C Operators

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise, etc.

There are following types of operators to perform different types of operations in C language.

- Arithmetic Operators
- Relational Operators
- Shift Operators
- Logical Operators
- Bitwise Operators
- Ternary or Conditional Operators
- Assignment Operator
- Misc Operator

Precedence of Operators in C

The precedence of operator species that which operator will be evaluated first and next. The associativity specifies the operator direction to be evaluated; it may be left to right or right to left.

Let's understand the precedence by the example given below:

```
1. int value=10+20*10;
```

The value variable will contain **210** because * (multiplicative operator) is evaluated before + (additive operator).

The precedence and associativity of C operators is given below:

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right

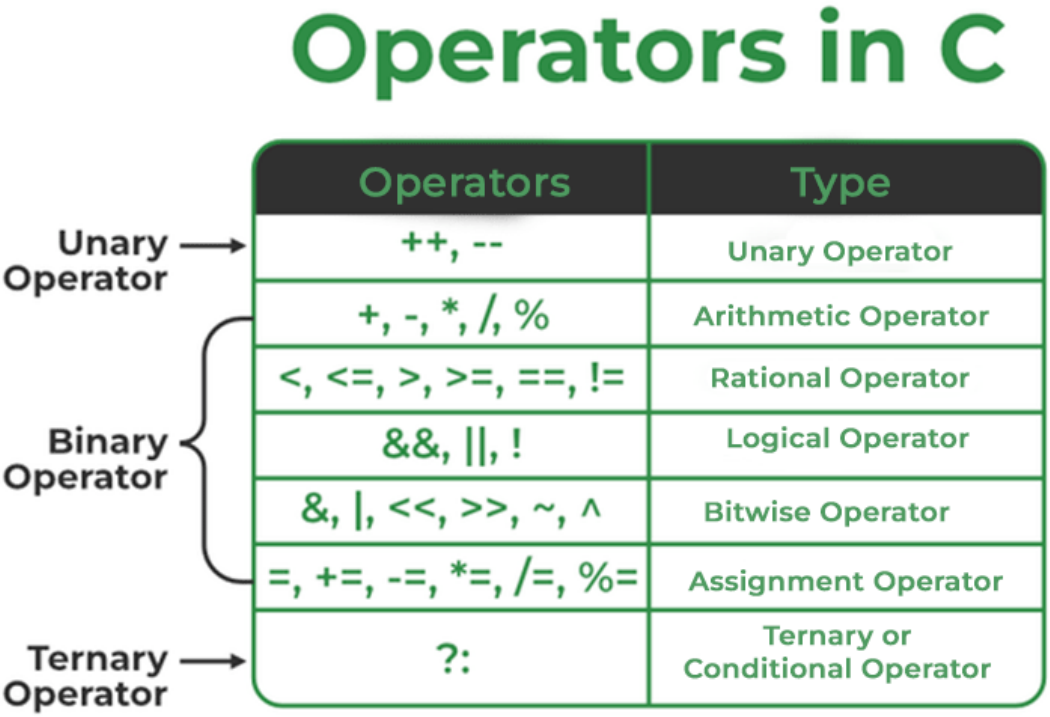
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Operators in C

In C language, operators are symbols that represent operations to be performed on one or more operands. They are the basic components of the C programming. In this article, we will learn about at all the built-in operators in C with examples.

What is a C Operator?

An operator in C can be defined as the symbol that helps us to perform some specific mathematical, relational, bitwise, conditional, or logical computations on values and variables. The values and variables used with operators are called operands. So we can say that the operators are the symbols that perform operations on operands.



For example,

```
c = a + b;
```

Here, '+' is the operator known as the addition operator, and 'a' and 'b' are operands. The addition operator tells the compiler to add both of the operands 'a' and 'b'.

Types of Operators in C

C language provides a wide range of operators that can be classified into 6 types based on their functionality:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators

5. Assignment Operators
6. Other Operators

1. Arithmetic Operations in C

The arithmetic operators are used to perform arithmetic/mathematical operations on operands. There are 9 arithmetic operators in C language:

S. No.	Symbol	perator	escription	Syntax
1	+	Plus	Adds two numeric values.	a + b
2	–	Minus	btracts right operand om left operand.	a – b
3	*	Multiply	ultiply two numeric lues.	a * b
4	/	Divide	vide two numeric lues.	a / b
5	%	Modulus	Returns the remainder after diving the left operand with the right operand.	a % b
6	+	Unary Plus	Used to specify the positive values.	+a
7	–	Unary Minus	Flips the sign of the value.	-a
8	++	Increment	Increases the value of the operand by 1.	a++
9	—	Decrement	Decreases the value of the operand by 1.	a–

Example of C Arithmetic Operators

C

```c
C program to illustrate the arithmatic operators
#include <stdio.h>

int main()
```

```c
int a = 25, b = 5;

// using operators and printing results
printf("a + b = %d\n", a + b);
printf("a - b = %d\n", a - b);
printf("a * b = %d\n", a * b);
printf("a / b = %d\n", a / b);
printf("a % b = %d\n", a % b);
printf("+a = %d\n", +a);
printf("-a = %d\n", -a);
printf("a++ = %d\n", a++);
printf("a-- = %d\n", a--);

return 0;
```

Output

```
a + b = 30
a - b = 20
a * b = 125
a / b = 5
a % b = 0
+a = 25
-a = -25
a++ = 25
a-- = 26
```

2. Relational Operators in C

The relational operators in C are used for the comparison of the two operands. All these operators are binary operators that return true or false values as the result of comparison.

These are a total of 6 relational operators in C:

S. No.	Symbol	Operator	Description	Syntax
1	<	Less than	Returns true if the left operand is less than the right operand. Else false	a < b
2	>	Greater than	Returns true if the left operand is greater than the right operand. Else false	a > b
3	<=	Less than or equal to	Returns true if the left operand is less than or equal to the right operand. Else false	a <= b

S. No.	Symbol	Operator	Description	Syntax
4	>=	Greater than or equal to	Returns true if the left operand is greater than or equal to right operand. Else false	a >= b
5	==	Equal to	Returns true if both the operands are equal.	a == b
6	!=	Not equal to	Returns true if both the operands are NOT equal.	a != b

Example of C Relational Operators

C

```c
C program to illustrate the relational operators
 #include <stdio.h>

 int main()


 int a = 25, b = 5;

 // using operators and printing results
 printf("a < b  : %d\n", a < b);
 printf("a > b  : %d\n", a > b);
 printf("a <= b: %d\n", a <= b);
 printf("a >= b: %d\n", a >= b);
 printf("a == b: %d\n", a == b);
 printf("a != b : %d\n", a != b);

 return 0;
```

Output

```
a < b  : 0
a > b  : 1
a <= b: 0
a >= b: 1
a == b: 0
a != b : 1
```

Here, 0 means false and 1 means true.

3. Logical Operator in C

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**.

S. No.	Symbol	perator	escription	Syntax
1	&&	gical AND	turns true if th the operands e true.	a && b
2	||	gical OR	turns true if th or any of the erand is true.	a || b
3	!	gical NOT	turns true if the erand is false.	!a

Example of Logical Operators in C

C++

```
C program to illustrate the logical operators
#include <stdio.h>

  int main()


 int a = 25, b = 5;

 // using operators and printing results
 printf("a && b : %d\n", a && b);
 printf("a || b : %d\n", a || b);
 printf("!a: %d\n", !a);

 return 0;
```

Output

```
a && b : 1
a || b : 1
!a: 0
```

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.

There are 6 bitwise operators in C:

S. No.	Symbol	Operator	escription	Syntax
1	&	Bitwise AND	Performs bit-by-bit AND operation and returns the result.	&& b

S. No.	Symbol	Operator	escription	Syntax
2	|	**Bitwise OR**	Performs bit-by-bit OR operation and returns the result.	**| | b**
3	^	**Bitwise XOR**	Performs bit-by-bit XOR operation and returns the result.	**^ b**
4	~	**Bitwise First Complement**	Flips all the set and unset bits on the number.	
5	<<	**Bitwise Leftshift**	Shifts the number in binary form by one place in the operation and returns the result.	**<< b**
6	>>	**Bitwise Rightshilft**	Shifts the number in binary form by one place in the operation and returns the result.	**>> b**

Example of Bitwise Operators

C

```c
C program to illustrate the bitwise operators
   #include <stdio.h>
    int main()

 int a = 25, b = 5;

 // using operators and printing results
 printf("a & b: %d\n", a & b);
 printf("a | b: %d\n", a | b);
 printf("a ^ b: %d\n", a ^ b);
 printf("~a: %d\n", ~a);
 printf("a >> b: %d\n", a >> b);
 printf("a << b: %d\n", a << b);

 return 0;
```

Output

```
a & b: 1
a | b: 29
a ^ b: 28
~a: -26
a >> b: 0
```

```
a << b: 800
```

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. The assignment operators can be combined with some other operators in C to provide multiple operations using single operator. These operators are called compound operators.

In C, there are 11 assignment operators :

S. No.	Symbol	Operator	Description	Syntax
1	=	**Simple Assignment**	Assign the value of the right operand to the left operand.	**a = b**
2	+=	**Plus and assign**	Add the right operand and left operand and assign this value to the left operand.	**a += b**
3	-=	**Minus and assign**	Subtract the right operand and left operand and assign this value to the left operand.	**a -= b**
4	*=	**Multiply and assign**	Multiply the right operand and left operand and assign this value to the left operand.	**a *= b**
5	/=	**Divide and assign**	Divide the left operand with the right operand and assign this value to the left operand.	**a /= b**
6	%=	**Modulus and assign**	Assign the remainder in the division of left operand with the right operand to the left operand.	**a %= b**
7	&=	**AND and assign**	Performs bitwise AND and assigns this value to the left operand.	**a &= b**
8	|=	**OR and assign**	Performs bitwise OR and assigns this value to the left operand.	**a |= b**
9	^=	**XOR and assign**	Performs bitwise XOR and assigns this value to the left operand.	**a ^= b**
10	>>=	**Rightshift and assign**	Performs bitwise Rightshift and assign this value to the left operand.	**a >>= b**

S. No.	Symbol	Operator	Description	Syntax
11	<<=	Leftshift and assign	Performs bitwise Leftshift and assign this value to the left operand.	a <<= b

Example of C Assignment Operators

C

```c
C program to illustrate the arithmatic operators
 #include <stdio.h>

 int main()


 int a = 25, b = 5;

 // using operators and printing results
 printf("a = b: %d\n", a = b);
 printf("a += b: %d\n", a += b);
 printf("a -= b: %d\n", a -= b);
 printf("a *= b: %d\n", a *= b);
 printf("a /= b: %d\n", a /= b);
 printf("a %= b: %d\n", a %= b);
 printf("a &= b: %d\n", a &= b);
 printf("a |= b: %d\n)", a |= b);
 printf("a >>= b: %d\n", a >> b);
 printf("a <<= b: %d\n", a << b);

 return 0;
```

Output

```
a = b: 5
a += b: 10
a -= b: 5
a *= b: 25
a /= b: 5
a %= b: 0
a &= b: 0
a |= b: 5
)a >>= b: 0
a <<= b: 160
```

6. Other Operators

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

sizeof Operator

- sizeof is much used in the 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 the unsigned integral type which is usually denoted by size_t.

- Basically, the sizeof the operator is used to compute the size of the variable or datatype.

Syntax

```
sizeof (operand)
```

To know more about the topic refer to this article.

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.

Syntax

```
operand1 , operand2
```

To know more about the topic refer to this article.

Conditional Operator (? :)

- The conditional operator is the only ternary operator in C++.
- Here, Expression1 is the condition to be evaluated. If the condition(Expression1) is *True* then we will execute and return the result of Expression2 otherwise 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 with conditional operators.

Syntax

```
operand1 ? operand2 : operand3;
```

To know more about the topic refer to this article.

dot (.) and arrow (->) Operators

- Member operators are used to reference individual members of classes, structures, and unions.
- The dot operator is applied to the actual object.
- The arrow operator is used with a pointer to an object.

Syntax

```
structure_variable . member;
```

and

```
structure_pointer -> member;
```

To know more about dot operators refer to this article and to know more about arrow(->) operators refer to this article.

Cast Operator

- Casting operators convert one data type to another. For example, int(2.2000) would return 2.
- A cast is a special operator that forces one data type to be converted into another.
- The most general cast supported by most of the C compilers is as follows – **[(type) expression]**.

Syntax

```
(new_type) operand;
```

To know more about the topic refer to this article.

addressof (&) and Dereference (*) Operators

- Pointer operator & returns the address of a variable. For example &a; will give the actual address of the variable.
- The pointer operator * is a pointer to a variable. For example *var; will pointer to a variable var.

To know more about the topic refer to this article.

Example of Other C Operators

C

```
 C Program to demonstrate the use of Misc operators
nclude <stdio.h>

t main()

  // integer variable
  int num = 10;
  int* add_of_num = #

  printf("sizeof(num) = %d bytes\n", sizeof(num));
  printf("&num = %p\n", &num);
  printf("*add_of_num = %d\n", *add_of_num);
  printf("(10 < 5) ? 10 : 20 = %d\n", (10 < 5) ? 10 : 20);
  printf("(float)num = %f\n", (float)num);

  return 0;
```

Output

```
sizeof(num) = 4 bytes
&num = 0x7ffe2b7bdf8c
*add_of_num = 10
(10 < 5) ? 10 : 20 = 20
(float)num = 10.000000
```

Unary, Binary and Ternary Operators in C

Operators can also be classified into three types on the basis of the number of operands they work on:

1. **Unary Operators:** Operators that work on single operand.
2. **Binary Operators:** Operators that work on two operands.
3. **Ternary Operators:** Operators that work on three operands.

Operator Precedence and Associativity in C

In C, it is very common for an expression or statement to have multiple operators and in these expression, there should be a fixed order or priority of operator evaluation to avoid ambiguity.

Operator Precedence and Associativity *is the concept that decides which operator will be evaluated first in the case when there are multiple operators present in an expression.*

The below table describes the precedence order and associativity of operators in C. The precedence of the operator decreases from top to bottom.

Precedence	Operator	Description	Associativity
1	()	Parentheses (function call)	left-to-right
	[]	Brackets (array subscript)	left-to-right

Precedence	Operator	Description	Associativity
	.	Member selection via object name	left-to-right
	->	Member selection via a pointer	left-to-right
	a++ , a–	Postfix increment/decrement (a is a variable)	left-to-right
2	**++a , –a**	Prefix increment/decrement (a is a variable)	right-to-left
	+ , –	Unary plus/minus	right-to-left
	! , ~	Logical negation/bitwise complement	right-to-left
	(type)	Cast (convert value to temporary value of type)	right-to-left
	*****	Dereference	right-to-left
	&	Address (of operand)	right-to-left
	sizeof	Determine size in bytes on this implementation	right-to-left
3	*** , / , %**	Multiplication/division/modulus	left-to-right
4	**+ , –**	Addition/subtraction	left-to-right
5	**<< , >>**	Bitwise shift left, Bitwise shift right	left-to-right
6	**< , <=**	Relational less than/less than or equal to	left-to-right
	> , >=	Relational greater than/greater than or equal to	left-to-right
7	**== , !=**	Relational is equal to/is not equal to	left-to-right
8	**&**	Bitwise AND	left-to-right

Precedence	Operator	Description	Associativity
9	^	Bitwise exclusive OR	left-to-right
10	|	Bitwise inclusive OR	left-to-right
11	**&&**	Logical AND	left-to-right
12	||	Logical OR	left-to-right
13	**?:**	Ternary conditional	right-to-left
14	=	Assignment	right-to-left
	+= , -=	Addition/subtraction assignment	right-to-left
	*= , /=	Multiplication/division assignment	right-to-left
	%= , &=	Modulus/bitwise AND assignment	right-to-left
	^= , |=	Bitwise exclusive/inclusive OR assignment	right-to-left
	<<=, >>=	Bitwise shift left/right assignment	right-to-left
15	,	expression separator	left-to-right

Conclusion

In this article, the points we learned about the operator are as follows:

- Operators are symbols used for performing some kind of operation in C.
- There are six types of operators, Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, Assignment Operators, and Miscellaneous Operators.
- Operators can also be of type unary, binary, and ternary according to the number of operators they are using.
- Every operator returns a numerical value except logical, relational, and conditional operator which returns a boolean value (true or false).
- There is a Precedence in the operators means the priority of using one operator is greater than another operator.

FAQs on C Operators

Q1. What are operators in C?

Answer:

Operators in C are certain symbols in C used for performing certain mathematical, relational, bitwise, conditional, or logical operations for the user.

Q2. What are the 7 types of operators in C?

Answer:

There are 7 types of operators in C as mentioned below:

- *Unary operator*
- *Arithmetic operator*
- *Relational operator*
- *Logical operator*
- *Bitwise operator*
- *Assignment operator*
- *Conditional operator*

Q3. What is the difference between the '=' and '==' operators?

Answer:

'=' is a type of assignment operator that places the value in right to the variable on left, Whereas '==' is a type of relational operator that is used to compare two elements if the elements are equal or not.

Q4. What is the difference between prefix and postfix operators in C?

Answer:

Prefix operations are the operations in which the value is returned prior to the operation whereas in postfix operations value is returned after updating the value in the variable.

Example:

```
b=c=10;
a=b++;      // a==10
a=++c;      // a==11
```

Q5. What is the Modulo operator?

Answer:

The Modulo operator(%) is used to find the remainder if one element is divided by another.

Example:

```
a % b (a divided by b)
5 % 2 == 1
```