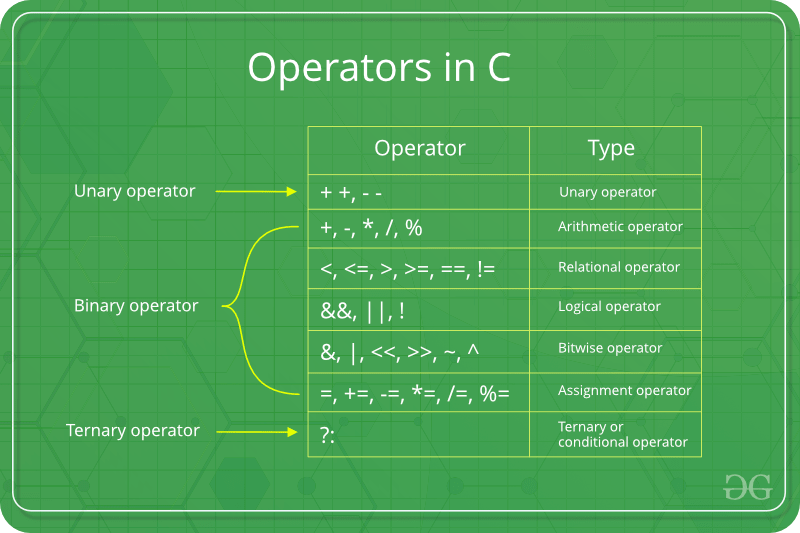
**Operators in C**



1. Arithmetic Operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | Name | Use / Purpose | Result | Example(s) |
| + | Addition | To add two operands | Sum | 10+20, a+b |
| - | Subtraction | To subtract the right operand from the left operand | Difference | 10-20, a-b |
| \* | Multiplication | To multiply two operands | Product | 10\*20, a\*b |
| / | Division | To find the quotient of a division operation | Quotient | 10/20, a/b |
| % | Modulo Division | To find the remainder of a division operation | Remainder | 10%20, a%b |

1. Relational or Comparison Operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | Name | Use / Purpose | Result | Example(s) |
| > | Greater than | Checks if left operand is greater than right operand, evaluates TRUE if so, evaluates FALSE otherwise. | A Boolean value (TRUE or FALSE) | 10>20, a>b |
| < | Less than | Checks if left operand is less than right operand, evaluates TRUE if so, evaluates FALSE otherwise. | A Boolean value (TRUE or FALSE) | 10<20, a<b |
| >= | Greater than or equals to | Checks if left operand is greater than or equals to the right operand, evaluates TRUE if so, evaluates FALSE otherwise. | A Boolean value (TRUE or FALSE) | 10>=20, a>=b |
| <= | Less than or equals to | Checks if left operand is less than or equals to the right operand, evaluates TRUE if so, evaluates FALSE otherwise. | A Boolean value (TRUE or FALSE) | 10<=20, a<=b |
| == | Equals to | Checks if left operand is equals to right operand, evaluates TRUE if so, evaluates FALSE otherwise | A Boolean value (TRUE or FALSE) | 10==20, a==b |
| != | Not equals to | Checks if left operand is not equals to right operand, evaluates TRUE if so, evaluates FALSE otherwise | A Boolean value (TRUE or FALSE) | 10!=20, a!=b |

1. Logical Operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | Name | Use / Purpose | Result | Example(s) |
| && | Logical AND | Evaluates to TRUE if both left and right operands are TRUE, evaluates to FALSE otherwise (Mostly used between two relational operations) | A Boolean value (TRUE or FALSE) | (10<20)&&(20>10),  (a>b) && (b<a) |
| || | Logical OR | Evaluates to FALSE if both left and right operands are FALSE, evaluates to TRUE otherwise (Mostly used between to relational operations) | A Boolean value (TRUE or FALSE) | (10<20)||(20<10),  (a<b)||(b>a) |
| ! | Logical NOT | Evaluates FALSE if the expression is TRUE, evaluates TRUE if the expression is FALSE. We can call this as  INVERSE TRUTH) | A Boolean value (TRUE or FALSE) | !(10>20), !(a<b) |

1. Assignment Operators

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Name | Use / Purpose | Example(s) |
| = | Equals | Assigns the value (right operand) to a variable (left operand) | Age = 20,  number = 67 |
| += | Add and Assign | Adds the value of right operand to the left operand and assigns the result to left operand | Age+=10,  number+=20 |
| -= | Subtract and assign | Subtracts the value of right operand from the left operand and assigns the result to left operand | Age-=5  number-=14 |
| \*= | Multiply and assign | Multiplies left and right operands and assigns the result to the left operand | Age\*=2  number\*=3 |
| /= | Divide and assign | Divides left operand with right operand and assigns the quotient to left operand | Age/=3  number/=4 |
| %= | Modulo divide and assign | Divides left operand with right operand and assigns the remainder to left operand | Age%=2  number%=5 |

1. Bitwise Operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | Name | Use / Purpose | Result | Example(s) |
| & | Bitwise AND | Compares each bit of the first operand to the corresponding bit of the second operand. If both bits are 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to 0 | Corresponding Decimal value of resultant Binary value |  |
| | | Bitwise OR | Compares each bit of the first operand to the corresponding bit of the second operand. If either of the bits is 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to 0 | Corresponding Decimal value of resultant Binary value |  |
| ^ | Bitwise XOR | Compares each bit of the first operand to the corresponding bit of the second operand. If one bit is 0 and the other bit is 1 or vice versa, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to 0 | Corresponding Decimal value of resultant Binary value |  |
| ~ | Bitwise NOT |  |  |  |
| << | Left-shift | Bitwise right shift operator , it takes two numbers, left shifts the bits of the first operand, and the second operand decides the number of places to shift | Corresponding Decimal value of resultant Binary value |  |
| >> | Right-shift | Bitwise right shift operator, it takes two numbers, right shifts the bits of the first operand, and the second operand decides the number of places to shift. | Corresponding Decimal value of resultant Binary value |  |

**Binary Representation of Decimal values:**

C Programming

Base 10 --> Decimal

Base 2 --> Binary

Base 8 --> Octal

Base 16 --> Hexa Decimal

Decimal:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Binary:

0, 1

**Decimal to Binary Conversion:**

Decimal – 13 How to represent in Binary form?

**Step 1**: Keep on dividing the given decimal value with 2, until you get quotient as 0, but keep the track of the remainder of every division operation.

**Step 2:** Write all the remainders from bottom to top (backwards) to get the binary representation of given decimal value.

**Ex 1:**

**Decimal Binary**

(13)10 --> **(1101)2**

13/2 -> 1

6/2 -> 0

3/2 -> 1

1/2 -> 1

**(1101)2**

**Ex 2:**

**Decimal Binary**

(29)10 **(11101)2**

29/2 -> 1

14/2 -> 0

7/2 -> 1

3/2 -> 1

1/2 -> 1

**(11101)2**

|  |  |
| --- | --- |
| **Decimal (d)10** | **Binary (b)2** |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |
| 13 | 1101 |
| 14 | 1110 |
| 15 | 1111 |
| 16 | 10000 |
| 17 | 10001 |
| 18 | 10010 |
| 19 | 10011 |
| 20 | 10100 |
| 21 | 10101 |
| 22 | 10110 |
| 23 | 10111 |
| 24 | 11000 |
| 25 | 11001 |
| 26 | 11010 |
| 27 | 11011 |
| 28 | 11100 |
| 29 | 11101 |
| 30 | 11110 |

**Binary to Decimal:**

(11001)2 --> 1 x 24 + 1 x 23 + 0 x 22 + 0 x 21 + 1 x 20

16 + 8 + 0 + 0 + 1 = **(25)10**

(1011011001)2 --> No. of digits (n)

dn-1 x 2n-1 + … + d3 x 23 + d2 x 22 + d1 x 21 + d0x 20

512 + 128 + 64 + 16 + 8 + 1 --> 729

Bitwise Operations:

Ex:

& | ^

& - Bitwise AND (&) compares every single bit of the first operand with the corresponding bit of the second operand. If both bits are 1 it will set the resultant bit to 1, otherwise 0.

**1 & 2 --> 0**

**0 1**

**1 0**

**0 0**

**Ans: 0**

| - Bitwise OR (|) compares every single bit of the first operand with the corresponding bit of the second operand. If either of the bits is 1 it will set the resultant bit to 1, otherwise 0.

1 | 2 --> 3

0 1

1 0

1 1

**(11)2 --> (3)10**

^ - Bitwise XOR (^) compares every single bit of the first operand with the corresponding bit of the second operand. If both one bit is 1 and the other is 0 or vice versa it will set the resultant bit to 1, otherwise 0.

1 ^ 2 --> 3

0 1

1 0

1 1

**(11)2 --> (3)10**

Ex:

22 --> 10110

18 --> 10010

& - 18

| - 22

^ - 4.

1. Unary Operators (Increment / Decrement Operator)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | Name | Use / Purpose | Result | Example(s) |
| ++ | Unary  + or Increment Operator | Increments the value of the operand by 1 | Operand + 1 | 10++, a++, 20++ |
| -- | Unary – or Decrement Operator | Decrements the value of the operand by 1 | Operand - 1 | 10--, a--, 20-- |