CS -01

PROBLEM SOLVING METHODOLOGIES AND PROGRAMMING IN C

By Rachel

UNIT 1- PART 5

INTRODUCTION OF

LOGIC DEVELOPMENT TOOLS

What is Programming Logic?

It is a set of principles applied in a disciplined manner to achieve an acceptable result.

Necessary instructions for developing logic

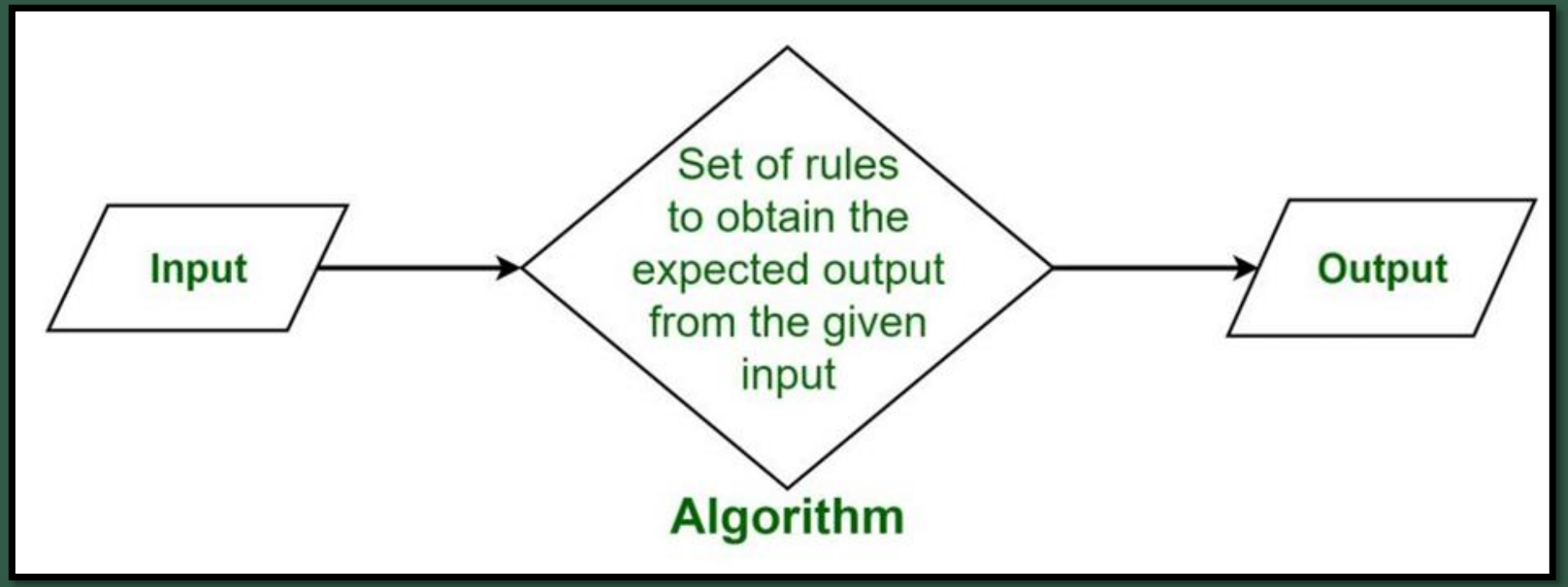
- Improve fundamentals of coding
- Analyze the code written by others
- Develop problem solving skills
- Improve the coding skills
- Use flow chart to represent the flow of code
- Practice coding step by step using flow chart

What are the main logic development techniques?

- Algorithm
- Flow chart
- Dry run

1. Algorithm

An algorithm is a step-by-step procedure to solve a given problem. In the context of computer science, particularly with the C programming language, an algorithm is used to create a solution that computers can understand and execute.



Algorithm AVERAGE

Step 0 START

Step 1 INPUT first number into variable A

Step 2 INPUT second number into variable B

Step 3 INPUT third number into variable C

Step 4 COMPUTE SUM = A + B + C

Step 5 COMPUTE AVG = SUM / 3

Step 6 DISPLAY AVG

Step 7 END

Algorithm DIVISION

```
Step 0   START
Step 1   INPUT first number into variable A
Step 2   INPUT second number into variable B
Step 3   IF B != 0 THEN
                    Q = A/B
                    DISPLAY Q
         END IF
Step 4   END
```

2. Flowchart

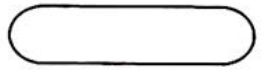
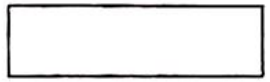
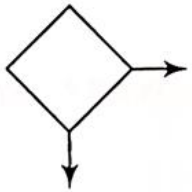
- It is a diagrammatic representation that illustrates the sequence of operations to be performed to get the solution of a problem.

- Once a flowchart is drawn, it becomes easier to write the program in any high level language.

Guidelines for drawing a flowchart

1. Standard symbols should be used while drawing flowchart.
2. Ensure that flowchart has START (or BEGIN) and STOP (or END).
3. Flowchart should be neat, clean and easy to follow. There should be no any ambiguity.
4. The usual direction of flowchart is from top to bottom or from left to right.
5. The terminal symbol, that is, START/BEGIN or STOP/END should have only one flow line.
6. Only one flow line should come out from process symbol.
7. Only one flow line should enter a decision symbol, but two or three flow-lines, one for each possible answer, can leave the decision symbol.
8. If the flowchart is lengthy and complex connector symbol should be used to reduce the number of flow lines.
9. Avoid intersection of flow lines.
10. Use annotation symbol to describe steps more clearly.

Notations used in flowchart

Name	*Description*	*Symbol*
Terminal symbol	This symbol is used to mark the beginning and end of a flowchart.	
Input/output box	This box is used to represent input operations or output operations.	
Process box	This box is used to represent processing operations. More specifically, this symbol is used to represent the imperative logic construct.	
Decision box	This diamond-shaped symbol is used to represent conditional statement constructs. Each such box has two exit points which correspond to the evaluation of the condition to true and false.	
Connector (intra-page)	This symbol is used to transfer the flow of control from one point to another within a page. A connector is usually labeled to match with its counterpart.	
Off-page connector (inter-page)	This symbol is used to transfer the flow of control from one point on a page to another point on a different page. This notation is useful when the flowchart spans more than one page.	
Arrowed lines	These are used to show the sequence of flow from one box to another.	

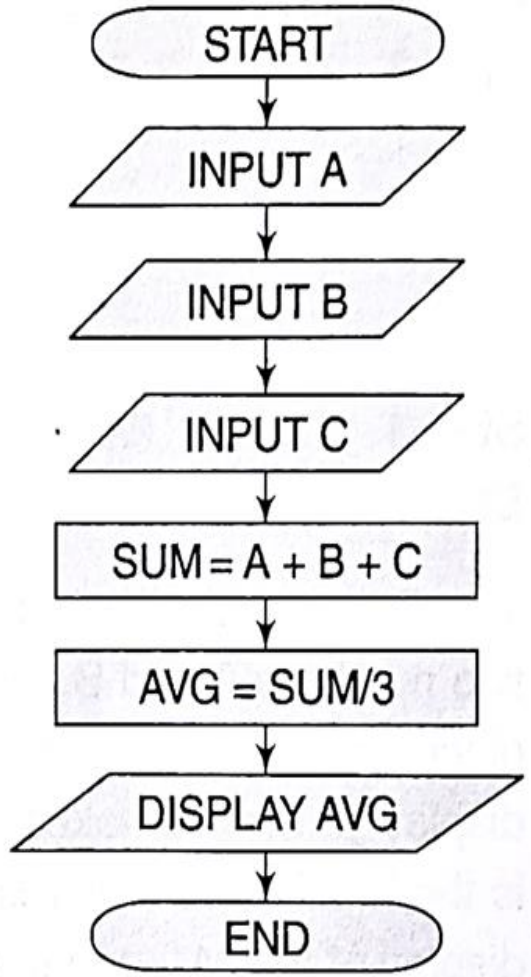
Problem – Develop an algorithm to find the average of three numbers taken as input from the user.

Algorithm AVERAGE

Step 0 START
Step 1 INPUT first number into variable A
Step 2 INPUT second number into variable B
Step 3 INPUT third number into variable C
Step 4 COMPUTE SUM = A + B + C
Step 5 COMPUTE AVG = SUM/3
Step 6 DISPLAY AVG
Step 7 END

The steps in the algorithm shown in the left match with the steps of the flowchart shown in the right.

Flowchart AVERAGE



Problem – Develop an algorithm to divide one number by another and find the quotient.

Algorithm DIVISION

Step 0 START

Step 1 INPUT first number into variable A

Step 2 INPUT second number into variable B

Step 3 IF B != 0 THEN

Q = A / B

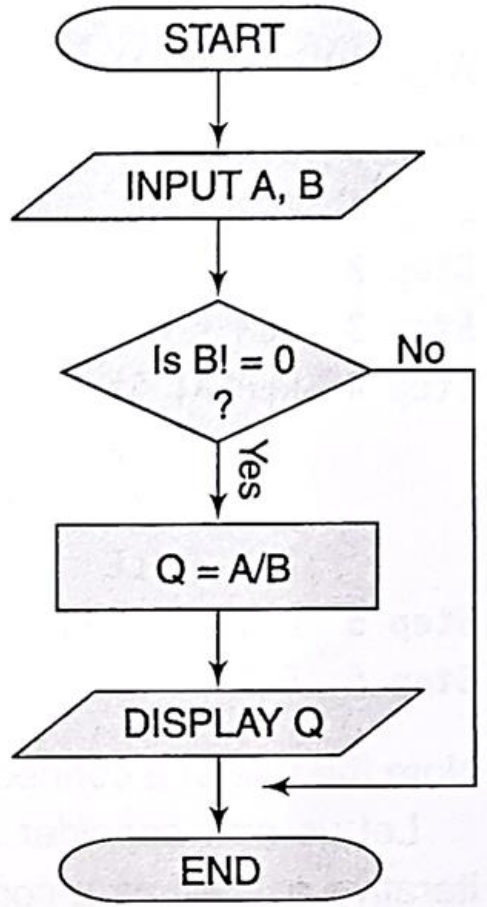
DISPLAY Q

END IF

Step 4 END

In the flowchart shown on the right, notice the merging of two input operations into one input box. Also notice the use of the decision box. The calculation of the quotient and the display of the same takes place along the 'Yes' path, i.e. when the condition 'B != 0' evaluates to true. In case the condition evaluates to false, the control directly moves to the terminal box labeled 'END'.

Flowchart DIVISION



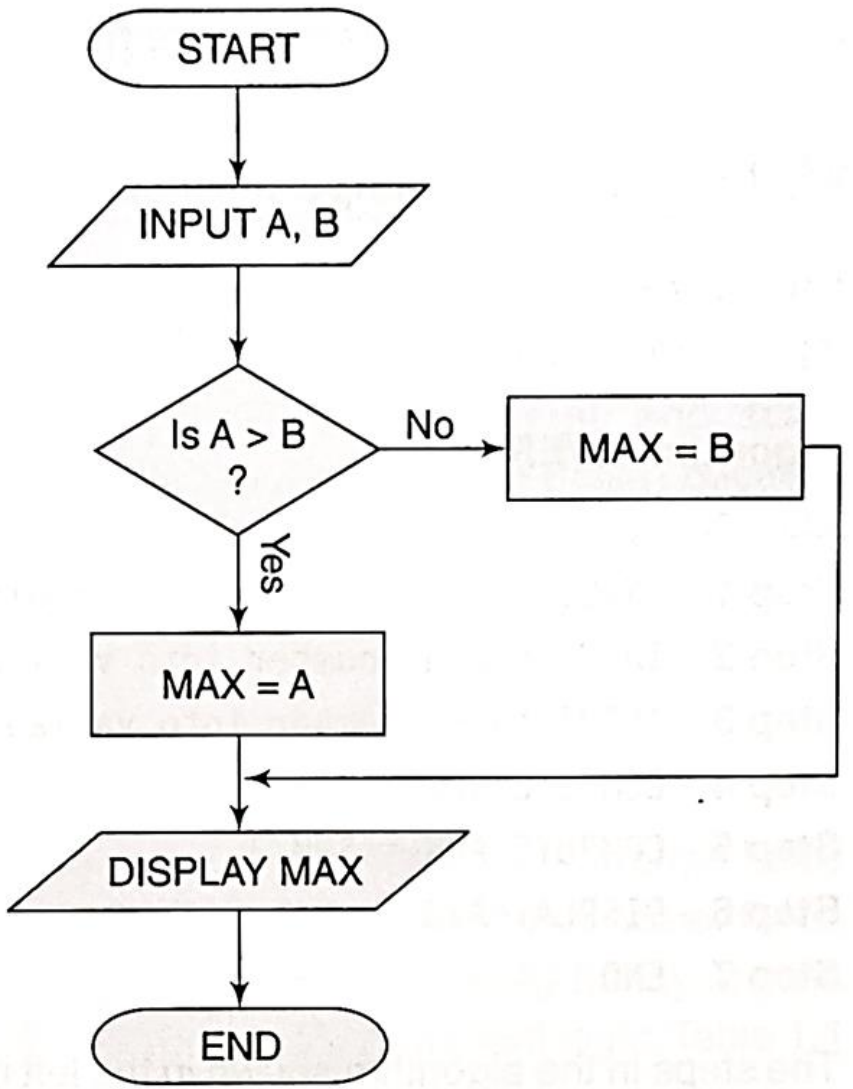
Problem – Develop an algorithm to find the maximum of two numbers input by the user.

Algorithm MAXIMUM

Step 0 START
Step 1 INPUT first number into variable A
Step 2 INPUT second number into variable B
Step 3 IF A > B THEN
MAX = A
ELSE
MAX = B
END IF
Step 4 DISPLAY MAX
Step 5 END

The flowchart shown on the right computes the maximum of two numbers A and B. Do note the separate actions that take place along the Yes and the No paths. Also note how the display of the MAX takes place irrespective of whether A or B is the maximum. The Yes and the No paths merge before the display of the MAX, and then finally the flowchart terminates.

Flowchart MAXIMUM



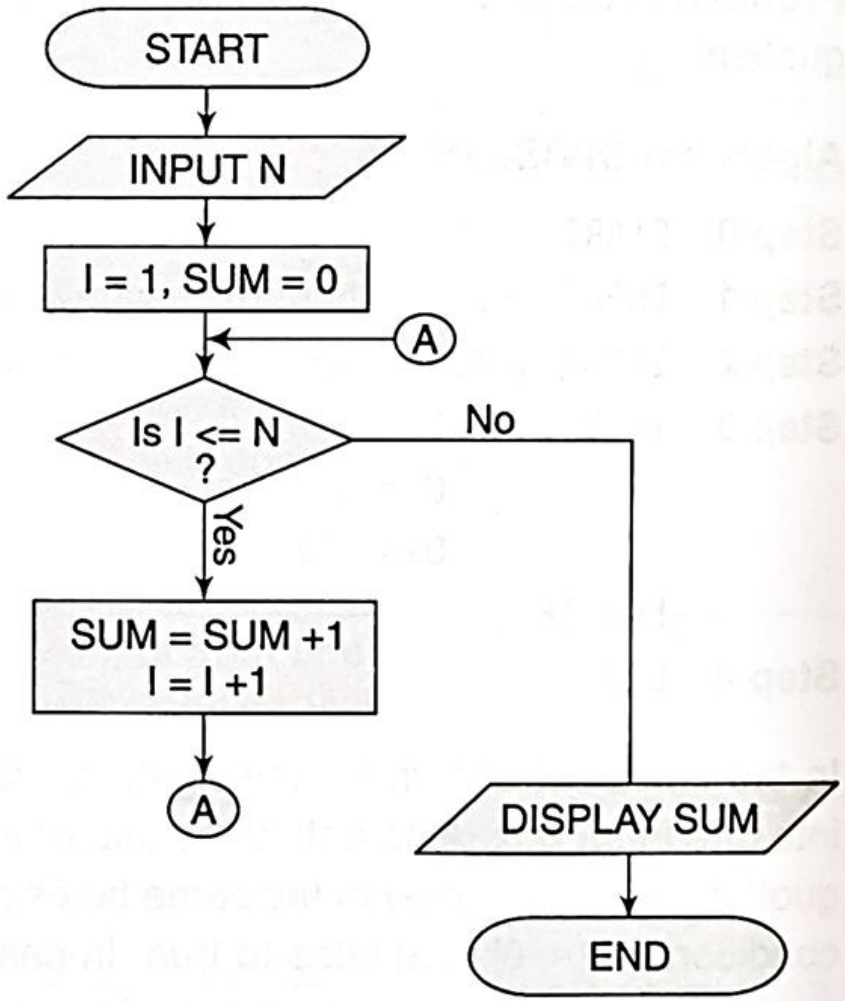
Problem – Develop an algorithm to find the sum of the first N natural numbers where N would be input by the user.

Algorithm SUM-N

```
Step 0  START
Step 1  INPUT N
Step 2  I = 1
Step 3  SUM = 0
Step 4  REPEAT Steps (a) – (b) WHILE I <=N
                (a) SUM = SUM + I
                (b) I = I + 1
        END WHILE
Step 5  DISPLAY SUM
Step 6  END
```

Note the use of a connector labeled, 'A' in this flowchart.

Flowchart SUM-N



3. Dry Run

- A dry run is a mental run of a computer program, where the computer programmer examines the source code one step at a time and determines what it will do when run.
- It is the mental run of an algorithm.
- A program can be checked without using a computer by dry running it on a paper.

Uses of Dry Run

- Catch errors before they cause problems
- Improve your understanding of the code
- Debug your code faster and easier
- Save time and resources
- Enhance your coding skills

Example for Dry Run

- L1 Declare two variables , first num second num
- L2 Initialise both variables to 0
- L3 first num = 0 second num = 0
- L4 Ask user to enter first number
- L5 Assign user input to first num variable
- L6 Ask user to enter second number
- L7 Assign user input to second num variable
- L8 Add first num to second num
- L9 Print result
- Do a Dry run for this program assuming the user enters 17 for first number and 24 for the second

After execution	First num	Second num	Result
Line 2	0	0	
Line 5	17	0	
Line 7	17	24	
Line 8	17	24	41