Chapter 12 ： 函式覆載、遞迴式函式與函式指標

12.1 函式覆載(Function Overloading)

說明:

函式覆載是指不同的函式定義可以有相同的函式名稱(函式的名稱可以重複)，但這些同名函式必須具有不同的參數列，編譯系統可以就其彼此參數列之不同爲依據，判斷選擇出其中最接近呼叫條件的一個函式來執行。

不同的參數列是指同名的函式，其後之參數列具有不同的參數個數、或不同的資料型態、或二者皆有。程式將依呼叫時所用的參數，判斷出最接近者，並執行此一函式。需注意的是，不同傳回值型態，不能作爲辨別同名覆載函式之依據。

函式覆載通常用於產生多個同名函式以用於處理相同或相近的工作，但各自針對不同的資料型態。

例:

以下爲合法的函式覆載：

```
double max(double x, double y);
int max(int x, int y);                  // 不同參數資料型態

int min(int x, int y);
int min(int x, int y, int z);           // 不同參數個數
```

以下的函式爲不合法的函式覆載：

```
int sum(int x, int y);
double sum(int x, int y);               //不同傳回值型態
```

程式範例: cpp_ex58.cpp

注意要點:

當呼叫所用的參數需作變數轉換，而造成函式條件相當而無法決定呼叫何者時，即造成錯誤。**例：** cout << max(5, 55.5);

12.2 遞迴式(Recursive)函式

說明:

如某函式中的程式亦呼叫函式自己本身，則此函式稱爲「遞迴式函式」。運用遞迴式函式的功能對於某些運算很有用，如範例，但使用遞迴式函式並不能提升程式效率與節省記憶體。

例:

```
void functionA( )
{
    ……..
    functionA( );
}
```

程式範例: cpp_ex59.cpp

12.3 函式指標

說明:

如同一維陣列一樣，某函式的名稱亦爲一指標，其指到此函式所在之記憶體(存放此函式程式所在記憶體區塊的位址)。使用函式指標可將某一函式傳入另一個函式，供此函式做運用。使用函式指標來呼叫函式，也需以「*」依址取值(函式)。

所傳入之函式(如下例 functionA)，其參數列與傳回值，需與函式指標(如下例 fun)所宣告的參數列與傳回值完全相同。

格式:

```
// 函式宣告，變數與輸入函式名可省略
int functionA(int a);
int functionB(int b, int (*fun)(int));

// 函式呼叫
x = functionB(y, functionA);
```

程式範例: cpp_ex60.cpp