3. Control structure in 'C'

Control Structure for decision Making

1. if…else statement
2. switch statement
3. go to statement
4. break statement

Decision Making with If statement.

1.Simple if

- General Syntax of simple if statement is as follow:
- **Syntax:**

```
If(condition)
 {
        Statement 1;
        Statement 2;
 }
Statement - X;
```

- In the above syntax the statement block may be a single statement or a group of statements.
- The simple if statement is executed in the following order.
  1. first the condition is checked.
  2. if the condition is true then the statement block is executed and then statement-x is executed.
  - if the condition is false then only statement –x is executed.

Flow chart :

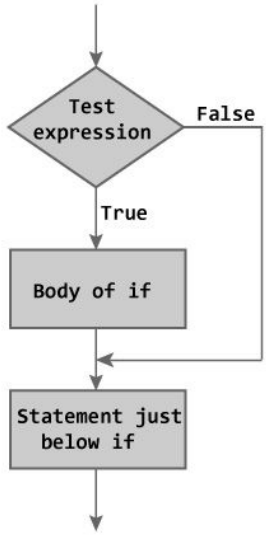


Figure: Flowchart of if Statement

Example :

```
If(x>0)
{
        printf("X is positive number");
 }
printf("General statement ");
```

2. If..Else statement.

- The general syntax of if..else statement is as follow:
- **Syntax:**

```
If(condition)
    {
        True Block statement(s);
    }
Else
    {
        False block statement(s);
    }
```

The if..else statement is executed in the following order:

1. first the condition is checked.
2. if condition is true, the true statement is executed.
3. if condition is false, the false statement is executed.

Flow chart :

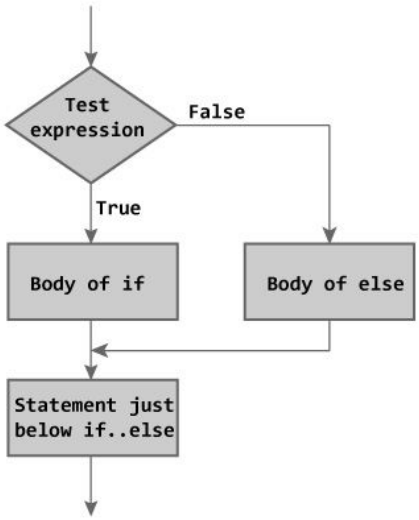


Figure: Flowchart of if...else Statement

Example :

```
if(x>=0)
{
        printf("X is positive number");
}
else
{
            printf("X is negative number");
 }
```

3.Nesting if-else statement.

- When the more then one condition is to be checked then we can use nesting of if..else first the condition is checked.
- The syntax of nesting if..else statement is as follow:

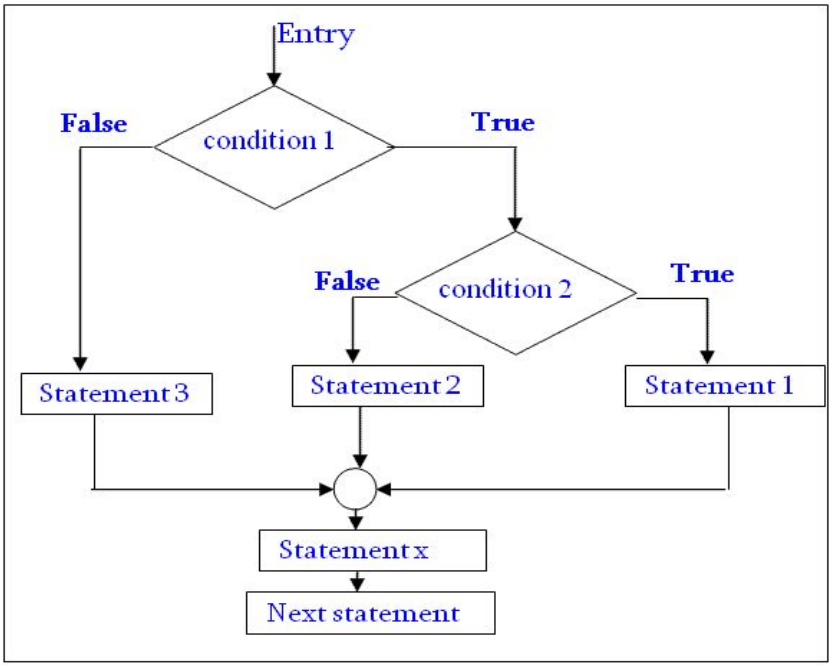
Syntax:

```
If(condition 1)
{
    If(condition 2)
    {
        Statement-1;
    }
  Else
    {
        Statement-2;
    }
}
Else
{
    Statement-3;
}
```

The nesting if-else statement is executed in the following manner.

1. First the condition 1 is checked.
2. if the condition 1 is true then condition 2 is checked. If condition 2 is true then statement-1 is executed.
3. but if the condition 2 is false the statement-2 is executed.
4. if condition 1 is false the statement-3 is executed.

Flow chart :



Example :

```
If(a>b)
 {
        If(a>c)
        {
            Printf("a is maximum");
        }
        Else
        {
            Printf("c is maximum");
        }
 }
Else
{
        If(b>c)
        {
            Printf("b is maximum");
        }
        Else
        {
             Printf("c is Maximum");
        }
}
```

4.Else-if ladder.

- The if..else ladder statement provide two-way decision where we select one of the alternative.
- It is used for multiple choice.
- The two way decision is done by nested if..else is not sufficient.
- Following is the syntax of the if..else ladder.

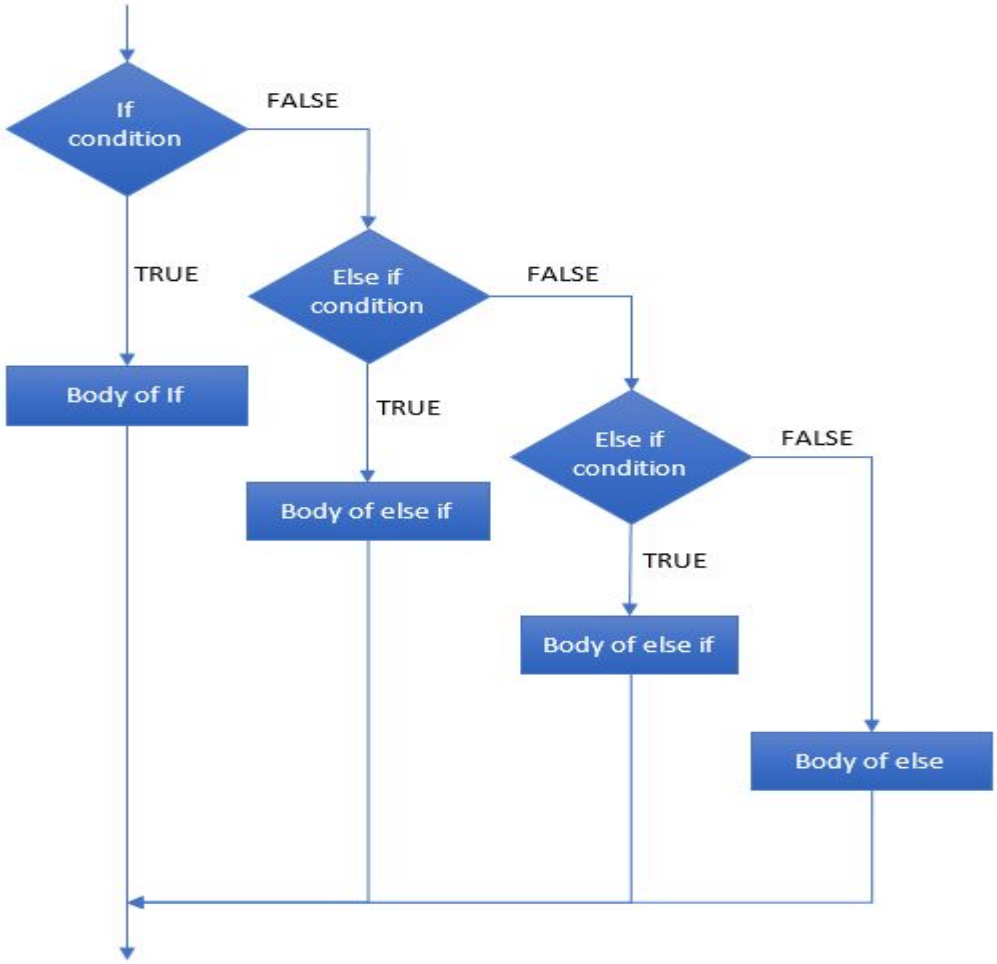
Syntax:

```
If(condition-1)
{
        Statement(s)-1;
}
Else if(condition-2)
{
        Statement(s)-2;
}
Else if(condition-3)
{
        Statement(s)-3;
}
   .
   .
   .
Else if(condition-N)
{
        Statement(s)-N;
}
Else
{
        Default statement(s)-N;
}
```

- **if-else-if ladder is executed in the following order:**

1. first condition-1 is executed , if the condition-1 is true then the statement-1 is executed.
2. if the condition-1 is false then condition-2 is checked. If condition-2 is true then statement-2 is executed.
3. This procedure repeated until all the condition is checked. if all the condition became false then the default statement is executed.\

Flow chart :



Example :

```
if( n == 1)
{
   printf("Monday");
}
 else if(n==2)
 {
   printf("Tuesday");
 }
  else if(n==3)
 {
   printf("Wednesday");
 }
  else if(n==4)
 {
   printf("Thursday");
 }
  else if(n==5)
 {
   printf("Friday");
 }
  else if(n==6)
 {
   printf("Saturday");
 }
  else if(n==7)
 {
   printf("Sunday");
 }
 else
 {
   printf("Invalid input");
 }
```

Decision Making with Switch statement.

- The switch statement is also known as multi-choice or multi decision statement.
- Writing the code using the multiple if..else becomes lengthy and also difficult to manage. Using switch statement it is done by easy.
- It provide the choice for each value of variable or expression.
- The switch statement test the value of a given variable against a list of case values and when the match is found, a statement associated with the case is executed.
- General syntax of switch case statement is as follow:

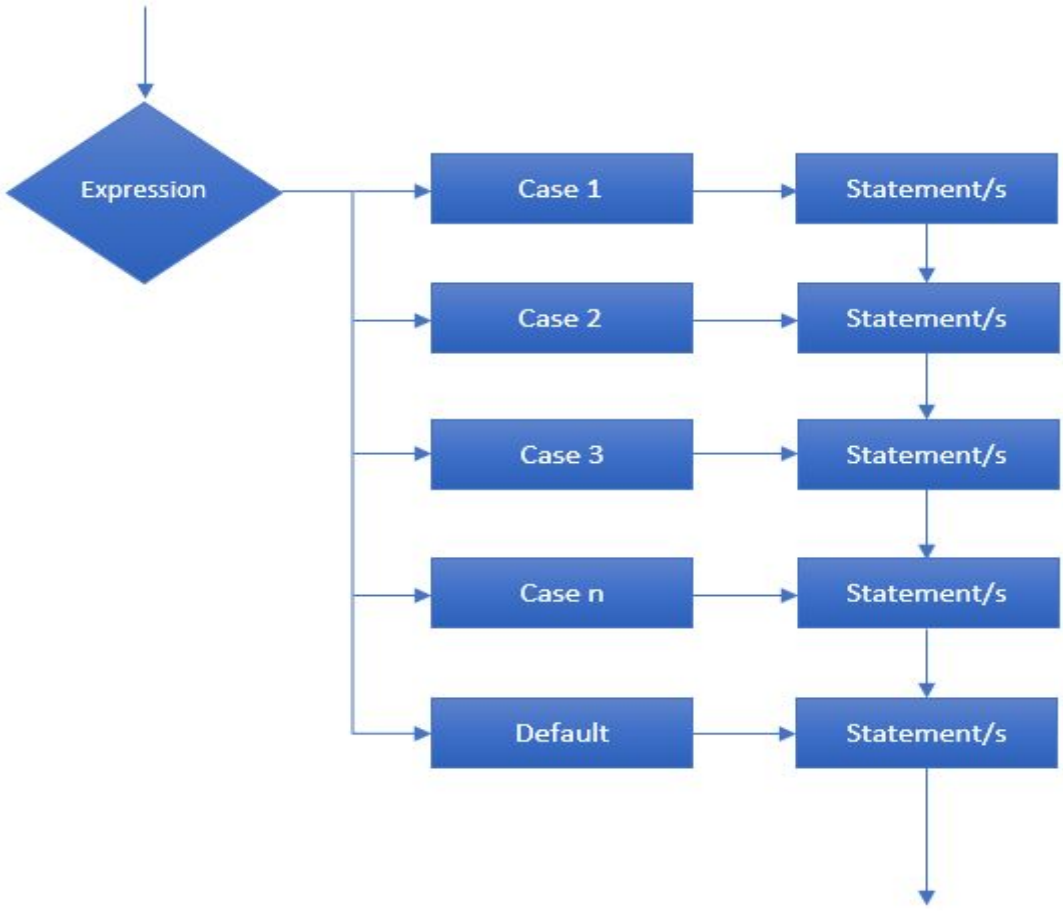
```
Switch(variable name or expression)
{
 Case 1:
        statement(s)1;
        Break;
 Case 2:
        statement(s)2;
        Break;
              .
              .
 Case N:
        statement(s)N;
        Break;
 Default  :
        default_statement(s);
}
```

- The expression or variable name is an integer or characters.
- Value-1,value-2 is constant or known as case labels ,each case label value must be unique with a switch statement. each case label must end with (;).

- **The switch..case statement is executed in the following order:**

1. the value of the expression is compared against the value of case label.
2. if the case is found whose value match with value of the expression ,then the statement associated with the case is executed. there is a single statement or multiple statement. There is break which send the control to the next statement.
3. if the value of the expression does not match with any case value then the statement associated with the default case is executed.

Flow chart :



Example :

```
switch(ch)
{
      case '+':
         printf("%d + %d = %d",a, b, a+b);
         break;
      case '-':
         printf("%d - %d = %d",a, b, a-b);
         break;
      case '*':
         printf("%d * %d = %d",a, b, a*b);
         break;
      case '/':
         printf("%d / %d = %d",a, b, a/b);
         break;
      default:
         printf("Error! operator is not correct");
   }
```

Break statement :

- We have use the break statement with the switch statement.
- The function of of break statement is exit form the switch body.
- If it is not written after each case statement, then control pass to the next statement ,so remaining statement of the next case statement will also execute even if the case value do not match and the program will not function properly.

Default keyword :

- When this keyword is write then the statement written in that part get executed if any of the previous case value do not match.

- This keyword is written after the all the cases in the switch case.

Decision Making with goto statement.

Use: the goto statement in the c programming is used to transfer the control unconditionally from one part of the program to other part of the program.

- 'c' language provide a unconditional branching mechanism called as goto statement.
- The 'c' is a structural programming language where use of goto statement is a dangerous because the use of the goto statement in the program makes it difficult to understand and debug.
- The syntax of the goto statement is as follow:
- Syntax: **label:**

  .....

  .....

  Goto label;

- Here , label is the label to the statement to which goto transfer control.
- Following shows the two possible use of goto statement.
  1. **forward reference**
  2. **backward reference**

Forward jump	Backward jump
Goto label; .... Label: Statement;	Label: Statement; .... Goto label;
Target statement comes after the goto	Target statement comes before the goto.

The ? : operators.

- The conditional operator is also known as ternary operator. It is known as ternary operator because it has three operands.
- The general form of ternary operator is follow:

  (Condition)? Statement-1: statement-2;
- First the condition is checked. If the condition is true then the statement is followed the? Is executed otherwise the statement followed the : is executed.

Example:

(a>b)? printf(“a is max”): printf(“b is max”);

- Here if the value of a is greater than b then it prints a is max otherwise it print b is max.

Continue statement :

- Continue statement is mostly used inside loops. Whenever it is encountered inside a loop, control directly jumps to the beginning of the loop for next iteration, skipping the execution of statements inside loop’s body for the current iteration.

Syntax:

continue;

Example:

```
for (int j=0; j<=8; j++)
{
        if (j==4)
        {
                continue;
        }
        printf("%d ", j);
}
```

o/p: 0 1 2 3 5 6 7 8

When j=4, the program encountered a continue statement, which makes it to jump at the beginning of for loop for next iteration, skipping the statements for current iteration.

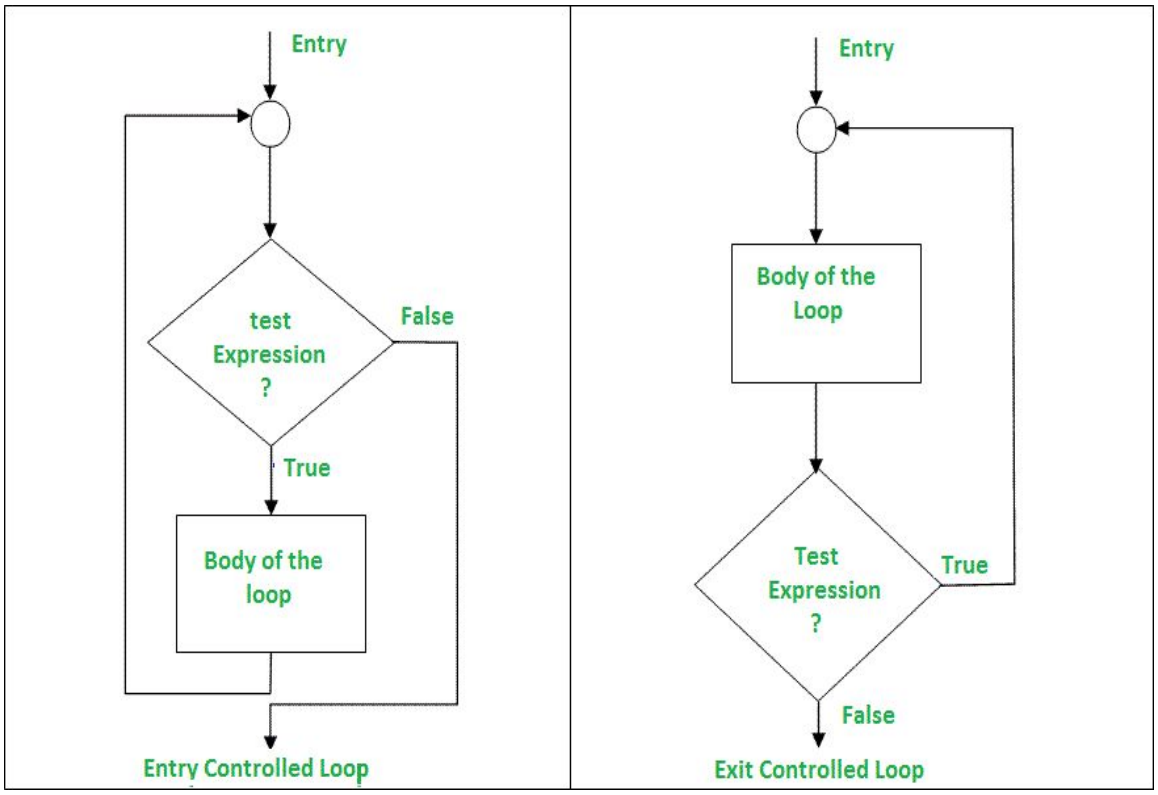
Control Structure for Looping

1. while Loop
2. do…while Loop
3. for Loop

- **There are two parts of loop.**
  1. **condition**: the control statement tests some some condition .if condition is satisfied then the loop is executed otherwise the statement follows the loop is executed.
  2. **body:** this statement consists of single or group of statement.
- Depending on where the condition is checked, we can have two types of loop structure:
  1. **Entry control loop.**
  2. **exit control loop.**

  - **Entry control loop**: the condition is written first and then the body of statement. If the condition is tested before the body of loop is called Entry control loop. if condition is true then the body of loop is executed otherwise the loop is not executed.
  - **Exit control loop:**the body of statement is written first then the condition is written. If The condition condition is tested after the body of the loop then it is known as exit control loop. So first body of the loop is executed and then the condition is checked.

Difference between Entry Controlled & Exit Controlled Loop in C.



Entry Controlled	Exit Controlled Loop
Test condition is checked first, and then loop body will be executed.	Loop body will be executed first, and then condition is checked.
If Test condition is false, loop body will not be executed.	If Test condition is false, loop body will be executed once.
Examples: for loop & while loop.	Examples: do while loop
Entry Controlled Loops are used when checking of test condition is mandatory before executing loop body.	Exit Controlled Loop is used when checking of test condition is mandatory after executing the loop body.

1. While loop :

- While loop is the entry control loop. Because in while loop first the condition is checked.

Syntax:

```
While(condition)
{
        Body of the loop;
}
```

- {} is known as body of the loop.
- The while loop is executed in the following format.
- Here the condition is evaluated first and if it is true then the statement in the body of the loop is executed.
- After executing body , the condition is evaluated again and if it is true body is executed again. This process is repeated as long as the condition is true. The control move out once the condition is false.

Flow chart :

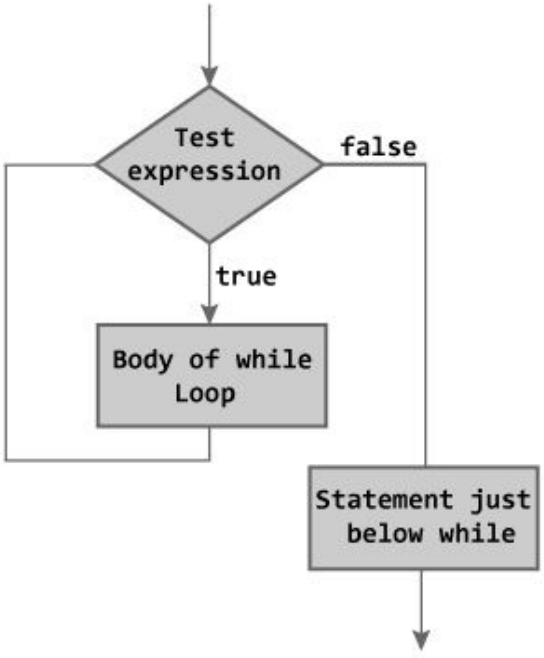


Figure: Flowchart of while Loop

Example:

```
while(i <= 10)
{
   printf("%d\n",i);
   i++;
 }
```

2. Do While loop :

- Do..while loop is a exit control loop.Because after the executing the body of the loop the condition is checked .
- **Syntax:**

```
do
{
        Body of the loop
 }while(condition);
```

- **The do while loop is executed in the following format.**

- In do..while loop the body is executed first and then the condition is checked.
- If the condition is true the body is executed again and again, as long as the condition is true.
- This ensure that the body of the loop is executed at lease once even if the condition is false first time.
- The control moves out of loop once, the condition become false.

Flow chart :

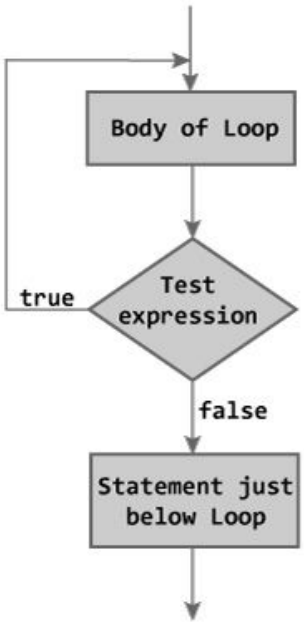


Figure: Flowchart of do...while Loop

Example:

```
do
{
   printf("%d\n",i);
   i++;
 }while(i <= 10);
```

3.for loop :

Syntax:

```
For(initialization;condition;increment or decrement)
{
    Body of the loop;
}
```

- **Initialization:** We require one variable which is used to control the loop, which is called as control variable. The control variable is initialized in the initialization expression.

- **Condition:** the condition statement is checked the value of the control variable. If the condition statement is true, the body of the loop is executed.
- **Increment/Decrement:** The increment/decrement of the control variable is done in this part. after the incrementing/decrementing the value of control variable is tested using condition if condition is true than again the body of loop is executed and this process is repeated until the condition become false.

Flow chart :

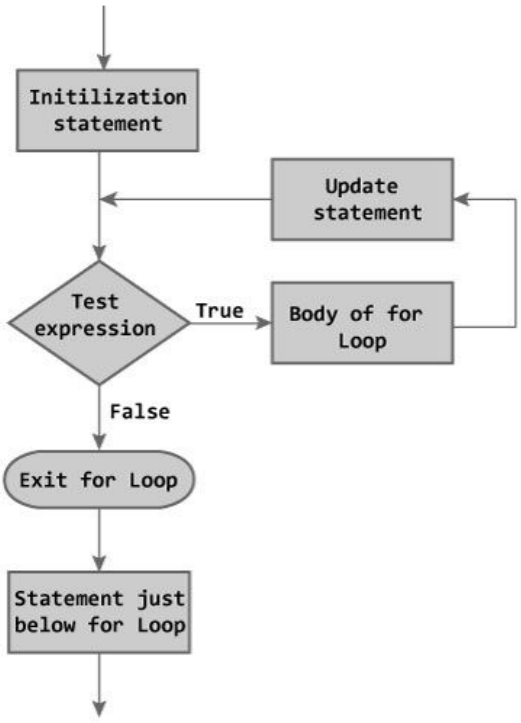


Figure: Flowchart of for Loop

Example:

```
for(i=1;i<=10;i++)
{
      printf("%d\n",i);
}
```

Nested for loop :

- ➢ When the loop inside another loop is called nesting of loops.
- ➢ The nesting can be for any number of levels, for certain problem the nesting of loop are needed.

The outer loop should not end between the starting of inner loop

Syntax:

```
for (initialization;condition;increment or decrement)
{
      for (initialization;condition;increment or decrement)
      {
         statement(s);
      }
  statement(s);
}
```

Flow chart :

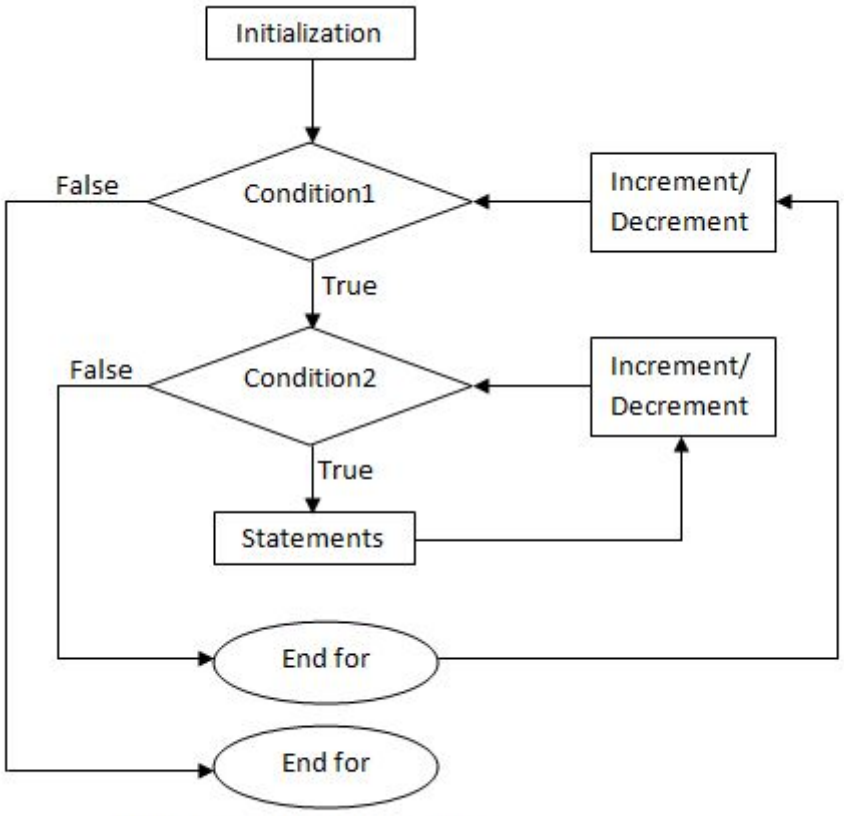


Fig: Flowchart for nested for loop

Example :

```
for (int i=0; i<=3; i++)
{
    for (int j=0; j<=i; j++)
    {
        printf(“*");
     }
     printf(“\n”);
}
```

Output :

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

- In the above example for(i=0;i<=3;i++)is outer for loop. ’i’ is the control variable for outer loop and ‘j’ is the control variable for inner loop.