Operators in C

Last Updated : 13 May, 2025

In C language, operators are symbols that represent some kind of operations to be performed. They are the basic components of the C programming. In this article, we will learn about all the operators in C with examples.

What is an Operator in C?

A **C operator** 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
#include <stdio.h>
int main() {

    // Expression for getting sum
    int sum = 10 + 20;

    printf("%d", sum);
    return 0;
}
```

Output

```
30
```

In the above expression, '+' is the **addition operator** that tells the how operators are used with data structures, the **C Programming Course Online with Data Structures** covers this topic thoroughly.

Types of Operators in C

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

Table of Content

1. Arithmetic Operators

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

S. No.	Symbol	Operator	Description	Syntax
1	+	**Plus**	Adds two numeric values.	**a + b**
2	-	**Minus**	Subtracts right operand from left operand.	**a - b**
3	*	**Multiply**	Multiply two numeric	**a * b**

S. No.	Symbol	Operator	Description	Syntax
			values.	
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

```
#include <stdio.h>

int main() {

    int a = 25, b = 5;
    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

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	a < b

S. No.	Symbol	Operator	Description	Syntax
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**
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**

```c
#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

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**.

There are 3 logical operators in C:

S. No.	Symbol	Operator	Description	Syntax

S. No.	Symbol	Operator	Description	Syntax
			operands are true.	
2	||	**Logical OR**	Returns true if both or any of the operand is true.	**a || b**
3	!	**Logical NOT**	Returns true if the operand is false.	**!a**

Example of Logical Operators in C

```c
#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
```

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	Description	Syntax
1	&	**Bitwise AND**	Performs bit-by-bit AND operation and returns the result.	**a & b**
2	|	**Bitwise OR**	Performs bit-by-bit OR operation and returns the result.	**a | b**
3	^	**Bitwise XOR**	Performs bit-by-bit XOR operation and returns the result.	**a ^ b**
4	~	**Bitwise First Complement**	Flips all the set and unset bits on the number.	**~a**
5	<<	**Bitwise Leftshift**	Shifts the number in binary form by one place in	**a << b**

S. No.	Symbol	Operator	Description	Syntax
6	>>	Bitwise Rightshilft	Shifts the number in binary form by one place in the operation and returns the result.	a >> b

Example of 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
```

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**

S. No.	Symbol	Operator	Description	Syntax
			operand and assign this value to the left operand.	
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	**a |= b**

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

Example of C Assignment 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);
```

```c
    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: 0
a <<= b: 0
```

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.

```
sizeof (operand)
```

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. It can act as both operator and separator.

Syntax

```
operand1 , operand2
```

Conditional Operator (? :)

The **conditional operator** is the only ternary operator in C++. It is a conditional operator that we can use in place of if..else statements.

Syntax

```
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** otherwise if the condition(**Expression1**) is ***false*** then we will execute and return the result of **Expression3**.

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.

```
structure_variable . member;
structure_pointer -> member;
```

Cast Operators

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.

Syntax

```
(new_type) operand;
```

addressof (&) and Dereference (*) Operators

Addressof operator & returns the address of a variable and the dereference operator * is a pointer to a variable. For example *var; will pointer to a variable var.

Example of Other C Operators

```
// C Program to demonstrate the use of Misc operators
#include <stdio.h>

int main()
{
    // integer variable
    int num = 10;
    int* add_of_num = &num;

    printf("sizeof(num) = %d bytes\n", sizeof(num));
    printf("&num = %p\n", &num);
    printf("*add_of_num = %d\n", *add_of_num);
```

```
        return 0;
    }
```

Output

```
sizeof(num) = 4 bytes
&num = 0x7ffdb58c037c
*add_of_num = 10
(10 < 5) ? 10 : 20 = 20
(float)num = 10.000000
```

Unary, Binary and Ternary Operators

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, it is very common for an expression or statement to have multiple operators and in this 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			

Precedence	Operator	Description	Associativity
	[]	Brackets (array subscript)	left-to-right
	.	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			

Precedence	Operator	Description	Associativity
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
9	^	Bitwise XOR	left-to-right
10	|	Bitwise 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

Precedence	Operator	Description	Associativity
	*= , /=	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

Comment More info

Campus Training Program

Next Article

Arithmetic Operators in C

Similar Reads

# and ## Operators in C

In C, # and ## operators are preprocessor operators using in macros for token manipulation. They are known as stringizing and token pasting...

15 min read

dot (.) Operator in C

In C, the dot (.) operator is used to access members of user defined data types such as a structure or union. Also known as the direct member...

11 min read

Unary Operators in C

In C programming, unary operators are operators that operate on a single operand. These operators are used to perform operations such as...

15+ min read

sizeof operator in C

Sizeof is a much-used operator in the C. It is a compile-time unary operator which can be used to compute the size of its operand. The resul...

15+ min read

C Logical Operators

Logical operators in C are used to combine multiple conditions/constraints. Logical Operators returns either 0 or 1, it depends on whether the...

15+ min read

Bitwise Operators in C

In C, the following 6 operators are bitwise operators (also known as bit operators as they work at the bit-level). They are used to perform bitwis...

15+ min read

15+ min read

Relational Operators in C

In C, relational operators are the symbols that are used for comparison between two values to understand the type of relationship a pair of...

15+ min read

Assignment Operators in C

In C, assignment operators are used to assign values to variables. The left operand is the variable and the right operand is the value being assigned...

15+ min read

Conversion Operators in C++

In C++, the programmer abstracts real-world objects using classes as concrete types. Sometimes, it is required to convert one concrete type to...

15+ min read

Company
About Us
Legal
Privacy Policy
Careers
In Media
Contact Us
GfG Corporate Solution
Placement Training Program

Explore
Job-A-Thon Hiring Challenge
GfG Weekly Contest
Offline Classroom Program
DSA in JAVA/C++
Master System Design
Master CP
GeeksforGeeks Videos

Languages
Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial

DSA
Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
DSA Interview Questions
Competitive Programming

Data Science & ML
Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths

Web Technologies
HTML
CSS
JavaScript
TypeScript

Deep Learning

Tailwind CSS

Python Tutorial

Python Programming Examples
Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar

Databases

SQL
MYSQL
PostgreSQL
PL/SQL
MongoDB

Preparation Corner

Company-Wise Recruitment Process
Aptitude Preparation
Puzzles
Company-Wise Preparation

More Tutorials

Software Development
Software Testing
Product Management
Project Management
Linux
Excel
All Cheat Sheets

Machine Learning/Data Science

Complete Machine Learning & Data Science Program - [LIVE]
Data Analytics Training using Excel, SQL, Python & PowerBI - [LIVE]
Data Science Training Program - [LIVE]
Data Science Course with IBM Certification

Programming Languages

C Programming with Data Structures
C++ Programming Course
Java Programming Course
Python Full Course

Clouds/Devops

GATE 2026

GATE Rank Predictor