

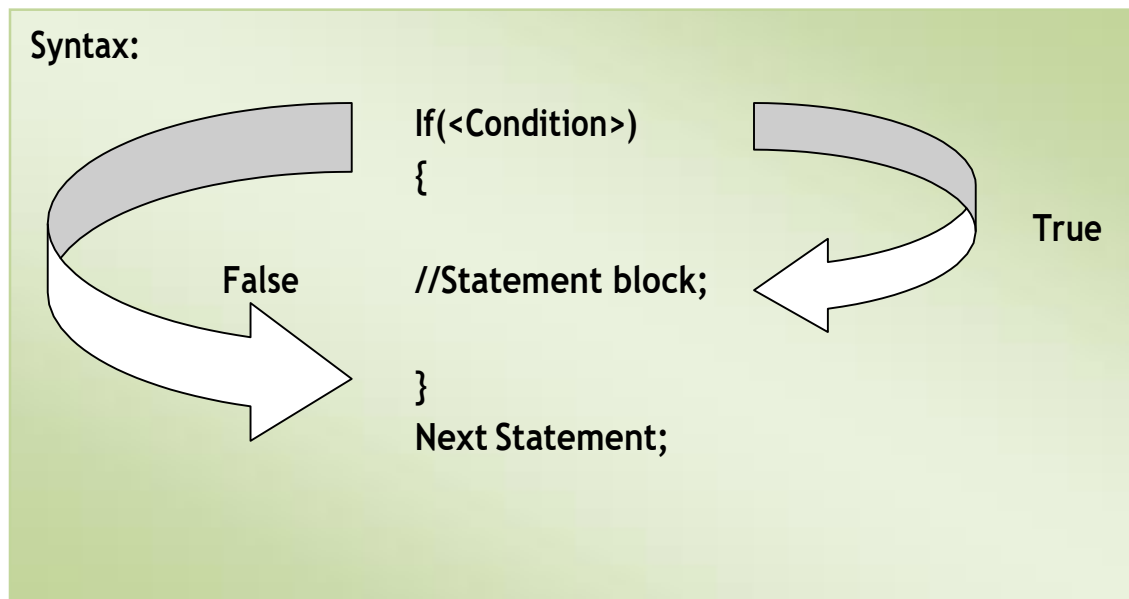
**Chapter No: 3****Control Structure in C**

CHAPTER NO: 3	Control Structure in C
1	Explain simple if statement with example.
2	Explain if-else statement with example.
3	Explain nested if with example.
4	Explain ladder if-else with example.
5	Explain switch statement with suitable example.
6	Explain goto statement.
7	Explain while loop with example.
8	Difference between entry control and exit control (while and do-while).
9	Explain do while loop with example.
10	Explain for loop with example.
11	What is nested loop?
12	Explain break and continue statement with example.

**Q-1 Explain simple If Statement with example.**

**Answer:**

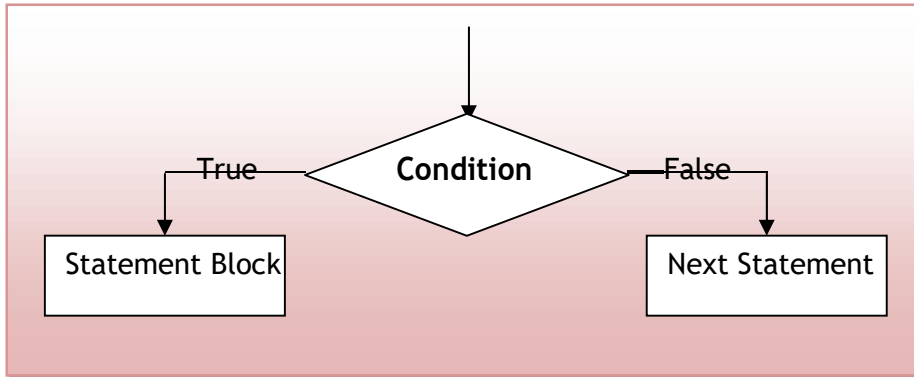
When we have situation to check some conditions and based on that condition some statements will be executed then we can use if statement.



Here, first condition is checked. If condition becomes true then statement inside if block will be executed.

If condition becomes false then statement outside if block will be executed.

We can also put more than one condition to check through if statement.



If statement is executed in following order:

- 1) First condition is checked.
- 2) If condition is true then statement block is executed first and then next statement will be executed
- 3) If condition is false then next statement outside if block will be executed.

**Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int var;
    printf("\n enter the value of the variable");
    scanf("%d",&var);
    if(var<=100)
        printf("\n the value entered is less than 100");
    printf("program is over");
}
```

**Note:**

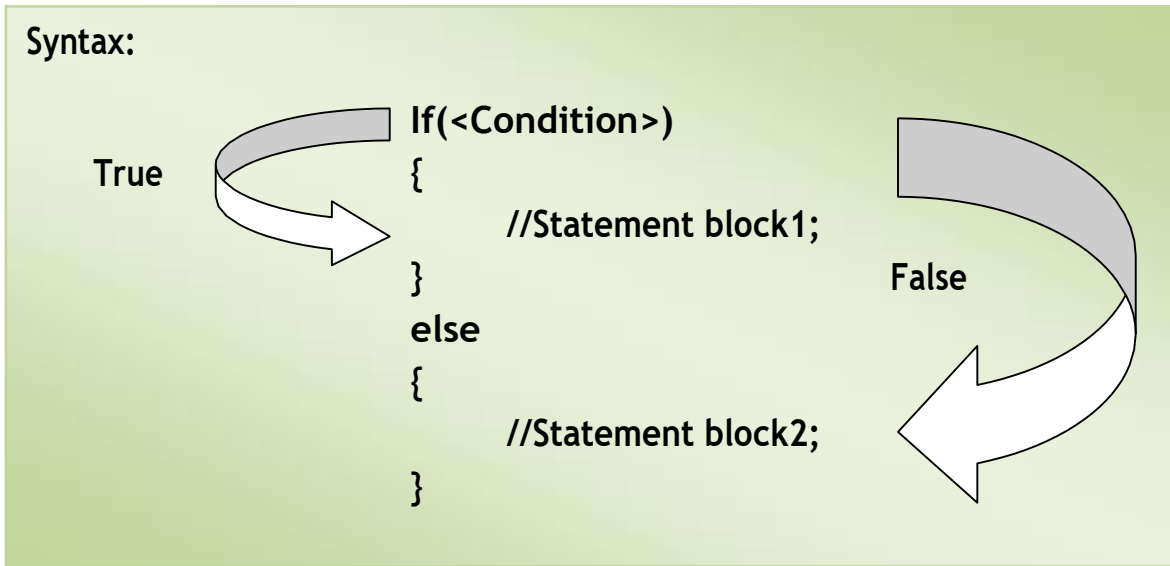
Every statement in C ends with semicolon (;) but if statement does not end with semicolon.

When we want to execute multiple statements, then {} is compulsory, otherwise not compulsory.

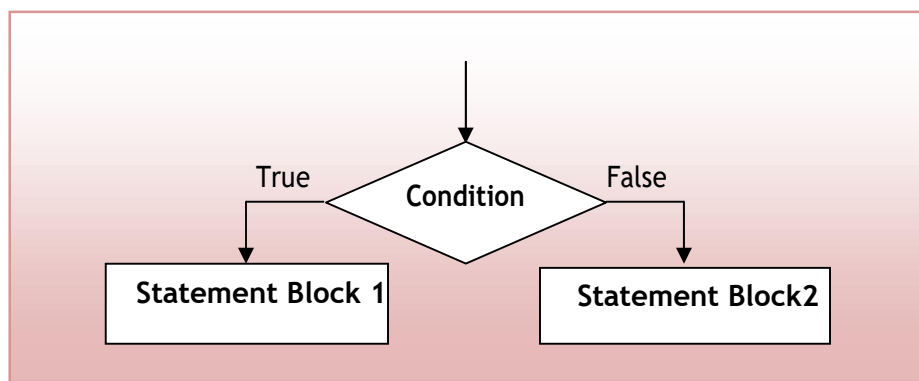
**Q-2. Explain if else statement with example.**

**Answer:**

When we have situation to execute some statement if condition becomes true and other statement if condition becomes false then we can use if else statement.



Here first condition is checked first, if it becomes true then statement block1 will be executed and if condition becomes false then statement block2 will be executed.



If else statement is executed in following order:

- 1) First condition is checked.
- 2) If condition is true then statement block1 is executed.
- 3) If condition is false then statement block2 inside else block will be executed.

**Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int var;
    printf("\n enter the value of the variable");
    scanf("%d",&var);
    if(var<=100)
        printf("\n the value entered is less than 100");
    else
        printf("\n the value entered is greater than 100");
    printf("program is over");
}
```

Q-3. Explain nested if statement with example.

**Answer:**

If statement inside if statement is known as nested if statement.

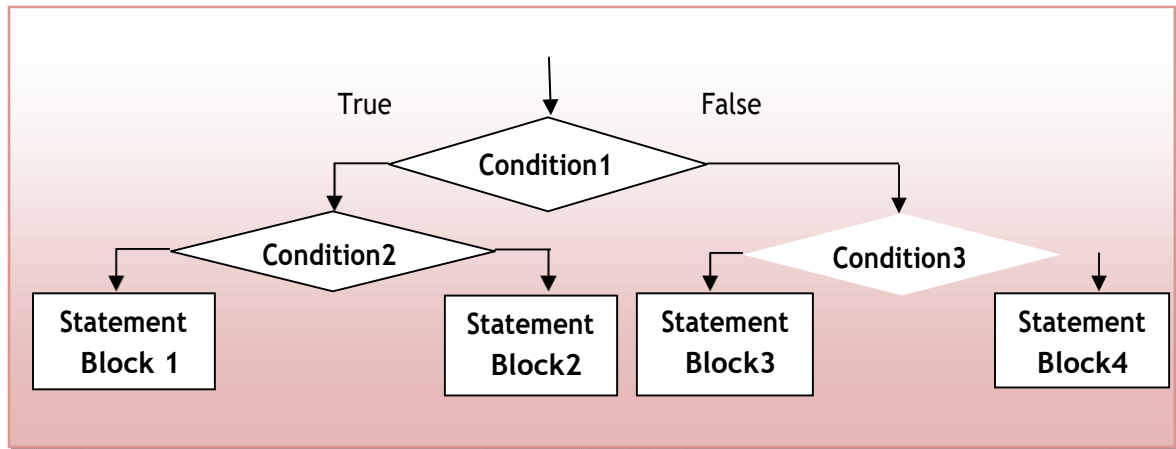
**Syntax:**

<b>If(&lt;Condition1&gt;)</b>	<b>else</b>
{	{
<b>If(&lt;condition2&gt;)</b>	<b>if(&lt;condition3&gt;)</b>
{	{
//statement block1;	//statement block3;
}	}
<b>else</b>	<b>else</b>
{	{
//statement block2;	//statement block4;
}	}

Nested if statement is executed in following order:

- 1) First condition1 is checked.
- 2) If condition1 is true then condition2 is checked. If condition2 is true then statement block1 is executed and if condition2 is false then statement block2 is executed.

- 3) If condition2 is false then condition3 is checked. If condition3 is true then statement block3 is executed and if condition3 is false then statement block4 is executed.

**Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n1,n2,n3;
    printf("\n enter the value of the n1,n2 and n3");
    scanf("%d %d %d",&n1,&n2,&n3);
    if(n1>n2)
    {
        if(n1>n3)
            printf("\n n1 is max");
        else
            printf("\n n3 is max");
    }
    else
    {
        if(n2>n3)
            printf("\n n2 is max");
        else
            printf("\n n3 is max");
    }
    getch();
}
```

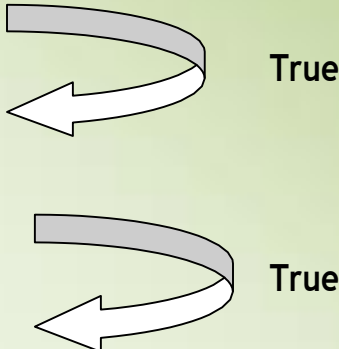
**Q-4 Explain if-else-if ladder statement with example.**

**Answer:**

If we are having different - different test conditions with different - different statements, then for these kind of programming we need else if ladder.

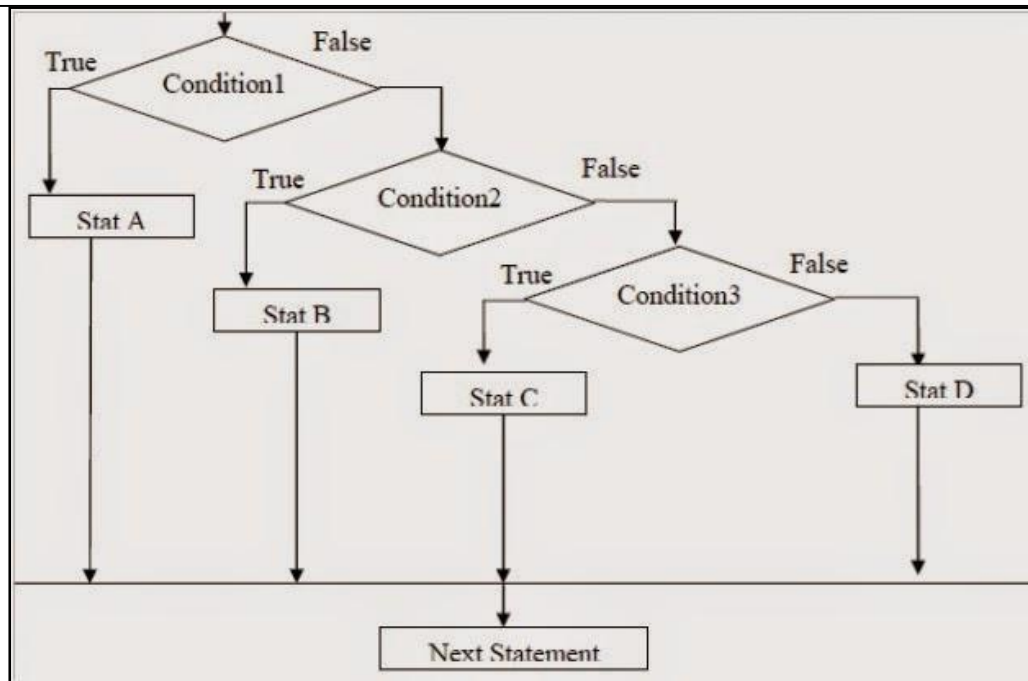
**Syntax:**

```
If(<condition1>)
{
    //statement block1;
}
else if(<condition2>)
{
    //statement block2;
}
.
.
.
else
{
    //statement blockn;
}
```



If else if ladder statement is executed in following order:

- 1) First condition1 is checked. If it is true then statement block 1 is executed.
- 2) If condition 1 is false then condition 2 is checked. If it becomes true then statement block 2 is executed. So likewise this all condition is checked and based on the result statement block will be executed.
- 3) If all condition becomes false then statement block n (inside else block) will be executed.

**Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int no;
    printf("\n enter value of no variable");
    scanf("%d",&no);
    if(no>0)
        printf("\n positive");
    else if(no<0)
        printf("\n negative");
    else if(no==0)
        printf("\n zero");
    else
        printf("\n invalid value");
    getch();
}
```

**Q-5. Explain switch case with example.**

**Answer:**

When one option should be selected from many options then we can use if-else-if statement but becomes complex sometimes, so instead of it we can use switch statement.

The selection in switch case is based upon the current value of the variable that is included within the switch statement.

The switch statement supports only character and integer variables. It does not support float or any other type of variables.

The switch statement checks the value of variable against the list of case values and when a match is found, a block of statements associated with that case is executed.

**Syntax:**

```
switch(<variable>)  
{  
    case <value 1>://statement1;  
        break;  
    case <value 2>://statement2;  
        break;  
    ....  
    default:    //default statement block;  
}
```

In the above syntax, switch, case and break are the keywords of C language.

With switch statement, variable is provided and its possible values are given with the case statement. The case ends with ' : '.

The value with the case statement, provided is constant.

When the switch is executed, the value of the variable is compared with value1, value2....and soon.

When the value is found for the variable, the related block is executed and at the end of block the break statement will break the block and causes to exit from the case statement and next statement is executed. The default is optional.

When no case is executed then default case will be executed.

**Example:**

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
    int no;  
    printf("enter value of variable");
```



```
scanf("%d",&no);
switch(no)
{
    case 1: printf("one");
            break;
    case 2: printf("two");
            break;
    case 3: printf("three");
            break;
    default: printf("greater than 3 or invalid");
            break;
}
getch();
}
```

**Q-6. Explain goto statement with example.**

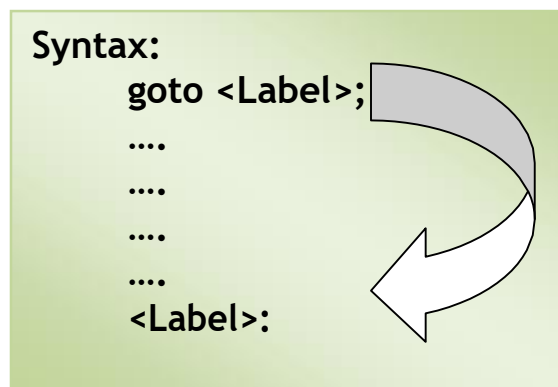
**Answer:**

Normally C language program is executed line by line. So change the sequence of execution or to jump from one statement to another statement goto statement is used.

There are two type of goto statement.

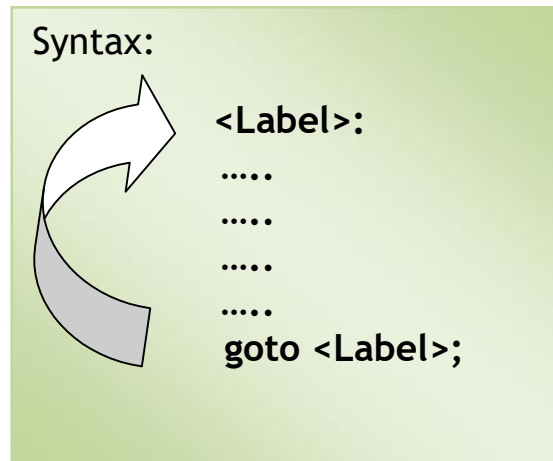
- 1) Forward goto statement
- 2) Backward goto statement

**1) Forward goto statement:**



In this label must be same for both <Label>.

**2) Backward goto statement:**

**Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a,i;
    i=0
    clrscr();
    a=1;
    If(i>5)
        goto b;
    else
    {
        printf(“%d”,a);
        a++;
    }
    b:
        getch();
}
```

**Q-7 Explain while loop with example.**

**Answer:**

When we want to execute same statement or group of statements multiple times then we can use loop structure.

while loop is known as entry control loop.

**Syntax:**

```
while(<condition>)  
{  
    //statement(s);  
}
```

**While loop is work as follow:**

First condition is checked and if it is true then statement inside while loop will be executed.

Now again condition is checked and statement inside while will be executed till the condition becomes true.

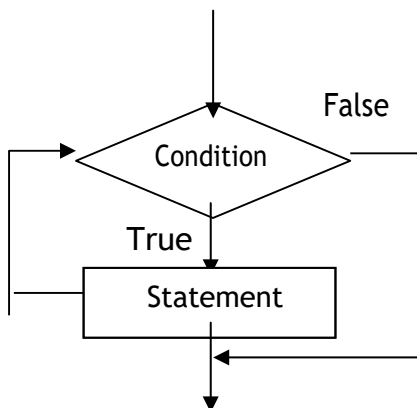
If condition becomes false at very first time then statement inside loop will never executed.

**Example:**

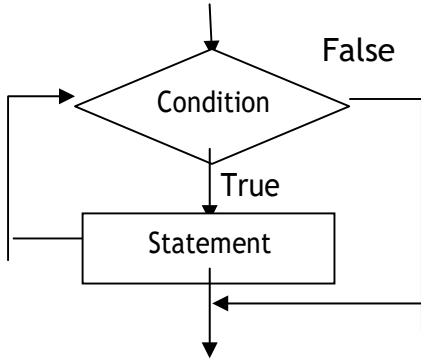
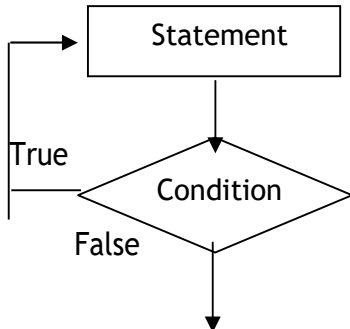
```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
    int i;  
    i=1;  
    while(i<5)  
    {  
        printf("%d",i);  
        i++;  
    }  
    getch();  
}
```

**Note:**

In this example value of i variable is 1. Now condition is checked which becomes true so statement inside loop will be executed. Loop will be executed till the value of i remain less than 5.



**Q-8 Difference between while and do while loop.****Answer:**

While Loop	Do while Loop
It is also known as entry control loop.	It is also known as exit control loop.
In this first condition is checked and then statement will be executed.	In this first statement will be executed and then condition is checked.
User can't give guaranteed to execute statement at least once even condition is false at very first time.	User can give the guaranteed to execute statement at least once even condition is false at very first time.
<pre>while(&lt;condition&gt;) {     //statement; }</pre>	<pre>do {     //statement; }while(&lt;condition&gt;);</pre>
In syntax there is no semicolon at end of condition.	In syntax there is semicolon after condition.
	

**Q-9 Explain do while loop with example****Answer:**

Do while loop is known as exit control loop.

In this loop first statement inside loop is executed and then condition is checked; now if condition becomes true then again statement is executed. So, in do while loop we can guarantee that statement inside loop must be executed at least once even condition is false.

**Syntax:**

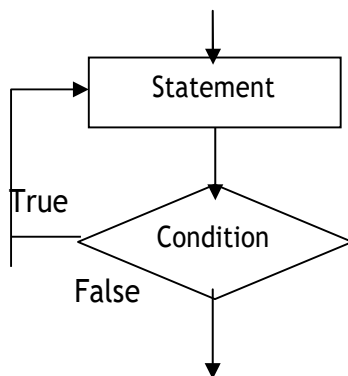
```
do
{
    //statement(s);
}while(<condition>);
```

**do while loop is work as follow:**

First condition is checked and if it is true then statement inside do while loop will be executed. Now again condition is checked and statement inside do while will be executed till the condition becomes true.

**Example:**

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



**Q-10. Explain for loop with example.**

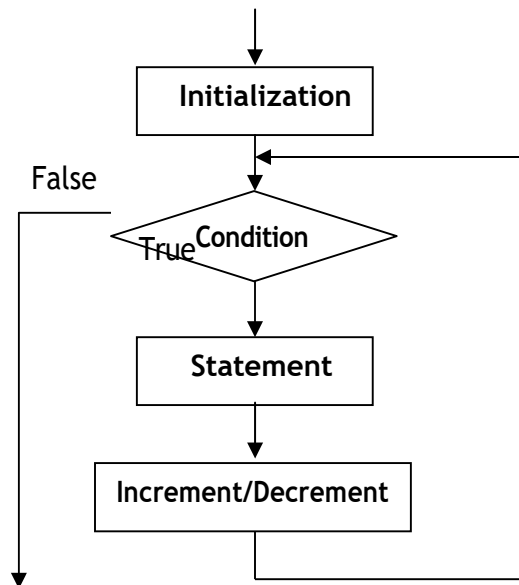
**Answer:**

When you know in advance, how many times you want to execute loop then for loop can be used. It is entry control loop.

**Syntax:**

```
for(<initialization>;<condition>;<increment/decrement>)
{
    //statement;
}
```

In for loop first initialization is executed only once.  
It initializes the control variables which control the overall loop.  
Then condition is checked, if it is true then statement inside for loop will be executed.  
Then increment or decrement statement will be executed.  
The statement inside loop will be executed till the condition remains true.



Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a;
    clrscr();
    for(a=1;a<5;a++)
    {
        printf("%d",a);
    }
    getch();
}
```

Q-11. Explain nested for loop with example.

Answer:

For loop within another for loop is known as nested for loop.

Loop inside loop is known as inner loop which is executed first, once all inner loops are executed outer loop is start to execute.

**Syntax:**

```
for(<initialization>;<condition>;<increment/decrement>)  
{  
    for(<initialization>;<condition>;<increment/decrement>)  
    {  
        ....  
        ....  
    }  
}
```

Loop can be nested at any number of levels.

If inner loop is executed 3 times and outer loop is executed 3 times than total 9 times statement inside loop are executed.

**Example:**

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
    int i,j,a=1;  
    clrscr();  
    for(i=1;i<=3;i++)  
    {  
        for(j=1;j<=3;j++)  
        {  
            printf("%d",a);  
        }  
    }  
    getch();  
}
```

**Note:**

Here, value of a is displayed 9 times because when the value of variable i is 1 then inner loop is executed 3 times, when value of variable i is 2 then again inner loop is executed 3 times and so on.

**Q-12 Explain break and continue statement with example.**

**Answer:**

**Break Statement:**

Break statement is used to break any loop from any stage.

It is also used inside switch case statement.

If you have an infinite loop then to break the loop you can use break statement and it will be executed when certain condition becomes true.

**Syntax:**

```
<Any loop>
{
    Some condition
    break;
}
```

**Continue statement:**

It will be used same as break statement but when break statement is used you will be out of loop and when you use continue statement then current iteration is skipped but rest of loop will be continued.

**Syntax:**

```
<Any loop>
{
    Some condition
    continue;
}
```

**Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n,i=0;
    printf("give last even number");
    scanf("%d",&n);
    printf("even number are:");
    while(1)
    {
        if(i%2==1)
        {
            i++;
        }
    }
}
```



```
        continue;
    }
    if(i<n)
        break;
    printf("%d",i);
    i++;
}
getch();
}
```

### Course Outcomes:

- 1: Illustrate the flowchart and design an algorithm for a given problem and to develop C programs using operators
- 2: Develop conditional and iterative statements to write C programs