Unit-3

Conditional Program Execution.

Normally The statements in a program are executed in the order in which they appear in program. This type of execution is called sequential execution

→ Some times it is necessary to

* Select a set of statements for several alternatives.
* Skip certain statements depending on some conditions in the programs and continue the execution from some other point.
* Repeat a set of statements for a known number of times or untill a specified condition is fulfilled.

Condition Control Statements

* Some time it is necessary to check the condition to make the decision.
* This involves performing a logical test, the result

* Depending upon the condition of given statement which statement will execute next is determined.

* After that, the control is transferred to that statement & starts execution from that point.

* This entire process is known as conditional execution which includes both decision making & branching.

If Statement:

* Using If statement the logical condition is tested which results in either true or false.

* If the result is true (nonzero) then statements that immediately follow "If" is executed.

* If the result is false (zero) then control transfers to next executable statement, that immediately follow it.

→ Syntax of writing (if)

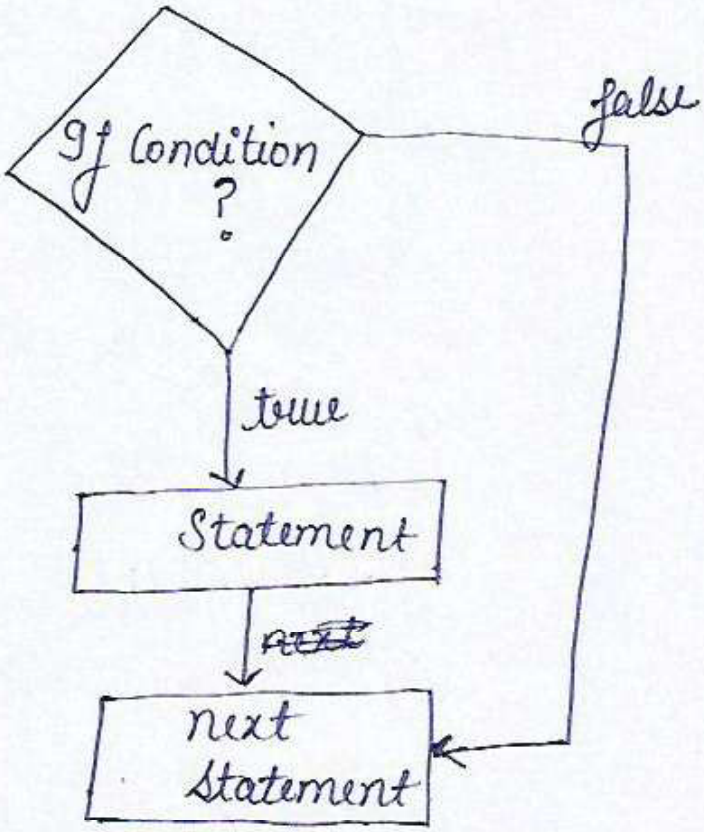
```
if (condition)
{
  statement
  ---
}
```

where:

Condition → Logical expression that result in true or false

Statement → simple or compound statement

Note: for simple (or single statement) no braces are required.



Prog

```
main ()
{ int num;
printf ("Enter a number\n");
scanf ("%d", &num);
if (num == 0)
```

```
printf("You entered Zero");
}
```

Prog

```
main()
{
    float x, y, ratio;
    printf("Enter X & Y \n");
    scanf("%f %f", &x, &y);
    ratio = x/y;
    if (ratio > 0)
    {
        printf("Ratio is greater than Zero\n");
        printf("You are using if condition\n");
    }
    printf("Ratio is less than or equal to Zero");
}
```

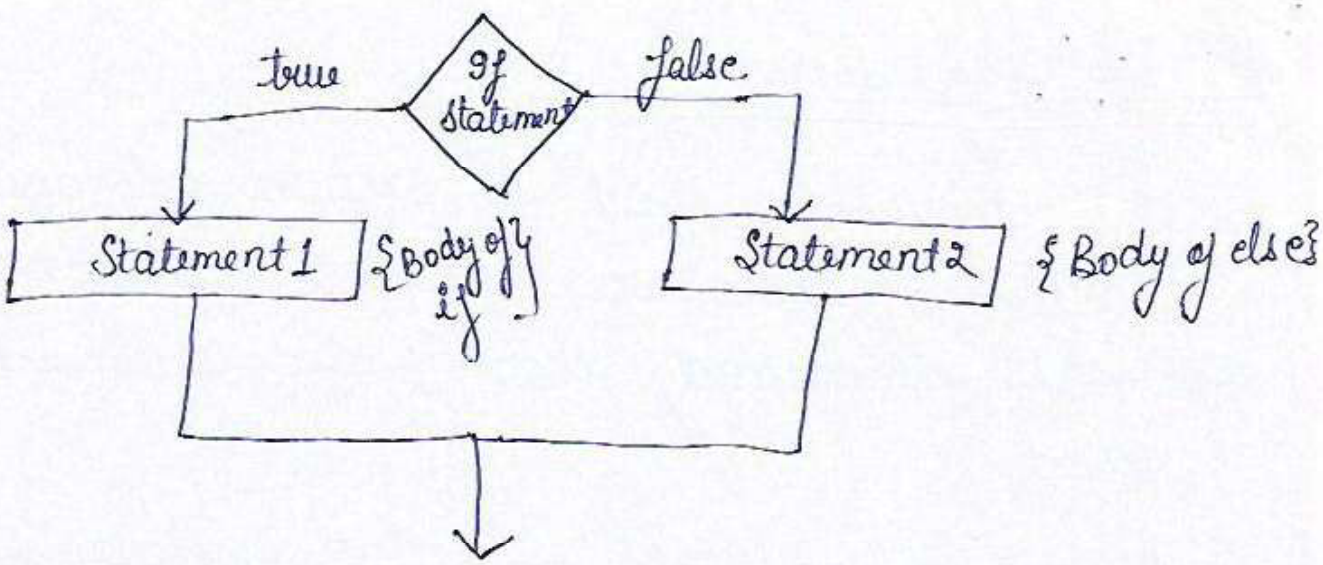
If-Else Statement

* If statement is used to execute only one action, If there are two statements to be executed alternatively then if-else statement is used.
* The if-else statement is a two way branching.

⟹ Syntax

```
if (condition)
  {
    statement1;
  }
else
  {
    statement2;
  }
```

* The condition is tested and if the result of this logical test is true, then statement 1 is executed otherwise statement2 is executed.
* After executing one of these two statement, the control transfers to the statement immediately following if-else structure.



Prog

```
main()
{
  int m, n;
  printf("Enter the value of M & N\n");
  scanf("%d %d", &m, &n);
  if (m==n)
    { printf(" Both numbers are equal\n");
    }
else
    {
      printf("Numbers are not equal\n");
    }
}
```

Output

Case 1

Enter value of M & N

26

26

Both numbers are equal

Case 2

Enter value of M & N

34

44

Both numbers are not equal.

Prog

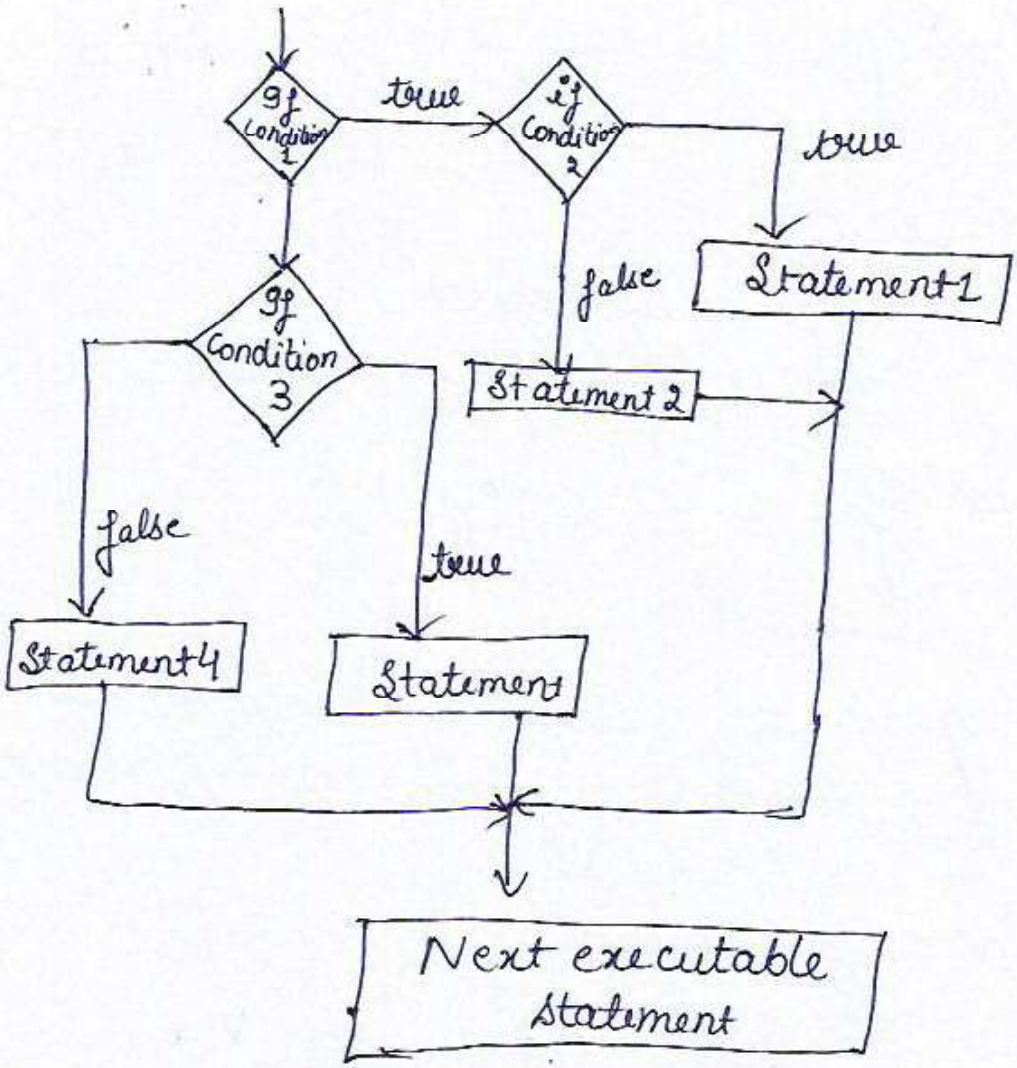
```
main()
{
  char ch;
  printf("Enter a character\n");
  scanf("%c", &ch);
  if (ch=='y' || ch=='Y')
    printf("Yes");
  else
    printf("No");
}
```

Nested If statement

* If there are more than two alternatives to select, then the nested-if statements are used.
* Enclosing if within another if is called a nested if statement.

Syntax

```
if (condition1)
{
    if (condition2)
    {
      statement1;
    }
    else
    {
      statement2;
    }
}
else if (condition 3)
    { statement 3;
    }
else {
      statement4;
    }
```



Program to find largest number out of given 3 Nos

```
main()
{ int a, b, c;
  printf("Enter the value of A, B, C \n");
  scanf("%d %d %d", &a, &b, &c);
  if (a>b)
    { if (a>c)
      printf("a is largest");
```

```
else
    printf ("C is largest\n");
}
if (b>c)
    printf ("B is largest");
else
    printf ("C is largest");
}
```

Q. Write a Program to accept two integers as coordinate of point & determine its quadrant

```
#include <stdio.h>
#include <conio.h>
void main()
{   int x, y;
printf ("Enter the value of x,y for a point\n");
scanf ("%d %d", &x, &y);
if (x>0 && y>0)
  printf (" Point lies in first quadrant\n");
if (x>0 && y<0)
  printf (" Point lies in fourth quadrant\n");
```

```
if (x<0 && y>0)
    printf ("Point lies in second quadrant \n");
if (x<0 && y<0)
    printf ("Point lies in Third quadrant \n");
if (x==0 && y==0)
    printf ("Point lies at origin");
}
```

Darling Else Problem :

In nesting if-else statement, there exists a problem when there is no matching else for every if. This problem is known as Darling else problem.

eg

```
if (condition1)
    if (condition2)
        statement1;
else
    Statement2
```

Solution

```
if (expression 1)
{
    if (expression 2)
        statement 1;
}
else
    statement 2;
```

body of if

body of else

The If- Else if ladder

* To make multiway decision based on several condition if-else if ladder is used.

* As soon as one of the given if or else if gives a true result, the following statement or block is executed, & no further comparisions are peryormed.

* If none of the condition is true, the final else is executed

* Final else is optional, if final else is not present no action took place even if all the condition are false.

Syntax

```
if (condition)
    statement;
else if ( condition)
    statement;
else if (condition)
    statement;
else
    statement;
```

Prog

```
if (result>=75)
    printf (" Grade A \n");
else if ( result >=60)
    printf ( " Grade B \n");
else if ( result >= 40)
    printf ("Grade C \n");
else
    printf (" fail");
```

Q. Write a program in C to test whether the given year is leap year or not.

Prog

```
#include <stdio.h>
#include <conio.h>
void main()
{
  clrscr();
  int year;
  printf("Enter the year to be checked\n");
  scanf("%d", &year);
  if (year % 4 == 0)
    { printf("Year is leap");
    }
  else
    { printf("Year is not a leap year");
    }
  getch();
}
```

Q. Write a program to check whether a given number is "odd" or "even".

Program

```
#include <stdio.h>
#include <conio.h>
Void main()
  {
    clrscr();
    int n;
    printf("Enter the number");
    scanf("%d", &n);
   if (n%2==0)
     {
       printf("Number is even");
     }
  if (n%2!==0)
     {
       printf("Number is Odd");
     }
   getch();
  }
```

Q) Calculation of Area of Circle or Rectangle. depending upon choice of user (by entering 'C' or 'R')

Program

```
#include <stdio.h>
#include <conio.h>
Void main()
    {
      Char ch;
      float a1, a2, area;
    printf("Enter C for area of circle OR
                R for area of Rectangle");
    scanf("%c", &ch);
  if (ch=='C' || ch=='c').
    {
      printf("Enter Radius of circle\n");
       scanf("%f", &a1);
       area = 3.14 * a1 * a1;
       printf("The area of circle is %f", area);
    }
  if (ch=='R' || ch=='r')
```

```
{
printf("Enter length & width : ");
scanf("%f %f", &a1, &a2);
area = a1 * a2;
printf("Area of rectangle is %f", area);
}
}
```

Switch Statement :

* Switch statement is a multi-way decision that tests whether an expression matches one of the a number of constant integer values and branches accordingly.
* This is another form of multiway decision.

Syntax

```
switch (expression)
{
  case Value1 :
                statements
                break;
  case Value2 : statements
```

```
    case value3:
                statement
                break;
    default :
                statement
}
```

* The keyword switch is followed by switch variable or expression in parenthesis
* The case keyword is followed by a constant, followed by colon.
* When switch statement is executed, the expression is evaluated and its value is successively compared with case constants.
* When a match is found, the statement sequence associated that case is executed untill the break statement or end of switch is reached
* Default statement is executed if no match is found

Prog

```
#include <stdio.h>
void main()
{
  int n1, n2, op;
  printf ("Enter two numbers");
  scanf ("%d %d", &n1, &n2);
  printf ("\n Menu");
  printf ("\n 1. Addition");
  printf ("\n 2. Substraction");
  printf ("\n 3. multiplication");
  printf ("\n 4. Division");
  printf ("\n Select your choice by entering
          a Number");
  scanf ("%d", &op);
  switch (op)
  {
   case 1:
          printf ("\n Addition is %d", n1+n2);
          break;
  case 2:
      printf ("\n Difference is %d", n1-n2);
          break;
```

```
Case 3:
        printf ("\nMultiplication is %d", n1*n2);
        break;
Case 4:
        printf ("\nQuotient is %d", n1/n2);
        printf ("\n Remainder is %d", n1%n2);
    }
}
```

Q. Write a program that calculate area of Circle, Rectangle, Triangle, Square by displaying Menu.

Program

```
#include <stdio.h>
#include <conio.h>
void main()
    { clrscr();
      int code;
    float side, base, height, length, breadth,
      area, radius;
printf("menu\n");
```

```
printf ("\n 1. Circle");
printf ("\n 2. Square");
printf ("\n 3. Triangle");
printf ("\n 4. Rectangle");
printf ("\nEnter your choice by entering number");
scanf ("%d" & code);
    switch (code)
      Case 1:
            printf ("\n Enter Radius of circle");
            scanf ("%f", & radius);
            area = 3.14 * radius * radius;
          printf ("\n the area of circle is %f", area);
            break;
      Case 2:
            printf ("\n Enter side of square");
            scanf ("%f", & side);
            area = side * side;
          printf ("\n the area of square is %f", area);
            break;
```

```
Case 3:
        printf("\n Enter the Base & height of
                                triangle");
        scanf("%f %f", &base, &height);
        area = 0.5 * base * height;
        printf("\n Area of triangle is %f", area);
            break;

Case 4:
        printf("\n Enter length & breadth of Rectangle");
        scanf("%f %f", &length, &breadth);
        area = length * breadth;
        printf("\n Area of Rectangle is %f", area);
            break;

    default: printf("\n Error in code");
}
}
```

Q. Write a program to accept a letter & print grade as per given table:

letter	Remark
S	Super
A	Very Good
B	Fair
Y	Absent
F	Fail

Prog

```
#include <stdio.h>
#include <conio.h>
Void main()
    {
      Char letter;
      printf ("Enter the letter\n");
      scanf ("%c", &letter);
     switch (letter)
     {
      case 'S' :
                printf ("Super\n");
                break;
```

```
Case 'A':
        printf("Very Good\n");
          break;
Case 'B':
        printf("Fair\n");
          break;

 Case 'Y':
        printf("Absent\n");
            break;

  case 'F':
        printf("Fail\n");
          break;

  default:
        printf("wrong choice");
   }

}
```

GOTO Statement

⇒ C Language supports unconditional control statement "goto" to transfer the control from one point to another in program.

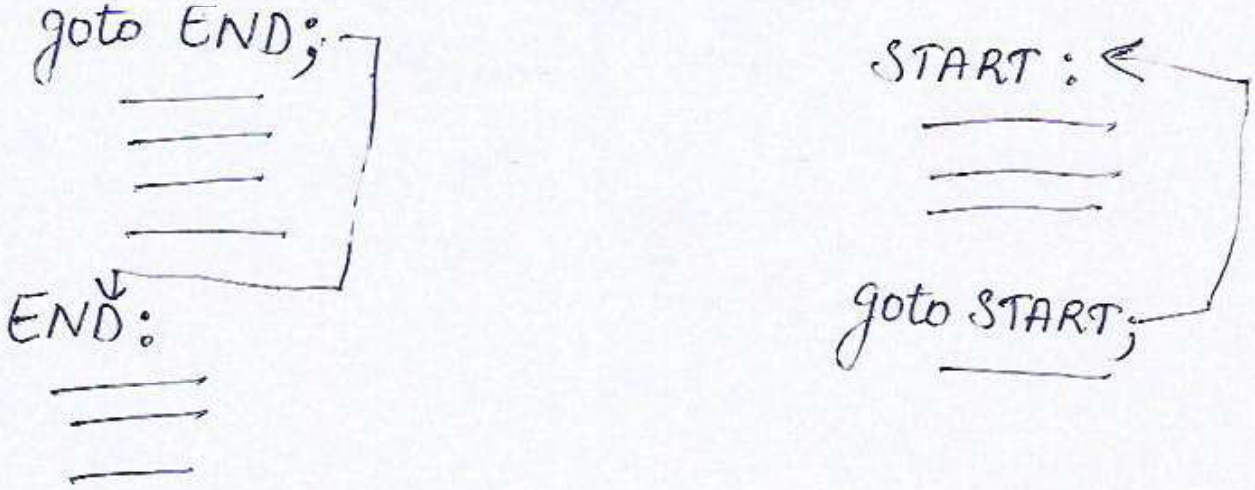
Syntax

```
goto label;
```

where goto → a keyword
label → a symbolic constant where the control will transfer

Note: A label can be placed anywhere in the C Program either before or after goto statement.

eg



Program

```
# include < stdio.h >
# include < conio.h >
 void main ( )
    {
      int goals;
    printf ("Enter Number of goals \n");
    scanf ("%d", &goals);
    if ( goals <= 5 )
      goto sos;
   else
      {
       printf (" Good Player");
      }
  sos:
      printf (" Try Your Best");
}
```

Looping

⇒ looping is a powerful programming technique through which a group of statements are executed repeatedly, untill a specific condition is satisfied

⇒ looping is also called a repetitive or an iterative control mechanism.

⇒ Loop consists of two parts, one is called the body of loop and other is called control statement.

⇒ Control statement perform logical test whose result is either true or false. If the result of this test is true, then the body of loop will execute otherwise loop terminates.

⇒ If control statement is placed before the body of loop, it is called entry control loop & if control statements are at end of (body) it is called exit controlled loop.

⇒ Three efficient looping mechanism are:

① Initialisation: To set initial value of loop counter

② Decision: A test condition to decide whether loop will execute or not

③ Updation: Incrementing or decrementing counter value.

Examples of Loops

① While statement

② do-while statement

③ for statement

While Loop

While loop executes untill the condition specified in while becomes false

Syntax

```
while (expression)
{
    statements;
}
```

where

while → is a keyword

expression → logical expression that results either true or false

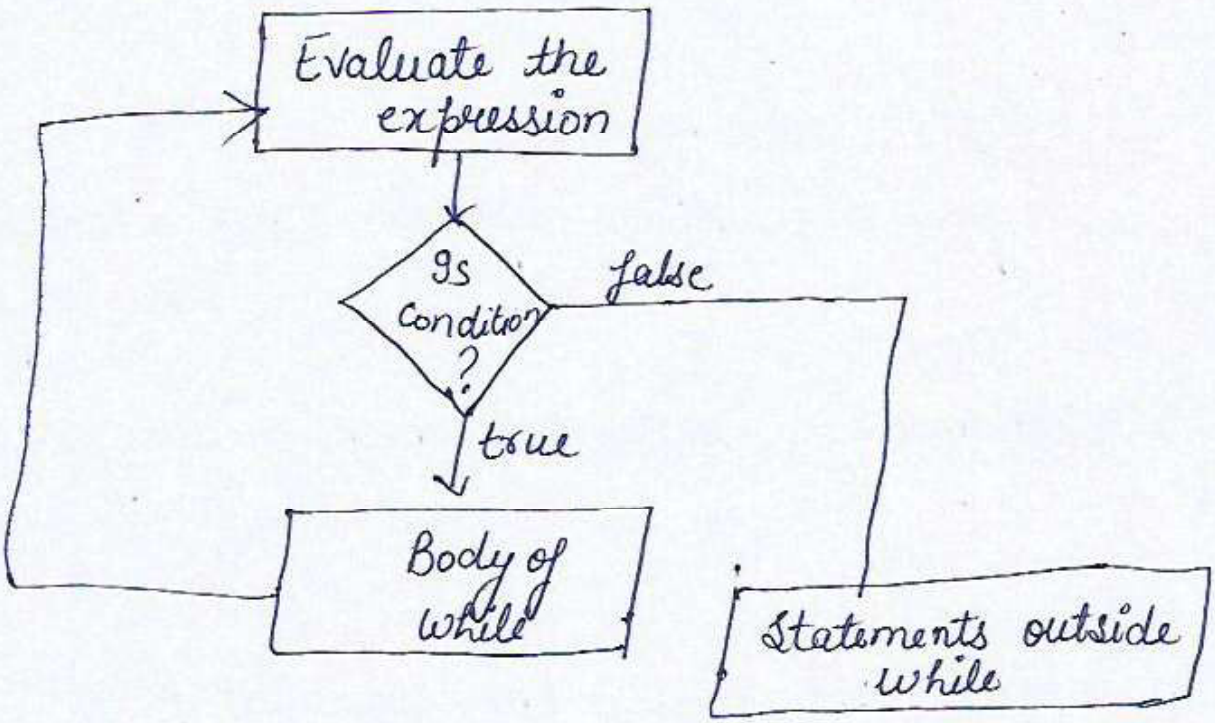
Statements → statement which executes for while

* In while loop, firstly the logical expression is tested and if the result is true, the statements in while body will executes, this entire process continues untill the logical expression become false.

* This is also called as pretested loop.

* The value of loop counter will change during each pass of while loop.

Note: If the value of loop counter doesn't changes in each pass, the loop will enters into infinite loop condition



Program to print your name in desired number of times

```
# include <stdio.h>
void main()
  {
    int R;
    int i;
  printf("How many times you want to print your name\n");
    scanf("%d", &R);
    i=0;
    while (i<R)
    {
      printf("Ashish Chauhan\n");
      i++;
```

Qe Program to find sum of first 'n' numbers

Prog
```
# include <stdio.h>
# include <conio.h>
Void main ( )
    {
      Clrscr ( );
      int n, i, sum=0;
     printf ("Enter the number \n");
     scanf ("%d", & n)
      i=1;
     while (i<=n)
      {
         sum += i;
          i++;
      }
     printf ("The sum of given number is %d", sum)
    }
```

Qe Factorial of number

```
while (num > 0)
    {
      ans = ans * num;
      num--;
    }
```

Do while loop

⇒ Do while loop is used to execute a set of statements repeatedly, untill the logical test result becomes false.

⇒ This is called post-test loop because the test is made at the end of each pass

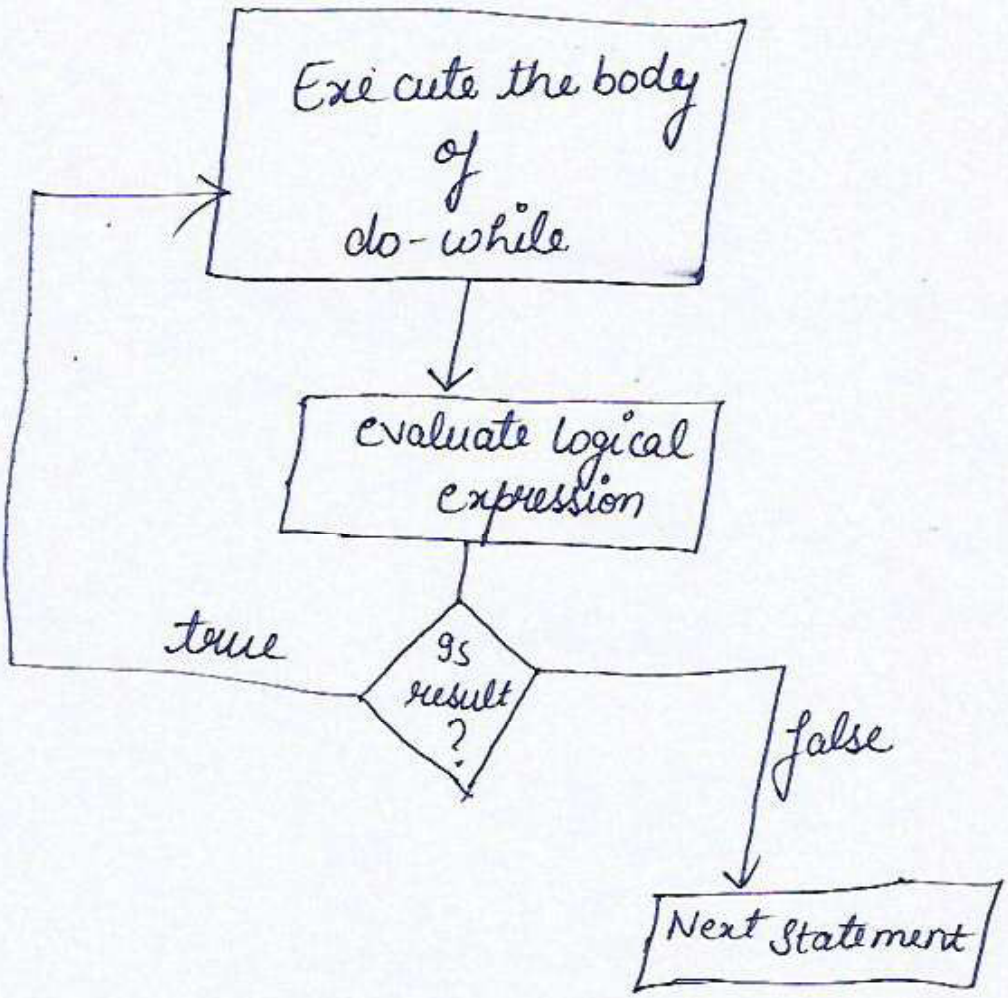
Syntax

```
do
{
  statements
}
while (expression);
```

Note:

(i) The body of do while will execute atleast once because the logical test is carried out at the end of pass.

(ii) The body of do while must contain statement to modify the variable tested in the logical expression condition of while.



Qo Program to find sum of all odd numbers between 1 to 50

```
#include <stdio.h>
void main()
  {
    int odd_num=1, sum=0;
    do {
        sum += odd_num;
        odd_num+=2;
       }
    while (odd_num<=50);
```

```
printf ("Sum = %d \n", sum);
}
```

Q0 Print Colour code of Rainbow as long as user wants

```
# include <stdio.h>
# include <conio.h>
void main()
    {
      clrscr();
    int color-code, choice;
  do
    { printf ("Enter Color Code \n");
      scanf ("%d", & color-code);
  switch (color-code)
   {
     case 1: printf ("Color is voilet \n");
             break;
     case 2:
             printf (" color is Indigo \n");
             break;
     case 3:
             printf (" Color is Blue \n");
             break;
```

```
case 4:
        printf ("Color is Green\n");
            break;
case 5:
        printf ("Color is Yellow\n");
            break;
case 6:
        printf ("Color is orange\n");
            break;
case 7:
        printf ("Color is Red\n");
            break;
default: printf ("Rainbow has only 7 colors\n");
}
printf ("Do you want to continue if yes press1 if
        no print 0\n");
scanf ("%d", &choice);
}
  while (choice==1);
}
```

Program to find sum of series

$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7}$ ----- upto n terms

```
# include <stdio.h>
# include <conio.h>
Void main ()
    {
      int n, sign, const;
      float sum=0, term, num, den;
      printf ("Enter the number of terms in series\n");
      scanf ("%d", &n);
       sign = 1;
       num = 1;
       den = 1;
       Const = 1;
     do {
         term = (num/den) * sign;
         sum = sum + term;
          den++;
          sign *= (-1);
       2. Const++;
```

```
while (const <= n);
printf ("Sum of series = %d \n", sum);
}
```

For loop

⇒ For loop is used when the programmer knows how many times a set of statements are to be executed.

Syntax

```
for (expression1 ; expression2 ; expression3)
    {
      statements ;
    }
```

where:

expression1 → initialises the loop index before loop begins

expression2 → Conditional expression that tests whether the loop will continue or stops

expression3 → modifies loop index after each iteration

statements → body of loop which executes for loop

Q. Program to print first 10 numbers using FOR.

```
#include <stdio.h>
#include <conio.h>
Void main()
    {
      int n, i;
    for(i=0; i<=10; i++)
      {  printf("\n");
         printf("%d", i);
      }
    }
```

Also see

```
#include <stdio.h>
Void main()
{ int i=0, limit=5;
  for(; i<limit;)
    { i++;
      printf("%d\n", i);
    }
}
```

Q. Write a program to calculate the sum of fibonacci series upto 100 terms.

```
# include < stdio.h >
# include < conio.h >
  void main()
    {
      int i ;
      int fib1 = 0, fib2 = 1, fib3, count = 0;
printf ("%d", fib1);
printf ("%d", fib2);
      for (i = 0; i <= 100; i++)
        {
          fib3 = fib1 + fib2;
            printf ("%d", fib3);
          fib1 = fib2;
           fib2 = fib3;
        }
```

Nested for Statement

⇒ In nested "for" there are two for statements, The first one is called outer loop & the second one is the inner loop.

⇒ The statement1 & statement2 are repeatedly execute the number of times specified by the maximum limit of the second for loop.

```
for (exp11; exp12; exp13)
{
    for (exp21; exp22; exp23)
    {
        Statement 1;
        Statement 2;
    }
}
```

Prog To print.

```
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

Sol^n

```
# include <stdio.h>
void main ()
{
  int i, j, num=5;
  printf("Number Pattern \n");
  for (i=num; i>1; i--)
  {
    for (j=1; j<=i; j++)
    {
      printf("%d", num);
    }
    printf("\n");
  } num--

}
```

Program to print following Pattern

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

```
# include < stdio.h>
void main ()
  {
    int i,j;
    Char ch = '*';
  for (i=0; i<5; i++)
    { for (j=0; j<=i; j++)
      {
        printf ("%c", ch);
      }
      printf ("\n");
    }

  }
```

Jump in loop

Break Statement:

→ Break statement allows to terminate the execution of loop & jump out of the loop, without getting back to conditional test.

→ when break statement is encountered inside a loop, the loop is immediately terminated & control of execution is ~~ter~~ transferred to next statement immediately following the loop.

eg

```
for (i=0; i<100; i++)
  {
    printf ("%d", i);
    if (i==10)
      break;
  }
_____
_____ next statement
_____
```

Program to check the sum of first 'n' numbers & if the sum is greater than 20 print the sum

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int sum=0, n, i;
    printf("Enter the limit \n");
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        sum+=i;
        if (sum>20)
            break;
    }
    printf("The sum is %d", sum);
}
```

Continue Statement :

→ Continue statement is used for the next iteration of the loop to take place, skipping any code in between.

→ It is used to terminate the current iteration & continue with the next iteration of the loop.

→ Continue statement causes the updation & then the conditional test portion of loop to execute.

eg

```
for(i=1; i<=2; i++)
{
    for(j=1; j<=2; j++)
    {
        if(i==j)
            continue;
        printf("%d%d\n", i, j);
    }
}
```

output

```
12
21
```

```
#include <stdio.h>
void main()
{
    int kout; sum1 = 0, sum2 = 0, limit, rem;
    printf(" Enter limit\n");
    scanf("%d", & limit);
    for (kout = 0; kout < limit; kout++)
    {
        rem = (kout % 2);
        if (rem == 0)
        {
            sum1 += kout;
        } continue;
        sum2 += kout;
    }
    printf (" sum1 = %d\n", sum1);
    printf (" sum2 = %d\n", sum2);
}
```