Operators in C

C Operators are symbols that represent operations to be performed on one or more operands. C provides a wide range of operators, which can be classified into different categories based on their functionality. Operators are used for performing operations on variables and values.

What are Operators in C?

Operators can be defined as the symbols that help us to perform specific mathematical, relational, bitwise, conditional, or logical computations on operands. In other words, we can say that an operator operates the operands. For example, '+' is an operator used for addition, as shown below:

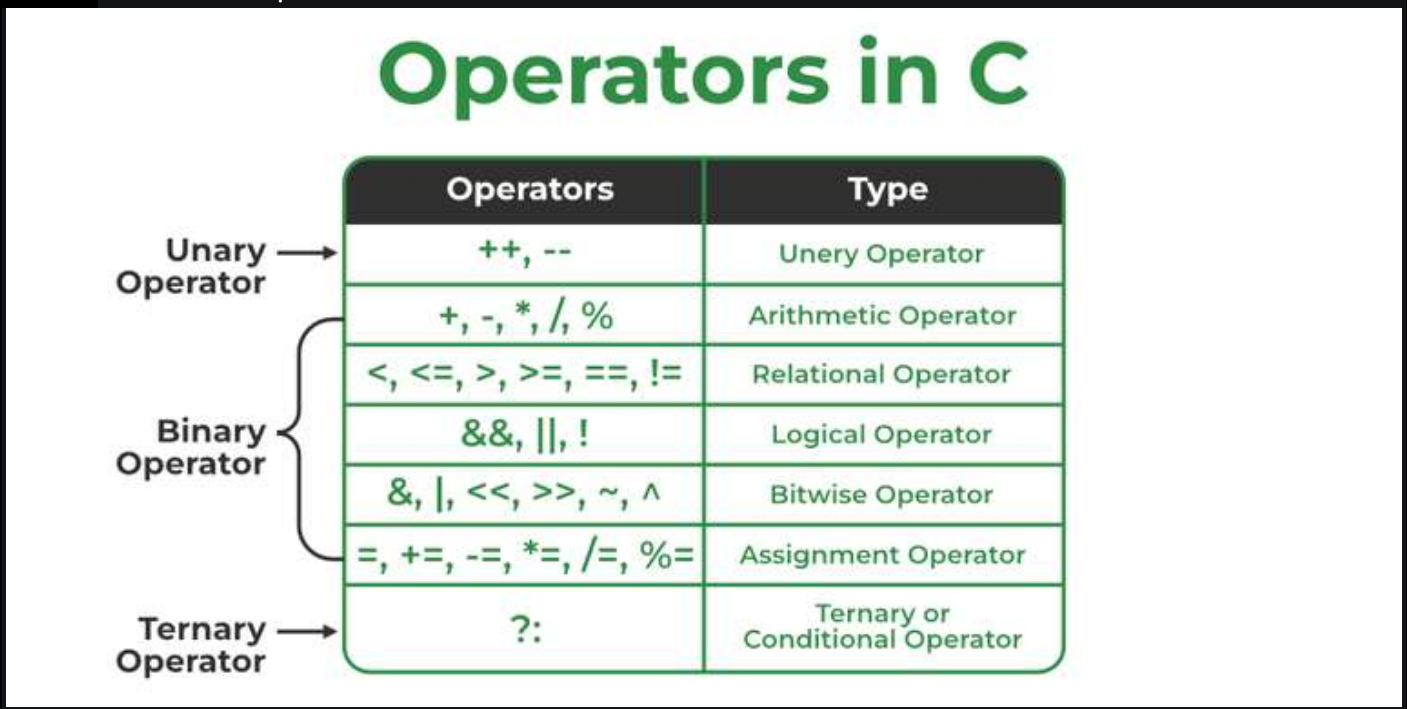
```
c = a + b;
```

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'. The functionality of the C programming language is incomplete without the use of operators.

Types of Operators in C

C has many built-in operators and can be classified into 6 types:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Other Operators



Operators in C

The above operators have been discussed in detail:

1. Arithmetic Operations in C

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

a) Unary Operators:

Operators that operate or work with a single operand are unary operators. For example: Increment(++) and Decrement(–) Operators

```
int val = 5;
cout<<++val;  // 6
```

b) Binary Operators:

Operators that operate or work with two operands are binary operators. For example: Addition(+), Subtraction(-), multiplication(*), Division(/) operators

```
int a = 7;
int b = 2;
cout<<a+b; // 9
```

2. Relational Operators in C

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

```
int a = 3;
int b = 5;
cout<<(a < b);
// operator to check if a is smaller than b
```

3. Logical Operator in C

Logical Operators are used to combining 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** represented as the **'&&' operator in C** 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)(See this article for more reference).

```
cout<<((4 != 5) && (4 < 5));     // true
```

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. For example, the **bitwise AND** operator represented as **'&' in 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(True).

```
int a = 5, b = 9;   // a = 5(00000101), b = 9(00001001)

cout << (a ^ b);   //  00001100

cout <<(~a);       // 11111010
```

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.

Different types of assignment operators are shown below:

a) "="

This is the simplest assignment operator. This operator is used to assign the value on the right to the variable on the left.

Example:

```
a = 10;

b = 20;

ch = 'y';
```

b) "+="

This operator is the combination of the '+' and '=' operators. This operator first adds the current value of the variable on left to the value on the 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. Then (a += 6) = 11.
```

c) "-="

This operator is a combination of '-' and '=' operators. This operator first subtracts the value on the 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. Then (a -= 6) = 2.
```

d) "*="

This operator is a combination of the '*' and '=' operators. This operator first multiplies the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.

Example:

```
(a *= b) can be written as (a = a * b)
```

```
If initially, the value stored in a is 5. Then (a *= 6) = 30.
```

e) “/=”

This operator is a combination of the ‘/’ and ‘=’ operators. This operator first divides the current value of the variable on left by the value on the right and then assigns the result to the variable on the left.

Example:

```
(a /= b) can be written as (a = a / b)

If initially, the value stored in a is 6. Then (a /= 2) = 3.
```

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:

i. 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.

To know more about the topic refer to this article.

ii. 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.

To know more about the topic refer to this article.

iii. Conditional Operator

- The 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 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.

To know more about the topic refer to this article.

iv. dot (.) and arrow (->) Operators

- Member operators are used to referencing 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.

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

v. 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]**.

To know more about the topic refer to this article.

vi. &,* Operator

- 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.

C Operators with Example

- C

```c
// C Program to Demonstrate the working concept of
// Operators
#include <stdio.h>

int main()
{

    int a = 10, b = 5;
    // Arithmetic operators
    printf("Following are the Arithmetic operators in C\n");
    printf("The value of a + b is %d\n", a + b);
    printf("The value of a - b is %d\n", a - b);

    printf("The value of a * b is %d\n", a * b);
    printf("The value of a / b is %d\n", a / b);
    printf("The value of a % b is %d\n", a % b);
    // First print (a) and then increment it
    // by 1
```

```
    printf("The value of a++ is %d\n", a++);

    // First print (a+1) and then decrease it
    // by 1
    printf("The value of a-- is %d\n", a--);

    // Increment (a) by (a+1) and then print
    printf("The value of ++a is %d\n", ++a);

    // Decrement (a+1) by (a) and then print
    printf("The value of --a is %d\n", --a);

    // Assignment Operators --> used to assign values to
    // variables int a =3, b=9; char d='d';

    // Comparison operators
    // Output of all these comparison operators will be (1)
    // if it is true and (0) if it is false
    printf(
        "\nFollowing are the comparison operators in C\n");
    printf("The value of a == b is %d\n", (a == b));
    printf("The value of a != b is %d\n", (a != b));
    printf("The value of a >= b is %d\n", (a >= b));
    printf("The value of a <= b is %d\n", (a <= b));
    printf("The value of a > b is %d\n", (a > b));
    printf("The value of a < b is %d\n", (a < b));

    // Logical operators
    printf("\nFollowing are the logical operators in C\n");
    printf("The value of this logical and operator ((a==b) "
           "&& (a<b)) is:%d\n",
           ((a == b) && (a < b)));
    printf("The value of this logical or operator ((a==b) "
```

```
            "|| (a<b)) is:%d\n",
            ((a == b) || (a < b)));
    printf("The value of this logical not operator "
            "(!(a==b)) is:%d\n",
            (!(a == b)));

    return 0;
}
```

Output

```
Following are the Arithmetic operators in C
The value of a + b is 15
The value of a - b is 5
The value of a * b is 50
The value of a / b is 2
The value of a % b is 0
The value of a++ is 10
The value of a-- is 11
The value of ++a is 11
The value of --a is 10

Following are the comparison operators in C
The value of a == b is 0
The value of a != b is 1
The value of a >= b is 1
The value of a <= b is 0
The value of a > b is 1
The value of a < b is 0

Following are the logical operators in C
The value of this logical and operator ((a==b) && (a<b)) is:0
The value of this logical or operator ((a==b) || (a<b)) is:0
The value of this logical not operator (!(a==b)) is:1
```

Time and Space Complexity

Time Complexity: O(1)

Auxiliary Space: O(1)

Precedence of Operators in C

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
	.	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

Precedence	Operator	Description	Associativity
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 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.*
- *The operation can be mathematical, logical, relational, bitwise, conditional, or logical.*
- *There are seven types of Unary operators, Arithmetic operator, Relational operator, Logical operator, Bitwise operator, Assignment operator, and Conditional operator.*
- *Every operator returns a numerical value except logical and conditional operator which returns a boolean value(true or false).*
- *'=' and '==' are not same as '=' assigns the value whereas '==' checks if both the values are equal or not.*
- *There is a Precedence in the operator means the priority of using one operator is greater than another operator.*

Frequently Asked Questions(FAQs)

1. What are operators in C?

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

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

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

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

'=' 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.

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

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
```

5. What is the Modulo operator?

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
```