

ⓘ 本主題有部分內容為機器翻譯。



## 版本

依標題篩選

選升「多載」

## 運算子多載

運算子多載的一般規則

多載一元運算子

二元運算子

指派

函式呼叫

下標

成員存取

類別和結構

C++ 中的 Lambda 運算式

陣列

reference

指標

C++ 中的例外狀況處理

判斷提示和使用者提供的訊息

模組

範本

下載 PDF

# 運算子多載

2016/11/04 • ● ● ● ●

**Operator** 關鍵字會宣告函式，以指定套用至類別實例時的運算子符號。這讓運算子有多個意義，或稱為「多載」。編譯器會檢查運算元的類型，以區別運算子的不同意義。

## 語法

類型運算子運算子-符號 (參數清單)

## 備註

您可以用全域方式或以逐一類別為基礎，重新定義大多數內建運算子的函式。多載運算子實做為函式。

多載運算子的名稱是 **operator x**，其中 x 是下表中顯示的運算子。例如，若要多載加號，請定義稱為 **operator +** 的函式。同樣地，若要多載加法/指派運算子，**+=**，請定義稱為 **operator + =** 的函式。

## 可重新定義的運算子

運算子	名稱	類型
,	Comma (逗號)	Binary
!	邏輯 NOT	一元 (Unary)

此頁面有所助益嗎？

Yes No

## 本文內容

語法

備註

範例

本節內容

另請參閱

<b>!=</b>	不等	Binary
<b>%</b>	Modulus	Binary
<b>%=</b>	模數指派	Binary
<b>&amp;</b>	位元 AND	Binary
<b>&amp;</b>	傳址	一元 (Unary)
<b>&amp;&amp;</b>	邏輯 AND	Binary
<b>&amp;=</b>	位元 AND 指派	Binary
<b>()</b>	函式呼叫	—
<b>()</b>	轉換運算子	一元 (Unary)
<b>*</b>	乘	Binary
<b>*</b>	指標取值 (Dereference)	一元 (Unary)
<b>*=</b>	乘法指派	Binary
<b>+</b>	加	Binary
<b>+</b>	一元加號	一元 (Unary)
<b>++</b>	增量 <sup>1</sup>	一元 (Unary)
<b>+=</b>	加法指派	Binary
<b>-</b>	減	Binary
<b>-</b>	一元負運算	一元 (Unary)

--	遞減 <sup>1</sup>	一元 (Unary)
--	減法指派	Binary
->	成員選取	Binary
->*	成員指標選取	Binary
/	除	Binary
/=	除法指派	Binary
<	小於	Binary
<<	左移	Binary
<<=	左移指派	Binary
<=	小於或等於	Binary
=	指派	Binary
==	等式	Binary
>	大於	Binary
>=	大於或等於	Binary
>>	右移	Binary
>>=	右移指派	Binary
[]	陣列註標	—
^	互斥 OR	Binary

<code>^=</code>	互斥 OR 指派	Binary
<code> </code>	位元包含 OR	Binary
<code> =</code>	位元包含 OR 指派	Binary
<code>  </code>	邏輯 OR	Binary
<code>~</code>	一補數	一元 (Unary)
<b><code>delete</code></b>	刪除	—
<b><code>new</code></b>	新增	—
轉換運算子	轉換運算子	一元 (Unary)

<sup>1</sup>兩個版本的一元遞增和遞減運算子存在：前置遞增和後置遞增。

如需詳細資訊，請參閱運算子多載的[一般規則](#)。多載運算子的各種分類限制描述於下列主題：

- [一元運算子](#)
- [二元運算子](#)
- [指派](#)
- [函式呼叫](#)
- [註標](#)
- [類別成員存取](#)
- [遞增和遞減](#)。
- [使用者定義類型轉換](#)

下表中顯示的運算子無法多載。資料表包含 **#** 和 **##** 的預處理器符號。

## 不可重新定義的運算子

運算子	名稱
◦	成員選取
.*	成員指標選取
::	範圍解析
?:	條件式
#	前置處理器轉換成字串
##	前置處理器串連

雖然多載運算子通常由編譯器在程式碼中遇到時隱含地呼叫，不過也能像任何成員或非成員函式呼叫一樣地明確叫用：

```
Point pt;  
pt.operator+( 3 ); // Call addition operator to add 3 to pt.
```

## 範例

下列範例會多載 **+** 運算子，以加入兩個複數並傳回結果。

```
// operator_overloading.cpp  
// compile with: /EHsc  
#include <iostream>  
using namespace std;
```

using namespace std;

```
struct Complex {
    Complex( double r, double i ) : re(r), im(i) {}
    Complex operator+( Complex &other );
    void Display( ) {    cout << re << ", " << im << endl; }
private:
    double re, im;
};

// Operator overloaded using a member function
Complex Complex::operator+( Complex &other ) {
    return Complex( re + other.re, im + other.im );
}

int main() {
    Complex a = Complex( 1.2, 3.4 );
    Complex b = Complex( 5.6, 7.8 );
    Complex c = Complex( 0.0, 0.0 );

    c = a + b;
    c.Display();
}
```

6.8, 11.2

## 本節內容

- [運算子多載的一般規則](#)
- [多載一元運算子](#)
- [二元運算子](#)
- [指派](#)
- [函式呼叫](#)

- [註標](#)
- [成員存取](#)

## 另請參閱

[C++ 內建運算子、優先順序和順序關聯性  
關鍵字](#)

## 意見反應



提交並檢視相關的意見反應

本產品



本頁

 [檢視所有頁面意見反應](#) 

 [中文 \(繁體\)](#)  [佈景主題](#)

[舊版文件](#)

[部落格](#)

[參與](#)

[隱私權與 Cookie](#)

[使用規定](#)

[網站意見反應](#)

[商標](#)

© Microsoft 2020