6-5 全域變數與靜態變數

全域變數(Global variable)

一般來說，我們使用函數時會將操作到的變數限制在函數裡，也就是以區域變數的方式使用。如有需要操作到函數外面的變數，我們會用傳參考或傳址的方式來處理。

我們以一個抽號碼牌的程式來示範。

練習：抽號碼牌(1)

```
#include <iostream>

using namespace std;

int getTicket(int &num)  // 以傳參考方式遞增外面的 num 變數值
{
    num++;
    return num;
}

int main()
{
    int num = 0;  // 記錄目前發到幾號

    cout << "I have ticket No." << getTicket(num) << endl;
    cout << "I have ticket No." << getTicket(num) << endl;
    cout << "I have ticket No." << getTicket(num) << endl;

    return 0;
}
```

I have ticket No.1 I have ticket No.2 I have ticket No.3

使用這種方式沒什麼問題，但是每次都要傳遞變數 num。如果想避免這個麻煩，可以使用全域變數，也就是把 num 宣告在所有函數(包含 main)的外面。

練習：抽號碼牌(2)

```
#include <iostream>

using namespace std;

int num = 0;  // 記錄目前發到幾號。宣告在這裡是全域變數

int getTicket()  // 沒有參數
```

```
{
    num++;  // 因為 num 是全域變數，所以到處都可以存取它
    return num;
}

int main()
{
    cout << "I have ticket No." << getTicket() << endl;  // 沒有引數
    cout << "I have ticket No." << getTicket() << endl;  // 沒有引數
    cout << "I have ticket No." << getTicket() << endl;  // 沒有引數

    return 0;
}
```

I have ticket No.1 I have ticket No.2 I have ticket No.3

使用全域變數雖然很方便，但是它有一個極大的缺點，就是大家都可以動到它。

有時候你會很納悶，明明我沒動它，它的值怎麼變了。找了半天才發現在某個不起眼角落或函數裡的程式碼動到它的值。

函數裡的靜態變數(static variable)

一般來說宣告在函數裡的變數都是區域變數(local variable)，一但離開函數後就會消滅，下次被呼叫時才會重新產生出來。

但是如果在宣告時，在前面加上 `static` 修飾詞，它就會是個靜態變數，在離開函數時變數會記得當下的值，不會消滅。下次函數被呼叫時，它依然活著不會被重新產生和給定初值。

練習：抽號碼牌(3)

```
#include <iostream>

using namespace std;

int getTicket()  // 沒有參數
{
        static int num = 0;  // 靜態變數，只在程式開始時指定一次初值
    num++;
    return num;
}

int main()
{
    cout << "I have ticket No." << getTicket() << endl;  // 沒有引數
    cout << "I have ticket No." << getTicket() << endl;  // 沒有引數
    cout << "I have ticket No." << getTicket() << endl;  // 沒有引數
```

```
    return 0;
}
```

I have ticket No.1 I have ticket No.2 I have ticket No.3

在某些情況下，靜態變數是很好用的！

Revision #5
★Created 2024-06-03 13:08:26 UTC by huihui
Updated 2024-08-01 13:03:52 UTC by huihui