Chapter 8 ： 變數的範疇(區域變數 vs.全域變數)

8.1 變數的範疇

說明:

變數的範疇(scope)指某變數於程式中可被參考使用的有效範圍，如某變數被宣告於某一「程式區塊」({ } 中的程式)，則其只能在此「區塊本身」及其內所含「子區塊」中被參考使用，其生命週期自其宣告時開始，直到其宣告所在之程式區塊執行結束為止。Question：(1)程式區塊何時出現？(2)區塊間的關係有幾種？

位於所有程式區塊外之變數稱為全域變數(global variable)，區塊中之變數稱為區域變數(local variable)，例如宣告於主程式中的任何變數。

C/C++允許全域與各層區域之「不同」變數使用相同變數名，二變數如各自宣告產生於不同的區塊中，例如主程式與自訂函式，即使同名，彼此間也無任何關係，也不會相互影響(各有其獨有的記憶體空間)。

當某區塊內包含子區塊，且二區塊各自宣告了同名的變數，則於內部區塊中，外部區塊中所宣告的變數(全域變數或外部區域變數)會被暫時隱藏(非消失)，而於內部區塊內所宣告的變數(內部區域變數)為顯現而被使用，直到內部區塊結束為止。換句話說，於內部區塊中，使用此同名之變數，其所代表為於內部區塊所新宣告之變數(的記憶體空間)，當內部區塊結束後，此變數名恢復為代表先前於外部區塊內所宣告之變數(的記憶體空間)。

基於變數宣告的位置，變數的範疇有以下四種：(1)檔案範疇，(2)函式範疇，(3)程式區塊(if, for….)範疇，以及(4)函式原型宣告範疇。

例:

```
int x;     //全域變數

void main()
{
      int x;      //主程式(main 函式)區域變數
      ……..
}
```

```
int fun( )
{
    int x;      //fun 函式區域變數
    …….
    for( ; ; )
    {
        int x;      //for 子區塊區域變數
        ……..
    }
}
```

程式範例: **cpp_ex37.cpp　lab8-ext.cpp**

8.2　範疇解析運算子(Unary Scope Resolution Operator)

格式:

```
::全域變數
```

說明:

C/C++允許全域與區域變數使用相同變數名，於內部區塊中可使用範疇解析運算子來得到全域變數(不宣告於任何{ }內者)所含之值。

例:

```
double value = 1.234;

void main( )
{
    int value = 8;
    cout<< value    << endl;
    cout<< ::value << endl;
}
```

程式範例: **cpp_ex38.cpp**

8.3 函式之區域變數與參數傳遞

函式之參數也是屬於其函式區塊內之區域變數，只不過其是於函式定義之( )中宣告，而非其後之{ }函式區塊內，其格式是為了能接收函式呼叫時所傳入之值。註：for(int i=0;….){….}之情況亦相同，其為何？

如呼叫函式時所用之變數名稱，剛好與函式之參數相同，則其彼此之間除同名外，並沒有任何關係，一如不同區塊中所宣告的同名變數，彼此獨立且有各自的記憶體。

函式呼叫時，系統之運作為將輸入變數之值拷貝到其相對應函式參數之記憶體巢中，而非令此二變數參考到同一記憶體。

例:

```
int fun(int x, int y)
{
    int z = x + y;
    return z;
}

void main( )
{
    …….
    z = fun(x, y);        //呼叫時，有如函式內執行： int x = x;  int y = y;
}
```

程式範例: cpp_ex39.cpp

注意要點:

我們經常會想要將主程式的變數與函式中的變數結合再一起(參考到同一記憶體)，或是說傳入變數，而非傳入變數值，以期能由函式內修改到主程式內變數的值，或是使函數傳回兩個以上的值，此時便無法藉由一般變數之參數傳遞達到，而必須要用指標或參考型態，詳見以下之函式參數傳遞章節。