brain of the computer system. All major calculation and comparisons are made inside the CPU and it is also responsible for activation and controlling the operation of other unit. This unit consists of two major components that are arithmetic logic unit (ALU) and control unit (CU).

- Arithmetic Logic Unit (ALU)

Here arithmetic logic unit performs all arithmetic operations such as addition, subtraction, multiplication and division. It also uses logic operation for comparison such as less than,greater than etc.

- Control Unit (CU)

The control unit of a CPU controls the entire operation of the computer. It also controls all devices such as memory, input/output devices connected to the CPU.

2.Input unit and Output Unit

The input and output unit consists of devices used to transmit information between the external world and computer memory. The information fed through the input unit is stored in computer's memory for processing and the final result stored in memory can be recorded or display on the output medium.

Eg: Mouse, Keyboard, Printer, Monitor, etc.

3.Memory Unit

Computer **memory** is any physical device capable of storing information temporarily, like RAM (random access memory), or permanently, like ROM (read-only memory).

Memories can be classified into two categories

- Primary Memory
- Secondary Memory

Primary memory(Main memory) is computer **memory** that is accessed directly by the CPU. There are two types of primary memory.

- Read Only Memory (ROM)
- Random Access Memory (RAM)

The content of ROM cannot be changed and can be used only by CPU. It is needed to store Basic Input Output System (BIOS), which is responsible for booting. This memory is permanent in storage (non volatile) and is very small in size.

The RAM is a volatile memory i.e. its contents get destroyed as soon as the computers is switched off. All kinds of processing of CPU are done in this memory.

RAM	**ROM**
Temporary Storage	Permanent Storage
Store data in GBs (1-128 GB per chip)	Store Data in MBs(4-8MB per chip)
volatile	Non volatile
Used in normal operations	Used for startup process of computer

Writing data is faster	Writing data is slower

Types of ROM

1. PROM (Programmable read-only memory) – It can be programmed by user. Once programmed, the data and instructions in it cannot be changed.
2. EPROM (Erasable Programmable read only memory) – It can be reprogrammed. To erase data from it, expose it to ultra violet light. To reprogram it, erase all the previous data.
3. EEPROM (Electrically erasable programmable read only memory) – The data can be erased by applying electric field, no need of ultra violet light. We can erase only portions of the chip.

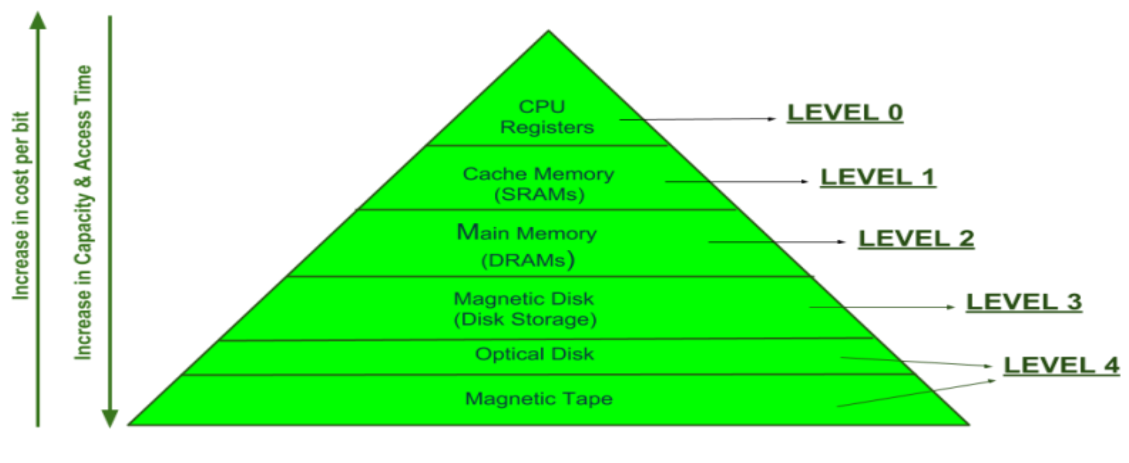
Secondary Memory

Primary memory has limited storage capacity and is volatile. Secondary memory overcome this limitation by providing permanent storage of data and in bulk quantity. Secondary memory is also termed as external memory and refers to the various storage media on which a computer can store data and programs. The Secondary storage media can be fixed or removable. Fixed Storage media is an internal storage medium like hard disk that is fixed inside the computer. Storage medium that are portable and can be taken outside the computer are termed as removable storage media.

eg: Hard disk, Solid state driveMagnetic Tapes, Pen drive

Memory Hierarchy

In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The figure below clearly demonstrates the different levels of memory hierarchy:



MEMORY HIERARCHY DESIGN

Registers

Purpose of having register is fast retrieval of data for processing by CPU. It is high speed storage areas, least storage capacity, directly accessed & manipulated by CPU during instruction execution.

Eg. of Registers: ACC, IR, PC, MAR, MDR ,GPR(R0,R1…)

Number of registers: Ten to hundreds

Examples of registers are:

1.Accumulator(ACC)

-Used to store data taken from memory.

2.Memory Address Register(MAR)

-address of the location to be accessed from memory.

3.Memory Data Register(MDR)

-Contains data to be written onto or to be readout from the addresses location.

4.Program Counter(PC)

Used to keep the track of execution of the program. It contains the memory address of the next instruction to be fetched from the memory.

5.Instruction Register(IR)

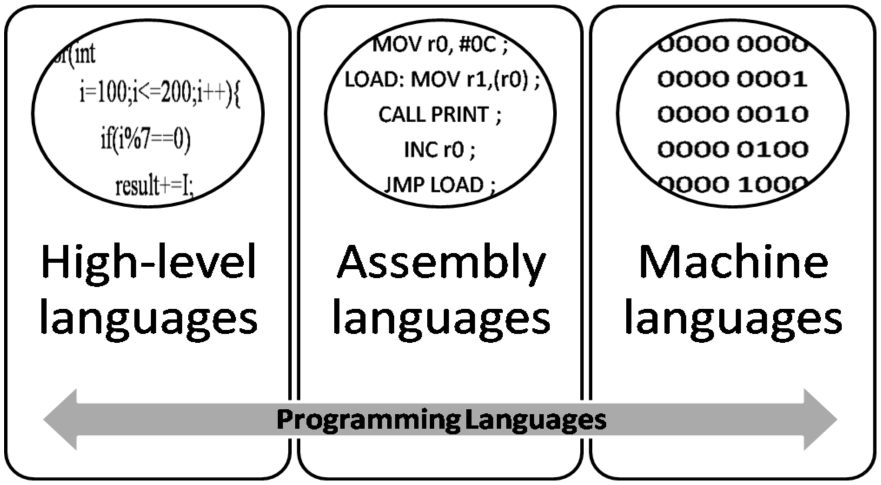
-It holds the instruction which is just about to be executed

Cache Memory

- Very high speed memory in between RAM & CPU
- Increases speed of processing
- CPU first checks cache for data. If data is not found in cache, then looks for it in RAM

[Explanation of Primary Memory(Main memory) and Secondary memory is specified in page 1 and 2.]

Types of Programming Languages



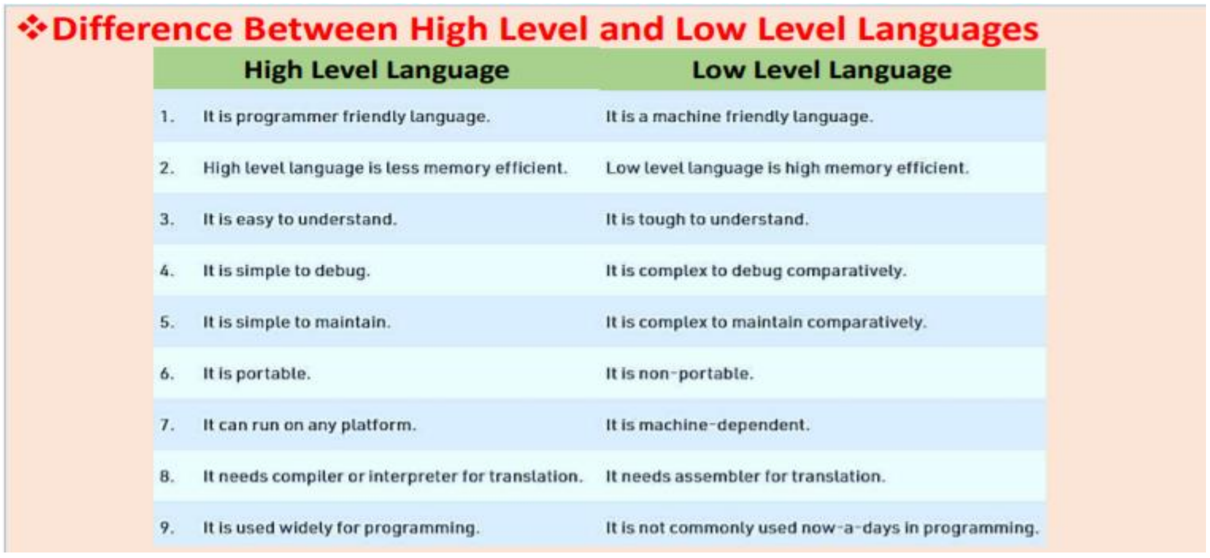
1.**Machine Language**: It is a low level language made up of binary numbers or bits that a computer can understand. It is the only language the computer can understand.

2.Assembly Language: Programming language that overcomes the limitations of machine level language.

Example: 8085 programing

Eg:-ADD A, B;[It adds the values at the operands addressed by A and B, result at A]

3.High Level Languages: High level language **(**HLL**)** is a programming language such as C, FORTRAN, or Pascal that enables writing of computer instructions in a way that is easily understandable and close to human language**.** Such languages are considered high-level because they are closer to human languages.



❖Difference Between High Level and Low Level Languages

	High Level Language	Low Level Language
1.	It is programmer friendly language.	It is a machine friendly language.
2.	High level language is less memory efficient.	Low level language is high memory efficient.
3.	It is easy to understand.	It is tough to understand.
4.	It is simple to debug.	It is complex to debug comparatively.
5.	It is simple to maintain.	It is complex to maintain comparatively.
6.	It is portable.	It is non-portable.
7.	It can run on any platform.	It is machine-dependent.
8.	It needs compiler or interpreter for translation.	It needs assembler for translation.
9.	It is used widely for programming.	It is not commonly used now-a-days in programming.

Translator Software

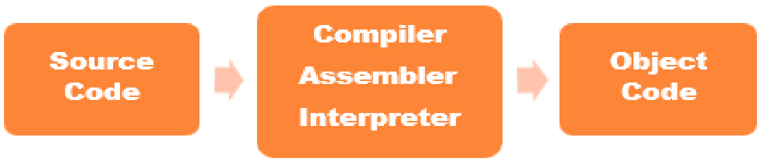
Converts program in High-level language & Assembly language to Machine-level language

Three kinds:

— Assembler
— Compiler
— Interpreter

Assembler converts Assembly language program to Machine language.

Compiler & Interpreter converts High-level language program to Machine language.



Compiler	Interpreter
Looks at entire source code	Looks at a source code line-by-line
Entire source code converted into object code	Line-by-line conversion
Object code executed multiple times	For each execution, source code interpreted and then executed
During execution, Source Code and Compiler not needed	Source code and Interpreter needed during execution
Compiled programs execute faster	Programs execute slower

Software

Software is a set of instructions, data or programs used to operate computers and execute specific tasks.

There are 2 types of softwares: 1)System software 2)Application software

System Software and Application Software

System Software:

System Software is the type of software which is the interface between application software and system. Low level languages are used to write the system software. System Software maintain the system resources and give the path for application software to run. An important thing is that without system software, system cannot run. It is a general purpose software.

Application Software:

Application Software is the type of software which runs as per user request. It runs on the platform which is provide by system software. High level languages are used to write the application software. It is a specific purpose software.

The main difference between System Software and Application Software is that without system software, system cannot run on the other hand without application software, system always runs.

SL.NO	SYSTEM SOFTWARE	APPLICATION SOFTWARE
1.	System Software maintain the system resources and give the path for application software to run.	Application software is built for specific tasks.
2.	Low level languages are used to write the system software.	High level languages are used to write the application software.
3	Machine Dependent	Machine independent
4	It is a general-purpose software.	It is a specific purpose software.
5.	Without system software, system can't run.	Without application software system always runs.
6	System software runs when system is turned on and stop when system is turned off.	While application software runs as per the user's request.
7	Compiler, Operating System, Interpreter	Photoshop, Microsoft Office, VLC

System Translator

A translator is a programming language processor that converts a computer program from one language to another. It takes a program written in source code and converts it into machine code. It discovers and identifies the error during translation. There are 3 different types of translators as follows:

1) **Compiler**

A compiler is a translator used to convert high-level programming language to low-level programming language.

Eg: gcc, javac, g++

2) **Interpreter**

Just like a compiler, is a translator used to convert high-level programming language to low-level programming language.

Example: Python, jvm

3) **Assembler**

An assembler is a translator used to translate assembly language to machine language.

Example: Fortran Assembly Program (FAP), Macro Assembly Program (MAP)

BASIS FOR COMPARISON	COMPILER	INTERPRETER
Input	It takes an entire program at a time.	It takes a single line of code or instruction at a time.
Output	It generates intermediate object code.	It does not produce any intermediate object code.
Working mechanism	The compilation is done before execution.	Compilation and execution take place simultaneously.
Speed	Comparatively faster	Slower
Errors	Display all errors after compilation, all at the same time.	Displays error of each line one by one.
Error detection	Difficult	Easier comparatively
Example	Gcc, g++, javac	Python, jvm

Structured Approach to Programming

Structured Programming Approach can be defined as a programming approach in which the program is made as a single structure. It means that the code will execute the instruction by instruction one after the other. It doesn't support the possibility of jumping from one instruction to some other with the help of any statement like GOTO, etc. Therefore, the instructions in this approach will be executed in a serial and structured manner. The languages that support Structured programming approach are:

- C
- C++
- java

Advantages of Structured Programming Approach:

1. Easier to read and understand
2. Easier to Maintain
3. Easier to Debug
4. Machine-Independent, mostly.

Disadvantages of Structured Programming Approach:

1. Since it is Machine-Independent, so it takes time to convert into machine code.
2. The converted machine code is not the same as for assembly language.

Control Structures

Control structures specify the statements to be executed and the order of execution of statements.

Three types

Sequential— instructions are executed in linear order

Selection (branch or conditional)—it asks a true/false question and then selects the next instruction based on the answer.

Iterative (loop)—it repeats the execution of a block of instructions.

Flowchart, Algorithm and Pseudo Code

Algorithm

It is a complete step by step representation of the solution of the problem, represented in English like Languages. An algorithm can be abstract or quite detailed. A detailed algorithm consists of every step, equivalent to one instruction of a programming language.

Characteristics of good algorithm:

1. An algorithm should be finite.
2. Steps in an algorithm should be clear and unambiguous.
3. Algorithm should be precise.
4. Algorithm must have 0 or more inputs and 1 or more output.

Find the sum of two numbers

```
Step 1 : start
Step 2 : input first number as a
Step 3 : input second number as b
Step 4 : add a and b and assign to a variable sum
Step 5 : print sum
Step 6 : stop
```

Pseudo Code

It is a more formal representation than the algorithm. Here, we represent every step in a formal way which is very close to the actual programming language representation. Pseudocode is the basis of algorithm. All pseudocodes will start with the keyword “START” or BEGIN and complete with keyword “STOP” or “END”.

Example: Pseudocode to find the sum of two numbers

```
BEGIN
NUMBER a, b, sum
OUTPUT("Input number1:")
INPUT a
OUTPUT("Input number2:")
INPUT b
sum=a+b
OUTPUT sum
END
```

Example: Pseudocode to find the area of a circle

```
BEGIN
NUMBER r, area
INPUT r
area=3.14*r*r
OUTPUT area
END
```

Comparison between algorithm and Pseudocode

Algorithm	**Pseudocode**
Written in plain English lang or natural language	Written in format similar to high level language
Time consuming and certain branch and loop statements are difficult to depict in algorithm	Pseudocode is easy to write and understand
Algorithm is the basis of any programming language	Pseudocode is the basis of algorithm
Provides the procedure in which the program would be carried out to get the expected output	Provides outline of the flow of program

Flowchart

Very popular method to represent the steps of the solution is the flowchart, which uses many graphical symbols and thus, is more understandable. The symbol used for various different types of statements are as shown

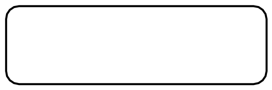
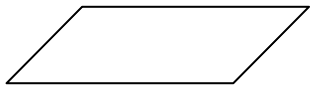
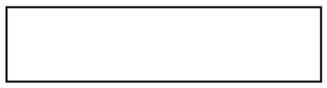
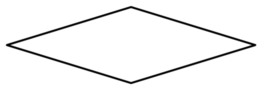
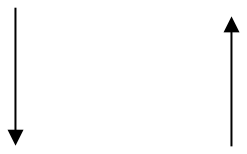
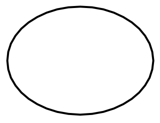
Advantages:

1. Better way of communication-provides better way of communicating the logic of the system.
2. Effective Analysis- using flowchart problem can be analyzed more efficiently.
3. Efficient coding- it is very useful during program development phase.
4. Proper Debugging- it helps in debugging.

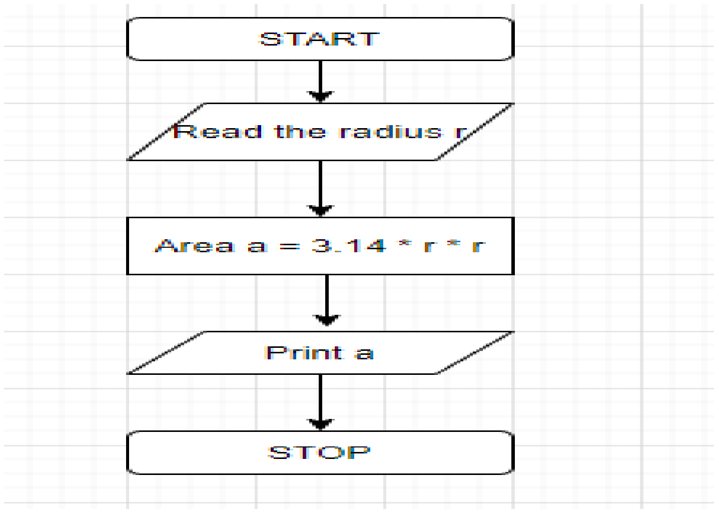
Disadvantages:

1.Complex logic makes flowchart become complex.

2. Alteration or modification require redrawing of flowchart

Start, Stop	
Read, Print	
Processing Statements	
Condition Check	
Direction of flow	
Connectors (for longer flow chart)	

Example: Flowchart for finding area of circle



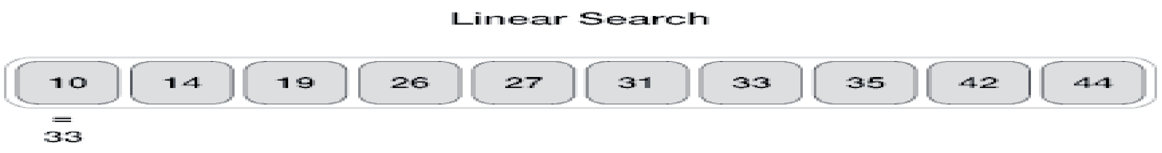
Comparison between algorithm and flowchart

Algorithm	Flowchart
• **Algorithm is step by step procedure to solve the problem** • **Plain text is used** • **Complex to understand** • **Algorithm doesn’t follow any rules**	• **It is pictorial representation of algorithm** • **Symbols/shapes are used** • **Simple to understand** • **It follows rules to construct**

Linear search

- In this type of search, a sequential search is made over all items one by one.
- Every item is checked and if a match is found then that particular item is returned

otherwise the search continues till the end of the data collection.



Algorithm

step1:start

Step2: read the element to be searched into key

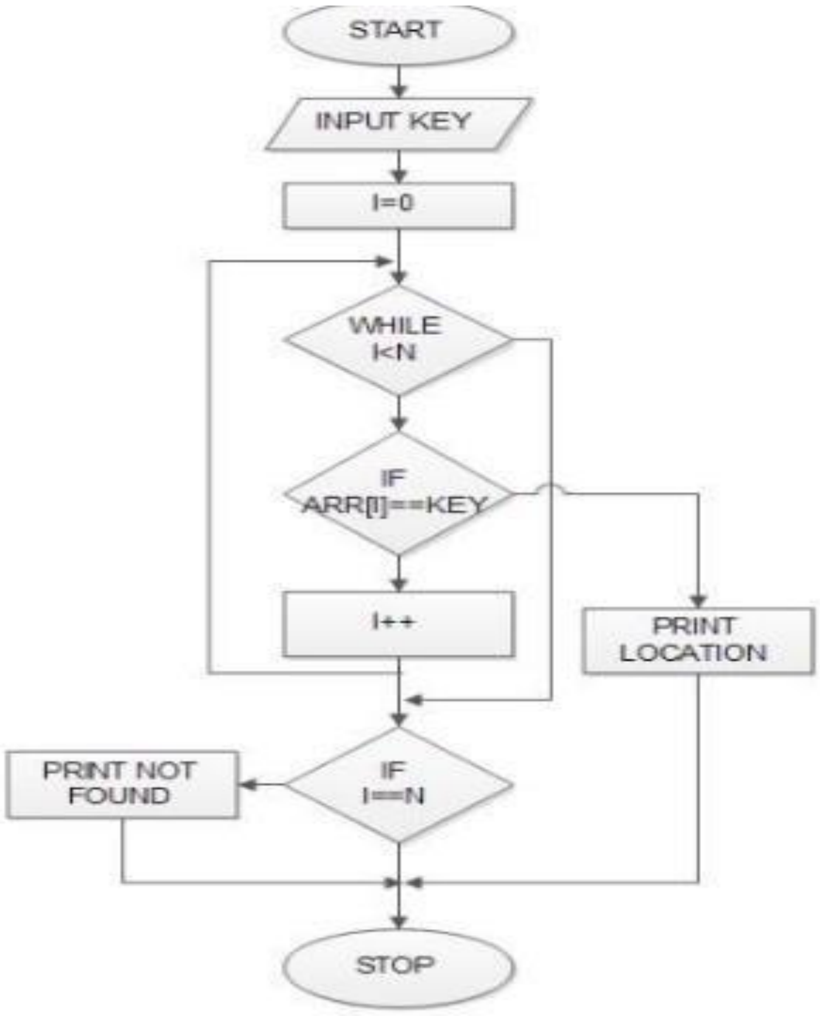
Step 3: initialize i as 0 Step 4: repeat untill i < n

Step 4.1: if current element arr[i]== key then print element found ,go to step 5

Step 4.2: Set i to i + 1

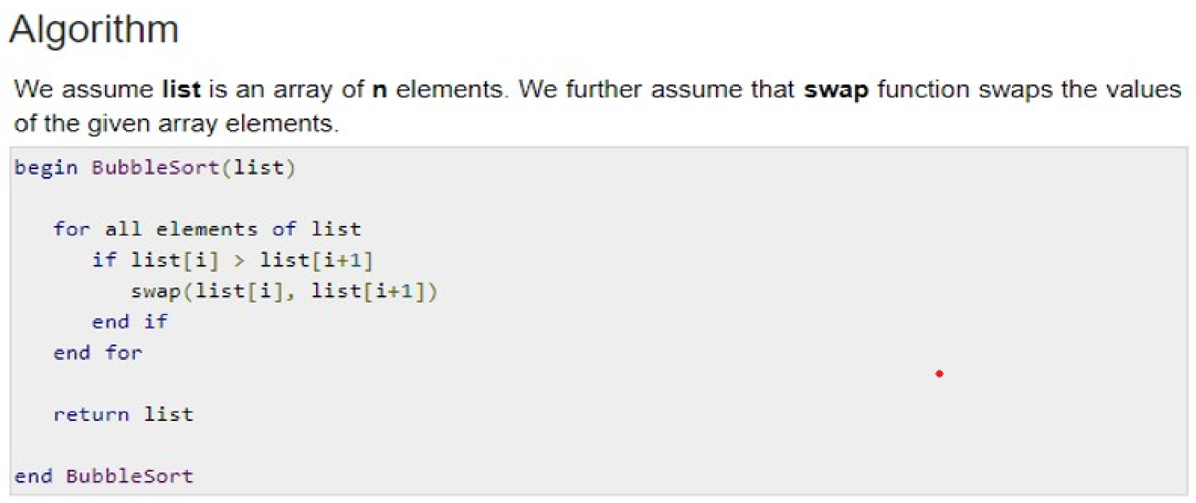
Step 4: Print element not found

Step 5: stop



Bubble Sort

- Bubble sort is a simple sorting algorithm.
- It is a comparison based algorithm in which each pair is compared and the elements are swapped if they are not in order.



Algorithm

We assume **list** is an array of **n** elements. We further assume that **swap** function swaps the values of the given array elements.

```
begin BubbleSort(list)

   for all elements of list
      if list[i] > list[i+1]
         swap(list[i], list[i+1])
      end if
   end for

   return list

end BubbleSort
```

Module 1 - University Questions

1.Compare and contrast various types of programming languages.

2.Draw the flowchart and develop the algorithm for finding the area of a triangle by reading three sides.

3.What are compilers and assemblers.

4.Differentiate between system software and application software?

5.Differentiate between Compiler and interpreter?

6.Explain different types of memory used in a computer?

7.Explain the bubble sort with an example. Draw a flowchart and write pseudocode to perform bubble sort on an array of numbers.

8.What do you mean by flowchart? what are its components? Explain any five components of a flowchart with example.

9.Write an algorithm to check whether the number is odd or even?

10.Construct the algorithm and flowchart for interchanging(swapping)the numeric value of two variables.

11.What are the properties of algorithm?

12.Draw a flowchart to find the sum of first 50 natural numbers.

13.Write an algorithm to convert temperature given in Celsius scale to Farenheit scale.

14.Write an algorithm to convert kilometer(km) into meter(m).

15.Explain the different hardware components of computer?

16.Write an algorithm and draw flowchart to find the average mark of one subject in a class of 50 students .

Samples of Algorithms and Flowchart

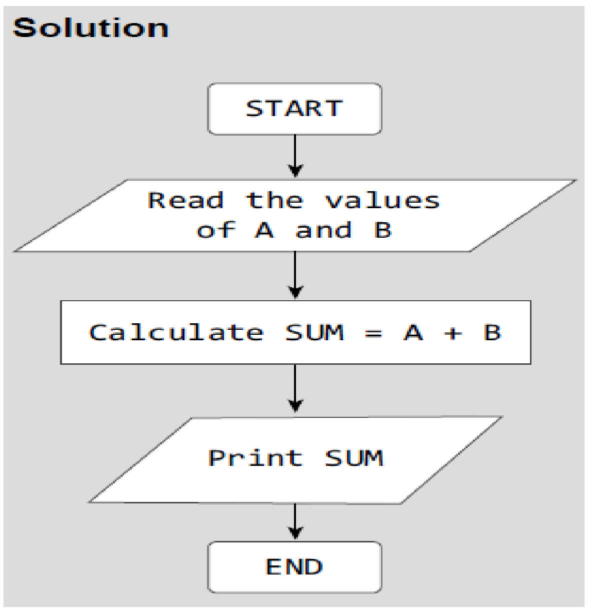
Step 1 : start

Step 2 : input the values of A and B

Step 3 : add A and B and assign to a variable sum

Step 4 : print sum

Step 5 : stop



Algorithm & Flowchart to find Area and Perimeter of Rectangle

L : Length of Rectangle
B : Breadth of Rectangle
AREA : Area of Rectangle
PERIMETER : Perimeter of Rectangle

Algorithm

Step-1 Start

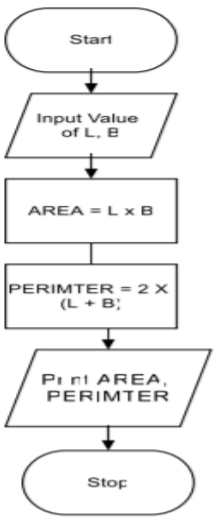
Step-2 Input Side Length & Breadth say L, B

Step-3 Area = L x B

Step-4 PERIMETER = 2 x (L + B)

Step-5 Display AREA, PERIMETER

Step-6 Stop



find the largest of two numbers

Step 1: Start

Step 2: Read two numbers a & b.

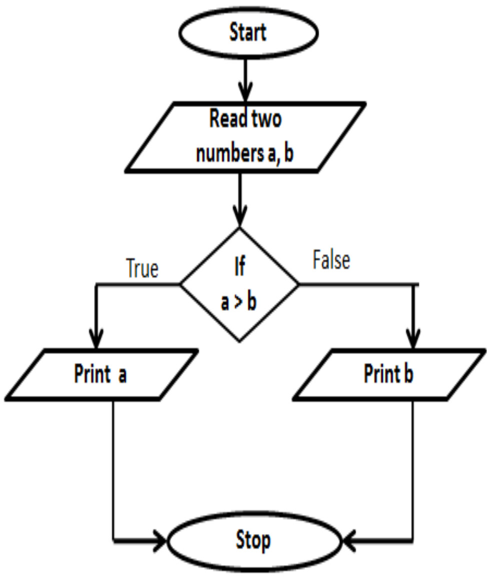
Step 3: If a>b then

Step 3.1 : Print largest number is a

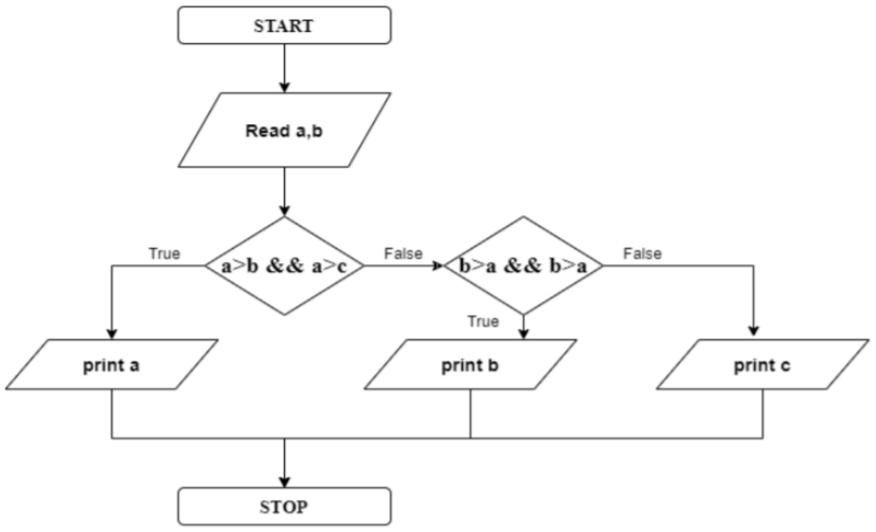
Step 3.2 : Go to step 5 Step 4: else

Step 4.1 : Print largest number is b

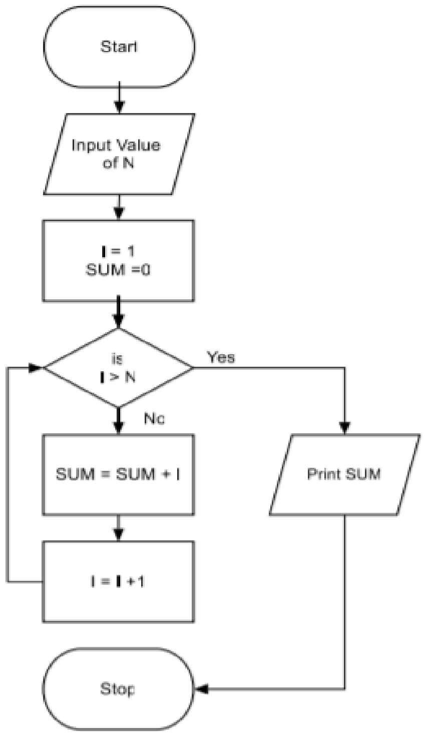
Step 5: Stop



3) Largest of three numbers



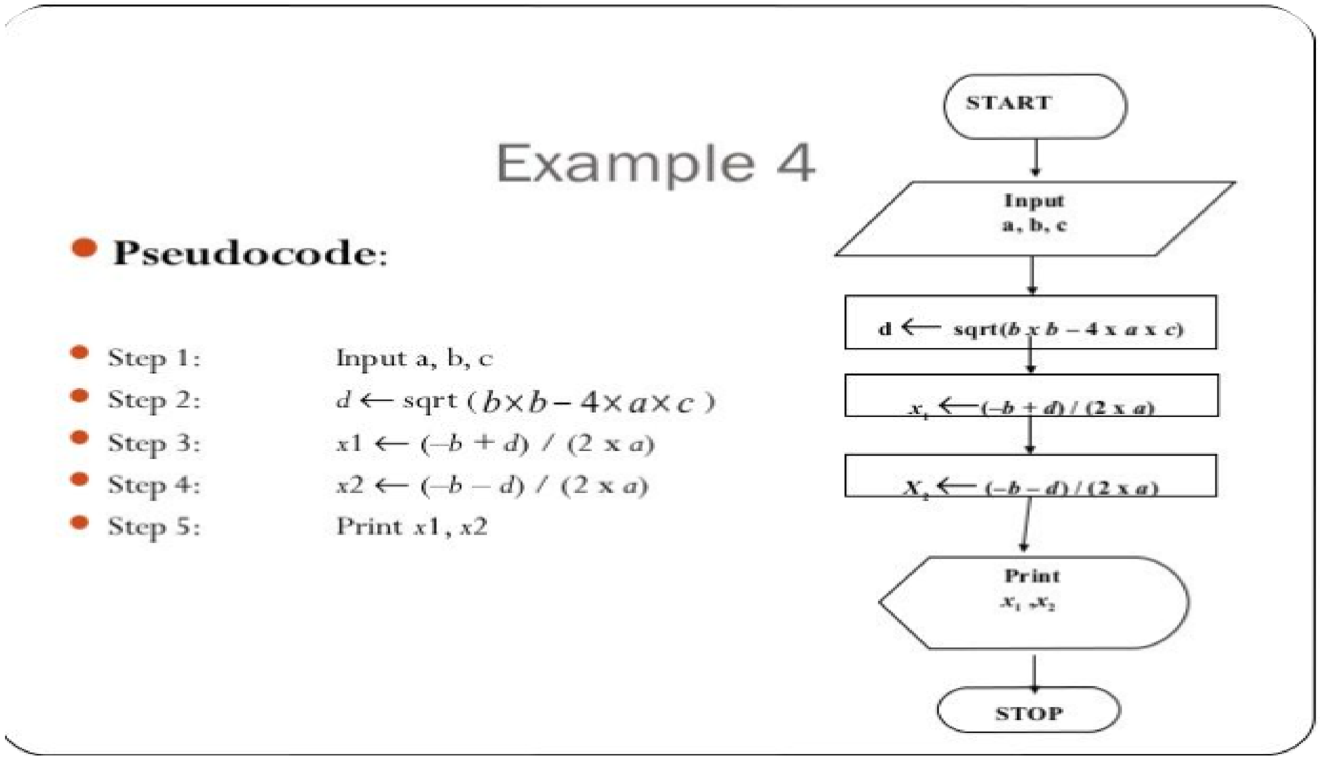
Algorithm & Flowchart to find sum of series 1+2+3+.....+N



Algorithm

```
Step-1   Start
Step-2   Input Value of N
Step-3     I = 1, SUM=0
Step-4     IF (I >N)  THEN
             GO TO Step-8
            ENDIF
Step-5     SUM = SUM + I
Step-6      I = I + 1
Step-7     Go to step-4
Step-8    Display value of SUM
Step-9    Stop
```

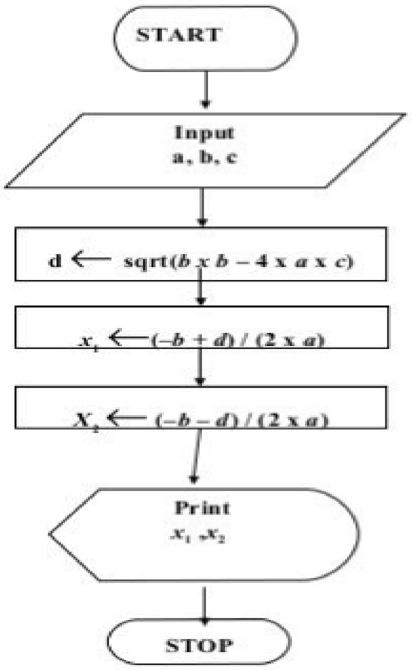
Pseudocode and flowchart to find the roots of a quadratic equation

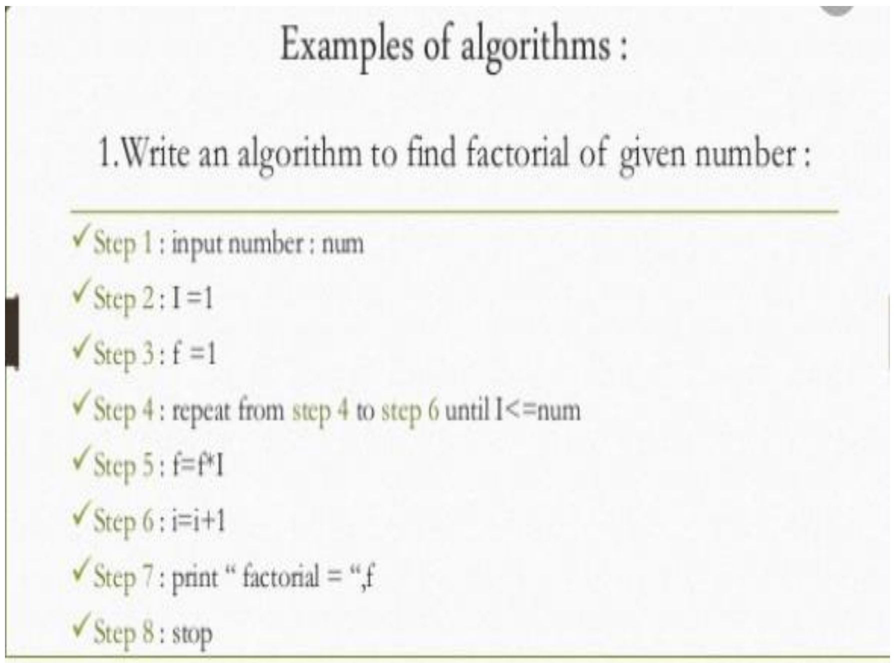


Example 4

- **Pseudocode:**

```
Step 1:    Input a, b, c
Step 2:    $d \leftarrow \text{sqrt}(b \times b - 4 \times a \times c)$
Step 3:    $x1 \leftarrow (-b + d) / (2 \times a)$
Step 4:    $x2 \leftarrow (-b - d) / (2 \times a)$
Step 5:    Print $x1, x2$
```


START
Input a, b, c
d ← sqrt(b x b – 4 x a x c)
x_1 ← (–b + d) / (2 x a)
X_2 ← (–b – d) / (2 x a)
Print x_1 ,x_2
STOP



Examples of algorithms :

1.Write an algorithm to find factorial of given number :

```
✓Step 1 : input number : num
✓Step 2 : I =1
✓Step 3 : f =1
✓Step 4 : repeat from step 4 to step 6 until I<=num
✓Step 5 : f=f*I
✓Step 6 : i=i+1
✓Step 7 : print " factorial = ",f
✓Step 8 : stop
```