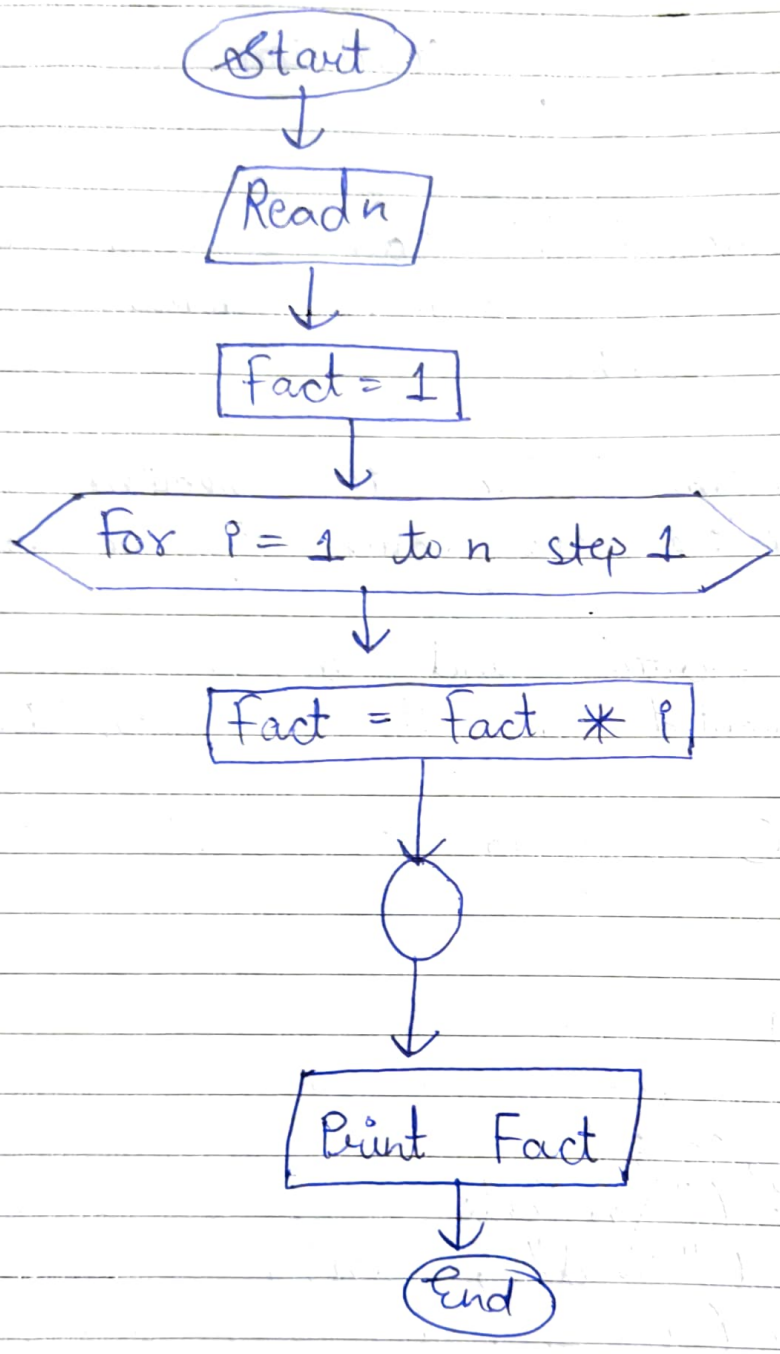



```
main ()
{
    int i, n , fact = 1 ;
    Print f ("\n Enter the value of n");
    Scan f ("%d", &n);

for (i = 1 ; i <= n ; i + 1)
```

```
    fact = fact * i ;
    Print f ("\n factorial = %d", Fact) ;
}.
```

<u>While loop</u> :- A while loop is used to execute or repeat a block of statement depending on a condition it has the following form

```
while (< condition >)
{
  < statement block >
}.
```

where condition is a relational or logical expression which will have the value true or false

<u>Eg</u> :-

```
p = 1
while (i < = 10)
{
    s = s + i ;
    p = p * i ;
    i = i + 1 ;
}
```

when this statement is executed the

Computer evaluate the value of a condition if the value ~~of~~ is True the statement block is executed and repeated until the value of the condition is false

<u>do - while statement or loop</u>

do while statement is use to execute and repeat a block of statement depending on a condition. It has the following form

```
while(<condition>) do
        {
          < statement - block >
        }
```

where condition is a relational or logical expression which will have the value T or F.

```
i = 1;
do
  {
      s = s + i;
      p = p * i;
      i++;
  } while (i <= 10);
```

when the statement is executed the computer will execute the same statement

block respective to the condition at the end of the statement block the condition is evaluated.

If the value of the condition is true the statement block is executed again & is repeated the condition is false.

Q. Write a program for sum of given digits in any number.

```
main ()
{
 int n, q, s, r ;
printf ("\n Enter n");
scanf ("%d", &n);
 q = n;
 s = 0;
while (q>0)
 {
  r = q % 10;
  s = s + r;
  q = q/10
 }
 printf ("\n sum of %d digits is : %d",
                n, s);
}.
```

Nested - for statement :-

The statement block of a for loop lie completely inside the block of another for loop. These is refered as nested - for loop.

Eg :-

```
For (i=1; i<=3; i++)
  {
    for (J=1; J<=5; J++)
    {
      <statement_block>
    }
  }
```

outer loop.

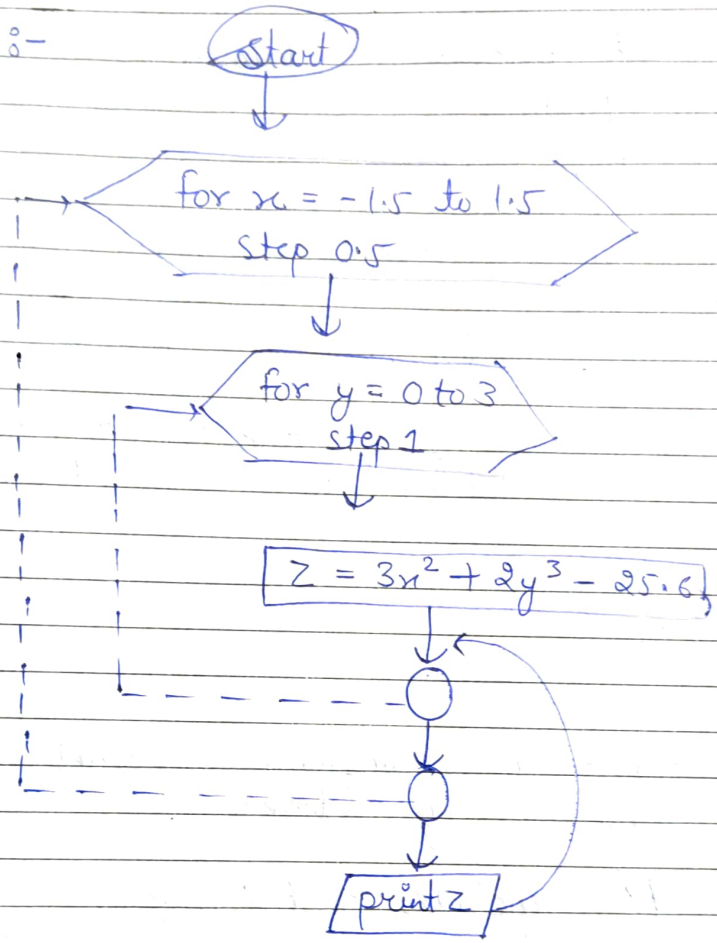
inner loop

Q. Z to be computed x and y according to $Z = 3x^2 + 2y^3 - 25.6$. Compute the value of x as x = -1.5 to 1.5 in increment of 0.5 and y = 0 to 3 in increment of 1. Write a C program to compute Z for all pair of (x, y).

Z (-1.5, 0)
Z (-1.5, 1)
Z (-1.5, 2)
Z (-1.5, 3)
Z (-1, 0)
⋮
Z (1.5, 0)

what we want in output

Algorithm :-



$x = -1.5$ $\quad$ $y = 0$

$Z(-1.5, 0) = 3 * (-1.5)^2 + 2 * (0)^3 - 25.6$

$Z(-1.5, 1) = 3 * (-1.5)^2 + 2 * (1)^3 - 25.6$

$Z(-1.5, 2) = ?$

$Z(-1.5, 3) = \text{-----}$

$Z(-1, 0) = \text{-----}$

$\vdots$

$Z(1.5, 3) = \text{-----}$

Q. Write a C program to print multiplication Table from 1 to 5.

```
# include < stdio.h >
  main ()
 {
  int j, i;
  for ( j = 1 , j<=5, j++)
   {
     for ( i = 1 ; i<=10 ; i++)
     {
      printf ("\n %5d x %5d = %5d", J, i,
   J * i);
     }
  }
  getch ( );
  return 0;
 }.
```

Q Write a program to evaluate the series.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \text{ ------ } \frac{1}{N}$$

```
# include < stdio.h >
int main ( )
{
  int n, i, sum = 0;
printf (" Input any number :")
scanf ( " % d", &n);
```

```
s = 0.0;
for (i = 1; i <= n; i++)
        s = s + 1.0/i;
printf("\n sum of series = %8.2f", s);
}
```

* <u>Unconditional Control statement</u> :-

1. goto statement :- The goto statement is an unconditional transfer of control statement. It is use to transfer the control from one part of the program another the place to which the control is transfer is identified by a statement label.

It has ~~st~~ the following form :-

```
goto statement :
        goto label ;
        -------
        ----
```

```
Eg :- goto display ;
      ------
      ----
      display :
      ------
      ----
```

when the statement is executed the control is transfer to the statement label display which is followed by a column

(2) <u>Break Statement</u> :-

The break statement is used to transfer the control to the end of a statement block in a loop. It is an unavoidable statement to transfer the control to the end of a switch statement after executing any one statement block.

Eg :-

```
printf("\n press B to break; any other key
                                to continue");
For (i=1; i<=60; i++)
{
  ch = getche()
  if (ch == 'B')
        break;
        ----
        ----
}
```

when this statement is executed the control is transfer to the end of the loop and the statement which are written outside the for loop will be executed when the key 'B' is press.

* Continue statement :-

The continue statement is used to transfer the control to the beginning of a statement block in a loop. It has following form

```
Continue ;
```

Eg :-

```
for (i=1 ; i<=60 ; i++)
{
    ch= getche ();
    if (ch==='c')
    {
        print ("\n C for continue");
        continue ;
    }
}
```

when this statement is executed the control is transfered to the beginning of the loop such that the loop is repeated 60 times irrespective to the key press. The message C for continue is display whenever the key is press.

Exit () :-

The exit function is used to transfer the control to the end of a program. It uses

one argument is () and the value is 0 for the normal termination or non-zero for abnormal termination.

```
if (n<0)
{
 print ("\n Factorial of negative Integer is not
available");
 exit (0);
}
```

#

```
main ()
{
 int i, n, j;
 printf ("\n Enter number of rows");
 scanf ("%d", &n);
for
    (i=1; i<=n; i++)
  {
   for
     (j=1; j<=i; j++)
      {
       print f("*");
      }
       print f ("\n");
 }
```

Enter number of rows 3

$n = 3$

when

$i = 1$	$i = 2$	$i = 3$
$i <= 3$	$i <= 3$	$3 <= 3$
$j = 1$	$j = 1$	$j = 1$
$1 <= 1$	$j <= 2$	$1 < 3$
(x) $2 <= 1$	$2 <= 2$	$2 < 3$
	(x) $3 <= 2$	$3 < 3$
		(x) $4 < 3$

output

```
*
* *
* * *
```

```
main ()
{
  int m=1 , n=5;
  for (i=n ; i>=1; i--)
  {
    for (j=1; j<= i-1; j++)
    {
      print ("  ");
    }
    for (k=1; k<=m; k++)
    {
      print (" *");
    }
      printf ("\n");
      m++;
}.
```

output :-

Q. Write a program to calculate sum and avg of +ve number and Enter a -ve no. get exit from program

```
# include <stdio.h>.
main ()
{
  int n = 50, i;
  ~~for~~ float num, sum = 0.0, avg;
  for (i=1; i<=n; ++i)
  {
  printf ("\n Enter number one by one");
  scanf ("%f", &num);
    if (num<0)
          goto end;
    sum = sum + num;
  }

end :
      avg = sum/(i-1);
      printf ("\n sum = %6.2f", sum);
      printf ("\n Average = %6.2f", avg);
```

Array :-

A list of related values is stored in RAM in the form of array values in an array are identified using array name using subscript.

Single subscripted variable is referred as 1D or linear array, 2 subscripted variable

variable is refered to a 2D or matrix array and so on

$$x = \{ \underset{x_0}{3.1}, \underset{x_1}{3.2}, \underset{x_2}{4.6}, \underset{x_3}{2.9}, \underset{x_4}{3.8}, \underset{x_5}{3.2} \}$$

These are the values and these are called $x_0, x_1, x_2, x_3, x_4, x_5$ in mathematics

This is concept of array.

we can also represent in this way

Subscript →	0	1	2	3	4	5
x	3.1	3.2	4.6	2.9	3.8	3.2

array name

Ex :- float x [50] ; Here, this shows that x is the name of array which can store maximum of 50 value

using for loop we can read list :-

```
# include <stdio.h>
main ()
{
  float x [50];
  int n;
```

```
printf ("\n How many elements");
scanf ("%d", &n);
printf ("\n Enter Elements");
for (i=0 ; i<=n ; i++)
scanf ("%f", &x[i]);
```

Suppose we have to print :-

```
for (i=0 ; i<=n ; i++)
printf ("%6.2f", x[i]);
```

if we initilize

```
# include <stdio.h>
main ()
{
float x[ ] = {3.1, 3.2, 4.6, 2.9, 3.8, 3.2};
```