Operators Precedence and Associativity

This page lists all C operators in order of their precedence (highest to lowest).
Operators within the same box have equal precedence.

Precedence	Operator	Description	Associativity

1	`()`
	`[]`
	`.`
	`->`
	`++ --`
		Parentheses (grouping)
		Brackets (array subscript)
		Member selection via object name
		Member selection via pointer
		Postfix increment/decrement (see Note 1)
			left-to-right

2	`++ --`
	`+ -`
	`! ~`
	`(type)`
	`*`
	`&`
	`sizeof`
		Prefix increment/decrement
		Unary plus/minus
		Logical negation/bitwise complement
		Cast (change type)
		Dereference
		Address
		Determine size in bytes
			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
6	`< <=` `> >=`	Relational less than/less than or equal to 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
		Addition/subtraction assignment
		Multiplication/division assignment
		Modulus/bitwise AND assignment
		Bitwise exclusive/inclusive OR assignment
		Bitwise shift left/right assignment
			right-to-left

15	`,`	Comma (separate expressions)	left-to-right

Note 1 - Postfix increment/decrement have high precedence, but the actual increment or decrement of the operand is delayed (to be accomplished sometime before the statement completes execution). So in the statement `y = x * z++;` the current value of z is used to evaluate the expression (i.e., `z++` evaluates to `z`) and `z` only incremented after all else is done.

Operator Precedence - When an expression contains two or more operators, normal operator precedence rules are applied to determine the order of evaluation. If two operators have different levels of precedence, the operator with the highest precedence is evaluated first. For example, multiplication is of higher precedence than addition, so the expression `2+3*4` is evaluated as

```
3 * 4 = 12
2 + 12 = 14
```

The evaluation order can be explicitly controlled using parentheses; e.g., (2+3)*4 is evaluated as

```
2 + 3 = 5
5 * 4 = 20
```

Operators in the previous table are presented in groups from highest to lowest precedence.

Operator Associativity - If two operators in an expression have the same precedence level, they are evaluated from left to right or right to left depending on their associativity. For example, addition's associativity is left-to-right, so the expression `2+3+4` is evaluated as `(2+3)+4`. In contrast, the assign operator's associativity is right-to-left; so the expression `x=y=z` is evaluated as `x=(y=z)`.