C++ built-in operators, precedence, and associativity

Article • 08/03/2021 • 8 contributors

Feedback

In this article

The C++ language includes all C operators and adds several new operators. Operators specify an evaluation to be performed on one or more operands.

Precedence and associativity

Operator *precedence* specifies the order of operations in expressions that contain more than one operator. Operator *associativity* specifies whether, in an expression that contains multiple operators with the same precedence, an operand is grouped with the one on its left or the one on its right.

Alternative spellings

C++ specifies alternative spellings for some operators. In C, the alternative spellings are provided as macros in the <iso646.h> header. In C++, these alternatives are keywords, and use of <iso646.h> or the C++ equivalent <ciso646> is deprecated. In Microsoft C++, the /permissive- or /Za compiler option is required to enable the alternative spellings.

C++ operator precedence and associativity table

The following table shows the precedence and associativity of C++ operators (from highest to lowest precedence). Operators with the same precedence number have equal precedence unless another relationship is explicitly forced by parentheses.

Expand table

Operator Description	Operator	Alternative
Group 1 precedence, no associativity		
Scope resolution	::	
Group 2 precedence, left to right associativity		
Member selection (object or pointer)	. or ->	
Array subscript	[]	
Function call	()	
Postfix increment	++	
Postfix decrement	--	
Type name	typeid	
Constant type conversion	const_cast	
Dynamic type conversion	dynamic_cast	
Reinterpreted type conversion	reinterpret_cast	
Static type conversion	static_cast	
Group 3 precedence, right to left associativity		
Size of object or type	sizeof	
Prefix increment	++	
Prefix decrement	--	
One's complement	~	`compl`
Logical not	!	`not`
Unary negation	-	
Unary plus	+	
Address-of	&	
Indirection	*	
Create object	new	
Destroy object	delete	
Cast	()	
Group 4 precedence, left to right associativity		
Pointer-to-member (objects or pointers)	.* or ->*	
Group 5 precedence, left to right associativity		
Multiplication	*	
Division	/	
Modulus	%	
Group 6 precedence, left to right associativity		
Addition	+	
Subtraction	-	
Group 7 precedence, left to right associativity		
Left shift	<<	
Right shift	>>	
Group 8 precedence, left to right associativity		
Less than	<	
Greater than	>	
Less than or equal to	<=	
Greater than or equal to	>=	
Group 9 precedence, left to right associativity		
Equality	==	
Inequality	!=	`not_eq`
Group 10 precedence left to right associativity		
Bitwise AND	&	`bitand`
Group 11 precedence, left to right associativity		
Bitwise exclusive OR	^	`xor`
Group 12 precedence, left to right associativity		
Bitwise inclusive OR	|	`bitor`
Group 13 precedence, left to right associativity		
Logical AND	&&	`and`
Group 14 precedence, left to right associativity		
Logical OR	||	`or`
Group 15 precedence, right to left associativity		
Conditional	? :	
Assignment	=	
Multiplication assignment	*=	
Division assignment	/=	
Modulus assignment	%=	
Addition assignment	+=	
Subtraction assignment	-=	
Left-shift assignment	<<=	
Right-shift assignment	>>=	
Bitwise AND assignment	&=	`and_eq`
Bitwise inclusive OR assignment	|=	`or_eq`
Bitwise exclusive OR assignment	^=	`xor_eq`
throw expression	throw	
Group 16 precedence, left to right associativity		
Comma	,	

See also

Operator overloading

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A