Control Statements

Control statements define how the control is transfered to other parts of the program in C language. C supports four types of control statements.

1) if ... else
2) goto
3) switch
4) Loops
   a) while
   b) do ... while
   c) for

Compound statements or Block

A compound statement or a block is a group of statements enclosed within a pair of curly braces { }

Syntax

```
{
  statement 1;
  — — — —
  statement N;
}
```

for Example

```
{
  x=4;
  y=2
  ____
}
```

1) <u>If --- Else</u>

This is also known as bi-directional condition control statement. This is used to test condition and take one of the two possible actions. If the condition is true then a single or Block of statements are executed otherwise another single or block of statements are executed

<u>Syntax 1</u>

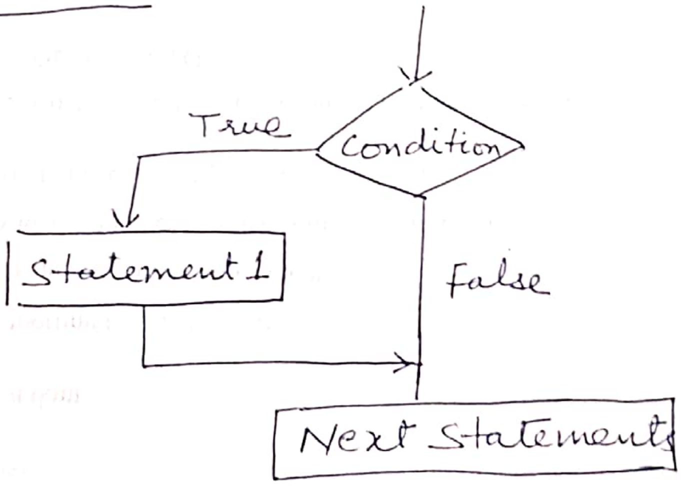
```
if (condition)
    statement;
```

<u>syntax 2</u>

```
if (condition)
    statement 1;
Else
    statement 2;
```

<u>Block diagram</u>

<u>Syntax 1</u>

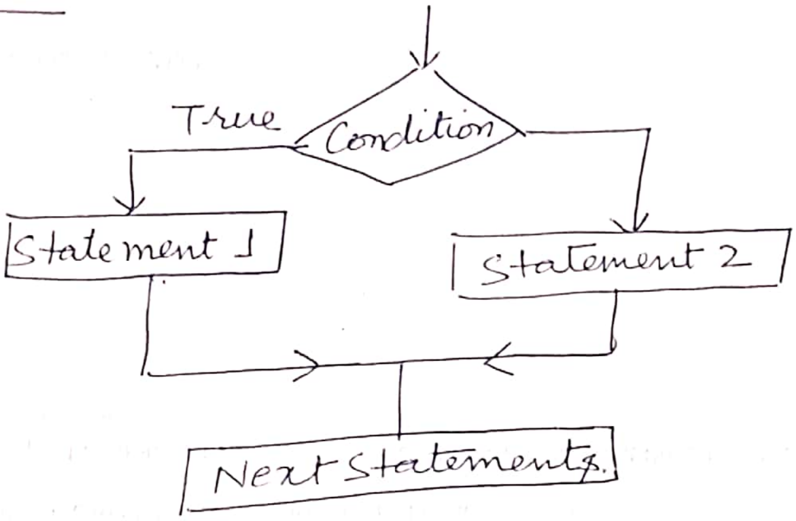


<u>Syntax 2</u>

```
if (condition)          — True/False
{ statements;     }
  ________        } True
  ________        }
}
Else { statements; } false
       ________
}
```

Block diagram

Syntax 2



Here if condition is True than statement 1 is Executed Else statement 2 is executed. and after that control is transfered to next statement immediately after the if ... Else statement.

Nesting of if .... Else

we can have another if ... else statement in if block or the else block that is called Nesting of if .... else statements

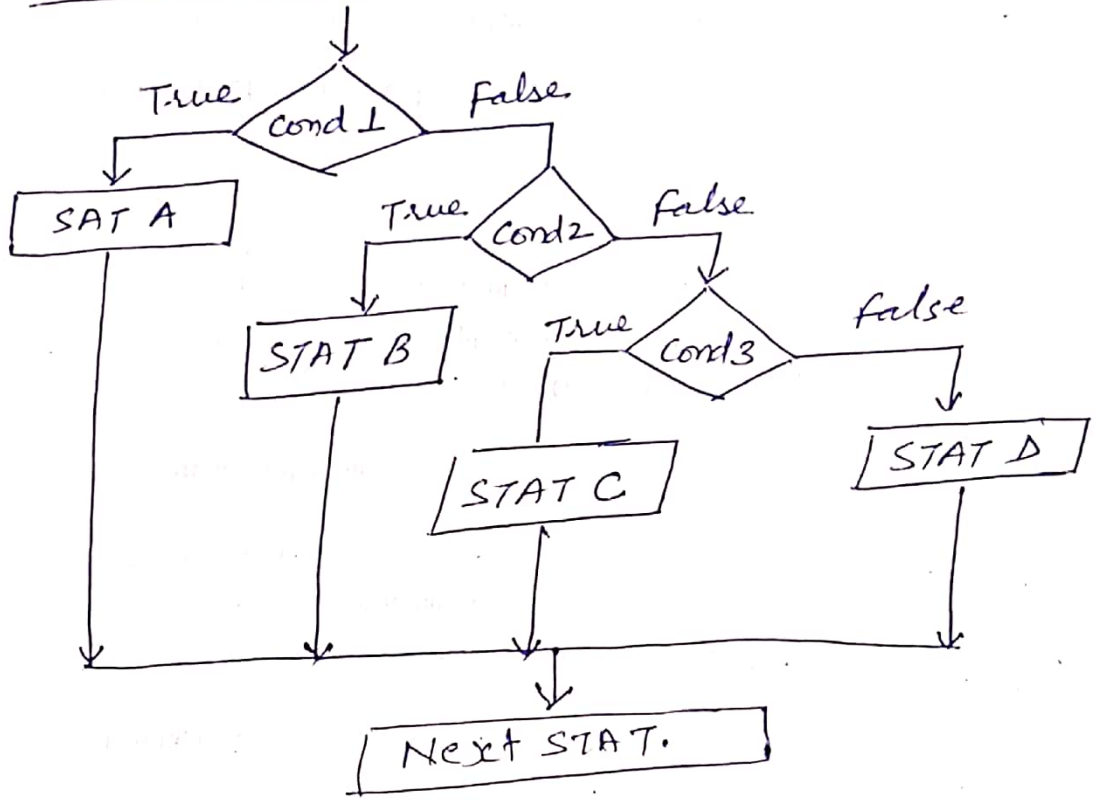
~~for~~ for Example

```
if (condition)
    { if (condition 2)
         { statement;
         }
      Else statement;
    }
Else
    { if (cond 3)
         {
           statement;
         }
      Else
      { statement;
      }
    }
```

Else if ladder

```
if (condition 1)
        statement A;
Else if (condition 2)
        statement B;
Else if (condition 3)
        statement C;
Else
        statement D;
```

Block diagram



Multiple if If we dont want to use else if ladder, the equivalent code for this problem would be.

```
if (cond)
    statement A;
if (con)
    statement B;
if (cond)
    statement C;
```

Difference

In if ---else ladder whenever a condition is found true other conditions will not be checked. while in Multipee if all conditions will be checked.

Loops

Loops are used when we want to execute a part of the program or a block of statements several times for example, suppose we want to print 'C' is best 10 times. One way to get the result is to write 10 printf statements which is not preferable. Other ways is using ~~to~~ loop. Using loop we can print the above statement by using only one printf.

There are Three loop statements in C.

(1) while
(2) do while
(3) for.

(1) while loops

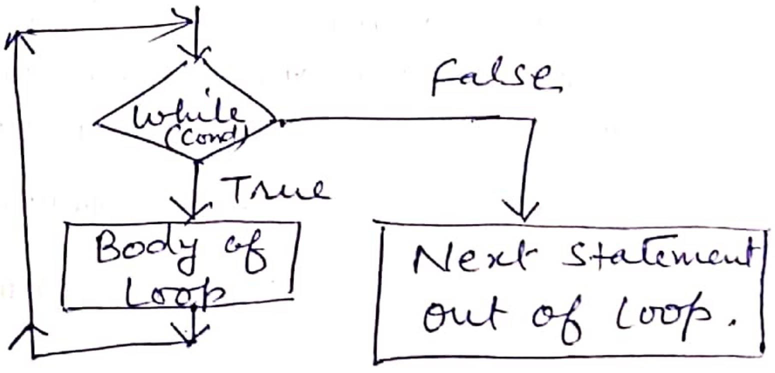
Syntax 1

```
while (condition)
    statement;
```

Syntax 2.

```
while (condition)
{
    statement 1;
    statement 2;
}.
```

Block Diagram / flow chart.



In while loop first condition is evaluated ; if it is true then statements in the body of loop are executed. After the execution again condition is checked. if it is found to be true then again the statements in the body of loop are executed. That means that these statements are executed continuosly till the condition is true and when the condition become false, the loop terminates and control comes out of the loop. Each execution of the loop body is known as iteration.

(2) do while

The do while statement is also used for looping. The body of this loop may contains a single statement or a block of statements

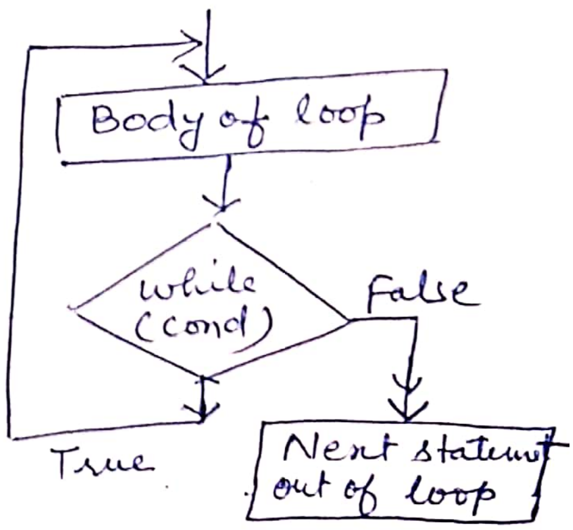
Syntax 1

```
do
  statement ;
while (condition);
```

Syntax 2

```
do
{
   statement 1;
   statement 2;
   ____
} while (condition);
```

Block diagram



Here, statements inside loop body are executed first and then the condition is evaluated. If the condition is true, then again the loop body is executed and process continues until the condition becomes false.

<u>Note</u> :- Here semicolon is placed after the Condition

(3) <u>For loop</u>.

The for loop statement is very useful in c programming. It has three expressions and semicolons are used for seperating these Expressions.

<u>Syntax 1</u>

```
for( Expression 1; Expression 2; expression3)
        statement;
```

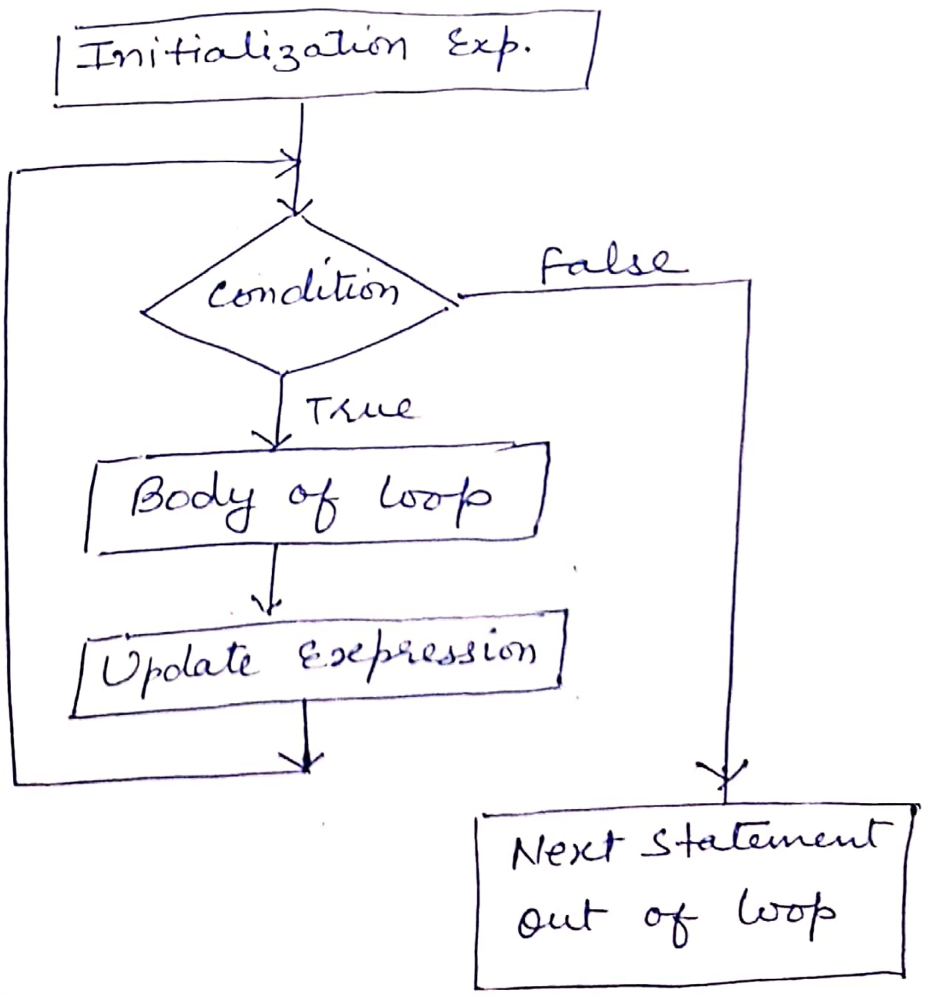
<u>Syntax 2.</u>

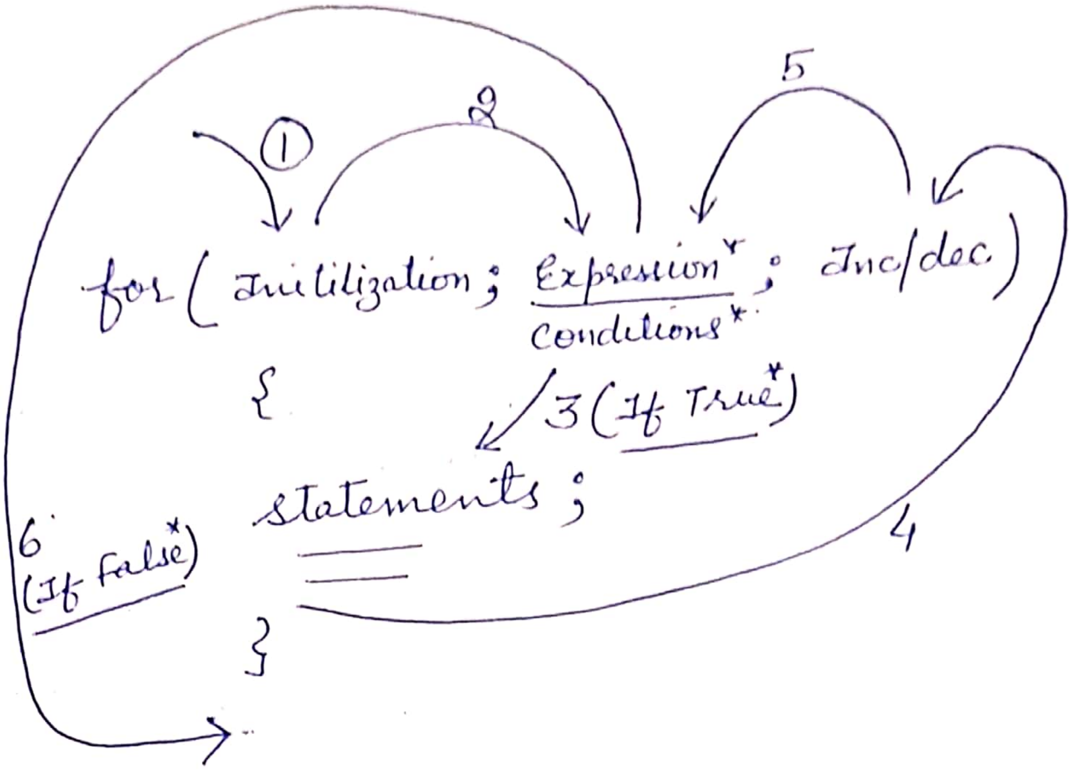
```
for ( Expression 1; Expression 2; Expression 3)
    {
      statement 1;
      statement 2;

    }.
```

The loop body can be a single statement or block of statements;

Expression 1 → Initilization Expression
Expression 2 → Test Expression
Expression 3 → update Expression (Inc/dec)

Let us know how this loop works. Firstly the initilization expression is executed and the loop variables are initilized, ~~and~~ the condition is checked, if the condition is true then the body of loop is executed. After that control transfers to Expression 3 (update Expression) and it modifies the loop variables and then again condition is checked. ~~[illegible]~~, and if it is true, the body of loop is executed. This process continues till condition remains True when condition become false the loop is terminated.





flow of for loop.

Nesting of Loops.

when a loop is inside a loop (body of loop) then it is known as nesting of loop. Any type of loop can be inserted in any other type of loop. for example a for loop may be nested inside another for loop or inside a while loop. similarly while and do while can be nested.

Infinite loops:-

The loops that go on executing infinitely and never terminate are called infinite loops.

for Example:-

```
while (1)
{
   ═
}
```

```
do
 {
   ===
 } while(1);
```

```
for ( ; ; )
 {
   ===
 }
```

```
while (n=2)
 {
   ===
 }
```

```
int i=1;
while (i<=5);
  printf("%d", i++);
```

$\Longleftarrow$

This loop will produce no output and will go on executing. As we have written(;) after the condition. so it will treat the loop as

```
   while (i<=5)
```

break statement :-

Break statement is used inside loop and switch statement. Sometimes it ~~bo~~ becomes necessary to come out of the loop even before the loop conditions become false. In such situation break statement is used to terminate the loop.

Syntax

```
   break;
```

Continue :-

The continue statement is used when we want to go to the next iteration of the loop after skipping some statements of the loop.

Syntax

```
   continue;
```

Switch :-

This is a multi-directional control statements sometimes there is a need in program to make choice among number of alternatives. For making this choice, we use ~~sti~~ 'switch statements.

Syntax

```
switch (Expression)
  {
    case condition 1:
            statements;

    case condition 2:
            statements;

    ~~Caste~~
    case condition N:
              statements;

    default:
            statements;

  }
```

Valid :-

switch(a), switch(a>b), switch(d+e-3),
switch(a>b && b>c)

Invalid :-

switch(float), switch (a+4.5)

Valid :-

case 4 :, case 'a' :, case 2+4 :, ~~Case 'a~~'

Case 'a' > 'b' :

Invalid : -

case "Second" :

Case 2.3 :

Case a ;

Case a > b :

Case a+2 :

Case 2, 3, 4 :

Case 2 : 3 : 5 :