C++ 运算符优先级

下表列出 C++ 运算符的优先级和结合性。各个运算符以优先级的降序从上至下列出。

优先级	运算符	描述	结合性
1	`::`	作用域解析	从左到右

2	`a++` `a--`
	`type``()` *`type`*`{}`
	`a()`
	`a[]`
	`.` `->`
		后缀自增与自减
		函数风格转型
		函数调用
		下标
		成员访问	

3	`++a` `--a`
	`+a` `-a`
	`!` `~`
	`(`*`type`*`)`
	`*a`
	`&a`
	`sizeof`
	`co_await`
	`new` `new[]`
	`delete` `delete[]`
		前缀自增与自减
		一元加与减
		逻辑非和逐位非
		C 风格转型
		间接（解引用）
		取址
		取大小[注 1]
		await 表达式 (C++20)
		动态内存分配
		动态内存分配
			从右到左

4	`.*` `->*`	成员指针	从左到右
5	`a*b` `a/b` `a%b`	乘法、除法与余数	
6	`a+b` `a-b`	加法与减法	
7	`<<` `>>`	逐位左移与右移	
8	`<=>`	三路比较运算符(C++20 起)	
9	`<` `<=` `>` `>=`	分别为 < 与 ≤ 的关系运算符 分别为 > 与 ≥ 的关系运算符	
10	`==` `!=`	分别为 = 与 ≠ 的相等性运算符	
11	`a&b`	逐位与	
12	`^`	逐位异或（互斥或）	
13	`|`	逐位或（可兼或）	
14	`&&`	逻辑与	
15	`||`	逻辑或	

16	`a?b:c`
	`throw`
	`co_yield`
	`=`
	`+=` `-=`
	`*=` `/=` `%=`
	`<<=` `>>=`
	`&=` `^=` `|=`
		三元条件[注 2]
		throw 运算符
		yield 表达式 (C++20)
		直接赋值（C++ 类默认提供）
		以和及差复合赋值
		以积、商及余数复合赋值
		以逐位左移及右移复合赋值
		以逐位与、异或及或复合赋值
			从右到左

17	`,`	逗号	从左到右

1. ↑ `sizeof` 的操作数不能是 C 风格转型：表达式 `sizeof (int) * p` 无歧义地解释成 `(sizeof(int)) * p`，而非 `sizeof((int)*p)`。
2. ↑ 条件运算符中部（`?` 与 `:` 之间）的表达式分析为如同其带有括号：忽略其相对于 `?:` 的优先级。

分析表达式时，列于上面表中某行的运算符，将比列于低于它的行中拥有较低优先级的任何运算符，更紧密地与其实参相绑定（如同用了括号）。例如，表达式 `std::cout << a & b` 和 `*p++` 会被分析为 `(std::cout << a) & b` 和 `*(p++)`，而非 `std::cout << (a & b)` 或 `(*p)++`。

拥有相同优先级的运算符以其结合性的方向与各参数绑定。例如表达式 `a = b = c` 会被分析为 `a = (b = c)` 而非 `(a = b) = c`，因为赋值具有从右到左结合性，但 `a + b - c` 会被分析为 `(a + b) - c` 而非 `a + (b - c)`，因为加法和减法具有从左到右结合性。

结合性规定对于一元运算符是冗余的，只为完备而给出：一元前缀运算符始终从右到左结合（`delete ++*p` 为 `delete(++(*p))`）而一元后缀运算符始终从左到右结合（`a[1][2]++` 为 `((a[1])[2])++`）。要注意，结合性对成员访问运算符是有意义的，即使在它们与一元后缀运算符组合时也是如此：`a.b++` 会被分析为 `(a.b)++` 而非 `a.(b++)`。

运算符优先级不受运算符重载影响。例如，`std::cout << a ? b : c;` 会被分析为 `(std::cout << a) ? b : c;`，因为算术左移的优先级高于条件运算符。

注解

优先级和结合性是编译时概念，与求值顺序无关，后者是运行时概念。

标准自身不指定优先级。它们是从文法导出的。

表中并未包括 `const_cast`、`static_cast`、`dynamic_cast`、`reinterpret_cast`、`typeid`、`sizeof...`、`noexcept` 及 `alignof`，因为它们决不会有歧义。

一些运算符拥有代用写法（例如，&& 可为 `and`、|| 可为 `or`、! 可为 `not` 等）。

C 中，三元条件运算符拥有高于赋值运算符的优先级。因此，表达式 `e = a < d ? a++ : a = d` 在 C++ 中剖析成 `e = ((a < d) ? (a++) : (a = d))`，但在 C 中会由于 C 的语法或语义制约而编译失败。细节见对应的 C 页面。

参阅

常见运算符						
赋值	自增 自减	算术	逻辑	比较	成员访问	其他

`a = b`
`a += b`
`a -= b`
`a *= b`
`a /= b`
`a %= b`
`a &= b`
`a |= b`
`a ^= b`
`a <<= b`
`a >>= b`
	`++a`
	`--a`
	`a++`
	`a--`
		`+a`
		`-a`
		`a + b`
		`a - b`
		`a * b`
		`a / b`
		`a % b`
		`~a`
		`a & b`
		`a | b`
		`a ^ b`
		`a << b`
		`a >> b`
			`!a`
			`a && b`
			`a || b`
				`a == b`
				`a != b`
				`a < b`
				`a > b`
				`a <= b`
				`a >= b`
				`a <=> b`
					`a[b]`
					`*a`
					`&a`
					`a->b`
					`a.b`
					`a->*b`
					`a.*b`
						`a(...)`
						`a, b`
						`? :`

特殊运算符						
`static_cast` 转换一个类型为另一相关类型 `dynamic_cast` 在继承层级中转换 `const_cast` 添加或移除 cv 限定符 `reinterpret_cast` 转换类型到无关类型 C 风格转型 以 `static_cast` 、 `const_cast` 及 `reinterpret_cast` 的混合转换一个类型到另一类型 `new` 创建有动态存储期的对象 `delete` 销毁先前由 `new` 表达式创建的对象，并释放其所拥有的内存区域 `sizeof` 查询类型的大小 `sizeof...` 查询形参包的大小(C++11 起) `typeid` 查询类型的类型信息 `noexcept` 查询表达式是否能抛出异常(C++11 起) `alignof` 查询类型的对齐要求(C++11 起)						

C 运算符优先级 的 C 文档

来自“https://zh.cppreference.com/mwiki/index.php?title=cpp/language/operator_precedence&oldid=69772”