2022

Table Of Operators

OPERATORS IN PROGRAMMING

MOHSEN GHOLAMI

Table of operators

The below table is primarily meant to be a reference chart that you can refer back to in the future to resolve any precedence or associativity questions you have.

Notes:

- Precedence level 1 is the highest precedence level, and level 17 is the lowest. Operators with a higher precedence level get evaluated first.
- L->R means left to right associativity.
- R->L means right to left associativity.

Prec/Ass	Operator	Description	Pattern
1 None	:: ::	Global scope (unary) Namespace scope (binary)	::name class_name::member_name
2 L->R	( ) ( ) ( ) { } type() type{} [ ] . -> ++ –– typeid const_cast dynamic_cast reinterpret_cast static_cast sizeof… noexcept alignof	Parentheses Function call Initialization Uniform initialization (C++11) Functional cast Functional cast (C++11) Array subscript Member access from object Member access from object ptr Post-increment Post-decrement Run-time type information Cast away const Run-time type-checked cast Cast one type to another Compile-time type-checked cast Get parameter pack size Compile-time exception check Get type alignment	(expression) function_name(parameters) type name(expression) type name{expression} new_type(expression) new_type{expression} pointer[expression] object.member_name object_pointer->member_name lvalue++ lvalue–– typeid(type) or typeid(expression) const_cast<type>(expression) dynamic_cast<type>(expression) reinterpret_cast<type>(expression) static_cast<type>(expression) sizeof…(expression) noexcept(expression) alignof(Type)
3 R->L	+ - ++ –– ! ~ (type) sizeof co_await & * new new[]	Unary plus Unary minus Pre-increment Pre-decrement Logical NOT Bitwise NOT C-style cast Size in bytes Await asynchronous call Address of Dereference Dynamic memory allocation Dynamic array allocation	+expression -expression ++lvalue ––lvalue !expression ~expression (new_type)expression sizeof(type) or sizeof(expression) co_await expression &lvalue *expression new type new type[expression]

	delete delete[]	Dynamic memory deletion Dynamic array deletion	delete pointer delete[] pointer
4 L->R	->* .*	Member pointer selector Member object selector	object_pointer->*pointer_to_member object.*pointer_to_member
5 L->R	* / %	Multiplication Division Modulus	expression * expression expression / expression expression % expression
6 L->R	+ -	Addition Subtraction	expression + expression expression - expression
7 L->R	<< >>	Bitwise shift left Bitwise shift right	expression << expression expression >> expression
8 L->R	<=>	Three-way comparison	expression <=> expression
9 L->R	< <= > >=	Comparison less than Comparison less than or equals Comparison greater than Comparison greater than or equals	expression < expression expression <= expression expression > expression expression >= expression
10 L->R	== !=	Equality Inequality	expression == expression expression != expression
11 L->R	&	Bitwise AND	expression & expression
12 L->R	^	Bitwise XOR	expression ^ expression
13 L->R	|	Bitwise OR	expression | expression
14 L->R	&&	Logical AND	expression && expression
15 L->R	||	Logical OR	expression || expression
16 R->L	throw co_yield ?: = *= /= %= += -= <<= >>= &= |= ^=	Throw expression Yield expression Conditional Assignment Multiplication assignment Division assignment Modulus assignment Addition assignment Subtraction assignment Bitwise shift left assignment Bitwise shift right assignment Bitwise AND assignment Bitwise OR assignment Bitwise XOR assignment	throw expression co_yield expression 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
17 L->R	,	Comma operator	expression, expression

https://about.me/iMohsen02 :)