# Flowcharts

## INTRODUCTION

Here are the steps that may be followed to solve an algorithmic problem:

- Analyzing the problem statement means making the objective of the program clear in our minds like what the input is and what is the required output.
- Sometimes the problems are of complex nature, to make them easier to understand, we can break-down the problem into smaller sub-parts.
- In order to save our time in debugging our code, we should first-of-all write down the solution on a paper with basic steps that would help us get a clear intuition of what we are going to do with the problem statement.
- In order to make the solution error-free, the next step is to verify the solution by checking it with a bunch of test cases.
- Now, we clearly know what we are going to do in the code. In this step we will start coding our solution on the compiler.

Basically, in order to structure our solution, we use flowcharts. A flowchart would be a diagrammatic representation of our algorithm - a step-by-step approach to solve our problem.

## Flowcharts

### Uses of Flowcharts

➔ Used in documentation.
➔ Used to communicate one's solution with others, basically used for group projects.
➔ To check out at any step what we are going to do and get a clear explanation of the flow of statements.

### Flowchart components

- **Terminators**


Mainly used to denote the start point of the program.

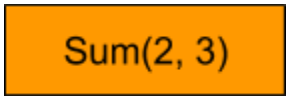

Used to denote the end point of the program.

- **Input/Output**


Used for taking input from the user and store it in variable ‘var’.


Used to output value stored in variable ‘var’.

- **Process**




Used to perform the operation(s) in the program. For example: Sum(2, 3) just performs arithmetic summation of the numbers 2 and 3.

- **Decision**


Used to make decision(s) in the program means it depends on some condition and answers in the form of TRUE(for yes) and FALSE(for no).

- **Arrows**

Generally, used to show the flow of the program from one step to another. The head of the arrow shows the next step and the tail shows the previous one.

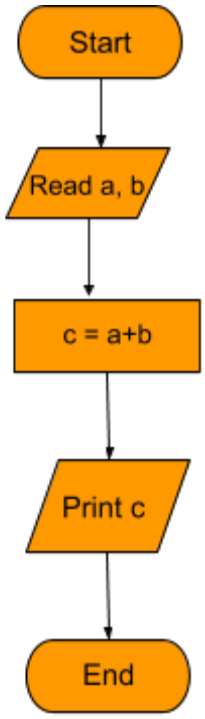
- **Connector**


Used to connect different parts of the program and are used in case of break-through. Generally, used for functions(which we will study in our further sections).

### Example 1:

Suppose we have to make a flowchart for adding 2 numbers a and b.

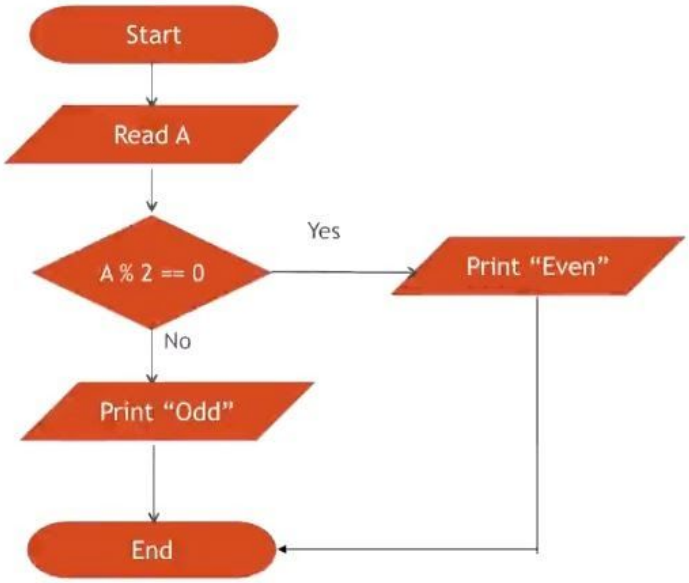
### Solution:




### Example 2:

Suppose we have to check if a number is even or odd.

### Solution:



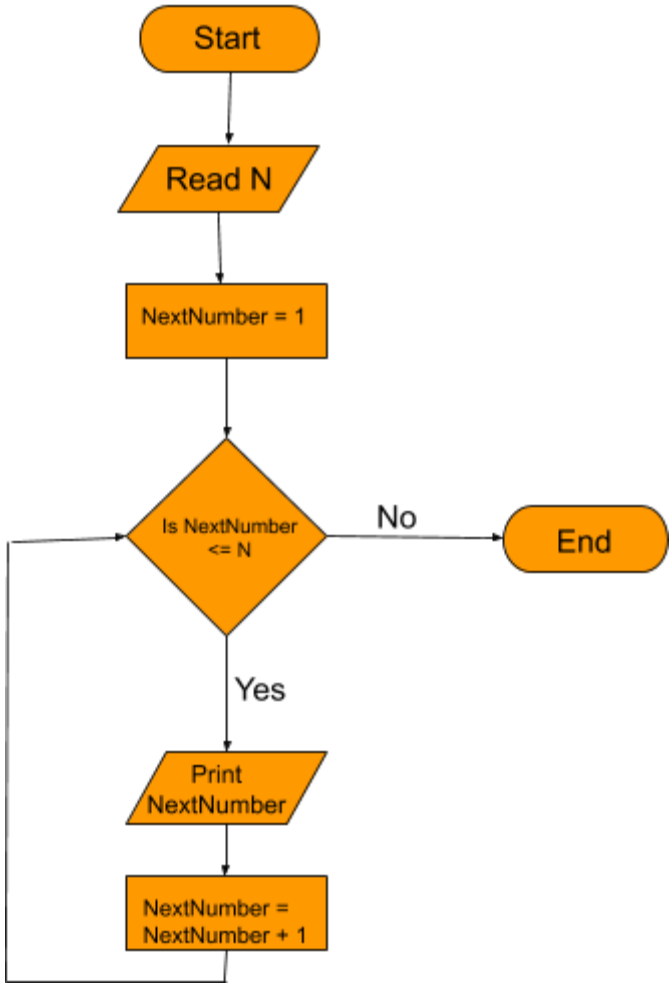

**Note:** Operator % is used to find the remainder of a number with respect to another number. For example:

- 4 % 3 = 1
- 10 % 5 = 0
- 4 % 2 = 0
- 4 % 4 = 0
- 0 % 1 = 0
- 1 % 0 = undefined (as it leads to division by 0)

## Example 3:

To print the numbers less than or equal to n where n is given by the user.

## Solution:




## Summary

- Flowcharts are the building-block of any program written in any language.
- Different shapes used to have different meanings.
- Every problem can be represented in the form of a flow chart.

- Sometimes, it becomes a bulky process to represent any program using flowchart. In those cases, try to find out the optimal solution to the given problem.

## Practice Problems

Till now in examples we learnt how to draw decision making blocks and how to manage looping the same part again and again until the condition is not satisfied.

Using the concepts shown above, here are few tricky flowchart problems for your practice :

- **Check if a given number, N is positive or negative.**
- **Find the average of 3 given numbers.**
- **Given 3 numbers, check whether a valid triangle can be formed using these numbers or not. Print YES or NO.**
- **Given a number N, print N! (N factorial).**
- **Given a number N, print even numbers upto N.**
- **Given a number N, check if it is prime or not. Print YES or NO.**