Algorithm

A logical sequence of steps for solving a problem is known as Algorithm.

Example of Algorithm to find area of a rectangle.

- Step 1: Input W,L
- Step 2: A ← L x W
- Step 3: Print A

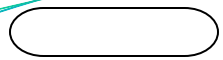
Properties of an Algorithm

- **Finiteness**
  - An algorithm must terminate after finite number of steps.
- **Definiteness**
  - The steps of the algorithm must be precisely defined.
- **Generality**
  - An algorithm must be generic enough to solve all problems of a particular class.
- **Effectiveness**
  - The operations of the algorithm must be basic enough to be put down on pencil and paper.
- **Input-Output**
  - The algorithm must have certain initial and precise inputs, and outputs that may be generated both at its intermediate and final steps.

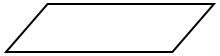
Flowchart

Flowchart is the graphical representation of an algorithm.

Flow chart symbols

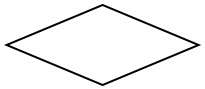


Oval — **Start/End/Terminal**



Parallegram — **Input/output**

Rectangle — **Process**



Diamond — **Decision**

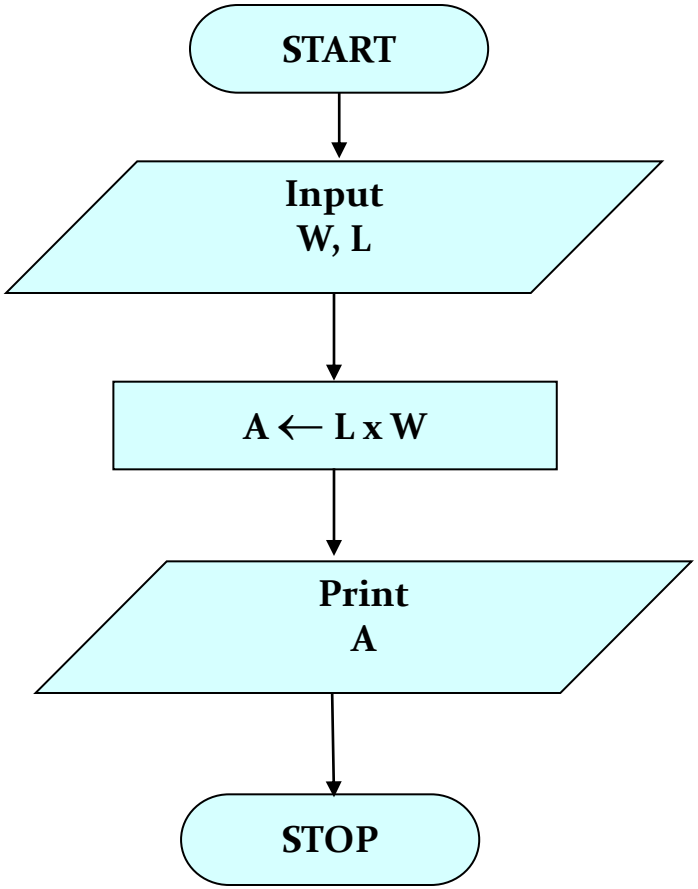
Circle — **Connector**

Arrow — **Flow Line**

Example to understand Algorithm and Flowchart

Algorithm

- Step 1: Input W,L
- Step 2: A ← L x W
- Step 3: Print A



Control Structure

A **control structure** is a block of programming that analyzes variables and chooses a direction in which to go based on given parameters.

There are three types of control structures:

- **1. Sequence logic, or Sequential flow**
- **2. Selection logic, or Conditional flow**
- **3. Iteration logic, or Repetitive flow**

Sequence Logic or Sequential Flow

Algorithm

:

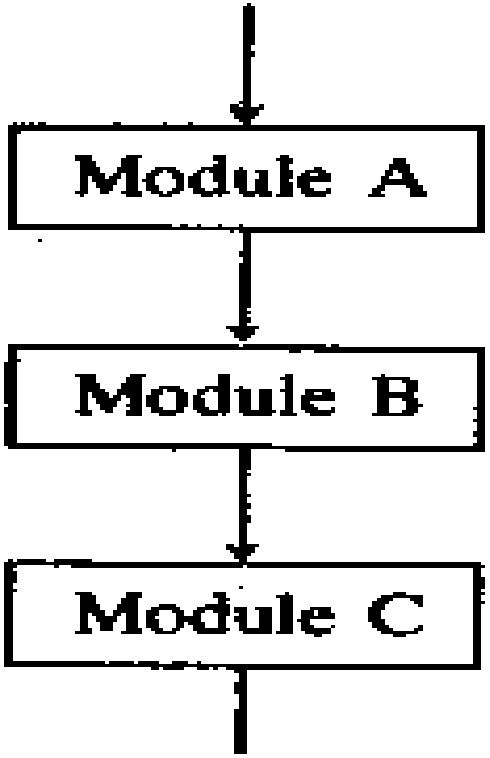
Module A

Module B

Module C

:

Flowchart equivalent

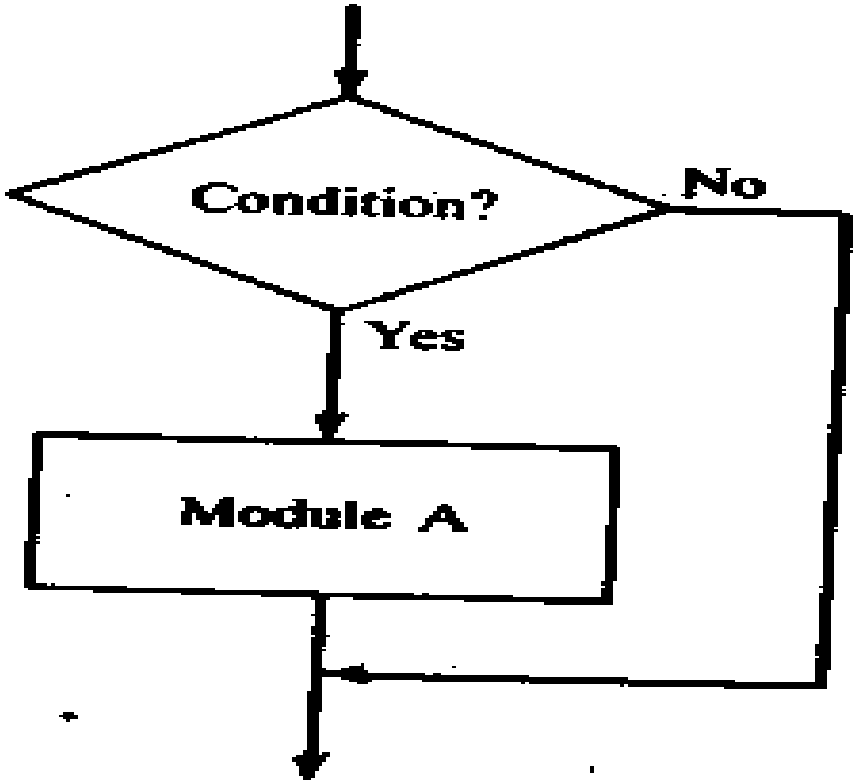


Selection logic/Conditional flow

1. Single Alternative (if)
2. Double Alternative (if-else)
3. Multiple Alternative (else-if/if-else if-else)

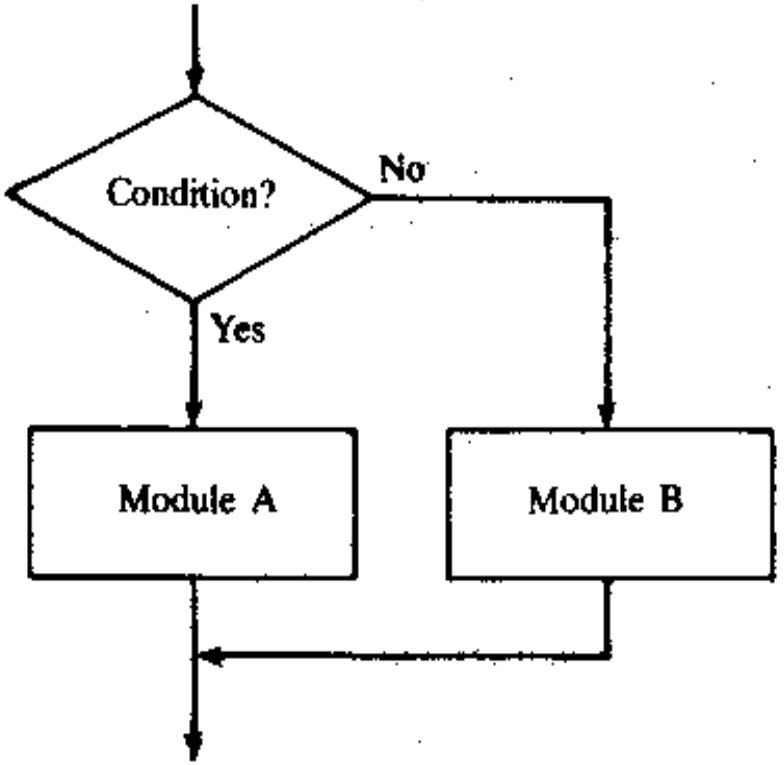
Single Alternative Syntax & Flowchart

If condition, then:
[Module A]
[End of If structure]



Double Alternative Syntax & Flowchart

If condition, then:
    [Module A]
Else:
    [Module B]
[End of If structure]



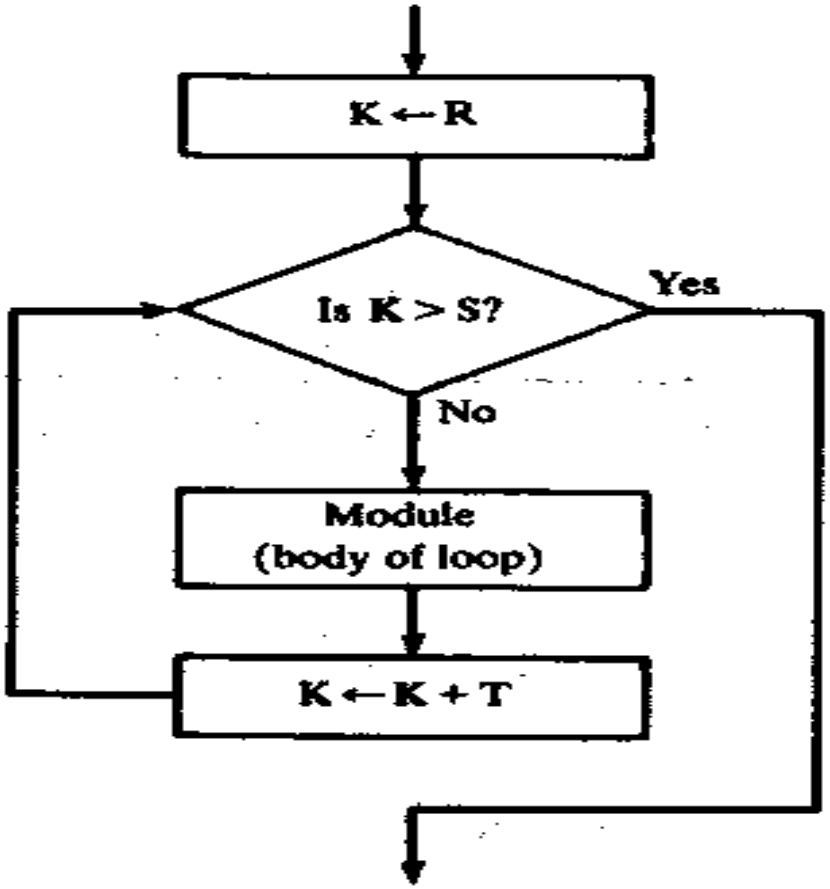
Multiple Alternative Syntax

If condition(1), then:
[Module A1]
Else if condition(2), then:
[Module A2]
.....
.....
Else if condition (M), then:
[Module A_M]
Else:
[Module B]
[End of If structure]

Iteration Logic (Repetitive Flow)

Repeat for loop

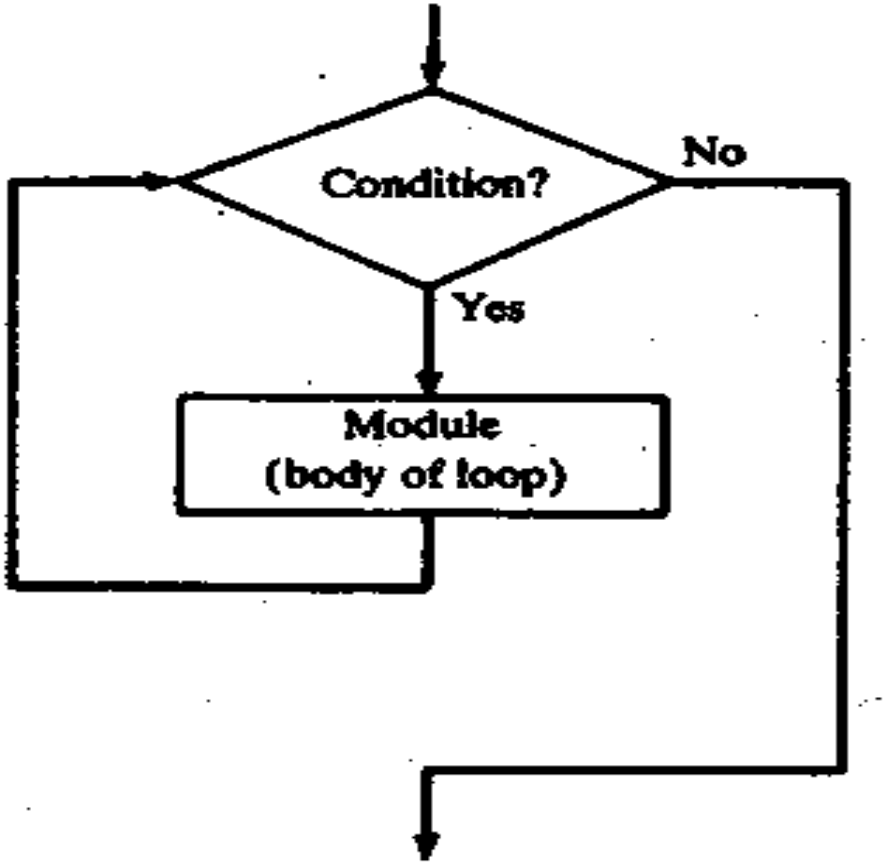
Repeat for K=R to S by T:
    [Module]
[End of loop]



Iteration Logic (Repetitive Flow)

Repeat While loop

Repeat while condition:
    [Module]
[End of loop]



Algorithm to find the roots of Quadratic Equation

Step 1: Read: A,B,C.

Step 2: Set D: $B^2 - 4AC$.

Step 3: If D>0, then :

a) Set $X1 := (-B + \sqrt{D})/2A$ and $X2 := (-B - \sqrt{D})/2A$.

b) Write: X1,X2.

Else if D=0, then :

a) Set X:= -B/2A.

b) Write: "UNIQUE SOLUTION",X.

Else:

Write: "NO REAL SOLUTIONS".

[End of If structure.]

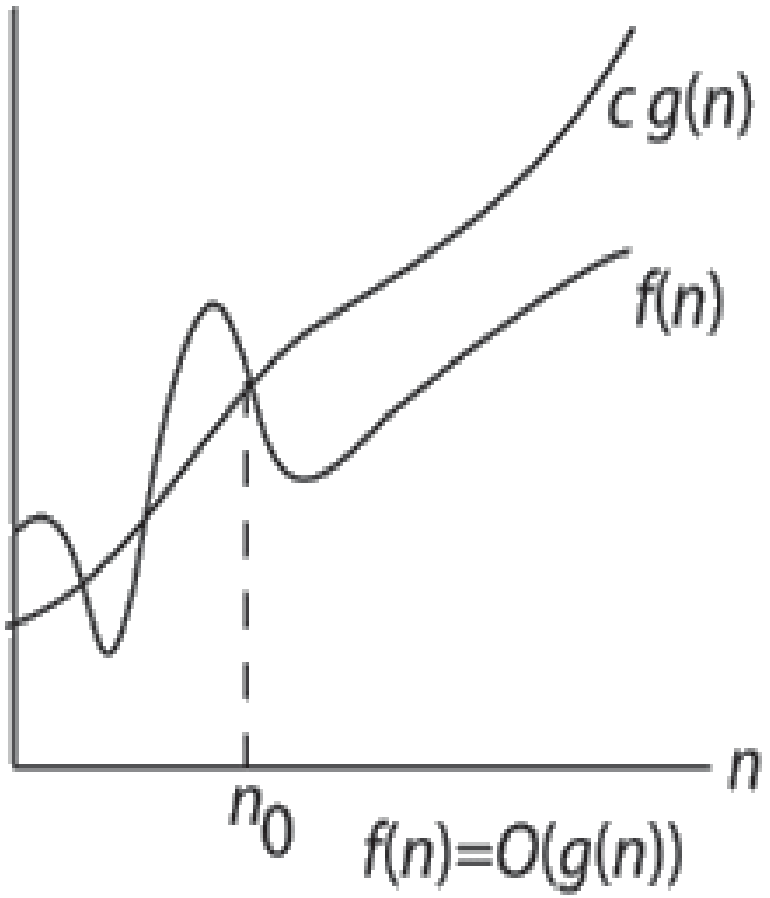
Step 4. Exit.

Algorithm Analysis and Complexity

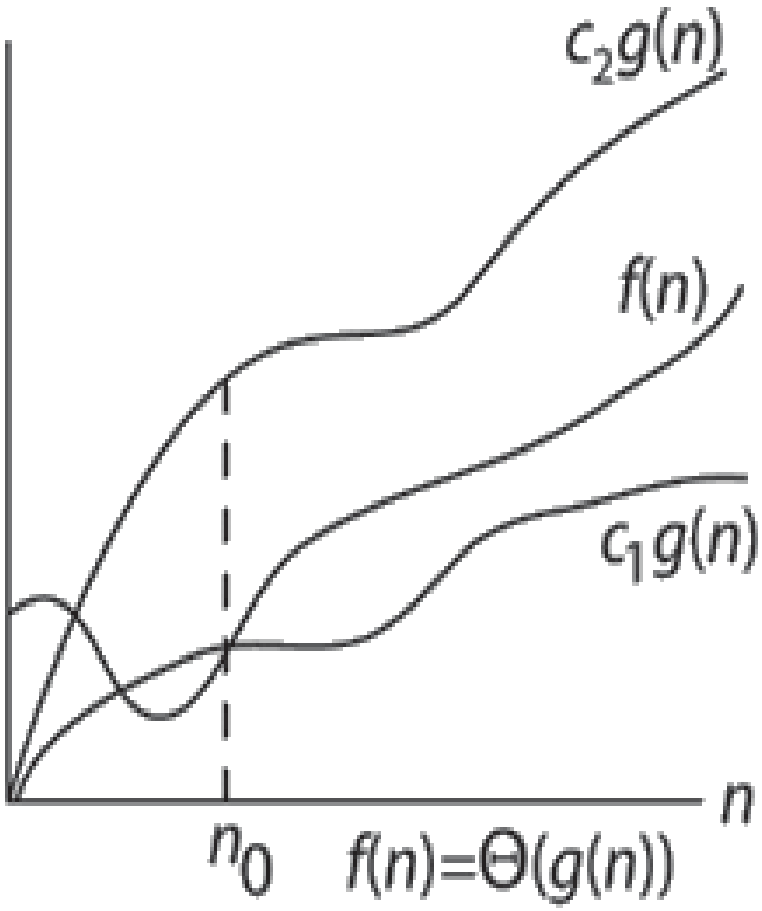
- The performances of algorithms can be measured on the scales of **Time and Space**.
- The **Time Complexity** of an algorithm or a program is a function of the running time of the algorithm or a program.
- The **Space Complexity** of an algorithm or a program is a function of the space needed by the algorithm or program to run to completion.

Analysis of Algorithms

- Time complexity
  - The amount of time that an algorithm needs to run to completion
- Space complexity
  - The amount of memory an algorithm needs to run
- We will occasionally look at space complexity, but we are mostly interested in time complexity in this course.
- Thus in this course the better algorithm is the one which runs faster (has smaller time complexity)



(a)



(b)

Big Oh Notation

- Computer scientists use a notation to represent an algorithm's complexity.
- To say "Algorithm A has a worst-case time requirement proportional to *n*"
  - We say A is O(n)
  - Read "Big Oh of *n*" or "order of at most n"
- For the other two algorithms
  - Algorithm B is $O(n^2)$
  - Algorithm C is O(1)