Unit - 2

Operators:

Introduction, Arithmetic Operators, Relational Operators, Logical Operators, Assignment Operators, Increment and Decrement Operators, Conditional Operator, Bitwise Operators, Conditional Operator, Special Operators: Comma and 'sizeof' operator.

Expressions:

Arithmetic Expressions, Evaluation of Expressions, Precedence of Arithmetic Operators, Type Conversions in Expressions, Operator Precedence and Associativity.

Operators:
C supports a rich set of built-in operators.
C operators can be classified into a number of categories. They include:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Increment and decrement operators
6. Conditional operators
7. Bitwise Operators
8. Special Operators

Note:

1. If operator requires only one operand, then it is called unary
2. If operator requires two operands, then it is called binary
3. If operator requires three operands, then it is called ternary

1. Arithmetic Operators:

C provides all the basic arithmetic operators. They are listed as given below:

Sl.No.	**Operator**	**Meaning**
1	+	Addition
2	-	Subtraction
3	*	Multiplication
4	/	Division
5	%	Modulo division

All the above are binary operators.

2. Relational Operators:

It is used to compare two operands. C Supports 6 relational operators. These operators and their meaning are as shown below:

Sl.No.	**Operator**	**Meaning**
1	<	Less than
2	<=	Less than or equal to
3	>	Greater than
4	>=	Greater than or equal to
5	==	Equal to
6	!=	Not equal to

3. **Logical Operator:**
In addition to the relational operators, C has the following three logical operators:

Sl.No.	**Operator**	**Meaning**
1	&&	Logical AND
2	||	Logical OR
3	!	Logical Not

The logical operators && and || are used when we want to test more than one condition and make decisions. An example is:

(a>b) && (a>c)

Logical AND uses AND truth table:

AND truth table:

Input1	**Input 2**	**Output**
0	0	0
0	1	0
1	0	0
1	1	1

Logical OR used OR truth table:

OR truth table:

Input1	**Input 2**	**Output**
0	0	0
0	1	1
1	0	1
1	1	1

Logical NOT used to negate the input. It uses NOT truth table:

NOT truth table

Input	**Output**
0	1
1	0

4. **Assignment Operators:**
They are used to assign the result of an expression to a variable. “=” is the assignment operator. To the left of assignment operator, it is compulsory to have variable and to the right side it can be a variable or constant or an expression.

Syntax:

Variable = expression;

X=67
A=b;
R=(a+b);

In addition to the above assignment operator, ‘C’ Has a set of shorthand assignment operator . They are also called as arithmetic assignment operator.

Syntax:
v = v op (exp)

where ‘v’ is the variable
‘op’ is the operator
‘exp’ is the expression.

Ex:

Statement with simple assignment operator	Statement with Arithmetic assignment operator
a=a+1	a+=1
a=a-b	a-=b
a=a*35	a*=35
a=a/1	a/=b
a=a%b	a%=b

5.

Postfix increment

```
/* example for Increment operator */
#include<stdio.h>
int main()
{
   int x=10;
   x++;
   printf("Value of x = %d\n",x);
}

Value of x = 11
```

Prefix increment

```
/* example for Increment operator */
#include<stdio.h>
int main()
{
   int x=10;
   ++x;
   printf("Value of x = %d\n",x);
}

Value of x = 11
```

Prefix decrement

```
/* example for decrement operator */
#include<stdio.h>
int main()
{
   int x=10;
   --x;
   printf("Value of x = %d\n",x);
}
```

```
Value of x = 9
```

Postfix decrement

```
/* example for decrement operator */
#include<stdio.h>
int main()
{
   int x=10;
   x--;
   printf("Value of x = %d\n",x);
}
```

```
Value of x = 9
```

```
/* example for increment operator with assignment operator */
#include<stdio.h>
int main()
{
   int x=10,y;
   y=x++;
   printf("Value of x = %d\n",x);
   printf("Value of y = %d\n",y);
}
```

```
Value of x = 11
Value of y = 10
```

```
/* example for increment operator with assignment operator */
#include<stdio.h>
int main()
{
   int x=10,y;
   y=++x;
   printf("Value of x = %d\n",x);
   printf("Value of y = %d\n",y);
}
```

```
Value of x = 11
Value of y = 11
```