**Operators:**

Operators are symbols or combinations of symbols that directs the computer to perform some operation upon operands. C includes large number of operators, which fall into several different categories.

* Arithmetic Operators
* Assignment Operators
* Unary Operators
* Comparison (Relational) Operators
* Shift Operators
* Bit-wise Operators
* Logical Operators
* Miscellaneous Operators

**Arithmetic Operators**

There are five arithmetic Operators in C. They are:

Suppose variable a holds 10 and b holds 20

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Description | Expression | Value |
| + | 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 (remainder after an integer division) | b%a | 0 |

Suppose X1 and X2 are character-type variables that represent the character M and U respectively. Some arithmetic expressions that make use of these variables are shown below.

X1 + X2 =162

X1+ X2+ ‘5’ = 215

Note that M is encoded as 77 , U is encoded as 85, and 5 is encoded as 53 in ASCII character set.

**Assignment Operator:**

Assignment operator assigns the value of right operator or expression to the variable in left side. There are many variations of assignment operator as described below;

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| = | Assigns values from right side operands to left side operand. | C = A + B will assign value of A + B into C |
| += | Adds the operands and assigns the result to the left operand. | C += A is equivalent to C = C + A |
| -= | Subtracts right operand from the left operand and assign the result to left operand. | C -= A is equivalent to C = C - A |
| \*= | Multiplies right operand with the left operand and assign the result to left operand. | C \*= A is equivalent to C = C \* A |
| /= | Divides left operand with the right operand and assign the result to left operand. | C /= A is equivalent to C = C / A |
| %= | Takes modulus using two operands and assign the result to left operand. | C %= A is equivalent to C = C % A |

**Unary Operators:**

A unary operator in C, is an operator that takes a single operand in an expression or a statement. There are basically two unary operators in C. They are:

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Description | Example | Explanation |
| ++ | Increases the value of the operand by one. | a++ | Equivalent to a=a+1 |
| -- | Decreases the value of the operand by one. | a-- | Equivalent to a=a-1 |

The increment operator ++, can be used in two ways:

* As a prefix

In prefix, the operator precedes the variable i.e. ++var . In this form the value of the variable is first incremented and then used in the expression as illustrated below;

var1=20; var2=++var1;

This code is equivalent to the following set of codes:

var1=20; var1=var1+1; var2=var1;

At the end, both variables var1 and var2 store value 21.

* As a postfix

Likewise, in postfix, the operator follows the variable i.e. var++. In this form, the value of variable is used in the expression and then incremented as illustrated below;

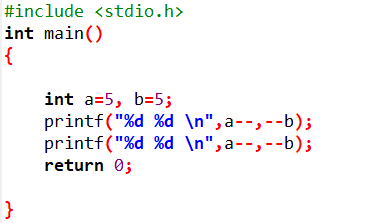
var1=20; var2=var1++;

The equivalent of this code is:

var1=20; var2=var1; var1=var1+1;

At the end, var1 has the value 21 while var2 reminds set to 20.

Example: Post decrement and Pre decrement

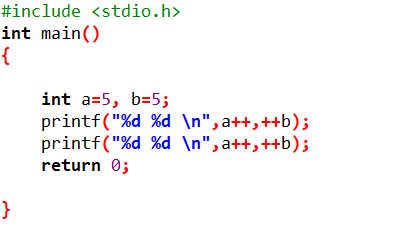


Output:

5 4

4 3

Example: Post Increment and Pre Increment



Output:

5 6

6 7

**Comparison Operator**

Comparison operators evaluate to true or false. They are also called relational operators.

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

**Shift Operators**

Data is stored internally in binary format. A bit can have a value of one or zero. Eight bits form a byte. Shift operators work on individual bits in a byte. Using the shift operator involves moving the bit pattern left or right. We can use them only on integer data type and not on the char , float , or double data types.

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Description | Example | Explanation |
| >> | Shifts bits to the right, filling sign bir at the left. | a=10>>3 | The result of this is 10 divided by 2^3. |
| << | Shifts bits to the left, filling zeros at the right. | a=10<<3 | The result if this is 10 multiplied by 2^3. |

If the int data type occupies four byte in the memory, the rightmost eight bits of the number are represented in binary as “00001010”

When we do right shift by 3, the result is “00000001” which is equivalent to 1 and when we do left shift by 3, the result is “01010000” which is equivalent to 80.

**Bit-wise operators**

Bitwise operator works on bits and perform bit-by-bit operation. The truth tables for &, |, and ^ are as follows −

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **p** | **q** | **p & q** | **p | q** | **p ^ q** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Assume if A = 60; and B = 13; now in binary format they will be as follows −

A = 0011 1100

B = 0000 1101

-----------------------------

Binary And Operator:

A&B = 0000 1100

Binary Or Operator:

A|B = 0011 1101

Binary XOR Operator:

A^B = 0011 0001

Binary Ones Complement Operator:

~A = 1100 0011

## **Logical Operators**

There are following logical operators supported by C language.

Assume variable A holds 1 and variable B holds 0, then

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| && | Called Logical AND operator. If both the operands are non-zero, then condition becomes true. | (A && B) is false. |
| || | Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true. | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false. | !(A && B) is true. |

## **Miscellaneous Operators**

The following table lists some other operators that C supports.

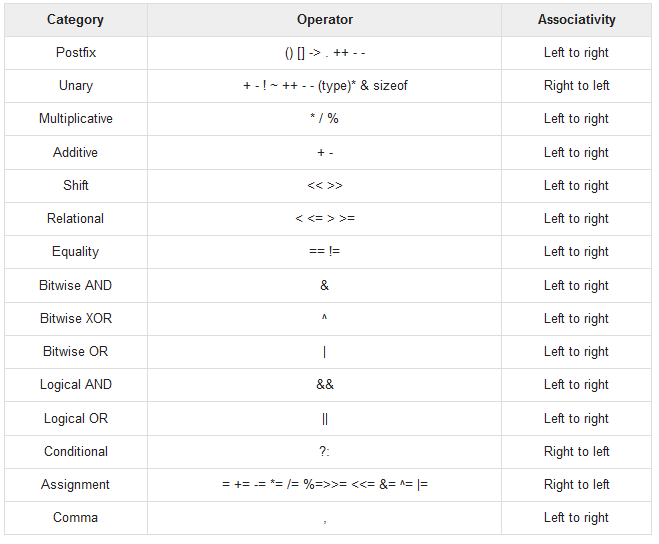
|  |  |
| --- | --- |
| Sr.No | Operator & Description |
| 1 | sizeof  [**sizeof operator**](https://www.tutorialspoint.com/cplusplus/cpp_sizeof_operator.htm) returns the size of a variable. For example, sizeof(a), where ‘a’ is integer, and will return 4. |
| 2 | Condition ? X : Y  [**Conditional operator (?)**](https://www.tutorialspoint.com/cplusplus/cpp_conditional_operator.htm). If Condition is true then it returns value of X otherwise returns value of Y.  Example:  result: (marks>50 ? “pass”: “fail”) ;  In above example, if the marks obtained is more than 50, result will be pass else fail. |
| 3 | ,  [**Comma operator**](https://www.tutorialspoint.com/cplusplus/cpp_comma_operator.htm) causes a sequence of operations to be performed. The value of the entire comma expression is the value of the last expression of the comma-separated list. |
| 4 | . (dot) and -> (arrow)  [**Member operators**](https://www.tutorialspoint.com/cplusplus/cpp_member_operators.htm) are used to reference individual members of classes, structures, and unions. |
| 5 | Cast  [**Casting operators**](https://www.tutorialspoint.com/cplusplus/cpp_casting_operators.htm) convert one data type to another. For example, int(2.2000) would return 2. |
| 6 | &  [**Pointer operator &**](https://www.tutorialspoint.com/cplusplus/cpp_pointer_operators.htm) returns the address of a variable. For example &a; will give actual address of the variable. |
| 7 | \*  [**Pointer operator \***](https://www.tutorialspoint.com/cplusplus/cpp_pointer_operators.htm) is pointer to a variable. For example \*var; will pointer to a variable var. |

**Precedence of Operators**

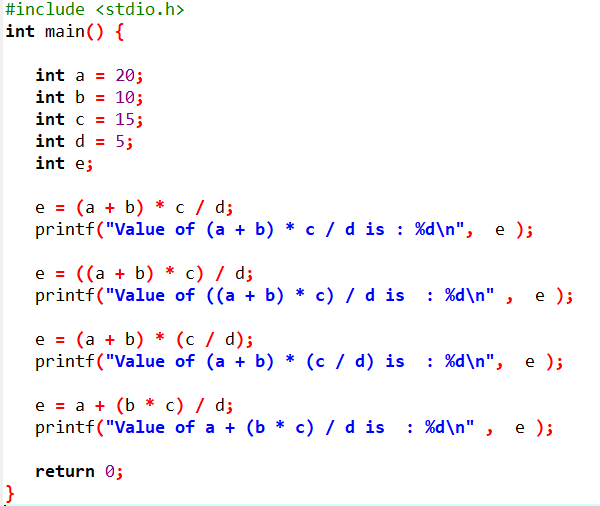
If more than one [operators](https://www.programiz.com/c-programming/c-operators) are involved in an expression, C language has a predefined rule of priority for the operators. This rule of priority of operators is called operator precedence.

In C, precedence of arithmetic operators( \*, %, /, +, -) is higher than relational operators(==, !=, >, <, >=, <=) and precedence of relational operator is higher than logical operators(&&, || and !).

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.



**Example:**

****

**Output:**

Value of (a + b) \* c / d is : 90

Value of ((a + b) \* c) / d is : 90

Value of (a + b) \* (c / d) is : 90

Value of a + (b \* c) / d is : 50

**Associativity of Operators**

If two operators of same precedence (priority) is present in an expression, Associativity of operators indicate the order in which they execute.

