Module 1: Algorithms & Flowcharts

Why should we learn C Programming ?

1. Builds a strong foundation of programming
2. Efficient Coding
3. Simplistic Approach
4. Embedded Programming
5. Versatile / Platform Independent
6. Easy to Learn & Apply
7. Middle Level Language
8. Less Execution Time
9. Fewer Libraries to remember
10. Backbone of many computer programs.

C is one of the oldest programming lang.

A beginner should ideally start with C.

ALGORITHM

An algorithm is an "effective plan or procedure" for solving a real world problem in finite amount of steps.

A well designed algorithm will always provide an answer.

It may or may not be the desired answer, but there will always be an answer.

Whatever be the size of the input data, the algorithm should be able to solve the given problem in finite amount of time.

Graphical Representation of an algorithm can be done using a

FLOWCHART

Points to be considered for developing an algorithm.

1. **Non Ambiguous :-**

Each and every statement in the algorithm must be clear and precise. There must be nothing vague in any of the statements.

2. **Speed :-**

It should produce the results at a fast pace or efficiently.

3. **Finite :-**

The algorithm should never lead to a never ending loop and it should always halt after a finite amount of time.

4. **Representative :-**

The algorithm could be represented in multiple ways.

Different Ways of Representing an algo.

1) Step - Form
2) Pseudo Code
3) Flowchart

Step Form

The procedure for solving the problem is stated with written statements.

Each statement solves a part of the problem and these together complete the solution.

Example: Problem: Making a pot of tea

Solution : 1. Fill the kettle with water
2. Put the kettle on the stove
3. Take out the teapot, make sure it's empty
4. Place the tea leaves in the teapot
5. Pour the boiling water from the kettle to the pot.
6. Switch off the kettle.

Pseudo Code :-

* It's also a written form representation of the algorithm.

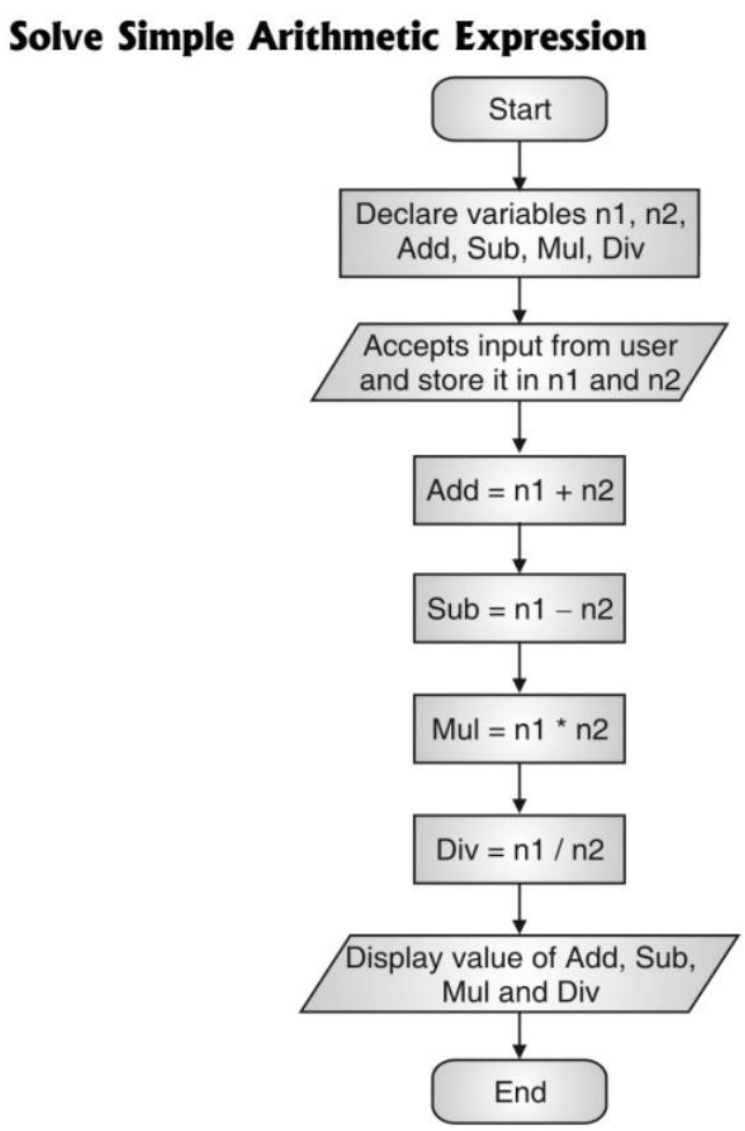
* It although uses a restricted vocabulary of words.

* It is more finite as compared to Step-form.

Flow chart :-

* Graphical representation of the algorithm.

* It uses symbols & language to represent sequence, decision and repition.



3 constructs of an Algorithm

a) Sequence

Sequence means that each step or process in the algorithm is executed in the specified order, otherwise the algorithm will fail.

Examples

1. Write an algorithm to display a statement Hello World.

Step Form:

Step 1: Display the sentence "Hello World"

Step 2: Stop

Pseudo Code Form:

Step 1: START
Step 2: PRINT "Hello World"
Step 3: STOP

2. Write an algorithm to accept a number and display its square.

Step Form:

S1: Indicate the user to enter the i/p number by displaying suitable sentence.

S2: Wait for the user to enter the number.

S3: Calculate the square of the user entered number.

S4: Display the calculated result

S5: Stop.

Pseudo Code Form:

```
S1 :  START
S2 :  PRINT "Enter a number"
S3 :  INPUT n
S4 :  a = n * m
S5 :  PRINT a
S6 :  STOP
```

3. Write an algorithm to accept 2 numbers and display its product.

Step Form:

S1 : Indicate the user to enter the i/p numbers by displaying suitable sentence.

S2 : Wait for the user to enter the 2 numbers.

S3 : Calculate the product of the user entered numbers

S4 : Display the calculated result.

S5 : Stop.

Pseudo Code Form:-

S1 : START

S2 : PRINT "Enter 2 numbers"

S3 : INPUT a, b.

S4 : x = a * b

S5 : PRINT x

S6 : STOP

4. Write an algorithm to calculate and display the area and perimeter of a rectangle

Step Form :-

S1: Indicate the user to enter the length and breadth of the rectangle as input from the user by displaying suitable sentence.

S2 : Wait for the user to enter the input

S3 : Calculate the area and perimeter using the corresponding formulae

S4 : Display the calculated area & perimeter.

S5 : Stop.

Pseudo Code Form :-

```
S1: START
S2: PRINT "Enter the length & breadth of
    the rectangle".
S3: INPUT l, b
S4: a = l x b
S5: p = 2(l + b)
S6: PRINT a, p
S7: STOP
```

5. Write an algorithm to swap 2 numbers

Step Form :-

S1 : Indicate the user to enter 2 numbers by displaying suitable sentence

S2 : Wait for the user to enter the input.

S3 : Using a temporary variable, swap the 2 numbers.

S4 : Display the 2 numbers indicating the 2 numbers are swapped.

S5 : Stop.

Pseudo Code Form:-

```
S1:  START
S2:  PRINT "Enter 2 numbers"
S3:  INPUT a,b
S4:  temp = a
S5:  a = b
S6:  b = temp
S7:  PRINT a,b
S8:  STOP
```

b) Repetition

Repetition can be implemented using constructs such as repeat loop, while loop, and if ... then ... go to ... loop.

The Repeat loop is used to iterate or repeat a process or sequence of processes until some condition becomes true.

Examples

1. Write an algorithm to display the word "Computer" five times

Step Form :-

Step 1: Initialize a variable as a counter with the initial value as 1.

Step 2: Check if the counter variable is greater than 5, if yes go to step 6.

Step 3: Display the word "Computer"

Step 4: Increment the counter.

Step 5: Go to Step 2.

Step 6: Stop

Pseudo Code Form:-

Step 1 : START

Step 2 : $i = 1$

Step 3 : IF $i > 5$ THEN GOTO step 7

Step 4 : PRINT "Computer"

Step 5 : $i = i + 1$

Step 6 : GOTO step 3

Step 7 : STOP

2. Write an algorithm to display the first ten natural numbers.

Step Form:-

Step 1 : Initialize a counter with the value as 1.

Step 2: Check if the counter is greater than 10. If yes, then go to step 6.

Step 3: Display the value of the counter variable.

Step 4: Increment the counter variable

Step 5: Go to Step 2

Step 6: Stop

Pseudo Code Form:-

Step 1: START

Step 2: i = 1

Step 3: IF i > 10 THEN GoTo step 7

Step 4: PRINT i

Step 5: i = i + 1

Step 6: GOTO step 3

Step 7: STOP

c) Decision

In algorithms the outcome of a decision is either true or false ; it cannot be both or anything in between.

The outcome of the decision is based on some condition that can only result in a true or false value.

Examples :-

1. Write an algorithm to find the factorial of a number.

Step Form :-

S1 : Indicate the user to enter a number by displaying the suitable sentence

S2 : Wait for the user to enter the i/p.

S3 : Initialize a counter with the value as 1 and "fact" variable as 1.

S4 : Check if the counter is greater than the user entered variable. If YES , then go to step 8.

S5 : Multiply the "fact" variable with the counter variable.

S6: Increment the counter variable.

S7: Go to step 4.

S8: Display the valve of the factorial.
i.e the variable "fact".

S9: Stop.

Pseudo Code Form:-

S1 : START

S2 : PRINT "Enter a number"

S3 : INPUT n

S4 : i=1 , fact = 1

S5 : IF i >n THEN GO TO step 9.

S6 : fact = fact * i

S7 : i = i +1

S8 : GoTo step 5

S9 : PRINT fact

S10 : STOP

2. Write an algorithm to check if the entered number is a prime number or not.

Step Form :-

S1: Indicate the user to enter the input number by displaying suitable sentence.

S2: Wait for the user to enter the number to be checked.

S3 : Initialize a divisor variable with the value as 2.

S4 : Check if the user entered variable is divisible by the divisor variable.
If YES, then go to step 7.

S5 : Increment the divisor variable

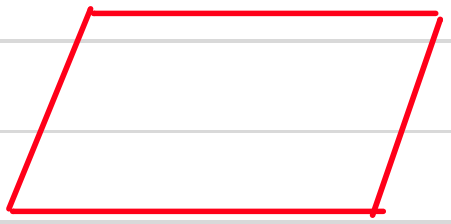
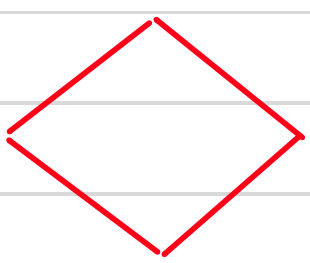
S6 : Go to step 4.

What is a flowchart ?

A flowchart is a graphical representation of the algorithm.

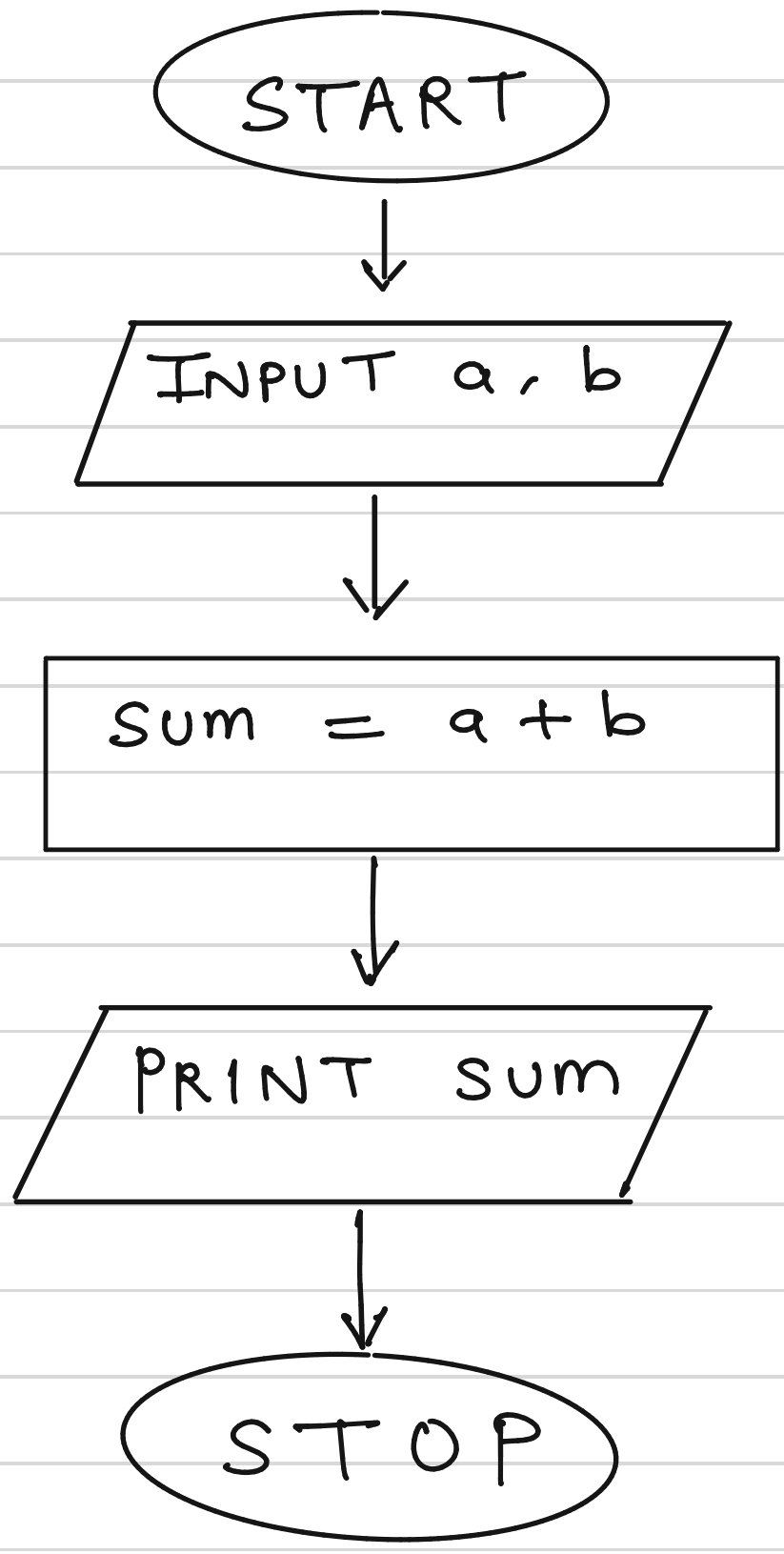
The flowchart gives a very nice idea of the flow of the algorithm in sequence as well as the conditional changes in the sequence.

The various symbols used for drawing the flowcharts are as follows:-

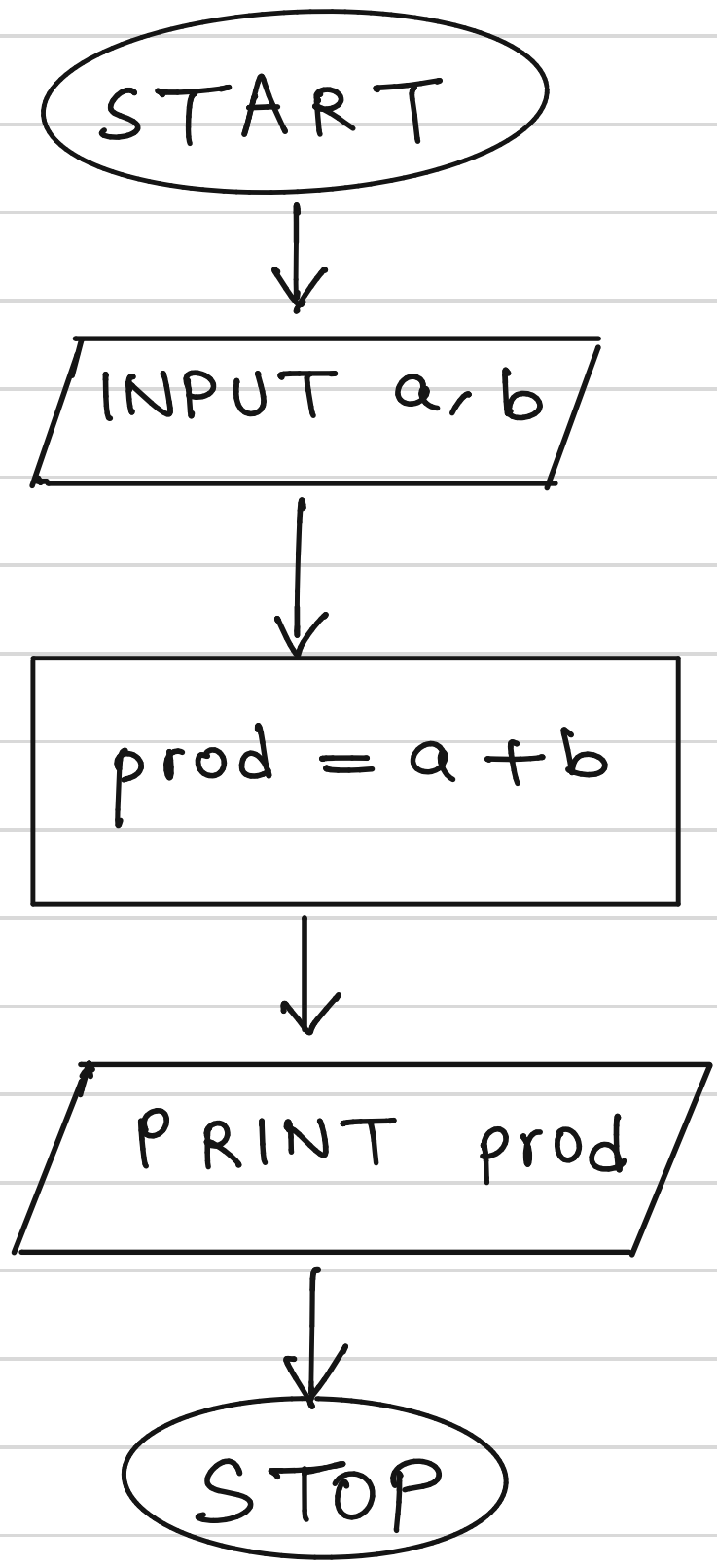
Oval		Used for start or stop of an algorithm
Flow Line		Used to denote the direction of flow
Parallelogram		Used to perform input or output
Rectangle		Used to indicate operation to be performed.
Diamond		Used to take a decision.

Examples

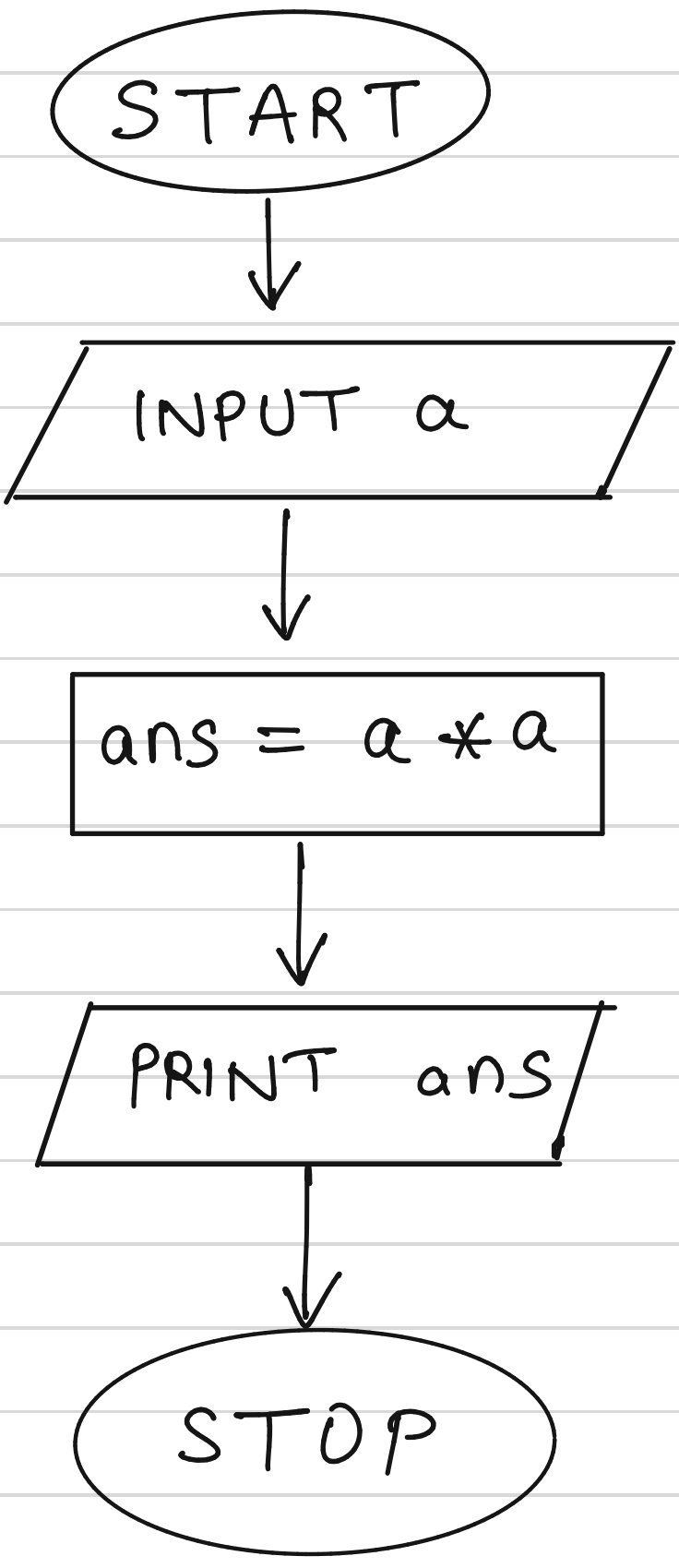
1. Draw a flowchart to add two numbers.



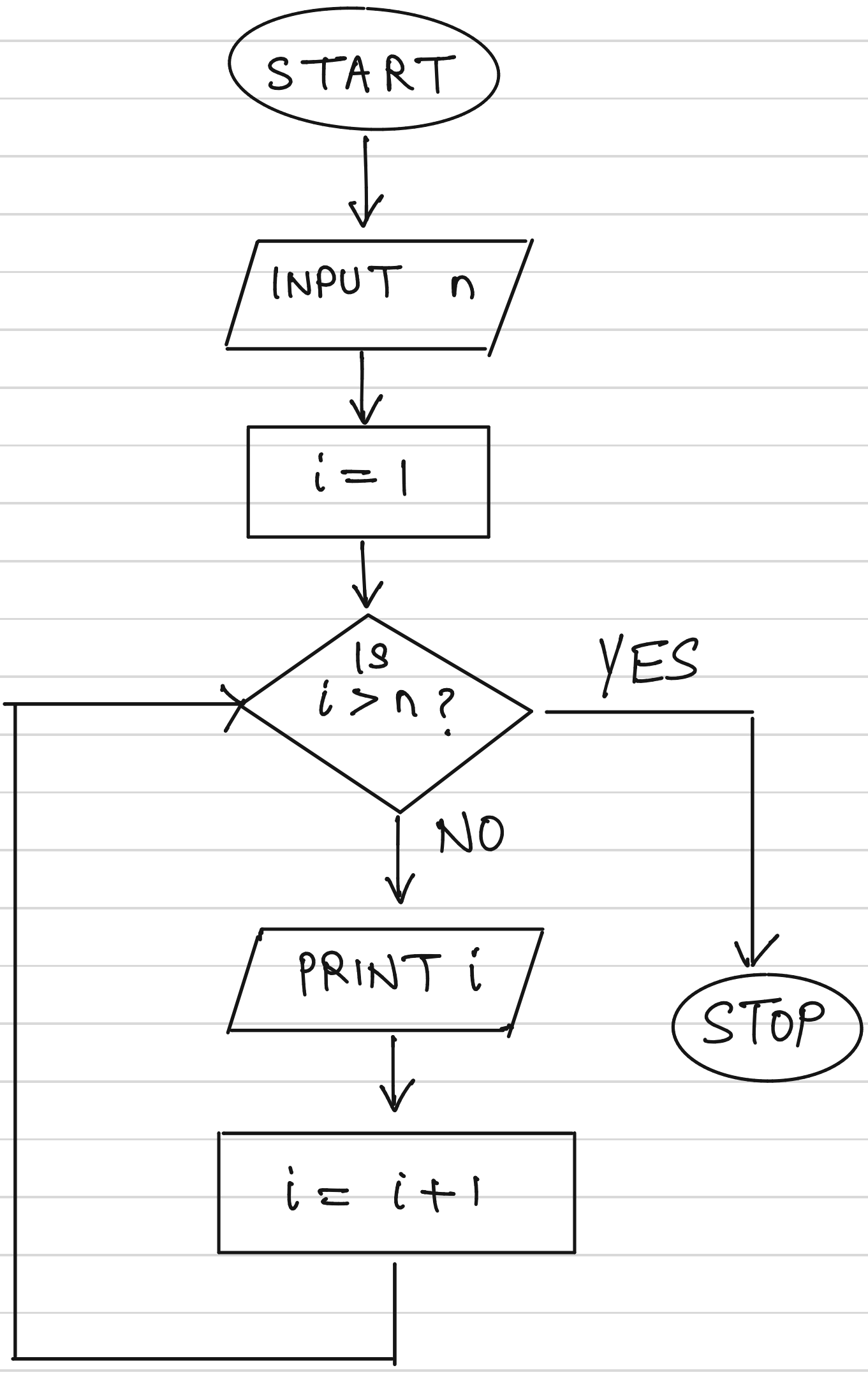
2. Draw a flowchart to find the product of two numbers.



3. Draw a flowchart to find the square of a number



4. Draw a flowchart to display the first 'n' natural numbers.



Efficiency of an Algorithm

To calculate the efficiency of an algorithms, the various parameters are required to be considered.

1. Space Complexity

The space complexity can be defined as the memory space required for an algorithm to be executed.

The space required in memory for an algorithm to be executed can be divided into two parts.

viz. constant (C) and instance (S_p)

The constant space is required for storing the variables, program etc.

The instance space is required depending on the problem case or is input dependent.

Thus, the space required for implementing an algorithm can be given as

$$S(p) = C + S_p$$

2. Time Complexity

The time complexity of an algorithm is the time required by the computer to execute the program implemented according to the corresponding algorithm.

The time required to execute the program depends on various other parameters besides the algorithm implementation.

The different parameters on which the execution time depends are instruction set used, other programs running in the computer, hardware or processor speed, etc.

For example,

If the instructions are to be executed for 'n' times

(Where 'n' is the input size or number of input values),

then the time complexity is said to be 'n'.