# 重載運算子(Operator Overlorded)

摘錄自

<http://openhome.cc/Gossip/CppGossip/OverloadOperator.html>

以下C++檔案，請參考 子目錄 t15\_7(Point2D)

在C++中，預設除了基本資料型態可以使用運算子進行運算，例如int、double、char等，如果您要將兩個物件相加，預設上是不可行的。

然而很多情況下，您會想要將兩個物件的某些屬性值相加，並傳回運算後的結果，例如座標相加，如果您定義了Point2D類別，當中有x與y兩個屬性成員， 您會想要透過+或-運算子的動作得到座標相加或相減的動作，或是透過++與--來達到遞增或遞減的運算，在C++中，這可以透過重載運算子來達到目的。

運算子的重載其實是函式重載的一個延伸應用，您指定要重載哪一個運算子，並在類別中定義運算子如何動作，運算子重載的語法宣告如下所示：

**傳 回值 類別名稱::operator#(參數列) {**
**// 實作重載內容**
**}**

其中#中需指明您要重載哪一個運算子，例如重載一個+運算子，#處就替換為+運算子。

如果要重載++或--運算子，必須要注意到前置與後置的問題，例如一個變數x，您知道++前置時（++x）與++後置時（x++）實際上意義並不相同，在 重載時為了要區別前置與後置，C++中使用一個int參數來作區別：

**傳 回型態 operator++();  // 前置，例如++x**
**傳 回型態 operator++(int); // 後置，例如x++**
**傳 回型態 operator--();  // 前置 ，例如 --x**
**傳 回型態 operator--(int); // 後置，例如 x--**

在後置中會傳入一個0，但實質上沒有作用，只是作為識別前置與後置之用，通常在重載++與--運算子時，前置與後置都要重載；下面這個範例告訴您如何重載 +與-運算子，以及++與--運算子，以完成上面所提及的座標相加、相減、遞增、遞減的運算：

* Point2D.h

**class** Point2D {
**public**:
 Point2D();
 Point2D(**int**, **int**);
 **int** x() {**return** \_x;}
 **int** y() {**return** \_y;}
 Point2D **operator**+(**const** Point2D&); // 重載+運算子
 Point2D **operator**-(**const** Point2D&); // 重載-運算子
 Point2D& **operator**++(); // 重載++前置，例如 ++p
 Point2D **operator**++(**int**); // 重載++後置，例如 p++
 Point2D& **operator**--(); // 重載--前置，例如 --p
 Point2D **operator**--(**int**); // 重載--後置，例如 p--

**private**:
 **int** \_x;
 **int** \_y;
};

* Point2D.cpp

#include "Point2D.h"

Point2D::Point2D() {
 \_x = 0;
 \_y = 0;
}

Point2D::Point2D(**int** x, **int** y) {
 \_x = x;
 \_y = y;
}

Point2D Point2D::**operator**+(**const** Point2D &p) {
 Point2D tmp(\_x + p.\_x, \_y + p.\_y);
 **return** tmp;
}

Point2D Point2D::**operator**-(**const** Point2D &p) {
 Point2D tmp(\_x - p.\_x, \_y - p.\_y);
 **return** tmp;
}

Point2D& Point2D::**operator**++() {
 \_x++;
 \_y++;

 **return** \***this**;
}

Point2D Point2D::**operator**++(**int**) {
 Point2D tmp(\_x, \_y);
 \_x++;
 \_y++;

 **return** tmp;
}

Point2D& Point2D::**operator**--() {
 \_x--;
 \_y--;

 **return** \***this**;
}

Point2D Point2D::**operator**--(**int**) {
 Point2D tmp(\_x, \_y);
 \_x--;
 \_y--;

 **return** tmp;
}

* main.cpp

#include <iostream>
#include "Point2D.h"
**using** **namespace** std;

**int** main() {
 Point2D p1(5, 5);
 Point2D p2(10, 10);
 Point2D p3;

 p3 = p1 + p2;
 cout << "p3(x, y) = ("
 << p3.x() << ", " << p3.y()
 << ")" << endl;

 p3 = p2 - p1;
 cout << "p3(x, y) = ("
 << p3.x() << ", " << p3.y()
 << ")" << endl;

 p3 = ++p1;
 cout << "p3(x, y) = ("
 << p3.x() << ", " << p3.y()
 << ")" << endl;

 **return** 0;
}

執行結果：

p3(x, y) = (15, 15)
p3(x, y) = (5, 5)
p3(x, y) = (6, 6)

在重載 + 與 - 號運算子時，所接收的物件引數來自被重載的運算子右邊，例如在程式碼中加法運算時，+ 右邊是p2，所以傳入的物件引數就是p2物件，減法運算 時 - 號右邊是p1，所以傳入的就是p1物件，在傳入引數時，您使用傳參考的方式進行，這可以省去物件複製的動作，您也可以不使用傳參考，這對這個程式並不造成結果的差異，但使用傳參考方式可以節省CPU在複製物件時的處理時間。

大部份的運算子都是可以被重載的，除了以下的運算子之外：

**.   ::   .\*   ?:**

### C++ 快速導覽 - 類別 重載運算子

### 節錄自

### <http://pydoing.blogspot.tw/2013/02/cpp-overload-operator.html>

運算子 (operator) 可依類別 (class) 的需要進行重載 (overload) ，舉例如下

|  |  |
| --- | --- |
|  | #include <iostream> class Demo {public:    Demo() {        a = 1;        b = 1;    }     Demo(int pa, int pb) {        a = pa;        b = pb;    }         Demo operator+(const Demo& p) {        Demo demo;        demo.a = this->a + p.a;        demo.b = this->b + p.b;        return demo;    }                  void do\_something() {        std::cout << a + b << std::endl;    } private:    int a;    int b;     }; int main(void) {    Demo d1(2, 6);    d1.do\_something();    Demo d2(10, 24);    d2.do\_something();    Demo d3;    d3 = d1 + d2;    d3.do\_something();     return 0;}  |

運算子重載在， **operator** 後面接上要重載的運算子，另外要有 **const** 的參考 (reference) 參數 (parameter) ，表示運算子後面的運算元 (operand)

|  |  |
| --- | --- |
|  | Demo operator+(const Demo& p) {    Demo demo;    demo.a = this->a + p.a;    demo.b = this->b + p.b;    return demo;} |