

下載範例檔案

- https://github.com/ck1001099-Teaching/cpp_course_2021summer
- lesson6 -> Lecture -> Rectangle_template.cpp

C++程式設計基礎

lesson 6

陳毅

學習目標

- 能夠獨自撰寫簡易的C++程式
- 了解基礎的物件導向程式設計方法
- 閱讀程式碼的能力

時間表

- | | | |
|--------|---|----------------------------|
| • 8/4 |] | • 資料型態、變數、基本輸入輸出 |
| • 8/6 | | • 流程控制（條件控制、迴圈控制） |
| • 8/9 |] | • 函式與陣列 |
| • 8/11 |] | • 指標 |
| • 8/13 | | |
| • 8/16 |] | • 資料結構
• 物件導向程式設計基礎 (2) |
| • 8/18 | | |
| • 8/20 | | |

上節回顧

- 指標
 - 動態記憶體
 - 配置與釋放
 - 動態陣列

動態記憶體

- 程式不會自動回收不再使用的變數或陣列記憶體。
- 若程式需要使用很多變數或陣列，佔據的記憶體就會越來越多。
- 記憶體使用過量所產生的問題
 - 程式可用空間不足（現今一台電腦最多也差不多就**128GB**而已）
 - 程式執行的速度（影響存取變數的速度）
- 若有些變數或陣列不再使用，想要釋放佔用的記憶體空間，則可以使用**配置動態記憶體**的方式。

動態記憶體－變數

- new 運算符

- 用來配置動態記憶體，並傳回一個起始指標。
- 配置失敗時，回傳NULL值。

```
變數指標 = new 資料型態(起始資料);
```

- delete 運算符

- 用來釋放動態記憶體指標。
- 只能用來釋放已配置的動態記憶體指標。

```
delete 變數指標;
```

動態記憶體－陣列

- new 運算符
 - 與配置動態變數指標類似。
 - 「new 資料型態」後面，要加上「[長度]」，用來配置陣列長度。

```
陣列指標 = new 資料型態[長度]
```

- delete 運算符
 - 與釋放動態變數指標類似。
 - delete 運算符之後，必須加上中括號，表示被釋放的指標是陣列指標。
 - 只能用來釋放已配置的動態記憶體指標。

```
delete[] 陣列指標;
```


Memory leak (記憶體流失)

- 若在用**delete**釋放記憶體前，就將指標指向其它地方，會發生什麼問題？
- 記憶體流失
 - 可以使用的記憶體越來越少。
 - 若記憶體用光，會導致程式無法運作。
- 參考說明：
<https://zh.wikipedia.org/wiki/%E5%86%85%E5%AD%98%E6%B3%84%E6%BC%8F>

本週概要

- 物件導向程式設計基本概念 – Part 1
- C++字串類別：string
- 資料結構（STL container）
 - vector
 - map

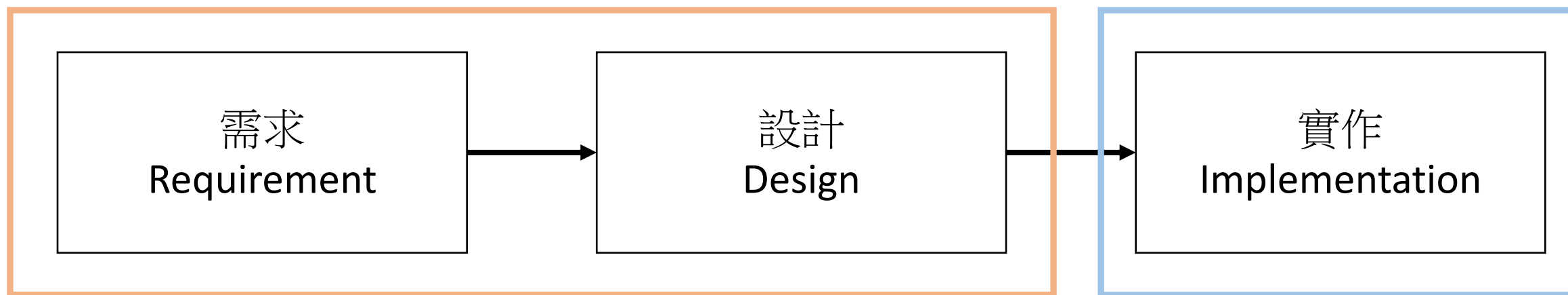
開始之前...

需求

- 請為銀行設計一個管理程式，用以管理客戶的帳戶及資料，並處理相關業務。
 - 一間銀行擁有許多客戶和許多帳戶。客戶名稱不會重複。帳戶的帳號也不會重複。
 - 客戶有名稱、出生年月日。
 - 每個客戶可以向銀行開設一個以上的帳戶。
 - 每個帳戶記載著帳號、客戶、歷史交易紀錄等資料。
 - 交易紀錄有2種類型：存入、提出。當客戶向銀行提出交易時，若交易成功，會將交易紀錄記載在帳戶中。
 - 客戶可以向銀行查詢自己帳戶的歷史交易紀錄。

實務經驗上...

- 我們要如何根據**需求**（**requirement**），去**設計**（**design**）我們的程式呢？



物件導向程式設計基本概念 – Part 1

物件導向程式設計

- 到目前為止，我們所學的C++設計方式還只是結構化程式設計。
- 結構化程式設計特點：
 - 程式由許多函數構成。
 - 不容易看出函數與函數間的關係。
 - 每個函數都可以存取程式中的任何資料，開發大程式時，不僅造成開發困難，也造成維護程式變得困難。

物件導向程式設計

- 結構化程式設計

```
1  #include <iostream>
2  using namespace std;
3
4  bool UpdateBoard(int[][3], int, int, int);
5  int CheckWinner(int[][3]);
6  void OutputBoard(int[][3]);
7
8  int main(){ ...
75 }
76
77 bool UpdateBoard(int board[][3], int row, int col, int playerIndex){ ...
90 }
91
92 int CheckWinner(int board[][3]){ ...
124 }
125
126 void OutputBoard(int board[][3]){ ...
146 }
```


物件導向程式設計 (Object-oriented programming) OOP

- 為了解決大程式函數與函數間隱含的連結，以及防止函數不小心存取不相關的變數資料，因此，將函數與相關的資料結合在一起，形成獨立的模組，稱作類別 (class)。
- 物件導向程式設計，是以類別為主的程式設計。

物件導向程式設計

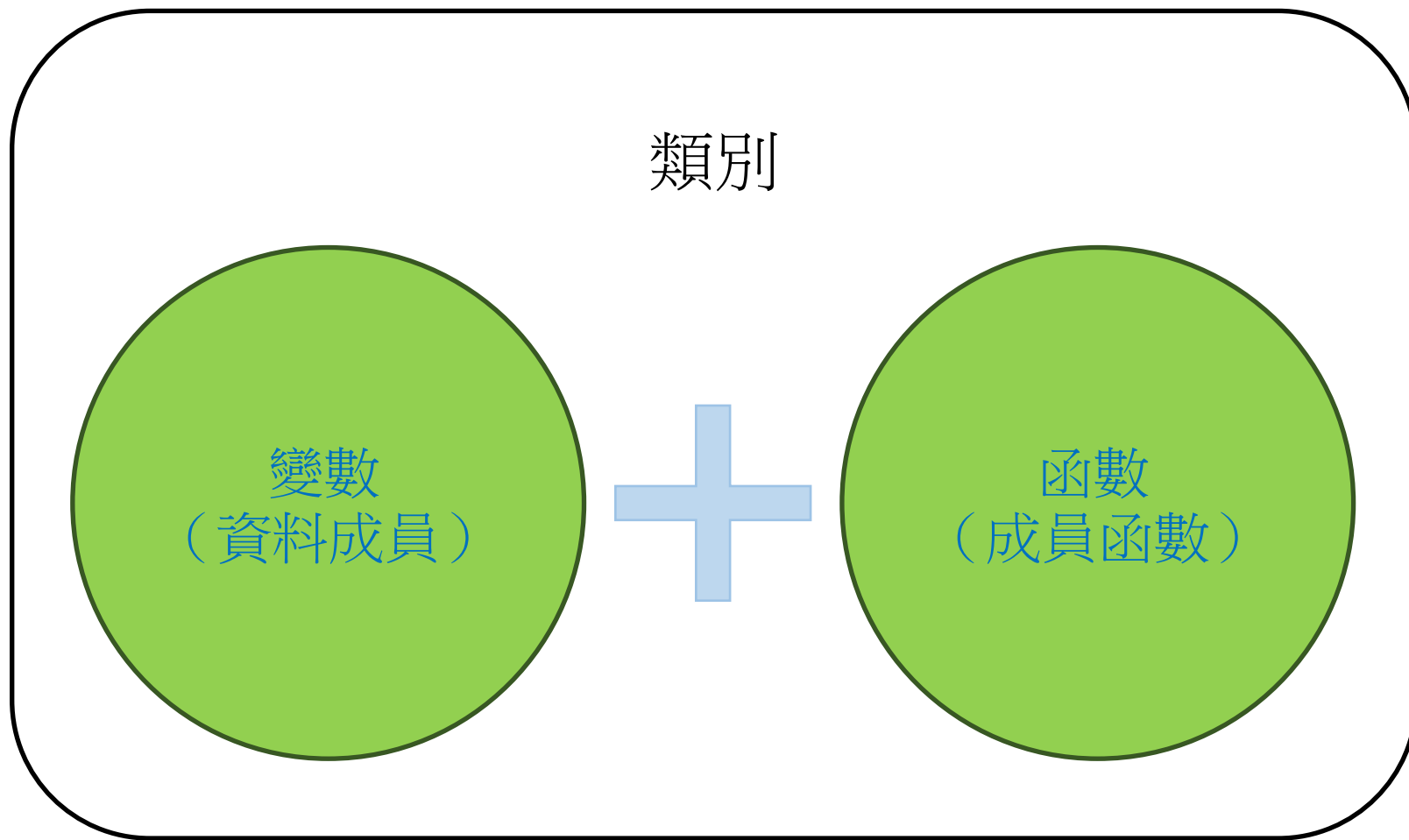
- 物件導向程式設計

```
13 class MyTouchTask{
14 public:
15     MyTouchTask();
16     double GetSuccessRate(vector<string> *tasks, map<string, json*> *data);
17     double TapTask(json *taskData);
18     //double TapTaskScrollView(json *taskData);
19     double LongPressTask(json *taskData);
20     double HorizontalScrollTask(json *taskData);
21     double VerticalScrollTask(json *taskData);
22     double SwipeTask(json *taskData);
23     double PinchTask(json *taskData);
24     double RotationTask(json *taskData);
25
26     void SetScreenSize(double x, double y);
27     void SetOutputStream(ofstream* _ofs);
28 private:
29     double successRate;
30     double TapMaximumDuration = 1.5;
31     double TapAllowableMovement = 10;
32     double LongPressMinimumPressDuration = 0.5;
33     double LongPressAllowableMovement = 10;
34     double PanHysteresis = 10;
35     double PanMagicNumberX = 0.35;
36     double PanMagicNumberY = 0.35;
37     double SwipeMaximumDuration = 0.5;
38     double SwipeMinimumMovement = 30;
39     double SwipeMinimumVelocity = 200;
40     double PinchHysteresis = 8;
41     double RotationHysteresis = 10.0 / 180 * M_PI;
42     double RotationMaximumAngleEachTimestamp = 0.785;
43
44     double screenSizeX;
45     double screenSizeY;
46
47     ofstream* ofs;
48 };
```

什麼是類別？

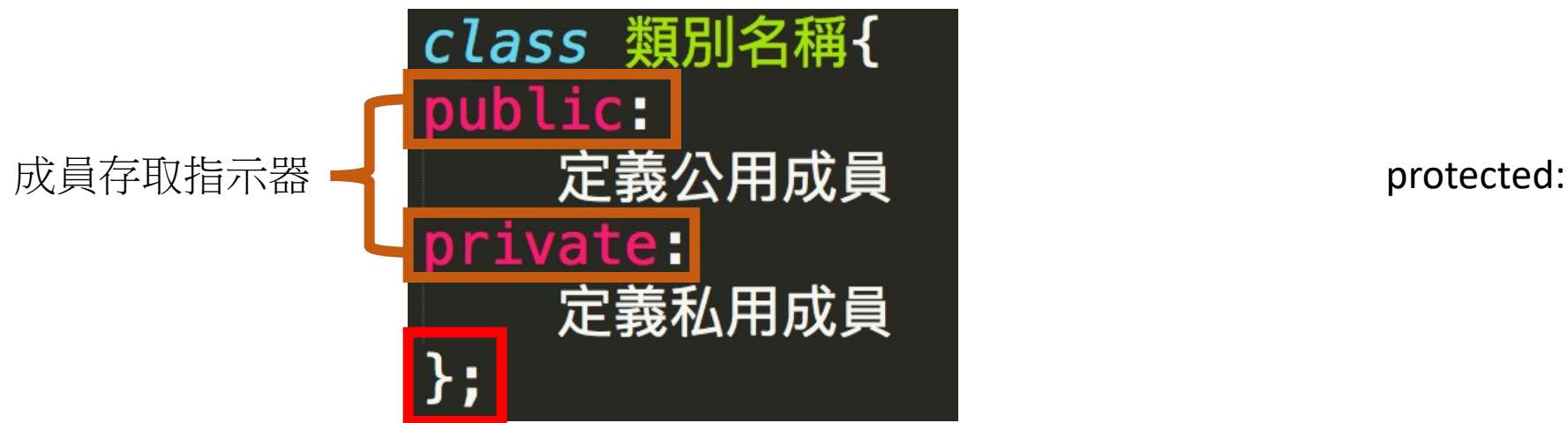
- 一種使用者自定的資料型態，由許多資料型態及函數集合而成。
- 資料變數稱作「資料成員(data member)」。
- 處理資料的函數稱作「成員函數 (member function)」。

什麼是類別？



類別

- 宣告類別名稱
 - `class`：宣告類別名稱的關鍵字
 - `public`標籤：公用成員，可任意存取及呼叫。
 - `private`標籤：私用成員，只供本類別或`friend`函數存取及呼叫。



The diagram illustrates the syntax of a C++ class declaration. It shows the following code snippet on a dark background:

```
class 類別名稱{  
    public:  
        定義公用成員  
    private:  
        定義私用成員  
};
```

Annotations and labels include:

- A bracket on the left labeled "成員存取指示器" (Member Access Indicator) points to the `public:` and `private:` access specifiers.
- The word "protected:" is written to the right of the `private:` section.
- The closing brace and semicolon `};` are enclosed in a red box.

分號很重要！

類別－資料成員 (data member)

- 定義類別資料成員與定義一般變數、陣列是一樣的。

資料型態 變數名稱;

- 由於「資料成員」是定義在類別內，因此該變數是屬於該類別的。
 - 不同類別的資料成員是互相獨立的。

```
class A{  
public:  
    double a;  
private:  
    int b;  
    float c;  
};
```

≠

```
class B{  
public:  
    string a;  
    bool b;  
private:  
    char c;  
};
```

類別－成員函數(member function)

- 定義成員函數的語法與定義一般函數的語法是一樣的。

```
傳回型態 函數名稱(參數){  
    敘述...  
}
```

- 在進行類別的設計時，一般會使用成員函數去存取私用的資料成員。

```
class A{  
public:  
    int GetNumber(){  
        return Number;  
    }  
private:  
    int Number;  
};
```

類別－成員函數(member function)

- 可是，如果將所有函數的定義都寫在class裡面，class就會變得異常肥大。

```
class A{
public:
    int GetNumber(){
        return Number;
    }
    void SetNumber(int number){
        Number = number;
    }
    void Plus(int value){
        Number = Number + value;
    }
    void Minus(int value){
        Number = Number - value;
    }
    int GetAbsNumber(){
        return abs(Number);
    }
private:
    int Number;
};
```


類別－成員函數(member function)

- 一個好的解決方法，是在類別中，只定義**函數雛形**，在其他地方進行**函數宣告**。

函數雛型(Declaration)

用來告訴編譯器，這個程式會有哪些函數。

型態 **函數名稱**(引數1型態, 引數2型態, ...);

函數宣告(Definition)

用來定義一個函數實際的執行內容。

```
型態 函數名稱(引數1, 引數2, ...){  
    程式敘述;  
    ...  
}
```

```
#include <iostream>  
using namespace std;  
  
void I_AM_A_FUNCTION(int, int);  
  
int main(){  
  
    I_AM_A_FUNCTION(1, 2);  
  
    return 0;  
}  
  
void I_AM_A_FUNCTION(int a, int b){  
    cout << a << " " << b << endl;  
    return;  
}
```

類別－成員函數(member function)

- 怎麼在類別外進行函數宣告？
- 使用範圍運算符號(::)（雙冒號）
 - 將函數指定給宣告此成員函數雛形的類別。
- 例如
 - A::GetNumber()是A類別下的GetNumber()函數。
 - A::SetNumber()是A類別下的SetNumber()函數。

類別－成員函數(member function)

```
class A{  
public:  
    int GetNumber(){  
        return Number;  
    }  
    void SetNumber(int number){  
        Number = number;  
    }  
    void Plus(int value){  
        Number = Number + value;  
    }  
    void Minus(int value){  
        Number = Number - value;  
    }  
    int GetAbsNumber(){  
        return abs(Number);  
    }  
private:  
    int Number;  
};
```

```
class A{  
public:  
    int GetNumber();  
    void SetNumber(int);  
    void Plus(int);  
    void Minus(int);  
    int GetAbsNumber();  
private:  
    int Number;  
};
```

```
int A::GetNumber(){  
    return Number;  
}  
void A::SetNumber(int number){  
    Number = number;  
}  
void A::Plus(int value){  
    Number = Number + value;  
}  
void A::Minus(int value){  
    Number = Number - value;  
}  
int A::GetAbsNumber(){  
    return abs(Number);  
}
```

類別

- 建立類別物件

```
類別名稱 物件名稱;
```

- 注意：物件與類別是不同的東西！物件是變數，類別是資料型態。

- 存取類別成員

```
物件名稱.類別成員();
```

```
A testClass;  
  
testClass.SetNumber(4);  
testClass.GetNumber();
```

練習 (Rectangle_template.cpp)

- 建立一個長方形 (Rectangle)類別，其資料成員與成員函數如下：
 - 定義private資料成員length與width，分別存放長方形的長和寬。
 - 定義public成員函數setLength與getLength，用來設定與取得length。
 - 定義public成員函數setWidth與getWidth，用來設定與取得width。
 - 定義public成員函數perimeter與area，分別用來計算長方形的周長與面積。
- 撰寫main()函數，由鍵盤輸入資料，分別存入length與width中，然後計算並顯示長方形周長與面積。

```
int main(){
    Rectangle rect;
    int l, w;
    cin >> l >> w;
    rect.setLength(l);
    rect.setWidth(w);
    cout << "Perimeter: " << rect.perimeter() << endl;
    cout << "Area: " << rect.area() << endl;
    return 0;
}
```

Class diagram

Rectangle

- length: int
- width: int

屬性（資料成員）
Attributes, properties

+ setLength(int): void
+ getLegnth(): int
+ setWidth(int): void
+ getWidth(): int
+ perimeter(): int
+ area(): int

行為（成員函數）
behaviors

```
class Rectangle{  
public:  
    void setLength(int l){ length = l; }  
    int getLength(){ return length; }  
    void setWidth(int w){ width = w; }  
    int getWidth(){ return width; }  
    int perimeter(){ return 2 * (width + length); }  
    int area(){ return width * length; }  
private:  
    int length;  
    int width;  
};
```

```
int perimeter(){  
    計算周長並回傳  
}  
  
int area (){  
    return 長x寬  
}
```

Pseudo code
（偽代碼）

軟體工程
UML (class diagram)

C++字串類別：string

簡介

- C++字串類別是一個抽象的資料型態，它並不是C++原本內建的資料型態。
- C++字串類別及其相關函數式定義於C++的新型標題檔(header)中，因此使用這些函數前，必須將其引入。
- 要引入標頭檔：`#include <string>`
- 被定義在std下，因此要輸入`using namespace std;`

C型態字串 v.s. C++字串類別

- **C型態字串**：使用字元陣列或指標來定義字串。

```
char *name = "JOHN";  
char name[20] = "JOHN";
```

- **C++字串類別**：宣告**string**類別的字串物件來處理字串。

```
string s1;  
string s2("JOHN ARCHER");  
string s3 = "MARY ARCHER";  
string s4("A", 4);  
string s5(s2);  
string s6(s2, 0, 4);
```

C++字串類別

- 宣告及初始化方式

- C++ reference: <http://www.cplusplus.com/reference/string/string/string/>

- 輸入C++字串

- cin >> 字串物件;
 - getline(cin, 字串物件);

```
string s1, s2;

cout << "請輸入 s1 字串: ";
getline(cin, s1);
cout << "請輸入 s2 字串: ";
cin >> s2;
cout << "s1 = " << s1 << endl;
cout << "s2 = " << s2 << endl;
```

C++字串類別

- 常用運算符號

運算符號	功能說明
=	指定資料
+	串接字串
+=	連接並指定字串
==	相等
!=	不相等
<, >, <=, >=	逐一比較字元大小
[]	存取字元
<<	輸出
>>	輸入

C++字串類別

- 字串物件陣列：與宣告一般陣列無異。

```
string array[10];
```

```
string s1[] {"Java", "Assembly", "Delphi", "Basic", "Fortran", "Cobol"};
```

C++字串類別

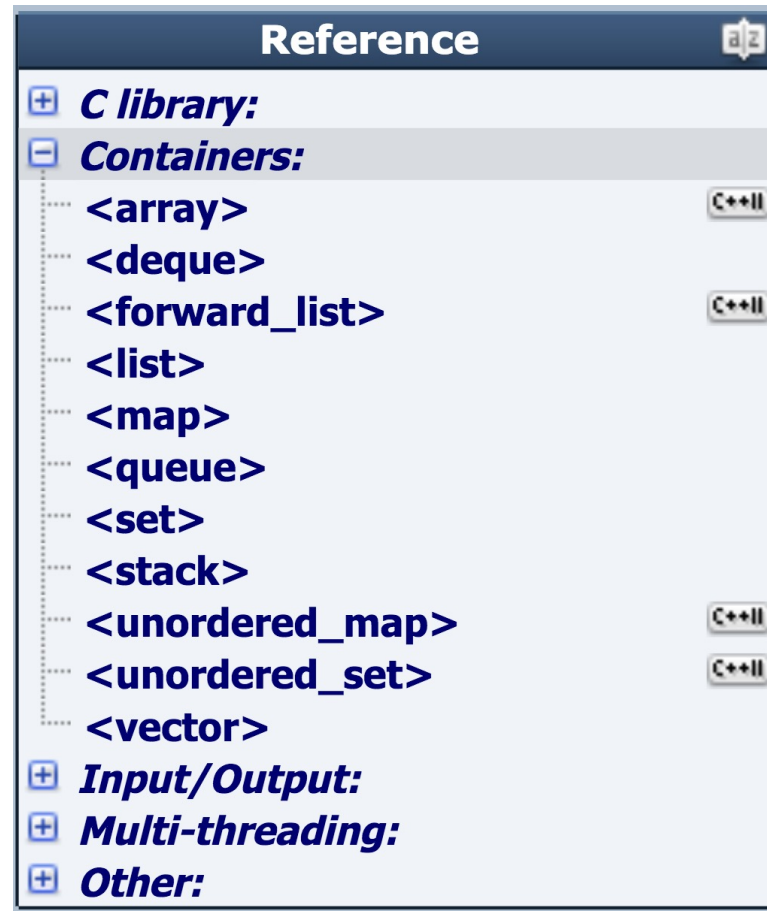
- C++字串類別成員函數（參見課本表**9.2**）
 - C++ reference: <http://www.cplusplus.com/reference/string/string/>

成員函數	功能
s1.append(s2)	連接字串
s1.at(位置)	存取指定位置
s1.clear()	清除字串全部內容
s1.length()	取得字串長度
s1.swap(s2)	對調字串
s1.replace(起始位置, 字串長度, s2)	取代部分字串
s1.insert(起始位置, s2)	插入字串
s1.find(s2) s1.find(s2, 起始位置)	找尋字串
s1.copy(s2, 起始位置, 字串長度)	複製字串

STL container

什麼是STL container？

- STL container是被設計用來儲存資料的容器，正常會使用到的資料結構，都已經包含在裡面了。
- 舉例
 - 陣列：vector
 - 字典：map
 - 佇列：queue（First in, First out）
 - 堆疊：stack（First in Last out）



STL container - vector

簡介

- **vector**可視為會自動擴展容量的陣列。
 - 支援隨機存取
 - 在集合尾端增刪元素很快，但是在集合中間增刪元素比較費時
 - 使用動態陣列方式實作
- 容器類型：陣列

C 語言

```
int* array = new int[10];  
delete[] array;  
array = new int[20];
```

使用方法

- 要引入標頭檔：`#include <vector>`
- 被定義在std下，因此要輸入`using namespace std;`
- C++ referenece: <http://www.cplusplus.com/reference/vector/vector/>
- Wikipedia: [https://zh.wikipedia.org/wiki/Vector_\(STL\)](https://zh.wikipedia.org/wiki/Vector_(STL))
- 宣告vector物件
 - `vector<資料型態> 物件名稱;`
 - 在<>中間的資料型態，為vector容器所能存放的資料的型態。
 - 資料型態可以是任何型態
 - 變數型態：int, float, double, bool, char等。
 - 指標型態：int*, double*, char**等。
 - 類別(class)：自定義類別, STL container等。

成員函數

- 存取元素
 1. `vec[i]`：存取第*i*個元素
 2. `vec.at(i)`：存取索引值為*i*的元素的參照
- 新增或移除元素
 1. `vec.push_back()`：在最末端加入元素
 2. `vec.pop_back()`：刪除最末端的元素
 3. `vec.insert()`：插入元素
 4. `vec.erase()`：刪除元素
 5. `vec.clear()`：清空元素

成員函數

- 取得長度/容量
 1. `vec.size()`：目前vector持有的元素個數
 2. `vec.empty()`：若vector為空，則會回傳true，反之回傳false
- iterator (疊代器)
 1. `vec.begin()`：回傳一個iterator，它指向vector第一個元素
 2. `vec.end()`：回傳一個iterator，它指向vector最後一個元素的下一個位置

練習

- 寫一個C++程式，輸入一些數字，並算出總和、平均、標準差。
 - 輸入數字皆大於0，若輸入0，則輸出計算結果，並停止程式。
- 寫一個C++程式，練習移除vector內的元素。
 - 第一行：n, m。
 - 第二行：有n個數字，請加入至vector中。
 - 第三行：有m個數字，代表要移除的元素的編號。

STL container - map

簡介

- map具有一對一mapping 功能。
 - 第一個稱為關鍵字 (key)，每個關鍵字只能在map中出現一次。
 - 第二個稱為該關鍵字的值 (value)。
- 容器類型：字典
 - 在一本字典裡，一個單字 (key)只會出現一次。
 - 根據單字 (key)，我們可以查到該單字所對應的解釋 (value)。

使用方法

- 要引入標頭檔：`#include <map>`
- 被定義在std下，因此要輸入`using namespace std;`
- C++ reference: <http://www.cplusplus.com/reference/map/map/>
- 宣告map物件
 - `map<資料型態, 資料型態> 物件名稱;`
 - map中有兩個資料型態，前者為key的資料型態，後者為value的資料型態。
 - 資料型態可以是任何型態

map成員函數

- 插入元素

```
// 用 insert 函數插入 pair  
mapStudent.insert(pair<string, string>("r000", "student_zero"));
```

```
//用 "array" 方式插入  
mapStudent["r123"] = "student_first";  
mapStudent["r456"] = "student_second";
```

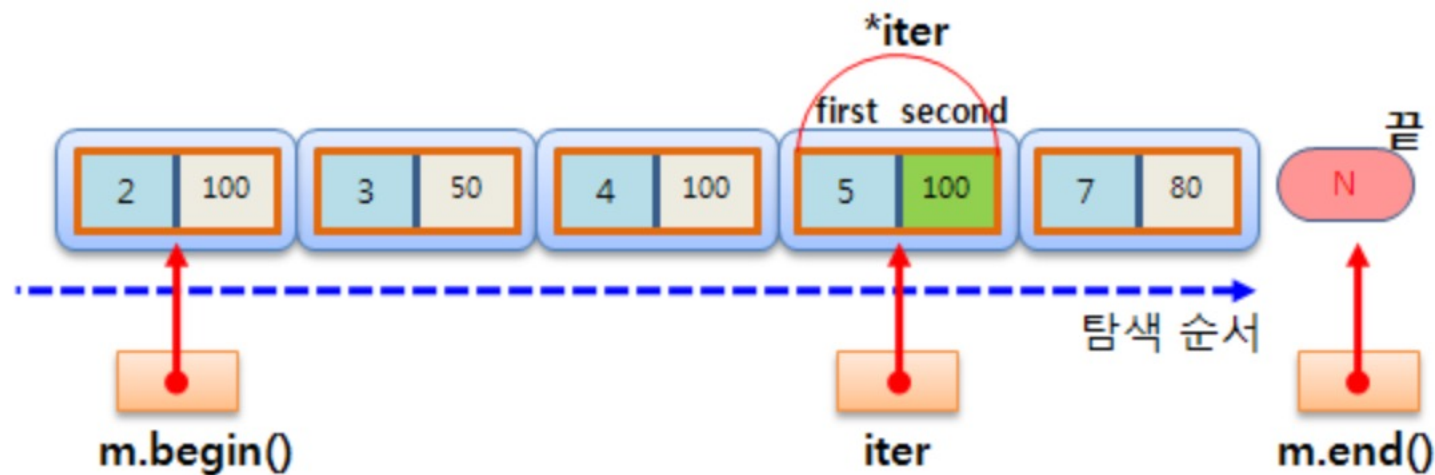
map成員函數

- 尋找元素

```
iter = mapStudent.find("r123");
```

```
if(iter != mapStudent.end())  
    cout<<"Find, the value is"<<iter->second<<endl;
```

```
else  
    cout<<"Do not Find"<<endl;
```



map成員函數

- 移除元素

//迭代器刪除

```
iter = mapStudent.find("r123");  
mapStudent.erase(iter);
```

//用關鍵字刪除

```
int n = mapStudent.erase("r123");//如果刪除了會返回1，否則返回0
```

//用迭代器範圍刪除：把整個map清空

```
mapStudent.erase(mapStudent.begin(), mapStudent.end());  
//等同於mapStudent.clear()
```

練習

- (一維對照表) 學號及姓名的對照表。
 - 請使用`map`建立右方的對照表。
 - 請問學號為**b001**的人是誰？
 - 請問學號為**r003**的人是誰？

學號	姓名
r001	Amy
r002	Bob
b001	Cindy
d001	Dancy
r003	Elsa
b002	Franck

練習

- (二維對照表) 班級學生各科成績表。
 - 請使用map建立右方的成績表。
 - 請問Amy的English成績是多少？
 - 請問Elsa的Science成績是多少？
 - 請問Franck的Chinese成績是多少？
 - 請問Bob的數學成績有比Dancy的國文成績高嗎？

姓名	Chinese	English	Math	Science
Amy	14	55	76	100
Bob	84	83	27	37
Cindy	23	34	46	27
Dancy	34	53	83	75
Elsa	74	76	24	44
Franck	58	39	66	37

下週概要

- 物件導向程式設計基本概念 – Part 2
- 「需求（Requirement）」與「設計（Design）」

練習

- Zerojudge

- d586 : 哈密瓜美語
- d827 : 買鉛筆
- d460 : 山六九之旅
- a291 : nAnB problem
- d209 : 古代神祕文字
- c807 : 一維凸包問題
- c188 : 拉瑪努金的逆襲

- Reading: *C++ reference -> Reference -> Containers -> <vector> & <map>*