本書目錄

第 01 章 最夯的程式語言－Visual C++ 2008

第 02 章 資料型別與運算子

第 03 章 流程控制

第 04 章 陣列

第 05 章 指標

第 06 章 函式

第 07 章 物件與類別

第 08 章 物件繼承與委派

第 09 章 視窗程式、表單介面設計、常用控制項

第 10 章 進階控制項

第 13 章 滑鼠鍵盤與共享事件

第 14 章 繪圖與多媒體

第 15 章 資料庫程式設計工具的使用

第 16 章 檔案與資料夾處理

第 17 章 專題製作

CHAPTER 7

物件與類別

物件 (Object) 可說是一件東西，物件具有屬性 (Property)、方法 (Method) 以及觸發的事件 (Event)。例如：David (物件) 是一個人 (類別)，David 的身高是 183 公分 (屬性)，David 會跑步 (方法)。

類別 (Class) 是一個物件的設計藍圖，而物件則是由該類別所定義出來的實體。例如：建築師設計了一張房子的設計圖 (Class)，工程人員可以根據這張設計圖，蓋出了很多棟的房子 (Object) 來。

學習重點

- **物件導向程式設計**
- **物件與類別**
- C++ **定義類別**
- CLR **定義類別**
- **屬性**
- **方法**
- **多載**
- **建構式**
- **解構式**
- **事件**
- **靜態成員**

7.1 物件與類別

7.1.1 物件導向程式設計

目前的程式設計主流是物件導向 (OOP)，C++ 已提供完整的物件導向功能。在物件導向設計裡，物件可以被封裝 (保護) 和繼承 (重複使用)。因為物件的封裝會使物件受到保護，要存取物件必須透過屬性 (Property)、方法 (Method) 和事件 (Event)。物件的封裝如同日常生活中的 ATM 提款機，其提款必須透過正確的方式才能提款 (輸入帳號和密碼)，而不可用不當破壞方式提款。

物件的繼承可使程式碼重覆再利用，當產生一個表單或拖曳一個按鈕到表單中時，就已經繼承了別人撰寫的物件。因為物件的繼承使得前人的經驗和智慧可以被不斷的累積，如此，才能創造出像 Windows 這麼龐大的系統。

7.1.2 物件與類別的觀念

物件 (Object) 可說是一件東西，物件具有屬性 (Property)、方法 (Method) 以及觸發的事件 (Event)。例如：David (物件) 是一個人 (類別)，David 的身高是 183 公分 (屬性)，David 會跑步 (方法)。如果用 C++ 程式表示如下：

```
Person David;      //建立 David 物件屬於 Person 人類類別
David.Height=183; //David 的身高是 183 公分
David.Walk(5);     //David 每小時可以跑 5 公里
```

在 C++ 工具箱中的控制項，通常都有事先已經定義好的屬性、方法和事件，由這些控制項可建立物件。例如：由 Button 控制項建立的按鈕 MyButton1 就是一個物件，它具有 Text 、Backcolor 等屬性，Show()、Hide() 等方法和 Click() 等事件。

類別 (Class) 是一個物件的設計藍圖，而物件則是由該類別所定義出來的實體。例如：建築師設計了一張房子的設計圖 (Class)，工程人員可以根據這張設計圖，蓋出了很多棟的房子 (Object) 來。

本書會先以原生 C++ 的語法介紹物件導向的觀念，隨後再以 CLR 的語法詳細說明物件導向的實作、屬性、方法、建構式、解構式、多載、繼承…。

C++ 建立實體物件的語法有三種，第一種語法為：

```
類別名稱 物件變數名稱;   //根據類別建立實體物件
```

例如以 Person 類別建立 David 物件的語法如下：

```
Person David;
```

第二種語法為建立指標物件，再將指標指向物件變數配置記憶體位址：

```
類別名稱 物件變數名稱;
類別名稱 *指標物件變數;         //先宣告指標物件變數屬於某個類別
指標物件變數 = &物件變數名稱; //再將指標指向物件變數配置記憶體位址
```

例如：

```
Person David;
Person *Amy;
Amy=&David;
```

第三種語法為建立動態配置的指標物件：

```
類別名稱 *指標物件變數;                  //先宣告指標物件變數屬於某個類別
指標物件變數 = new 類別名稱([初始值]); //再使用 new 動態配置記憶體空間
```

例如：

```
Person *David;         //先宣告指標物件變數 David
David = new Person(); //配置記憶體空間
```

上面兩行也可合併成一行：

```
類別名稱 *指標物件變數 = new 類別名稱([初始值]);
```

例如：

```
Person *David = new Person(); //宣告並建立 Person 類別的 David 物件
```

要建立物件必須先有類別，有了類別才能產生物件，類別就像建築的藍圖，而物件則是由藍圖所建出的一棟棟的房子。

7.1.3 C++ 定義類別

要自行定義類別 C++ 必須使用「class 類別名稱{ };」語法，並依需要在類別中定義類別成員(包括欄位、方法和事件等)，其中欄位相當於變數或常數，可以直接存取。

C++ 定義類別的語法

```
class 類別名稱
{
  private:
     [static] 資料型別 成員;
  public:
     [static] 資料型別 成員;
  protected:
     [static] 資料型別 成員;
};
```

例如：

```
class Person          //定義 Person 類別
{
   private:
       int Weight;    //宣告 Weight 體重欄位為私有型態
   public:
       int Tall;      //宣告 Tall 身高欄位為共用型態
       void Walk(int var) //定義 Walk 方法為共用型態
       {
           cout << "每小時跑公里數=" << var;
       }
};
```

存取修飾詞中，資料成員若宣告為 public 代表該資料為共用型態，其存取沒有限制，即允許目前應用程式中的所有類別均可存取。若宣告為 private 代表私有型態，只允許在類別內部使用。而 protected 則可以在類別內部以及其子類別中使用，若省略關鍵字，系統的預設值為 private。

「欄位」相當於變數或常數，用以儲存物件的資料，可用一般的資料型別或其他的類別來宣告。「屬性」可以保護物件欄位，屬性的建立較為複雜，將在 7.3 節中詳細說明。「方法」用以表示該類別所擁有的行為。

類別的最後面記得要加上「;」號，否則會產生編譯錯誤。

有了類別後，就可以依據類別來建立物件，並以「.」運算子存取其成員。

```
Person David;    //宣告並建立 Person 類別的 David 物件
David.Tall=183; //存取成員
```

也可以指標物件宣告建立動態物件指標，並以「->」運算子存取其成員。

```
Person *Tony = new Person(); //建立物件指標宣告
Tony->Name = 183;            //存取成員
```

範例：C++ 自行定義類別

定義 Person 類別，並建立 Name 姓名欄位、Tall 身高欄位為共用欄位，Walk() 方法顯示跑步的速度。最後在 _tmain() 函式中使用 Person 類別建立兩個物件。(FirstClass1.sln)

執行結果

(可以開啟 <C:\VC++2008\ch07\完成參考\FirstClass1\FirstClass1.sln> 專案觀看結果)

動手實作 (本例請建立 **Win32 主控台應用程式** 專案)

1. 新增專案：新增 **Win32 主控台應用程式** 專案，專案名稱為「FirstClass1」。

2. 在 FirstClass1.cpp 內輸入下列程式碼來定義 Person 類別。

```
1  // FirstClass1.cpp : 定義主控台應用程式的進入點。
2  //
3
4  #include "stdafx.h"
5  #include "iostream"
6  #include "string"
7  using namespace std;
8
```

```
9   class Person
10  {
11      public:
12      int Tall;    //宣告 Tall 身高欄位為公用型態
13      string Name; //宣告 Name 姓名欄位為公用型態
14      void Walk(int var)     //定義跑步方法
15      {
16          cout << Name << "每小時可跑" << var << " 公里" << endl;
17      }
18  };
19
20  int _tmain(int argc, _TCHAR* argv[])
21  {
22      Person David; //宣告並建立 Person 類別的 David 物件
23      David.Name = "David";        //設定 Name 欄位
24      David.Tall = 183;            //設定 Tall 欄位
25      cout << "David 身高=" <<  David.Tall << endl; //取得 Tall 欄位
26      David.Walk(5);  //呼叫 Walk() 方法，並傳遞參數值 5
27      cout << "-------------------------\n";
28
29      // 使用動態指標宣告
30      Person *Tony;          //先宣告 Person 類別的 Tony 指標物件
31      Tony = new Person();  //配置記憶體空間
32      Tony->Name = "Tony";
33      Tony->Tall = David.Tall;
34      cout << "Tony 身高=" <<  Tony->Tall << endl; //取得 Tall 欄位
35      Tony->Walk(8);
36      system("pause");
37      return 0;
38  }
```

3. 儲存專案及執行。

Person 類別程式說明	
9-18	定義 Person 類別。建立的類別並不會主動執行，必須用類別建立物件後，由物件以呼叫屬性、方法等方式來執行。
12-13	建立二個欄位。
14-17	建立 Walk 方法，此方法可以傳送「速度 (var)」參數，同時顯示速度，此方法的存取是 public 如此才可以被不同類別 (_tmain() 函式) 呼叫。

_tmain() 程式說明

22	Person David，建立 Person 類別的 David 物件。
23-24	設定二個欄位的初始值，物件成員存取語法是「物件.成員」。
25	顯示欄位值。
26	呼叫 Walk 方法，並傳遞參數，於是第 16 列程式會顯示 David 的速度。
30-31	Person *Tony 以指標物件建立另一個物件 Tony。
32-35	指標物件成員存取語法是「物件->成員」。

7.1.4 CLR 定義類別

了解 C++ 如何定義類別後，接著要探討 CLR 定義類別的語法，CLR 定義類別的語法和 C++ 相似，但 CLR 類別名稱類別前應加上關鍵字 [public] [value|ref]。

CLR 定義類別的語法

```
[public][value|ref] class 類別名稱
{
  private:
     [static] 資料型別 成員;
  public:
     [static] 資料型別 成員;
  protected:
     [static] 資料型別 成員;
};
```

例如：宣告 Person 類別是全域性類別。

```
public class Person{};
```

省略 public 宣告，預設是 public，所以上面宣告可以省略為：

```
class Person{};
```

上面不使用 ref 或 value 關鍵字的語法其實是為了和舊的 C++ 原生語法相容，亦即在 CLR 模式也可以使用原生 C++ 相容的語法「class 類別名稱{};」來建立類別，但使用這種語法建立的類別，在 CLR 模式會有一些限制，例如：無法建立堆積變數、屬性。

例如：在 CLR 模式使用原生 C++ 建立類別語法建立類別，並無法建立堆積變數、屬性。

```
class Car // 註：class 前未加 ref 或 value
```

```
{
private: int mSpeed;
public :
    int n;
    // 在非 ref class 不可建立屬性，以下建立屬性將產生編輯錯誤
    /*
    property int Speed      //定義 Speed 屬性
    {
        int get(){ return mSpeed;  }
         void set(int value){ mSpeed = value; }
    }*/
};

int main(array<System::String ^> ^args)
{
    Car a1; // 建立物件方法一
    a1.n=1;
    Console::WriteLine(L"a1.n= {0}",a1.n);

    Car b1; // 建立物件方法二
    Car *a2;
    a2=&b1;
    a2->n=2;
    Console::WriteLine(L"a2->n= {0}",a2->n);

    Car *a3=new Car();  // 建立物件方法三
    a3->n=3;
    Console::WriteLine(L"a3->n= {0}",a3->n);

    // 在非 ref class 類別建立堆積變數，以下建立^a4 將產生編輯錯誤
    /* Car ^a4=gcnew Car();  */

    Console::Read();
    return 0;
}
```

最好是完全採用標準的 CLR 模式語法來建立類別，亦即必須再加上關鍵字 ref 或 value，ref 代表建立的類別是參考類別，value 代表建立的類別是數值類別。