

# 下載範例檔案

- [https://github.com/ck1001099/cpp\\_course\\_2020fall](https://github.com/ck1001099/cpp_course_2020fall)
- lesson8 -> Lecture -> Calculate\_template.cpp

# C++程式設計基礎

## lesson 8

陳毅

# 上週回顧

- 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

# 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)」。

# 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。

成員存取指示器

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

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

- 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();
```



# 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

# 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)。

# 本週概要

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

# 物件導向程式設計基本概念 – Part 2

# 建立者(constructor)與破壞者(destructor)

- 你可能曾經在哪裡看過它們

<i>fx</i> Member functions	
<b>(constructor)</b>	Construct vector (public member function )
<b>(destructor)</b>	Vector destructor (public member function )
<b>operator=</b>	Assign content (public member function )

- 你會發現，大多數的class都會定義constructor與destructor。
- 那什麼是constructor與destructor？

# 建立者(creator)與破壞者(destructor)

- ~~某二次元手遊（少○前X）中的破壞者~~



# 建立者(constructor)與破壞者(destructor)

- constructor
  - 在建立物件時，會自動被呼叫，一般用於初始化物件內容。
- destructor
  - 在釋放物件時，會自動被呼叫，一般用於釋放物件內動態指派的記憶體。
- constructor與destructor都是**成員函數(member function)**。
- constructor與destructor在概念上是互補的。

# 建立者(constructor)

- constructor是一個成員函數 (member function) 。
- 但是，其函數宣告方式與一般函數有差異！！

```
類別名稱(參數列){  
    敘述...  
}
```

- 宣告限制
  1. constructor不能有「傳回型態」。
  2. 「函數名稱」固定為「類別名稱」。
  3. 必須為public成員。



# 建立者(constructor)

- 若為宣告constructor時，物件被建立時，會呼叫預設的constructor。
- 預設的constructor是一個無參數、無敘述的空函數。

```
類別名稱() {}
```

# 建立者(constructor)

- 宣告constructor範例

```
class A{  
public:  
    A(double);  
    A(int, int);  
private:  
    int _a, _b;  
    double _c;  
}  
  
A::A(double c){  
    _c = c;  
}  
  
A::A(int a, int b){  
    _a = a;  
    _b = b;  
}
```

- 使用constructor初始化物件

```
A obj1(12.3);  
A obj2(3, 4);
```

# 破壞者(destructor)

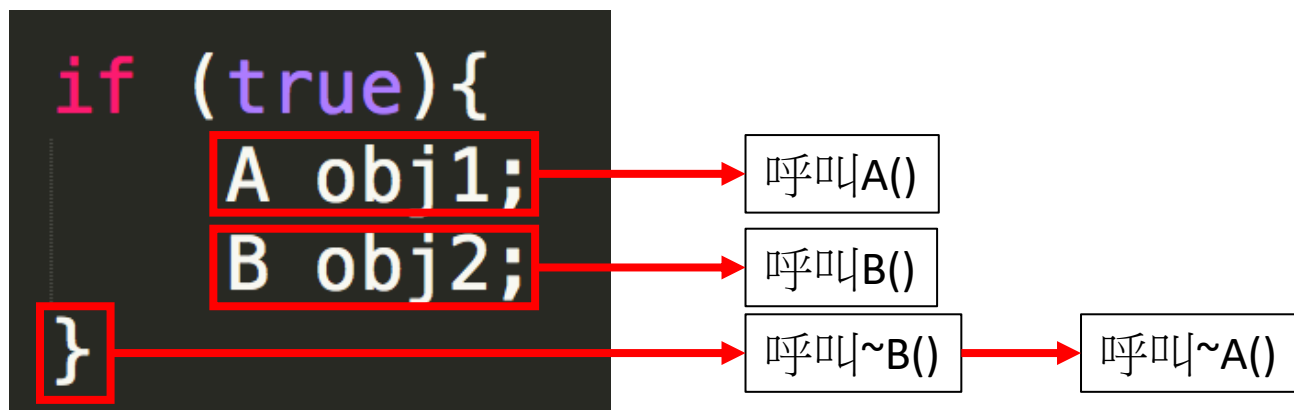
- destructor是一個成員函數 (member function) 。
- 但是，其函數宣告方式與一般函數有差異！！

~類別名稱();

- 宣告限制
  1. destructor不能有「傳回型態」，也不能有「任何參數」。
  2. 「函數名稱」是「~(否定符號)」加上「類別名稱」。
  3. 必須為public成員。

# 破壞者(destructor)

- 在建立物件的函數結束時，將自動呼叫破壞者函數。
- 若函數中有多個建立者函數，破壞時將以反序方式破壞，也就是先建立者後破壞（**first-construct-last-destruct**）。



# 練習 (Calculate\_template.cpp)

- 定義一個運算(Calculate)類別，其資料成員與成員函數如下：
  - 定義private資料成員x，用來存放運算值。
  - 定義建立者(constructor)函數，並設定x的初值為0。
  - 定義public成員函數add、sub、mul與div，處理加、減、乘、除四則計算。
  - 定義public成員函數result，輸出x。
- 撰寫main()函數，輸入算式，並輸出運算結果。

# 類別與指標

- this指標
  - 建立物件時，指向被建立的物件本身。
  - 會被自動傳遞給所有非靜態(non-static)的成員函數(member function)。
  - 可用「this->資料成員」或「(\*this).資料成員」來存取資料成員。
- 優點
  - 不管成員函數的參數名稱如何宣告，都可以存許到自身的資料成員。

```
A::A(double c){  
    this->c = c;  
}
```

# 類別與陣列

- 宣告類別形態的陣列與宣告一般資料型態的陣列一樣，只是**資料型態改為使用者自訂的類別型態**。

```
類別名稱 陣列名稱[陣列長度];
```

「需求（Requirement）」與「設計（Design）」



# 練習 (圖書資料管理) (程式11-8)

- 請為圖書館設計一個程式，用以幫助管理員管理館內書籍的資料。
  - 一本書有書名、作者、書號、價格。
  - 當有新的書籍入館時，管理員需要輸入該書籍的資料，將它新增至資料庫中。
  - 管理員可以透過此程式，看到現有的館內書籍清單。

# 圖書資料管理－延伸

- 請為圖書館設計一個程式，用以幫助管理員管理館內書籍的資料。
  - 一本書有書名、作者、書號、價格。
  - 當有新的書籍入館時，管理員需要輸入該書籍的資料，將它新增至資料庫中，系統會自動給這本書籍一個唯一的編號。
  - 管理員可以透過此程式，看到現有的館內書籍清單。
  - 當管理員關閉程式時，程式會將書籍資料儲存到硬碟中，下次開啟程式時，若已有書籍資料，則會將這些資料載入到系統中。
  - 當有書籍報廢時，管理員需要輸入該書籍的編號，將該書籍移出系統。

# 練習

- [HackerRank](#) (Practice -> C++ -> Classes)
- Class
- Classes and Objects
- Box It!

The screenshot shows the HackerRank interface for C++ practice. The top navigation bar includes links for PRACTICE, COMPETE, JOBS, and LEADERBOARD. The user's profile 'shps\_40717' is visible in the top right. The main content area displays the C++ practice path: Practice > C++ > C++. The user's current rank is 124655 and points are 50/70. A progress bar indicates 20 more points to the next star. The main list of topics includes Structs, Class, Classes and Objects, and Box It!, each with a 'Solve Challenge' button. On the right, a sidebar filters by STATUS (Solved, Unsolved), DIFFICULTY (Easy, Medium, Hard), and SUBDOMAINS (Introduction, Strings, Classes, STL). The 'Classes' subdomain is currently selected.

**Structs**  
Easy, Max Score: 10, Success Rate: 96.52%,  
Learn how to create and use structures.

**Class**  
Easy, Max Score: 10, Success Rate: 95.95%,

**Classes and Objects**  
Easy, Max Score: 20, Success Rate: 98.49%,

**Box It!**

**STATUS**  
☐ Solved  
☐ Unsolved

**DIFFICULTY**  
☐ Easy  
☐ Medium  
☐ Hard

**SUBDOMAINS**  
☐ Introduction  
☐ Strings  
☒ Classes  
☐ STL

# 作業

- 從以下題目任選兩題完成，下次上課時檢查。
  - a059：完全平方和
  - b680：百米賽道編排
  - d550：物件排序
  - a225：明明愛排列
  - d555：平面上的極大點
- Reading: 課本 Ch11.1 ~ 11.4
- 若遇到作業問題，歡迎隨時寄信至：[ck1001099@gmail.com](mailto:ck1001099@gmail.com)