Operators in C

An operator in C is a symbol that enables the execution of specific mathematical, relational, bitwise, conditional, or logical operations on values and variables. The values and variables on which these operators act are referred to as operands. Therefore, we can say that operators are symbols used to perform actions on operands.

Types of Operator

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Sizeof Operators
- Increment & Decrement Operators
- Conditional Operators
- Pointer Operators

Arithmetic Operators

We are very familiar with arithmetic operators, which are used to carry out arithmetic operations on operands. The most commonly used ones are addition (+), subtraction (-), multiplication (*), and division (/).

Additionally, the modulo operator (%) plays a key role in arithmetic, as it calculates the remainder of a division. Arithmetic operators are essential in constructing arithmetic expressions. These operators are binary, meaning they require two operands and work on numeric operands, which can be numeric literals, variables, or expressions.

Operator	Symbol	Syntax
Plus	+	a+b
Minus	-	a-b
Multiply	*	a*b
Divide	/	a/b
Modulus	%	a%b

Relational Operator

We are also familiar with relational operators from secondary-level mathematics. These operators compare two operands and return a boolean value (true or false). They are commonly used in boolean expressions.

The most widely used relational operators include less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=), equal to (==), and not equal to (!=). Like arithmetic operators, relational operators are binary, meaning they require two numeric operands.

Operators	Symbol	Syntax
Less than	<	a<b
Greater than	>	a>b
Less than or equal to	<=	a<=b
Greater than or equal to	>=	a>=b
Equal to	==	a==b
Not Equal to	!=	a!=b

Logical Operator

Logical operators are used to combine two or more conditions or to reverse the evaluation of the given condition. The outcome of a logical operator's operation is a Boolean value, either true or false.

Operators	Symbol	Syntax
Logical AND	&&	a&&b
Logical OR	||	a||b
Logical NOT	!	!a

Bitwise Operators

Bitwise operators are utilized to carry out operations at the bit level on operands. The operands are first converted to their binary form, and then the calculations are executed. Mathematical operations like addition, subtraction, and multiplication can be performed at the bit level to enhance processing speed.

Operators	Symbol	Syntax
Bitwise AND	&	a&b
Bitwise OR	|	a|b
Bitwise XOR	^	a^b
Bitwise First Complement	~	a~b
Bitwise Left Shift	<<	a<<b
Bitwise Right Shift	>>	a>>b

Assignment Operators

Assignment operators are used to assign a value to a variable. The operand on the left side of the assignment operator is a variable, while the operand on the right side is the value to be assigned. The value on the right must match the data type of the variable on the left, otherwise, the compiler will generate an error.

In C, assignment operators can be combined with other operators to perform multiple operations using a single operator. These are known as compound operators.

Operators	Symbol	Syntax
Simple Assignment	=	a=b
Plus and assign	+=	a+=b
Minus and assign	-=	a-=b
Multiply and assign	*=	a*=b
Divide and assign	/=	a/=b
Modulus and assign	%=	a%=b
AND and assign	&=	a&=b
OR and assign	|=	a|=b
XOR and assign	^=	a^=b
Right shift and assign	>>=	a>>=b
Left Shift and assign	<<=	a<<=b

sizeof Operator

- The sizeof operator is frequently used in the C programming language.
- It is a compile-time unary operator that calculates the size of its operand.
- The result of sizeof is of an unsigned integral type, commonly represented by size_t.
  Essentially, the sizeof operator is used to determine the size of a variable or datatype.

Syntax: sizeof (operand)

Increment and Decrement Operators

These are used to increase or decrease the value of a variable by 1.
++ (Increment)
-- (Decrement)

Conditional (Ternary) Operator

This is used to evaluate a condition and return one of two values depending on the result.
? : (Conditional expression)

Comma Operator

Allows multiple expressions to be evaluated in a single statement, returning the value of the last expression.
, (Comma)

Pointer Operators

These are used with pointers.
* (Dereference operator)
& (Address-of operator)

Type Cast Operator

This is used to explicitly convert one data type into another.
(type) (Type casting)