CONDITIONAL STATEMENTS

Format Specifiers:

Format specifiers start with a percentage % operator and followed by a special character for identifying the type of data.

%d = Integer Format Specifier

%f = Float Format Specifier

%c = Character Format Specifier

%s = String Format Specifier

%u = Unsigned Integer Format Specifier

%ld = Long Int Format Specifier

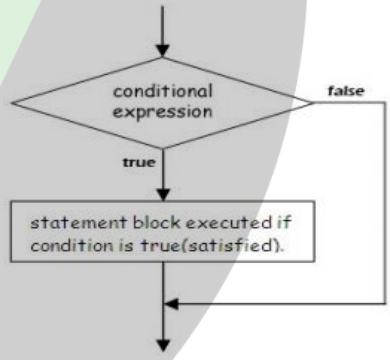
Conditional Statements:

C conditional statements allow you to make a decision, based upon the result of a condition. These statements are called Decision Making Statements or Conditional Statements.

1. If statement:

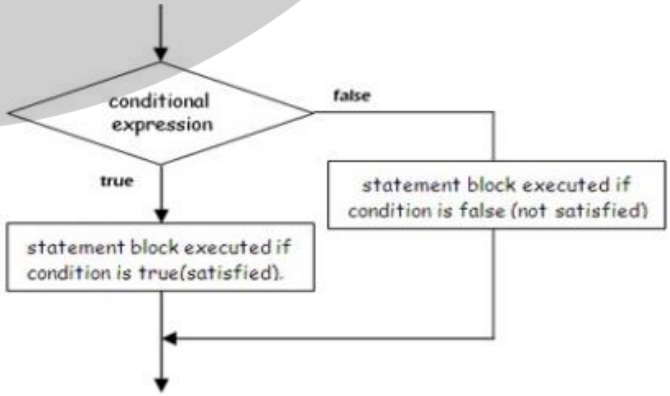
This is the most simple form of decision control statement. In this form, a set of statements are executed only if the condition given with **if** evaluates to true. Its general syntax and flow chart is as under: -

```
if(condition)
{
Statements;
}
```



2. If else:

Use the if statement to specify a block of C code to be executed if a condition is true.Use the else if statement to specify a new condition if the first condition is false. Use the else statement to specify a block of code to be executed if the condition is false.



Syntax

```
if (condition1) {

  // block of code to be executed if condition1 is true

} else if (condition2) {

  // block of code to be executed if the condition1 is false and condition2 is true

} else {

  // block of code to be executed if the condition1 is false and condition2 is false

}
```

Example:-

```
int time = 20;

if (time < 18) {

  printf("Good day");

}

else {

   printf("Good evening");

}
```

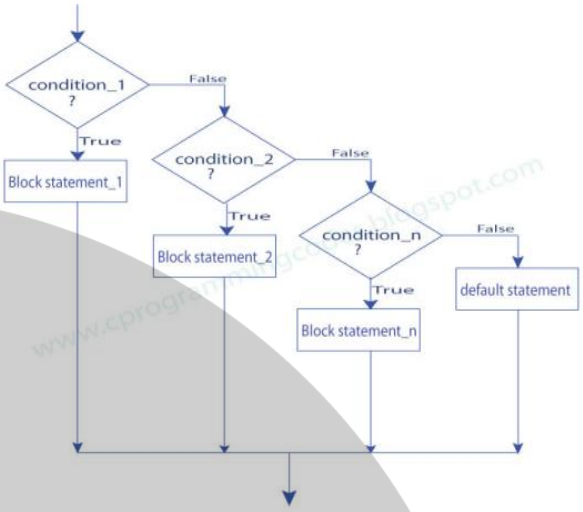
3.Nested if-else:-

If we have **if-else** statement within either the body of an **if** statement or the body of **else** statement or in the body of both **if** and **else**, then this is known as nesting of if else statement. The general form of nested if-else statement is as follows:-

```
if(condition1)
    {
      if(condition2)
          statements;
      else
          statements;
}
else
{
     if(condition3)
         Statements;
     else
         statements;
}
```

4.Else if ladder:-

This is a type of nesting in which there is an if-else statement in every else part except the last else part. This type of nesting is called else if ladder.



The general syntax and flow chart of else if ladder is as follows:-

```
If(condition1)
        statement A;
else if(condition2)
        statement B;
else if(condition3)
        statement C;
else
        statement D;
```

5. Switch:

Use the switch statement to select one of many code blocks to be executed.

The break Keyword:

- When C reaches a break keyword, it breaks out of the switch block.
- This will stop the execution of more code and case testing inside the block.
- When a match is found, and the job is done, it's time for a break.

The default Keyword:

The default keyword specifies some code to run if there is no case match:

Example:-

```
int day = 4;
switch (day) {
  case 6:
        printf("Today is Saturday");
   break;
  case 7:
        printf("Today is Sunday");
   break;
  default:
printf("Looking forward to the Weekend");
}
```

Ternary Operator / Conditional Operator

The conditional operator takes less space and helps to write the if-else statements in the shortest way possible.

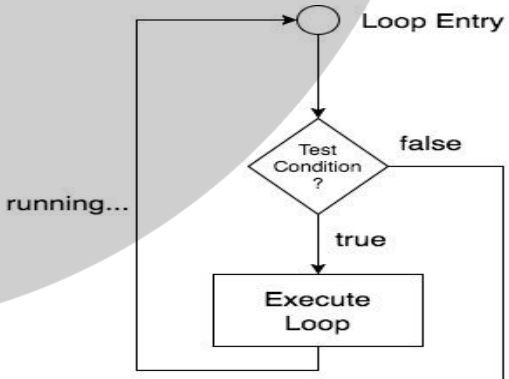
Syntax:

(Condition) ? Statement 1 : Statement 2 ;

Example:

```c
// C program to find largest among two
// numbers using ternary operator

#include <stdio.h>

int main()
{
    int m = 5, n = 4;

    (m > n) ? printf("m is greater than n that is %d > %d",
                     m, n)
            : printf("n is greater than m that is %d > %d",
                     n, m);

    return 0;
}
```


Loops in C



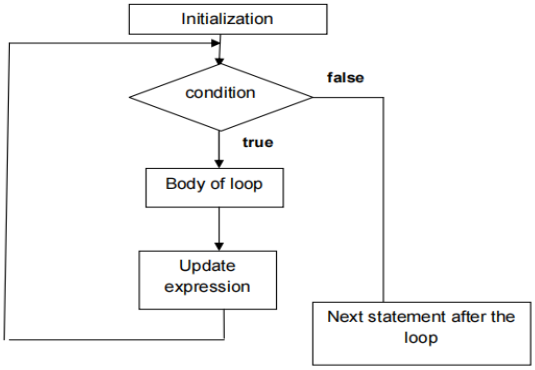
In any programming language including C, loops are used to execute a set of statements repeatedly until a particular condition is satisfied.

The diagram depicts a loop execution.
There are three types of loops in C:

1. For loop

Syntax:

for(initialization; condition; increment/decrement)
{
statement-block;
}



In for loop we have exactly two semicolons, one after initialization and second after the condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. But it can have only one **condition**.

The For loop is executed as follows:

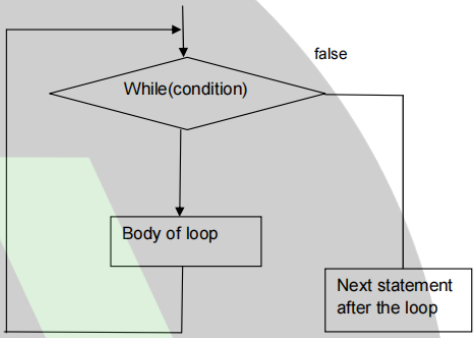
1. It first evaluates the initialization code.
2. Then it checks the condition expression.
3. If it is **true**, it executes the for-loop body.
4. Then it evaluates the increment/decrement condition and again follows from step 2.

When the condition expression becomes **false**, it exits the loop.

2. While loop:

while loop can be addressed as an **entry control** loop. It is completed in 3 steps.

- Variable initialization.(e.g int x = 0;)
- condition(e.g while(x <= 10))
- Variable increment or decrement (x++ or x-- or x = x + 2)



Syntax :

```
variable initialization;
while(condition)
{
   statements;
   variable increment or decrement;
}
```

Example:

```
#include<stdio.h>
void main( )
{
   int x;
   x = 1;
   while(x<=10)
     {
        printf("%d\t", x);
        // below statement means, do x = x+1, increment x by 1 //
        x++;
      }
}
```

```
While(Alive)
{
eat();
//sleep();
code();
if Dead(Break);
}
```

Output:

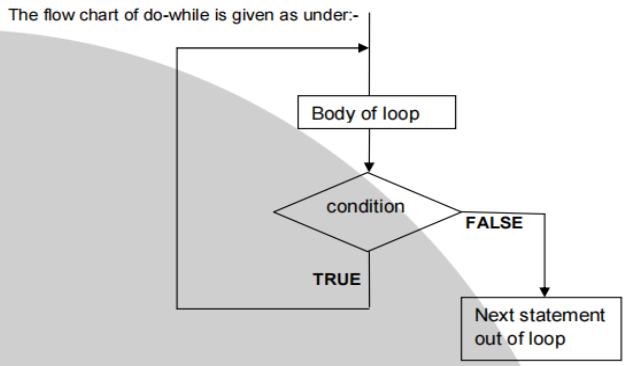
1 2 3 4 5 6 7 8 9 10

3. do while loop

In some situations it is necessary to execute the body of the loop before testing the condition. Such situations can be handled with the help of a do-while loop. do statement evaluates the body of the loop first and at the end, the condition is checked using while statement. It means that the body of the loop will be executed at least once, even though the starting condition inside while is initialized to be **false**.

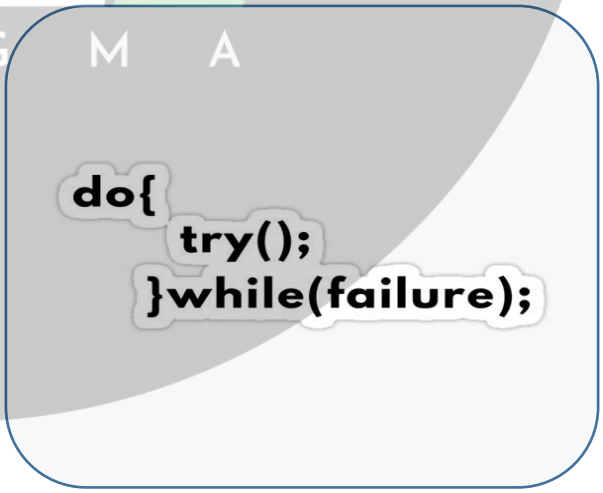
Syntax:

```
do
{
   Statements;
   .....
}
while(condition);
```



Example:

```
#include<stdio.h>
void main()
{
   int a, i;
   a = 5;
   i = 1;
   do
   {
      printf("%d\t", a*i);
      i++;
   }
   while(i <= 10);
}
```



OutPut:

5 10 15 20 25 30 35 40 45 50

Difference between while and do-while loop

While	do-while
While loop is pre test loop.	do-while is post test loop
The statements of while loop may not be executed even once	The statements of do-while loop are executed atleast once
There is no semi colon given after while(condition)	There is semi colon given after while(condition);

The syntax of while loop is
While(condition)
{
Statements
}
	The syntax of do-while loop is as under:-
	Do
	{
	Statements;
	}while(condition);

Jump statements

Jump statements are used to transfer the control from one part of the program to another part.

The various jump statements in C are as under:-

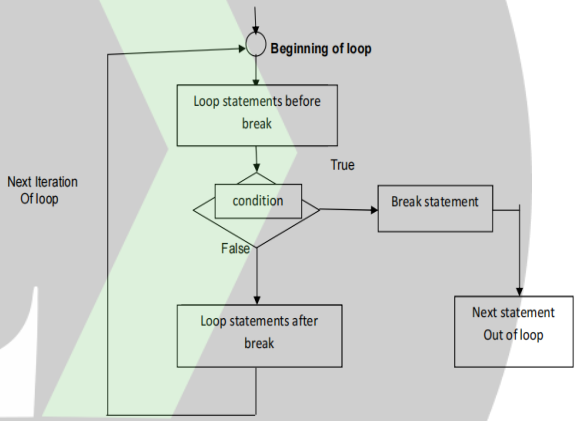
1. break statement
2. continue statement

break statement :-

break statement is used inside the loops or switch statement. This statement causes an immediate exit from the loop or the switch case block in which it appears.

It can be written as

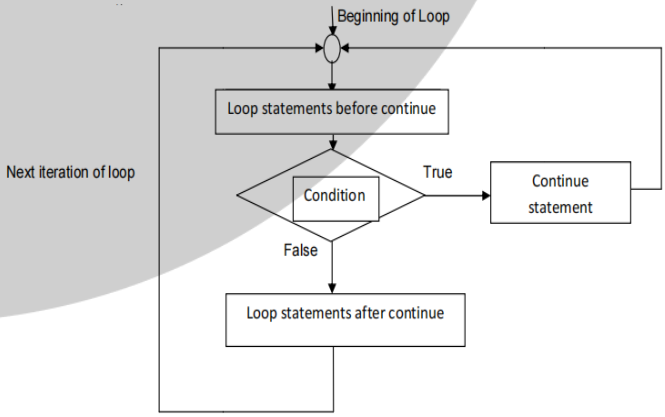
break;



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

The continue statement is written as under :-

Continue;



The **difference between break and continue** is that when a break statement is encountered the loop terminates and the control is transferred to the next statement following the loop, but when a continue statement is encountered the loop is not terminated and the control is transferred to the beginning of the loop.