**Precedence of operators.**

A single expression may have multiple operators. For example:

|  |  |  |
| --- | --- | --- |
|  | x = 5 + 7 % 2; |  |

In C++, the above expression always assigns 6 to variable x, because the % operator has a higher precedence than the + operator, and is always evaluated before. Parts of the expressions can be enclosed in parenthesis to override this precedence order, or to make explicitly clear the intended effect. Notice the difference:

|  |  |  |
| --- | --- | --- |
| 1 2 | x = 5 + (7 % 2); // x = 6 (same as without parenthesis)  x = (5 + 7) % 2; // x = 0 |  |

From greatest to smallest priority, C++ operators are evaluated in the following order:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Level** | **Precedence group** | **Operator** | **Description** | **Grouping** |
| 1 | Scope | :: | scope qualifier | Left-to-right |
| 2 | Postfix (unary) | ++ -- | postfix increment / decrement | Left-to-right |
| () | functional forms |
| [] | subscript |
| . -> | member access |
| 3 | Prefix (unary) | ++ -- | prefix increment / decrement | Right-to-left |
| ~ ! | bitwise NOT / logical NOT |
| + - | unary prefix |
| & \* | reference / dereference |
| new delete | allocation / deallocation |
| sizeof | parameter pack |
| (*type*) | C-style type-casting |
| 4 | Pointer-to-member | .\* ->\* | access pointer | Left-to-right |
| 5 | Arithmetic: scaling | \* / % | multiply, divide, modulo | Left-to-right |
| 6 | Arithmetic: addition | + - | addition, subtraction | Left-to-right |
| 7 | Bitwise shift | << >> | shift left, shift right | Left-to-right |
| 8 | Relational | < > <= >= | comparison operators | Left-to-right |
| 9 | Equality | == != | equality / inequality | Left-to-right |
| 10 | And | & | bitwise AND | Left-to-right |
| 11 | Exclusive or | ^ | bitwise XOR | Left-to-right |
| 12 | Inclusive or | | | bitwise OR | Left-to-right |
| 13 | Conjunction | && | logical AND | Left-to-right |
| 14 | Disjunction | || | logical OR | Left-to-right |
| 15 | Assignment-level expressions | = \*= /= %= += -= >>= <<= &= ^= |= | assignment / compound assignment | Right-to-left |
| ?: | conditional operator |
| 16 | Sequencing | , | comma separator | Left-to-right |

When an expression has two operators with the same precedence level, *grouping* determines which one is evaluated first: either left-to-right or right-to-left.

Enclosing all sub-statements in parentheses (even those unnecessary because of their precedence) improves code readability.