Operators

➲ **What is an operator?**

An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables.

➲ **What is an operand?**

A constant or a variable or a function which returns a value is an operand. An operator may have one or two or three operands.

Example: a+b-c*d/e%f

In this example there are :

6 operands i.e.., a, b, c, d, e, f

5 operators i.e.., +,-,*,/ and %.

What are the types of operators?

➲ **C operators can be classified into a number of categories based on type of operation**

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

Arithmetic operators

➲ C provides all basic arithmetic operators. The operators are +, - ,* , / , %.

➲ Let a=10, b=5. Here, a and b are variables and are known as operands.

Description	Symbol	Example	Result	Priority
Addition	+	a+b = 10 + 5	15	2
Subtraction	-	a-b = 10 – 5	5	2
Multiplication	*	a* b = 10 * 5	50	1
Division(Quotient)	/	a/b = 10/5	2	1
Modulus(Reminder)	%	a % b = 10 % 5	0	1

➲ **/*Program to illustrate the use of arithmetic operators is given below*/**

```
#include<stdio.h>
        void main()
        {
                int  a=30,b=5,c,d,e,f,g;
                c=a+b;
                d=a-b;
                e=a*b;
                f=a/b;
                g=a%b;
                printf("%d, %d, %d ,%d ,%d\n", c,d,e,f,g);
        }
```

Output: 35,25,150,6,0

Relational Operators

➲ We often compare two quantities depending on their relation, take certain decisions.

➲ For example, we compare the age of two persons, or the price of two items, and so on. These comparisons can be done with the help of relational operators.

An expression such as

A< B or 1<20

➲ Expression with a relational operator is termed as a relational expression. The value of relational expression is either **one or zero**. It is one if the specified relation is **true** and zero if the relation is **false**.

Example: 10 < 20 is true.

20 < 10 is false.

➲ C supports six relational operators in all. These operators are

Operator	Meaning	Priority
<	is less than	1
<=	is less than or equal to	1
>	is greater than	1
>=	is greater than or equal to	1
==	is Equal to	2
!=	is not equal to	2

➲ When arithmetic expressions are used on either side of a relational operator, the arithmetic expressions will be evaluated first and then the results compared. That is arithmetic operators have a higher priority over relational operators.

Example: a+b>c+d

- In the above example first it will perform addition operation on both sides and afterwards it will compare the results.Relational expressions are used in decision statements such as if and while to decide the course of action of a running program.

Logical Operators

- In addition to the relational operators, C has the following three logical operators.

Operator	*Meaning*
&&	logical AND
||	logical OR
!	logical NOT

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

  Example: a>b && x==10

- The above logical expression is true only if a>b is true and x==10 is true. If either (or both) of them are false, the expression is false.

Assignment Operators

- Assignment operators are used to assign the result of an expression to a variable or the RHS value to the LHS variable. We have seen the usual assignment operator,"=".
- In addition, C has a set of "shorthand" assignment operators of the form. (+=,-=,/=,*=,etc..)
- **Syntax:** v op = exp;

  Where: v is a variable,

  exp is an expression

  op is a C library arithmetic operator

  op= is known as the shorthand assignment operator.
- The assignment statement:

  v op= exp;

  is equivalent to v= v op (exp).

  with v evaluated only once.
- Consider an example

  x + = y+1;

  This is same as the statement

  x = x+ (y+1);

Increment and Decrement Operators

- C allows two very useful operators not generally found in other languages. These are the increment and decrement operators:

  ++ and --

- The operator ++ adds 1 to the operand value, while -- subtracts 1. Both are unary operators takes the following.
  - ++m; or m++;
  - --m; or m--;
  - ++m is equivalent to m=m+1;
  - --m is equivalent to m=m-1;
- We use increment and decrement statements in for and while loops extensively. While ++m and m++ mean the same thing when they form the statements independently, they behave differently when they are used in expressions on the right hand side of an assignment statement.
- **Consider the following**

  1. m=5;

     y=++m; // In this case, the value of y and m would be 6.

  Suppose, if we rewrite the above statements as

  2. m=5;

     y=m++; // Here the value of y would be 5 and m would be 6.

- A prefix operand first adds 1 to the operand value and then result is assigned to the variable on left. On the other hand, a postfix operator first assigns the value to the variable on left and then increments the operand value.

Note:

- This operator has four types of representation:
  - Pre- Increment operator (++a): Value of Operand is incremented by 1 first then expression is evaluated.
  - Post-Increment Operator (a++): Value of Operand is incremented by 1 after then expression is evaluated.
  - Pre-Decrement Operator (--a): Value of Operand is decremented by 1 first then expression is evaluated.
  - Post-Decrement Operator (a--): Value of Operand is decremented by 1 after then expression is evaluated.

Conditional Operator (OR) Ternary Operator

- A ternary operator pairs"**? :**" is available in C to construct conditional expressions of the form
- **Syntax:** exp1? exp2: exp3

  Where:- exp1,exp2, and exp3 are expressions.
- The operator ?: works as follows: exp1 is evaluated first. If it is nonzero (true), then the expression exp2 is evaluated and becomes the value of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression. Note that only one of the expressions is evaluated.

Example:

Consider the following statements.

```
a=10;
b=15;
x = (a>b)? a:b;
```

In this example, x will be assigned the value of b.

This can also be achieved using if… else statements as follows:

```
if (a>b)
        x=a;
else
        x=b;
```

Bitwise Operators

- C has a distinction of supporting special operators known as bitwise operators for manipulation of data at bit level. These operators are used for testing the bits, or shifting them right or left. Bitwise operators may not be applied to float or double.

Operator	***Meaning***
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right
~	Bitwise Negate

Example:

The following table lists the Bitwise operators supported by C. Assume variable 'A' holds 60 and variable 'B' holds 13, then:

Operator	Description	Example
&	Operator copies a bit to the result if it exists in both operands	(A&B) = 12 i.e., 0000 1100
|	Operator copies a bit if it exists in either operand	(A | B) = 61 i.e., 0011 1101
^	Operator copies the bit if it is set in one operand but not both	(A ^ B) = 49 i.e., 0011 0001
~	Operator is unary and has the effect of 'flipping' bits	(~A) = 61 i.e., 1100 0011 in 2's complement form
<<	The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	The left operands value is moved right by the numb er of bits specified by the right operand	A >> 2 = 15 i.e., 0000 1111

Special Operators

- C supports some special operators of interest such as comma operator, sizeof operator.

✓ ***The Comma Operator***

- The comma operator can be used to link the related expressions together. A comma-linked list of expressions are evaluated from left to right and the value of right-most expression is the value of the combined expression.

  Example, the statement: value = (x=10,y=5,x+y)

  - First assigns the value 10 to x, then assigns 5 to y , and finally assigns 15 to value. Since comma operator has the lowest precedence of all operators, the parentheses are necessary.

✓ ***The sizeof() Operator***

- The sizeof() is a compile time operator and, when used with an operand, it returns the number of bytes the operand occupies. The operand may be a variable, or a constant or a data type qualifier.

  Examples: m=sizeof(sum);

  n= sizeof(5);

  It can also be used in printf statement:

  printf("size of integer data type is %d", sizeof(int));

Classification Based on number of Operands

The Operators are classified into four major categories based on the number of operands as shown below:

(i) Unary Operators

(ii) Binary Operators

(iii) Ternary Operators

(iv) Special Operators

- **Unary Operators:** An operator which acts on only one operand to produce the result is called unary operator.

  Examples: --b, a++, c--

- **Binary Operators:** An operator which acts on two operands to produce the result is called a binary operator. In an expression involving a binary operator, the operator is in between two operands.

  Examples: a + b, a*b

- **Ternary Operator:** An operator which acts on three operands to produce a result is called a ternary operator. Ternary operator is also called conditional operator.

  Example:a ?b : c;