OPERATOR & CONTROL STATEMENT

IN

‘C’

Operator:-

C supports a rich set of operators. Operator is a symbol that tells computer to perform certain mathematical and logical manipulations.

Types of operator

1) ARITHMETIC OPERATOR
2) RELATIONAL OPERATOR
3) LOGICAL OPERATOR
4) ASSIGNMENT OPERATOR
5) UNARY OPERATOR
6) CONDITIONAL OPERATOR
7) BITWISE OPERATOR

Arithmetic operator:-

Arithmetic operator *are used for mathematical calculation these operator are binary operator that work with integer floating point number and every character.*

:

Arithmetical operator are:

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Module division

Relational operator:-

Relational operator *are used to compare two operands. Operands may be variable, constant or expression.*

Operator	Meaning
<	Is less than
<=	Is less than equal to
>	Is greater than
>=	Is greater than equal to
==	Equal to
!=	is not equal to

Example of Relational operator:-

```
main()
{
      int a=10,b=20,c=30,d,e;
     d=a>b;
      e=b<=c;
      printf("%d %d",d,e);
      getch();
}
```

Output:-

0 1

LOGICAL OPERATOR:-

Are used to combine (compare) two or more condition.
Logical Operator are:-

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

Types of operator:-

- *Logical AND compare two operands and return 1if both condition are true else return 0 (false)*
- *Logical OR compare two operand and return 1 if any one condition true.*
- *Example:-*

Condition		AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

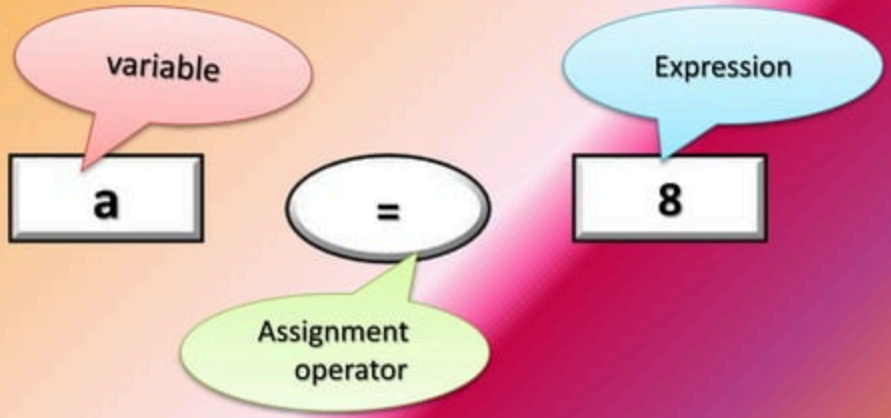
- *Logical NOT if the condition is true result is false and if the condition is false result true .*

Example:-

Condition	NOT
0	1
1	0

Assignment Operators:-

Assignment operator *are used to assign the value or an expression or a value of a variable to another variable*



UNARY Operator:-

Unary *operator is also called Increments & decrement operator. The increment operator (**++**) adder on to the variable and decrement (**- -**) subtract one from the variable. There are following unary operator*

Operator	Meaning
++x	Pre increment
- -x	Pre decrement
x++	Post increment
X- -	Post decrement

CONDITIONAL OPERATOR:-

*The **conditional** operator is ternary operator, which operates On the three operands.*

Example:-

```
main()
{
        int a=10,b=5,big;
        big=a>b ? a:b;
        printf("Big is %d",big);
        getch();
}
```

Output is:-

10

Bitwise Operator:-

Are used by the programmer to communicate directly with the hardware.These operator are used for designing bit or shifting them either right to left, left to right.

Exan

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift

Equality operator:-

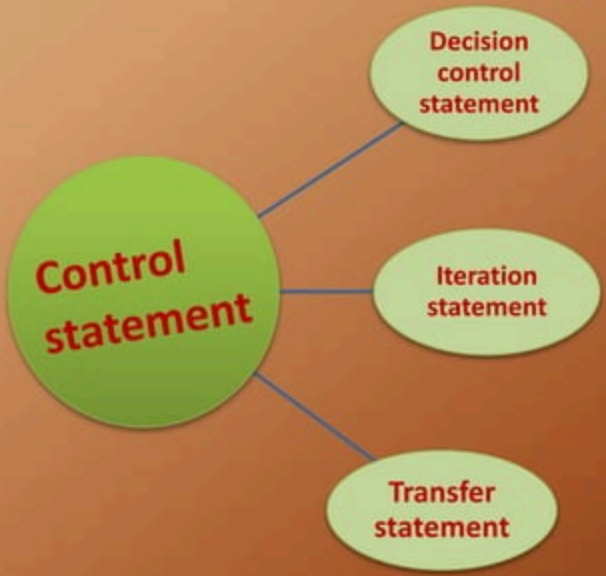
Equality *operator is used for compression between two operands these operator.*

Example:-

Operator	Meaning
==	Equal to
!=	Not equal to

Control Statement

There are three types of statement:-



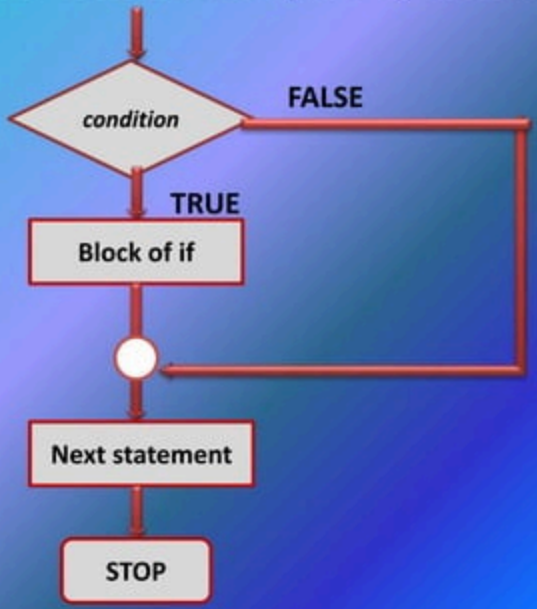
Decision Control statement:-

Decision *control statement disrupt or alter the sequential execution of the statement of the program depending on the test condition in program*

Types of Decision control statement:-

IF STATEMENT:

The If statement is a powerful decision making statement and is used to control the flow of execution of statement.

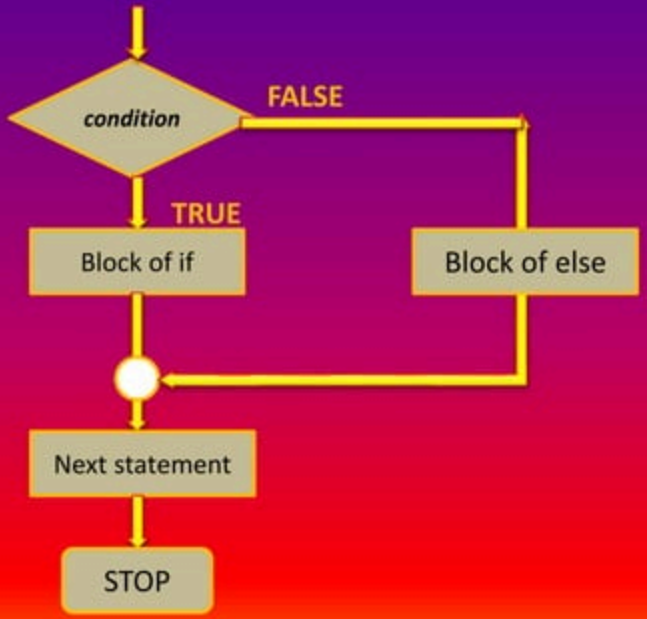


Example of IF statement

```
main()
{
        int a;
        printf("enter value of a");
        scanf("%d",&a);
 if(a>25)
  {
        printf("no.is greater than 25");
   }
        printf("\n bye");
       getch();
}
```

If else statement:-

If the condition is true the true block is execute otherwise False block is execute.



Example of If else statement

```
main()
{
        int n,c;
        printf("\n enter value of n");
        scanf("%d",&n);
        c=n%2;
  if(c==0)
        printf("no is even");
     else
        printf("no is odd");
        getch();
}
```

What Is Else If Ladder:

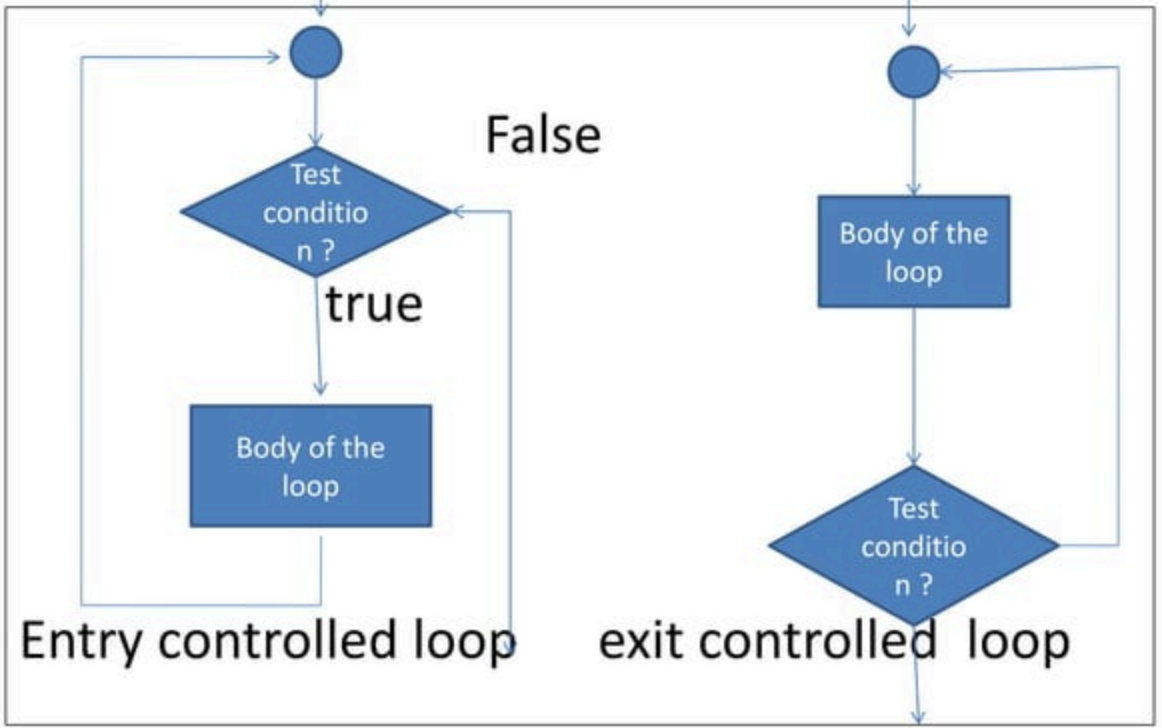
If we are having different - different test conditions with different - different statements, then for these kind of programming we need else if ladder

Syntax Of Else If Leader:

```
if(test_condition1)
  {
    statement 1;
  }
else if(test_condition2)
  {
    statement 2;
  }
else if(test_condition3)
  {
    statement 3;
  }
else if(test_condition4)
  {
    statement 4;
  }
else
{
}
```

```
void main ( )
{
 int num = 10 ;
if ( num > 0 )
printf ("\n Number is Positive");
 else if ( num < 0 )
printf ("\n Number is Negative");
 else printf ("\n Number is Zero");
 }
```

- Depending on the position of control statement in c,control structure may be classified
- Entry_ controlled loop
- Exit _controlled loop


Test
conditio
n ?
False
true
Body of the
loop
Entry controlled loop
Body of the
loop
Test
conditio
n ?
exit controlled loop

- C language provides three constructs for perfoming loop operations

- While statement

- Do statements

- For statements

```
While(test condition)
{

body of the loop
}
```

```
.....
.....
int_sum=0;
int_n=1;
while(int_n<=10)
{
  int_sum=int_sum+int_n;
  int_n=int_n+1;
    }
printf("sum=%d\n",int_sum);
.....
```

Do statement

```
do
{

Body of the loop
}
  While(test condition)
```

```
int_i=1;
 int_sum=0;
 do
{
   int_sum=int_sum+I
    i=i+2;
    }
while(sum<40||i<10);
printf("%d %d\n,I,sum);
```

For statements

```
For(intialization;testcondition;icrement)
                    {
                body of the loop
                    }
```

- Intialization of control variable done first using assignment statement
- The value of control variable tested using test condition,ie reletional expression such as i>10,that determine when the loop will exit
- If the condition is true ,the body of loop executed,otherwise terminated

```
   int_sum=0;
  for(int_n=1;int_n<=10;int_n++)
{
   int_sum=int_sum+int_n;
 }
 printf("sum=%d\n",int_sum);
```

Nesting of for loop

```
For(i=0;i<n;i++)
{
  ……..
For(j=0;j<n-1;j++)
{
 ……..
}
}
```

Jumping out of a loop

- Exit from a loop using break statement

  if a break statement encounteredd in a loop,the loop will immidiatly exited and the program continues with the statement immidiatly following loop;ie break will exit only a single loop

Eg:

```
while(test condition)
{
.....
.....
if(condition)
break;
```

Skipping a part of loop

Another statement ‘continue’,

- It tells the compiler skip the following statements and continue with next iteration

Eg:

```
While (test condition)
{
.....
If(.....)
Continue;
```

Switch statement

Switch *statement is a multi-way decision making statement which selects one of the several alternative based on the value of integer variable or expression.*

Syntax :-

```
switch(expression)
{
case constant : statement;
break;
default : statement;
}
```

EXAMPLE OF SWITCH STATEMENT

```
main()
{
char choice;
printf("enter any alphabet");
scanf("%d",& choice);
switch(choice)
{
case 'a':
        printf("this is a vowel  \n");
        break;
case 'e' :
        printf("this is a vowel  \n");
        break;
case 'i' :
        printf("this is a vowel  \n");
        break;
case 'o'  :
        printf("this is a vowel  \n");
        break;
case 'u' :
        printf("this is a vowel  \n");
        break;
default  :
  printf("this is not a vowel");
getch();
}
}
```

Go To statement

A GO TO statement can cause program control to end up anywhere in the program unconditionally.

Example :-

```
main()
{
            int i=1;
up :    printf("Hello To C")
         i++;
   If (i<=5)
         goto up
         getch();
}
```