Introduction to coding

Steps in solving a problem using computers

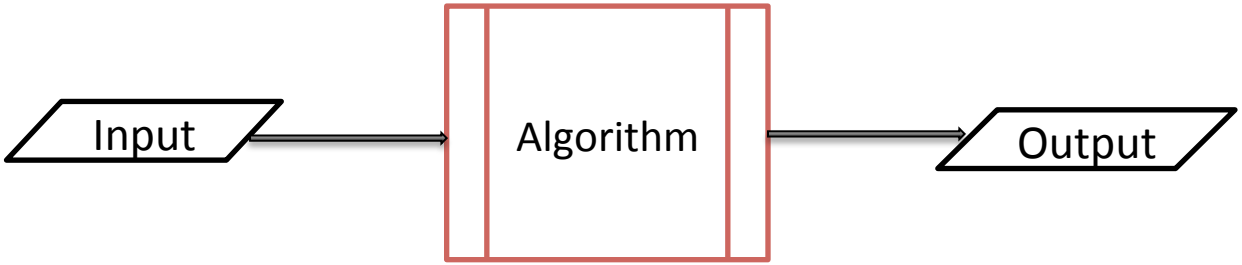
Computers enable us to efficiently solve routine human task or day-to-day problems such as ticket booking, scheduling of meeting etc.

Computer-based solving process:

- Analyze the problem to define it's principal components with their core functionalities. (define input/output)
- Develop an algorithm (design of solution to a problem)
- Convert algorithm into a program using a programming language also called as Coding
- Test and debug the computer program before deployment of software.

Algorithm

It can defined as a sequence of rules/instructions to be followed in solving a problem. It is a clearly defined sequence of steps to be followed in a computer program or a roadmap for programmer to code instructions for computer.



Characteristics of a good algorithm:

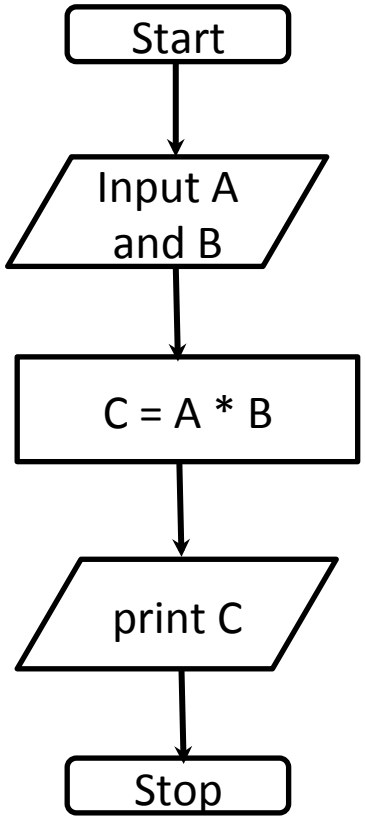
- **Well defined Input(s) and output(s)**: It should have defined input(s) and output(s).
- **Unambiguous or precision**: Steps are precisely defined or stated.
- **Uniqueness**: At each step, the results are uniquely defined and depends on its preceding inputs.
- **Finiteness**: The algorithm must be finite or it should stop after a finite number of steps.
- **Language independent**: It must be plain instructions, which can be implemented in any programming language.

Introduction to coding

Algorithm

Algorithms can be represented using **FLOWCHART** or **PSEUDOCODE.** This shows the logic of the code without implementation details!!

Example: Write algorithm to multiply two numbers and print output



Flowchart

START
INPUT a
INPUT b
COMPUTE c=a*b
PRINT c
END

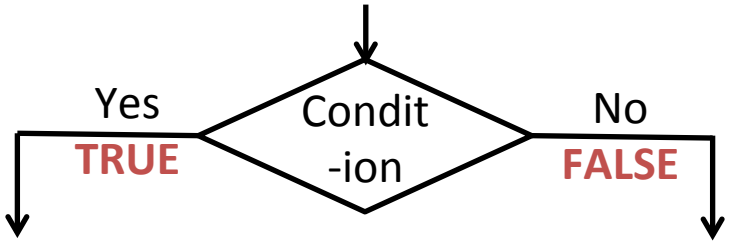
Pseudocode

Algorithm

Flow of sequence can be branched based on a **Decision** or needs to be **Repeated** for a finite number times.

Example: Find sum of all odd numbers from 1 to 100.

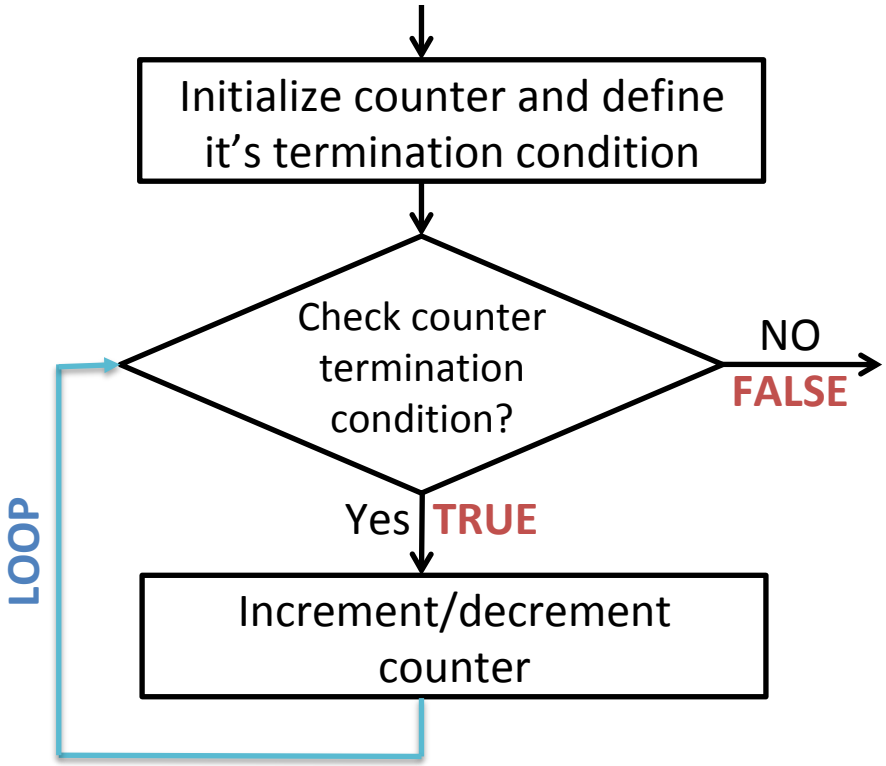
Conditionals: are used to check possibilities. Program can check for one of more conditions, which returns either TRUE or FALSE. Depending of either of these, next set of sequences. Multiple options for a condition can be checked by 'If-elif-else' conditions



If <condition> then
    sequence of steps when the condition is **TRUE**
otherwise:
    steps when condition is **FALSE**

Algorithm

Repetition: A task which needs to be repeated for finite number of times. The repetitions can be of a exact known number of steps or unknown number of repetitions. A **counter** is initialized to keep a check on repeating condition. This is known as **LOOP.**



Algorithm

Loops

For i=Initial value; increment/decrement;terminating value:
    perform some steps

Initialize *i*
WHILE *condition(i) true*:
    do something
    increase/decrease (*i*)

Example: Suppose you need to iterate from 1 to 10.

for i=1; i=i+1; i<=10:
    do something

i=1
WHILE (*i* <= 10):
    do something
    i=i+1 (Most common error)

i=10
WHILE (*i* > 0):
    do something
    i=i-1 (Most common error)

Introduction to coding

Algorithm

Example: Find sum of all odd numbers from 1 to100.

```
1. START
2. SET sum=0, i=1
3. WHILE i <101, REPEAT steps 4 to 5
4.        IF i % 2 > 0, sum=sum+i
5.        i=i+1
6. PRINT 'Sum of odd numbers is:', sum
7. END
```