Operator Precedence and Associativity

Note: The table below lists operators in order of **decreasing precedence** (highest precedence at the top, lowest precedence at the bottom).

Precedence	Operators	Associativity
U LtoR (Unary)	`( )` , `[]` , `->` , `.` , `(postfix)++` , `(postfix)--`	Left to Right
U RtoL (Unary)	`!` , `~` , `+` , `-` , `++(prefix)` , `--(prefix)` , `*` , `&` , `(type)` , `sizeof`	Right to Left
A (Arithmetic)	`*` , `/` , `%`	Left to Right
A (Arithmetic)	`+` , `-`	Left to Right
S (Shift)	`<<` , `>>`	Left to Right
C (Compare)	`<` , `<=` , `>` , `>=`	Left to Right
C (Compare)	`==` , `!=`	Left to Right
B (Bitwise)	`&`	Left to Right
B (Bitwise)	`^`	Left to Right
B (Bitwise)	'¦'	Left to Right
L (Logical)	`&&`	Left to Right
L (Logical)	`¦¦`	Left to Right
(ternary op)	`?:`	Right to Left
A (Assignment)	`=` , `+=` , `-=` , `*=` , `/=` , `%=` , `&=` , `^=` , `¦=` , `<<=` , `>>=`	Right to Left
C (Comma)	`,`	Left to Right

Keep in mind

- Associativity specification is redundant for Unary operators and is only shown for completeness.
- Unary prefix operators always associate **Right to Left**:
  - Example: `sizeof ++ *p ≈ sizeof (++(*p))`
- Unary postfix operators always associate **Left to Right**:
  - Example: `a[i][j]++ ≈ ( (a[i]) [j] ) ++`

Precedence:

- Precedence of `(->)` > Precedence of (`postfix ++` or `postfix --`)

- Example:

```
int x = *p -> a++;
// Equivalent to *[(p->a)++]
// because '->' and postfix '++' operators are Left-to-Right associative.

int x = *p++ -> a;
// Equivalent to *[(p++)->a]
// because '->' and postfix '++' operators are Left-to-Right associative
// and Precedence of (->) > Precedence of (*)

*p.c; // ≈ *(p.c)
*p++;  // *(p++)
// because precedence of (postfix ++) > precedence of (*)
```