

Flow Chart and Phases of Making an Executable Code

Drawing Flowcharts

- Flowchart is the graphic representations of the individual steps or actions to implement a particular module
- Flowchart can be likened to the blueprint of a building
- An architect draws a blueprint before beginning construction on a building, so the programmer draws a flowchart before writing a program
- Flowchart is independent of any programming language.

Flowchart Types



Basic Flowchart



Business Process Modeling Notation



Cross Functional Horizontal



Cross Functional Vertical



Data Flow Diagram



IDEF Diagram



Highlight Flowchart



List and Process



Work Flow Diagram



SDL Diagram

Flow Charts

- A flow chart is an organized combination of shapes, lines and text that graphically illustrate a process or structure.

Symbols used



Start/Stop



Process



Input/Output (Data)



Flow Lines

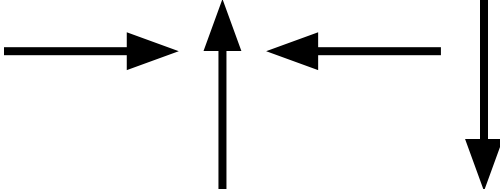
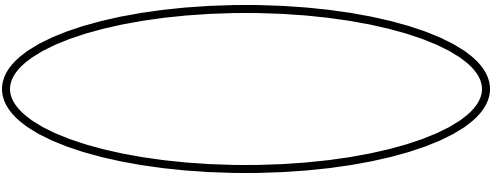



Decision symbol

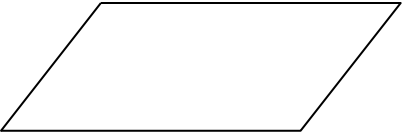
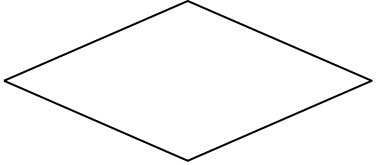

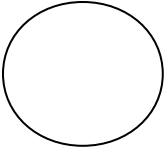
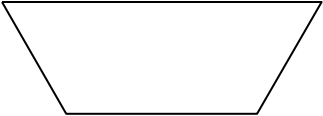


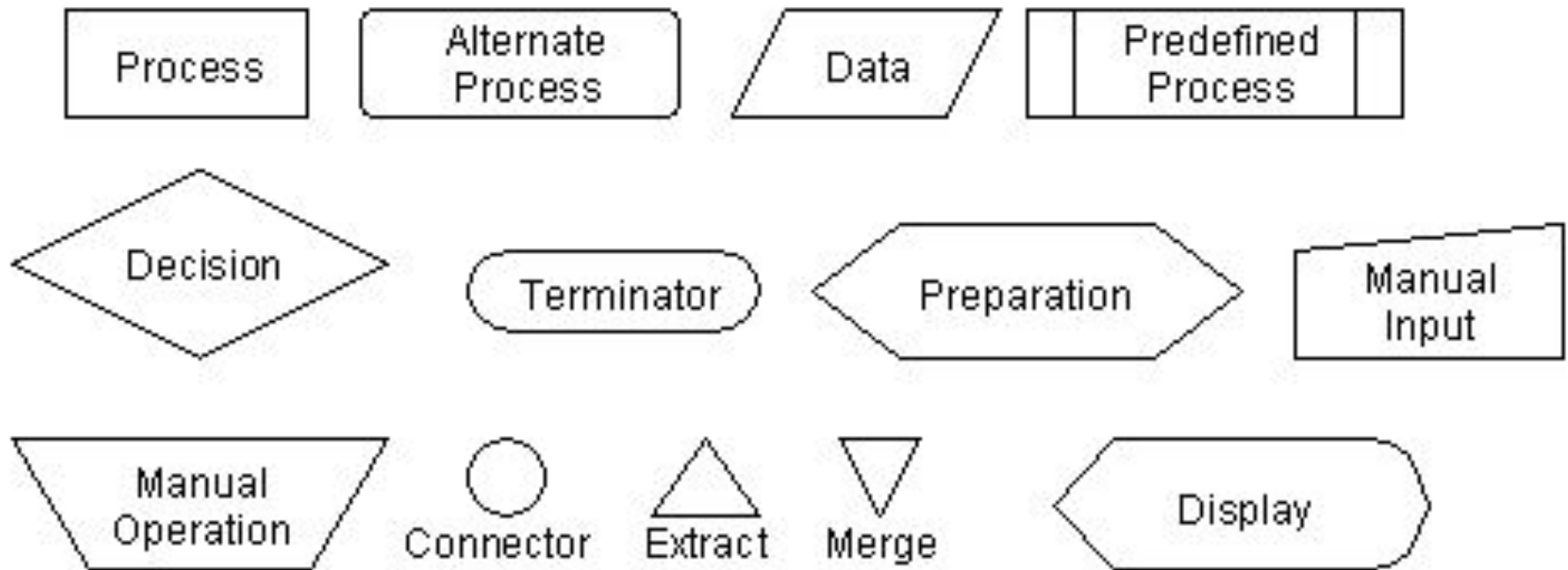
Connector

PRE-PROGRAMMING PHASE

Symbol	Function
	Show the direction of data flow or logical solution.
	Indicate the beginning and ending of a set of actions or instructions (logical flow) of a module or program.
	Indicate a process, such as calculations, opening and closing files.

PRE-PROGRAMMING PHASE

	Indicate input to the program and output from the program.
	Use for making decision. Either True or False based on certain condition.
	Use for doing a repetition or looping of certain steps.
	Connection of flowchart on the same page.
	Connection of flowchart from page to page.



Components of a Block Diagram or Flow Chart



State



Input /
Message from user



Output /
Message to



Primitive from call



Task /
Plane C code



Text / Declarations



Return



Stop (X)



Condition



Start



Procedure start



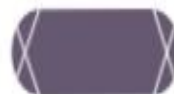
N-type start



X-type start



Raise



Exception handler



Handle



Task timer start



Task timer stop



Internal output



Block



Process



Service

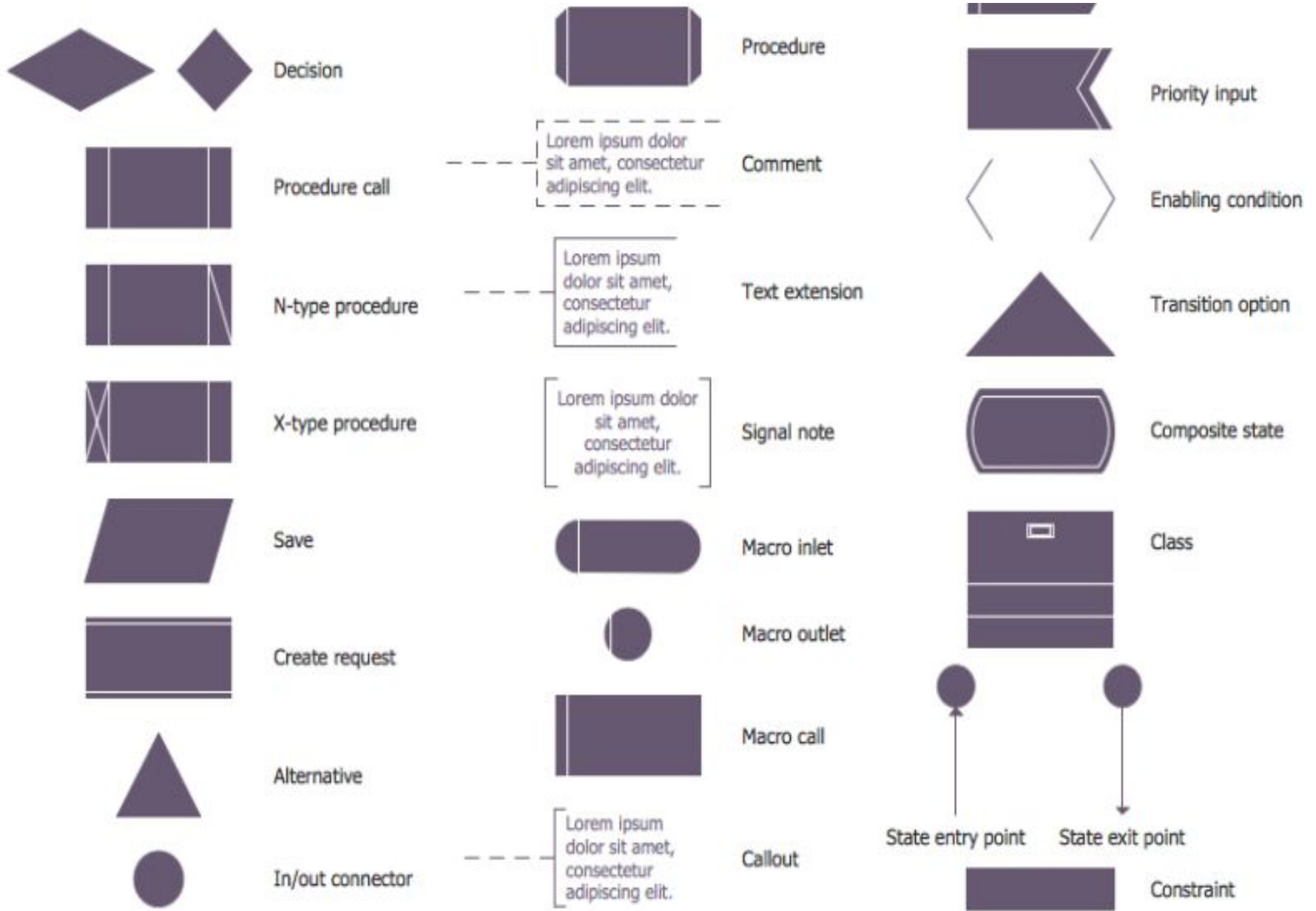


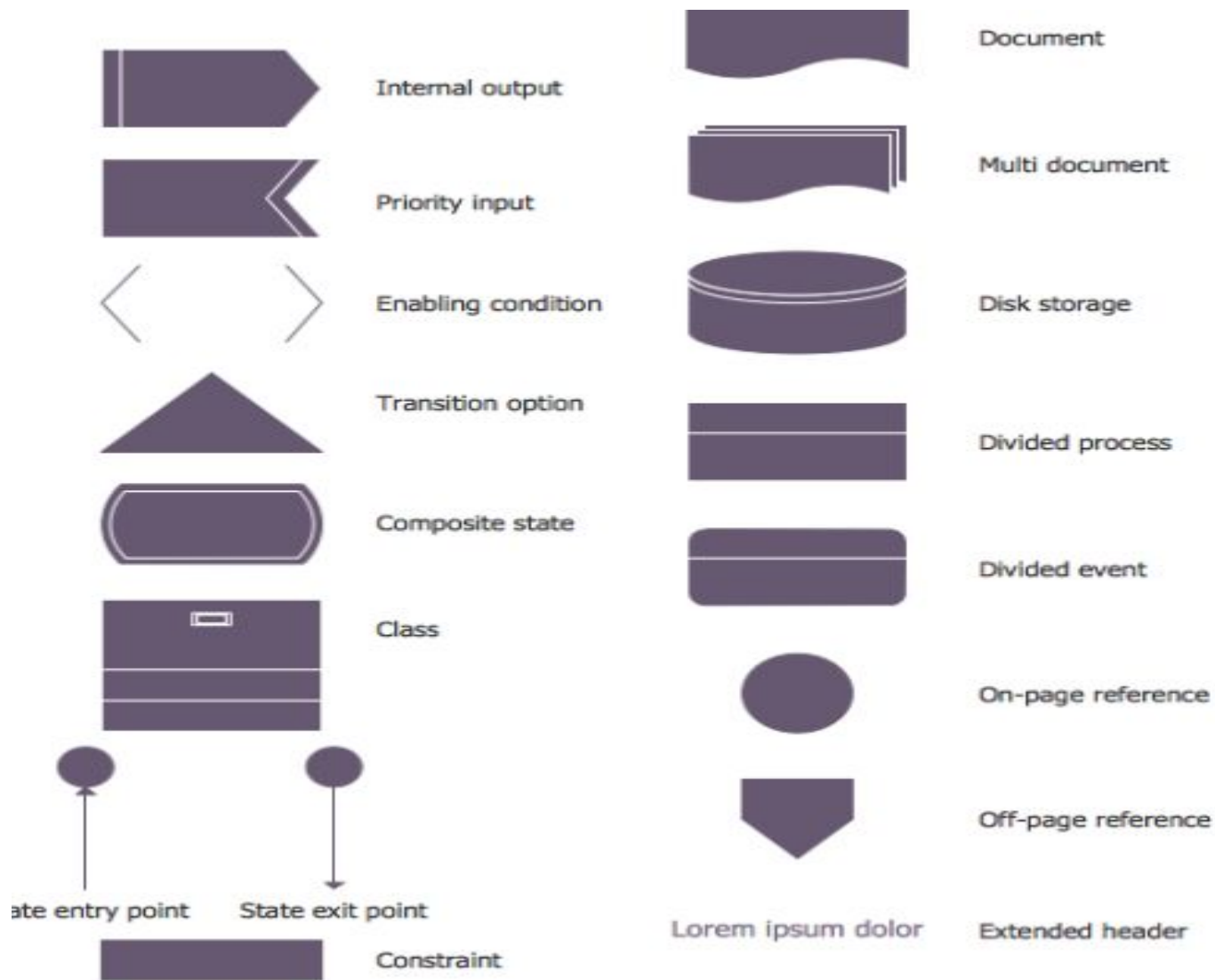
Service type



Document









Flow line



Channel - bidirectional



Transition



Channel - toggle to end arrow



Gate - bidirectional



Channel - toggle from end arrow



Gate - toggle to end arrow



Gate - toggle from end arrow



Delaying channel - bidirectional



Gate - bidirectional,
toggle virtual (inherited)



Delaying channel - toggle to end arrow



Gate - toggle to end arrow,
virtual (inherited)

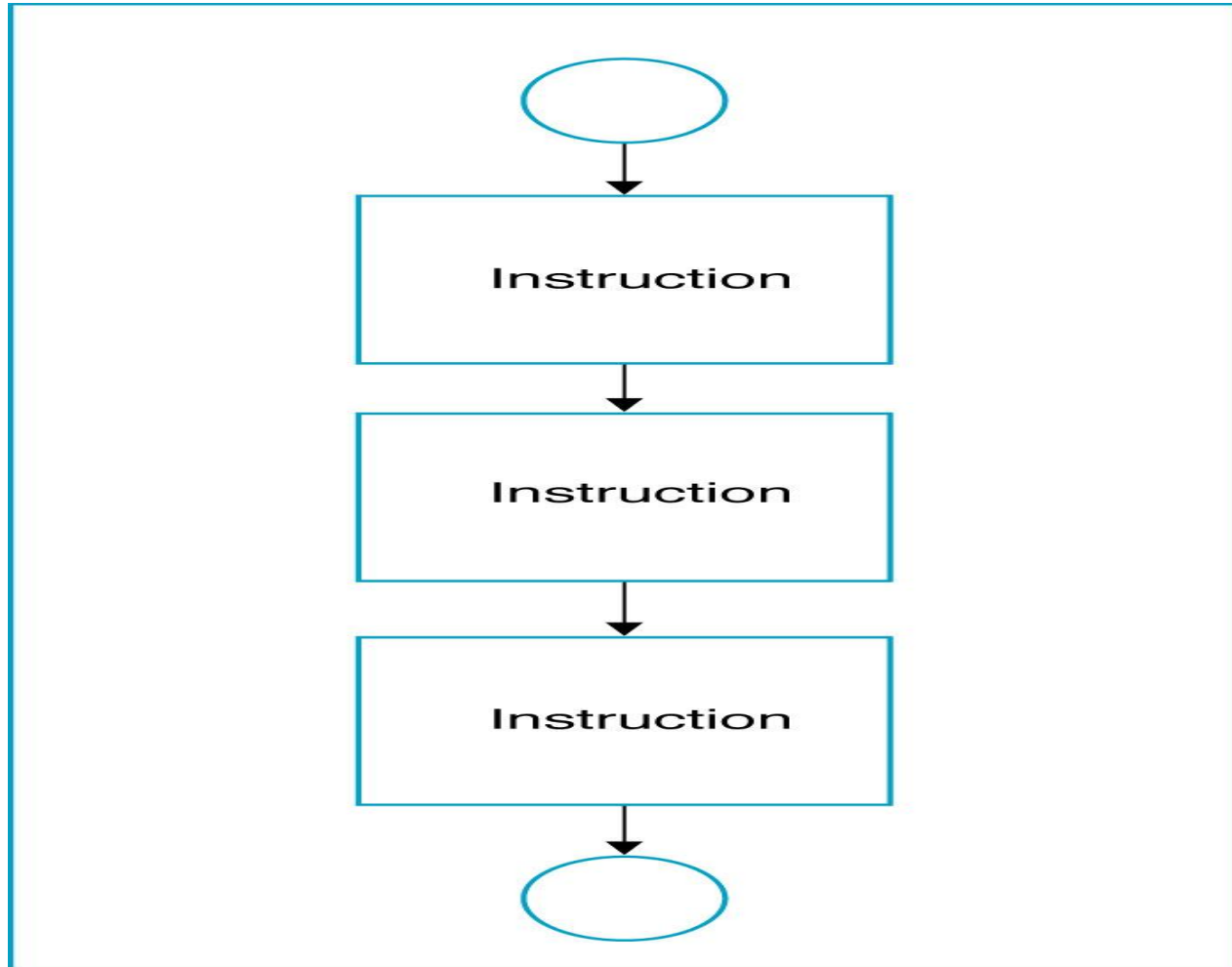


Delaying channel - toggle from end arrow



Gate - toggle from end arrow,
virtual (inherited)

Sequential Logic Structure



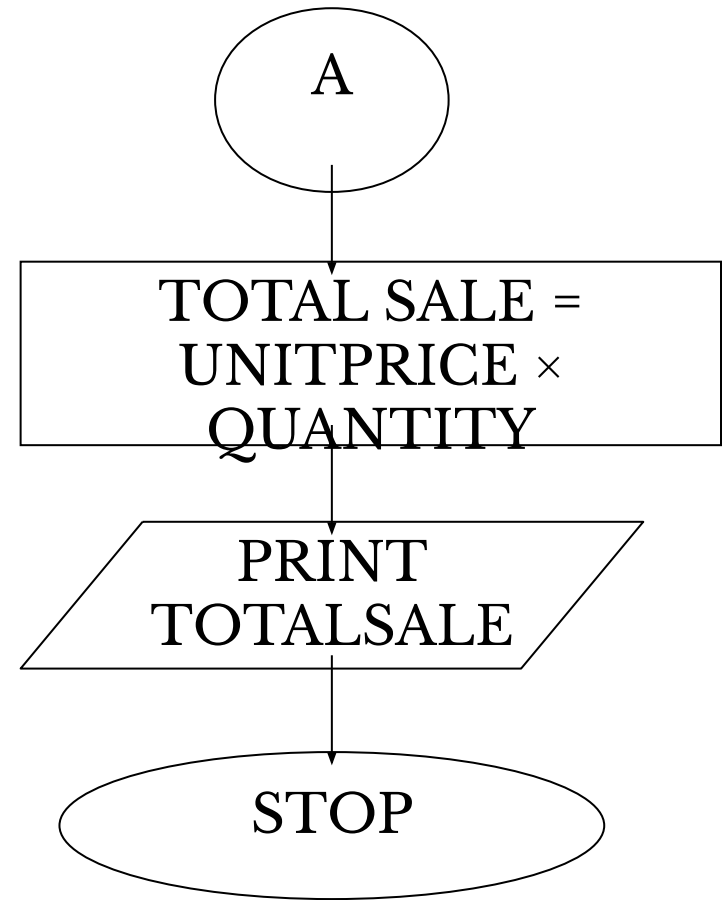
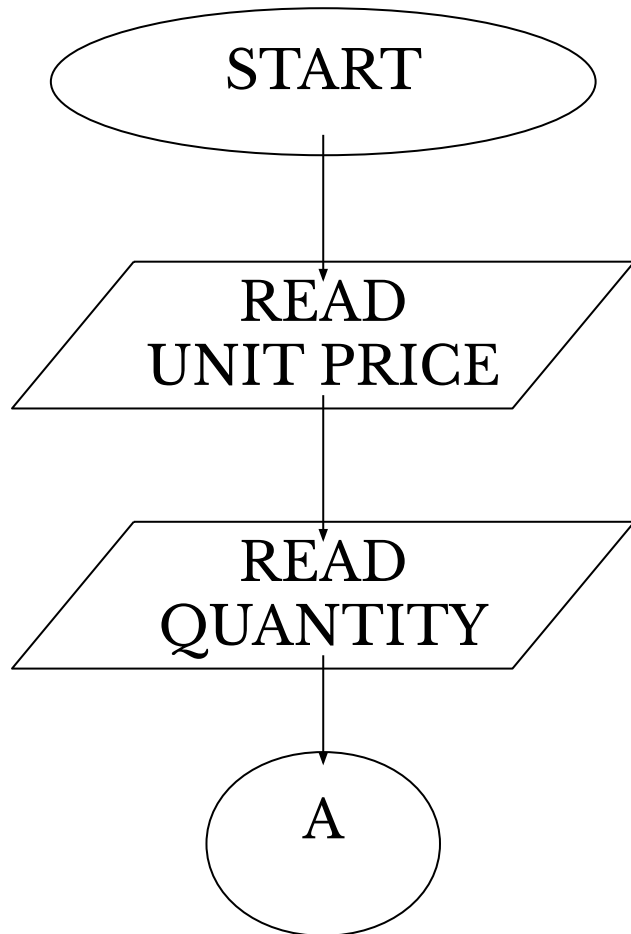
Sale Problem

Given the unit price of a product and the quantity of the product sold, draw a flowchart to calculate and print the total sale.

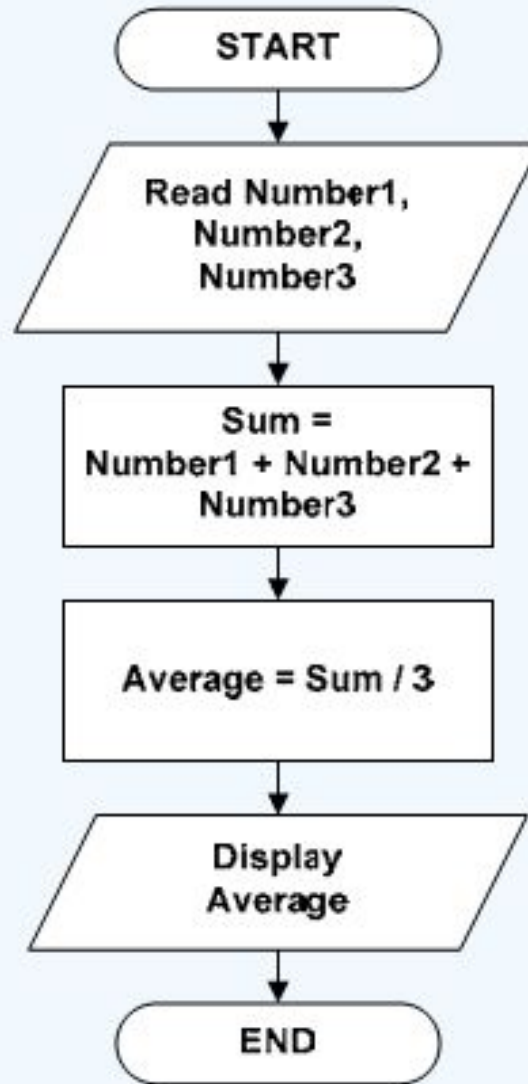
Solution: Stepwise Analysis of the Sale Problem

- Read the unit price and the quantity
- Calculate total sale = unit price and quantity
- Print total sale

PRE-PROGRAMMING PHASE



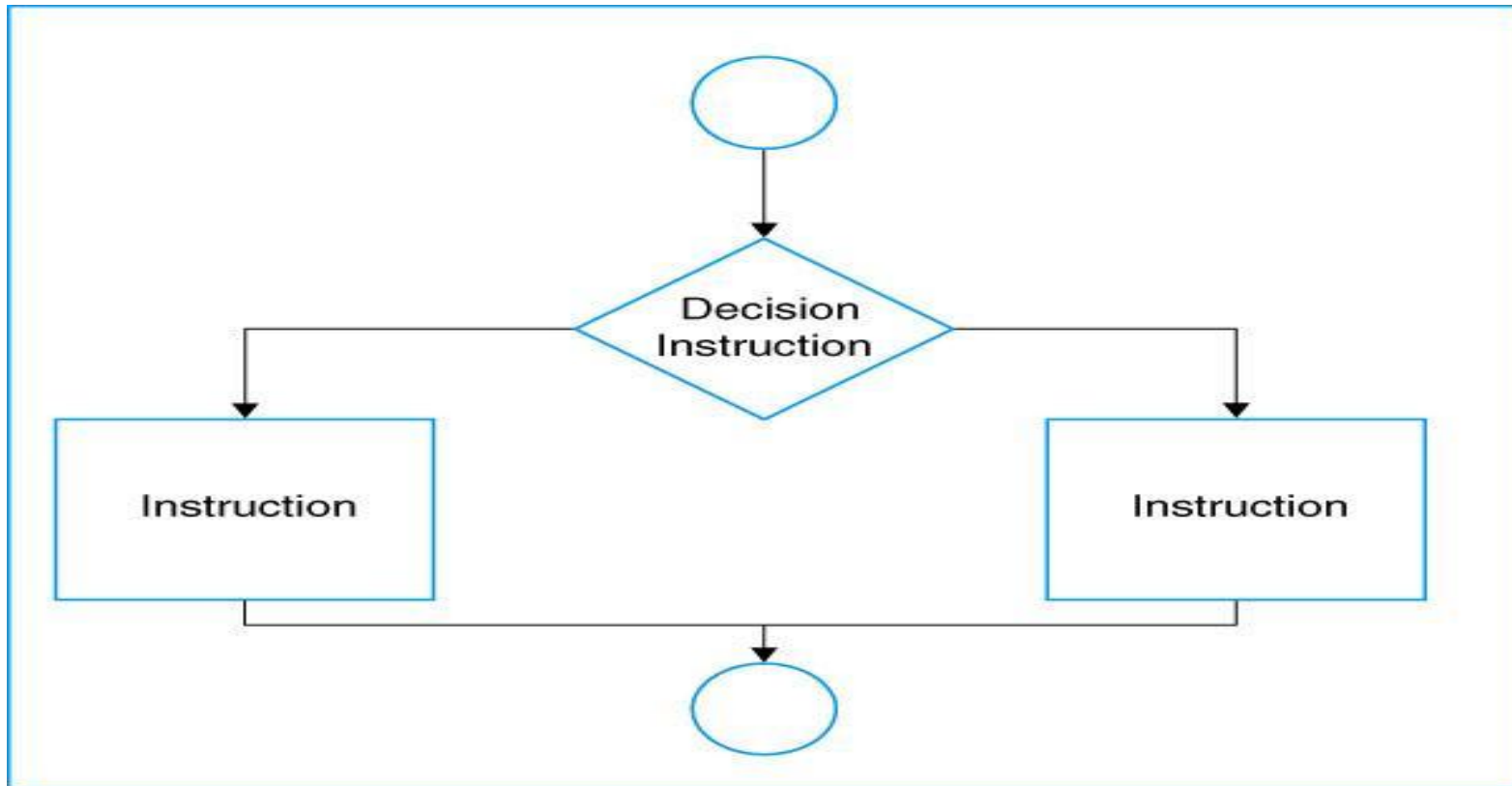
numbers



The Decision Logic Structure

- Implements using the IF/THEN/ELSE instruction.
- Tells the computer that IF a condition is true, THEN execute a set of instructions, or ELSE execute another set of instructions
- ELSE part is optional, as there is not always a set of instructions if the conditions are false.
- Algorithm:
IF <condition(s)> THEN
 <TRUE instruction(s)>
ELSE

Decision Logic Structure



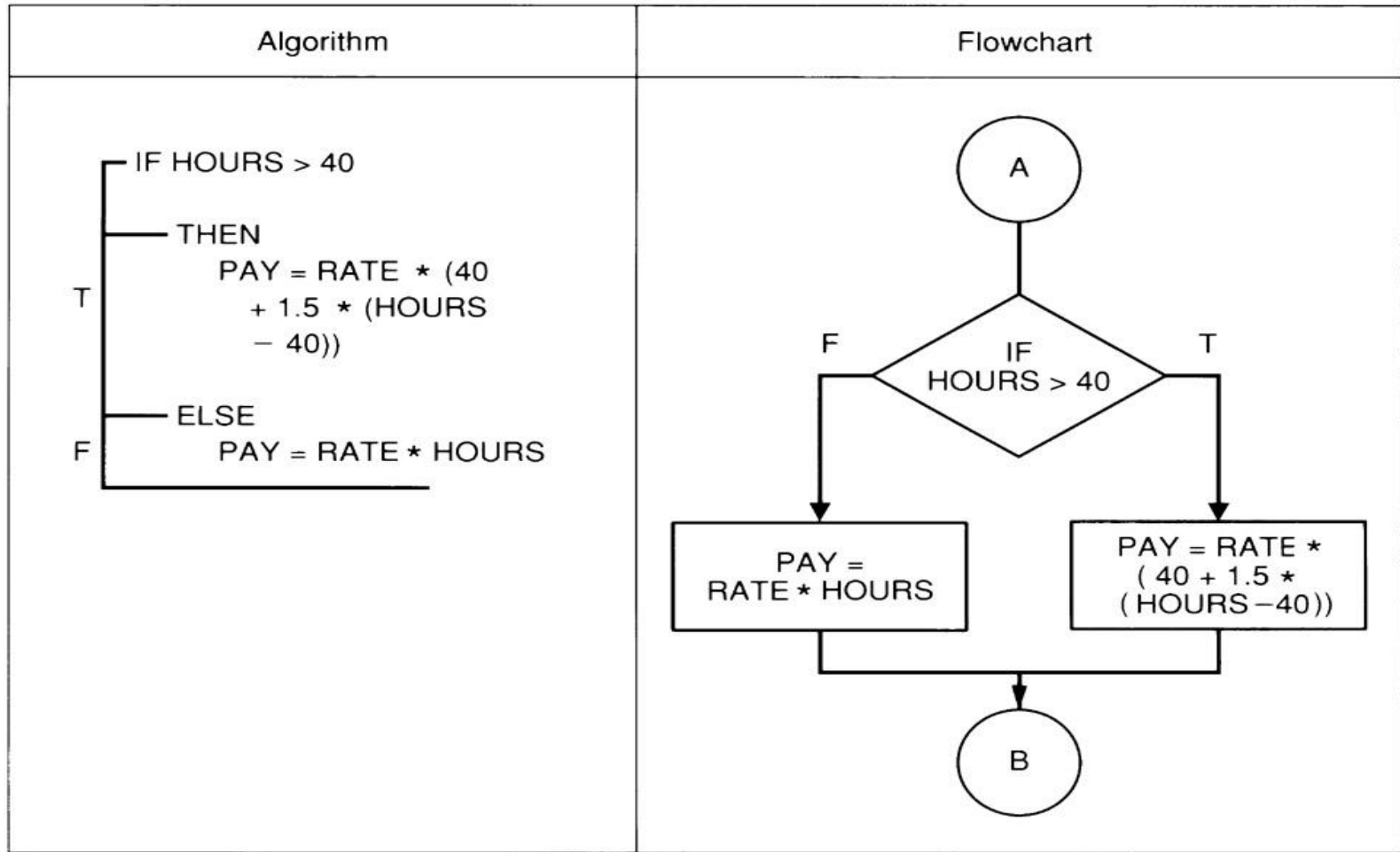
Examples of conditional expressions

- $A < B$ (A and B are the same data type – either numeric, character, or string)
- $X + 5 \geq Z$ (X and Z are numeric data)
- $E < 5$ or $F > 10$ (E and F are numeric data)
- DATAOK (DATAOK – logical datum)

Conditional Pay Calculation

- Assume you are calculating pay at an hourly rate, and overtime pay(over 40 hours) at 1.5 times the hourly rate.
- IF the hours are greater than 40, THEN the pay is calculated for overtime, or ELSE the pay is calculated in the usual way.

Example Decision Structure

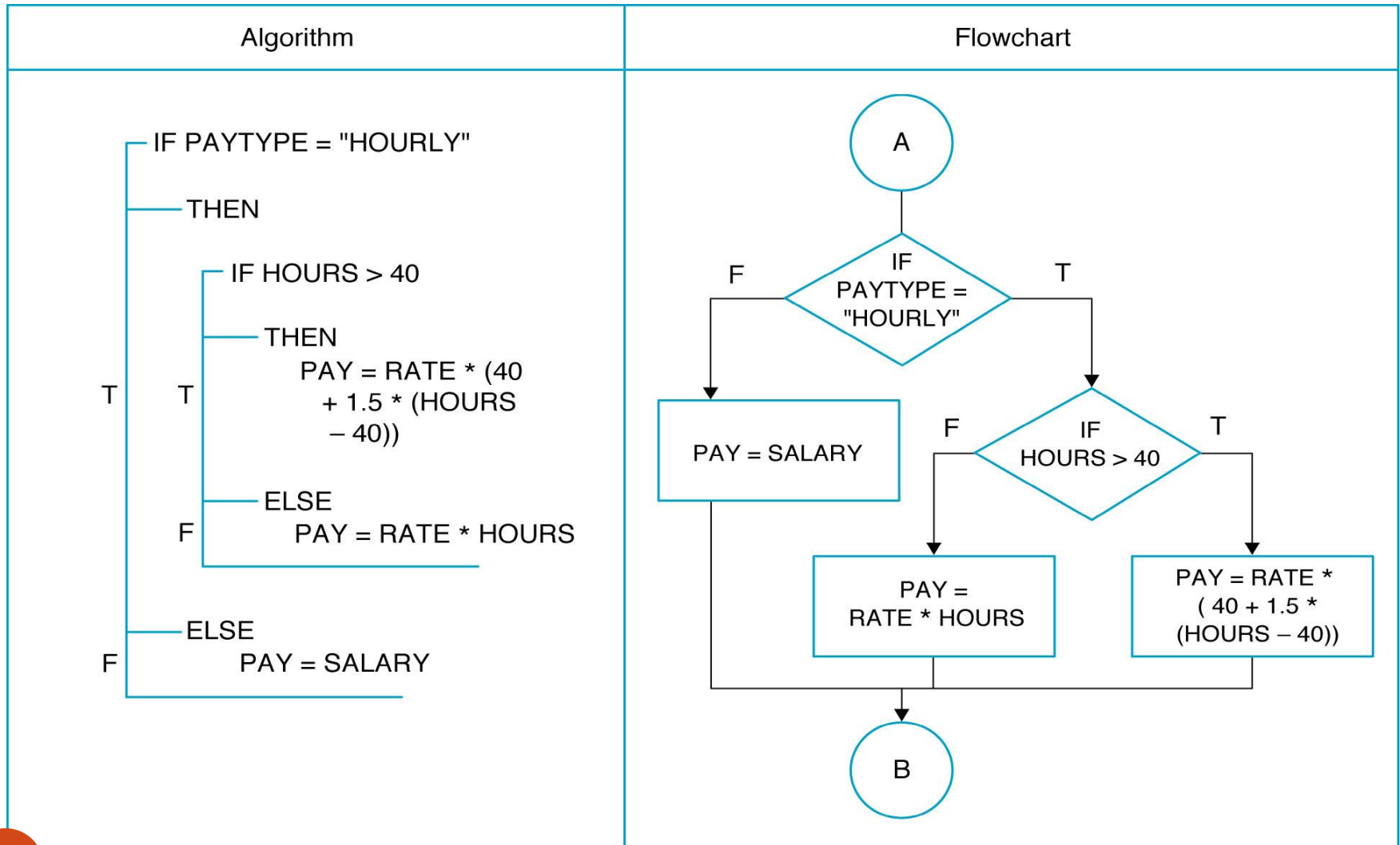


Note: For all flowcharts with decision blocks, T = TRUE and F = FALSE

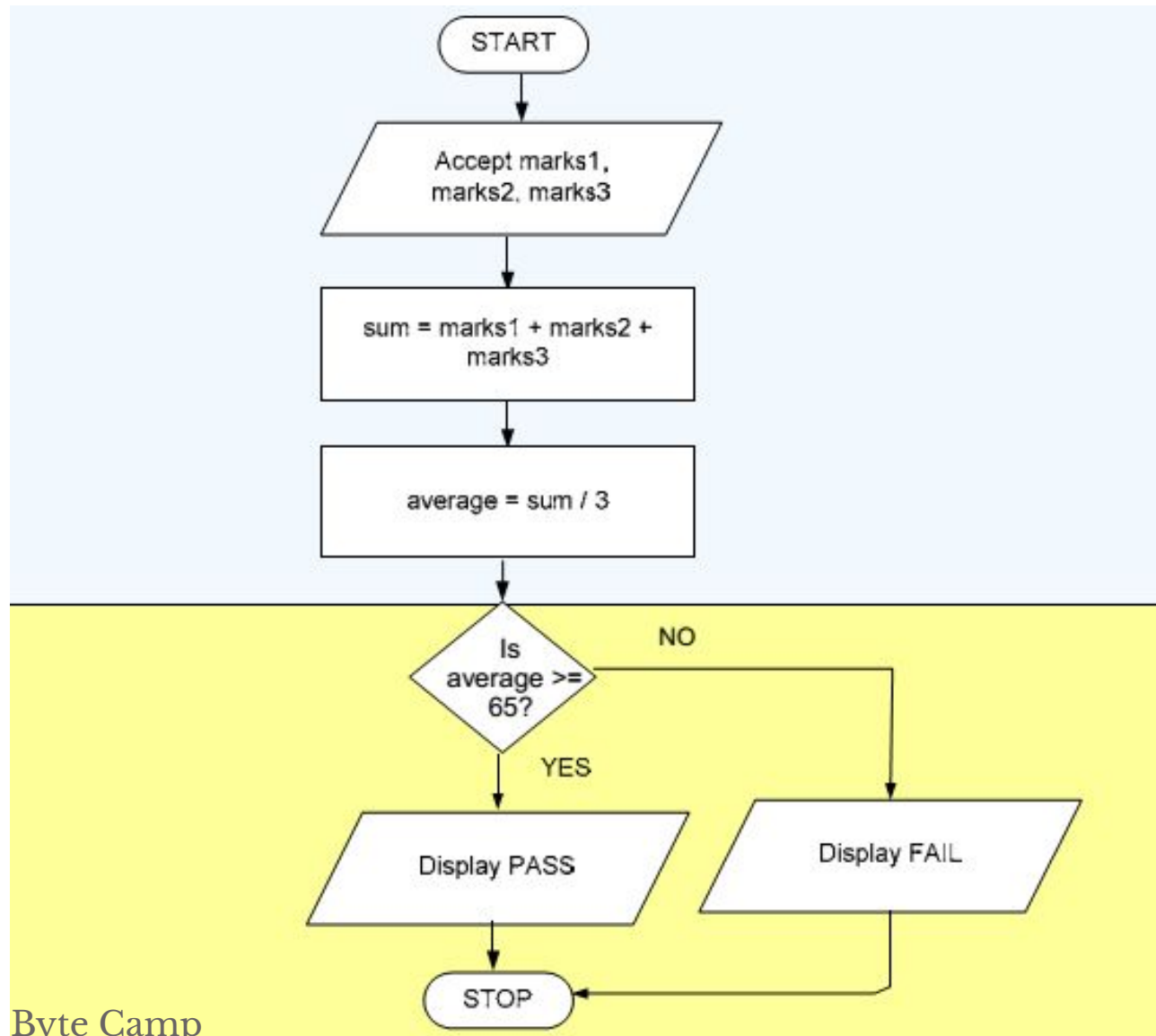
NESTED IF/THEN/ELSE INSTRUCTIONS

- Multiple decisions.
- Instructions are sets of instruction in which each level of a decision is embedded in a level before it.

NESTED IF/THEN/ELSE INSTRUCTIONS



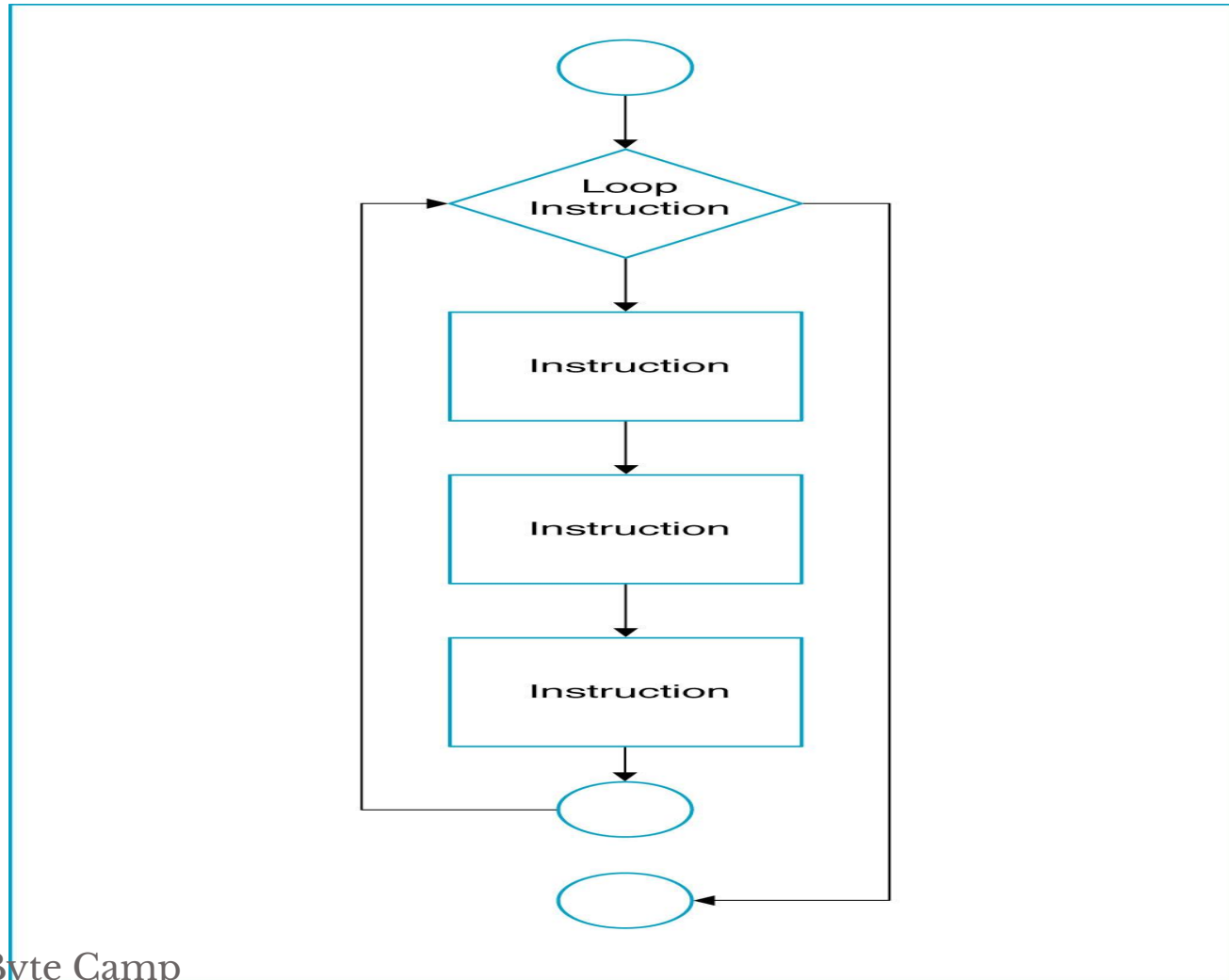
Flow Chart - Selectional



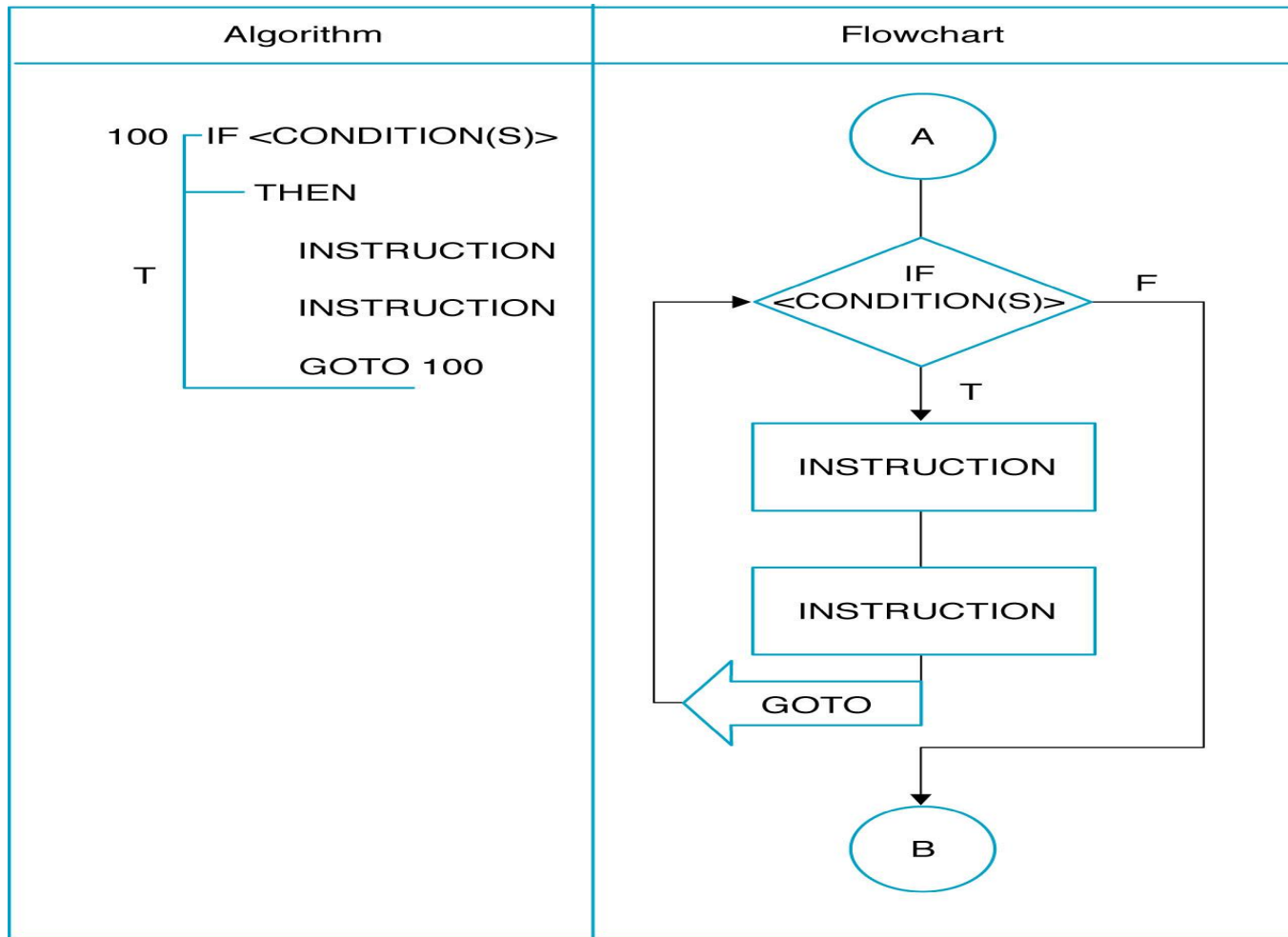
Iterational Structure

- Repeat structure
- To solve the problem that doing the same task over and over for different sets of data
- Types of loop:
 - WHILE loop
 - Do..WHILE loop
 - Automatic-Counter Loop

Loop Logic Structure



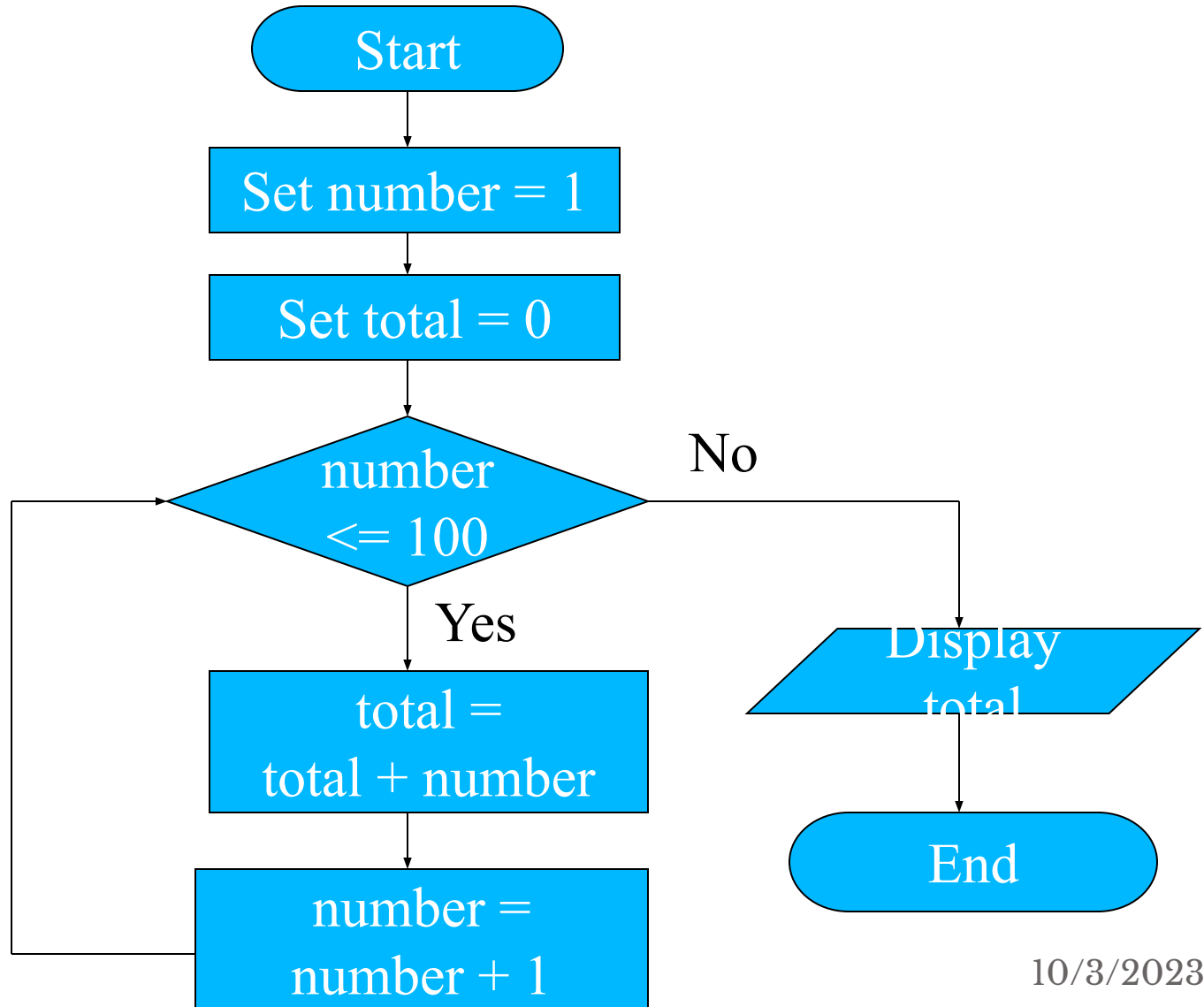
WHILE loop



WHILE loop

- Do the loop body if the condition is true.
- Example: Get the sum of 1, 2, 3, ..., 100.
 - Algorithm:
 - Set the number = 1
 - Set the total = 0
 - While (number <= 100)
 - total = total + number
 - number = number + 1
 - End While
 - Display total

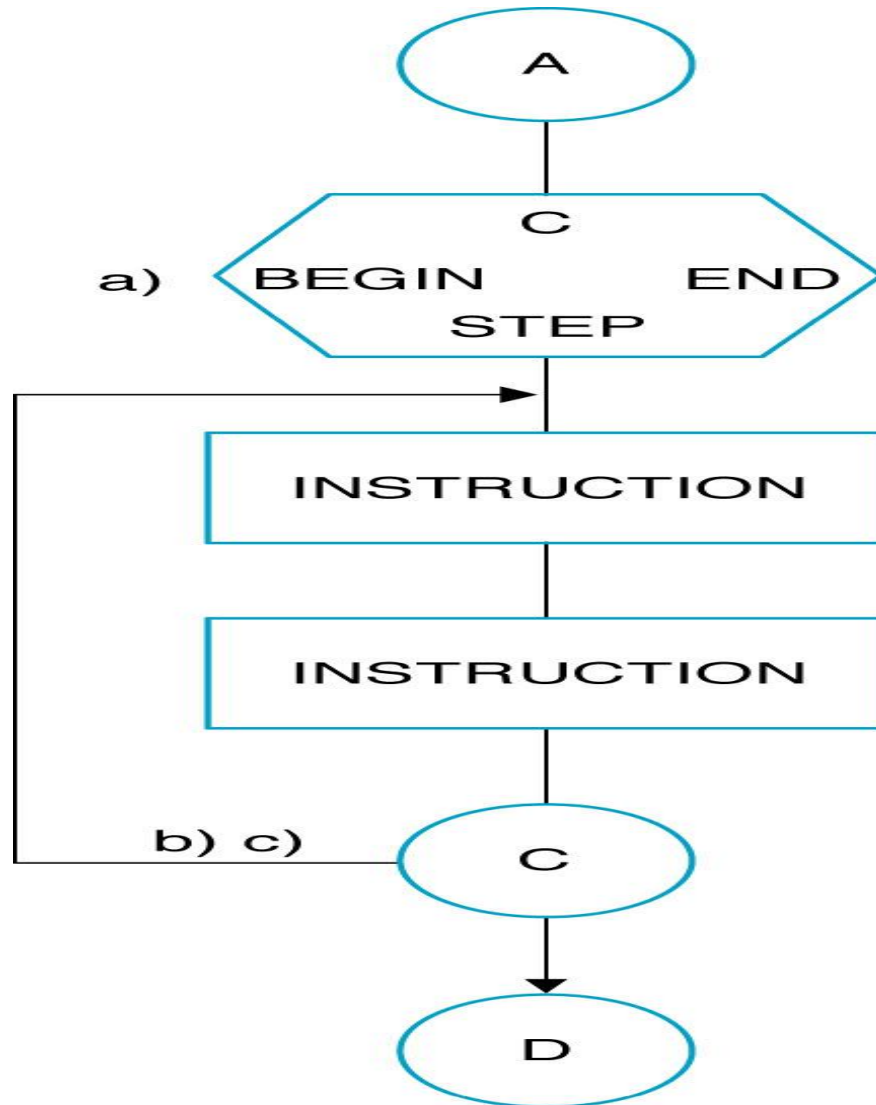
WHILE loop



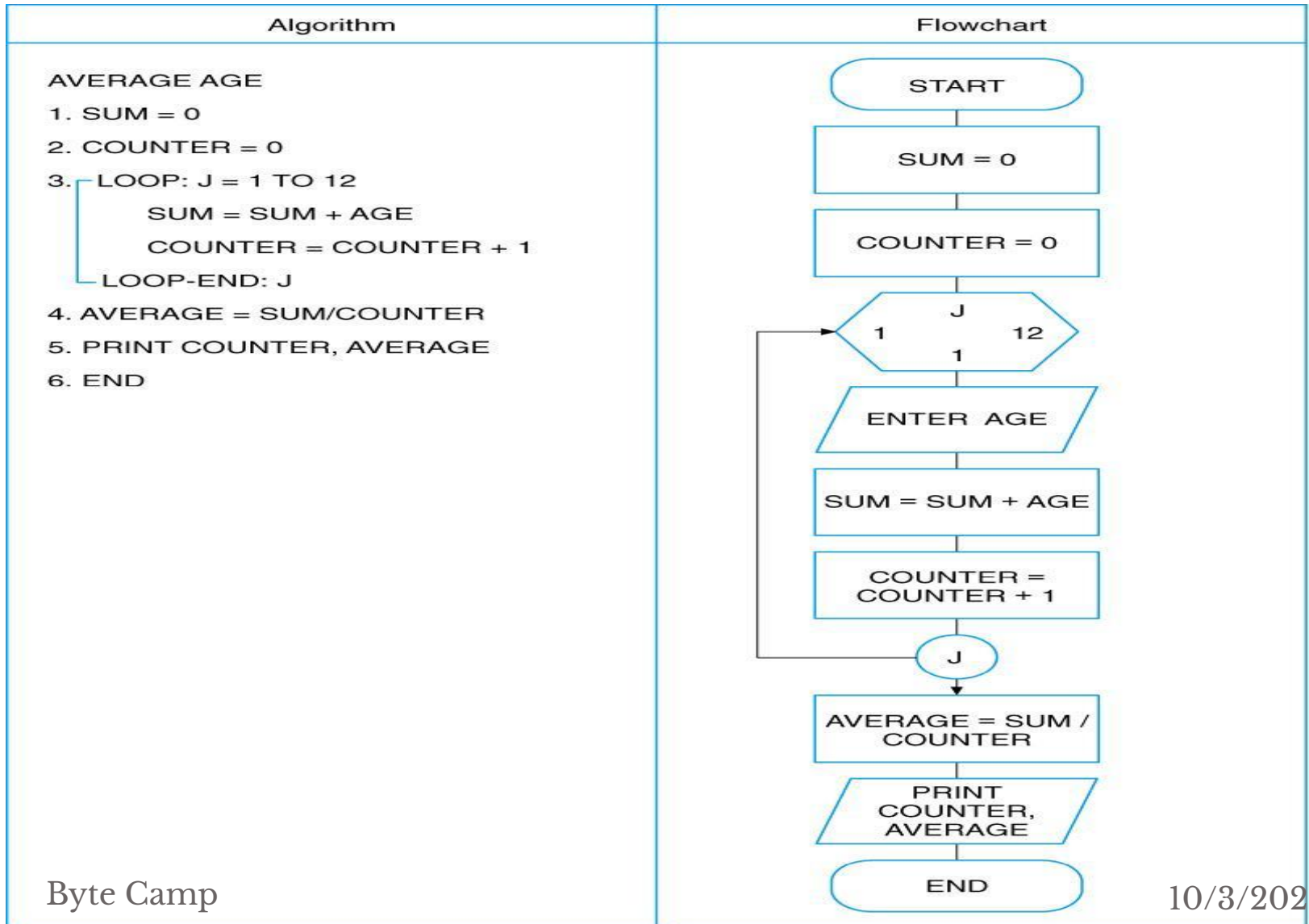
Automatic Counter Loop

- Use variable as a counter that starts counting at a specified number and increments the variable each time the loop is processed.
- The beginning value, the ending value and the increment value may be constant.
- They should not be changed during the processing of the instruction in the loop.

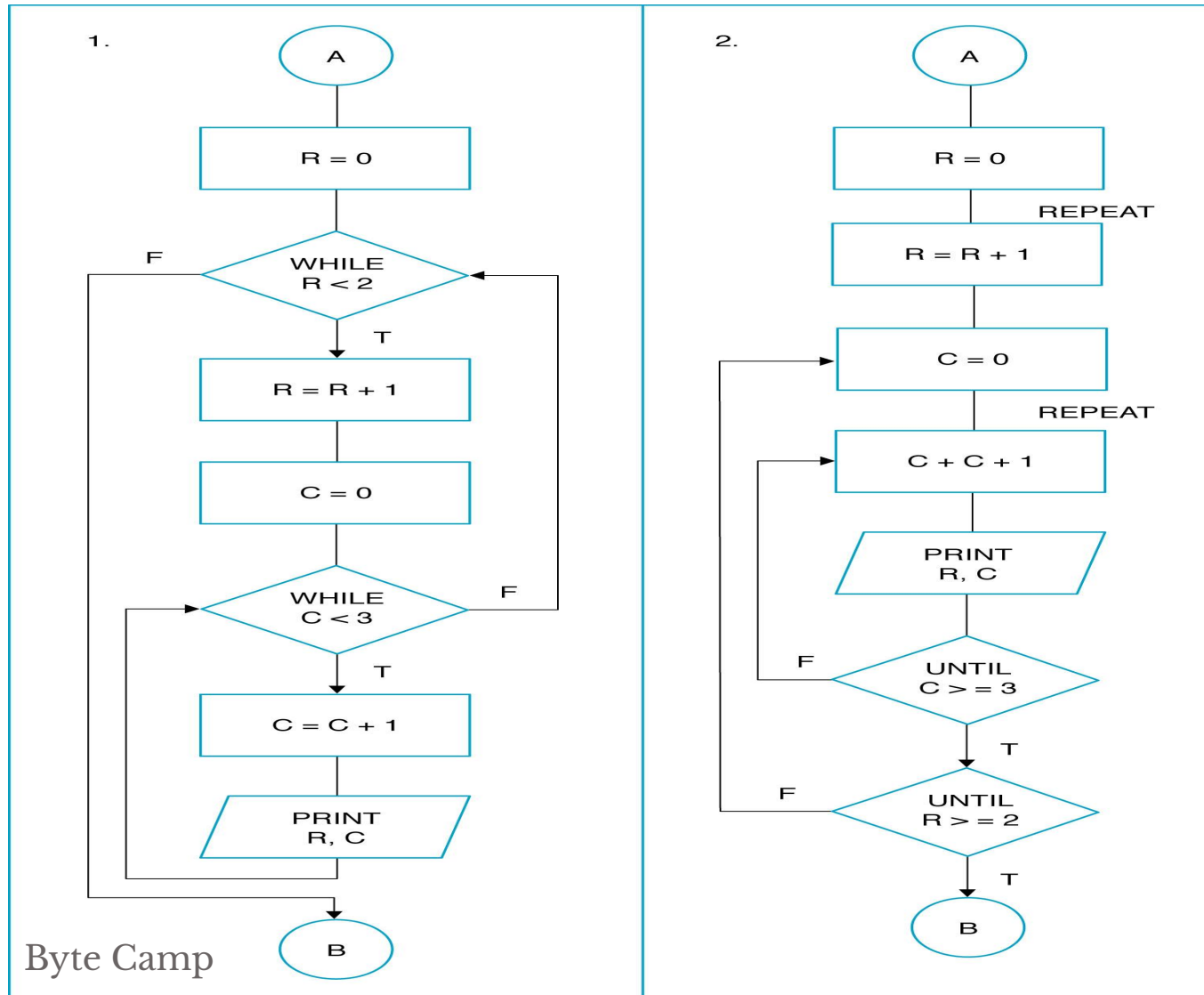
Automatic-Counter Loop



Automatic-Counter Loop

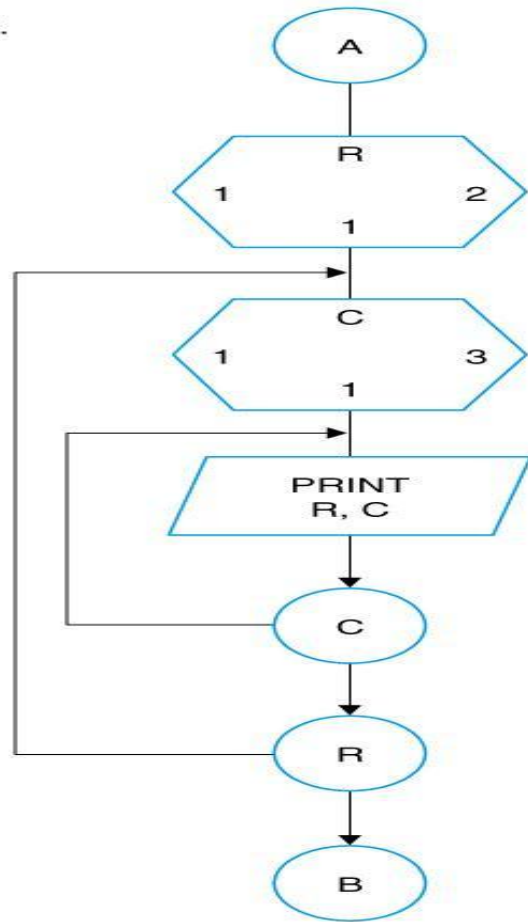


NESTED LOOP

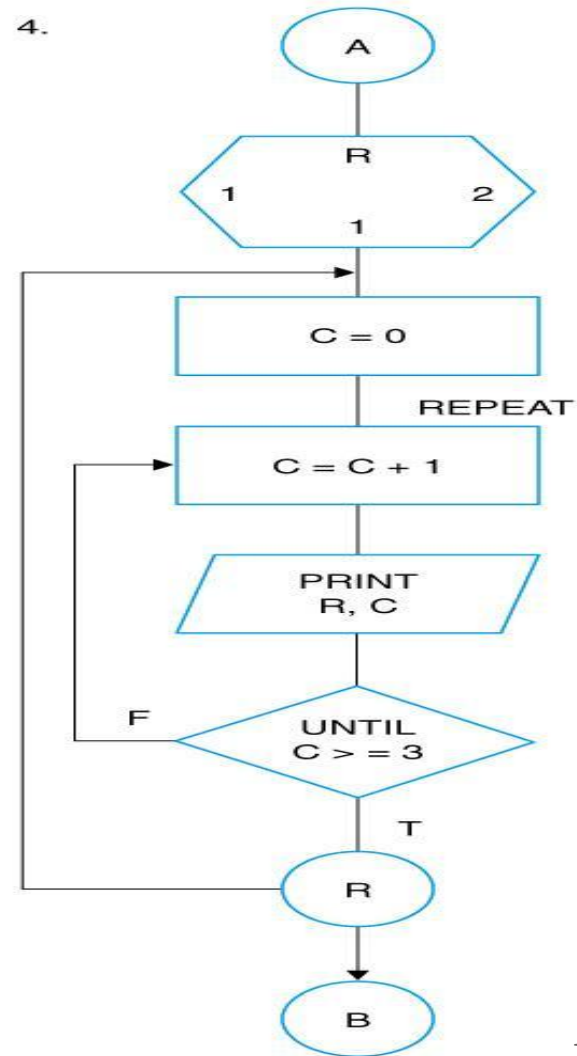


NESTED LOOP

3.



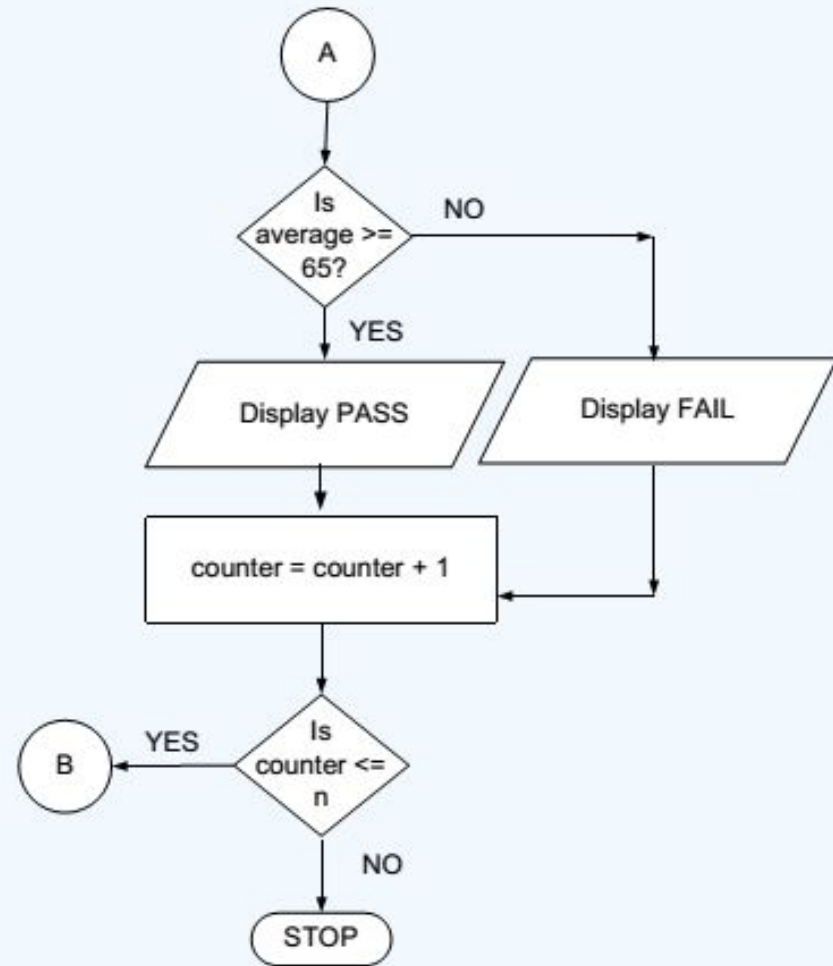
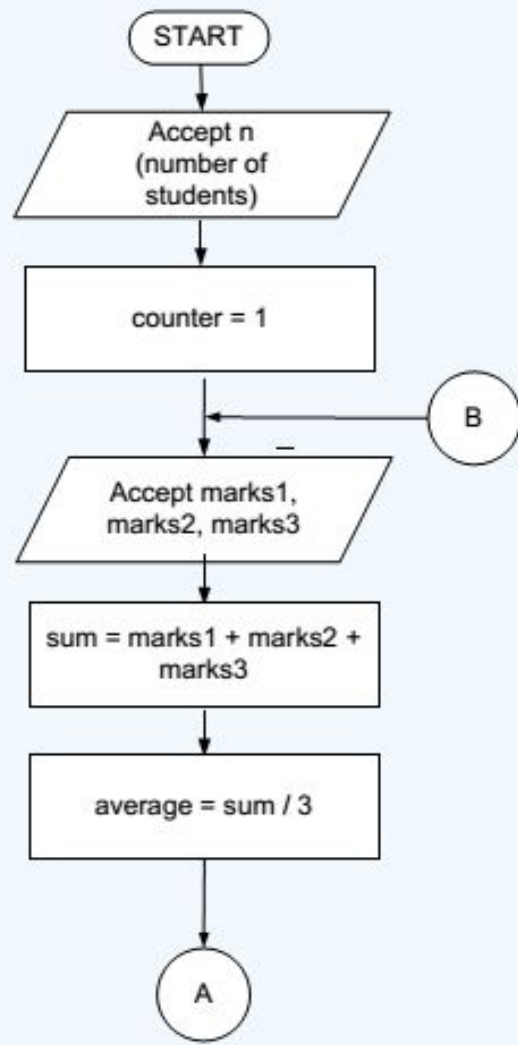
4.



Example (Iterational)

- Write a program to find the average of marks scored by him in three subjects for 'N' students. And then test whether he passed or failed. For a student to pass, average should not be less than 65.

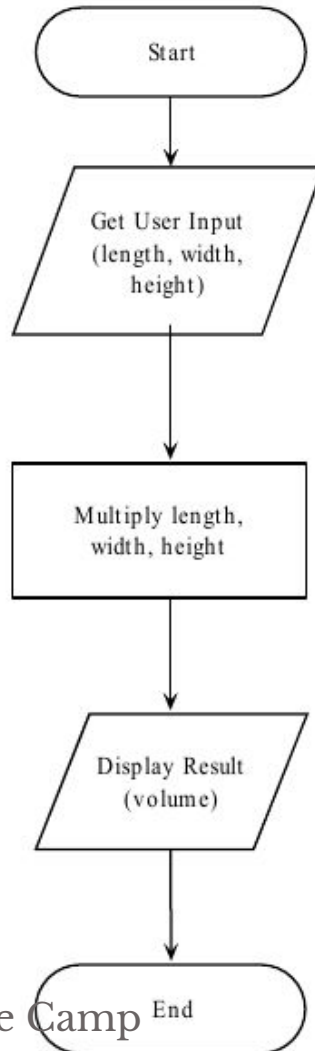
Flow Chart Iterational



Tool demo

- Yed tool shall be used for giving demo

Pseudocode – Partial English and Programming Language terms



Get length, width, height
Compute volume
 $\text{volume} = \text{length} * \text{width} * \text{height}$
Store volume
Display volume

Programming Or Implementation Phase

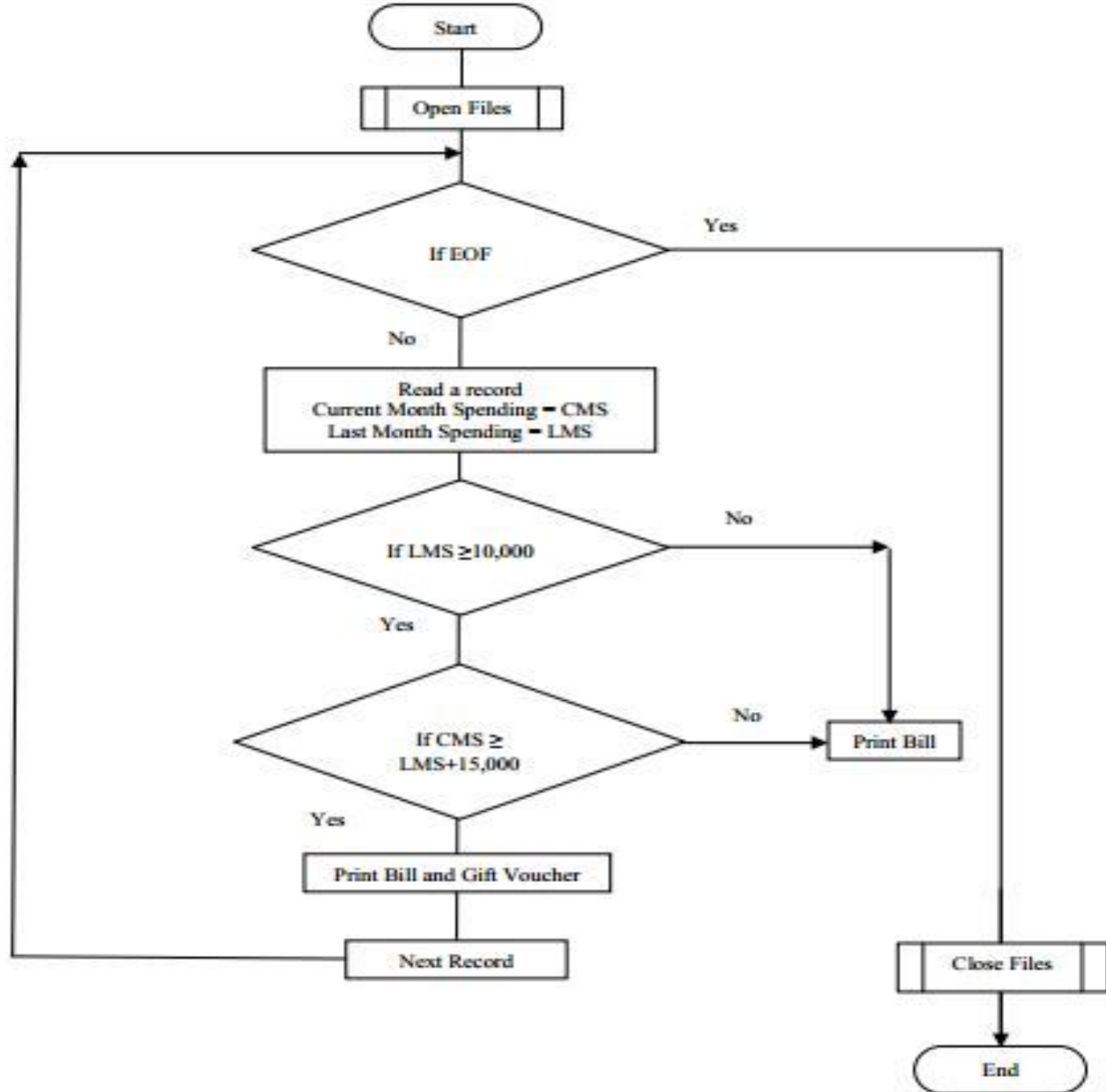
- Transcribing the logical flow of solution steps in flowchart or algorithm to program code and run the program code on a computer using a programming language.
- Programming phase takes 5 stages:
 - Coding.
 - Compiling.
 - Debugging.
 - Run or Testing.
 - Documentation and maintenance.

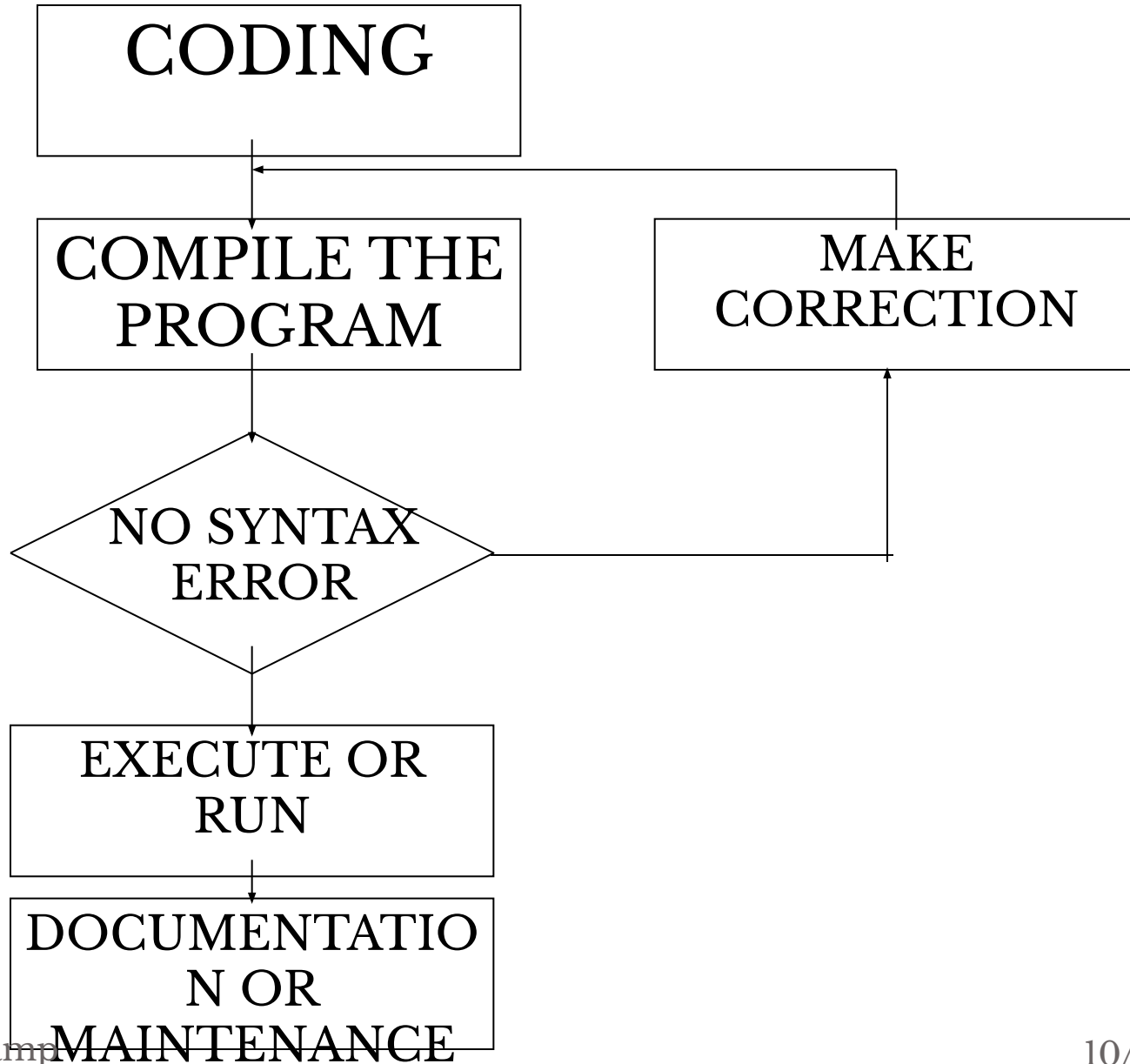
Programming Or Implementation Phase

- Once the program is coded using one of the programming language, it will be compiled to ensure there is no syntax error.
- Syntax free program will then be executed to produce output and subsequently maintained and documented for later reference.

Task: Create a Flow chart

- *Bank has launched a promotion for its credit card customers. According to the promotion, the customers will receive a gift voucher worth \$500 with their monthly bill if they spend \$15,000 more than their last month spending and their last month bill is not less than \$10,000. Access the bank cloud data and get the customers information and generate the flow chart.*





Coding

- Translation or conversion of each operation in the flowchart or algorithm (pseudocode) into a computer-understandable language.
- Coding should follow the format of the chosen programming language.

Compiling and Debugging

- Compiling - Translates a program written in a particular high-level programming language into a form that the computer can understand
- Compiler checks the program code so that any part of source code that does not follow the format or any other language requirements will be flagged as syntax error.
- This syntax error is also called bug, when error is found the programmer will debug or correct the error and then recompile the source code again
- Debugging process is continued until there is no

Testing

- The program code that contains no more error is called executable program. It is ready to be tested.
- When it is tested, the data is given and the result is verified so that it should produced output as intended.
- Though the program is error free, sometimes it does not produced the right result. In this case the program faces logic error.
- Incorrect sequence of instruction is an example that causes logic error.

Documentation and Maintenance

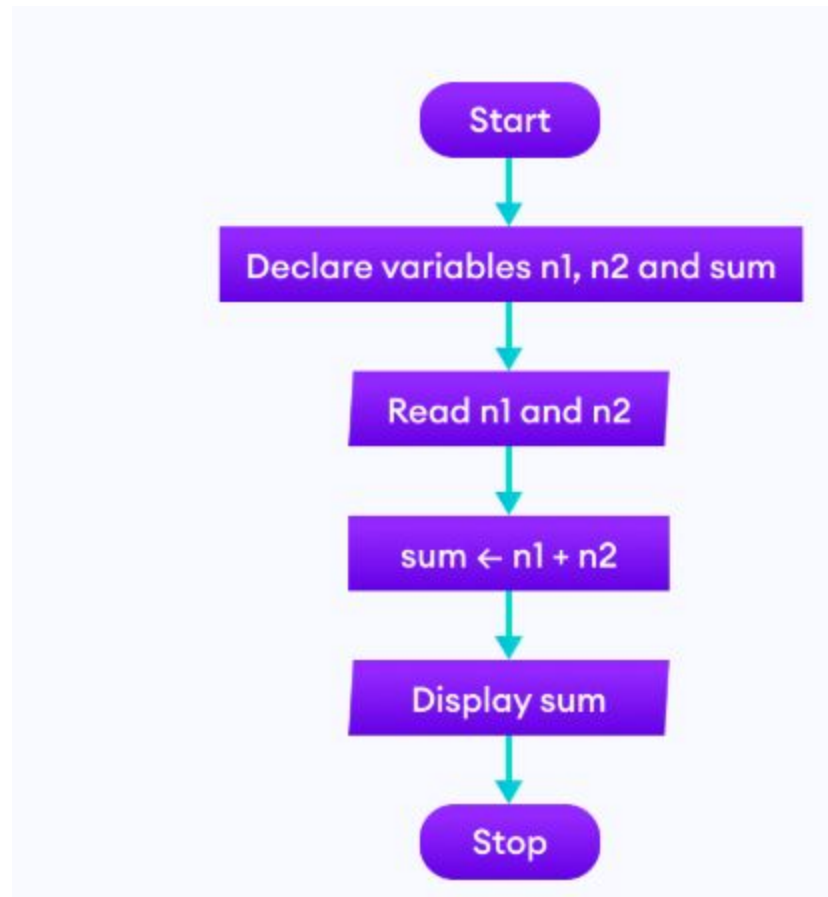
- When the program is thoroughly tested for a substantial period of time and it is consistently producing the right output, it can be documented.
- Documentation is important for future reference. Other programmer may take over the operation of the program and the best way to understand a program is by studying the documentation.
- Trying to understand the logic of the program by looking at the source code is not a good approach.
- Studying the documentation is necessary when the program is subjected to enhancement or modification.

- Documentation is also necessary for management use as well as audit purposes

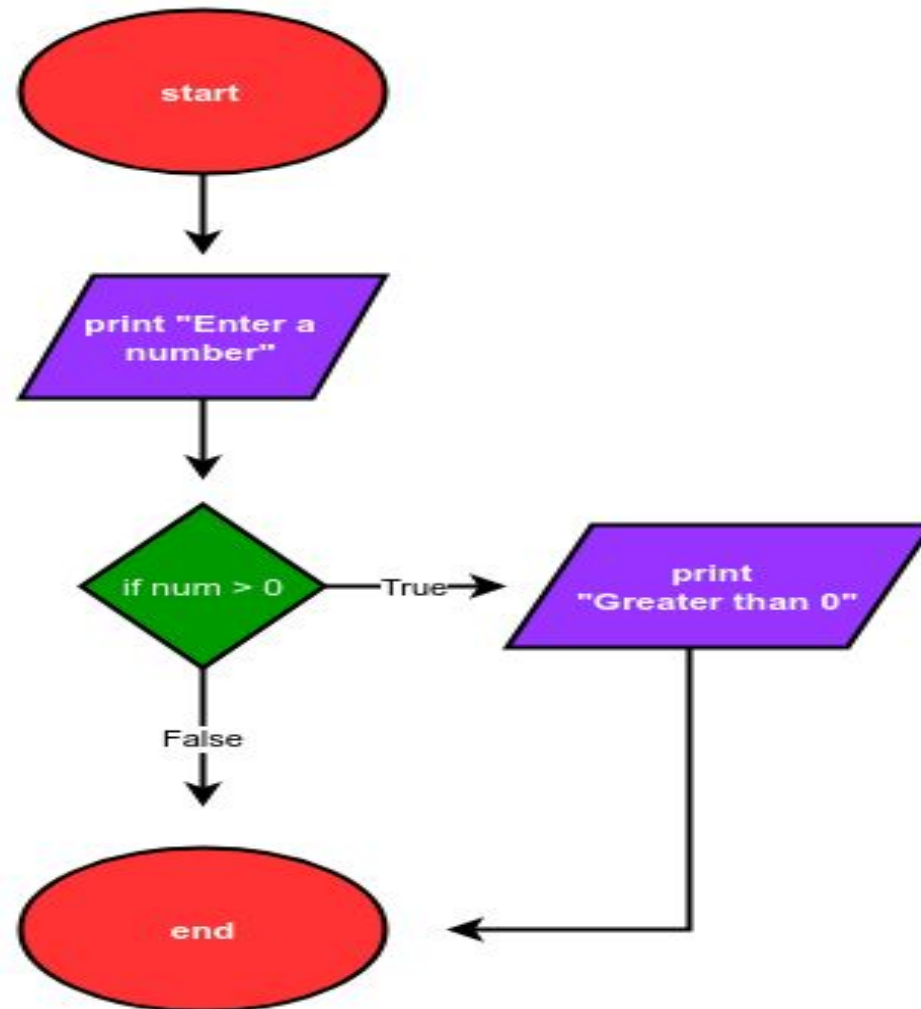
Best Practices

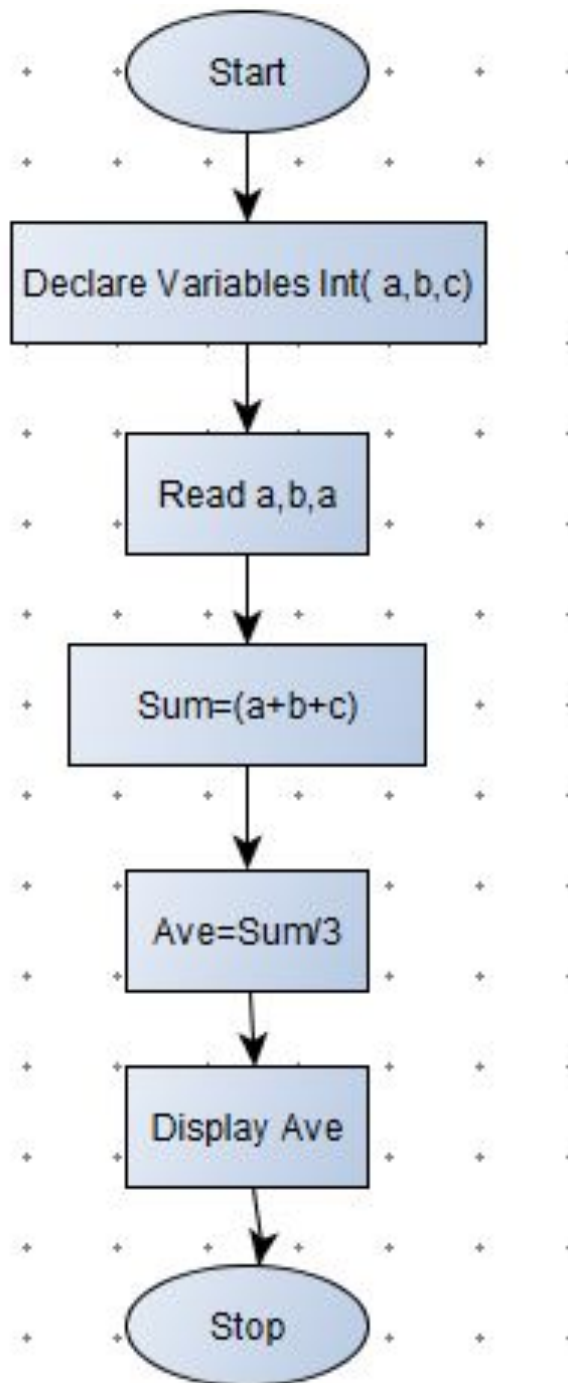
Develop efficient computer solution to problems:

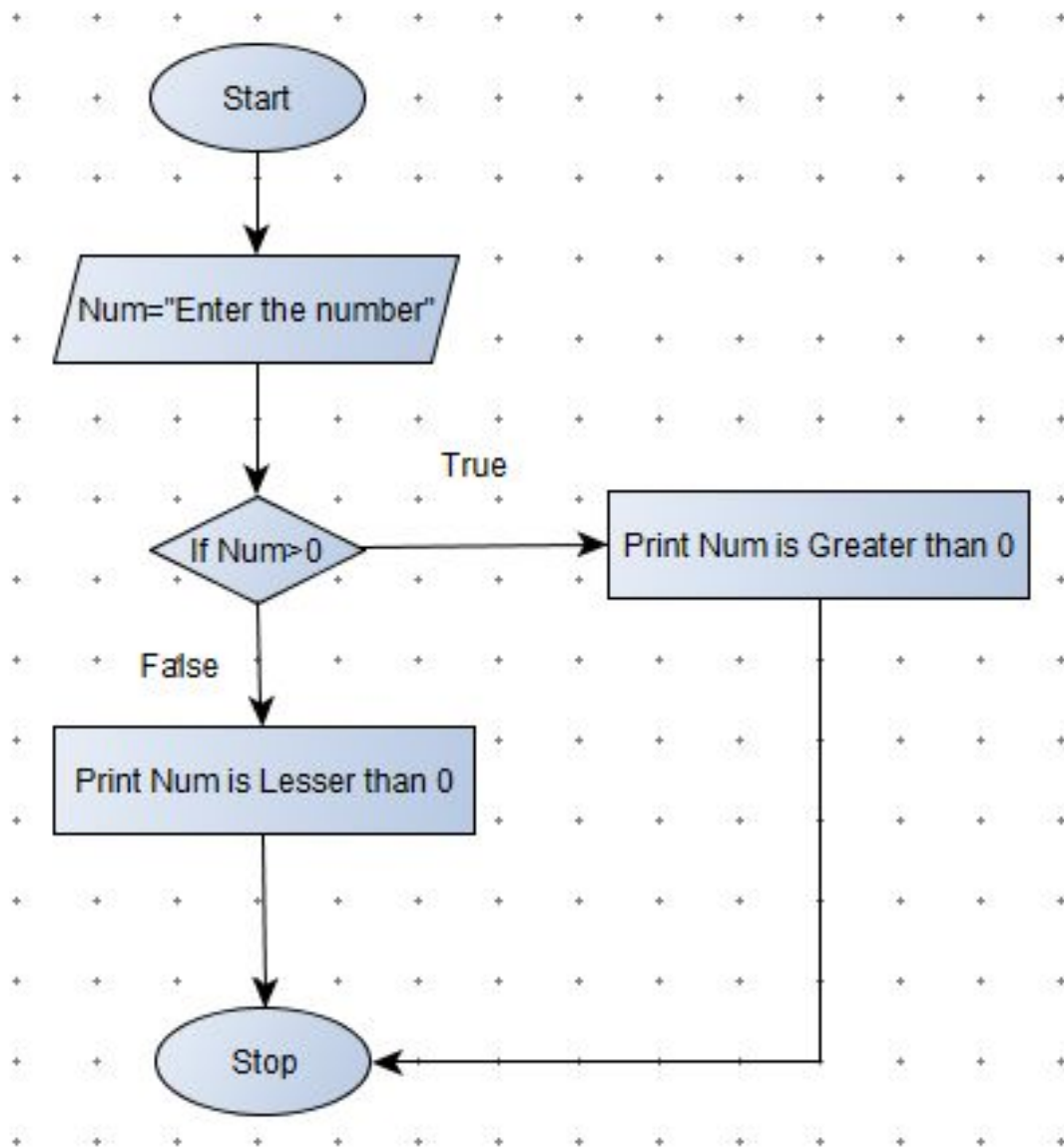
1. Use Modules
2. Use four logic structures
 - a. Sequential structure
 - Executes instructions one after another in a sequence.
 - b. Decision structure
 - Branches to execute one of two possible sets of instructions.
 - c. Loop structure
 - Executes set of instruction many times.
 - d. Case structure
 - Executes one set of instructions out of several sets.
3. Eliminate rewriting of identical process by using modules.
4. Use techniques to improve readability including four logic structure, proper naming of variables, internal documentation and proper indentation.

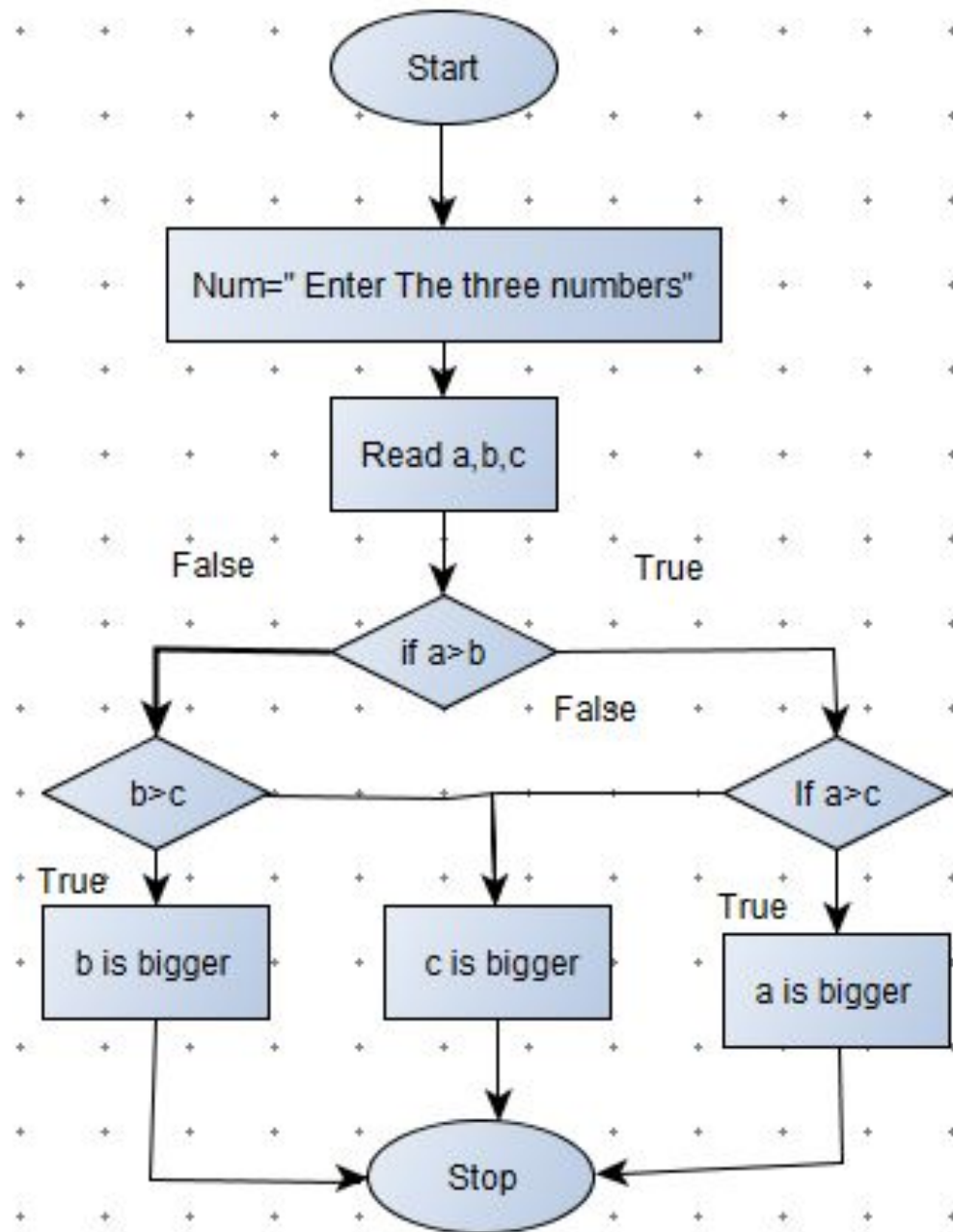


The program starts.
the program asks a user for a number.
If the number is greater than zero, the program prints "Greater than 0",
then the program ends.









- The program starts.
- the program asks a user for a number. If the number is greater than zero, the program prints "Greater than 0". If the number is less than zero, the program prints "Less than 0". Then the program prints "Done" and the program ends.

