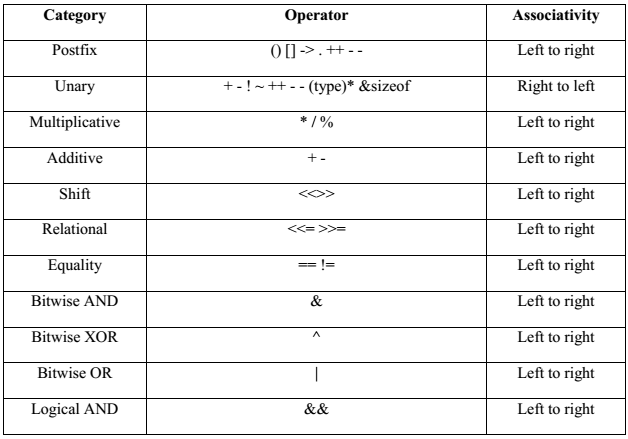
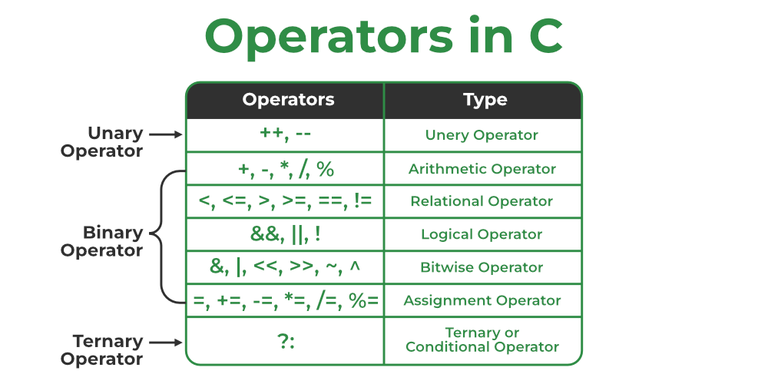
OPERATORS IN COMPUTER



Types of operators

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Other Operators



*Operators in C*

The above operators have been discussed in detail:

**1. Arithmetic Operations in C**

These operators are used to perform arithmetic/mathematical operations on operands. Examples: (+, -, \*, /, %, ++, –). Arithmetic operators are of two types:

**a) Unary Operators:**

Operators that operate or work with a single operand are unary operators. For example: Increment(++) and Decrement(–) Operators

int val = 5;

cout<<++val;

**b) Binary Operators:**

Operators that operate or work with two operands are binary operators. For example: Addition(+), Subtraction(-), multiplication(\*), Division(/) operators

int a = 7;

int b = 2;

cout<<a+b;

**2. Relational Operators in C**

These are used for the comparison of the values of two operands. For example, checking if one operand is equal to the other operand or not, whether an operand is greater than the other operand or not, etc. Some of the relational operators are (==, >= , <= )

int a = 3;

int b = 5;

cout<<(a < b);

// operator to check if a is smaller than b

**3. Logical Operator in C**

Logical Operators are used to combining two or more conditions/constraints or to complement the evaluation of the original condition in consideration. The result of the operation of a logical operator is a Boolean value either **true** or **false**.

For example, the **logical AND** represented as the **‘&&’ operator in C** returns true when both the conditions under consideration are satisfied. Otherwise, it returns false. Therefore, a && b returns true when both a and b are true.

count<<((4 != 5) && (4 < 5)); // true

**4. Bitwise Operators in C**

The Bitwise operators are used to perform bit-level operations on the operands. The operators are first converted to bit-level and then the calculation is performed on the operands. Mathematical operations such as addition, subtraction, multiplication, etc. can be performed at the bit level for faster processing. For example, the **bitwise AND** operator represented as **‘&’ in C** takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1(True).

int a = 5, b = 9; // a = 5(00000101), b = 9(00001001)

cout << (a ^ b)// 00001100

cout <<(~a); // 11111010

**5. Assignment Operators in C**

Assignment operators are used to assign value to a variable. The left side operand of the assignment operator is a variable and the right side operand of the assignment operator is a value. The value on the right side must be of the same data type as the variable on the left side otherwise the compiler will raise an error.

Different types of assignment operators are shown below:

### ****a) “=”****

This is the simplest assignment operator. This operator is used to assign the value on the right to the variable on the left.   
**Example:**

a = 10;

b = 20;

ch = 'y';

### ****b) “+=”****

This operator is the combination of the ‘+’ and ‘=’ operators. This operator first adds the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.   
**Example:**

(a += b) can be written as (a = a + b)

If initially value stored in a is 5. Then (a += 6) = 11.

### ****c) “-=”****

This operator is a combination of ‘-‘ and ‘=’ operators. This operator first subtracts the value on the right from the current value of the variable on left and then assigns the result to the variable on the left.   
**Example:**

(a -= b) can be written as (a = a - b)

If initially value stored in a is 8. Then (a -= 6) = 2.

### ****d) “\*=”****

This operator is a combination of the ‘\*’ and ‘=’ operators. This operator first multiplies the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.   
**Example:**

(a \*= b) can be written as (a = a \* b)

If initially, the value stored in a is 5. Then (a \*= 6) = 30.

### ****e) “/=”****

This operator is a combination of the ‘/’ and ‘=’ operators. This operator first divides the current value of the variable on left by the value on the right and then assigns the result to the variable on the left.   
**Example:**

(a /= b) can be written as (a = a / b)

If initially, the value stored in a is 6. Then (a /= 2) = 3.