Prec/Ass	Operator	Description	Pattern
1 L->R	`::`	Global scope (unary)	`::name`
	`::`	Namespace scope (binary)	`class_name::member_name`
2 L->R	`()`	Parentheses	`(expression)`
	`()`	Function call	`function_name(arguments)`
	`type()`	Functional cast	`type(expression)`
	`type{}`	List init temporary object (C++11)	`type{expression}`
	`[]`	Array subscript	`pointer[expression]`
	`.`	Member access from object	`object.member_name`
	`->`	Member access from object ptr	`object_pointer->member_name`
	`++`	Post-increment	`lvalue++`
	`--`	Post-decrement	`lvalue--`
	`typeid`	Run-time type information	`typeid(type) or typeid(expression)`
	`const_cast`	Cast away const	`const_cast<type>(expression)`
	`dynamic_cast`	Run-time type-checked cast	`dynamic_cast<type>(expression)`
	`reinterpret_cast`	Cast one type to another	`reinterpret_cast<type>(expression)`
	`static_cast`	Compile-time type-checked cast	`static_cast<type>(expression)`
	`sizeof…`	Get parameter pack size	`sizeof…(expression)`
	`noexcept`	Compile-time exception check	`noexcept(expression)`
	`alignof`	Get type alignment	`alignof(type)`
3 R->L	`+`	Unary plus	`+expression`
	`-`	Unary minus	`-expression`
	`++`	Pre-increment	`++lvalue`
	`--`	Pre-decrement	`--lvalue`
	`!`	Logical NOT	`!expression`
	`not`	Logical NOT	`not expression`
	`~`	Bitwise NOT	`~expression`
	`(type)`	C-style cast	`(new_type)expression`
	`sizeof`	Size in bytes	`sizeof(type) or sizeof(expression)`
	`co_await`	Await asynchronous call (C++20)	`co_await expression`
	`&`	Address of	`&lvalue`
	`*`	Dereference	`*expression`
	`new`	Dynamic memory allocation	`new type`
	`new[]`	Dynamic array allocation	`new type[expression]`
	`delete`	Dynamic memory deletion	`delete pointer`
	`delete[]`	Dynamic array deletion	`delete[] pointer`
4 L->R	`->*`	Member pointer selector	`object_pointer->*pointer_to_member`
	`.*`	Member object selector	`object.*pointer_to_member`

5 L->R	*	Multiplication	expression * expression
	/	Division	expression / expression
	%	Remainder	expression % expression
6 L->R	+	Addition	expression + expression
	-	Subtraction	expression - expression
7 L->R	<<	Bitwise shift left / Insertion	expression << expression
	>>	Bitwise shift right / Extraction	expression >> expression
8 L->R	<=>	Three-way comparison (C++20)	expression <=> expression
9 L->R	<	Comparison less than	expression < expression
	<=	Comparison less than or equals	expression <= expression
	>	Comparison greater than	expression > expression
	>=	Comparison greater than or equals	expression >= expression
10 L->R	==	Equality	expression == expression
	!=	Inequality	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
	and	Logical AND	expression and expression
15 L->R	||	Logical OR	expression || expression
	or	Logical OR	expression or expression
16 R->L	throw	Throw expression	throw expression
	co_yield	Yield expression (C++20)	co_yield expression
	?:	Conditional	expression ? expression : expression
	=	Assignment	lvalue = expression
	*=	Multiplication assignment	lvalue *= expression
	/=	Division assignment	lvalue /= expression
	%=	Remainder assignment	lvalue %= expression
	+=	Addition assignment	lvalue += expression
	-=	Subtraction assignment	lvalue -= expression
	<<=	Bitwise shift left assignment	lvalue <<= expression
	>>=	Bitwise shift right assignment	lvalue >>= expression
	&=	Bitwise AND assignment	lvalue &= expression
	|=	Bitwise OR assignment	lvalue |= expression
	^=	Bitwise XOR assignment	lvalue ^= expression
17 L->R	,	Comma operator	expression, expression