Chapter 11 ： 函式之傳回值

我們前兩章已討論了函式之傳入參數，現在我們來討論函式之傳回值，函式除了可傳回一般之數值或函式中變數/運算式之值外(使用 return)，另有以下特別的傳回值型態(指標/參考)。

11.1 由函式傳回陣列(以指標爲函式之傳回值型態)

格式:

```
int[20]    function(參數列);     //錯誤

int* function(參數列);     //正確，傳回之指標指向某一陣列
```

說明:

與函式之輸入參數不同，函式並不特別支援傳回陣列的格式，不過技術上我們可以將傳回值型態指定爲一指標，函式內並以 return 傳回陣列之起始位址，此將使「函式之呼叫」成爲指到回傳陣列之首的指標。

程式範例: cpp_ex55.cpp

注意要點:

一般而言，如欲使用函式對主程式中的陣列作更改，我們不需使用傳回陣列的方式，因爲所傳入之陣列參數其效果相當於傳指標或參考，主程式中所傳入之陣列可直接被函式更改其內容。故一種作法是於主程式中產生一陣列，並將其自參數列輸入函式中，專用於接收傳回值。

但如欲在函式中產生某一新陣列，並將其傳回，一般情形下並不妥當(但並非不可)，因爲函式中宣告產生之陣列爲函式內之區域變數，其所屬空間會於函式執行結束後結束其生命周期，自動自記憶體釋放，傳回此陣列並無意義。除非於函式內對所產生之新陣列做動態記憶體配置，如此一來其所屬空間將不會因爲函式的結束而自動自記憶體中釋放(不需遵守 scope rule)。有關動態記憶體配置將於 Chapter 13 介紹。

程式範例: cpp_ex56.cpp

11.2 以參考爲函式之傳回值型態

格式:

資料型態& 函式名稱(參數列)

說明:

將函式的傳回值宣告爲參考型態，「函式之呼叫」將成爲函式內以 return 傳回「變數」之參考(別名)，如此一來可以節省拷貝傳回值之時間與記憶體空間，當傳回的資料爲大型物件，如結構體或類別時，效率上會有顯著的差異。

使用參考型態作爲函式的傳回值，以 return 所傳回者必須要是一個與參考傳回值型態相同的「變數」，或必須要是能代表某個與參考傳回值型態相同的記憶體巢者，例如下例之*z。Why？

且如同傳回指標一樣，其傳回之變數如爲函式之中所宣告產生的暫存區域變數並不妥當(但並非不可)，如此爲必要，則必需給予此變數作(動態)記憶體配置(Chapter 13)。

例:

```
int& sum(int x, int y)
{
    int* z   =   new int(x + y);    //需做動態記憶體配置
    return *z;
}
```

程式範例: cpp_ex57.cpp

注意要點:

比較以下兩種呼叫方式的不同：

```
int z = sum(x,y);          // copy the value of sum to z
int& z = sum(x,y);         // z is a reference of sum
```