UNIT 1 BASICS OF C PROGRAMMING **6**

Problem Solving Techniques: Introduction to Algorithm, Pseudo code, Flow Chart, Structure of 'C' program. C Tokens: Keywords, Data Types, Constants, Variables - Declaration - Qualifiers – typedef

1.1 PROBLEM SOLVING TECHNIQUES.

Problem Solving is a scientific technique to discover and implement the answer to a problem. The computer is the manipulating device that follows the set of commands known as program.

Program:

Program is the set of instructions which is run by the computer to perform specific task. The task of developing program is called programming.

Problem Solving Technique:

Here are some basic steps to solve the problems

Step 1: Identify and Define Problem

Explain your problem clearly as possible as you can.

Step 2: Generate Possible Solutions

- List out all the solution that you find.
- Generate the maximum number of solution as you can without considering the quality of the solution.

Step 3: Evaluate Alternatives

After generating the maximum solution, Remove the undesired solutions.

Step 4: Decide a Solution

After filtering all the solution, you have the best solution only. Then choose one of the best solution and make a decision to make it as a perfect solution.

Step 5: Implement a Solution:

After getting the best solution, implement that solution to solve a problem.

Step 6: Evaluate the result

After implementing a best solution, evaluate how much your solution solve the problem. If your solution will not solve the problem, then you can again start with Step 2.

The following are the Problem solving Techniques used to develop solutions for a problem.

- Algorithms
- Flowcharts
- Pseudocode

1.1.1 ALGORITHM

- It is a step by step description for solving a task or a problem. The ordered sequence of instructions is executed in the specified sequence, to get the desired results.

Characteristics of Algorithm:

- In the algorithms each and every instruction should be precise and unambiguous.
- The instructions in an algorithm should not be repeated infinitely.
- Ensure that the algorithm will ultimately terminate.
- The algorithm should be written in sequence.
- It looks like normal English.

Qualities of a Good Algorithm:

(i) Time:

To execute a program, the computer system takes some amount of time. The lesser is the time required, the better is the algorithm.

(ii) Memory:

To execute a program, computer system takes some amount of memory storage. The lesser is the memory required, the better is the algorithm

(iii) Accuracy:

Multiple algorithms may provide suitable or correct solutions to a given problem, some of these may provide more accurate, such algorithms may be suitable.

(iv) Sequence:

The procedure of an algorithm must form in a sequence.

(v) Functionality:

The designed algorithm must solve a single isolated problem and algorithms are designed to handle a range of input data.

Representation of Algorithms:

- The algorithms can be represented in several ways. Generally, one of the following method is used.

  (i) Flow chart
  (ii) Pseudocode
  (iii) Program

(i) Flow chart:

The flow chart is a pictorial representation of an algorithm, the sequential steps in an algorithm can be represented as a flow chart using the standard symbols.

(ii) Pseudocode:

Pseudocode is also a formal design tool and It consist of short, readable instructions with the rules of structured programming.

(iii) Decision Table:

It is used for nested selection to help clarify the conditions to be tested and how these conditions should be nested to do proper actions.

(iv) Program:

The algorithms can be represented as a program using any high level language that becomes a program.

BUILDING BLOCKS OF ALGORITHM

Algorithm can be constructed from just three basic building blocks. These three building blocks are

- **Sequence**
- **Selection**
- **Iteration(Repetition)**
- **Functions**

(i) Sequence

This describes a sequence of actions that a program carries out one after another, unconditionally. Execute a list of statements **in order**.

Consider an example

Algorithm for Addition of two numbers:

Step 1: Start the program
Step 2: Get two numbers as input and store it in to a and b
Step 3: Add the number a & b and store it into c
Step 4: Print c
Step 5: Stop.

(ii) Selection

Selection is the program that allows a program to choose between different actions. Choose at most one action from several alternative conditions.

Example

Algorithm to find biggest among 2 numbers:

Step1: Start
Step 2: Get two numbers as input and store it in to A and B
Step 3: If a is greater than b then
Step 4: Print A is big
Step 5: else
Step 6: Print B is big
Step 7: Stop

(iii) Repetition

Repetition(loop) may be defined as a smaller program the can be executed several times in a main program. Repeat a block of statements while a condition is true.

Example

Algorithm to calculate factorial no:

Step1: Start the program

Step 2: Read the number num.

Step 3: Initialize i is equal to 1 and fact is equal to 1

Step 4: Repeat step 5 and 6 until i is equal to num

Step 5: fact=fact * i

Step 6: i=i+1

Step 7: Print f

(iv)Function:

Function is a small block of statements in a program**.** Function executes only when the function is called**.**

1.1.2 PSEUDOCODE

Pseudocode is a formal design tool. It consists of short, readable instruction set with the rules of structured programming for logically explaining an algorithm.

"pseudo" means false, "code" means the set of statements (or) instructions written in a programming language. Pseudocode is also called 'Program Design Language' (PDL)

Rules for writing Pseudocode:

(i) Write one statement per line.

(ii) Capitalize the keywords

(iii) Indent to show hierarchy

(iv) End Multiline structure

(iv) Keep statements programming language independent

(v) It is written using structured English.

Example: Pseudocode to calculate student total and average

START

READ name, class, m1, m2, m3

Total = m1 + m2 + m3

Average = TOTAL /3

WRITE name, class, total, average

END

Pseudocode is made up of the following logic structure,

- Sequential logic
- Selection logic
- Iteration logic

1) Sequence Logic

- It is used to perform instructions in a sequence, that is one after another
- Thus, for sequence logic., pseudocode instructions are written in an order in which they are to be performed.
- The logic flow of pseudocode is from top to bottom.

Example: Pseudocode to add two numbers:

START

READ a,b

COMPUTE c by adding a &b

PRINT c

STOP

2) Selection Logic

It is used for making decisions and for selecting the proper path out of two or more alternative paths in program logic.

- It is also known as decision logic.
- Selection logic is depicted as either an IF..THEN or an IF…THEN..ELSE Structure.

Example: Pseudocode to find biggest among two numbers:

START

READ a and b

IF a>b THEN

PRINT "A is big"

ELSE

PRINT "B is big"

ENDIF

STOP

3) Repetition Logic

- It is used to produce loops when one or more instructions may be executed several times depending on some conditions.
- It uses structures called **DO_WHILE, FOR and REPEAT__UNTIL**

Example: Pseudocode to print first 10 natural numbers

START

INITIALIZE a->0

WHILE a<10

PRINT a

ENDWHILE

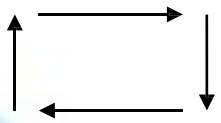
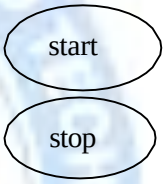
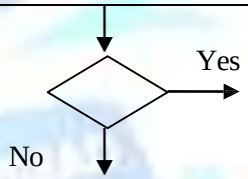
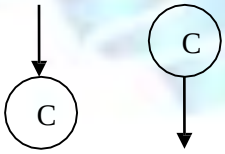
STOP

1.1.3 FLOW CHART

- A Flow chart is a pictorial representation of an algorithm.
- A flowchart uses different shaped symbols to denote the different appropriate instruction and these instructions can be written within the boxes using clear statements.
- Then these boxes are connected by lines having arrows to indicate the flow of operations.

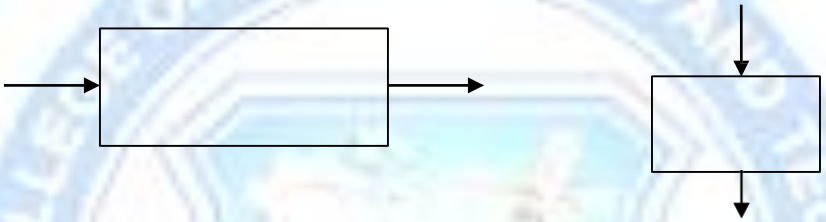
Flow chart symbols:

- A flowchart is drawn using different kinds of symbols. A symbol used in a flowchart is for a specific purpose.
- The flow chart symbols are available in word processors.

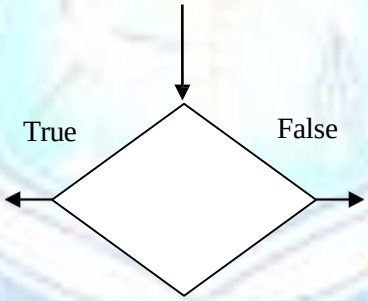
Description	Symbols
i) Flow lines: These are the left to right or top to bottom lines connecting symbols. These lines show the flow of control through the program	
ii) Terminal: The oval shaped symbol always begins and ends the flowchart. The start symbol will have only one flow line and stop will have and entering flow line.	
iii)Input/Output: The parallelogram is used for Input (Read)and output (write).	
iv) Process: The rectangle symbol is used for calculation and initialization of memory locations	
v) Decision: The diamond shaped symbol is used to indicate at a point at which a decision has to be made and a branch to one of two or more alternative point is possible	
vi) Connectors: A connector symbol is represented by a circle and a letter or digit is placed in the circle to specify the link. Whenever a complex flowchart is drawn more than one page, connector symbol is used	
vii)Predefined Process: This symbol used for create functions.	

Rules for Drawing flowchart:

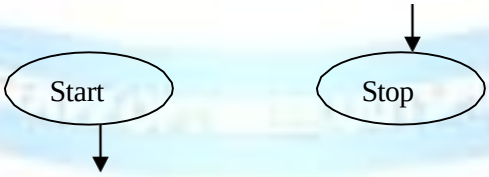
1. The standard symbols should only be used.
2. A flowchart must have a start and end.
3. The usual direction of the flow of procedure is from top to bottom; or left to right.
4. Only one flow line should come to a process symbol and only one flow line should out from a process symbol.



5. Only one flow line should enter a decision symbol, but two or three flow lines, one for each possible answer. Should leave the decision symbol.



6. Only one flow line is used in the terminal symbol.



7. The flow lines should not cross each other
8. Be consistent in using names and variables in the flowchart
9. Keep the flowchart as simple as possible.
10. The words in flowchart symbols should be common statements and easy to understand.
11. If a new page is needed for flowchart, then use connectors for better representation.

Advantages of Flowchart:

(i) To understand logic clearly:

Before coding the program, the programmer must be known about the logic of the program.

(ii) Better communication:

The flowcharts are better from communication, because "a picture is worth of thousand words". It is very easy for the programmer to explain the logic of the program through the flowchart

(iii) Effective Analysis:

The flowcharts are very well suited to analyse the problem and can be broken down into parts.

(iv) Effective synthesis:

A team of designers or programmers are associated with the design of complex system. Each designer or programmer is responsible for designing a part of the entire system.

(v) Effective coding:

After designing the flowchart, it is very easy to write programs for the programmers, because flowchart is the road map for the programmers.

(vi) Proper Program documentation:

The documentation involves collecting, organizing and storing a complete record of program.

(vii) Systematic Debugging:

After taking full care in program design there may be chances for some errors. Such errors can easily be found out in the flowcharts, and these can be eliminated easily.

(viii) Efficient program Maintenance:

The maintenance of operating system becomes easy with the help of flowchart.

Limitations of using flowcharts:

(i) Complex Logic:

Sometimes, the program logic is complicated. In such case, flowchart becomes complex and difficult to use.

(ii) Alterations and Modifications:

If any alternations required, the flowchart may require re-drawing completely.

(iii) Cost:

For large application the time and cost of flowchart drawing will be more.

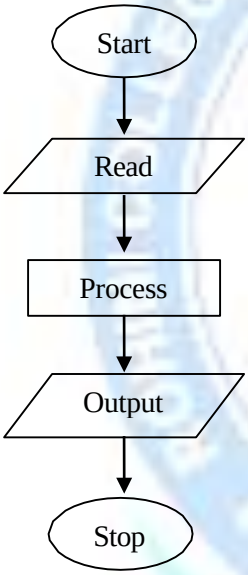
(iv) Not known:

Task start and finish dates, the duration of tasks, resources needed, physical locations of tasks are not known.

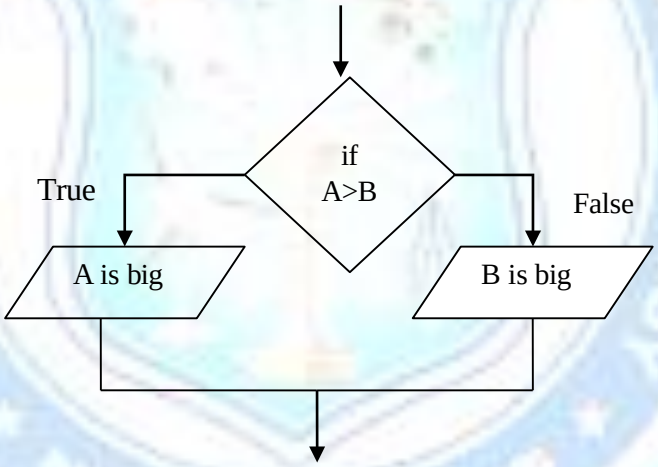
Flowchart is made up of the following logic structure,

- **Sequential logic**
- **Selection logic**
- **Iteration logic**

Sequential logic



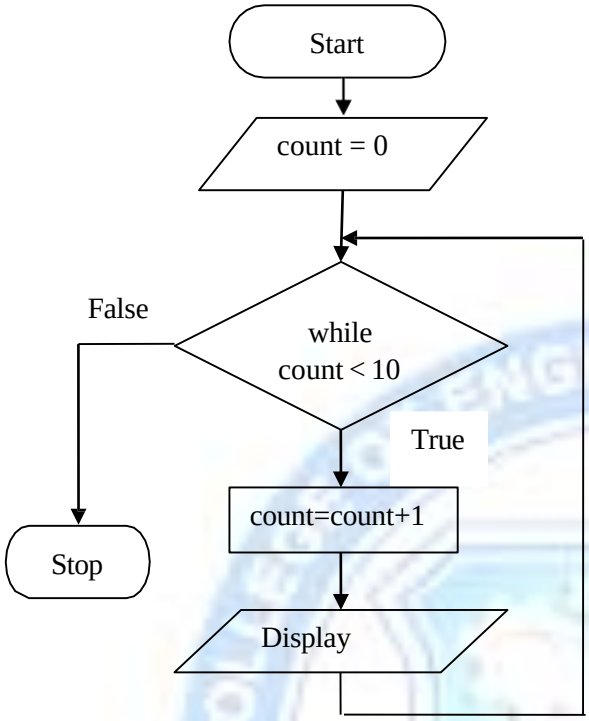
Selection Logic



Iteration logic

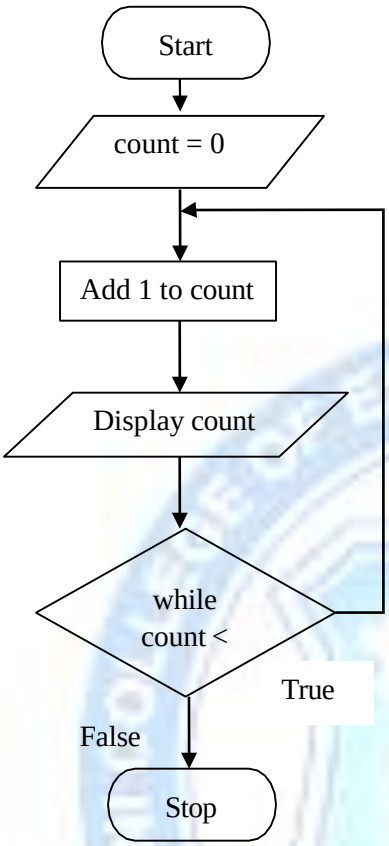
(a) Top Tested loop:

- If the condition is true, the control follows the flow line that goes into the loop.
- If it is false, control follows the flow line to the first statement after the loop.



(b) Bottom Tested Loop:

- The body of the loop will be executed, once before control even gets to the loop test.
- If the condition is false, control follows the flow line back to the loop.
- If the test is true, control follows the branch from the decision box that exists the loop structure.



SAMPLE PROBLEMS

1) Write an algorithm, Flowchart, Pseudocode to find the area and circumference of the circle.

Algorithm:

Step 1: Start

Step 2: Input the radius of the circle

Step 3: Find the area and circumference

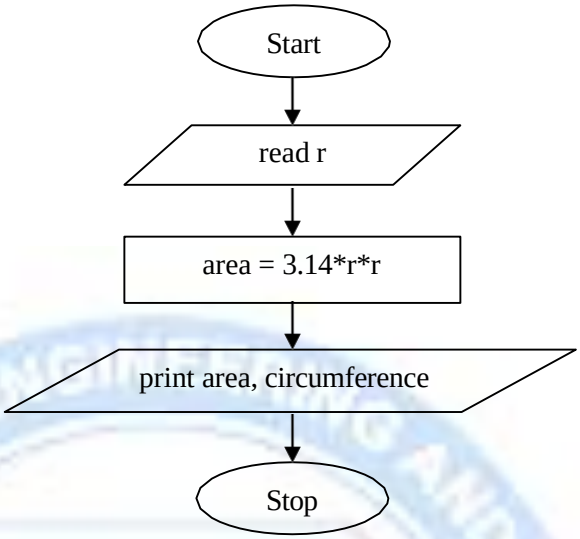
area ← 3.14 * r*r

circumference ← 2*3.14*r

Step 4: Print the area and circumference

Step 5: Stop

Flow chart:



Pseudocode:

READ the radius r

Find the area and circumference

Area = 3.14*r*r

circumference = 2*3.14*r

WRITE area and circumference.

2) Roots of the Quadratic Equation

Algorithm:

Step 1: start

Step 2: Enter the value of a, b, c

Step 3 : Find the value of ‘d’ using the formula d = b*b-4*a*c

Step 4: If d is greater than or equal to zero then find the two roots are

root 1← (-b + sqrt (d))/ (2*a)

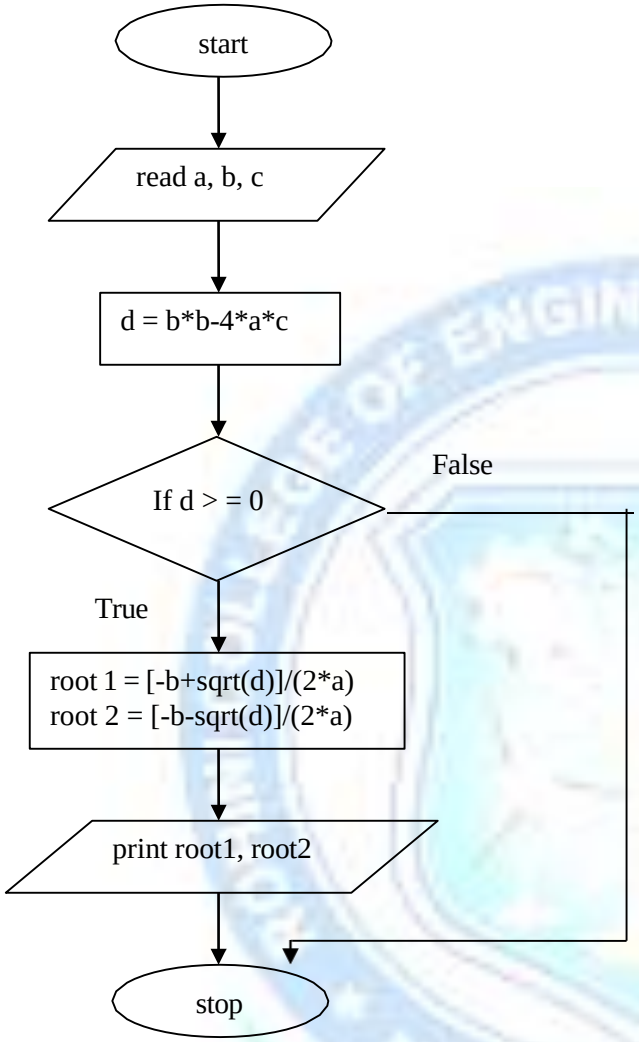
root 2← (-b- sqrt (d))/(2*a)

Step 5: Print the two roots, roots1, root2

Step 6: If d is not greater than or equal to zero,

then print the roots are imaginary

Step 7: Stop



Pseudocode:

READ the values of a, b, c

Find Discriminant

d← b*b - 4*a*c

IF d>=0 THEN

Calculate root1 = (-b + sqrt(d))/(2*a)

Root2 = (-b + sqrt (d))/(2*a)

ELSE

Roots are imaginary

ENDIF

WRITE root1, root2

3) Program for Simple calculator

Algorithm:

Step 1: Start

Step 2: Enter the values of a, b

Step 3: Find the sum

sum = a+ b, sub=a-b, pro=a*b, quo=a/b

Step 4 : Print the sum, sub, pro, quo

Step 5:

Stop

Pseudocode:

READ a,b

Find the sum difference, product, quotient

sum = a + b

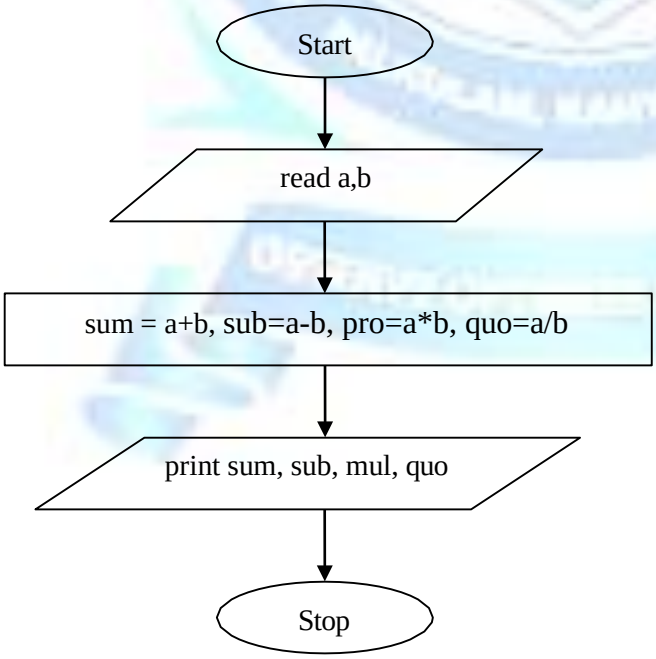
sub=a-b

pro=a*b

quo=a/b

WRITE sum, sub, pro, quo

Flowchart



4) To find the given year is leap year or not

Algorithm:

Step 1: Start

Step 2: Read the year

Step 3: If (year mod 4) = 0 THEN PRINT "leap year"

ELSE print "Not leap year"

Step 4: Stop

Pseudocode:

READ the year

IF (year % 4 = = 0) THEN

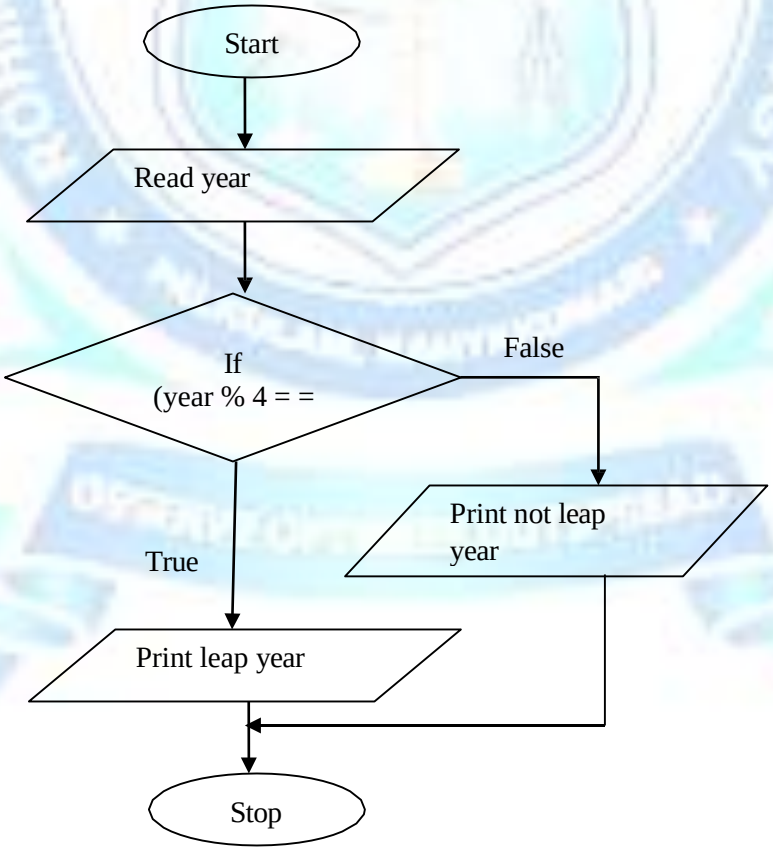
WRITE the year is leap year

ELSE

WRITE the year is not a leap year

ENDIF

Flow chart



5) To find the Greatest of three numbers

Algorithm:

Step1: start

Step 2: READ a, b, c

Step 3: IF (a >b) and (a>c) THEN

Step 3.1: Print ‘a’ is big

ELSE

Step 4: IF (b>c) THEN

Step 4.1: Print ‘b’ is big

ELSE

Step 4.2: Print ‘c; is big

Step 5: Stop

Pseudocode:

READ a, b, c

IF ((a>b) &&(a>c)) THEN

WRITE ‘a is big’

ELSE IF (b>c) THEN

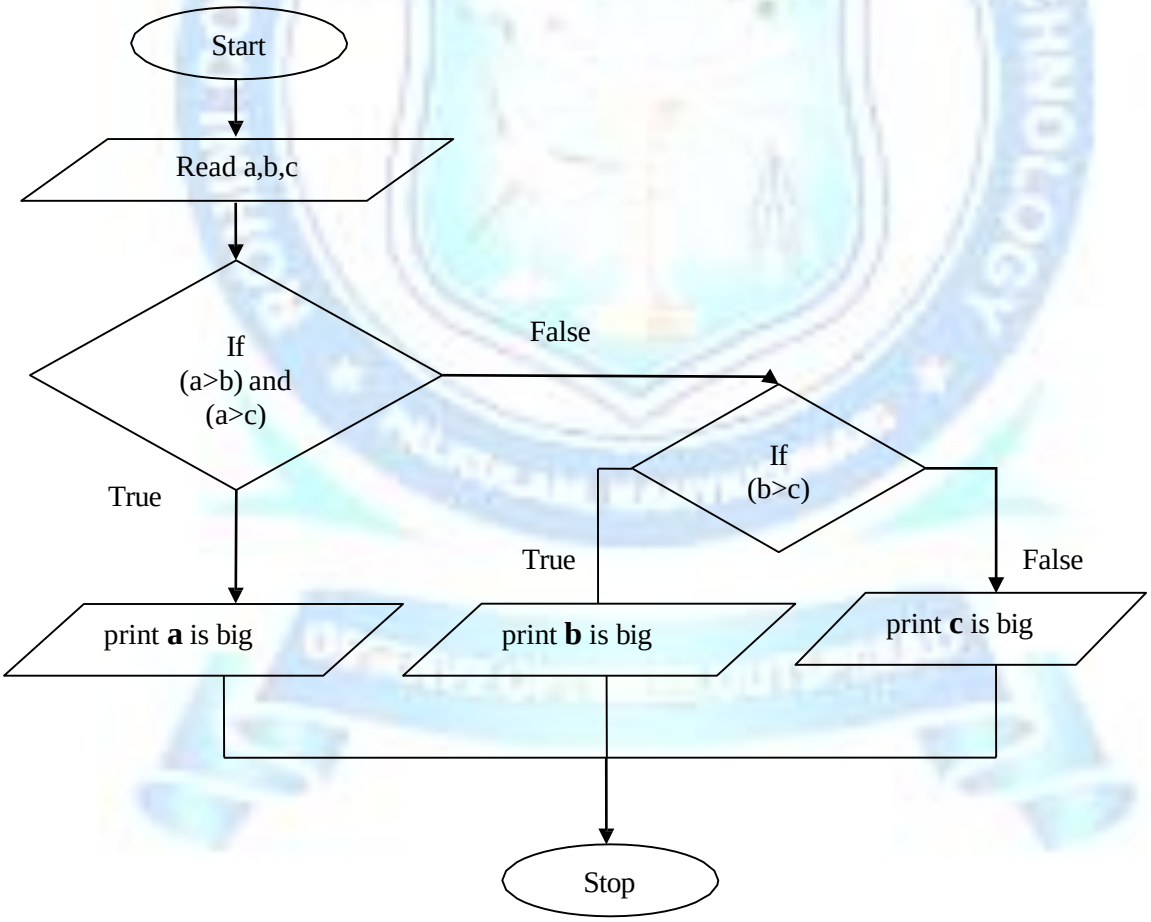
WRITE ‘b is big’

ELSE

WRITE ‘c is big’

ENDIF

Flowchart



6) To print the Reverse of a number

Algorithm:

Step 1: Start

Step 2: Enter the value of n

Step 3: Assign r = 0, sum = 0

Step 4: IF n >0 ELSE Goto step 6

Step 4.1: r← n mod 10.

Step 4.2: sum←sum *10+r

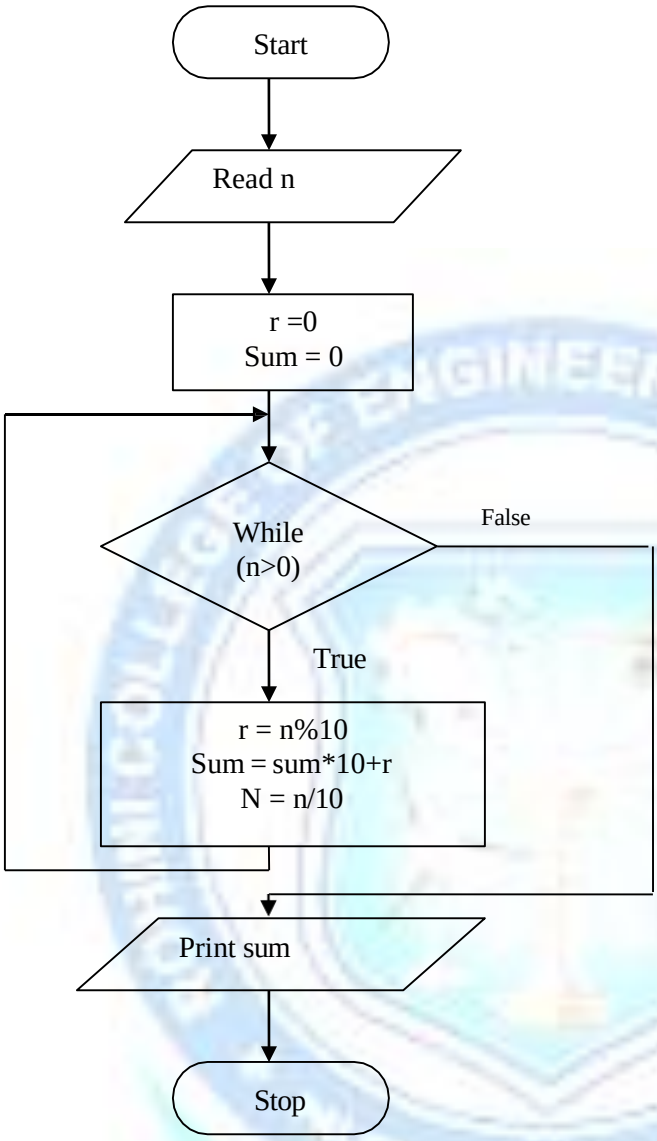
Step 4.3: n←n div 10

Step 5: Goto step 4

Step 6: Print sum

Step 7: Stop

Flow chart



Pseudocode

Set sum = 0, r=0

READ value of n

WHILE (n>0)

r =n% 10

sum = sum *10 + r

n = n/10

ENDWHILE

WRITE Sum

7) To find the sum & avg of n numbers

Algorithm

Step 1: Start

Step 2: Enter the value of n

Step 3: Assign , sum←0

Step 4: If n > 0, ELSE Go to step 6

Step 4.1: sum ← sum + r

Step 5: Goto step 4

Step 6: Print sum

Step 7: avg=sum/n

Step 8: print avg

Step 9: Stop

Pseudocode

Set Sum = 0,

READ the value of n

WHILE (n <= 0)

READ the value of num

sum = sum +num

n = n-1

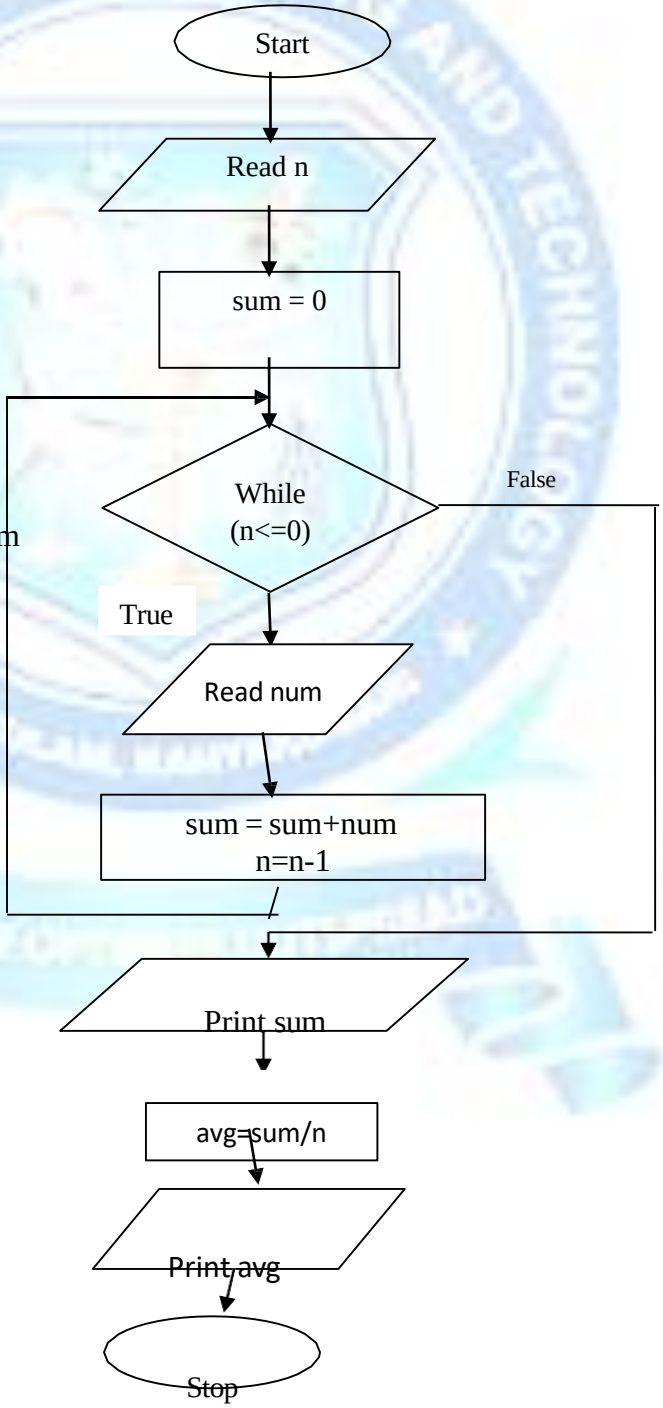
ENDWHILE

WRITE sum

Avg=avg/n

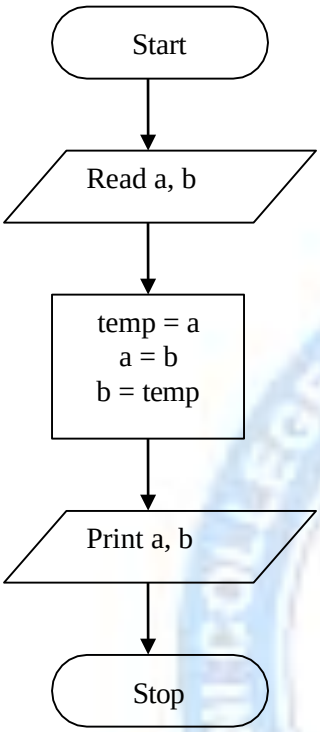
WRITE avg

Flow chart



10) To swap the variable using another variable

Flow chart



Algorithm

Step1: Start

Step 2: Read the value of a,b

Step 3: To swap using temp

Step 3.1: temp← a

Step 3.2: a ← b

Step 3.3: b ← temp

Step 4: Print value a and b

Step5: Stop

Pseudocode

READ a, b

Swap a and b using temp

temp = a

a = b

b = temp

WRITE a and b

11) To generate the Fibonacci series:

Algorithm

Step 1: Start

Step 2 : Assign f1←0, f2 ← 1, f←0

Step 3: Print f1, f2

Step 4: f ←f1+f2

Step 5: READ n

Step 6: IF (f < n), ELSE Goto Step 8

Step 6.1 : f←f1 + f2

Step 6.2: f1 ← f2

Step 6.3: f2←f

Step7: Goto step 6

Step 8: Stop

Pseudocode

Set f1 =0, f2 = 1, f = 0

WRITE f1, f2

f = f1 + f2

READ n

WHILE (f < n)

f = f1 + f2

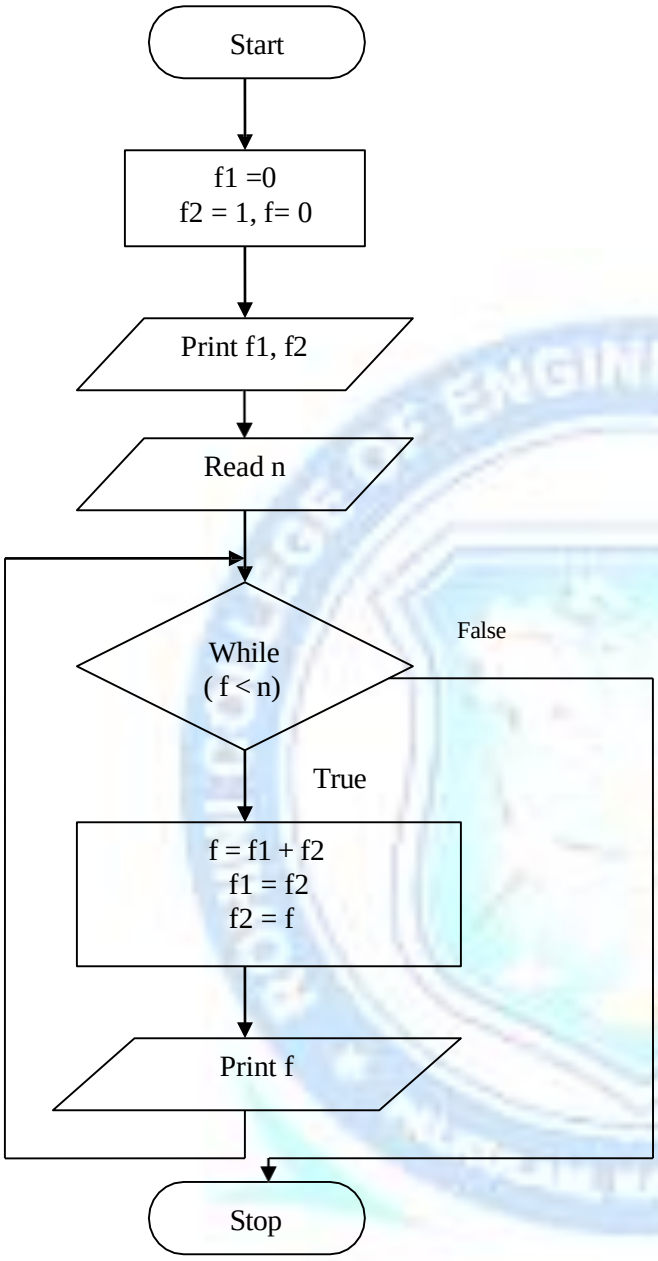
f1 = f2

f2 = f

WRITE f

ENDWHILE

Flowchart :


Start
f1 =0
f2 = 1, f= 0
Print f1, f2
Read n
While
(f < n)
False
True
f = f1 + f2
f1 = f2
f2 = f
Print f
Stop

12) Find the area of triangle:

Algorithm

Step 1: Start

Step 2: Read value of a, b, c

Step 3: Calculate the three sides

s ← (a + b + c)/2

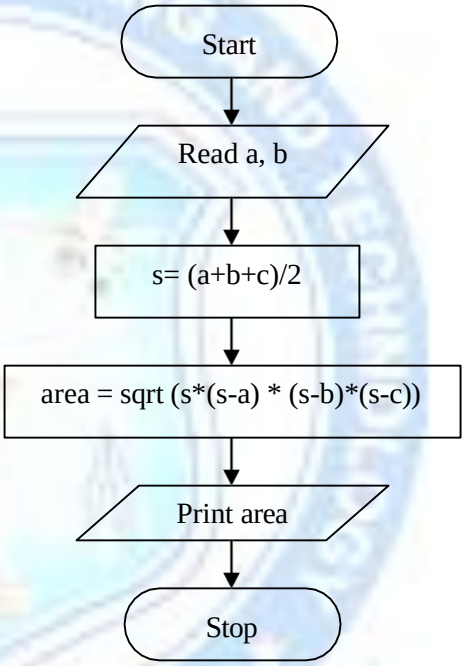
Step 4: Find area

area ← sqrt (s*(s-a)*(s-b)*(s-c))

Step 5: print area

Step 6: Stop

Flow chart



Pseudocode

READ a, b, c

CALCULATE s

s = a+b+c/2

area = sqrt (s*(s-a) * (s-b) * (s-c))

WRITE area

13) To convert the celcius into Fahrenheit

Algorithm

Step1: Start

Step 2: Read the celcius

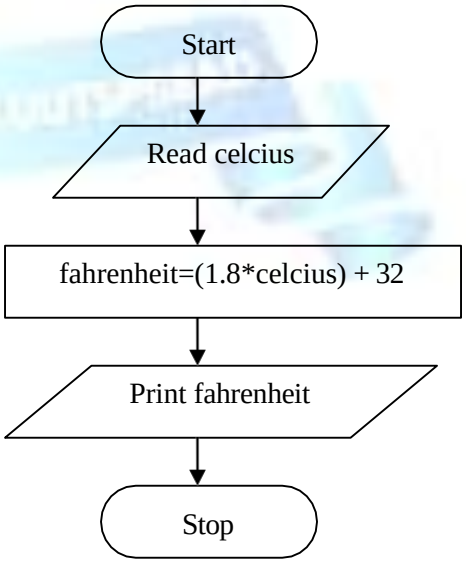
Step 3: Calculate the fahrenheit

fahrenheit ← (1-8 *celcius) + 32

Step 4: Print fahrenheit

Step5: Stop

Flow chart



Pseudocode

READ celcius

Find fahrenheit

fahrenheit = (1.8 * celcius) + 32

WRITE Fahrenheit

14) To find the sum of all odd integers between 1 and n

Algorithm

Step 1: Start

Step 2: Read the value of n

Step 3: Assign Sum ←0 and i← 1

Step 4: IF i<=n ELSE Goto step 6

Step 4.1: sum ←sum + i

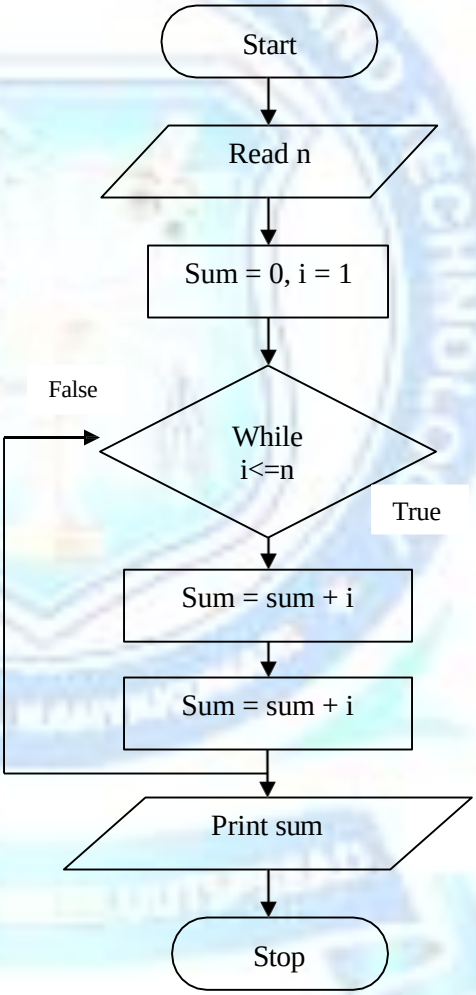
Step 4.2: i←i+2

Step 5: Goto step 4

Step 6: Print sum

Step 7: Stop

Flow chart



Pseudocode

Set sum = 0, i = 1

READ n

WHILE (i < = n)

sum = sum + i

i = i+ 2

ENDWHILE

WRITE SUM

15) To solve the series $S = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + ... \frac{x^n}{n!}$

Algorithm

Step 1: Start

Step 2: Read the value of x, n

Step3: assign s← 0, term ←x, i←1.

Pseudocode

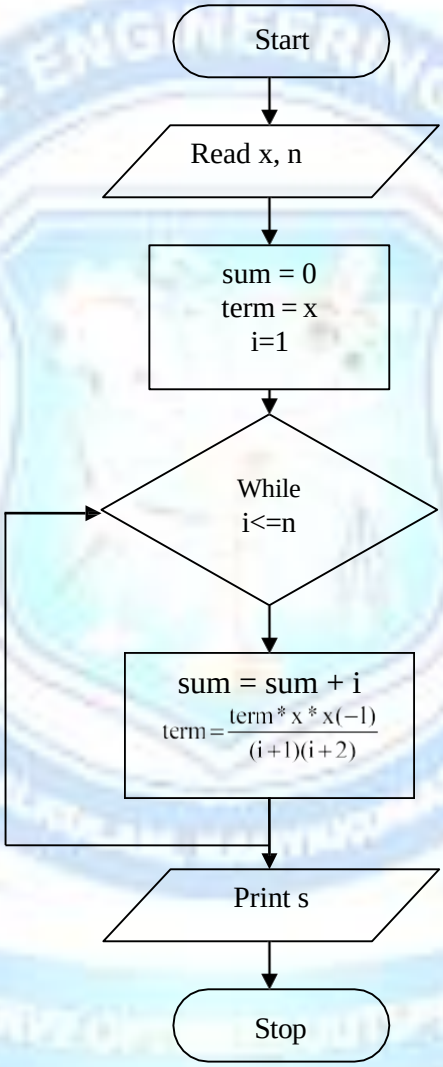
Set s= 0, term = x, I =1

READ x, n

WHILE (I < = n)

Step 4: s←s+term	s=s+term
Step 5: term = term*x*x (-1)/ (i+1)(i+2)	term = (term *x*x(-1)/(i+1)*(i+2)
Step 6: i←i+2	i=i+2
Step 7: Repeat step 4 to 6 WHILE (i<=n)	ENDWHILE
Step 8: Stop	WRITE result.

Flow chart



16. To convert a binary number to decimal number

Algorithm

Step 1: Start

Step 2: Read binary value n

Step 3: Initialise q = n, s= 0, k=0

Step 4: Chack (q>0)

Step 5: Calculate = r=q%10

s=s+r*pow (2, k)

q = q/10

k = k+1

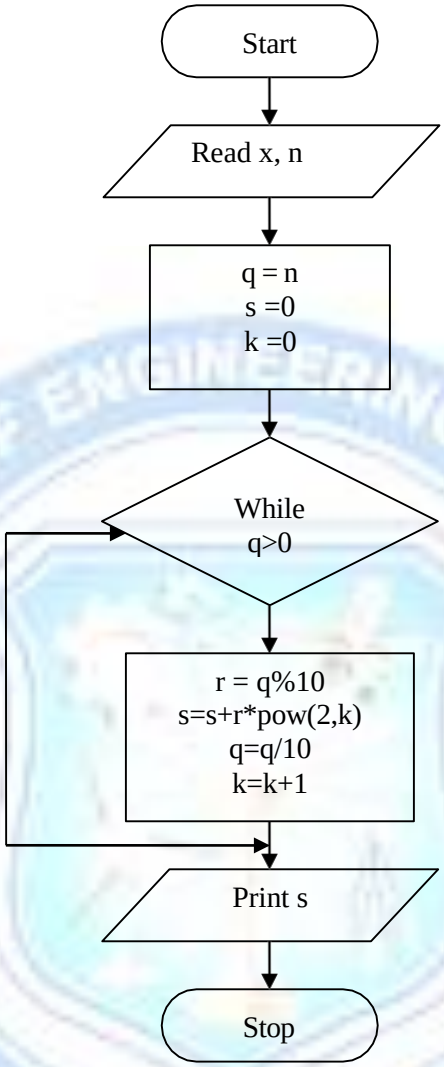
Step 6: Print S

Step 7: Stop

Pseudocode

```
set q = n, s=0, k=0
READ n
WHILE (q>0)
    r=q%10
    s=s+r * pow(2, k)
    q=q/10
    k = k+1
ENDWHILE
WRITE s
```

Flow chart:



17) To find the factorial of the given number

Algorithm

Step 1: Start

Step 2: Read the value n

Step 3: Set initial values fact= i=1

Step 4: IF i<=n ELSE Goto Step 6

Step 4.1 : fact ← fact *i

Step 4.2 : i ← i + 1

Step 5: Goto Step 4

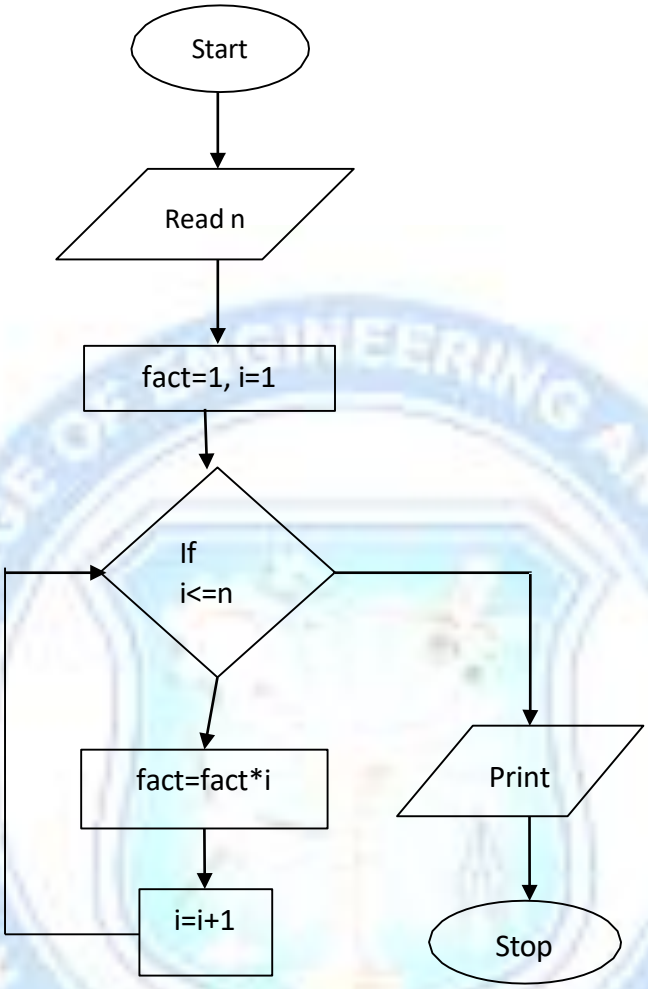
Step 6: print fact

Step 7: Stop

Pseudocode

```
Set fact = 1, i = 1
READ the value n
IF (i<=n) THEN
fact = fact *i
i =i + 1
ENDIF
WRITE fact
```

Flow chart:



18) To find whether the number is prime or not.

Algorithm

Step1: Start

Step 2: Assign i←2

Step 3: READ n

Step 4: REPEAT steps 4.1, 4.2

UNTIL i < = n – 1

Step 4.1: If (nmod I = 0) THEN

Print Not Prime

Step 4.2: i ← i +1

Step 5: IF (i=n) Then

Step 6: Print prime

Pseudocode

Set i = 2

READ n

WHILE (i < = n-1)

IF n mod i = = 0 THEN

WRITE not prime

EXIT

ENDIF

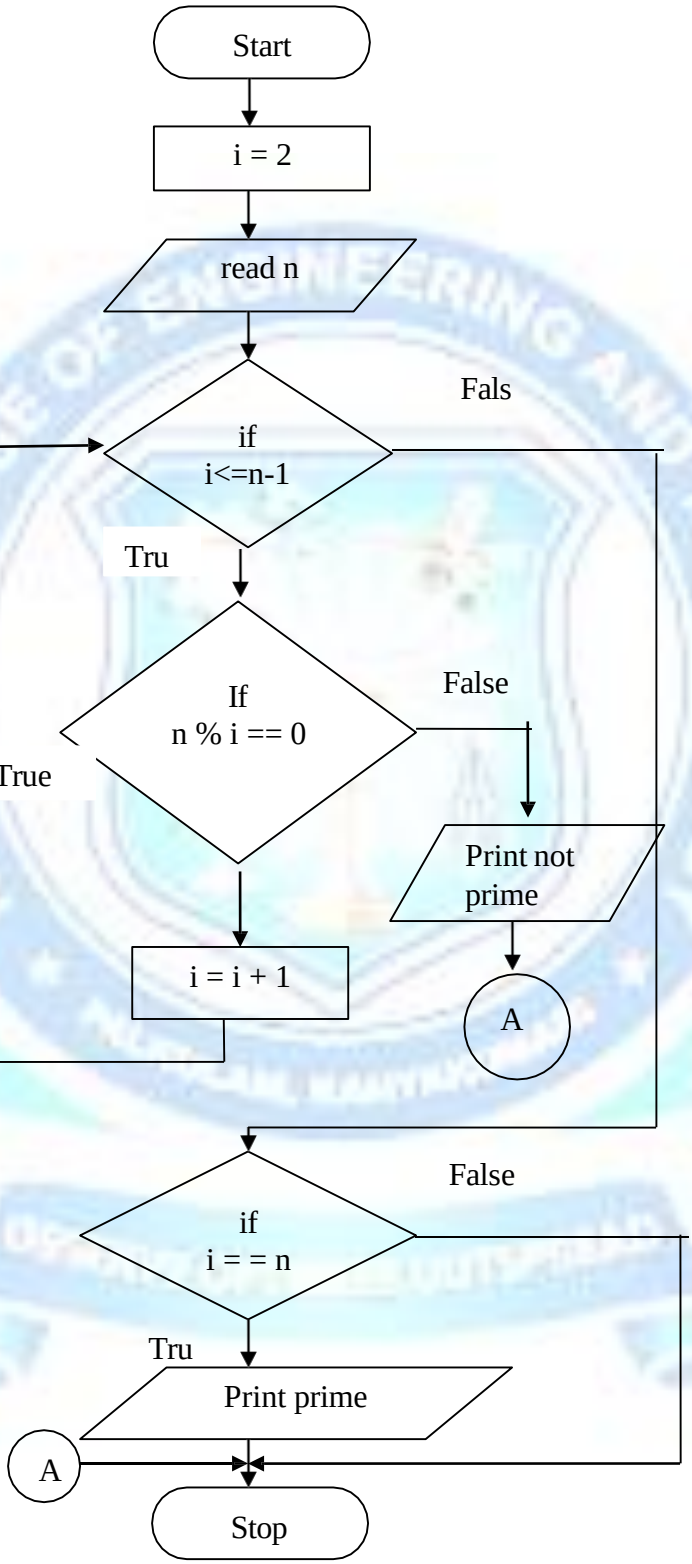
i = i + 1

END WHILE

IF (i = = n) THEN

WRITE prime

ENDIF


Start
i = 2
read n
Fals
if
i<=n-1
Tru
If
n % i == 0
False
True
Print not
prime
i = i + 1
A
False
if
i = = n
Tru
Print prime
A
Stop