**Each operator in C has two properties**

1. **Precedence/Priority** – The order in which the operators are evaluated based on the relative priorities.
2. **Associativity** – Direction in which the operators act upon operands

The precedence and associativity of all the operators in C are given below.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Associativity** |
| () [] . -> ++  -- | Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2) | left-to-right |
| ++  -- +  - !  ~ (*type*) \* & sizeof | Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (change *type*) Dereference Address Determine size in bytes | right-to-left |
| \*  /  % | Multiplication/division/modulus | left-to-right |
| +  - | Addition/subtraction | left-to-right |
| <<  >> | Bitwise shift left, Bitwise shift right | left-to-right |
| <  <= >  >= | Relational less than/less than or equal to Relational greater than/greater than or equal to | left-to-right |
| ==  != | Relational is equal to/is not equal to | left-to-right |
| & | Bitwise AND | left-to-right |
| ^ | Bitwise exclusive OR | left-to-right |
| | | Bitwise inclusive OR | left-to-right |
| && | Logical AND | left-to-right |
| || | Logical OR | left-to-right |
| ?: | Ternary conditional | right-to-left |
| = +=  -= \*=  /= %=  &= ^=  |= <<=  >>= | Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment | right-to-left |
| , | Comma (separate expressions) | left-to-right |
| **Note 1:**Parentheses are also used to group sub-expressions to force a different precedence; such parenthetical expressions can be nested and are evaluated from inner to outer.  **Note 2:**Postfix increment/decrement has 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. | | |

#### Rules for evaluation of expression

* First parenthesized sub expression left to right are evaluated..
* If parentheses are nested, the evaluation begins with the innermost sub expression. .
* The precedence rule is applied in determining the order of application of operators in evaluating sub expressions. .
* The associability rule is applied when two or more operators of the same precedence level appear in the sub expression. .
* Arithmetic expressions are evaluated from left to right using the rules of precedence. .
* When Parenthesis is used, the expressions within parenthesis assume highest priority.