

Operators in C

Gaurav Kr. suman 4/26/20 MAT08



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.

Operators

Ternary or Conditional

Operators

? :

Special

Unary Operators

++ increment

-- Decrement

- Unary minus

+ Unary plus

\* dereferencing & addressof

Binary Operators

Arithmetic Operators

**+, -, \*, /, %**

Relational Operators

**==, !=, >, <, >=, <=**

****Operator

sizeof operator

(type) or cast operator

Logical Operators

&&, ||, !

Assignment Operators

**=, +=, -=, \*=, /=, %= <<=, >>=, &=, ^=, |=**

Bitwise Operators

**<<, >>, ~, &, ^, |**

**1 |** P a g e

**C operators can be classified into following types:**

• Arithmetic operators

• Relational operators

• Logical operators

• Bitwise operators

• Assignment operators

• Conditional operators

• Special operators

The following table shows all the arithmetic operators supported by the C language. Assume variable A holds 10 and variable B holds 20 then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| **+** | Adds two operands. | A + B = 30 |
| **−** | Subtracts second operand from the first. | A − B = -10 |
| **\*** | Multiplies both operands. | A \* B = 200 |
| **/** | Divides numerator by de-numerator. | B / A = 2 |
| **%** | Modulus Operator and remainder of after an integer division. | B % A = 0 |

The following table shows all the relational operators supported by C. Assume variable A holds 10 and variable B holds 20 then –

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| **==** | Checks if the values of two operands are equal or not. If yes, then the condition becomes true. | (A == B) is not  true. |
| **!=** | Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true. | (A != B) is true. |
| **>** | Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true. | (A > B) is not true. |
| **<** | Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true. | (A < B) is true. |

**2 |** P a g e

|  |  |  |
| --- | --- | --- |
| **>=** | Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true. | (A >= B) is not  true. |
| **<=** | Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true. | (A <= B) is true. |

Following table shows all the logical operators supported by C language. Assume variable A holds 1 and variable B holds 0, then –

**Operator Description Example**

|  |  |  |
| --- | --- | --- |
| **&&** | Logical AND | (a && b) is false |
| **||** | Logical OR | (a || b) is true |
| **!** | Logical NOT | (!a) is false |

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 |

**3 |** P a g e

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.

Example

a = 0001000

b = 2

a << b = 0100000

a >> b = 0000010

|  |  |  |
| --- | --- | --- |
| **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 |

**4 |** P a g e

|  |  |  |
| --- | --- | --- |
| **\*=** | mutiply 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 |

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

**Write a program in C Find number is positive or negative**

#include<stdio.h>

void main ()

{

int num;

printf ("Enter a number: ");

scanf ("%d", &num);

(num>=0)? printf("Positive."): printf("Negative");

}

**5 |** P a g e

**Output:**

Enter a number: 8

Positive.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| **size of** | Returns the size of an variable | **size of(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** |

**6 |** P a g e