

C++ Programming

2023 Spring; week 6

Instructor: Cheng-Chun Chang (張正春)
Department of Electrical Engineering

Textbook: P. Deitel and H. Deitel, C HOW TO PROGRAM 8/E,
PEARSON, 2016

課程助教

協助**dubug**, 禁止同學看**code**照抄

第一排：

第二排：

第三排：

第四排：

第五排：

第六排：

打游擊：

Programming: it's all about format ...analogous to 唐詩宋詞



→低頭吃便當....



→處處蚊子咬....

KEY: 1) 用法 2) 用法 3) 用法, and then you can modify it.

16.6 將類別放在獨立檔案以提高重複使用性

- ▶ 建立類別定義的好處之一，就是若套件設定得當，程式設計者便可重複使用我們的類別。說不定其他程式設計師都可以用。例如，任何C++程式均可重複使用C++標準函式庫的**string**型別，只要在程式中包含標頭檔 **<string>** 即可。



-
- ▶ 但想使用GradeBook類別的程式設計者，無法在別的程式中直接包含圖16.7的檔案。若其它程式設計者把圖16.7的檔案包含進來，程式就有兩個**main**函式了，當編譯器要編譯第二個**main**函式時就會產生錯誤。因此，若將**main**跟類別定義擺在同一個檔案中，其他人就不能再利用此類別了。本節介紹如何將**GradeBook**類別放進獨立檔案中，跟**main**函式隔開，好讓別人再利用。



-
- ▶ 本章前述範例都由一個.cpp檔組成，此檔也叫**原始碼檔案 (source-code file)**，它包含GradeBook類別定義和一個main函式。建構物件導向C++程式時，通常會將可再利用的原始碼(如類別)定義在副檔名為.h的檔案中，這就是**標頭 (header)**。程式使用#include前置處理指令以含入標頭，藉此運用可再利用的軟體元件。



-
- ▶ 下個範例中，我們將圖16.7的程式碼分成兩個檔案--**GradeBook.h** (圖16.9) 與**fig16_10.cpp** (圖16.10)。為了讓您習慣本書後面及業界所碰到的較大程式，我們通常用另一個含**main**函式的獨立檔案來測試我們的類別，這就叫**測試程式 (driver program)**。



Programming codes 1

.h .cpp separation



```
1 // Fig. 16.9: GradeBook.h
2 // GradeBook class definition in a separate file from main.
3 #include <iostream>
4 #include <string> // class GradeBook uses C++ standard string class
5 using namespace std;
6
7 // GradeBook class definition
8 class GradeBook
9 {
10 public:
11     // constructor initializes courseName with string supplied as argument
12     GradeBook( string name )
13     {
14         setCourseName( name ); // call set function to initialize courseName
15     } // end GradeBook constructor
16
17     // function to set the course name
18     void setCourseName( string name )
19     {
20         courseName = name; // store the course name in the object
21     } // end function setCourseName
```

□圖16.9 獨立於main之外的GradeBook類別定義(1/2)



```
22
23 // function to get the course name
24 string getCourseName()
25 {
26     return courseName; // return object's courseName
27 } // end function getCourseName
28
29 // display a welcome message to the GradeBook user
30 void displayMessage()
31 {
32     // call getCourseName to get the courseName
33     cout << "Welcome to the grade book for\n" << getCourseName()
34         << "!" << endl;
35 } // end function displayMessage
36 private:
37     string courseName; // course name for this GradeBook
38 }; // end class GradeBook
```

□圖16.9 獨立於main之外的GradeBook類別定義(2/2)



```

1 // Fig. 16.10: fig16_10.cpp
2 // Including class GradeBook from file GradeBook.h for use in main.
3 #include <iostream>
4 #include "GradeBook.h" // include definition of class GradeBook
5 using namespace std;
6
7 // function main begins program execution
8 int main()
9 {
10     // create two GradeBook objects
11     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
12     GradeBook gradeBook2( "CS102 Data Structures in C++" );
13
14     // display initial value of courseName for each GradeBook
15     cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
16         << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
17         << endl;
18 } // end main

```

```

gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++

```

Programming codes 1-GradeBook.h(1)

// 製作一標頭檔GradeBook.h置入於專案中的標頭檔資料夾中

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
class GradeBook
```

```
{
```

```
public:
```

```
    GradeBook(string name)
```

```
    {
```

```
        setCourseName(name);
```

```
    }
```

```
    void setCourseName(string name)
```

```
    {
```

```
        courseName = name;
```

```
    }
```



Programming codes 1-GradeBook.h(2)

```
string getCourseName()
{
    return courseName;
}

void displayMessage()
{
    cout << "Welcome to the grade book for \n" << getCourseName()
    << "!" << endl;
}

private:
    string courseName;
};
```



Programming codes 1

```
#include <iostream>
#include "GradeBook.h"
using namespace std;

int main()
{
    GradeBook gradebook1("CS101 Introduction to C++ Programming");
    GradeBook gradebook2("CS102 Data Structures in C++");

    cout << "gradebook1 created for course: " << gradebook1.getCourseName()
         << "\ngradebook2 created for course: " << gradebook2.getCourseName()
         << endl;
}
```



-
- ▶ 含入一個內有使用者自訂類別的標頭
 - ▶ **GradeBook.h** (圖16.9) 這樣的標頭不能當作一個完整的程式，因為它沒有 **main** 函式。
 - ▶ 在圖16.10的第4行要求C++前置處理器，在編譯程式**之前**，先將此前置處理指令取代成**GradeBook.h** (也就是**GradeBook**類別定義)的內容。



-
- ▶ 現在，編譯原始碼檔案**fig16_10.cpp**時，它就含有**GradeBook**類別定義了(因為有**#include**)，編譯器也能決定**GradeBook**物件的建立方式，以及是否正確呼叫其成員函式。現在，類別定義放在標頭裡(沒有**main**函式)，**任何**程式都能包含此標頭檔，以重複使用**GradeBook**類別。



▶如何找到標頭

- ▶ 注意，圖16.10中第4行的**GradeBook.h**檔名是以雙引號 (" ") 包起來，而不是箭號 (<>)。程式原始碼檔案與使用者自訂標頭通常會放在相同目錄下。當前置處理器碰到雙引號中的標頭名時，它會在相同目錄中尋找標頭，就跟該檔在**#include**指令中出現的方式一樣。若前置處理器在該目錄中找不到標頭，它會從C++標準函式庫標頭檔所在的目錄中尋找。



▶ 其它軟體工程議題

- ▶ 現在，**GradeBook**類別已定義在標頭中，可被再利用了。不幸的是，將類別定義放在如圖16.9的標頭中，仍會把類別的所有實作暴露給用戶端。因為**GradeBook.h**只是個文字檔，誰都可以讀。
- ▶ 若用戶端知道類別的實作方式，該程式設計者可能就會依照類別實作的細節來寫程式。理想上，若類別實作變更，用戶端不應跟著變。



-
- ▶ 第16.7節介紹如何將**GradeBook**類別分成兩個檔案，以達成下列目的
 - 1.類別可重複使用。
 - 2.用戶端知道該類別提供哪些成員函式、如何呼叫它們，以及傳回型別為何。
 - 3.用戶端**不知道**該類別成員函式的實作方式。



16.7 將介面與實作分開

▶ 類別的介面

- ▶ **介面 (Interface)** 定義一種標準化方式，讓事物 (如人類) 與系統彼此互動。例如，收音機控制器就是使用者與內部元件之間的介面。介面說明了收音機提供「**什麼**」操作，但沒有說明「**如何**」實作這些操作。



-
- ▶ 同樣的，**類別的介面 (interface of a class)** 說明類別提供了「**什麼**」服務給用戶端，以及如何使用這些服務，但沒有說明「**如何**」實作這些服務。類別的**public**介面由類別的**public**成員函式組成，也叫做類別的**public服務 (public services)**。



▶ 將介面與實作分開

- ▶ 為了達到更優良的軟體工程，應將成員函式的定義與類別定義隔開，如此一來，成員函式的實作便能隱藏起來，不讓用戶端看到。
- ▶ 圖16.11-16.13的程式將圖16.9的類別定義分成兩個檔案，以將**GradeBook**的介面與實作分開。標頭**GradeBook.h** (圖16.11) 是**GradeBook**類別的定義，原始碼檔案**GradeBook.cpp** (圖16.12) 則是**GradeBook**成員函式的定義。



▶ **GradeBook.h**: 以函式原型定義類別介面

- ▶ 標頭 **GradeBook.h** (圖 16.11) 又是另一個版本的 **GradeBook** 類別定義 (第 9-18 行)。此版跟圖 16.9 的很像，但圖 16.9 的函式定義則由 **函式原型 (function prototype)** (第 12-15 行) 取代，它描述類別的 **public** 介面，但沒有暴露成員函式的實作。「函式原型」是一種函式的宣告，可告訴編譯器此函式的名稱、傳回型別和參數型別。



▶ **GradeBook.cpp**: 將成員函式定義在另一個獨立的原始碼檔案中

- ▶ 原始碼檔案 **GradeBook.cpp** (圖 16.12) 定義了 **GradeBook** 類別的成員函式，這些成員函式於圖 16.11 的第 12-15 行 **宣告**。第 9-32 行是成員函式的定義，幾乎跟圖 16.9 中第 12-35 行的成員函式定義一模一樣。



Programming codes 2

A good reference template!

Separate .h/.cpp/main.cpp



```
1 // Fig. 16.11: GradeBook.h
2 // GradeBook class definition. This file presents GradeBook's public
3 // interface without revealing the implementations of GradeBook's member
4 // functions, which are defined in GradeBook.cpp.
5 #include <string> // class GradeBook uses C++ standard string class
6 using namespace std;
7
8 // GradeBook class definition
9 class GradeBook
10 {
11 public:
12     GradeBook( string ); // constructor that initializes courseName
13     void setCourseName( string ); // function that sets the course name
14     string getCourseName(); // function that gets the course name
15     void displayMessage(); // function that displays a welcome message
16 private:
17     string courseName; // course name for this GradeBook
18 }; // end class GradeBook
```



```
1 // Fig. 16.12: GradeBook.cpp
2 // GradeBook member-function definitions. This file contains
3 // implementations of the member functions prototyped in GradeBook.h.
4 #include <iostream>
5 #include "GradeBook.h" // include definition of class GradeBook
6 using namespace std;
7
8 // constructor initializes courseName with string supplied as argument
9 GradeBook::GradeBook( string name )
10 {
11     setCourseName( name ); // call set function to initialize courseName
12 } // end GradeBook constructor
13
14 // function to set the course name
15 void GradeBook::setCourseName( string name )
16 {
17     courseName = name; // store the course name in the object
18 } // end function setCourseName
```



```
19
20 // function to get the course name
21 string GradeBook::getCourseName()
22 {
23     return courseName; // return object's courseName
24 } // end function getCourseName
25
26 // display a welcome message to the GradeBook user
27 void GradeBook::displayMessage()
28 {
29     // call getCourseName to get the courseName
30     cout << "Welcome to the grade book for\n" << getCourseName()
31         << "!" << endl;
32 } // end function displayMessage
```



```
1 // Fig. 16.13: fig16_13.cpp
2 // GradeBook class demonstration after separating
3 // its interface from its implementation.
4 #include <iostream>
5 #include "GradeBook.h" // include definition of class GradeBook
6 using namespace std;
7
8 // function main begins program execution
9 int main()
10 {
11     // create two GradeBook objects
12     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
13     GradeBook gradeBook2( "CS102 Data Structures in C++" );
14
15     // display initial value of courseName for each GradeBook
16     cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
17         << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
18         << endl;
19 } // end main
```

```
gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++
```

□圖16.13 **GradeBook**類別測試，此類別已將介面與實作分開

Programming codes 2-GradeBook.h

// 製作一標頭檔GradeBook.h置入於專案中的標頭檔資料夾中

```
#include <string>
```

```
using namespace std;
```

```
class GradeBook
```

```
{
```

```
public:
```

```
    GradeBook(string);
```

```
    void setCourseName(string);
```

```
    string getCourseName();
```

```
    void displayMessage();
```

```
private:
```

```
    string courseName;
```

```
};
```



Programming codes 2-GradeBook.cpp(1)

// 製作一C++(cpp)檔GradeBook.cpp，置入於專案中的來源檔案資料夾中

```
#include <iostream>
```

```
#include "GradeBook.h"
```

```
using namespace std;
```

```
GradeBook::GradeBook(string name)
```

```
{
```

```
    setCourseName(name);
```

```
}
```

```
void GradeBook::setCourseName(string name)
```

```
{
```

```
    courseName = name;
```

```
}
```



Programming codes 2-GradeBook.cpp(2)

```
string GradeBook::getCourseName()
{
    return courseName;
}
```

```
void GradeBook::displayMessage()
{
    cout << "Welcome to the grade book for \n" << getCourseName()
    << "!" << endl;
}
```




Programming codes 2

```
#include <iostream>
#include "GradeBook.h"
using namespace std;

int main()
{
    GradeBook gradebook1("CS101 Introduction to C++ Programming");
    GradeBook gradebook2("CS102 Data Structures in C++");

    cout << "gradebook1 created for course: " << gradebook1.getCourseName()
    << "\ngradebook2 created for course: " << gradebook2.getCourseName()
    << endl;
}
```



-
- ▶ 每個函式標頭的成員函式名稱 (第9、15、21、27行) 前面都有類別名稱和`::`，這叫做「**二元使用域解析運算子 (binary scope resolution operator)**」。它們會把每個成員函式「綁」到宣告成員函式與資料成員的**GradeBook**類別定義 (現在是另一個檔案，圖16.11) 上面。若不在函式名稱前面寫「**GradeBook::**」，編譯器就不知道這些函式是**GradeBook**類別的成員函式，而會把它們當成「自由」或「鬆散」的函式，如**main**。這些函式又稱作全域函式。
-
- 

-
- ▶ 為表示**GradeBook.cpp**裡的成員函式是**GradeBook**類別的一部分，我們必須先包含**GradeBook.h**標頭檔。編譯**GradeBook.cpp**時，編譯器會使用**GradeBook h**中的資訊
 - 1.以確保每個成員函式的第一行(第9、15、21、27行)都符合**GradeBook h**檔中的函式原型。



2.每個成員函式都知道類別的資料成員與其它成員函式。例如，第17和23行可存取**courseName**變數，因為它在**GradeBook.h**宣告為**GradeBook**的資料成員，而第11行與第30行可呼叫**setCourseName**與**getCourseName**函式，因為它們都在**GradeBook.h**中宣告為成員函式。



▶ 測試GradeBook類別

- ▶ 圖16.13所執行的**GradeBook**物件操作與圖16.10相同。將**GradeBook**的介面與成員函式實作分開，並不會影響用戶端使用類別的方式。



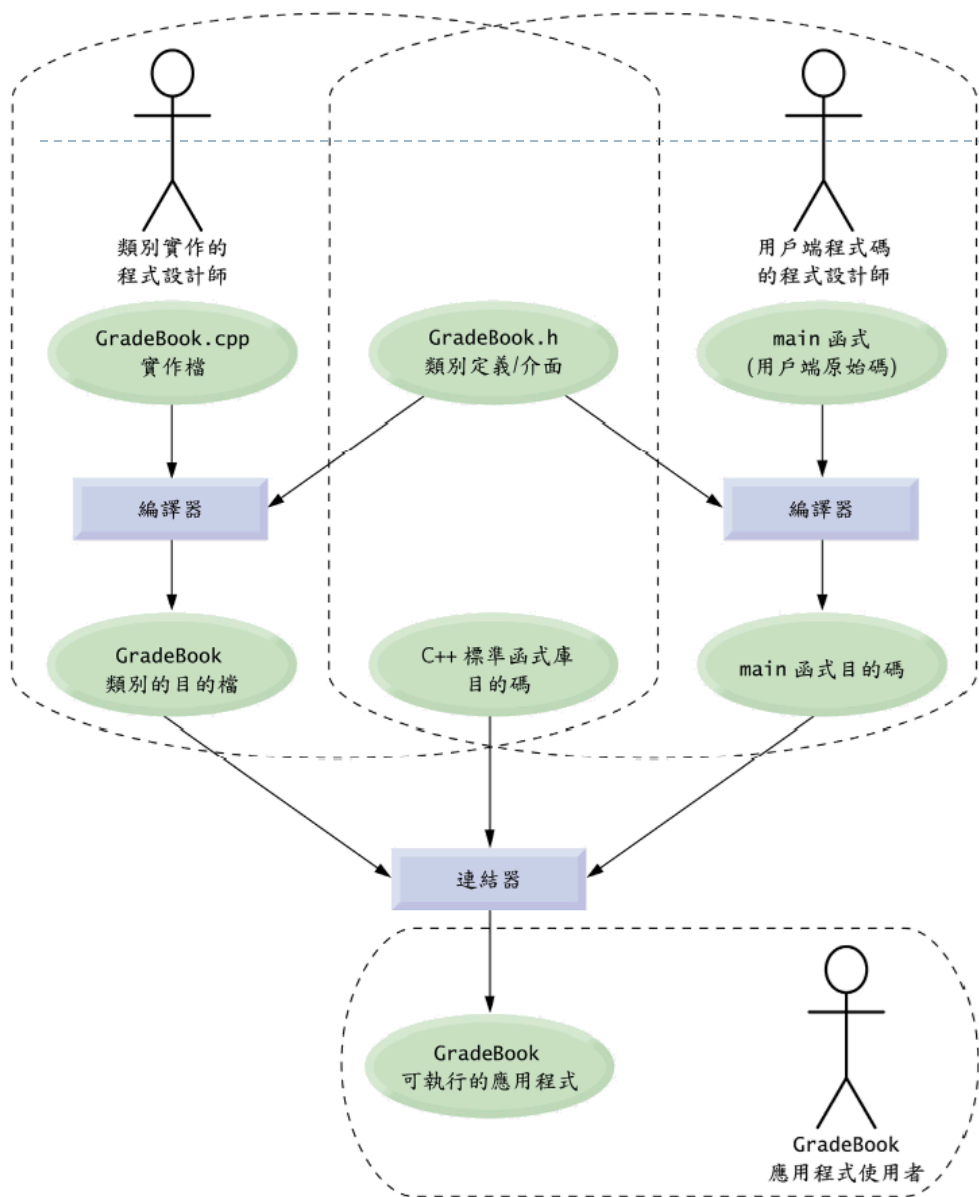


圖16.14 產生可執行程式的編譯與連結程序

16.8 以set函式驗證資料

- ▶ 第16.4節介紹過set函式，可讓用戶端修改**private**資料成員的值。圖16.5中，**GradeBook**類別所定義的成員函式**setCourseName**只是將**name**參數的值設給資料成員**courseName**。此成員函式不會檢查課程名稱是否符合特定格式。我們可能會要求**GradeBook**類別檢查其資料成員**courseName**，確保它不超過25個字元。



-
- ▶ 圖16.15-16.17的程式加強了**GradeBook**成員函式**setCourseName**的功能，讓它**執行驗證 (validation**，也叫**正確性檢查， validity checking)**。



▶GradeBook類別定義

- ▶ 注意，**GradeBook**的類別定義(圖16.15)跟圖16.11完全一樣，所以介面也沒變。因為介面沒變，所以**setCourseName**成員函式修改後，此類別的用戶端不須跟著改。用戶端只要將用戶端的目的碼連結更新過的**GradeBook**目的碼，就能享受**GradeBook**類別的增強功能。



-
- ▶ 以GradeBook的成員函式setCourseName驗證課程名稱
 - ▶ **GradeBook**類別的增強功能寫在setCourseName的定義中(圖16.16的第16-29行)。第18-19行的**if**敘述判斷**name**參數是否為有效的課程名稱(也就是小於25個字元的**string**)。若課程名稱有效，第19行就把課程名稱存入資料成員**courseName**。



Programming codes 3



```
1  // Fig. 16.15: GradeBook.h
2  // GradeBook class definition presents the public interface of
3  // the class. Member-function definitions appear in GradeBook.cpp.
4  #include <string> // program uses C++ standard string class
5  using namespace std;
6
7  // GradeBook class definition
8  class GradeBook
9  {
10 public:
11     GradeBook( string ); // constructor that initializes a GradeBook object
12     void setCourseName( string ); // function that sets the course name
13     string getCourseName(); // function that gets the course name
14     void displayMessage(); // function that displays a welcome message
15 private:
16     string courseName; // course name for this GradeBook
17 }; // end class GradeBook
```

```
1 // Fig. 16.16: GradeBook.cpp
2 // Implementations of the GradeBook member-function definitions.
3 // The setCourseName function performs validation.
4 #include <iostream>
5 #include "GradeBook.h" // include definition of class GradeBook
6 using namespace std;
7
8 // constructor initializes courseName with string supplied as argument
9 GradeBook::GradeBook( string name )
10 {
11     setCourseName( name ); // validate and store courseName
12 } // end GradeBook constructor
13
14 // function that sets the course name;
15 // ensures that the course name has at most 25 characters
16 void GradeBook::setCourseName( string name )
17 {
18     if ( name.length() <= 25 ) // if name has 25 or fewer characters
19         courseName = name; // store the course name in the object
```

圖16.16 GradeBook類別的成員函式定義，其set函式會驗證資料成員courseName的長度(1/2)



```

20
21     if ( name.length() > 25 ) // if name has more than 25 characters
22     {
23         // set courseName to first 25 characters of parameter name
24         courseName = name.substr( 0, 25 ); // start at 0, length of 25
25
26         cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
27             << "Limiting courseName to first 25 characters.\n" << endl;
28     } // end if
29 } // end function setCourseName
30
31 // function to get the course name
32 string GradeBook::getCourseName()
33 {
34     return courseName; // return object's courseName
35 } // end function getCourseName
36
37 // display a welcome message to the GradeBook user
38 void GradeBook::displayMessage()
39 {
40     // call getCourseName to get the courseName
41     cout << "Welcome to the grade book for\n" << getCourseName()
42         << "!" << endl;
43 } // end function displayMessage

```

□圖16.16 **GradeBook**類別的成員函式定義，其**set**函式會驗證資料成員**courseName**的長度(2/2)

```
1 // Fig. 16.17: fig16_17.cpp
2 // Create and manipulate a GradeBook object; illustrate validation.
3 #include <iostream>
4 #include "GradeBook.h" // include definition of class GradeBook
5 using namespace std;
6
7 // function main begins program execution
8 int main()
9 {
10     // create two GradeBook objects;
11     // initial course name of gradeBook1 is too long
12     GradeBook gradeBook1( "CS101 Introduction to Programming in C++" );
13     GradeBook gradeBook2( "CS102 C++ Data Structures" );
14
15     // display each GradeBook's courseName
16     cout << "gradeBook1's initial course name is: "
17          << gradeBook1.getCourseName()
18          << "\ngradeBook2's initial course name is: "
19          << gradeBook2.getCourseName() << endl;
```

□圖16.17 建立並操作**GradeBook**物件，其課程名稱不能超過25個字(2/2)



```
20
21 // modify myGradeBook's courseName (with a valid-length string)
22 gradeBook1.setCourseName( "CS101 C++ Programming" );
23
24 // display each GradeBook's courseName
25 cout << "\ngradeBook1's course name is: "
26     << gradeBook1.getCourseName()
27     << "\ngradeBook2's course name is: "
28     << gradeBook2.getCourseName() << endl;
29 } // end main
```

Name "CS101 Introduction to Programming in C++" exceeds maximum length (25).
Limiting courseName to first 25 characters.

gradeBook1's initial course name is: CS101 Introduction to Pro
gradeBook2's initial course name is: CS102 C++ Data Structures

gradeBook1's course name is: CS101 C++ Programming
gradeBook2's course name is: CS102 C++ Data Structures

□圖16.17 建立並操作**GradeBook**物件，其課程名稱不能超過25個字(1/2)

Programming codes 3-GradeBook.h

// 製作一標頭檔GradeBook.h置入於專案中的標頭檔資料夾中

```
#include <string>
```

```
using namespace std;
```

```
class GradeBook
```

```
{
```

```
public:
```

```
    GradeBook(string);
```

```
    void setCourseName(string);
```

```
    string getCourseName();
```

```
    void displayMessage();
```

```
private:
```

```
    string courseName;
```

```
};
```



Programming codes 3-GradeBook.cpp(1)

// 製作一C++(cpp)檔GradeBook.cpp，置入於專案中的來源檔案資料夾中

```
#include <iostream>
```

```
#include "GradeBook.h"
```

```
using namespace std;
```

```
GradeBook::GradeBook(string name)
```

```
{
```

```
    setCourseName(name);
```

```
}
```

```
void GradeBook::setCourseName(string name)
```

```
{
```

```
    if (name.length() <= 25)
```

```
        courseName = name;
```



Programming codes 3-GradeBook.cpp(2)

```
    if (name.length() > 25)
    {
        courseName = name.substr( 0, 25);

        cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
        << "Limiting courseName to first 25 characters.\n" << endl;
    }
}

string GradeBook::getCourseName()
{
    return courseName;
}

void GradeBook::displayMessage()
{
    cout << "Welcome to the grade book for\n" << getCourseName()
    << "!" << endl;
}
```



Programming codes 3(1)

```
#include <iostream>
#include "GradeBook.h"
using namespace std;

int main()
{
    GradeBook gradebook1("CSI01 Introduction to Programming in C++");
    GradeBook gradebook2("CSI02 C++ Data Structures");

    cout << "gradebook1's initial course name is: "
    << gradebook1.getCourseName()
    << "\ngradebook2's initial course name is: "
    << gradebook2.getCourseName() << endl;
    gradebook1.setCourseName("CSI01 C++ Programming");
```



Programming codes 3(2)

```
cout << "\ngradebook1's course name is: "  
<< gradebook1.getCourseName()  
<< "\ngradebook2's course name is: "  
<< gradebook2.getCourseName() << endl;  
}
```



▶ 測試GradeBook類別

- ▶ 圖16.17展現了**GradeBook**類別修改版(圖16.15-16.16)的驗證功能。第12行建立一個名為**gradeBook1**的**GradeBook**物件。第13行建立另一個名為**gradeBook2**的**GradeBook**物件，它傳給建構子的課程名稱剛好25個字。



-
- ▶ 圖16.17的第16-19行顯示**gradeBook1**切過的課程名稱以及**gradeBook2**的課程名稱。第22行直接呼叫**gradeBook1**的**setCourseName**成員函式，將**GradeBook**物件的課程名稱換成比較短的，就不用切了。接著，第25-28行再次輸出**GradeBook**物件的課程名稱。



▶ Set函式其它注意事項

- ▶ **public**的set函式 (如**setCourseName**) 應小心處理所有資料成員 (如**courseName**) 的數值修改，確保新數值對該資料項目是合理的。例如，將月份設成37就該拒絕，把人的體重設成0或負值也該拒絕，將考試成績設成185 (範圍應在0到100) 也要拒絕。



-
- ▶ 類別的set函式可傳回適當數值給用戶端，讓它知道設定的資料是無效的。用戶端可測試set函式的傳回值，判斷此物件修改是否成功，並採取適當動作。



16.9 總結

- ▶ 在本章中，你已經建立了使用者定義的類別，並建立與使用這些類別的物件。我們宣告了類別的資料成員，用來維護每個類別物件的資料。我們也定義了成員函式，用來操作資料。你學到了如何呼叫物件的成員函式，向它請求所提供的服務；也學到了如何將資料傳遞給這些成員函式，做為理·C我們討論了成員函式的區域變數以及類別的資料成員之間的差異。



-
- ▶ 我們也展示了要如何使用建構子。你學會了如何將類別介面和實作分開。我們以一張示意圖，說明類別實作者和用戶端程式設計者需要用來編譯程式碼的檔案。我們示範了如何使用set函式來驗證物件的資料。我們使用UML類別示意圖來模塑類別、其建構子、成員函式及資料成員。

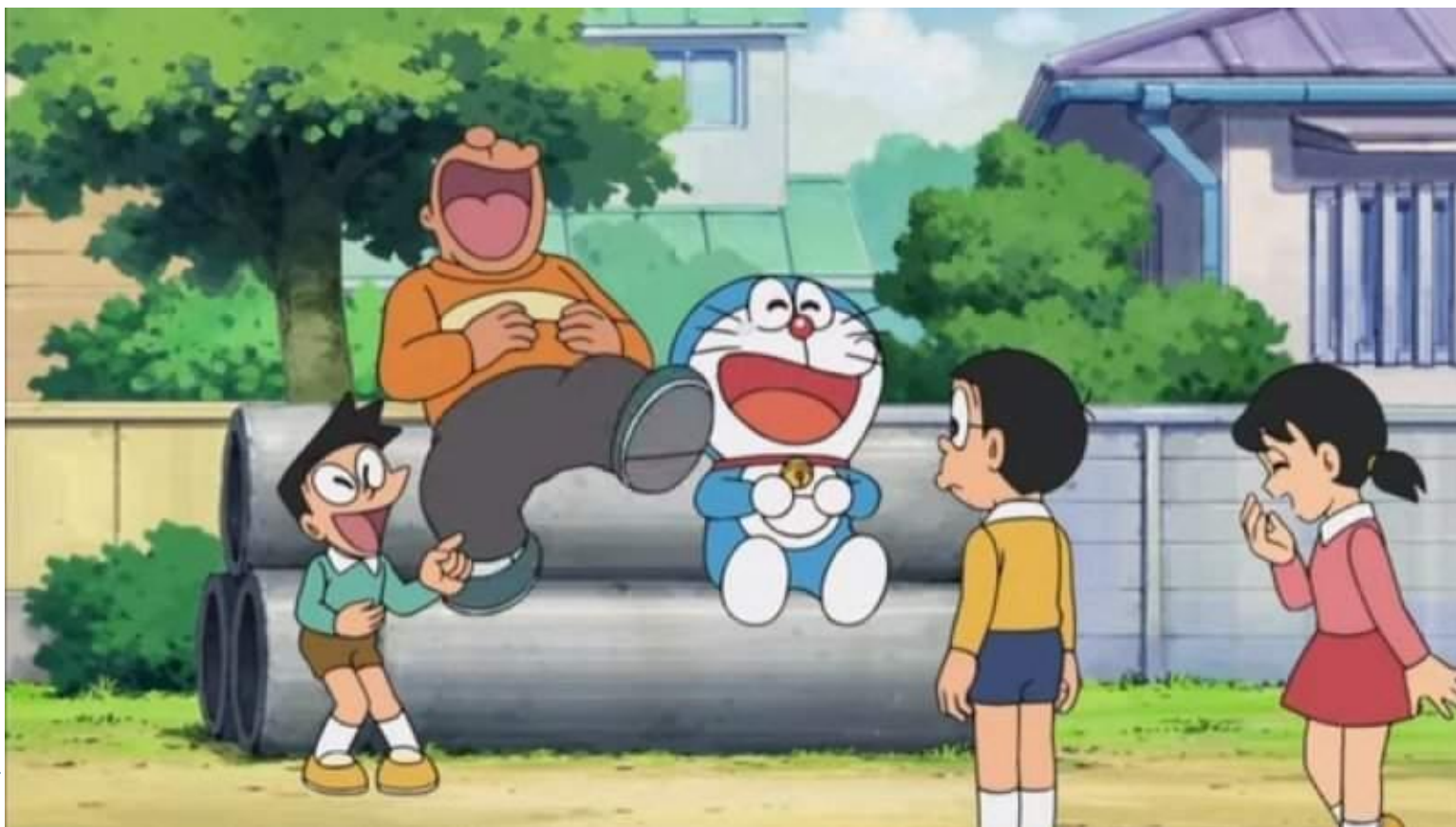


► Exercise



Exercise 1 程式功能

- ▶ 哆啦 A 夢場景中的空地因胖虎限制人數，不得超過 5 人，請使用 class 設計「人數管理系統」



Exercise 1 程式功能

- ▶ 當使用者按下**方向鍵↑**時，**總人數增加一人**。
若總人數超過 5 人，系統顯示「空地人數已滿！」。
- ▶ 當使用者按下**方向鍵↓**時，**總人數減少一人**。
若總人數低於 0，系統顯示「空地沒有人！」。
- ▶ 若更改後的總人數在範圍內，系統顯示目前人數。



Exercise 1 程式功能

- ▶ 使用者持續按下方方向鍵↑之執行成果

```
C:\Users\  
請使用方向鍵 ↑、↓ 增加、減少  
人數，按下 ESC 離開程式  
  
目前空地人數：0  
目前空地人數：1  
目前空地人數：2  
目前空地人數：3  
目前空地人數：4  
目前空地人數：5  
空地人數已滿！
```

- ▶ 使用者持續按下方方向鍵↓之執行成果

```
C:\Users\  
空地人數已滿！  
目前空地人數：4  
目前空地人數：3  
目前空地人數：2  
目前空地人數：1  
目前空地人數：0  
空地沒有人！
```

Exercise 1 程式功能規劃

1. **PlayGround.h & PlayGround.cpp** （功能介面）

提供功能：

(1) 增加人數 (2) 減少人數 (3) 取得目前人數

2. **main.cpp** （功能測試）

(1) 建立 PlayGround 物件 (2) 偵測使用者輸入

(3) 顯示目前空地人數或錯誤訊息

- ▶ PlayGround 類別僅提供程式開發人員的「介面」，因此所有輸出訊息（cout）僅能放置於 main.cpp 中
-

Exercise 1 程式碼：PlayGround.h

```
#pragma once

class Playground
{
public:
    Playground(int max_count = 5);
    // 若 current_count 小於 max_count，current_count 加 1 並回傳 true
    // 否則表示人數達上限，回傳 false
    bool AddCount();
    // 若 current_count 大於 0，current_count 減 1 並回傳 true
    // 否則表示已經沒有人，回傳 false
    bool DecreaseCount();
    // class 中的 get 方法，直接回傳 current_count
    int GetCurrentCount();

private:
    int max_count;
    int current_count;
};
```



Exercise 1 程式碼：PlayGround.cpp

```
#include "PlayGround.h"

PlayGround::PlayGround(int max_count) : current_count(0)
{
    PlayGround::max_count = (max_count >= 1) ? max_count : 5;
}

bool PlayGround::AddCount()
{
    // TO DO
}

bool PlayGround::DecreaseCount()
{
    // TO DO
}

int PlayGround::GetCurrentCount()
{
    // TO DO
}
```



Exercise 1 程式碼：[main.cpp](#)

```
#include <iostream>
#include "PlayGround.h"
#ifdef _WIN32
#include <conio.h>
#endif // _WIN32
using namespace std;

int main()
{
    PlayGround playGround(5);

    cout << "請使用方向鍵 ↑、↓ 增加、減少人數，按下  
ESC 離開程式" << endl << endl;
    cout << "目前空地人數：" <<
    playGround.GetCurrentCount() << endl;

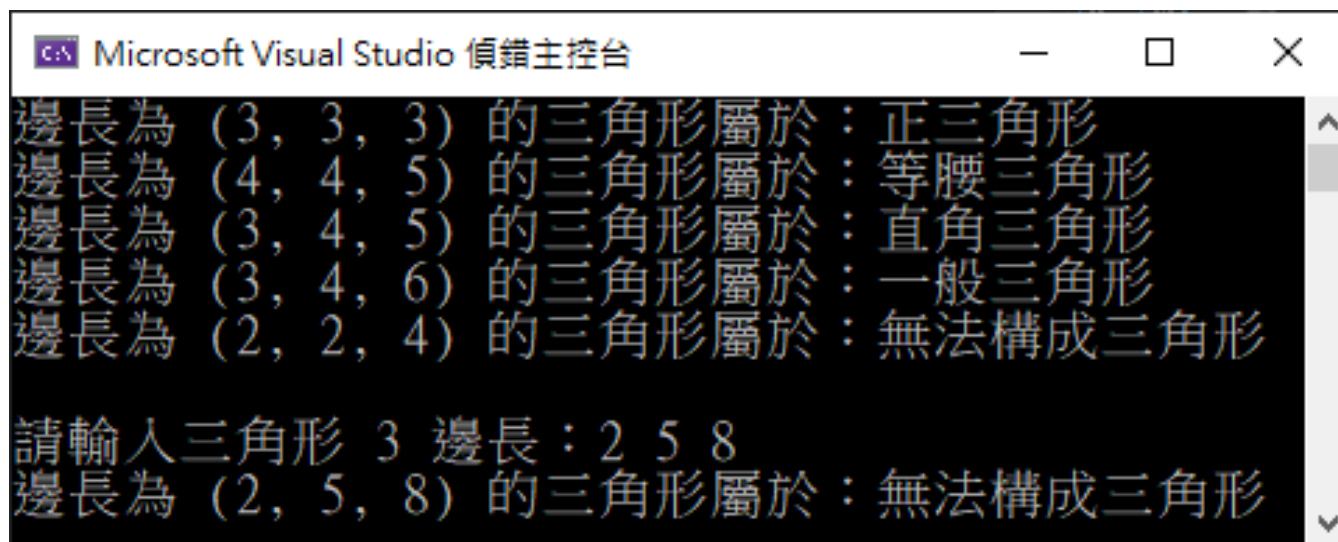
#ifdef _WIN32
    system("stty -icanon");
#endif // !_WIN32

    char key;
    do
    {
#ifdef _WIN32
        key = _getch();
#else
        key = getchar();
#endif // _WIN32
        if (key == (char)224) // 偵測是否按下方向鍵
        {
```

```
        #ifdef _WIN32
            key = _getch();
        #else
            key = getchar();
        #endif // _WIN32
        if (key == (char)72) // Arrow UP
        {
            if (playGround.AddCount() == false)
            {
                cout << "空地人數已滿！" << endl;
                continue;
            }
        }
        else if (key == (char)80) // Arrow DOWN
        {
            if (playGround.DecreaseCount() == false)
            {
                cout << "空地沒有人！" << endl;
                continue;
            }
        }
    }
    cout << "目前空地人數：" <<
    playGround.GetCurrentCount() << endl;
    } while (key != (char)27); // 按下 ESC 跳出迴圈
    return 0;
}
```

Exercise 2 程式功能

- ▶ 請設計三角形 class，由程式介面輸入的三邊長判斷三角形的類型。



The screenshot shows a Visual Studio console window titled "Microsoft Visual Studio 偵錯主控台". The console output displays the classification of five different triangles based on their side lengths. The results are as follows:

邊長 (Sides)	分類 (Classification)
(3, 3, 3)	正三角形 (Equilateral Triangle)
(4, 4, 5)	等腰三角形 (Isosceles Triangle)
(3, 4, 5)	直角三角形 (Right Triangle)
(3, 4, 6)	一般三角形 (General Triangle)
(2, 2, 4)	無法構成三角形 (Cannot form a triangle)

Below these results, the program prompts the user to input three sides: "請輸入三角形 3 邊長: 2 5 8". The user's input is shown, and the resulting classification is displayed: "邊長為 (2, 5, 8) 的三角形屬於: 無法構成三角形".

Exercise 2 程式功能規劃

1. **Triangle.h** & **Triangle.cpp** (功能介面)

- (1) 設定三角形 3 邊長
- (2) 判斷並取得三角形類型
- (3) 取得三角形 3 邊長

2. **main.cpp** (功能測試)

- (1) 建立 Triangle 物件並設定 3 邊長
 - (2) 顯示各種 Triangle 的三角形類型
 - (3) 提供使用者自訂三角形 3 邊長並顯示類型
-



Exercise 2 三角形類型判別備註

1. 正三角形 (Equilateral) : 三邊相等
2. 等腰三角形 (Isosceles) : 任意兩邊相等
3. 直角三角形 (Right) :
任意兩邊的平方和等於第三邊的平方
4. 一般三角形 (Normal) : 不是上述的三種情況
5. 無法構成三角形 (Invalid) :
任一邊的邊長 \geq 另外兩邊的邊長和



Exercise 2 程式碼：Triangle.h

```
#pragma once

class Triangle
{
public:

    enum Type {
        Equilateral, // 正三角形
        Isosceles,    // 等腰三角形
        Right,        // 直角三角形
        Normal,       // 一般三角形
        Invalid       // 無法構成三角形
    };

    // 由建構式輸入三角形的 3 邊長並儲存到
    // side_length 中
    Triangle(float length1, float length2, float
length3);

    // 判斷三角形的 3 邊長並回傳 enum 中的對應類
    // 型
    Type GetType();

    // 類別 get 方法，回傳紀錄於 side_length 的
    // 3 邊長
    float* GetSideLength();

private:
    float side_length[3];
};
```



Exercise 2 程式碼：Triangle.cpp

```
#include "Triangle.h"

Triangle::Triangle(float length1, float length2,
float length3)
{
    // TO DO
}

Triangle::GetType()
{
    // TO DO: 依據三角形的邊長回傳對應的類型
    // if(...)
    return Invalid;
    // if(...)
    return Equilateral;
    // if(...)
    return Isosceles;
    // if(...)
    return Right;
    // if(...)
    return Normal;
}

float* Triangle::GetSideLength()
{
    // TO DO
}
```



Exercise 2 程式碼：[main.cpp](#)

```
#include <iostream>
#include "Triangle.h"
using namespace std;

void DisplayTriangleType(Triangle &tri)
{
    float* side_length = tri.GetSideLength();
    cout << "邊長為 (" << side_length[0] << ", "
         << side_length[1] << ", "
         << side_length[2] << ") 的三角形屬於：";
    // TO DO: 根據 tri 物件使用 switch...case
    // 判斷並顯示三角形的類型
    // hint: switch(tri.GetType())
    // {
    //     case Triangle::Equilateral:
    //         ...
    //     break;
    //     ...
    // }
    cout << endl;
}
```

```
int main()
{
    // samples of various kinds of triangles
    Triangle triangle1(3, 3, 3), triangle2(4, 4, 5),
        triangle3(3, 4, 5), triangle4(3, 4, 6),
        triangle5(2, 2, 4);

    DisplayTriangleType(triangle1); // 正三角形
    DisplayTriangleType(triangle2); // 等腰三角形
    DisplayTriangleType(triangle3); // 直角三角形
    DisplayTriangleType(triangle4); // 一般三角形
    DisplayTriangleType(triangle5); // 無法構成三角形

    // triangle lengths from user
    float side1, side2, side3;
    cout << endl << "請輸入三角形 3 邊長：";
    cin >> side1 >> side2 >> side3;
    // display the input result
    Triangle user_tri(side1, side2, side3);
    DisplayTriangleType(user_tri);

    return 0;
}
```



wk5_ex1.txt



wk5_ex2.txt



To be continued.....

Instructor: Cheng-Chun Chang (張正春)
Department of Electrical Engineering