Operators

Operators are symbols that help us to perform specific mathematical and logical computations on **operands**. In simpler words, we can say that an operator operates the operands.
For example: consider the below statement:

c = a + b;

Here **'+'** is an operator which is representing ***addition*** of a and b.

Operands – Operands are the objects which are manipulated by applying operators.

Classification of Operators

The operators can be classified into various types according to the functions they perform. So, the types of operators used in programs are: -

Arithmetic Operators

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

I. **Unary Operators**: Operators that operates or works with a single operand are unary operators. *For example:* (++, --)
II. **Binary Operators**: Operators that operates or works with two operands are binary operators. *For example:* (+, –, *, /)

For example: a+b, a-b, a++, c-- etc. Here in the example a++ means a is to be increased by 1 and c-- means that c be reduced by 1.

Relational Operators

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

For example: a>=b signifies that a is either greater than or equal to b.

These operators are used in conditional statements to check whether condition given is true or false and operate the set of commands respectively.

Logical Operators

Logical Operators are used to combine 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 which is represented as '&&' operator 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 (i.e. non-zero).

Some of the logical operators are *&& (Logical AND), || (Logical OR), ! (Logical NOT)*. These are used in conditional statements.

Bitwise Operators

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. The mathematical operations such as addition, subtraction, multiplication etc. can be performed at bit-level for faster processing since all tasks are being performed at bit-level.

For example, the bitwise AND represented as & operator in C or 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.

These are also used in conditional statements as they give their output as 0 or 1 depending on the type of operator used.

Some common bitwise operators are: Bitwise AND, Bitwise OR.

Assignment Operators

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

Different types of assignment operators are shown below:

I. "**=**": This is the simplest assignment operator. This operator is used to assign the value on the right to the variable on the left.
For example:

a = 60;

b = 40; //integer input

ch = 'p'; //character input

ii. "**+=**": This operator is combination of '+' and '=' operators. This operator first adds the current value of the variable on left to the value on 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 2 and in b is 5 Then (a += 5) = 7.

iii. "**-=**": This operator is combination of '-' and '=' operators. This operator first subtracts the value on 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 and in b is 6. Then (a -= 6) = 2.

Iv. **"*="**: This operator is combination of '*' and '=' operators. This operator first multiplies the current value of the variable on left to the value on 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 and in b is 6. Then (a *= 6) = 30.

v. **"/="**: This operator is combination of '/' and '=' operators. This operator first divides the current value of the variable on left by the value on 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 6 and in b is 2. Then (a /= 2) = 3.

Other Operators

Apart from the above operators there are some other operators available in C or C++ used to perform some specific task. Some of them are discussed here:

a. **Sizeof operator**: sizeof is a much used in the C/C++ programming language. It is a compile time unary operator which can be used to compute the size of its operand. The result of sizeof is of unsigned integral type which is usually denoted by size_t. Basically, sizeof operator is used to compute the size of the variable. *For example:* It can calculate the size of strings, size of an array etc.

b. **Comma Operator**: The comma operator (represented by the token ‘,’) is a binary operator that evaluates its first operand and discards the result, it then evaluates the second operand and returns this value (and type). The comma operator has the lowest precedence of any C operator. Comma acts as both operator and separator.

c. **Conditional Operator**: Conditional operator is of the form Expression1? Expression2: Expression3 . Here, Expression1 is the condition to be evaluated. If the condition (Expression1) is True, then we will execute and return the result of Expression2 but if the condition (Expression1) is false then we will execute and return the result of Expression3. We may replace the use of if...else statements by conditional operators.