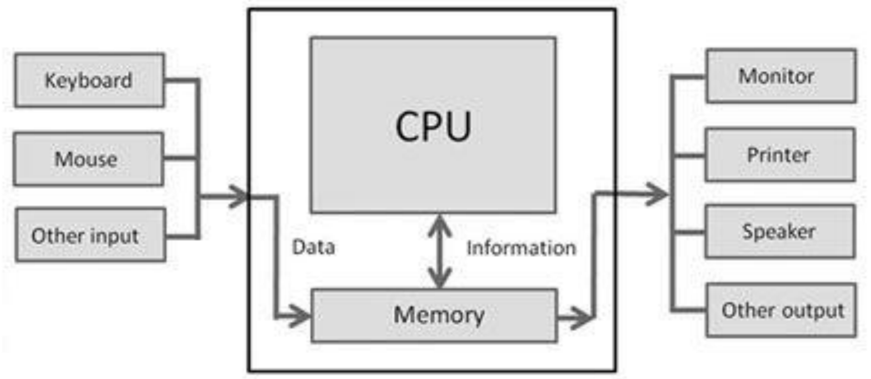
Module I

Basic of Computer Hardware and Softwares

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



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

Central Processor 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.

- 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.

Input /Output Unit

The input/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.

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 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.

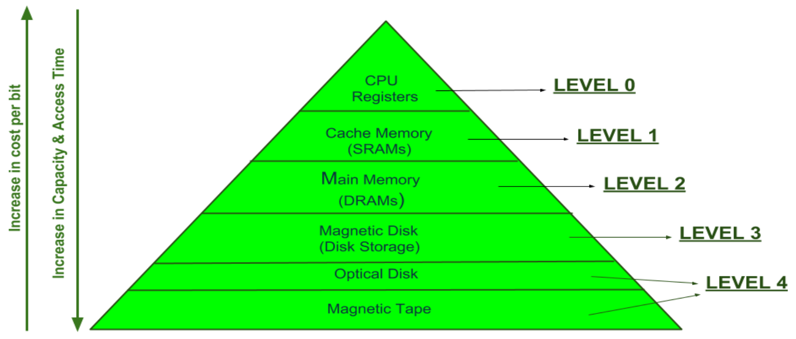
Secondary Memory

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, Magnetic Tapes, Pen drive

Memory Hierarchy (Out of Scope)

In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behavior known as locality of references. The figure below clearly demonstrates the different levels of memory hierarchy:



MEMORY HIERARCHY DESIGN

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.

S.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

Types of Languages

Computer programs can be written in high and low level languages, depending on the task and the hardware being used.

High Level Language

High level languages are written in a form that is close to human language, enabling to programmer to just focus on the problem being solved.

Advantages

- Easier to modify as it uses English like statements
- Easier/faster to write code as it uses English like statements
- Easier to debug during development due to English like statements
- Portable code – not designed to run on just one type of machine

Example: C, C++, Java, python

Low Level Language

Low level languages are used to write programs that relate to the specific architecture and hardware of a particular type of computer.

They are closer to the native language of a computer (binary), making them harder for programmers to understand.

Example: Machine Code

Assembly Language

An assembly language is a low-level programming language designed for a specific type of processor.

Example: 8085 programing

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.

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.

Example: Algorithm to find area of circle

1. Read the value of the radius
2. Calculate the area of circle
3. Print the area of circle.

Pseudo Code

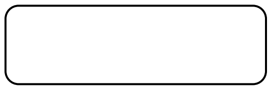
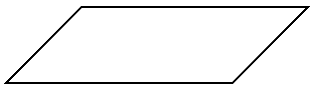
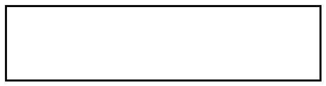
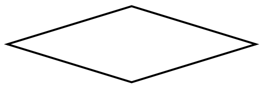
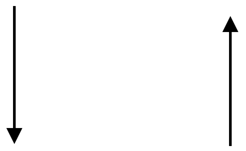
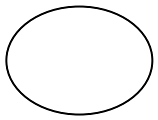
It is a more formal representation than the algorithm. Here, we represent every stp in a formal way which is very close to the actual programming language representation. In pseudocode, each of the steps will be written via operator and statements equivalent to some programming language instructions. The only difference will be that the exact syntax of the programming language will not be followed. All pseudocodes will start with the keyword "START" and complete with keyword "STOP" or "END".

Example: Pseudocode to find area of circle

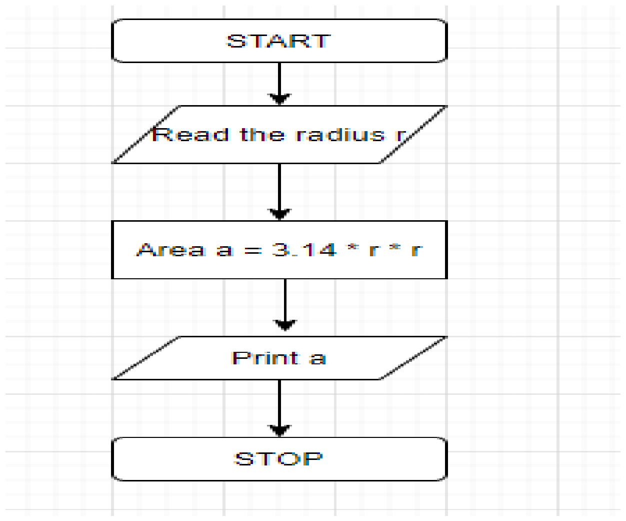
```
1. START
2. Read radius
3. area = 3.14 * radius * radius
4. print sum
5. STOP
```

Flowchart

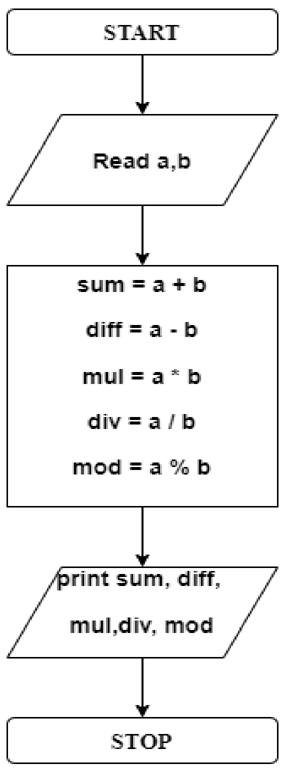
Very popular method to represent the steps of the solution is the flowchart, which uses many graphical symbols and thus, is more understandable. The symbol used for various different types of statements are as shown

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

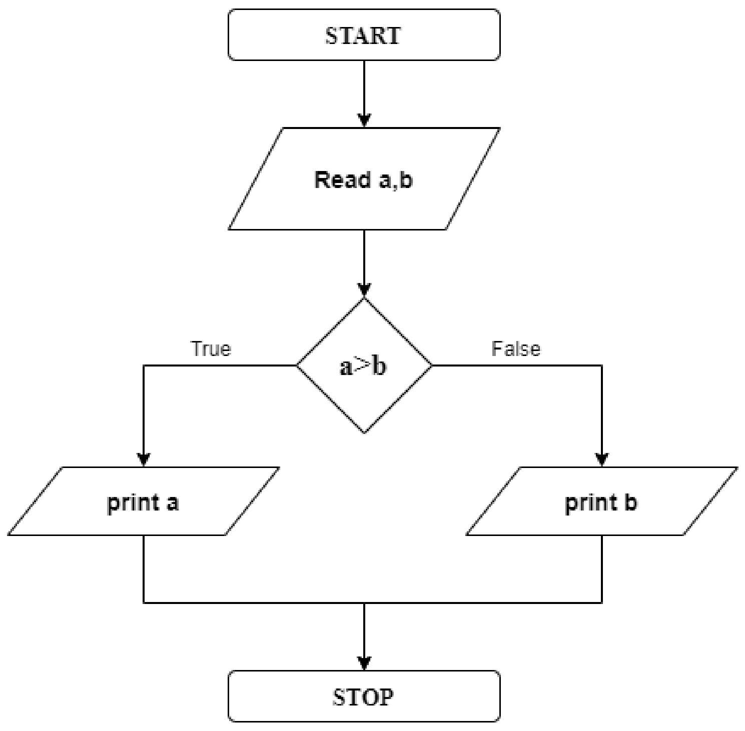
Example: Flowchart for finding area of circle



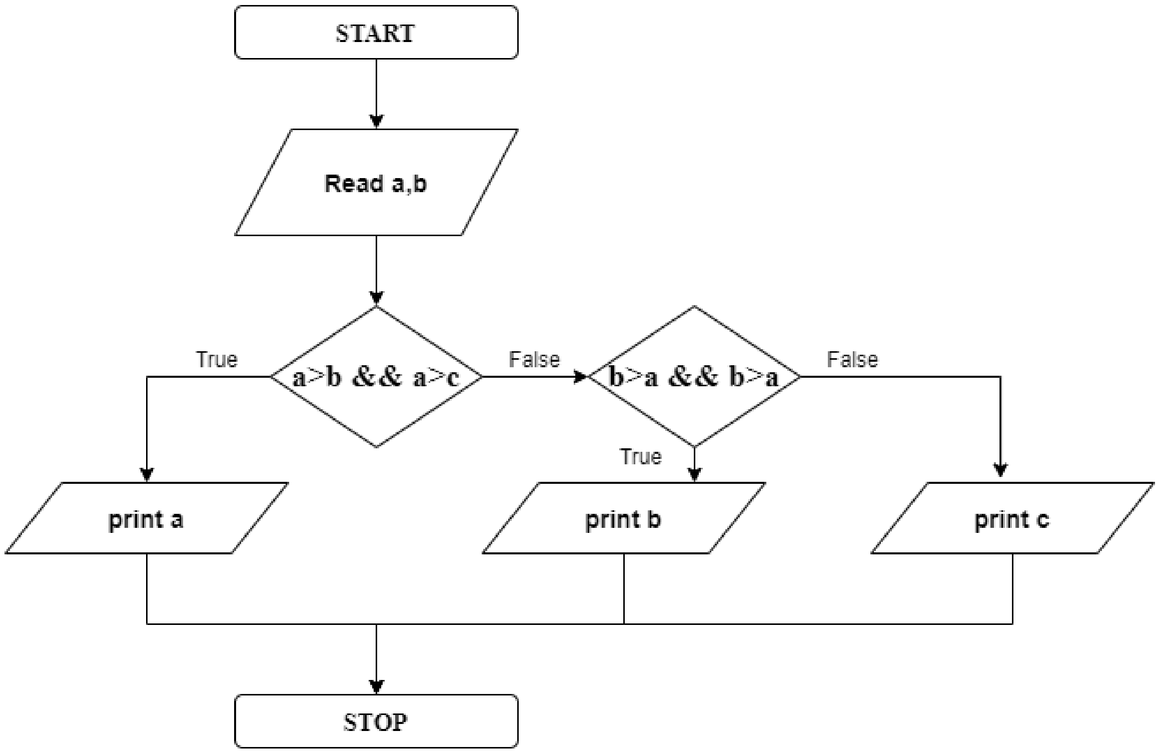
1) Arithmetic Operation



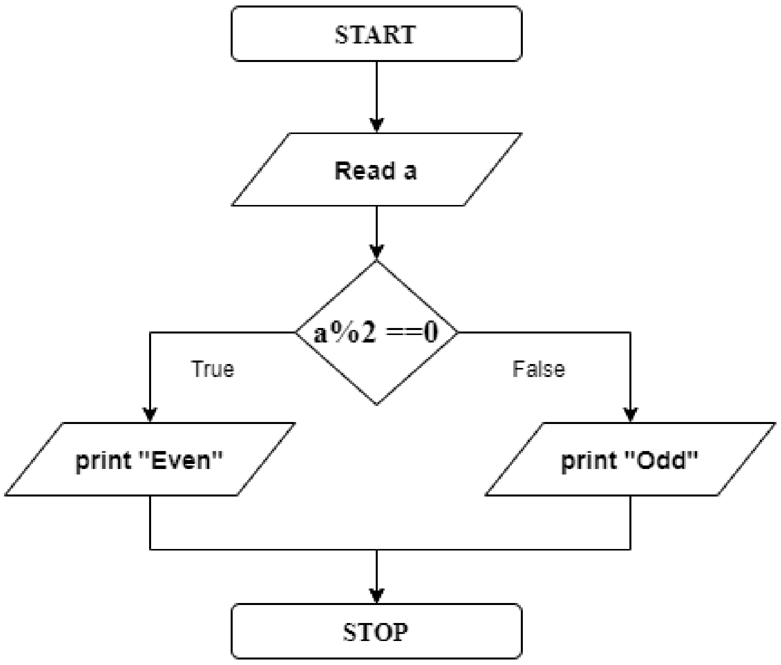
2) Largest of two numbers



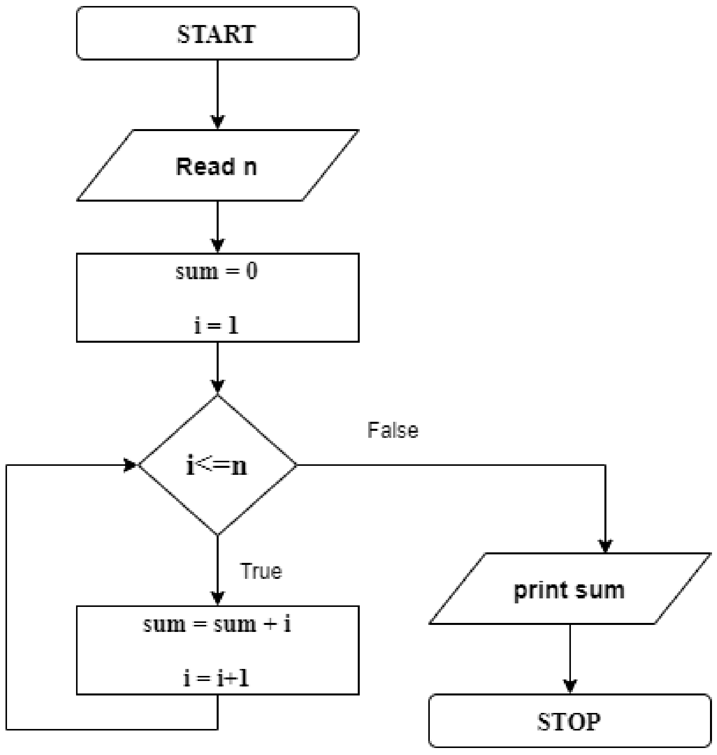
3) Largest of three numbers



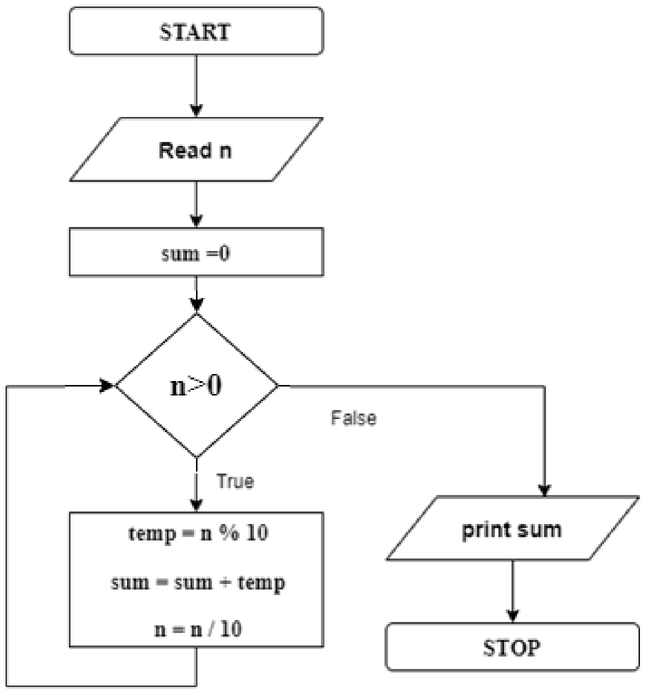
4) Odd or even



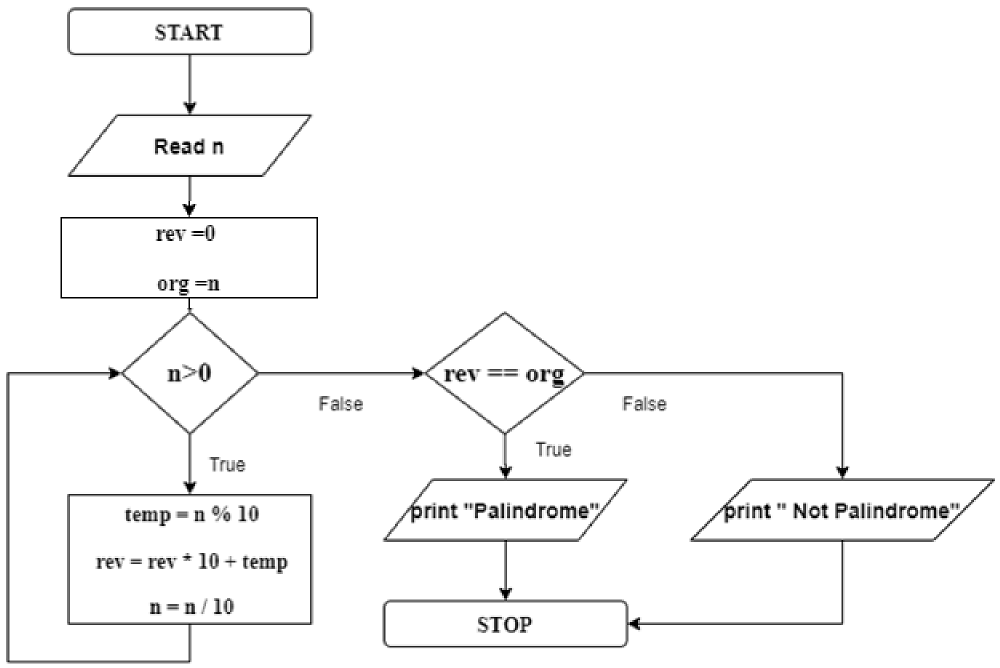
5) Sum of first n numbers



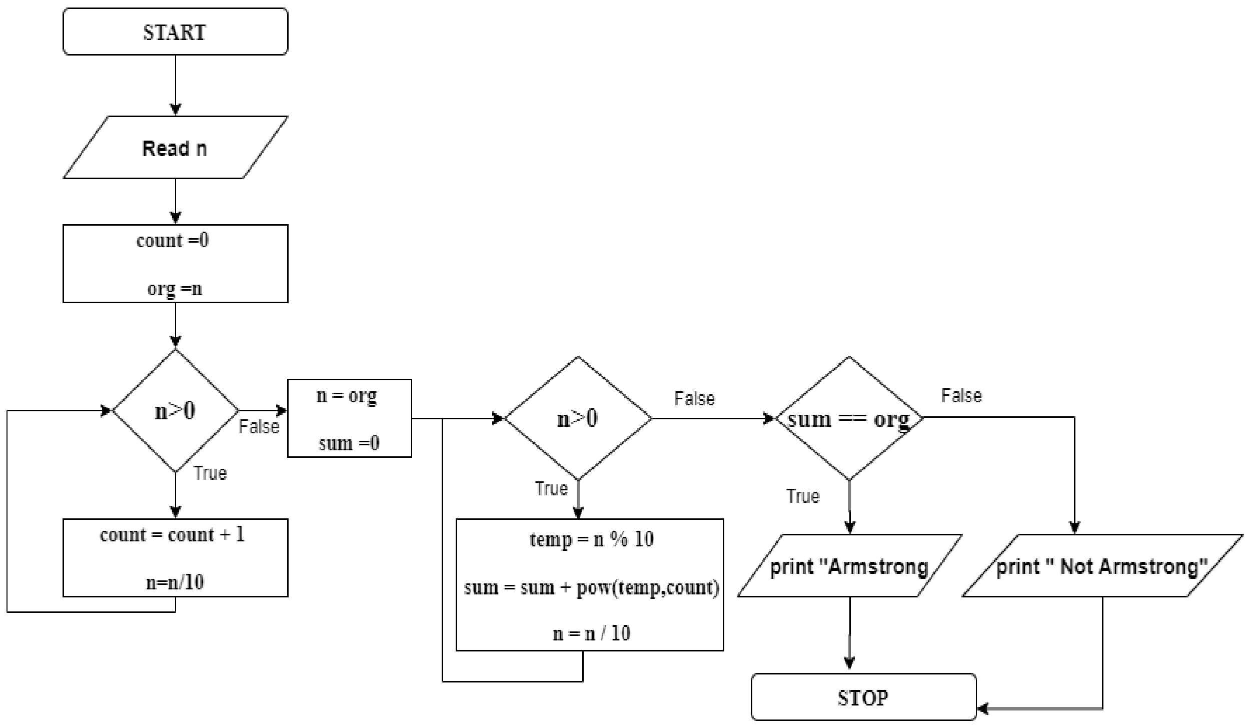
6) Sum of digit of a number



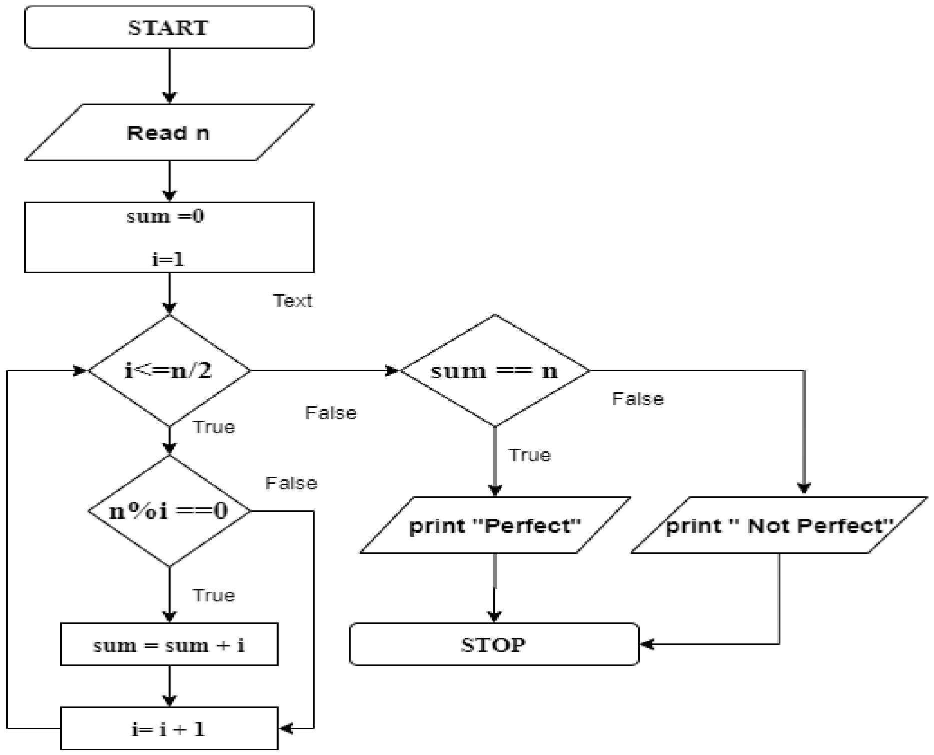
7) Check whether a number is palindrome or not



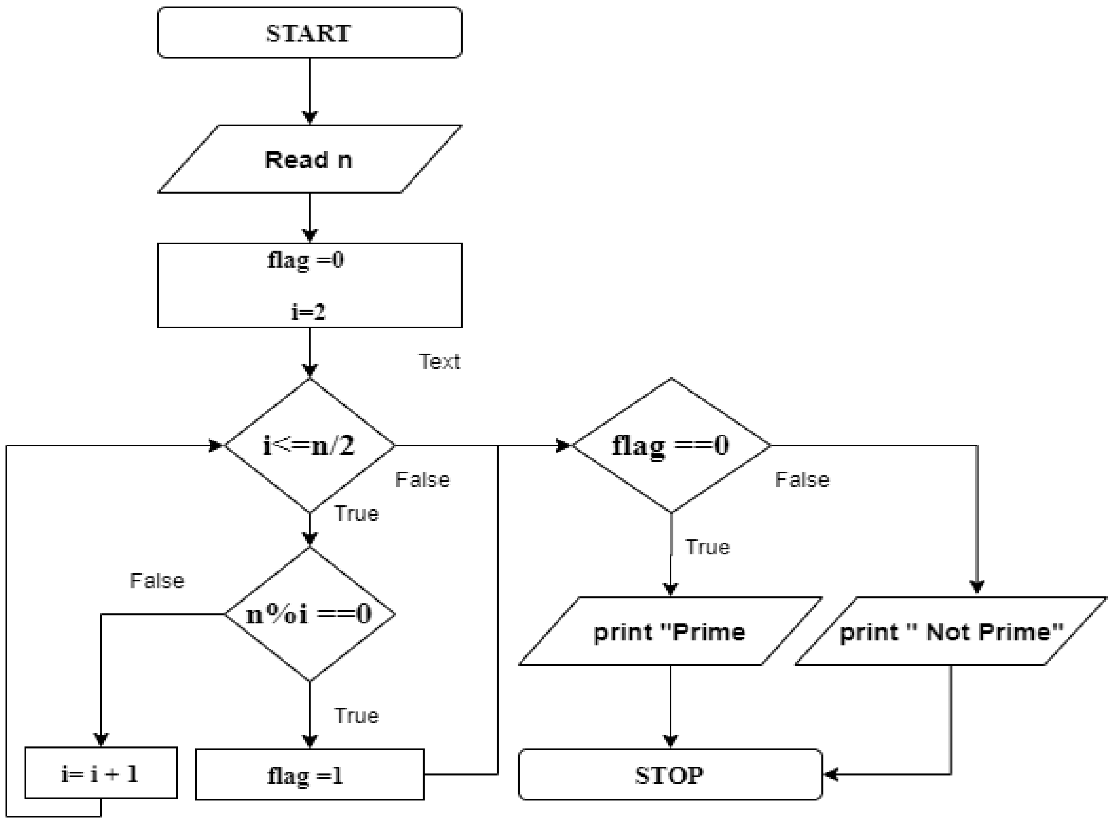
8) Check whether a number is Armstrong or not.



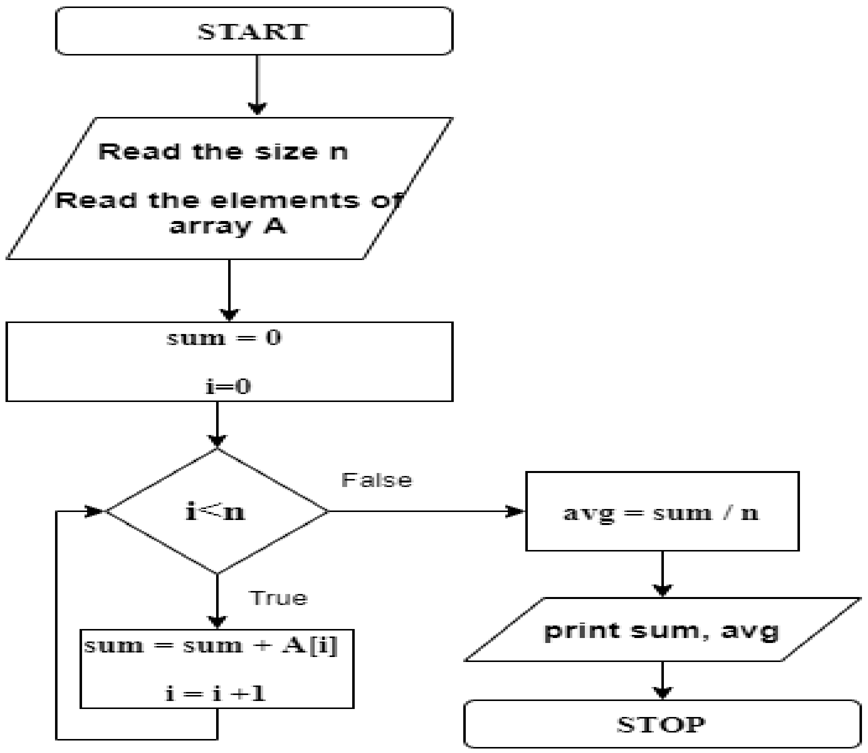
9) Check whether a number is perfect or not



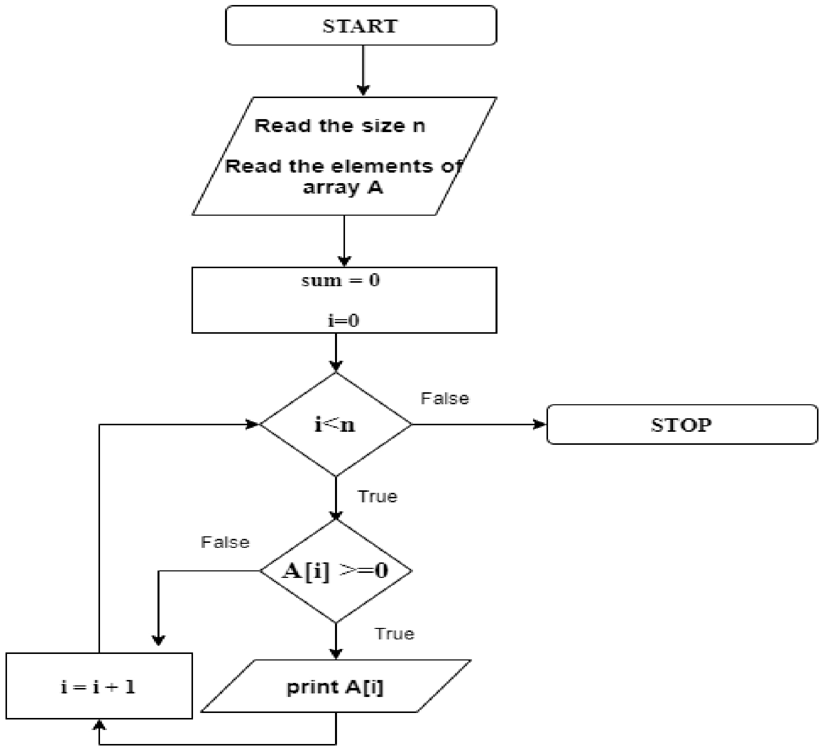
10) Check whether a number is prime or not.



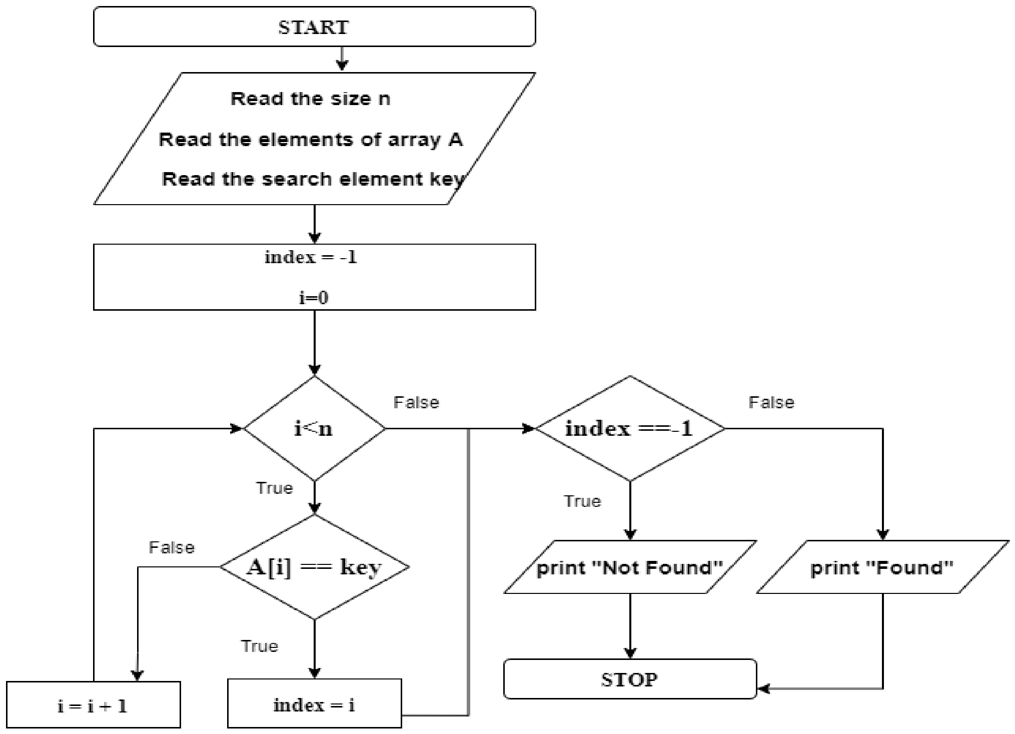
11) Sum and average of an array



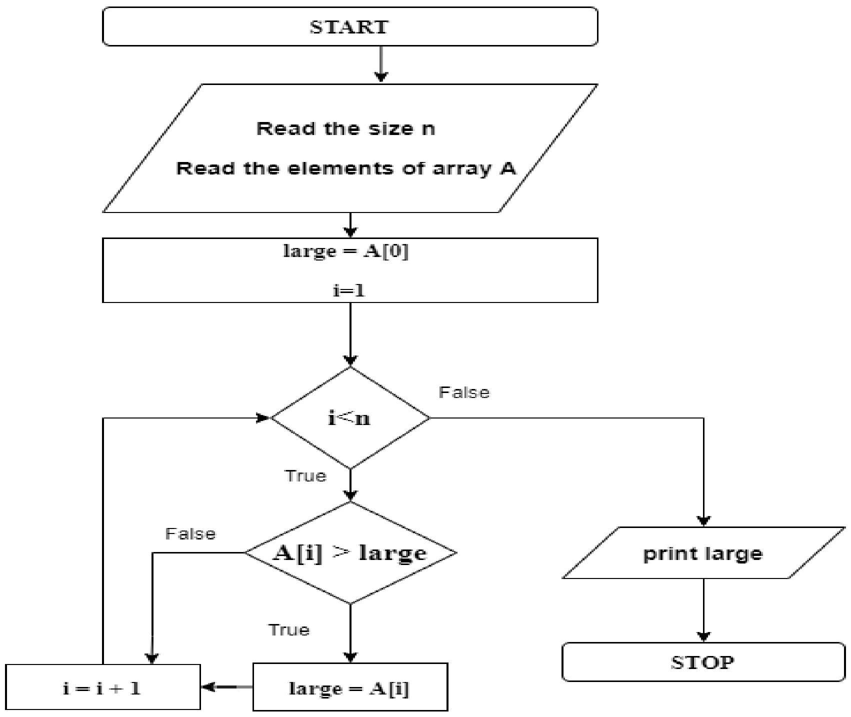
12) Display non – negatives numbers in an array.



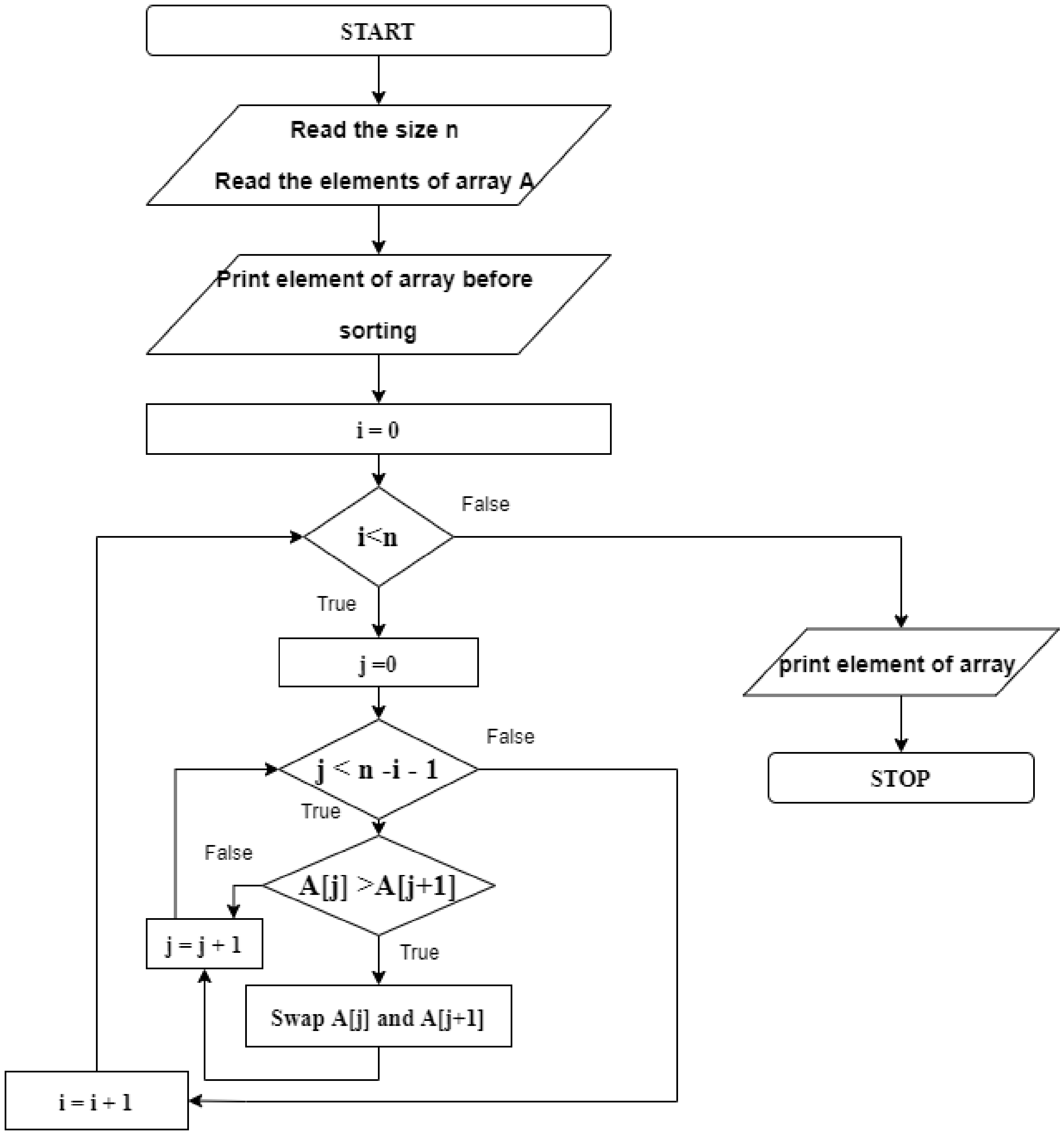
13) Linear Search



14) Largest element in an array



15) Bubble Sort


START
Read the size n
Read the elements of array A
Print element of array before sorting
i = 0
i<n
False
True
j =0
j < n -i - 1
False
True
A[j] >A[j+1]
False
True
j = j + 1
Swap A[j] and A[j+1]
i = i + 1
print element of array
STOP