| **Operator** | **Associativity** | **Description** | **Pattern** |
| --- | --- | --- | --- |
| :: :: | n/a | global scope (unary) namespace/class scope (binary) | ::name namespace\_name::entity, class\_name::member\_name |
| () () () {} type() type{} [] . -> ++ — typeid any\_cast const\_cast dynamic\_cast reinterpret\_cast static\_cast | left to right | parentheses function call initialization uniform initialization function-style cast function-style cast array subscript object member access pointer to object member access post-increment post-decrement run-time type information (RTTI) cast back to type from any cast away const run-time hierarchical down-cast cast pointers and integers type-checked cast | (expression) function\_name(parameters) type\_name(expression) type\_name(expression) new\_type(expression) new\_type{expression} pointer[expression] object.member\_name pointer\_to\_object->member\_name lvalue++ lvalue– typeid(type) or typeid(expression) any\_cast<type>(expression) const\_cast<type> (expression) dynamic\_cast<type> (expression) reinterpret\_cast<type>(expression) static\_cast<type>(expression) |
| + – ++ — !, not ~, compl (type) sizeof & \* new new[] delete delete[] | right to left | unary plus unary minus pre-increment pre-decrement logical not bitwise not C-style cast size in bytes address of dereference dynamic heap memory allocation dynamic array allocation dynamic heap memory release dynamic array release | +expression -expression ++lvalue –lvalue !expression ~expression (new\_type)expression sizeof(type) or sizeof(expression) &lvalue \*pointer\_expression new type new type[expression] delete pointer delete[] pointer |
| ->\* .\* | left to right | member pointer selector member object selector | object\_pointer->\*pointer\_to\_member object.\*pointer\_to\_member |
| \* / % | left to right | multiplication division modulo (remainder from division) | expression \* expression expression / expression expression % expression |
| + – | left to right | addition subtraction | expression + expression expression – expression |
| << >> | left to right | bitwise shift left bitwise shift right | expression << expression expression >> expression |
| < <= > >= | none | comparison less than comparison less than or equals comparison greater than comparison greater than or equals | expression < expression expression <= expression expression > expression expression >= expression |
| == != | none | test equality test inequality | expression == expression expression != expression |
| &, bitand | left to right | bitwise and | expression & expression |
| ^, bitxor | left to right | bitwise exclusive-or | expression ^ expression |
| |, bitor | left to right | bitwise or | expression | expression |
| &&, and | left to right | logical and | expression && expression |
| ||, or | left to right | logical or | expression || expression |
| ?: = \*= /= %= += -= <<= >>= &= |= ^= | right to left | conditional ternary operator assignment multiplication assignment division assignment modulo assignment addition assignment subtraction assignment bitwise shift left assignment bitwise shift right assignment bitwise and assignment bitwise or assignment bitwise exclusive-or assignment | expression ? expression : expression lvalue = expression lvalue \*= expression lvalue /= expression lvalue %= expression lvalue += expression lvalue -= expression lvalue <<= expression lvalue >>= expression lvalue &= expression lvalue |= expression lvalue ^= expression |
| throw | right to left | exception throw expression | throw expression |
| , | left to right | comma sequencing operator | expression, expression |

<https://learnmoderncpp.com/conditions-and-operators/>