國立屏東大學 資訊工程學系 C++程式設計入門教材

3. IPO程式設計思維方法

我們學習電腦程式設計的主要目的，就是為了幫助人們解決問題，而絕大多數的問題都與資料的輸入、輸出與處理有關。在進行更深入、更複雜的程式設計問題前，本章將先為讀者介紹一種簡單的程式設計思維方法 — 「輸入-處理-輸出模型（Input-Process-Output Model，簡稱為IPO模型），它將程式視為是由「輸入（Input）」、「處理（Process）」與「輸出（Output）」等三個階段所組成。我們將以BMI（Body Mass Index，身體質量指數）計算程式與手機門號違約金計算程式，實際示範使用IPO模型進行程式設計的過程，並且也為讀者說明如何使用C++語言來搭配IPO模型，完成資料的輸入、處理與輸出等相關的程式實作。

3.1 IPO模型

IPO模型（Input-Process-Output Model，輸入-處理-輸出模型）[1][2]是一種程式設計的思維方法，我們可以據以構思問題的解決方案或是用以描述程式的結構。具體來說，IPO模型將電腦程式的運作分解為三個與資料相關的階段：資料輸入（Input）、資料處理（Process）與資料輸出（Output），figure 1顯示了IPO模型的概念：電腦程式首先會在「輸入（Input）階段」接收使用者所輸入的一筆或多筆資料（有些程式在執行時，並不需要使用者輸入的資料，因此也可以略過此階段），然後在「處理（Process）階段」，依據所取得的資料進行與程式應用目的相關的資料處理，最後在「輸出（Output）階段」將處理完的一筆或多筆結果加以輸出。

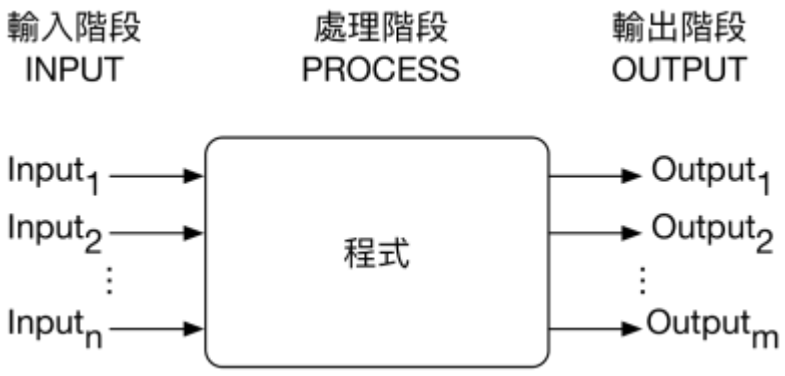

Fig. 1: IPO模型概念圖。

IPO模型適用於大部份與資料處理相關的電腦程式，對於初學者來說是相當適合的學習起點，只要熟悉這個模型就可以應用在許多簡單的程式設計問題之上。以下我們將以一個BMI計算程式示範如何使用IPO模型進行程式的設計與實作。

BMI(Body Mass Index，身體質量指數)是衡量肥胖程度的一種簡單的方式，其計算公式為體重（公斤）除以身高（公尺）的平方。一般而言，成年人的BMI指數介於18.5至24之間被視為是正常健康的狀態，小於18.5或大於24則被視為過輕或過重；若是BMI值大於27、30與35，則被視為是輕度、中度與重度肥胖。此程式會接收由使用者所輸入的體重與身高，然後依據BMI計算公式（體重除以身高的平方），再求出BMI值後加以輸出。若使用IPO模型，則可以表達如figure 2。

Fig. 2: BMI計算程式的IPO模型圖。

我們將此種表達方式稱為IPO模型圖（IPO Model Diagram），除了使用圖示法來表達一個IPO模型之外，我們也可以採用文字說明的方法，來描述此BMI計算程式三個階段的工作項目，例如使用table 1的IPO表（IPO Chart）：

Input輸入階段	**Process處理階段**	**Output輸出階段**
取得使用者所輸入的體重與身高。	依據「體重除以身高的平方」公式計算BMI值。	將計算完成的結果加以輸出。

Tab. 1: BMI計算程式的IPO表

本節後續將分別就Input、Process以及Output階段的程式實作加以說明。

3.1.1 Input階段與cin物件

Input階段的主要工作，就是取得使用者所輸入的資料。如果要在C++語言的程式中取得使用者輸入的資料，可以使用與「cout」[1]相似但用於取得使用者輸入的「cin」物件。如同cout一樣，cin也與標準的渠道連接，只不過cout連結的是stdout，而cin連結的則是標準輸入渠道（Standard Input，stdin）。在預設的情況下，stdin是連通到鍵盤，因此使用者從鍵盤所輸入的資料，就可以透過stdin渠道流動到cin，讓我們取得使用者所輸入的資料，如figure 3所示:

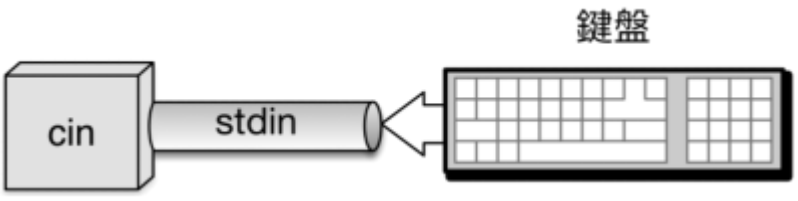

Fig. 3: cin與stdin渠道。

至於cin在執行時所取回的使用者輸入（由使用者從鍵盤輸入的資料），又該如何保存起來呢？事實上，電腦程式是透過作業系統載入到記憶體內加以執行，其執行時所需使用到任何資料也都是存放在記憶體內。因此，我們必須先取得一塊記憶體空間，才能用來存放使用者從鍵盤輸入的資料。具體的做法是，先在程式中說明「我們需要一塊記憶體空間」，才能讓作業系統在執行程式時為我們配置所需的記憶體空間。

由於這個記憶體空間所在的位置是由作業系統在執行程式時，依當時的情況所決定，我們無法事先指定特定的位置；因此，我們必須為這個記憶體空間取一個用以識別的名字，以便在後續的程式碼中使用保存在這個位置裡面的資料內容。另外，我們也必須提供相關的資訊給作業系統，以便讓它能為我們配置適當的空間（也就是我們必須提供所需要用以儲存資料的空間大小為何）。我們當然無法預知使用者會輸入什麼資料，但是對於所輸入的資料之型態卻可以先加以規範。換句話說，我們無法知道使用者所輸入的體重是多少，但我們卻可以事先知道這將會是一個數值型態的資料；更具體來說，使用者所輸入的體重應該會是一個「浮點數（Floating Number）」[2]。綜合上述的討論，為了要能夠在程式中使用一個記憶體空間來保存資料，且在後續的程式碼中使用該空間中的資料，我們必須在程式中先說明以下的事項：

1. 該空間所欲儲存的資料型態；
2. 用以識別該空間的名稱

以我們打算開發的BMI計算程式為例，我們必須取得使用者所輸入的體重（單位為公斤）與身高（單位為公尺），所以需要使用兩個記憶體空間來存放它們。依據BMI計算程式的需求，我們可以假設體重與身高都會是浮點數，所以必須在程式中先說明我們需要兩個用以存放浮點數的記憶體空間，並為它們分別取個用以識別的名稱。要注意的是，由於C++程式中用以識別的名稱不能使用中文，所以我們通常會使用具有意義的英文來做為識別的名稱，例如我們可以使用「weight」與「height」[3]做為這兩個記憶體空間的識別名稱 – 又稱為「識別字(Identifier)」。

在C++語言的程式中，像這樣用來保存資料的記憶體空間被稱為「變數（Variable）」，必須先在程式中說明其名稱與型態，才能讓作業系統為我們配置適當大小的記憶體空間。請參考以下的程式碼：

```
float weight;
float height;
```

這兩行程式碼被稱為「變數宣告（Variable Declaration）」，用以表示我們需要兩個名稱分別為「weight」與「height」的變數，且它們的型態皆為「float」，也就是C++語言中所提供的浮點數型態。另外，這兩行程式碼也是所謂的「宣告敘述（Declaration Statement）」，也必須使用分號「;」結尾。C++語言是使用連續的4個位元組的記憶體空間來存放一個被宣告為float（浮點數）的數值資料。關於C++語言所支援的資料型態與其所需空間的詳細說明，請參考本書第4章。當程式執行時，上述兩行變數宣告的程式碼，將會在記憶體內為我們配置兩個大小足以放置浮點數的記憶體空間（也就是連續的4個位元組），如figure 4所示：

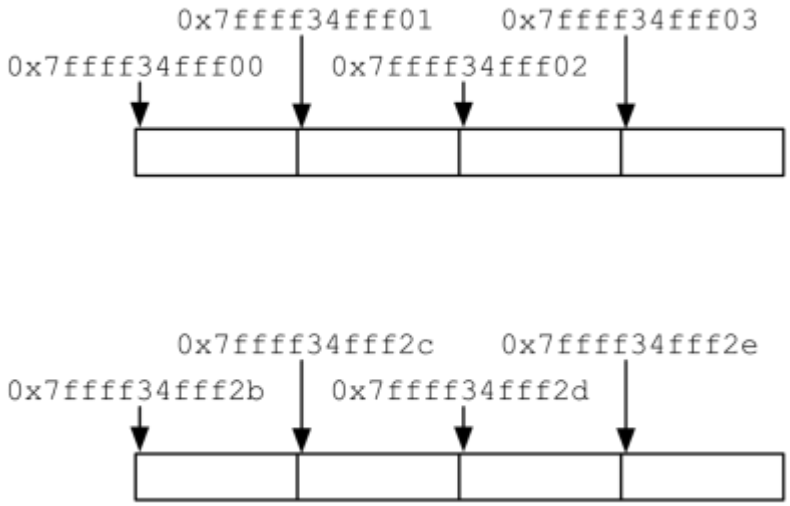

Fig. 4: 由變數宣告自動配置的兩個記憶體空間。

在figure 4中顯示了這兩個記憶體空間所配置到的記憶體位址，分別是位於0x7ffff34fff00至0x7ffff34fff03位址與0x7ffff34fff2b至0x7ffff34fff2e位址的兩個連續4個位元組位址。現代的作業系統大部份都是使用48位元的「記憶體位址（Memory Address）」來為每個位元組（Byte）編號，以圖2-4所顯示的記憶體位址（0x7ffff34fff00）為例，「0x」開頭表示該位址以16進制表示，其後接續的「7ffff34fff00」就是該位址的16進制值。要注意的是，一個記憶體位址僅表示一個位元組（Byte）的位址，由於 C++語言是使用連續的4個位元組的記憶體空間來存放一個浮點數（也就是型態為float的數值資料），也因此在figure 4中，這兩個變數所配置到的記憶體空間，分別是從「0x7ffff34fff00」至「0x7ffff34fff03」以及「0x7ffff34fff2b」至「0x7ffff34fff2e」的兩個連續4個位元組的空間。figure 4所顯示的記憶體配置，也可以使用figure 5的方式

來表達，其意義是相同的，只是採用橫式或直式來表達連續空間的差異而已，本書後續將視情況交替使用這兩種表達方式。

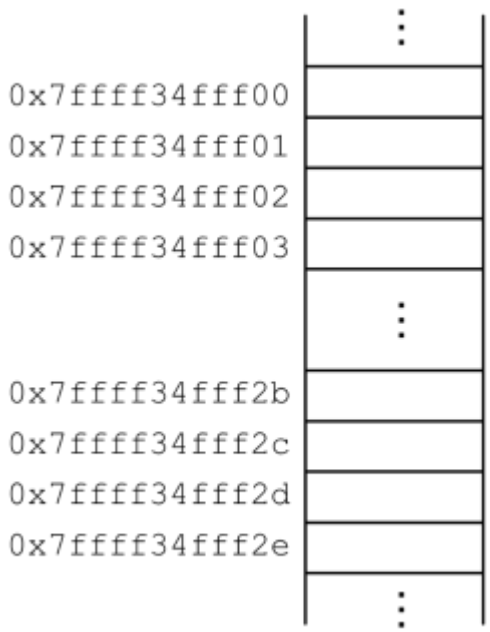

Fig. 5: 由變數宣告自動配置的兩個記憶體空間（以直式表達）。

就像是門牌號碼一樣，我們必須為記憶體內的每個位元組編號，才能夠去指定存取在記憶體中的特定位置。現代的作業系統大多都使用48位元的記憶體位址，來為每個位元組編號。由於48位元可用以表示不重複的281,474,976,710,656（也就是2^{48}）個數值，因此採用48位元的記憶體位址，將可以指定在256TB中的每個位元組。以目前資訊系統發展的脈絡來看，個人電腦還停留在搭配4G□8G或16G記憶體的情況下，這種48位元的記憶體位址，絕對還可以使用上一段很長、很長的時間。

還要特別注意的是，figure 4與figure 5所顯示的記憶體位址僅供參考，實際執行時其位址是由作業系統決定，並不是永遠都會配置到同一個位址[4)]。也正因為相同的理由，我們在後續的程式碼中要使用儲存在這些記憶體空間內的資料時，並不能直接去指定使用□0x7ffff34fff00至0x7ffff34fff03□與「0x7ffff34fff2b至0x7ffff34fff2e□這些位址，因為這些位址必須要等到程式被執行時才能確定，無法事前得知。更好的解決方法是使用當初在宣告時所給定的變數名稱(也就是識別字)來存取它們，例如本例中的□weight□與「height□□而非使用那些無法預知的記憶體位址。

到目前為止，其實讀者只需要理解變數其實就是對映到記憶體內的空間 – 程式執行時就是透過這些分配給變數的記憶體空間來存放資料以及進行各式的運算。但是未來隨著讀者們更深入學習程式設計的觀念與技術時，變術與記憶體間的關係將會變得更為關鍵，我們將在未來的相關章節內容中逐步為你披露。現在，我們可以先用一種較為簡單的方式，來思考變數在程式中所扮演的角色與功用，請參考figure 6，我們可以將變數所配置到的記憶體空間，視為是一個可以存放物品的盒子一樣，位來我們可以將資料存放進去，或是從中取回資料。

Fig. 6: 變數與記憶體空間配置概念圖。

相信經過上述的說明與討論，您應該已經大致瞭解變數的概念(關於變數更詳細的說明，請參考本書第4章)。現在讓我們接著來完成使用□cin□物件來取得使用者所輸入的資料，並將其存放於變數之中。請參考以下

兩行程式碼：

```
cin >> weight;
cin >> height;
```

這兩行程式碼的作用是使用cin物件來將透過stdin所取得的資料（也就是使用者從鍵盤所輸入的資料）存放到weight與height變數中。要特別注意的是，我們是使用「»」運算子，來指定資料流動的方向。以□cin » weight;□這行程式碼為例，其代表的是將資料往weight變數的方向流動，因此來自於鍵盤的資料才能夠存放到weight變數裡。如同使用cout的程式碼一樣，這兩行程式碼也是所謂的「運算敘述□Expression Statement□□□必須使用「;」做為結尾，不然在編譯時將會發生錯誤。

假設使用者所輸入的體重（單位為公斤）與身高（單位為公尺）分別為65.5與1.72，figure 7顯示了使用cin物件將它們存放在變數的過程。

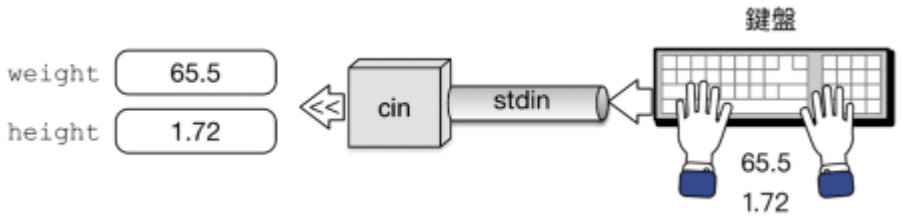

Fig. 7: 取得來自鍵盤的使用者輸入，並存放於變數中。

至此，我們完成了□Input階段」的目標，取回使用者所輸入的資料並存放於變數之中。本節後續將繼續說明IPO模型的下一個階段。

3.1.2 Proces階段與運算式

Process階段的工作就是將Input階段所取回的資料進行後續的處理，以得到有意義的運算結果。以BMI計算程式為例，此階段的工作就是要利用前一階段（也就是Input階段）所取得的weight與height變數，計算出其BMI值。BMI的計算公式是「體重除以身高的平方」，我們可以使用C++語言的算術運算敘述□Arithmetic Expression Statement□來完成此計算。

算術運算敘述係指由算術運算式□Arithmetic Exprssion□所組成的程式敘述，可以由包含加（＋）、減（－）、乘（×）、除（÷）在內的運算子來進行數值資料的算術運算，並可以使用括號來改變運算的優先順序。要注意的是，由於鍵盤沒有提供乘法（×）與除法（÷）這兩個符號，因此C++語言使用星號「*」與斜線「/」來代替乘法與除法。關於算術運算式將在本書第5章提供更完整的說明，不過要先提醒讀者注意的是，算術運算敘述也是敘述，也必須使用分號「；」做為結尾，因此可別忘了加上分號，否則就會產生編譯時的錯誤。

現在，讓我們先使用以下的敘述來建立一個名為BMI的變數（用以存放計算結果），以及用以完成計算的運算敘述如下：

```
float BMI;
BMI □ weight / ( height * height );
```

上述程式碼的第一行是一個宣告敘述，它會在記憶體中，建立一個可以存放浮點數的空間，其名稱為BMI。當然，我們也可以簡單地將其解讀為「宣告一個名為BMI的浮點數變數」，以便供後續的程式碼使用。接下來的程式碼是一個算術運算敘述，將BMI的值依「體重除以身高的平方」的公式完成計算。在這個運算敘述中，C++語言會先進行在等號「＝」右邊的運算，待整個右邊的計算都完成後，再把其運算結果指定給等號左方的變數。在此我們要提醒讀者，在這個運算敘述中的等號「=」與數學領域的等號意義並不相同；它並不是「相等」的意思，而是「指定」的意思 — 將等號右邊的運算結果指定給等號左邊的變數。具體來說，這個算術運算敘述會先將(weight / (height * height)的結果計算出來，再將其結果設定為BMI變數的數值。另一方面，為了降低學習的困難度，我們先使用「身高乘身高(height*height)來代替「身高的平方(height<sup>2<sup>) 後續再讓體重weight除以括號裡面的height*height的結果，如此就可以完成BMI值的計算。

注意：括號與運算順序

在算術運算敘述中的括號是很重要的，它表明了運算的順序。以(BMI = weight / (height * height);)為例，如果少了這組括號，此算術運算敘述將會變成：

```
BMI = weight / height * height;
```

看起來似乎是一樣的，但其執行結果將會變成先執行weight / height的部份，然後再乘上height(如果以括號來註明其運算的優先順序，這個算術運算敘述等同於：

```
BMI = ( weight / height ) * height;
```

由於任何數字除以x再乘上x(其值將等於其本身；因此上述敘述的運算結果將會是weight(是一個完全不正確的結果。關於算術運算敘述中的優先順序，我們將會在本書第5章提供更完整的說明。

3.1.3 Output階段與cout物件

前兩個階段的工作完成後，我們已經順利地取得使用者所輸入的體重與身高，並且使用運算敘述來得到BMI的運算結果。在最後的一個階段（也就是Output輸出階段），我們所要完成的工作就是將BMI的運算結果輸出。我們可以使用在Hello <nowiki>C++</nowiki>章中所介紹過的cout物件來完成，請參考以下的程式碼：

```
cout << BMI << endl;
```

這一行程式的作用是將BMI變數的內容輸出到Console模式的文字界面中，並且再將游標加以換行。至此，我們就完成了BMI計算程式所需要執行的所有工作，包含了資料的輸入、處理以及輸出。

3.2 IPO程式設計

經過上一節的討論後，我們已經大致瞭解了IPO模型以及其每個階段相關的C++語言程式設計方法，本節將統整相關的方法並以實際的程式範例，來說明如何配合IPO模型完成對應的程式設計工作。

3.2.1 IPO程式設計框架

大部份簡易的C++程式都可以適用IPO模型，來完成其對應的程式設計工作。本節提供一個適用此種程式設計的框架（Framework）可用以幫助讀者完成相關的程式設計工作，請參考以下的程式碼：

Example 1

```
// 函式標頭檔區（Header File Inclusion Section）
#include <iostream>

// 命名空間區（Namespace Declaration Section）
using namespace std;

// 程式進入點（Entry Point）
int main()
{
 // 變數宣告區（Variable Declaration Section）

 // 輸入階段區（Input Section）
 // 處理階段區（Process Section）

 // 輸出階段區（Output Section）

   return 0;
}
```

我們把Example 1的IPOFramework.cpp程式碼稱為「IPO程式設計框架」，它事實上就是一個C++程式的雛形，其中不但包含了程式的進入點，也包含了相關所需的載入檔及命名空間的宣告敘述。對於C++語言的初學者來說，可以先透過這個框架來完成許多程式的設計。具體來說，當您需要開發一個簡單的C++程式時，可以先依照IPOFramework.cpp的內容來撰寫程式碼，然後使用IPO模型來構思程式相關的輸入、處理與輸出階段所要完成的工作；最後只要依照C++語言的語法，使用相關的敘述來完成程式設計即可完成程式的開發。我們將框架中的各個分區說明如下：

- 函式標頭檔區（Header File Inclusion Section）：使用「#include<>」來載入程式所需的標頭檔。許多程式（尤其是符合IPO模型的程式）都會需要的與標準輸入輸出相關的「iostream」標頭檔，因此此框架預設會將此要頭檔載入。
- 命名空間區（Namespace Declaration Section）：宣告在程式中需要使用到的命名空間。由於IPO模型與資料的標準輸入、輸出緊密相關，因此使用「std」這個命名空間，也是在此區間中應包含的宣告。

所以using namespace std;是本區的預設程式敘述。

- 程式進入點（Entry Point）：定義main()函式，以做為程式的進入點（也就是開始執行的地方），在其大括號所包裹的範圍內，還需要完成以下四個分區的程式設計：

- 變數宣告區（Variable Declaration Section）：將程式中所有會使用到的變數加以宣告。
- 輸入階段區（Input Section）：使用cin物件來取得使用者的輸入，並將其存放於特定變數中。
- 處理階段區（Process Section）：使用變數的內容來完成程式所需的處理。
- 輸出階段區（Output Section）：使用cout物件來將處理完成的資料加以輸出。

3.2.2 BMI計算程式

現在，讓我們以BMI計算程式為例，示範如何使用IPO模型與IPO程式設計框架，來完成C++語言的程式設計。BMI計算程式是接收使用者所輸入的體重（公斤）與身高（公尺）後，依據公式（體重除以身高的平方）計算其BMI值後加以輸出。使用IPO模型來進行程式設計，其實一點都不困難，但必須循序漸進，先由較抽象的概念開始，再逐步拓展到撰寫C++語言的程式碼。

首先，讓我們先回顧一下table 1的BMI計算程式IPO表（為便利起見，再次列示如下）：

Input輸入階段	**Process處理階段**	**Output輸出階段**
取得使用者所輸入的體重與身高。	依據「體重除以身高的平方」公式計算BMI值。	將計算完成的結果加以輸出。

Tab. 1: BMI計算程式的IPO表

然而此處所描述的內容還不夠具體，還無法做為程式設計之用。我們可以把與BMI計算程式相關的變數名稱、計算方法以及所欲輸出的變數等細節，加入到IPO模型之中，使其工作描述變得更為具體一些，如table 3所示：

Input輸入階段	**Process處理階段**	**Output輸出階段**
取得使用者所輸入的體重與身高，並分別存放於weight與height變數中。	依據weight / (height * height)公式計算BMI值。	將計算完成的BMI值加以輸出。

Tab. 3: BMI計算程式的IPO表（加入輸入、輸出的變數與運算式）

由於最終是希望能以IPO模型來進行程式設計，何不在此時就把我們在IPO模型小節裡所學到的程式碼寫在IPO表中呢？請參考table 4。

Input輸入階段	**Process處理階段**	**Output輸出階段**
取得使用者所輸入的體重與身高，並分別存放於weight與height變數中。 cin » weight; cin » height;	依據weight / (height * height)公式計算BMI值。 BMI = weight / (height * height);	將計算完成的BMI值加以輸出。 cout « BMI « endl;

Tab. 4: BMI計算程式的IPO表（加入符合<nowiki

至此，這個IPO表已經趨於完善，BMI計算程式的設計也大致完成，我們可以開始使用此IPO表來進行程式的開發。在本章後續的內容中，我們將持續地開發這個BMI程式的多個版本，直到它具備所有所需的功能為止。首先請參考Example 2的BMI-1.cpp程式，代表我們的第一次嘗試（後續還會有BMI-2.cpp，BMI-3.cpp等版本），此版本以IPO程式設計框架做為基礎，將table 4中所設計的程式敘述加入到對應的區塊中：

Example 2

```
//函式標頭檔區(Header File Inclusion Section)
#include <iostream>

//命名空間區(Namespace Declaration Section)
using namespace std;

//程式進入點(Entry Point)
int main()
{
 //變數宣告區(Variable Declaration Section)

 //輸入階段區(Input Section)
   cin >> weight;
   cin >> height;

 //處理階段區(Process Section)
   BMI = weight / ( height * height );

 //輸出階段區(Output Section)
   cout << BMI << endl;
   return 0;
}
```

請趕快試試將BMI-1.cpp加以編譯，看看它的執行結果為何吧！？ 如果忘了如何編譯程式，可以參考附錄A 如何編譯與執行範例程式，以

```
[user@urlinux ch3]$ <nowiki>C++</nowiki> BMI-1.cpp
<nowiki>C++</nowiki> BMI-1.cpp
BMI-1.cpp:13:11: error: use of undeclared identifier 'weight'; did you mean
'right'?
   cin >> weight;
          ^~~~~~
          right
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/<nowiki>C++<
/nowiki>/v1/ios:976:1: note: 'right' declared here
right(ios_base& __str)
^
BMI-1.cpp:14:11: error: use of undeclared identifier 'height'; did you mean
'right'?
   cin >> height;
          ^~~~~~
          right
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/<nowiki>C++<
/nowiki>/v1/ios:976:1: note: 'right' declared here
```

```
right(ios_base& __str)
^
BMI-1.cpp:17:4: error: use of undeclared identifier 'BMI'
   BMI = weight / ( height * height );
   ^
BMI-1.cpp:17:21: error: use of undeclared identifier 'height'; did you mean
'right'?
   BMI = weight / ( height * height );
                    ^~~~~~
                    right
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/<nowiki>C++<
/nowiki>/v1/ios:976:1: note: 'right' declared here
right(ios_base& __str)
^
BMI-1.cpp:17:10: error: use of undeclared identifier 'weight'
   BMI = weight / ( height * height );
         ^
BMI-1.cpp:17:30: error: use of undeclared identifier 'height'
   BMI = weight / ( height * height );
                             ^
BMI-1.cpp:20:12: error: use of undeclared identifier 'BMI'
   cout << BMI << endl;
           ^
7 errors generated.
[user@urlinux ch3]$
```

嗯… 有沒有發現BMI-1.cpp其實根本無法執行！這是因為它還缺少了一個非常重要的元素，使其無法通過編譯！有沒有想到缺少的是什麼？如果您沒能發現，這是一個很好的學習機會，請仔細地看看編譯器所給您的錯誤訊息，有沒有發現類似以下的內容 ：

```
BMI-1.cpp:13:11: error: use of undeclared identifier 'weight';
…
BMI-1.cpp:14:11: error: use of undeclared identifier 'height';
…
BMI-1.cpp:17:4: error: use of undeclared identifier 'BMI';
…
```

一個程式往往會因為某處的錯誤，導致編譯器產生一連串的錯誤訊息，只要先將其錯誤的源頭加以修正，就可以將相關的錯誤一併更正。在此我們僅列示出最關鍵的錯誤源頭，有沒有其中有著這樣的錯誤訊息[5]— error: use of undeclared indentifier其意思是「錯誤：使用了未經宣告的識別字[6]」。經過這樣的解釋，相信讀者們應該都已經知道BMI-1.cpp錯誤的原因在哪了吧？沒錯，因為我們缺少了變數的宣告，所以包含weight、height與BMI在內的三個變數名稱，對編譯器來說都只是沒有意義的符號而已。請將這樣的錯誤訊息與其更正方式熟記起來，相信日後您再看到類似的編譯錯誤訊息時，應該就知道該如何解決了！像這樣找出程式錯誤所在並加以更正的過程被稱為「除錯（Debug）」是程式設計師最為重要的能力之一；畢竟「人非聖賢，孰能無過，過而能改，善莫大焉」，套用在程式設計師身上，這句話就變成「程式設計師只是凡人，還是會（常常）寫出有錯誤的程式，但能夠找出錯誤所在並加以更正，這就是程式設計的最高境界了」！

Example 3的BMI-2.cpp程式就是BMI-1.cpp改正後的正確版本，不過建議您先自己試著將BMI-1.cpp的程式改正，並加以編譯直到通過為止，這樣才能培養您為程式除錯的能力！

Example 3

```
// 函式標頭檔區（Header File Inclusion Section）
#include <iostream>

// 命名空間區（Namespace Declaration Section）
using namespace std;

// 程式進入點（Entry Point）
int main()
{
 // 變數宣告區（Variable Declaration Section）
    float weight;
    float height;
    float BMI;

 // 輸入階段區（Input Section）
    cin >> weight;
    cin >> height;

 // 處理階段區（Process Section）
    BMI = weight / ( height * height );

 // 輸出階段區（Output Section）
    cout << BMI << endl;
    return 0;
}
```

相信您已經完成這個程式的編譯了，現在請試著執行看看其結果為何？它應該有以下的執行結果（注意，此結果僅供參考，實際輸出結果會隨輸入資料的不同而有所差異）：

```
[user@urlinux ch3]$ <nowiki>C++</nowiki> BMI-2.cpp
[user@urlinux ch3]$ ./a.out
66.5
1.72
22.4784
[user@urlinux ch3]$
[user@urlinux ch3]$
```

在上面的執行結果中，使用者輸入了他的體重與身高，分別是66.5公斤與1.72公尺，而程式計算後輸出的BMI值為22.1403。

但是這個程式在執行時，可能會讓有些讀者不知所措，因為此程式從進入點（也就是main()函式）開始執行後，會先執行第11-13行的變數宣告（幫我們為變數配置記憶體空間），然後就會執行到第16行等待使用者輸入體重資訊，可是對於不知情的使用者來說，這個程式就像是「當」掉[7)]了一樣，沒有任何回應！像這樣的程式，除了設計者本身以外，其餘的使用者可能都不知道程式是在等待資料的輸入，而且也不知道所要輸入的是體重與身高的資料！就算知道，可能也不知道體重與身高分別是以公斤與公尺做為單位！。

有鑑於此，我們通常會在使用cin物件取得輸入前，加入一行由cout物件所印出的字串，利用這個字串的內容來提示使用者該做些什麼、或是該輸入什麼樣式的資料。請試著將原本在第16行與第17行的程式碼，修改如下：

```
 // 輸入階段區（Input Section）
    cout << "請輸入您的體重(公斤):";
    cin >> weight;
    cout << "請輸入您的身高(公尺):";
    cin >> height;
```

如此一來，當程式在執行時，就會在等待使用者輸入資料前，透過「請輸入您的體重(公斤):」以及「請輸入您的身高(公尺):」的幫助，讓使用者瞭解此時電腦在等待體重與身高資料的輸入，而且其單位為公斤與公尺！我們將這種用以提醒使用者的字串稱為「提示字串（Prompt String）」，其字串內容的輸出通常不會以endl結尾，好讓使用者可以在提示字串後接者輸入資料。

這樣的修改還不足夠，因為最後所輸出的計算結果對不知情的使用者來說，仍然不具有意義（使用者只會看到第23行輸出的一個數字而已，並無法瞭解該數字的意義為何？）！因此，請將第23行修改如下：

```
 // 輸出階段區（Output Section）
    cout << "您的BMI值為" << BMI << endl;
    return 0;
```

這行修改後的程式，是將字串「您的BMI值為」、BMI變數的值以及一個換行，使用「«」運算子讓它們流向cout物件（再透過stdout渠道），將計算結果顯示給使用者。現在，請您將Example 3的BMI-2.cpp原始程式，進行上述的修改並完成編譯。如果您在修改時遇到任何問題，可以參考Example 4所提供的最終修改完的版本 — BMI-3.cpp。

Example 4

```
// 函式標頭檔區（Header File Inclusion Section）
#include <iostream>

// 命名空間區（Namespace Declaration Section）
using namespace std;

// 程式進入點（Entry Point）
int main()
{
 // 變數宣告區（Variable Declaration Section）
    float weight;
    float height;
    float BMI;

 // 輸入階段區（Input Section）
```

```
    cout << "請輸入您的體重(公斤):";
    cin >> weight;
    cout << "請輸入您的身高(公尺):";
    cin >> height;

  // 處理階段區（Process Section）
    BMI = weight / ( height * height );

  // 輸出階段區（Output Section）
    cout << "您的BMI值為" << BMI << endl;
    return 0;
}
```

Example 4的BMI-3.cpp之執行結果如下（注意，此結果僅供參考，實際輸出結果會隨輸入資料的不同而有所差異）：

```
請輸入您的體重(公斤): 45
請輸入您的身高(公尺): 1.56
您的BMI值為18.4911
```

至此，我們終於使用C++語言完成了一個「完整」的BMI計算程式的開發。要注意的是，在此處我們故意輸入了另外一組體重與身高，程式也就依據這組輸入計算出了對應的BMI值。還有一點要注意的是，此處我們所輸入的體重是整數的45公斤，但用以保存體重的變數weight卻是被宣告為float（也就是浮點數型態）。在此種情況下（C++語言會將這個整數值45，轉換為浮點數的45.0後再存放到weight變數裡。換句話說（C++語言會在取得使用者所輸入的資料後，自動進行資料型態的轉換，以便讓資料內容能夠符合變數所宣告的型態[8]）。

3.2.3 門號違約金計算程式

2018年5月，由中華電信開頭的499吃到飽方案，帶動了全台一波全民申辦熱潮。本節將設計一個手機門號違約金計算程式，接收使用者所輸入的現有合約資訊，然後依合約剩餘期間計算其解約應繳之違約金額。使用者所需輸入的資訊如下：

- 合約總日數：原合約期間，以日為單位。
- 合約剩餘日數：扣除已履行合約期間後，所剩餘未履約的日數。
- 月租費優惠：在合約期間內，每月月租費優惠金額，以新台幣元為單位。
- 手機補貼款：綁約購買手機價與單購手機價之差額，以新台幣元為單位。

提前解約的違約金之計算，是以合約期間已享有之各項優惠總額（已履約期間月租費的優惠總額，再加上當初購買手機時的補貼款）乘上合約未履行的比率，其公式如下：

$違約金=((\frac{月租費優惠}{30天})\times (合約總日數-合約剩餘日數)+手機補貼款)\times\frac{合約剩餘日數}{合約總日數}$

由於撰寫程式碼時，上述的公式必須使用變數名稱以及加、減、乘、除等運算子來寫成C++語言的敘述，因此我們先將上述公式以及後續程式中可能會使用到的變數命名如下：

- compensation（違約金

- contractDays：合約總日數
- contractRemainingDays：合約剩餘日數
- monthlyFeeDiscount：每月月租費優惠
- subsidy：手機補貼款

上述這些變數的名稱，是依據其變數的英文字義或詞義加以命名的，原則上全部使用小寫字母，但自第二個單字起，每個單字的第一個字母採用大寫。其實此種變數命名的方法被稱為是「駝峰命名法（Camel Case Convention）」是業界慣用的變數命名方法，我們將在本書下一章變數、常數與資料型態中詳細為您說明。

完成變數命名後，我們就可以將前述的解約金計算公式，改以變數名稱加以定義如下：

$compensation=((\frac{monthlyFeeDiscount}{30})\times (contractDays-contractRemainingDays)+subsidy)\times\frac{contractRemainingDays}{contractDays}$

現在，讓我們使用IPO模型，將這個違約金計算程式表達如figure 8：

Fig. 8: 手機門號違約金計算程式的IPO模型

此IPO模型接收四個使用者輸入的資料，包含contractDays（合約總日數）、contractRemainingDays（合約剩餘日數）、monthlyFeeDiscount（月租費優惠）以及subsidy（手機補貼款），然後進行違約金的計算後加以輸出。我們也提供此程式的IPO表如table 5：

變數宣告

將輸入、處理與輸出階段會使用到的變數加以宣告，包含違約金（compensation）、合約總日數（contractDays）、合約剩餘日數（contractRemainingDays）、每月月租費優惠（monthlyFeeDiscount）、手機補貼款（subsidy）等變數，其中不論是日數或是金額都是整數型態。

```
int contractDays;
int contractRemainingDays;
int monthlyFeeDiscount;
int subsidy;
int compensation;
```

輸入階段

取得使用者所輸入的contractDays（合約總日數）、contractRemainingDays（合約剩餘日數）、monthlyFeeDiscount（月租費優惠）以及subsidy（手機補貼款）。

```
cout << "請輸入合約總日數:";
cin >> contractDays;
cout << "請輸入合約剩餘日數:";
cin >> contractRemainingDays;
cout << "請輸入合約期間每月月租費優惠金額:";
cin >> monthlyFeeDiscount;
cout << "請輸入手機補貼款金額:";
cin >> subsidy;
```

處理階段

進行違約金的計算，並將計算結果存放於compensation變數。

```
compensation = ((monthlyFeeDiscount/30)*(contractDays-contractRemainingDays)
+subsidy)*(contractRemainingDays/contractDays);
```

輸出階段	將違約金的計算結果加以輸出。 `cout << "您必須支付的違約金額為" << compensation << "元" << endl;`

Tab. 5: 手機違約金計算程式的IPO表

與table 4相比，table 5採用了直式的方式呈現，更重要的是table 5還將「變數宣告」包含了進來，這對於後續開發程式將更有助益。這個違約金計算程式所使用到的變數都是關於「日數」與「金額」，它們的共同點是 — 都是整數（Integer）。C++語言所支援的整數型態為（int）。我們將本例中所有的變數都宣告為（int）整數型態。關於資料型態的更多細節，請參考本書第4章變數、常數與資料型態。

相信讀者應該已經發現，當您在IPO表中放入更多的細節後（也就是放入更多的程式碼），其實已經離真正要完成程式不遠了，剩下的工作只需要在IPO程式設計框架中，將IPO表中的工作描述（特別是程式碼的部份）加入到框架中對應的區塊即可。現在，請試著自己以IPO程式設計框架與table 5的IPO表為基礎，將門號違約金計算程式設計完成。您所完成的程式碼，應該與Example 5的compensation-1.cpp程式類似：

Example 5

```
// 函式標頭檔區（Header File Inclusion Section）
#include <iostream>

// 命名空間區（Namespace Declaration Section）
using namespace std;

// 程式進入點（Entry Point）
int main()
{
 // 變數宣告區（Variable Declaration Section）
   int contractDays;          // 合約總日數
   int contractRemainingDays; // 合約剩餘日數
   int monthlyFeeDiscount;    // 每月月租費優惠金額
   int subsidy;               // 手機補貼款
   int compensation;          // 違約金

 // 輸入階段區（Input Section）
   cout << "請輸入合約總日數:";
   cin >> contractDays;
   cout << "請輸入合約剩餘日數:";
   cin >> contractRemainingDays;
   cout << "請輸入合約期間每月月租費優惠金額:";
   cin >> monthlyFeeDiscount;
   cout << "請輸入手機補貼款金額:";
   cin >> subsidy;

 // 處理階段區（Process Section）
   compensation = ( (monthlyFeeDiscount/30) *
                  (contractDays-contractRemainingDays) + subsidy ) *
```

```
                (contractRemainingDays/contractDays);

 // 輸出階段區（Output Section）
    cout << "您必須支付的違約金額為" << compensation << "元" << endl;
    return 0;
}
```

請將這個程式加以編譯與執行，看看其執行結果是否正確。

如果您自己所設計的程式，與我們在Example 5所提供的compensation-1.cpp完全一樣，那麼在執行時您應該會看到以下的執行結果（注意，此結果僅供參考，實際輸出結果會隨輸入資料的不同而有所差異）：

```
請輸入合約總日數: 720
請輸入合約剩餘日數: 540
請輸入合約期間每月月租費優惠金額: 100
請輸入手機補貼款金額: 2000
您必須支付的違約金額為0元
```

咦～ 怎麼會輸出0元呢？依據上述的使用者輸入與計算公式，違約金的計算結果應該是1950元：

（（100/30）× (720-540)+2000）×540/720）（（3.33×180+2000）×0.75）（2600×0.75）1950

但是這個看起來正確的程式，卻輸出了「您必須支付的違約金額為0元」！其實這個程式的錯誤是在於除法的運算出了問題（Example 6提供了一個簡單的測試：

Example 6

```
// 函式標頭檔區（Header File Inclusion Section）
#include <iostream>

// 命名空間區（Namespace Declaration Section）
using namespace std;

// 程式進入點（Entry Point）
int main()
{
 // 變數宣告區（Variable Declaration Section）
    int contractDays;          // 合約總日數
    int contractRemainingDays; // 合約剩餘日數
    float incompleteRatio; // 合約未完成比率

 // 輸入階段區（Input Section）
    cout << "請輸入合約總日數:";
    cin >> contractDays;
    cout << "請輸入合約剩餘日數:";
    cin >> contractRemainingDays;

 // 處理階段區（Process Section）
    incompleteRatio = contractRemainingDays/contractDays;
```

```
 //輸出階段區（Output Section）
   cout << "您的合約還有" << incompleteRatio << "未完成" << endl;
   return 0;
}
```

在這個測試程式中，我們新增了一個（incompleteRatio）變數，做為合約未完成的比率（這也是在違約金計算公式中的最後一個項目），這個數值應該會是一個介於0至1之間的浮點數，所以我們將它宣告為float型態，如第13行所示。此程式在第22行讓contractRemainingDays除以contractDays（以得到incompleteRatio的值；並在第25行將結果加以輸出。

如果合約總日數與合約剩餘日數分別輸入為720與540，則contractRemainingDays/contractDays的計算結果應該是0.75，但是這個測試程式的執行卻是以下的結果：

```
請輸入合約總日數: 720
請輸入合約剩餘日數: 540
您的合約還有0未完成
```

請仔細看看第22行的程式碼，我們將變數的型態標記在下方：

```
incompleteRatio = contractRemainingDays/contractDays;
   (float)             (int)              (int)
```

我們在本章前面已經說明過，此行程式碼在執行時，會先將等號右方的部份加以計算，然後把計算的結果指定做為等號左方變數的數值內容。由於等號右方進行的是一個兩個整數的除法（C++語言預設會將計算的結果視為是一個整數，因此當整數的540/720時，雖然其計算結果應該是0.75，但C++語言僅將其整數的部份保存起來，小數部份卻被無條件捨棄了。所以此處的0.75會被視為是整數0（incompleteRatio變數的值也就因此成為了0。

經過上述的討論，您應該已經可以瞭解為何我們以為是正確的compensation-1.cpp程式，在執行時卻會將違約金計算為0元！因為在違約金時，其計算公式最後要乘以合約未完成的比例，但此比例卻被C++語言視為是整數的0，而不是符點數的0.75！由於任何數值乘以0的結果都是0，所以其違約金就變成了0元了！類似的問題，也發生在違約金計算公式中的（monthlyFeeDiscount / 30）（此處原本該計算的是月租費優惠以日計算時的優惠金額，以100元的月租費優惠為例，其日優惠金額應為「100/30＝3.33元」；但是這又是一個整數除整數的情況，實際上它所計算出的來值，也只會剩下整數的部份，也就是「3元」。

既然我們已經知道錯誤的原因了，那麼門號違約金計算程式又該如何修改呢？其實修改的方法有很多種，此處我們先以一種最簡單的方式進行，待本書第5章運算式時，我們會再次針對此問題提供更完整的說明。由於錯誤是肇因於兩個整數相除的關係，因此我們只要將這些變數都改以浮點數的float型態加以宣告，問題就迎刃而解了！請參考以下的Example 7（

Example 7

```
//函式標頭檔區（Header File Inclusion Section）
#include <iostream>
```

```
// 命名空間區（Namespace Declaration Section）
using namespace std;

// 程式進入點（Entry Point）
int main()
{
 // 變數宣告區（Variable Declaration Section）
   float contractDays;         // 合約總日數
   float contractRemainingDays; // 合約剩餘日數
   float monthlyFeeDiscount;   // 每月月租費優惠金額
   float subsidy;              // 手機補貼款
   float compensation;         // 違約金

 // 輸入階段區（Input Section）
   cout << "請輸入合約總日數:";
   cin >> contractDays;
   cout << "請輸入合約剩餘日數:";
   cin >> contractRemainingDays;
   cout << "請輸入合約期間每月月租費優惠金額:";
   cin >> monthlyFeeDiscount;
   cout << "請輸入手機補貼款金額:";
   cin >> subsidy;

 // 處理階段區（Process Section）
   compensation = ( (monthlyFeeDiscount/30) *
                  (contractDays-contractRemainingDays) + subsidy ) *
                  (contractRemainingDays/contractDays);

 // 輸出階段區（Output Section）
   cout << "您必須支付的違約金額為" << compensation << "元" << endl;
   return 0;
}
```

此程式的執行結果如下：

```
請輸入合約總日數: 720
請輸入合約剩餘日數: 540
請輸入合約期間每月月租費優惠金額: 100
請輸入手機補貼款金額: 2000
您必須支付的違約金額為1950元
```

至此，門號違約金計算程式終於完成設計了！希望您透過本章所提供的這些例子，已經可以掌握到IPO程式設計的要領，並且也能夠初步瞭解不同資料型態對於程式執行結果的影響。後續我們將在接下來的兩章中，分別就變數的資料型態，以及其運算結果的差異進行完整的討論，屆時您將可以學到更多相關的知識與程式設計的技巧！

3.3 本章內容回顧

本章介紹了一個簡單（但很有用）的程式開發思維方法 — IPO模型（輸入-處理-輸出模型（Input-Process-Output Model））並使用相關的IPO程式設計框架與IPO表，帶領讀者進行兩個實際的應用程式開發。相信透

過本章的介紹與說明，讀者們對於IPO程式設計方法應該都有所瞭解，並有能力開發簡單的、以資料的輸入、處理和輸出相關的C++語言程式。以下我們為讀者彙整了本章相關的重點：

- IPO模型（IPO Model，全名為Input-Process-Output Model，輸入-處理-輸出模型），是一種將電腦程式表示為資料輸入、處理與輸出等三個階段的思維方法。
- IPO表（IPO Chart，）IPO模型的一種表達方式，以表格列示程式的輸入、處理與輸出的工作描述（或程式碼）。
- IPO程式設計框架（IPO Programming Framework，）符合IPO模型的程式碼範本，雖然是一個正確且可執行的C++程式，但尚不具備任何功能。我們只需要將IPO模型（或IPO表）的分析結果，填入此框架就可以完成程式的開發。
  - Input階段（IPO模型的第一階段，主要的工作是取得使用者所輸入的資料。通常搭配cin物件將使用者輸入取回後存放於變數中。
  - Process階段（IPO模型的第二階段，主要的工作是進行資料處理，其中最常見的處理是進行運算式的計算。
  - Output階段（IPO模型的第三階段，主要的工作是進行資料的輸出，通常會使用cout物件來將處理的結果輸出到Console文字界面中。
- cin物件：透過stdin將使用者從鍵盤所輸入的資料取回，並以「»」指定用以存放資料的變數。
- 變數（Variable，）用以存放資料內容的一塊記憶體空間，其所存放的內容在程式執行過程中可以加以改變，但其型態不能改變。
- 變數名稱（Variable Name，）用以識別變數及存取變數內容的名稱。
- 變數宣告（Variable Declaration，）每個變數在初次使用前都必須先加以宣告，其宣告內容包含變數名稱與資料型態。
- 資料型態（Data Type，）是資料內容的種類定義，用以限制資料內容的範圍。以數值資料為例，可分為整數型態（int）與浮點數型態（float，）整數型態的數值不能包含小數的部份，但浮點數型態的數值可以包含有整數與小數的部份。
  - float（C++語言所支援的浮點數型態。
  - int（C++語言所支援的整數型態。
- 算術運算敘述（Arithmetic Expression Statement，）包含有算術運算式（Arithmetic Expression）的敘述，可以用以進行包含加、減、乘、除在內的算術運算。
  - *：乘法運算子。
  - /：除法運算子。

---

[1] 關於cout的說明可參考本書上一章印出字串的cout物件小節。
[2] 浮點數又稱為實數（Real Number，）是指帶有小數的數值資料。
[3] weight與height分別是英文的體重與身高之意。
[4] 這是因為現代的作業系統使用一種稱為「位址空間組態隨機載入（Address space layout randomization，ASLR）有時又被稱做「位址空間隨機配置」或是「位址空間布局隨機化」）的安全技術，來防範駭客可能利用記憶體漏洞所進行的攻擊。因此，一個程式在執行時，將會被隨機配置到不同的記憶體位置，所以我們每次「看到」的記憶體位址並不會相同。
[5] 由於不同的開發環境所使用的編譯器不盡相同，因此其錯誤訊息也會有一些差異，不過這些由不同編譯器所產生的錯誤訊息，基本上相差不大。不論你所使用的編譯器為何，你應該都能看到類似的錯誤訊息。
[6] 識別字（Identifier）就是在程式碼中的變數或函式名稱。
[7] 「當」一詞代表故障或沒有反應，例如電腦發生某種錯誤導致系統沒有回應的時候，我們會說電腦「當機」了。本例中，使用者可能會誤會程式執行發生了錯誤，也就是這個程式的執行「當」掉了。在英文中，則常以crash代表相同的意思。
[8] 關於此種資料型態自動轉換的更多細節，可參考本書第4章。

From:
https://junwu.nptu.edu.tw/dokuwiki/ - **Jun Wu的教學網頁**
國立屏東大學資訊工程學系
CSIE, NPTU
Total: 174028

Permanent link:
https://junwu.nptu.edu.tw/dokuwiki/doku.php?id=cppbook:ch-ipo

Last update: **2024/02/23 02:53**