SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMMING FOR PROBLEM SOLVING
I YEAR - I SEM

UNIT 1 – Introduction to Problem Solving Techniques

TOPIC 5 – Notation [pseudo code, flow chart, and programming language]

NOTATIONS OF AN ALGORITHM

➤ Algorithm can be expressed in many different notations, including Natural Language, Pseudo code, flowcharts and programming languages.

➤ Natural language tends to be verbose and ambiguous.

➤ Pseudocode and flowcharts are represented through structured human language.

➤ A notation is a system of characters, expressions, graphics or symbols designs used among each others in problem solving to represent technical facts, created to facilitate the best result for a program

➤ In simple words Notations collectively represents the following:

- Pseudo code
- Flowcharts
- Programming languages.

PSEUDOCODE

➢ Pseudocode is an informal high-level description of the operating principle of a computer program or algorithm.

➢ It uses the basic structure of a normal programming language, but is intended for human reading rather than machine reading.

➢ It is text based detail design tool.

➢ Pseudo means 'false' and code refers to 'instructions' written in programming language.

➢ Pseudocode cannot be compiled nor executed, and there are no real formatting or syntax rules.

➢ The pseudocode is written in normal English language which cannot be understood by the computer.

Example:

Pseudocode: To find sum of two numbers

READ num1,num2

sum=num1+num2

PRINT sum

BASIC RULES TO WRITE PSEUDOCODE

1. Only one statement per line.
   Statements represents single action is written on same line. For example to read the input, all the inputs must be read using single statement.

2. Capitalized initial keywords
   The keywords should be written in capital letters.
   Eg: READ, WRITE, IF, ELSE, ENDIF, WHILE, REPEAT

3. Indent to show hierarchy
   Indentation is a process of showing the boundaries of the structure.

4. End multi-line structures
   Each structure must be ended properly, which provides more clarity.

5. Keep statements language independent.
   Pesudocode must never written or use any syntax of any programming language.

Example: 01
Pseudocode: Find the total and average of three subjects

READ name, mark1, mark2, mark3
Total=mark1+mark2+mark3
Average=Total/3
WRITE name, mark1, mark2, mark3

Example: 02
Pseudocode: Find greatest of two numbers

READ a, b
IF a>b then
        PRINT a is greater
ELSE
        PRINT b is greater
ENDIF

ADVANTAGES & DISADVANTAGES OF PSEUDOCODE

➢Advantages of Pseudocode
- Can be done easily on a word processor
- Easily modified
- Implements structured concepts well
- It can be written easily
- It can be read and understood easily
- Converting pseudocode to programming language is easy as compared with flowchart

➢Disadvantages of Pseudocode
- It is not visual
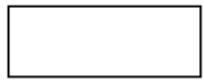- There is no standardized style or format
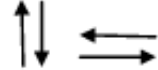
COMMON KEYWORDS USED IN PSEUDOCODE

1. //: This keyword used to represent a comment.
2. BEGIN,END: Begin is the first statement and end is the last statement.
3. INPUT, GET, READ: The keyword is used to inputting data.
4. COMPUTE, CALCULATE: used for calculation of the result of the given expression.
5. ADD, SUBTRACT, INITIALIZE: used for addition, subtraction and initialization.
6. OUTPUT, PRINT, DISPLAY: It is used to display the output of the program.
7. IF, ELSE, ENDIF: used to make decision.
8. WHILE, ENDWHILE: used for iterative statements.
9. FOR, ENDFOR: Another iterative incremented/decremented tested automatically.

FLOWCHART

➢ A graphical representation of an algorithm.

➢ Flowcharts is a diagram made up of boxes, diamonds, and other shapes, connected by arrows.

➢ Each shape represents a step in process and arrows show the order in which they occur.

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)
	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 SYMBOLS

Name	Symbol	Description
Process		Process or action step
Flow line	→	Direction of process flow
Start/ terminator		Start or end point of process flow
Decision	◇	Represents a decision making point
Connector	○	Inspection point
Inventory	▽	Raw material storage
Inventory	△	Finished goods storage
Preparation		Initial setup and other preparation steps before start of process flow
Alternate process		Shows a flow which is an alternative to normal flow
Flow line(dashed)	– – →	Alternate flow direction of information flow

Rules for drawing flowchart

1. In drawing a proper flowchart, all necessary requirements should be listed out in logical order.

2. Flow chart should be clear, neat and easy to follow. There should not be any room for ambiguity in understanding the flowchart.

3. The usual directions of the flow of a procedure or system is from left to right or top to bottom.

 Only one flow line should come out from a process symbol.

4. Only one flow line should enter a decision symbol, but two or three flow lines, one for each possible answer, cap leave the decision symbol.
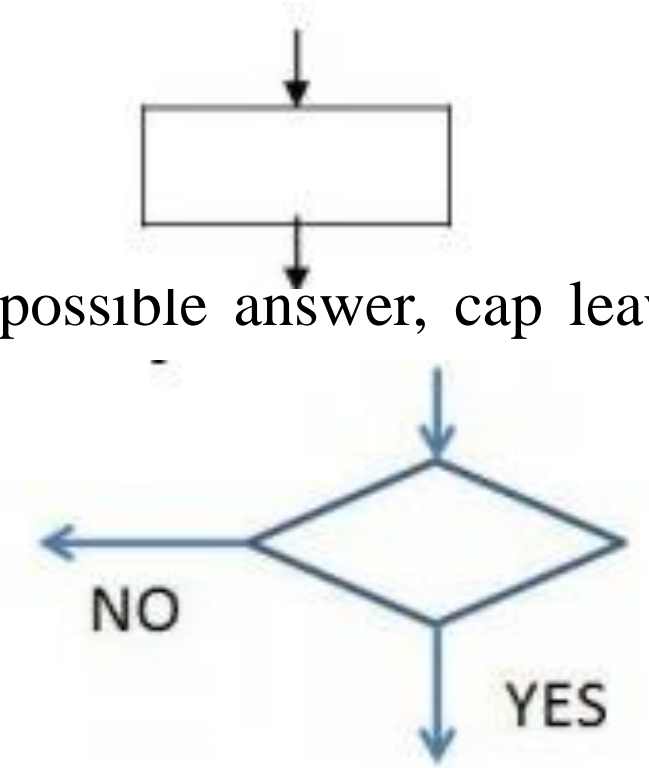
5. Only one flow line is used in conjunction with terminal symbol.
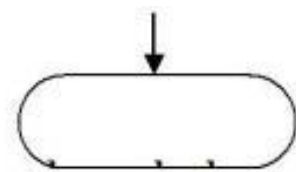
6. If flowchart becomes complex, it is better to use connector symbols to reduce the number of flow lines.

7. Ensure that flowchart has logical start and stop.

ADVANTAGES & DISADVANTAGES OF FLOWCHART

Advantages of Flowchart:

Communication:

   Flowcharts are better way of communicating the logic of the system.

Effective Analysis

   With the help of flowchart, a problem can be analyzed in more effective way.

Proper Documentation

   Flowcharts are used for good program documentation, which is needed for various purposes.

Efficient Coding

   The flowcharts act as a guide or blue print during the system analysis and program development phase.

Systematic Testing and Debugging

   The flowchart helps in testing and debugging the program

Efficient Program Maintenance

   The maintenance of operating program becomes easy with the help of flowchart.

   It helps the programmer to put efforts more efficiently on that part.

Disadvantages of Flowchart

Complex Logic:

   Sometimes, the program logic is quite complicated. In that case flowchart becomes complex and difficult to use.

Alteration and Modification:

   If alterations are required the flowchart may require redrawing completely.

Reproduction: As the flowchart symbols cannot be typed, reproduction becomes problematic.

➢ **Sequence Structure**
- A sequence is a series of steps that take place one after another.
- Each step is represented here by a new line

Pseudocode	Flow Chart
General Structure	
Process 1 …. Process 2 … Process 3	
Example	
READ a READ b Result c=a+b PRINT c	

Conditional Structure

➢ Conditional structure is used to check the condition.

➢ It will be having two outputs only (True or False)

➢ **IF** and **IF…ELSE** are the conditional structures used.

Pseudocode	Flow Chart
General Structure	

IF condition THEN
    Process 1
ENDIF	

Example	
READ a READ b IF a>b THEN PRINT a is greater	

CONDITIONAL STRUCTURE

IF… ELSE

IF…THEN…ELSE is the structure used to specify, if the condition is true, then execute Process1, else, that is condition is false then execute Process2

READ a

READ b

IF a>b THEN

PRINT a is greater

Pseudocode	Flow Chart
General Structure	

IF condition THEN
    Process 1
ELSE
    Process 2
ENDIF	

Example	
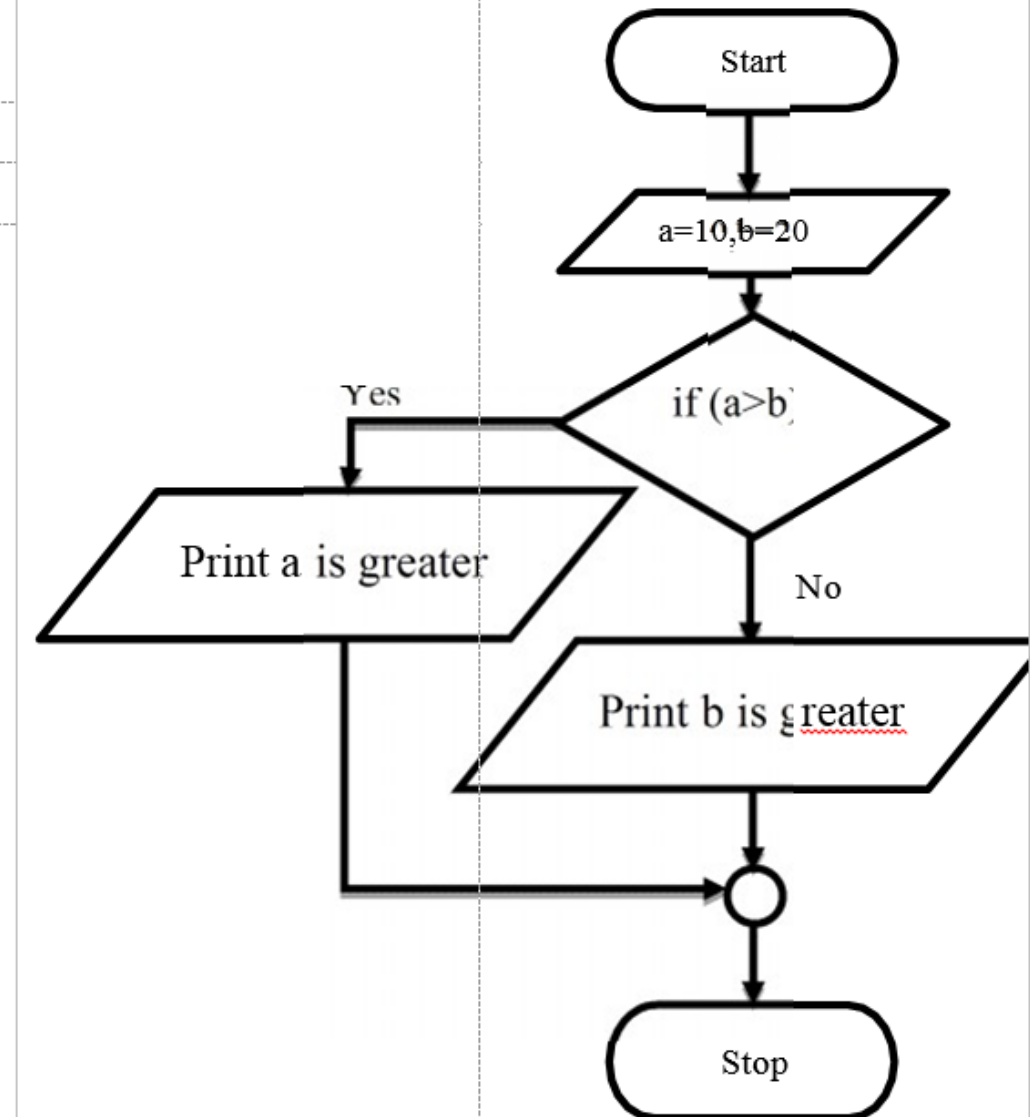
ITERATION OR LOOPING STRUCTURE

- ➤ Looping is generally used with WHILE or DO...WHILE or FOR loop.
- ➤ WHILE and FOR is **entry checked loop.**
- ➤ DO…WHILE is exit checked loop, so the loop will be executed at least once.

```
INITIALIZE a=1
WHILE a<10 THEN
        PRINT a
        a=a+1
ENDWHILE
```


Pseudocode	Flow Chart
General Structure	
WHILE condition Body of the loop ENDWHILE	
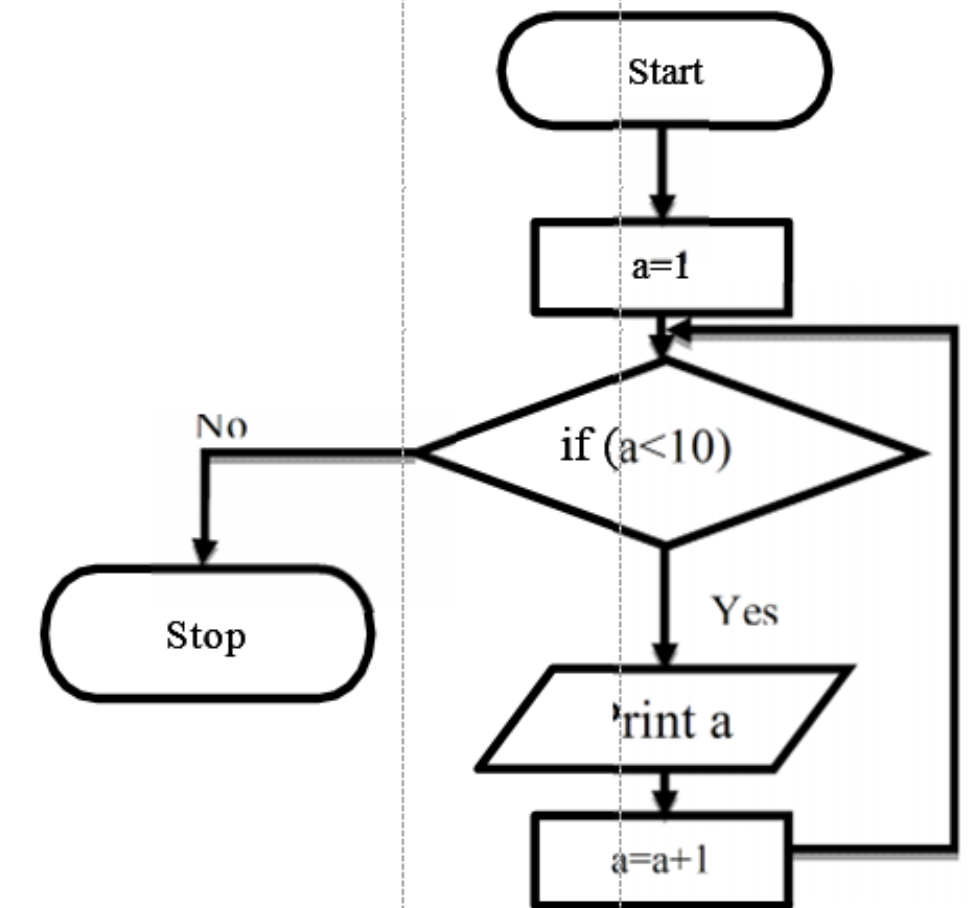
ALGORITHM vs. FLOWCHART vs. PSEUDOCODE

Algorithm	Flowchart	Pseudo code
An algorithm is a sequence of instructions used to solve a problem	It is a graphical representation of algorithm	It is a language representation of algorithm.
User needs knowledge to write algorithm.	not need knowledge of program to draw or understand flowchart	Not need knowledge of program language to understand or write a pseudo code.

PROGRAMMING LANGUAGE

➢ A programming language is a vocabulary and set of <u>grammatical rules</u> for instructing a computer or computing device to perform specific tasks.

➢ In other word it is <u>set of instructions</u> for the computer to solve the problem.

➢ Programming Language is a formal language with set of instruction, to the computer to solve a problem.

➢ The program will accept the data to perform computation.

➢ The programmers have to follow all the specified rules before writing program using programming language.

➢ The user has to communicate with the computer using language which it can understand.

Program= Algorithm + Data

NEED & TYPES OF PROGRAMMING LANGUAGES

➢ Need for Programming Languages
- Programming languages are also used to organize the computation.
- Using Programming language we can solve different problems.
- To improve the efficiency of the programs.

➢ Types of Programming Language

➢ In general Programming languages are classified into three types. They are
- Low – level or Machine Language
- Intermediate or Assembly Language
- High – level Programming language

TYPES OF LANGUAGES

➢ **Machine Language:**
- Machine language is the lowest-level programming language.
- Machine languages are the only languages understood by computers.
- It is also called as low level language.
  - » Example code:100110011
  - » 111001100

➢ **Advantages:**
  ➢ Translation free:
  - Machine language is the only language which the computer understands.
  - For executing any program written in any programming language, the conversion to machine language is necessary.
  - The program written in machine language can be executed directly on computer.
  - In this case any conversion process is not required.
  ➢ High speed:
  - The machine language program is translation free.
  - Since the conversion time is saved, the execution of machine language program is extremely fast.

➢ **Disadvantages:**
  - It is hard to find errors in a program written in the machine language.
  - Writing program in machine language is a time consuming process.

TYPES OF LANGUAGES

➢ Assembly Language:

- To overcome the issues in programming language and make the programming process easier, an assembly language is developed which is logically equivalent to machine language but it is <u>easier for people to read, write and understand</u>.
- Assembly language is symbolic representation of machine language.
- Assembly languages are symbolic programming language that uses symbolic notation to represent machine language instructions.
- They are called low level language because they are so closely related to the machines.
- An assembly language contains the same instructions as a machine language, but the <u>instructions and variables have names</u> instead of being just numbers.
- An assembly language consists of mnemonics, mnemonics that corresponds unique machine instruction.
  » Example code: start
    » Add x,y
    » Sub x,y

TYPES OF LANGUAGES

➢ Assembler:
➢ Assembler is the program which translates assembly language instruction in to a machine language.
  • Easy to understand and use.
  • It is easy to locate and correct errors.

➢ Disadvantages:
➢ Machine dependent:
  • The assembly language program which can be executed on the machine depends on the architecture of that computer.
➢ Hard to learn:
  • It is machine dependent, so the programmer should have the hardware knowledge to create applications using assembly language.
➢ Less efficient :
  • Execution time of assembly language program is more than machine language program.
  • Because assembler is needed to convert from assembly language to machine language.
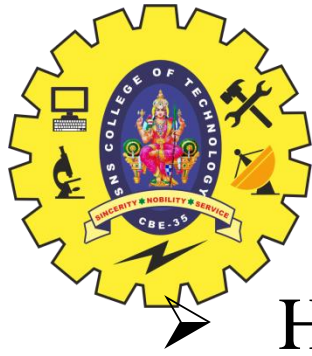
➢ <u>High – level Language:</u>

- High level language contains English words and symbols.
- The specified rules are to be followed while writing program in high level language.
- The **<u>interpreter or compilers</u>** are used for converting these programs in to machine readable form.
- A high-level language (HLL) is a programming language such as C, FORTRAN, or Pascal that enables a programmer to write programs that are more or less independent of a particular type of computer.
- Such languages are considered high-level because they are closer to human languages and further from machine languages.
- Ultimately, programs written in a high-level language must be translated into machine language by a compiler or interpreter.
  - » Example code: print("Hello World!")

➢ Translating high level language to machine language:

- ▪ The programs that translate high level language in to machine language are called <u>interpreter or compiler.</u>

TYPES OF LANGUAGES (Cont..)

➢ **Compiler:**
- A compiler is a program which translates the source code written in a high level language in to object code which is in machine language program.
- Compiler reads the whole program written in high level language and translates it to machine language.
- If any error is found it display error message on the screen.

➢ **Interpreter**
- Interpreter translates the high level language program in line by line manner.
- The interpreter translates a high level language statement in a source program to a machine code and executes it immediately before translating the next statement.
- When an error is found the execution of the program is halted and error message is displayed on the screen.

➢ **Advantages**
- Readability:
  – High level language is closer to natural language so they are easier to learn and understand.
- Machine independent:
  – High level language program have the advantage of being portable between machines.
- Easy debugging:
  – Easy to find and correct error in high level language

➢ **Disadvantages:**
- Less efficient:
  – The translation process increases the execution time of the program.
  – Programs in high level language require more memory and take more execution time to execute.

➢ High level programming languages are further divided and shown in the Table.

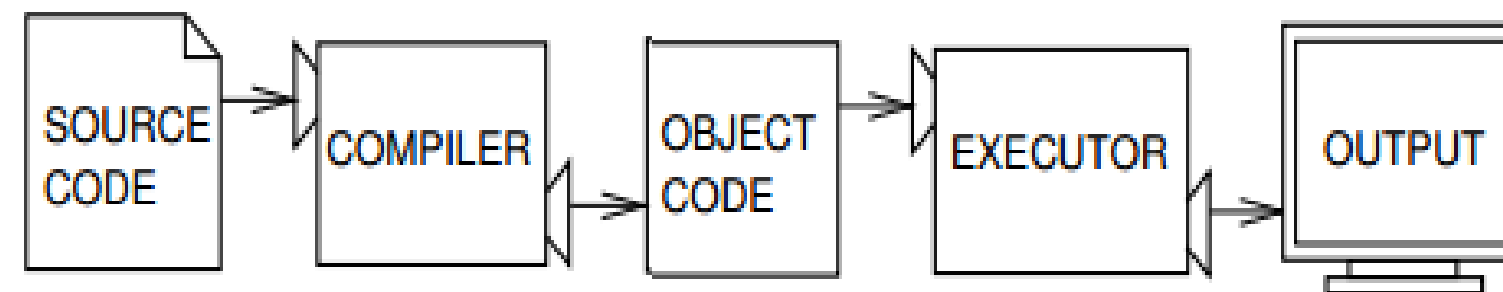| Language Type	Example
Interpreted Programming Language	Python, BASIC, Lisp
Functional Programming Language	Clean, Curry, F#
Compiled Programming Language	C++,Java, Ada, ALGOL
Procedural Programming Language	C,Matlab, CList
Scripting Programming Language	PHP,Apple Script, Javascript
Markup Programming Language	HTML,SGML,XML
Logical Programming Language	Prolog, Fril
Concurrent Programming Language	ABCL, Concurrent PASCAL
Object Oriented Programming Language	C++,Ada, Java, Python

TYPES OF LANGUAGES (Cont..)

➢ Interpreted Programming Language:

- Interpreter is a program that executes instructions written in a high-level language.
- An interpreter reads the source code one instruction or one line at a time, converts this line into a machine code and executes it.
- Ex Pascal, Python

SOURCE CODE → INTERPRETER → OUTPUT

➢ Compiled Programming Language:

- Compile is to transform a program written in a high-level programming language from source code into object code.
- This can be done by using a tool called compiler.
- A compiler reads the whole source code and translates it into a complete machine code program to perform the required tasks which is output as a new file. Ex: C, C++, JAVA

SOURCE CODE → COMPILER → OBJECT CODE → EXECUTOR → OUTPUT

Interpreted Programming Language	Compile Programming Language
Translates one statement at a time	Scans entire program and translates it as whole into machine code
It takes less amount of time to analyze the source code but the overall execution time is slower	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster
No intermediate object code is generated, hence are memory efficient	Generates intermediate object code which further requires linking, hence requires more memory
Continues translating the program until first error is met, in which case it stops. Hence Debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Eg: Python, Ruby	Eg: C,C++,Java