

Operators in C Language

C language supports a rich set of built-in operators. An operator is a symbol that tells the compiler to perform a certain mathematical or logical manipulation. Operators are used in programs to manipulate data and variables.

C operators can be classified into following types:

- Arithmetic operators
- Relational operators
- Logical operators
- Bitwise operators
- Assignment operators
- Conditional operators
- Special operators

Arithmetic operators

C supports all the basic arithmetic operators. The following table shows all the basic arithmetic operators.

Operator	Description
+	adds two operands
-	subtract second operands from first
*	multiply two operand
/	divide numerator by denominator
%	remainder of division
++	Increment operator - increases integer value by one

--	Decrement operator - decreases integer value by one
----	-----------------------------------------------------

Relational operators

The following table shows all relation operators supported by C.

Operator	Description
==	Check if two operand are equal
!=	Check if two operand are not equal.
>	Check if operand on the left is greater than operand on the right
<	Check operand on the left is smaller than right operand
>=	check left operand is greater than or equal to right operand
<=	Check if operand on left is smaller than or equal to right operand

Logical operators

C language supports following 3 logical operators. Suppose `a = 1` and `b = 0`,

Operator	Description	Example
&&	Logical AND	(<code>a && b</code>) is false
	Logical OR	(<code>a b</code>) is true

!	Logical NOT	(!a) is false
---	-------------	---------------

Bitwise operators

Bitwise operators perform manipulations of data at bit level. These operators also perform shifting of bits from right to left. Bitwise operators are not applied to float or double (These are datatypes, we will learn about them in the next tutorial).

Operator	Description
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR
<<	left shift
>>	right shift

Now lets see truth table for bitwise &, | and ^

a	B	a & b	a b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The bitwise shift operator, shifts the bit value. The left operand specifies the value to be shifted and the right operand specifies the number of positions that the bits in the value have to be shifted. Both operands have the same precedence.

Assignment Operators

Assignment operators supported by C language are as follows.

Operator	Description	Example
=	assigns values from right side operands to left side operand	a=b
+=	adds right operand to the left operand and assign the result to left	a+=b is same as a=a+b
-=	subtracts right operand from the left operand and assign the result to left operand	a-=b is same as a=a-b
=	multiply left operand with the right operand and assign the result to left operand	a=b is same as a=a*b
/=	divides left operand with the right operand and assign the result to left operand	a/=b is same as a=a/b
%=	calculate modulus using two operands and assign the result to left operand	a%=b is same as a=a%b

Conditional operator

The conditional operators in C language are known by two more names

1. Ternary Operator
2. ? : Operator

It is actually the if condition that we use in C language decision making, but using conditional operator, we turn the if condition statement into a short and simple operator.

The syntax of a conditional operator is :

```
expression 1 ? expression 2 : expression 3
```

Explanation:

- The question mark "?" in the syntax represents the if part.
- The first expression (expression 1) generally returns either true or false, based on which it is decided whether (expression 2) will be executed or (expression 3)
- If (expression 1) returns true then the expression on the left side of " : " i.e (expression 2) is executed.
- If (expression 1) returns false then the expression on the right side of " : " i.e (expression 3) is executed.

Special operator

Operator	Description	Example
sizeof	Returns the size of an variable	sizeof(x) return size of the variable x
&	Returns the address of an variable	&x ; return address of the variable x
*	Pointer to a variable	*x ; will be pointer to a variable x