Abdelhafid Boussouf University
Center of Mila

Structure of Computers and Applications
1st year ST – ENGINEERING

Part 2: The basics of Algorithm and Program

Courses 4_5: The approtch and analysis of a problem in computer Concept of an Algorithm/Program

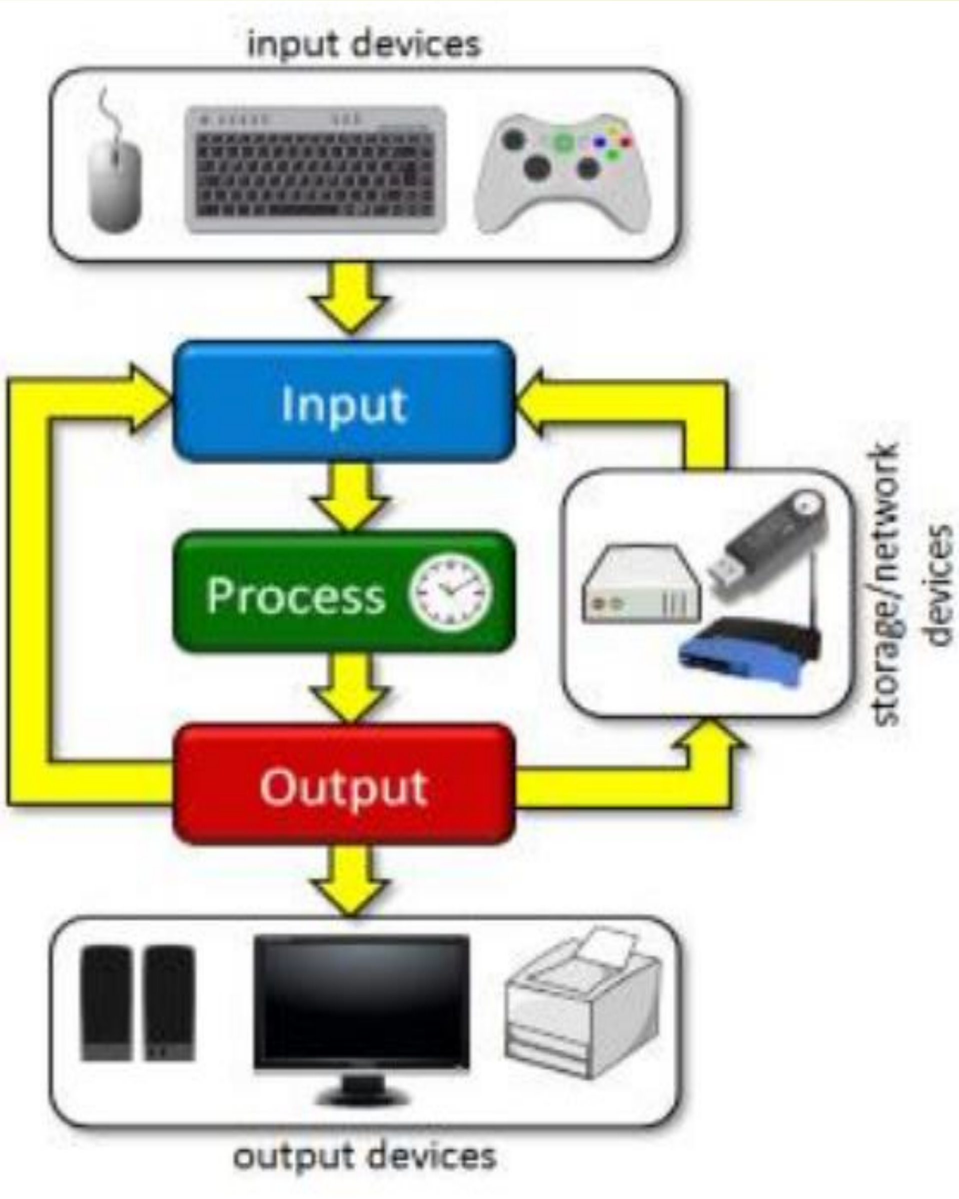
By

Dr. Farouk KECITA

Academic year : 2024/2025

5- The approatch and analysis of a problem in computer

- Computer science is all about **solving problems** with computers.
- The problems that we want to solve *can come from any* ***real-world*** *problem* or *perhaps even from the* ***abstract world***.
- We need to have a **standard systematic approach** to solving problems.
- Since we will be using computers to solve problems, it is important to first **understand** the computer's information processing model.

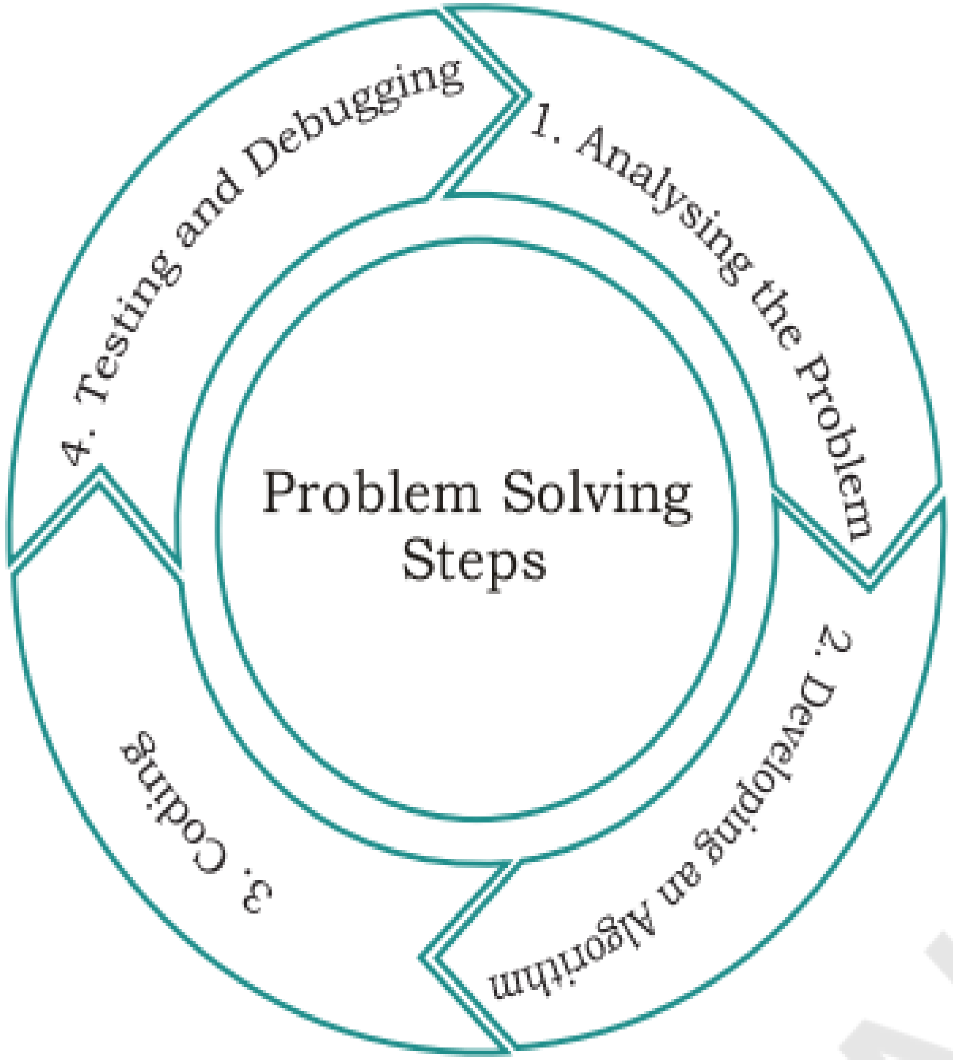


5- The approtch and analysis of a problem in computer

- ❑ **Problem Solving:** ***is the sequential process of analyzing information related to a given situation and generating appropriate response options.***

- ❑ There are ***six steps*** that you should follow in order to solve a problem:

- ➢ ***Understand the Problem***
- ➢ ***Formulate a Model***
- ➢ ***Develop an Algorithm***
- ➢ ***Write the Program***
- ➢ ***Test the Program***
- ➢ ***Evaluate the Solution***



6- Concept of an Algorithm/Program

The central concept underlying computation is that of the algorithm, a step-by-step sequence of instructions for carrying out some task.

- When we speak to people, we can assume that they understand certain basic facts of everyday life. For example, if we ask a person to buy a loaf of bread, we assume he/she knows how to do it.
- But computers are not people. If we were instructing a robot to buy a loaf of bread, we might have to be much more specific.
- When we instruct the computer to solve a problem, we must specify in detail all the steps in achieving the goal – an algorithm.
- Without the algorithm, the computer will not work.

6- Concept of an Algorithm/Program

6.1_ Algorithm

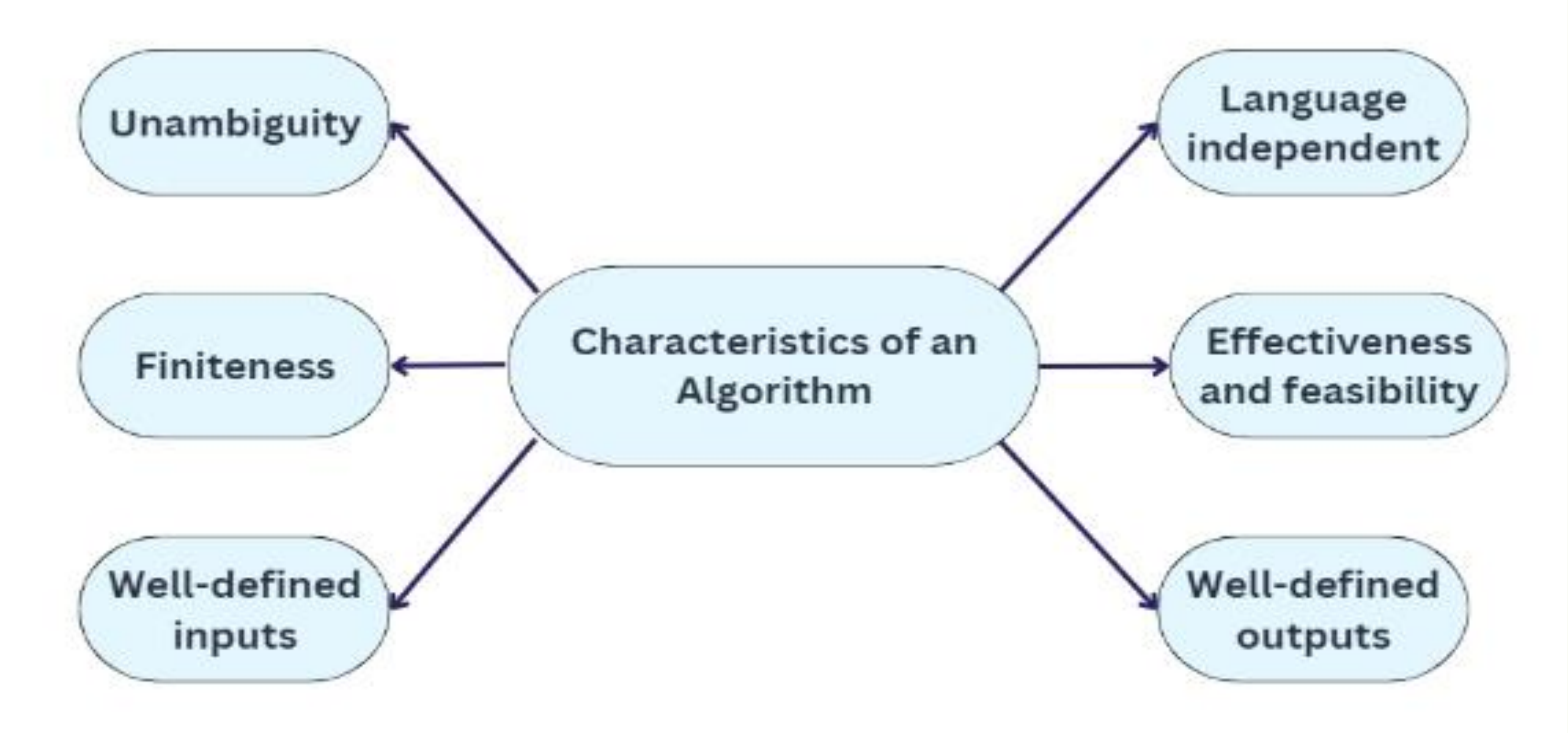
The word “algorithm” relates to the name of the mathematician **Al-khowarizmi**, which means a procedure or a technique.

- Software Engineer commonly uses an algorithm for **planning** and **solving** the problems. To write an algorithm, one must **know** how to **solve the problem**.
- An algorithm is a **sequence** of **steps** to **solve** a **particular problem** or algorithm is an ordered set of unambiguous steps that produces a **result** and terminates in a **finite time**.
- Algorithms need to be written **clearly** without **ambiguity**.
- Algorithms can be expressed in **many kinds** of notation, including **natural languages**, **pseudocode**, **flowcharts**, and **programming languages.**
- For each give problem, there may be **more** than one algorithm to solve the problem
- Some algorithms may be **faster than** others; some algorithms may require **different resources** (memory, special hardware, etc).

6- Concept of an Algorithm/Program

6.1_ Algorithm

What are the different characteristics of an algorithm?



6.1_ Algorithm

❑ HOW TO WRITE ALGORITHMS?

- Step 1: **Define your algorithms input**: Many algorithms take in data to be processed, e. to calculate the area of rectangle input may be the rectangle height and rectangle width
- Step 2: **Define the variables**: Algorithm's variables allow you to use it for more than on place. We can define two variables for rectangle height and rectangle width as HEIGHT and WIDTH (or H & W). We should use meaningful variable name e.g . instead of using H & use HEIGHT and WIDTH as variable name.
- Step 3: **Outline the algorithm's operations:** Use input variable for computation purpose e.g. to find area of rectangle multiply the HEIGHT and WIDTH variable and store the value in new variable (say) AREA.

  Step 4: **Output the results of your algorithm's operations**: In case of area of rectangle output will be the value stored in variable AREA. if the input variables described a rectangle with a HEIGHT of 3 and a WIDTH of 5, the output is 15.

6.1_ Algorithm

❑ Examples of Algorithm

Problem 01:

Write an algorithm to read two numbers and find their sum.

Inputs to the algorithm:

First Number1.

Second Number2.

Expected output:

Sum of the two numbers.

Algorithm:

Step1: Start

Step2: Read\input the first Number1 and the second Number2.

Step3: Sum= Number1+Number2 // calculation of sum

Step4: Print Sum

Step5: End

6.1_ Algorithm

❑ Examples of Algorithm

***Problem 02*:** Write an algorithm to find the value of A, B, C from the following equations: **A= X2+2Y , B = 2X-3A , C = A2-XB**

Where X and Y represents a circle area and circumference respectively.

Input the radius (R) and print the value of A, B and C.

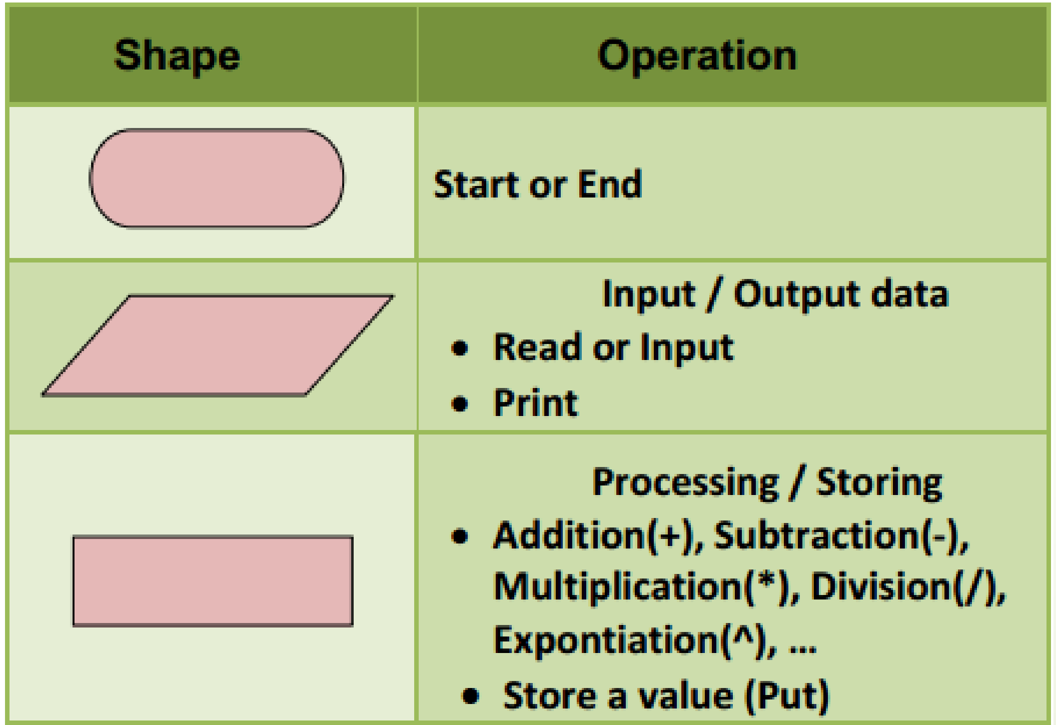
Algorithm:

```
Step1. Start
Step2. Input Radius (R)
Step3. Put PIE = 3.14
Step4. Find Area (X), X= R^2*PIE
Step5. Find Circumference (Y), Y= 2*R*PIE
Step6. Find A, A= X^2+2*Y
Step7. Find B, B=2*X-3*A
Step8. Find C, C=A^2-X*B
Step9. Print A, B, C
Step10.End
```

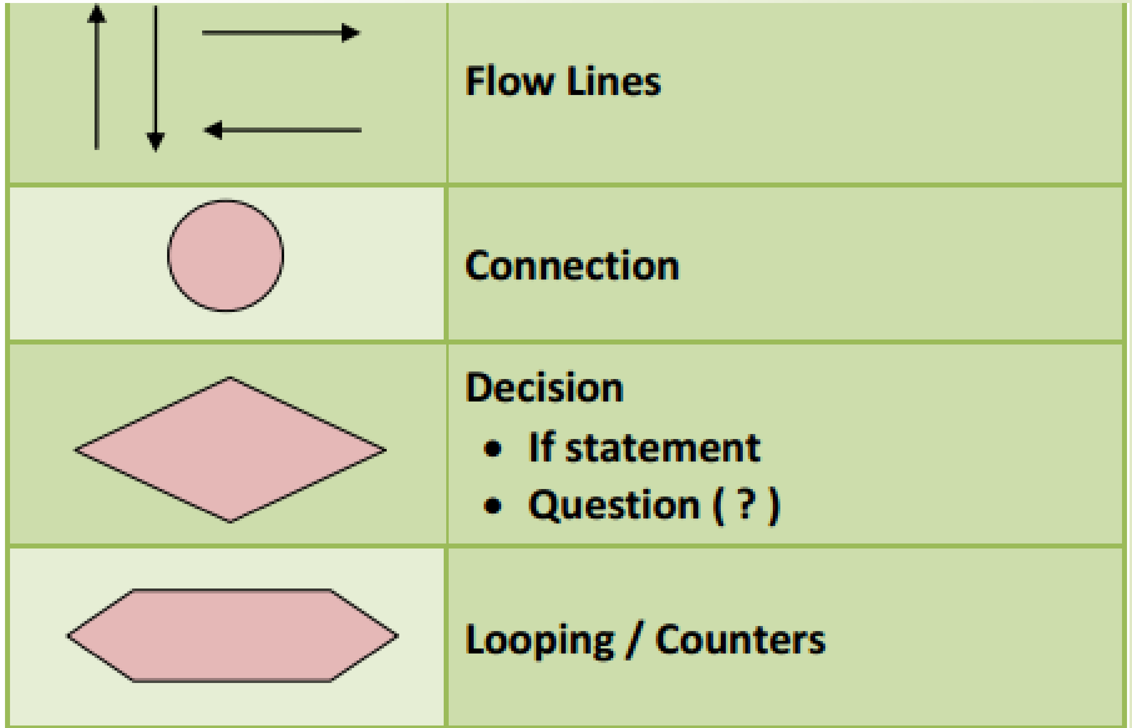
6.1_ Algorithm

❑ Representation of Algorithms (FLOWCHART)

- Is **diagrammatic** /**Graphical** representation of sequence of steps to solve a problem
- They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams.
- To **draw** a flowchart following standard symbols are use



Shape	Operation
⬭	Start or End
▱	Input / Output data • Read or Input • Print
▭	Processing / Storing • Addition(+), Subtraction(-), Multiplication(*), Division(/), Expontiation(^), ... • Store a value (Put)



↑ ↓ → ←	Flow Lines
○	Connection
◇	Decision • If statement • Question (?)
⬡	Looping / Counters

6.1_ Algorithm

❑ Representation of Algorithms (FLOWCHART)

➢ In general, we can divide flowcharts to a four shapes (charts):

1. Simple sequence charts
2. Branched charts.
3. Single loop charts.
4. Multi-loops (nested loops) charts.

➢ 1 .Simple sequence charts

The events arrangement of this type is as straight sequence from the beginning of the program to the end (Event-1 to Event-n), so this type of charts does not have any **branches** or **loops** (see figure (1-1)).

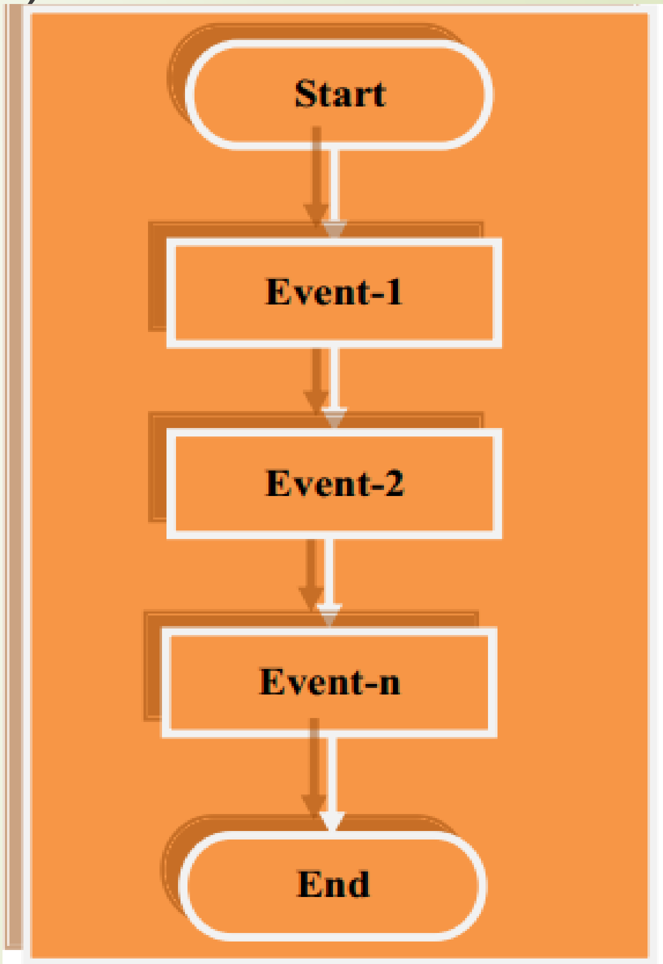


Figure (1-1) : A simple sequence chart

6.1_ Algorithm

1 .Simple sequence charts

Example:

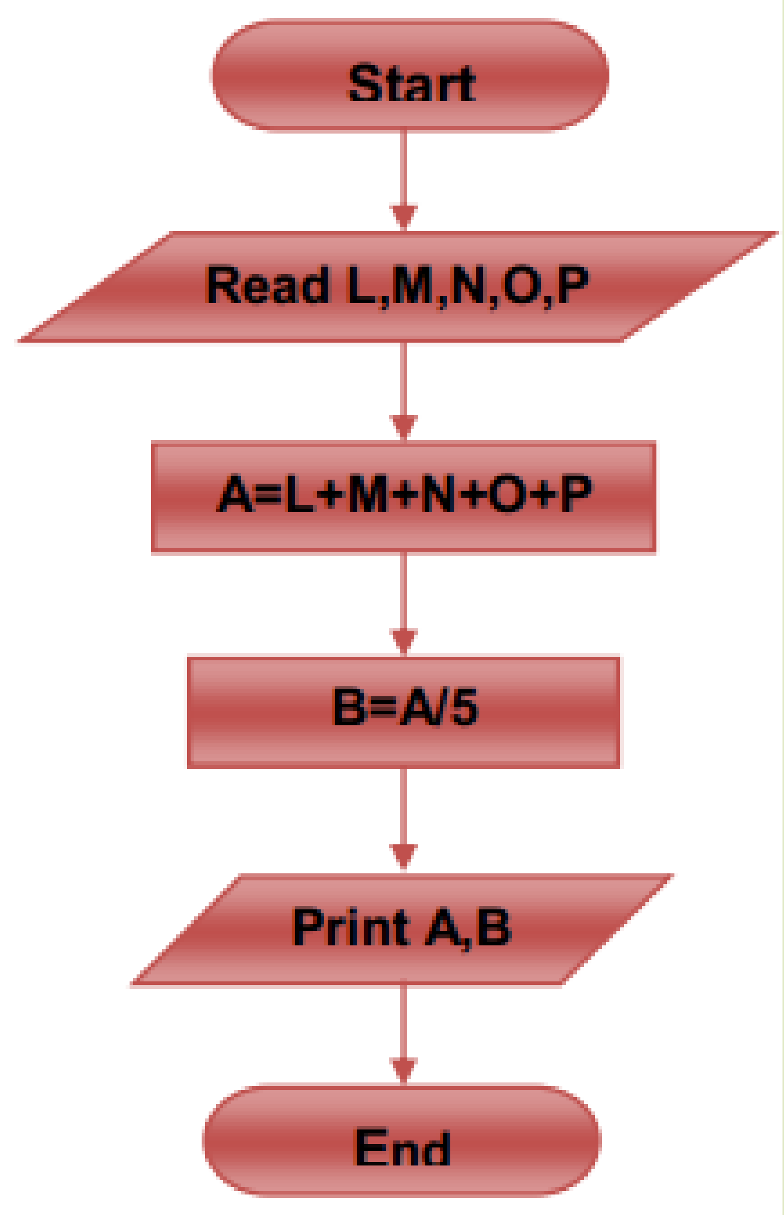
Write an algorithm and draw a flowchart to read five numbers and find their sum and average. Print the results.

Solution:

Algorithm:

```
Step1. Start
Step2. Read L, M, N, O, P
Step3. Find Sum (A) , A=L+M+N+O+P
Step4. Find Average (B) , B= A / 5
Step5. Print A, B
Step6. End
```

Flowchart:

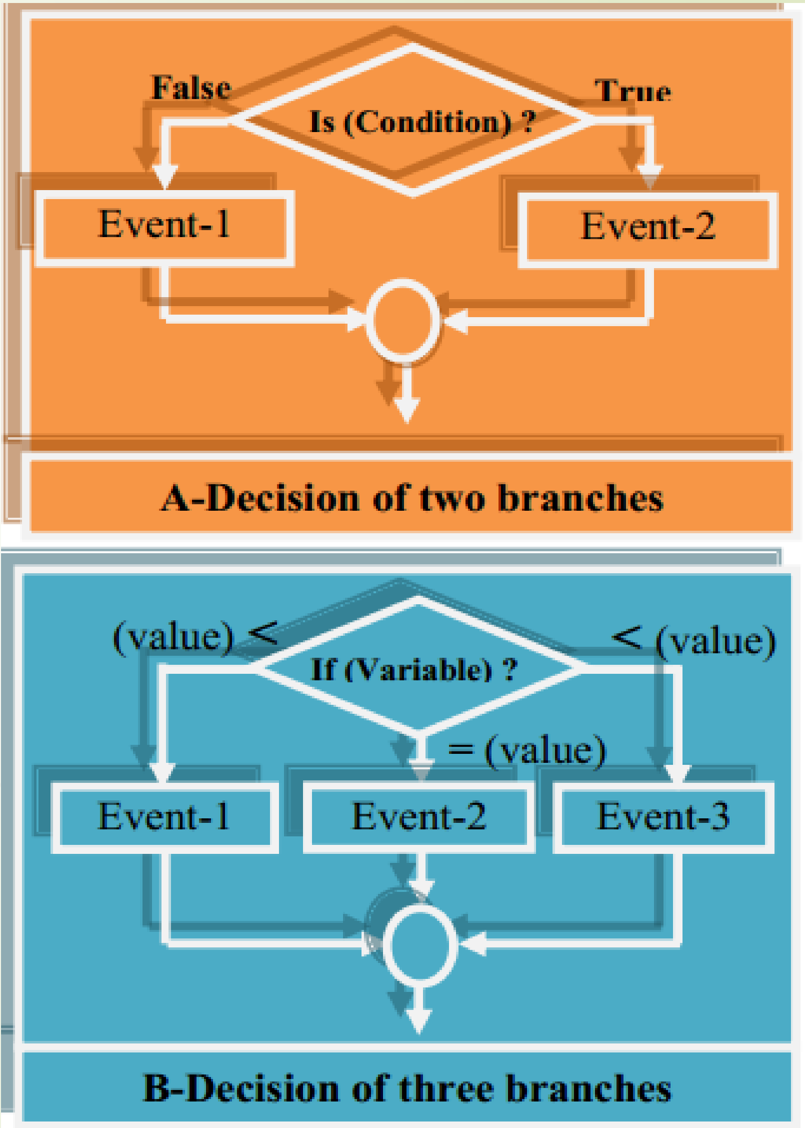


6.1_ Algorithm

2. Branched charts

- The need for the branching is to make decisions or comparison between two or more choices.
- Each choice will flow in different way (branch).
- Generally the branched charts may take one of the two forms shown in figure(2-1):

- **a. Decision of two branched**: The comparison in this type depends on: **Is (condition)** was satisfied (**True**) or not (**False**)
- **b. Decision of three branched**: The comparison in this type depends on: If (**variable**) was **equal(=), greater than(>)** or **less than(<)** any value?



A-Decision of two branches

B-Decision of three branches

Figure (2-1) : Branched charts

6.1_ Algorithm

➢ 2. Branched charts

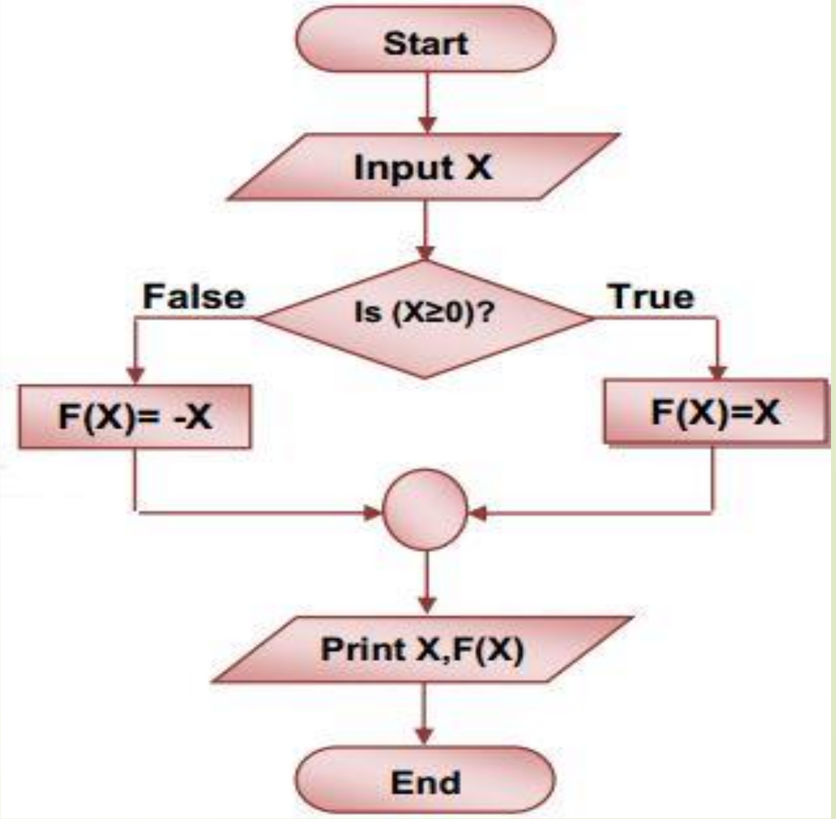
Example 01: Write an algorithm and draw a flowchart to find the value of the function F(X). Input X and print F(X) to each value of X.

$$F(X)=\begin{cases} X & :\text{if } X\geq 0 \\ -X & :\text{if } X<0 \end{cases}$$

Solution: Algorithm:

```
Step1. Start
Step2. Input x
Step3. Is $x \geq 0$ ?
        if "True" then continue.
        if "False" then go to step-5.
Step4. Find F(x), F(x) = x :Goto step-6
Step5. Find F(x), F(x) = -x.
Step6. Print x , F(x)
Step7. End
```

Flowchart:



6.1_ Algorithm

➢ 2. Branched charts

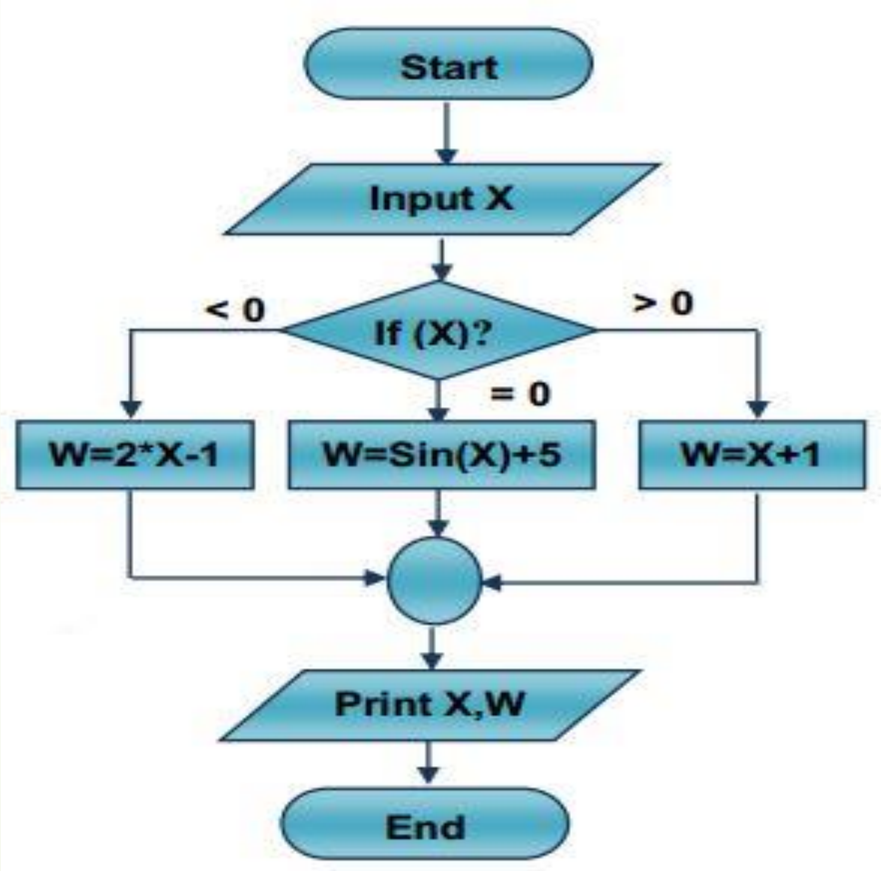
Example 02: Write an algorithm and draw a flowchart to evaluate W from the equation. Input X and print the value of W for each value of X.

$$W = \begin{cases} X+1 & : X > 0 \\ \mathrm{Sin}(X)+5 & : X = 0 \\ 2X-1 & : X < 0 \end{cases}$$

Solution: Algorithm:

```
step1. Start
step2. Input x
step3.   If x > 0 then continue
         If x = 0 then go to step-5
         If x < 0 then go to step-6
step4. Find W, W = X+1 : go to step-7
step5. Find W, W = Sin(X)+5: go to step-7
step6. Find W, W = 2*X-1
step7. Print X , W
step8. End
```

Flowchart:



6.1_ Algorithm

3. Single loop charts

- ✓ These charts are used when we need to **repeat** an operation or **group** of operations to specific number of times.
- ✓ These types of charts are used to **create the counters**.

What is counter?

Counter is used to repeat an operation or group of operations in specific number of times.

To make a counter we must know the following values:

- Counter name [literal value], (Let: I)
- Initial (Starting) [numerical value], (Let: S)
- End (Final) [numerical value], (Let: E)
- Step size [numerical value], (Let: Δ**)**

Counter can be designed using one of two forms :

A- Conventional form. **B- General form.**

6.1_ Algorithm

➢ 3. Single loop charts

A- Conventional form:

✓ This form is the simplest because all counter values (I, S, E, Δ) are mentioned in the same line (For I=S to E step Δ).

✓ At **starting** we use the looping shape (listed previously in table (1-1)) and at the **ending** use the **connection shape** putting a number inside it to know the **loop number**.

✓ The operation we want to **repeat** (which is containing **one** or **more** instructions) can be putted in between the **start** and **end** of the **counter**.

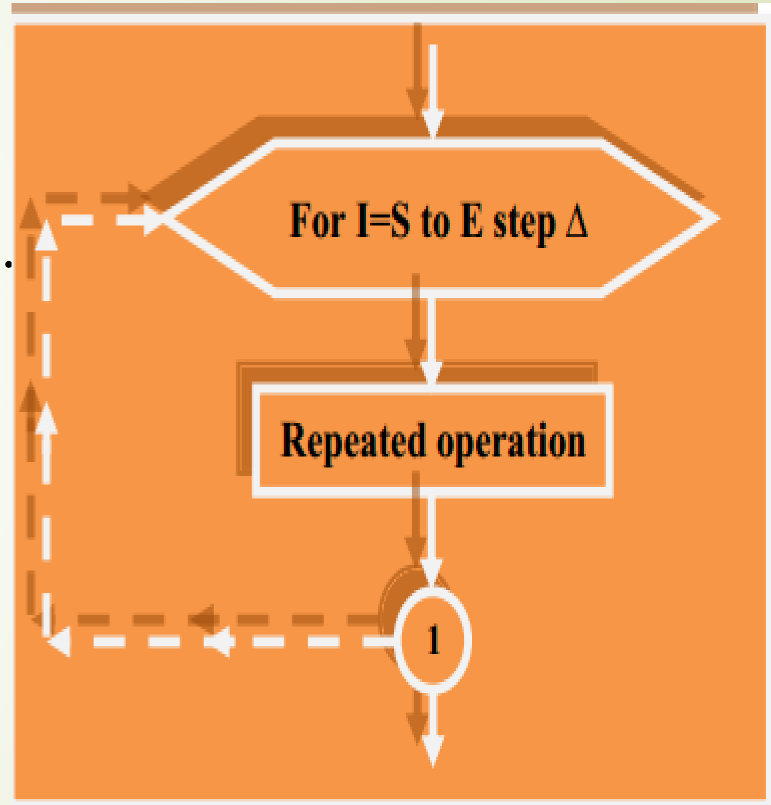


Figure (3-1) : Counter: Conventional-form charts

6.1_ Algorithm

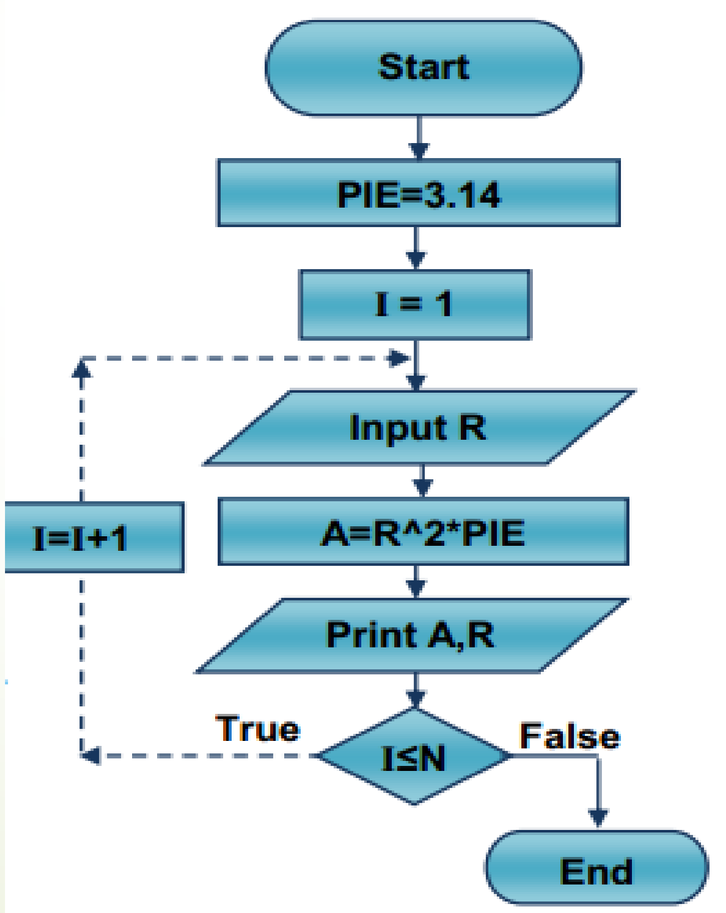
A- Conventional form:

Example : Write an algorithm and draw a flowchart to find the area of (N) circles. Input the circles and print the result. Use the general form

Solution: Algorithm:

```
1. Start
2. Put PIE = 3.14
3. Put I = 1
4. Input Radius (R)
5. Find Area (A), A=R^2*PIE
6. Print R , A
7. Is I ≤ N ?
If "True" Then I=I+1:Goto step-4
If "False" Then continue
8. End
```

Flowchart:



6.1_ Algorithm

➢ 3. Single loop charts

B- General form:

- ✓ This form is the complex because all counter values (I, S, E, Δ) are mentioned in the different line.
- ✓ At starting we put the starting value (I=S) and at the end we will put a condition to represent the end point (I ≥ E).
- ✓ The repeated operation will placed in between the start and end of the counter.
- ✓ A backward dashed line will return when the condition satisfied and in the middle of it the increasing (or decreasing) value will placed as (I=I+Δ) as shown in figure (4-1)

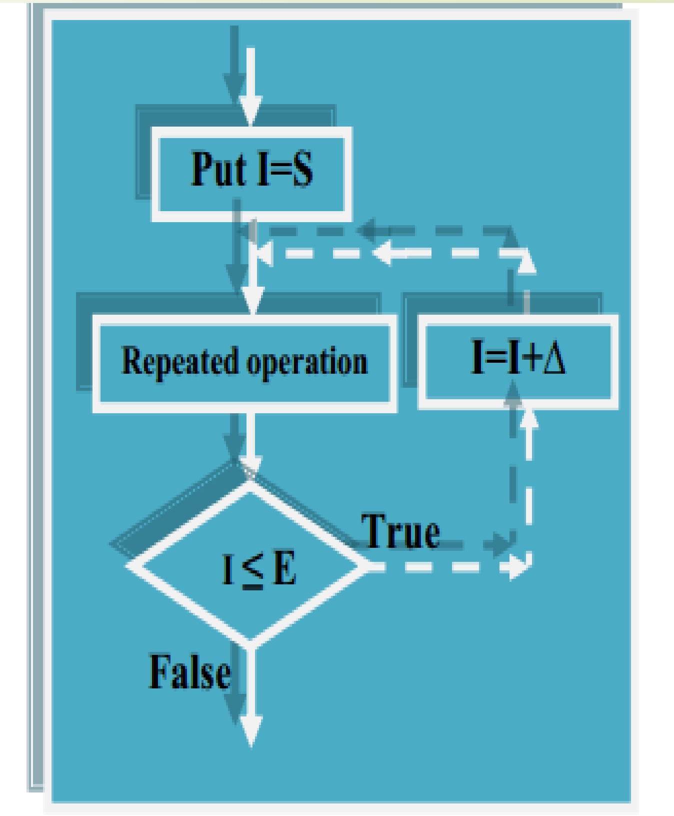


Figure (4-1) : Counter: General-form charts

Note: In conventional form if Δ=1 we can not write it but the genera form we write it .

6.1_ Algorithm

B- General form:

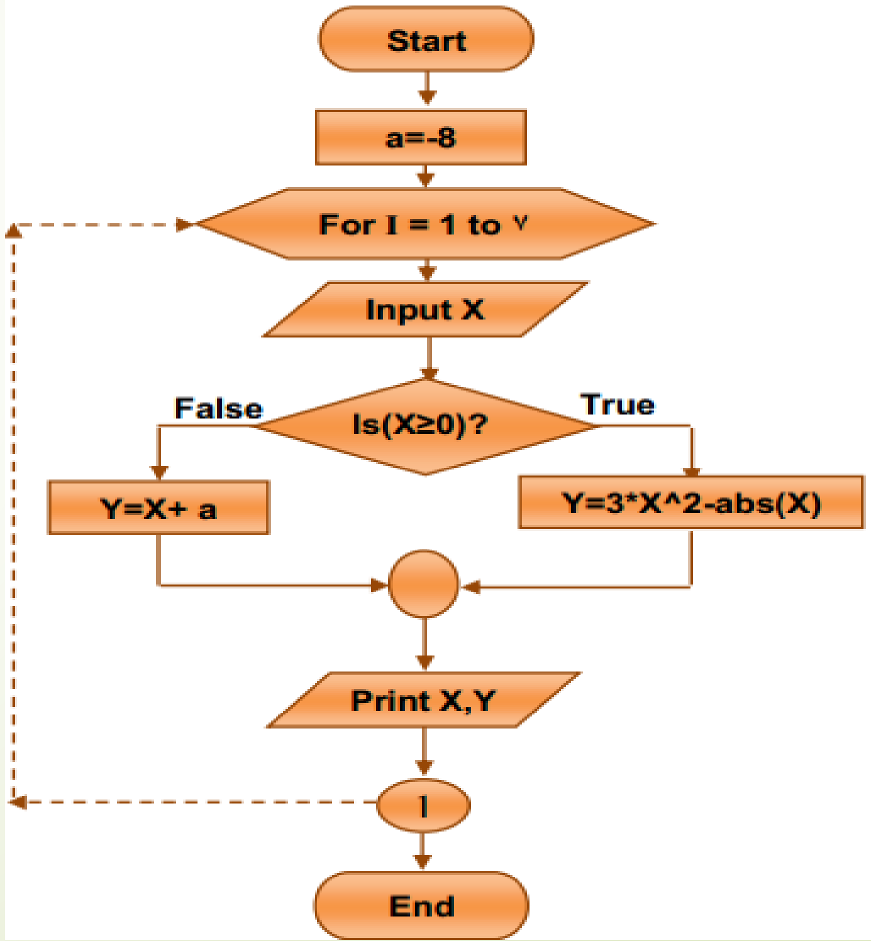
Example : Write an algorithm and draw a flowchart to evaluate Y from the equations for seven entering values of X. If you know that a = -8, print The value of Y for each value of X. Use the conventional form.

$$Y = \begin{cases} 3X^2 - |X| & : X \geq 0 \\ X + a & : X < 0 \end{cases}$$

Solution: Algorithm:

```
1.Start
2. Put a = -8
3. For I=1 To 7 step 1
4. Input X
5. Is X≥0 ?
If "True" then continue
If "False" : then go to Step-7
6. Find Y, Y=3*X^2-abs(X) : go to step-8
7. Find Y, Y= X+a
8. Print X,Y
9. Next I
10. End
```

Flowchart:



6.1_ Algorithm

4. Multi-loops (nested loops) charts

- ✓ Its so called because it contains many loops.
- ✓ These loops are nested together but without any intersections between these loops.
- ✓ As shown in figure (5-1), the loop number-1 is called "inner loop" and the loop number-2 is called the outer loop the priority of execution will be to the inner loops then sequentially to the outer loops.

Note:

The intersection will be caused when end

the outer loops before the inner or vice a versa..

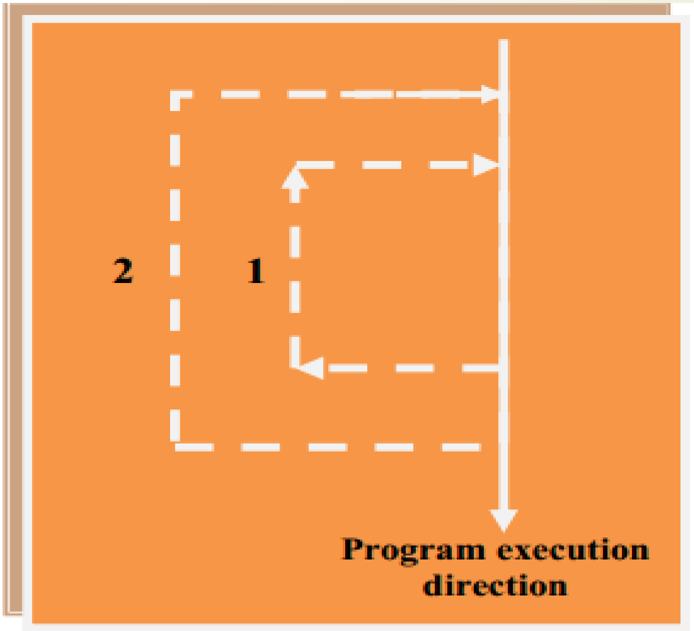


Figure (5-1) : Nested loops chart

6.1_ Algorithm

4. Multi-loops (nested loops) charts

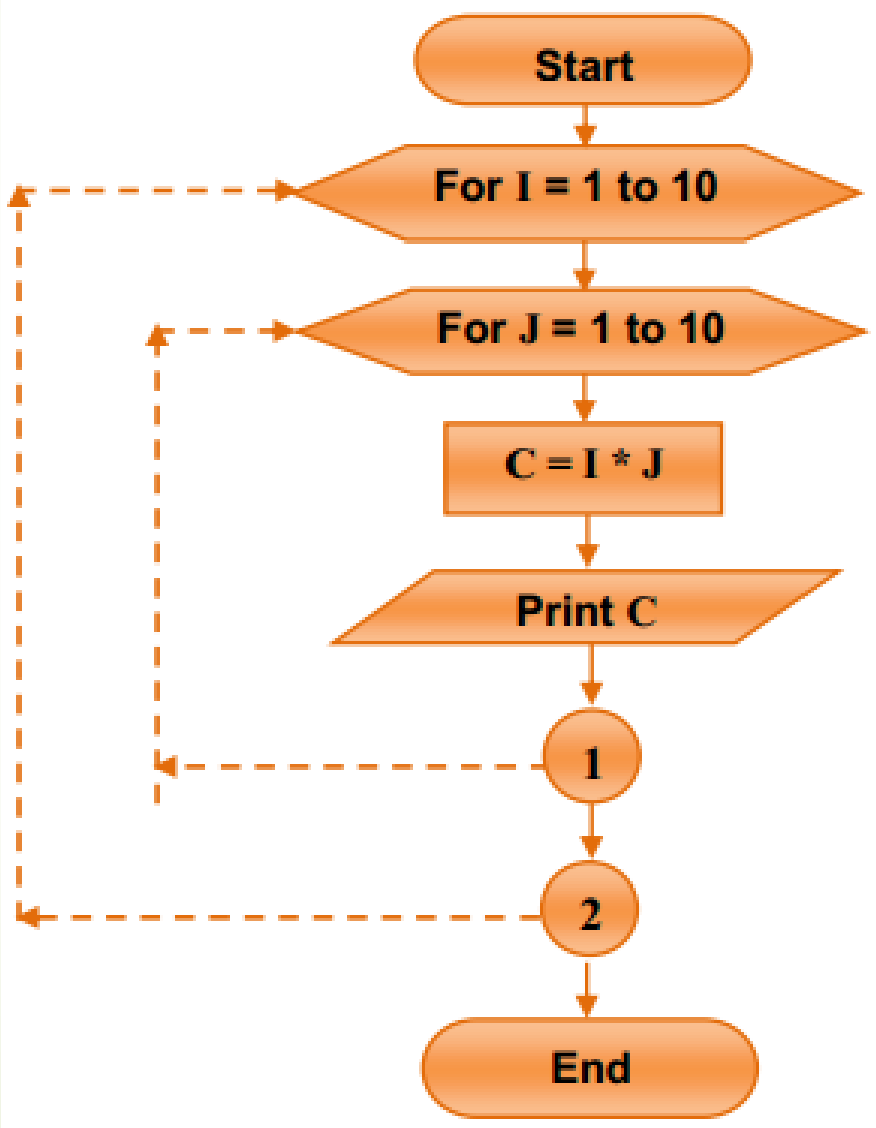
Example :

Write an algorithm and draw a flowchart to find an print the multiplication table from 1 to 10.

Solution: Algorithm:

```
1.Start
2. For I=1 to 10
3. For J=1 to 10
4. Find C, C= I * J
5. Print C
6. Next J
7. Next I
8. End
```

Flowchart:



6- Concept of an Algorithm/Program

6.2_ **Coding (Program)**

If you are to instruct the computer to accomplish certain task, you need to specify the detailed steps an then translate them into a **computer/programming language**

- A program is an algorithm for solving some problem which is written using some **computer/programming language**– so that it can be **understood** by the computer.
- A programming language is a set of **words** and **symbols** and **codes** that enables human to write a computer program.
- **Though similar**, the program and the algorithm are **not** the same:

1. The algorithm can be written using **human language** (English, Spanish, etc). But the computer does not understand it. The program must be written using **computer language** (C, C++, Python, Javascript, etc.). It can be understood by the computer. So an *algorithm needs to be* ***converted*** *to a program for the computer.*

2. The program follows **rigid formats** and **rules**. 3. Algorithms **predate** computers

6- Concept of an Algorithm/Program

6.2_ **Coding**

What are the steps involved in the creation and running of a program?

- **Writing** and **editing** the program using **Text editor** (source code).
- **Save** the code with an appropriate **file name** and **file extension**.
- **Compile** the program using any **compiler**, which **translates** the code into machine-**readable** instructions.
- Linking the program with the required library modules(object file)
- **Executing** the program. (. Exe file)

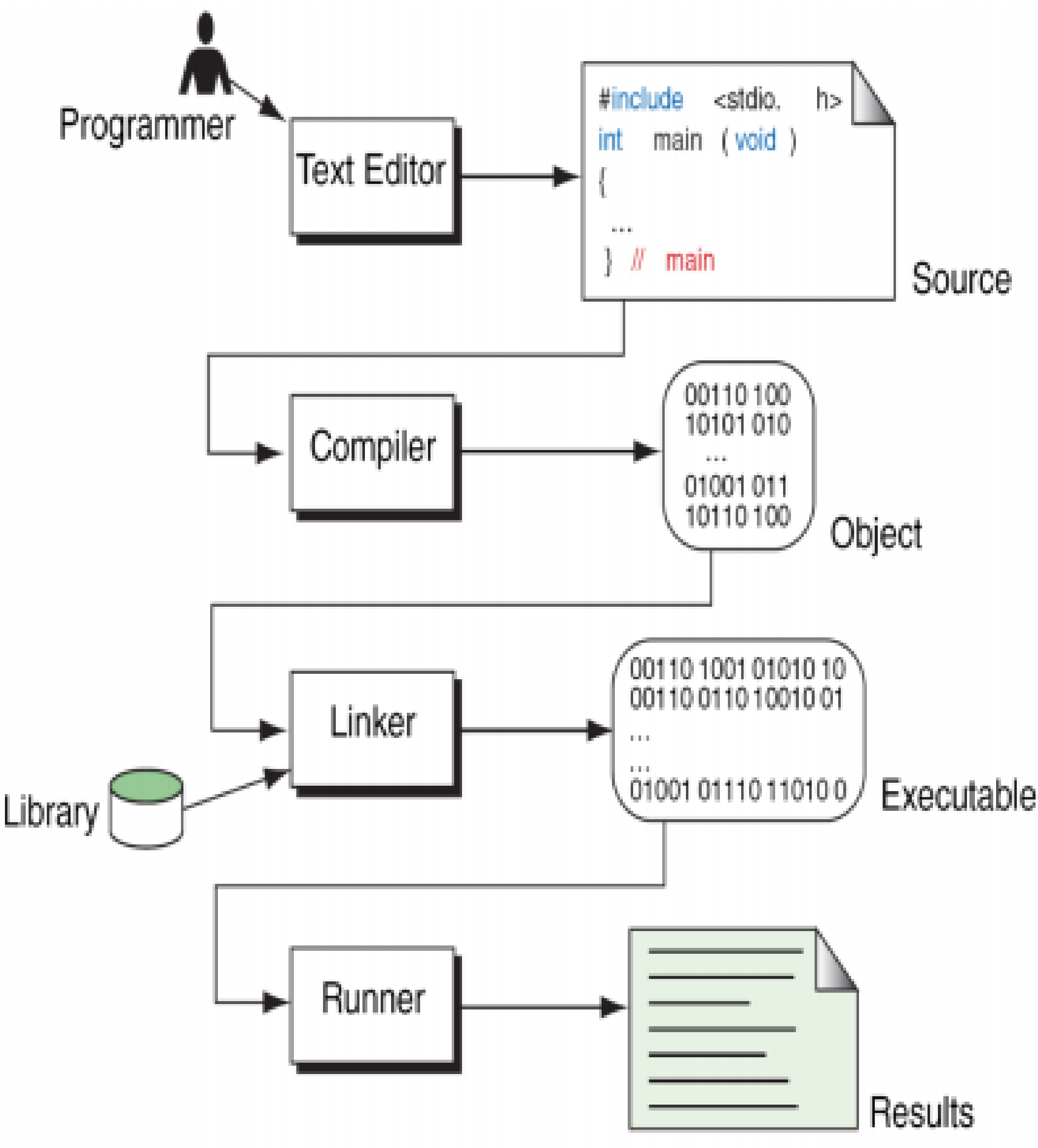
6- Concept of an Algorithm/Program

6.3_Executing program:

- Execution is the last step.
- In this step program starts execution.
- Its instructions start working

and output of the program display

on the screen.


Programmer
Text Editor
#include <stdio. h>
int main (void)
{
...
} // main
Source
Compiler
00110100
10101010
...
01001011
10110100
Object
Linker
Library
00110 1001 01010 10
00110 0110 10010 01
...
...
01001 01110 11010 0
Executable
Runner
Results