MODULE - 2

Branching and Looping.

Two way Selection:

The If Statement is a decesion making Statement and it is Used to Control the flow of executable Statements.

If Statement:-

General form of if Statement.

```
if (test expression)
{
    Statement-block;

}
Statement-x;
```

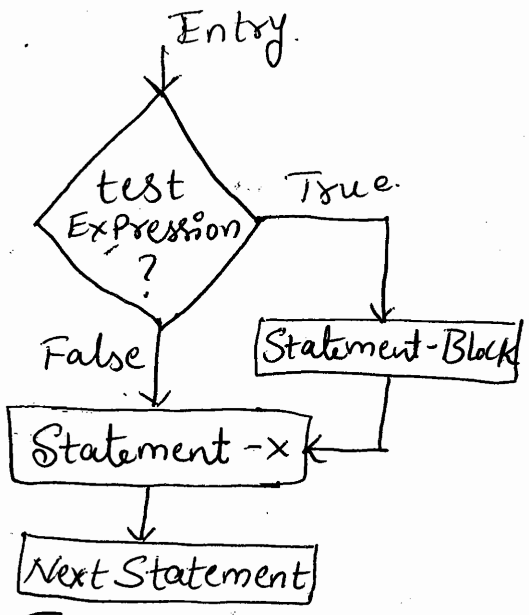


Fig:- Flow Chart.

The "Statment-block" may be a Single Statement or a Set of Statements. If the test expression is true the Statement-Block will be executed. Otherwise will Jump to the Statement-X

Example:-

```
        ------
        if ( Category == MEDICAL)
        {
            marks = marks + bonusmarks;
        }
        Printf ("%f", marks);
        ----
```

Program:
```
#include<stdio.h>
Void main()
{
    int  x;
    Printf(" Enter value of x ");
    Scanf(" %d", &x);

    if (x < 0)
    Printf(" The value entired is negative");
    getch();
}
```

If-else Statement:

The if-else is an extension of if statement

General form of if-else is

```
if ( test expression)
 {
   True-Block Statement(s)
 }
else
 {
   false-Block Statements(s)
 }
Statement - x
```

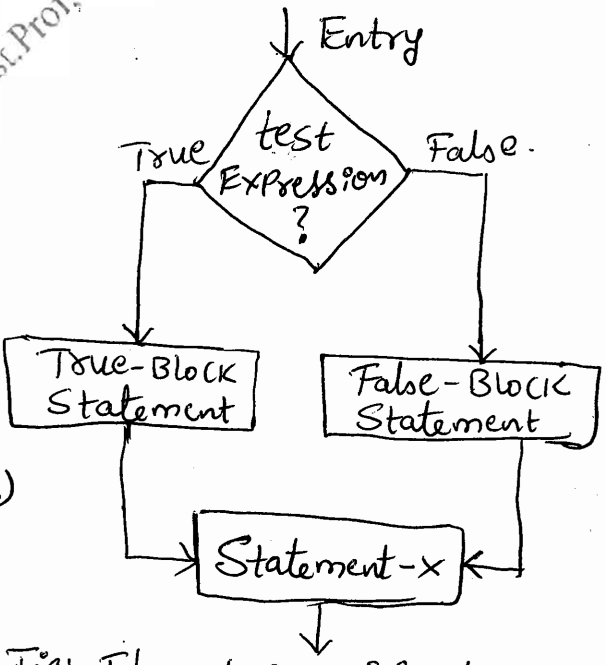


Fig: Flow Chart of if-else.

Example:
```
- - - -
- - - -
if (Code == 1)
   boy=boy+1;
else
   girl =girl+1;
- - - -
```

Program:
```
#include<stdio.h>
Void main()
{
   int a, b;
   Printf(" Enter values of a, b\n");
  Scanf(" %d %d", &a, &b);
   if(a > b)
    Printf(" The value of a %d is greater", a);
    else
    Printf(" The value of b %d is greater", b);
  getch();
}
```

Nested of if-else Statements:

If the condition-1 is false, the Statement-3 will be executed, otherwise it continues to Perform the Second test.

General form of Nested if else:

```
if (test Condition-1)
  { if (test Condition-2)
     {   Statement-1;
     }
    else
     { Statement-2
     }
  }else
  { Statement 3;
```

If the Condition-2 is true then the Statement-1 will be evaluated. Otherwise Statement-2 will be evaluated then the Control is transferred to Statement-x.

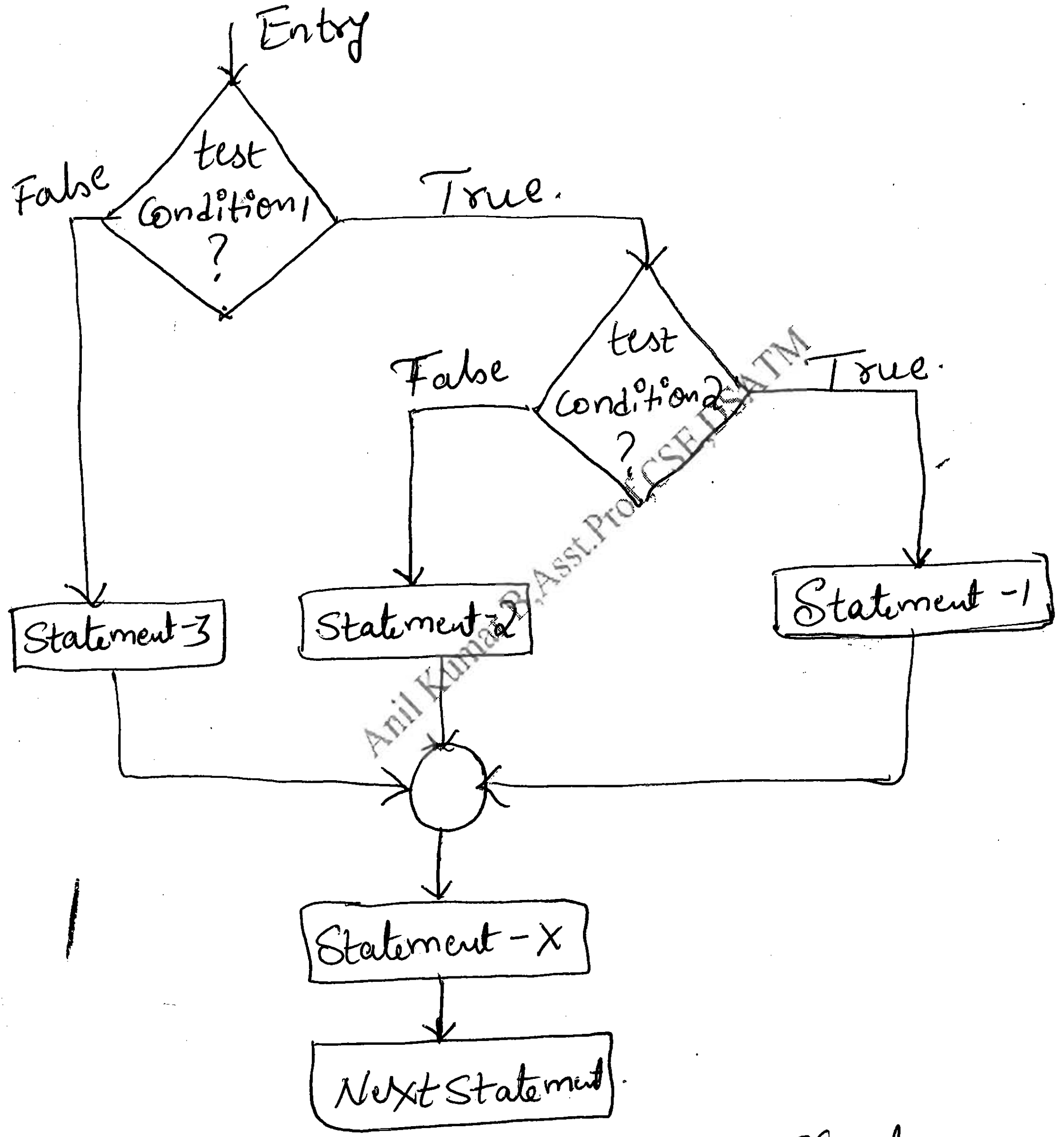


Fig: FlowChart of nested if else.

when nesting, Care should be exercised to match every if with an else. The example for ~~an~~ nested if-else Statement is illustrated as follows.

Example! ------

```
if (Gender is female)
{
    if ( balance >5000)
        bonus= 0.05 * balance;
    else
        bonus= 0.02 *balance;
}
else
    { bonus= 0.02 * balance;
    }
balance=balance +bonus;
-----
```

Program:-

```
#include<Stdio.h>
Void main()
{
  float A,B,C;
  Printf(" Enter three Values \n");
  Scanf(" %f %f %f", &A, &B, &C);
  Printf (" Largest Value is \n");
  if(A >B)
  {
    if(A >C)
      Printf(" %f \n", A);
    else
      Printf(" %f \n", C);
else
{
    if (C > B)
        Printf("%f \n", C);
    else
        Printf("%f", B);
    }
}
```

Output:-

```
Enter three values.
2856   1289   3456.

Largest Value is 3456.000000.
```

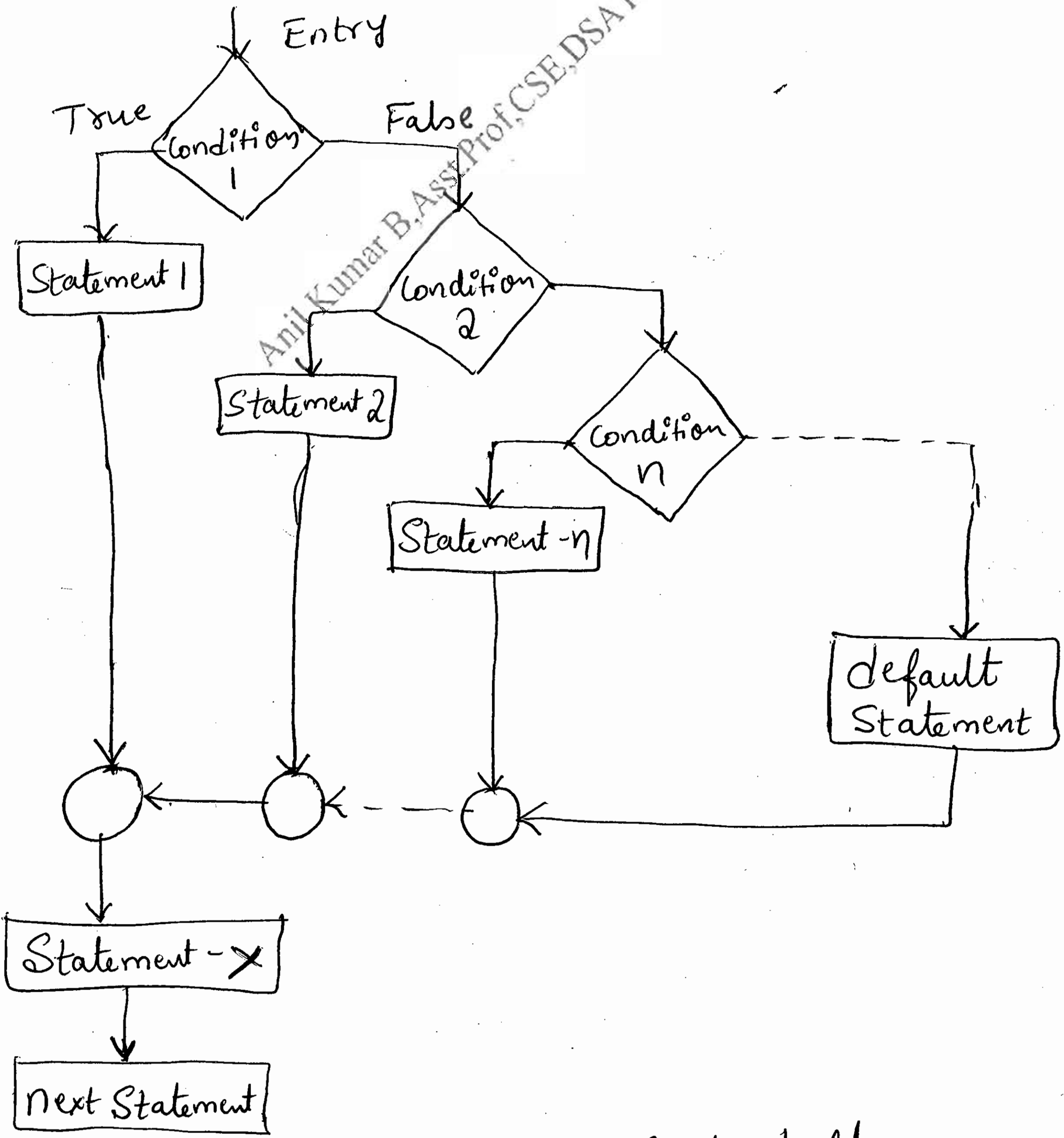
Cascaded if-else (Else if Ladder):-

A multipath decesion is a chain of ifs in which the statement is associated with each else is an if. General form is

```
if (condition 1)
    Statement-1;
else if (condition-2)
    Statement-2;
|
else if (condition-n)
    Statement-n.
|
else
    default-Statement
Statement-x
```

Example: Let ~~[illegible]~~ an grading the Students in academic record.

```
if ( marks > 75)
   grade = "FCD";
else if ( marks >60);
   grade = "First Class";
else if (mark > 50);
   grade= "Second";
else
grade = "Fail";
```



The chart of if-else ladder:

Program :-

```
#include <stdio.h>
Void main ( )
{
    int units , Cnum;
    float Charges;
   Printf(" Enter Custno &units Consumed\n");
   Scanf(" %d %d", &Cnum, &units);
   if ( units <= 200)
       Charges = 0.5 * units;
   else if (units <= 400)
       Charges = 100 + 0.65*(units -200);
   else if (units <= 600)
       Charges = 230 + 0.8 *(units -400);
       else
       Charges = 390 + (units - 600);
  Printf(" Custno = %d  Charges = %0.2f ",
                  Cnum, Charges);
}
```

Output:

```
Enter Custno & units Consumed 101 150
Custno = 101   Charges = 75.00

Enter custno & units Consumed 501 625
Custno = 501   Charges = 415.00 .
```

Switch ~~[illegible]~~ Statement :-

The C Program has a built-in multiway decesion Statement Known as Switch. The Switch Statement tests the Value of a given Variable (or expression) against a list of Case Values and When a match is found, a block of Statements associated with that Case is executed.

General form:

```
        Switch (expression)
        {
          Case Value-1:
                   block-1
                   break;
          Case Value-2:
                   block-2
                   break;
          ----
          ----
          default :
                 default-block;
                   break;
        }
        Statement-x;
```

Where,
-> The expression is an integer or character.
-> Value-1, Value 2 .... are Constant Known

Each of the Valuesie Case Values Should be unique within a Switch Statement.

-> Block-1 , Block-2 .... are statement lists may Contain Zero or more Statements.

-> Case labels end with Colon :

-> break Statement at end of each block indicates that end of Particular Case.

-> default is optional Case, executed if Value of expression does not match.

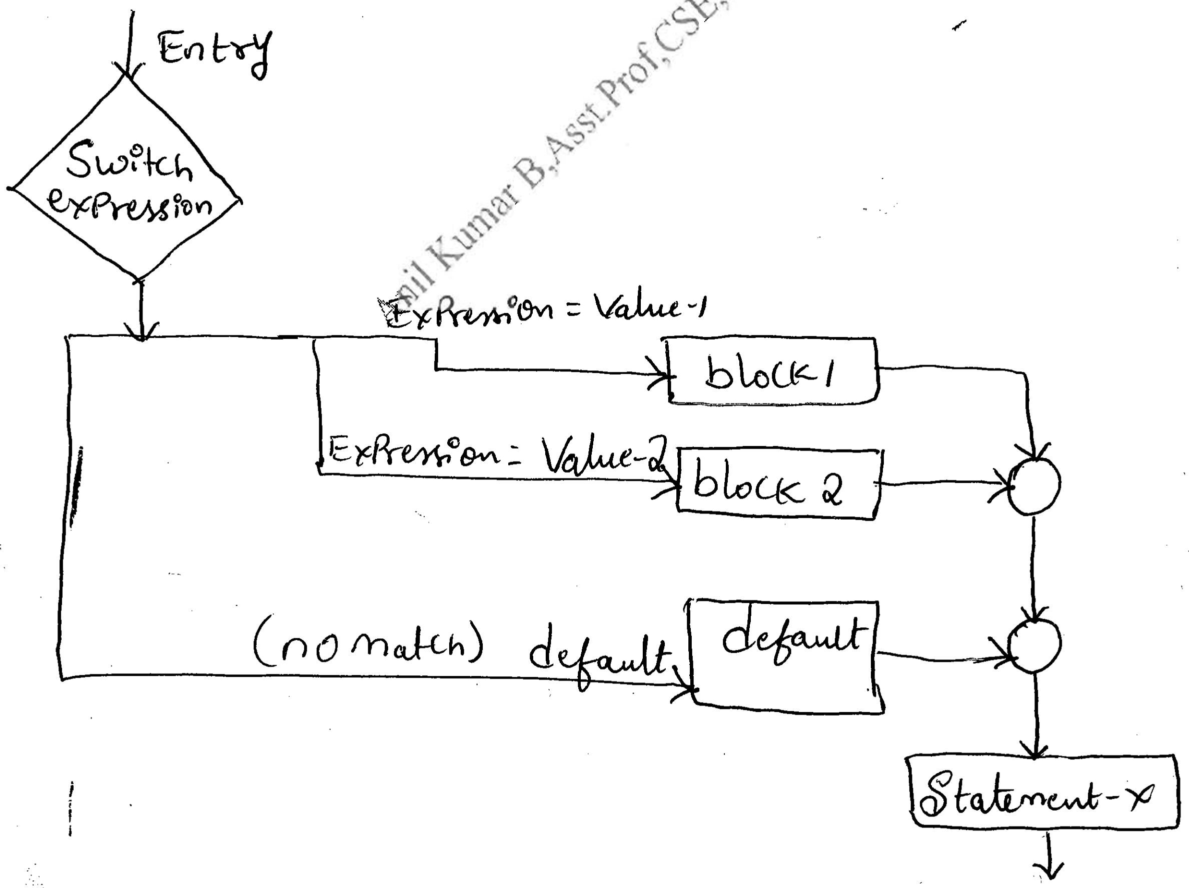


Fig:- Selection Process of Switch Statement

Example:

```
#include<stdio.h>
Void main()
{
    float a, b, res;
    Char op;
   Printf("Enter Expression \n");
   Scanf(" %f %c %f", &a, &op, &b);
   Switch (op)
   {
     Case '+' :
            res = a+b;
            break;
     Case '-' :
            res = a-b;
            break;
     Case '*' :
            res = a*b;
            break;
     Case '/' :
            if(b!=0)
             res= a/b;
            else
            {
              Printf(" divide by zero\n");
              exit(0);
            }
default : Printf(" Illegal operator\n");
            exit(0);
  }
  Printf(" %f %c %f = %f \n", a, op, b, res);
}
```

Output

```
Enter the expression.
 5+ 6

 5 + 6 = 11.
```

Ternary Operator:- (? :)

The operator which is combination of ? and : , and takes three operands. Operator known as the Conditional operator.

The general form:

Conditional expression ? expression1 : expression2

For example:-

```
if (x <0)
  flag=0;
else
  flag=1;
```

Can be rewritten as.

```
flag = (x<0)? 0 :1 ;
```

Example:

```
#include<stdio.h>
Void main()
{
    int x, y, Small;
   Printf(" enter value for x,y\n");
   Scanf("%d%d", &x,&y);
    Small= (x<y)?x:y;
    Printf(" Small= %d", Small);
}
```

Output:

```
Enter value of x,y
 5    10
 Small = 5
```

GOTO Statement:

The goto statement used to branch unconditionally from one point to another point in the Program.

The goto requires a label to identify the place where branch is to be made. A label may be any valid variable name and must be followed by colon.

The label is placed immediately [illegible] statement where control is to be transferred.

The general form:

```
Goto label; ─┐
- - - - -    │
- - - - - -  │
label: <─────┘
Statement;
```

Forward Jump

```
Label: <─────┐
Statement;   │
- - - -      │
- - - - - -  │
goto label; ─┘
```

Backward Jump

If the label: is before the Statement goto label; a loop will be formed and Some Statements will be executed repeatedly known as backward Jump.

If the label: is Placed after the goto label; Some Statements will be Skipped and Jump is known as a forward Jump.

Example:-

```
#include<Stdio.h>
Void main()
{
    double x, y;
    read:
    Scanf (" %f ", &x);
    if (x<0) Goto read;
    y= Sqrt(x);
    Printf("%f %f \n", x, y);
    Goto read;
```

Loops in C Programming:

* for loop
* while loop
* do-while loop

for loop:

The for loop is an entry-controlled loop that provides a more concise loop control structure.

General form:

```
for(initialization; test-condition; increment)
  {
      Body of loop
  }
```

The execution of for statement is as follows:

1. Initialization of the control variables is done first, using assignment statements such as i=1, count = 0. here, i and count are loop control variables.
2. The value of control variables is tested using the test-condition. Relational expressions are used as test conditions as i<10, etc.

3. The Control Variable is again tesated and incremented using an assignment Statement Such as i=i+1;

Example:

```
        ----
        Sum = 0;
        for(n=1; n<=10; n=n+1)
         {
              Sum = Sum+n*n;
         }
```

Here the body of the loop [Sum=Sum+n*n;] is executed 10 times for n=1, 2, 3, ---> 10 each time incrementing the Sum by the Square of the value of n.

<u>Program</u>:
```
#include <stdio.h>
Void main()
{
   int x;
   for (x=0; x<=9; x=x+1)
   {  Printf("%d", x);
   }
   Printf("\n");
}
```

<u>OutPut</u>
```
0    6
1    7
2    8
3    9
4
```

The While Statement:

General form of while:

```
while (test condition)
{
    body of the loop
}
```

The while is an entry-controlled loop Statement. The test-condition is evaluated and if the Condition is true then the body of the Loop is executed. After execution of the body, the test-condition is once again evaluated and if it is true, the body is executed once again. The body continues until test Condition is false and control transferred out of the Loop.

Example: - - - -

```
        Sum = 0;
        n = 1;
      ->while (n<=10)
      | {
Loop. |    Sum = Sum + n * n;
      |    n = n+1;
      ->}
        Printf(" Sum= %d \n", Sum);
        - - - -
```

Program:

```
#include<stdio.h>
Void main()
{
  float x, y;  int Count, n;
 Printf(" Enter Values of X and n \n");
 Scanf(" %f %d", &x, &n);

  y=1.0;
  Count=1;
  While (Count<=n)  /* Testing */
  {
     y=y * x;
     Count ++;            /* Incrementing */
  }
  Printf(" x=%f  n=%d  x to Power n =
                %f \n", x, n, y);
 }
```

Output:

```
Enter Values of X and n:
 2.5  4
x = 2.500000  n=4  X to Power n=39.062500
```

The above is the Program to evaluate the equation $y = x^n$.

The Do Statement: (do-while loop)

In Some Situations it is necessary to execute the body of loop before test is Performed. at Such Situations Can be handled With the help of the do Statement.

General form:

```
do
{
    body of the loop
}
while (test-Condition);
```

On reaching the do Statment, the Program Proceeds to evaluate the body of the loop first. at the end of loop the test Condition in while Statement is evaluated The Process Continues as long as the Condition is true.

Example:
```
do
{
  Printf("Input a number\n");
  number=getnum();
}
while (number>0);
```

Program:
```
#include<stdio.h>
Void main()
{
    int i=1;
    int Sum=0;                /*Initializing */
   do
   {
      Sum = Sum+i;
       i = i+2;             /* Incrementing */
    }
   While ( Sum<40 || i < 10);
   Printf("%d%d", i, Sum); /* Testing*/
}
```

Comparision of Three Loops:

for	while	do

for(n=1; n<=10; ++n)
{
- - - -
- - - -
}
	n=1;
	while (n<=10)
	{
	- - - -
	- - -
	n=n+1;
	}
		n=1
		do
		{
		- - - -
		- - - -
		n =n+1;
		}
		while(n<=10);
		- - -

The break Statement:

The break Statement is a Jump Statement Which Can be Used in Switch Statement and loops.

→ In Switch Statement break Causes the Control to terminate the Switch and following Statements will be executed.

→ In loops the Control Comes out of the loop and Statement following the loop will be executed [Ex:- for, while, do-while]

Syntax:

```
for (------)
{
    -------
        for (-----)
        {
            -----
                if (error1)
                    break;
            -----
        }
```

Note:- The break Statement Causes the inner Loop to be terminated. Outer loop will be active.

Example :-

```
int i=1;
for(;;)
{       if (i==5) break;
        Printf(" %d", i++);
}
```

Output:

```
1 2 3 4
```

The Continue Statement!

During execution of a loop, it may be necessary to skip a Part of the Loop based on some condition. In such case continue statement will be used. The continue is used to terminate the current iteration of the loop.

Syntax:

```
while (expression)
{
     Action-1;
     Continue;
     Action-n;
}
```

```
do
{
      Action-1;
      Continue;
      Action-n;
}while (expression);
```

```
for (exp1, exp2, exp3)
{
      Action-1;

      continue;
      Action-n;
}
```

Example:-

```
#include<stdio.h>
Void main()
{
    int i;
    for (i=1; i<=5; i++)
    {
        if (i==2) Continue;
        Printf("%d", i);
    }
}
```

Note:- i = 1 2 3 4 5

Output

```
1   3 4 5
```

The difference between Break and Continue:-

Break Statement	Continue Statement.
* When break is executed, the Statements following break are skipped and Causes the loop to be terminated.	* When Continue Statment is executed, the Statements following Continue are skipped and Causes Loop to be continued with next iteration.
* It Canbe used in Switch Statements	* It Cannot be used inside Switch.

* Example:-
`for(i=1;i<=5;i++)`
`{`
`  if(i==3) break;`
`  Printf("%d", i);`
`}`
 	
Output:
1 2
	* Example:-
	`for(i=1;i<=5;i++)`
	`{ if (i==3) Continue;`
	`  Printf("%d", i);`
	`}`
	
	Output:
	1 2 4 5