Chapter 3-
Operators In C

MAJOR C OPERATORS :

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Increment and Decrement Operators
- Special Operators

ARITHMETIC OPERATORS

Arithmetic operators are used to perform mathematical calculations like **addition, subtraction, multiplication, division and modulus**. List of Arithmetic operator in C:

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Division

ARITHMETIC OPERATORS

Operators +, -, / and * known to us. The % operator known as modulo division produces the reminder of an integer division. Example: Suppose a and b are two variables

a+b	**This performs addition on the operands a and b**
a-b	This performs Subtraction on the operands a and b
a*b	This performs Multiplication on the operands a and b
a/b	This performs Division on the operands a and b
a%b	This produces the reminder of the integer division of a and b

Note that the modulo division operator (%) are not applicable on floating point data.

ARITHMETIC OPERATORS

The following program shows use of some arithmetic operators:

```
#include <stdio.h>
int main(){
   int a=9,b=4,c;
   c=a+b;
   printf("a+b=%d\n",c);
   c=a-b;
   printf("a-b=%d\n",c);
   c=a%b;
   printf("Remainder when a divided by b=%d\n",c);
   return 0;
}
```

SPECIAL OPERATORS

Comma operator:

- Comma operators are used to link related expressions together. For example:
- int a, c=5,d;
- Here, three integer variable are declared in a line and c is assigned value 5 and the work is related so they are done in a line, separated using coma operator.

USING SCANF

The **scanf function allows you to accept input from standard in**, which for us is generally the keyboard.

```
#include <stdio.h>

int main()
{
    int a, b, c;
    printf("Enter the first value:");
    scanf("%d", &a);
    printf("Enter the second value:");
    scanf("%d", &b);
    c = a + b;
    printf("%d + %d = %d\n", a, b, c);
    return 0;
}
```

RELATIONAL OPERATOR

For comparing the value of two variables for making certain decision relational operators are used.

Suppose we want to compare weight of two people or price of two mobile and many more comparison, these comparison can be done using relational operators.

Relational operators supported by C:

Operator	Meaning
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
==	Equal to
!=	Not equal to

RELATIONAL OPERATOR

Expression containing relational operator in known as relational expression. The resulting value of a relation expression is either zero or one. **The result will be one if the condition is true and zero if false.**

Example:

4.5 < 2	**False**
30 > 29.5	True
10 != 3+7	False
20 == 4	False

RELATIONAL OPERATOR

The following program shows use of some relational operators:

```
#include <stdio.h>
int main()
{
   int m=40,n=20;
   if (m == n)
   {
     printf("m and n are equal");
   }
   else
   {
     printf("m and n are not equal");
   }
   return 0;
}
```

Output of the program will be like as:

```
m and n are not equal
```

LOGICAL OPERATOR

Logical operators supported by C are:

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

When we need to test more than one condition for decision making logical operators can be used. **Example: age1 > age2 && weight<=65. This means age1 is greater than age2 and weight is less or equal 65 than condition will be true.**

LOGICAL OPERATOR

An expression which combines relation expressions are called logical expression. The resulting value of an expression is either true or false.

- In case of **Logical AND** the result will be true if all the relational expression within the logical expression is true, otherwise result will be false.
- In case of **Logical OR** the result will be true if anyone of the relational expression within the logical expression is true, otherwise result will be false.
- In case of **Logical NOT** the result will be true if the relational expression is false.

LOGICAL OPERATOR

The following program uses the logical operator:

```
#include <stdio.h>
int main()
{
    int m=40,n=20;
    int o=20,p=30;
    if (m>n && m !=0)
    {
        printf("&& Operator : Both conditions are true\n");
    }
    if (o>p || p!=20)
    {
        printf("|| Operator : Only one condition is true\n");
    }
    if (!(m>n && m !=0))
    {
       printf("! Operator  : Both conditions are true\n");
    }
    else
    {

       printf("! Operator  : Both conditions are true. " \
            "But, status is inverted as  false\n");
    }
    return 0;

}
```

LOGICAL OPERATOR

The output of the program will be:

```
&& Operator : Both conditions are true
|| Operator : Only one condition is true
! Operator : Both conditions are true. But, status is inverted as false
```

INCREMENT & DECREMENT OPERATOR(LATER TO BE USED)

For incrementing value of variable such as x,

we write like **x = x + 1.**

We can write as like **x+=1**. This mean add 1 with x and assign the new value to x. This is a shorthand of assignment operator. There are some more shorthand of assignment operator, they are:

Normal Statement	Shorthand operator expression
a=a+1	a += 1
a=a-1	a -= 1
a=a*(x + y)	a *= (x + y)
a=a/(x + y)	a /= (x + y)
a=a%b	a %= b