

C++程式設計基礎

week 9

陳毅

上週回顧

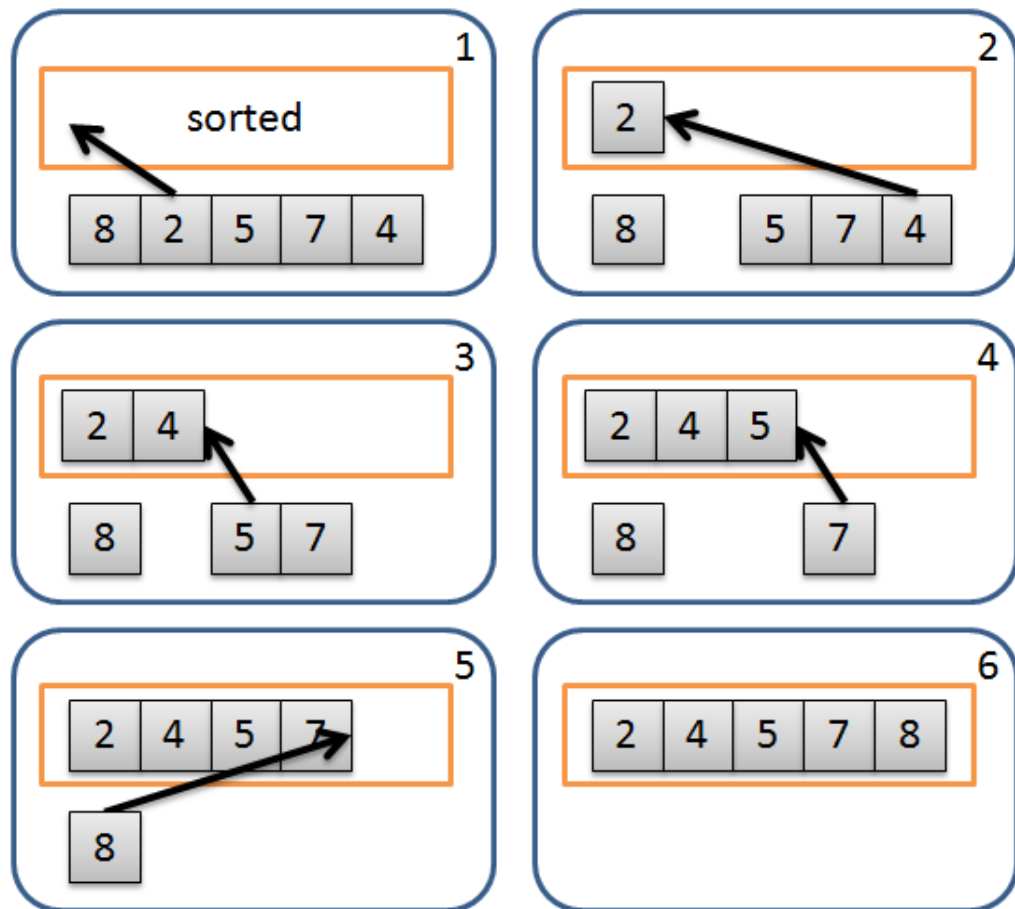
- 基礎演算法
 - 排序
 - 選擇排序法 (selection sort)
 - 泡泡排序法 (bubble sort)
 - 搜尋
 - 循序搜尋法 (sequential search)
 - 二元搜尋法 (binary search)

選擇排序法 (Selection sort)

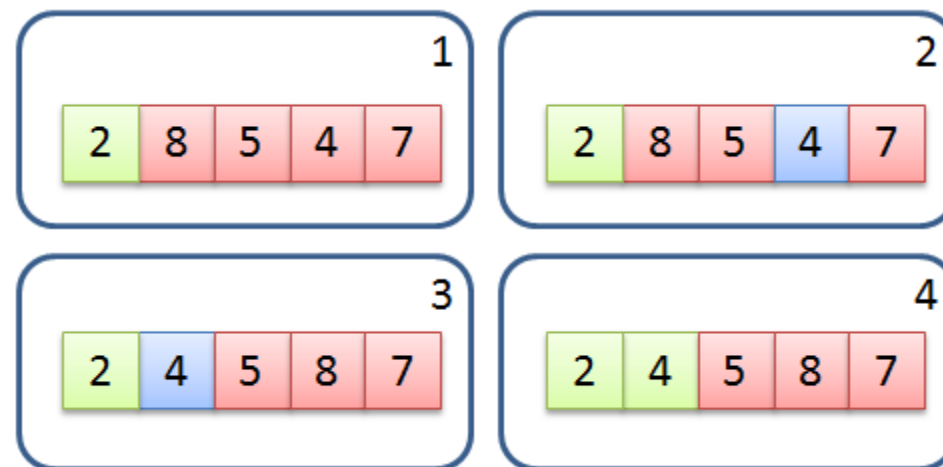
- **選擇排序法(Selection Sort)**是排序演算法的一種，也是一種簡單容易理解的演算法。
- 其概念是反覆從未排序的數列中取出最小的元素，加入到另一個的數列，結果即為已排序的數列。

流程圖

額外空間



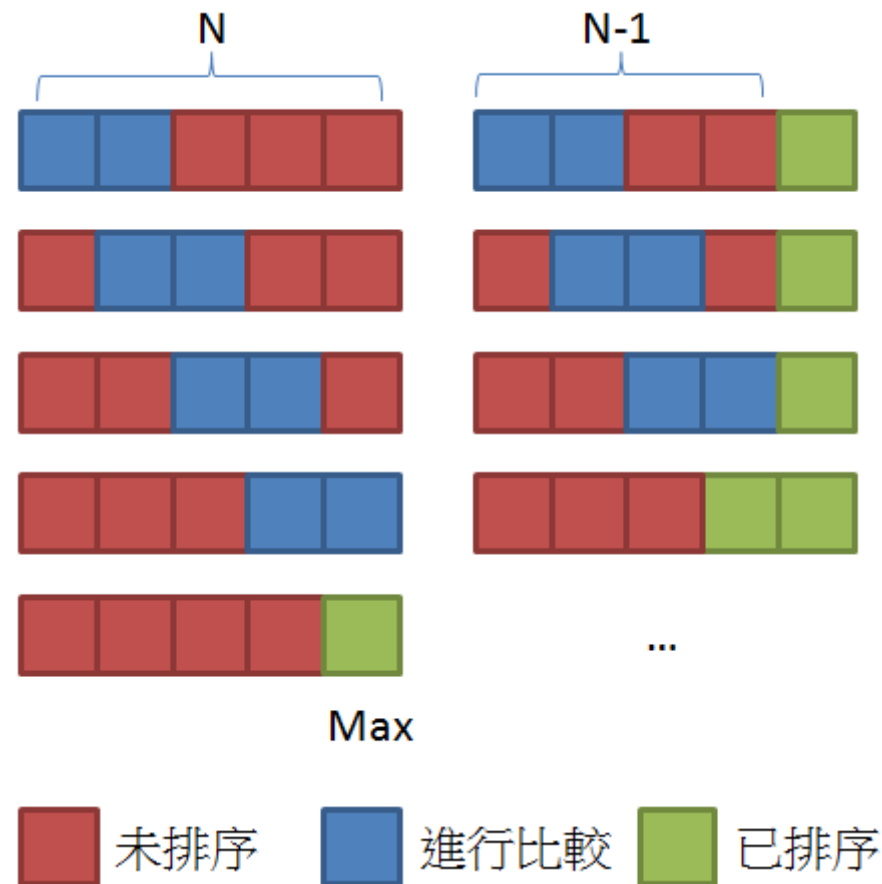
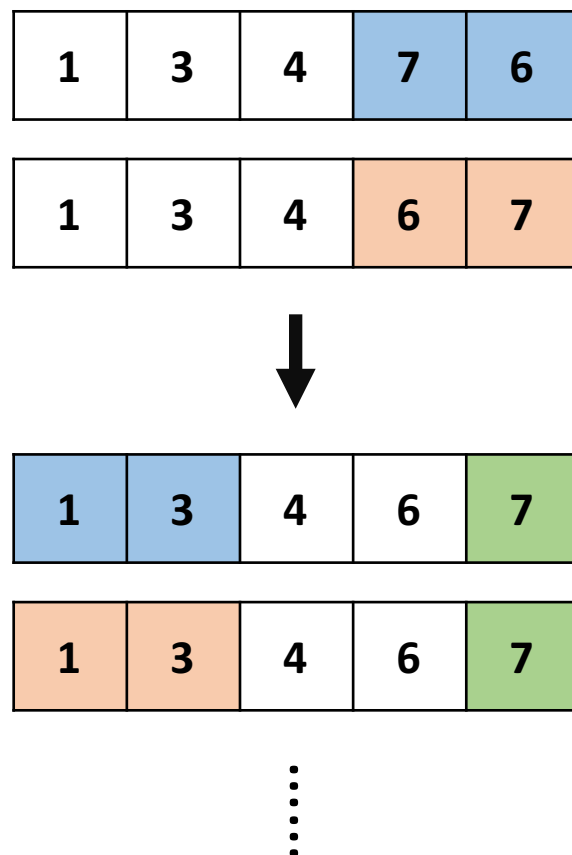
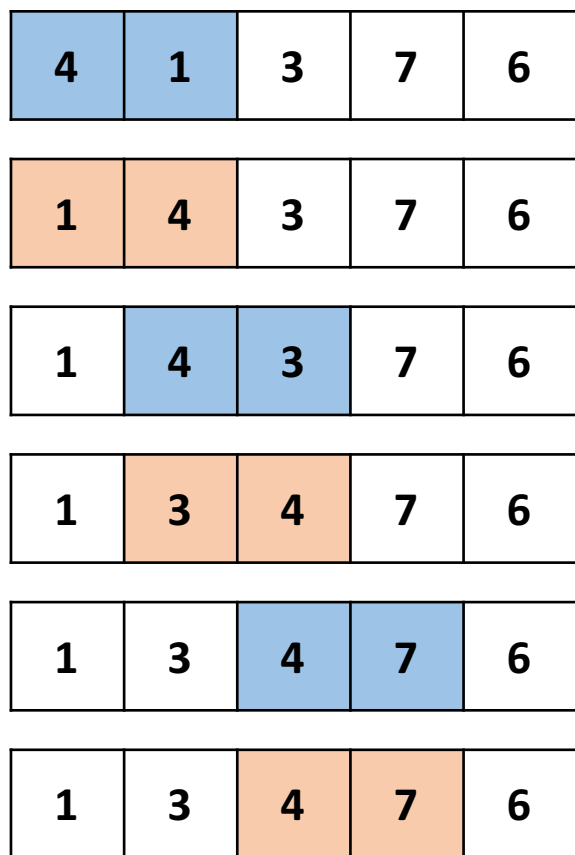
In-place



氣泡排序法 (Bubble sort)

- **氣泡排序法(Bubble Sort)**是最容易理解和實作的一種**排序演算法**，也翻譯作冒泡排序法。
- 由於它很容易學習，所以也是許多演算法課程中第一個學習的排序演算法。
- 演算法過程會將**最大的數值移動到陣列最後面**，而較小的數值則**逐漸的往陣列前端移動**，就像有許多氣泡慢慢從底部浮出，因此成為氣泡排序法。

流程圖



循序排序法 (Sequential search)

- **循序搜尋法**是相當直觀的**搜尋演算法**，又稱作**線性搜尋法**。
- 利用迴圈逐一比對要搜尋的資料，若相等則表示找到，並中斷迴圈，若全部都不等，則表示找不到。

二元排序法 (Binary search)

- **二元搜尋法**是常用的**搜尋演算法**，又稱作二分搜尋法。二元搜尋法的原理與「終極密碼」的流程十分類似。
- 使用二元搜尋法前，**必須將資料排序**。
- 假設資料已從小到大排序。
- 演算法過程是**尋找編號上限與編號下限的中間項**，然後**比較中間項與搜索資料**。
 - 若中間項小於搜尋資料，則「 $\text{編號下限} = \text{中間項編號} + 1$ 」
 - 若中間項大於搜尋資料，則「 $\text{編號上限} = \text{中間項編號} - 1$ 」
 - 若相等，則尋找到資料。

本週概要

- 類別化物件
 - 什麼是類別 (class) ?
 - 建立者(constructor)與破壞者(destructor)
 - 類別與指標

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

- 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)」。

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－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;
}
```

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	
(constructor)	Construct vector (public member function)
(destructor)	Vector destructor (public member function)
operator=	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)

- 在某手遊中的破壞者(誤)



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)

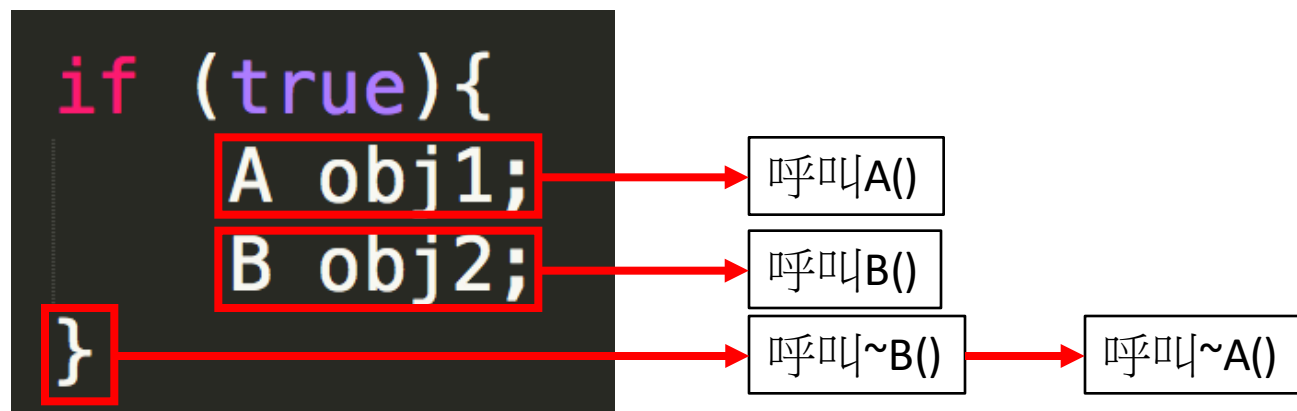
- destructor是一個成員函數 (member function)。
- 但是，其函數宣告方式與一般函數有差異！！

```
~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];
```

練習 (圖書資料管理) (程式11-8)

- 定義一個書籍資料(**book**)類別，其資料成員與成員函數如下：
 - 定義**private**資料成員**title**，用來儲存「書名」。
 - 定義**private**資料成員**author**，用來儲存「作者」。
 - 定義**private**資料成員**number**，用來儲存「書號」。
 - 定義**private**資料成員**price**，用來儲存「價格」。
 - 定義**public**成員函數**setBook**，以鍵盤輸入書籍資料。
 - 定義**public**成員函數**showBook**，顯示書籍資料。
- 在**main**函數中，宣告一個**book**陣列，用以儲存許多本書的書籍資料。

練習

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

The screenshot displays the HackerRank interface for C++ practice. The top navigation bar includes links for PRACTICE, COMPETE, JOBS, and LEADERBOARD, along with a search bar and user profile information (shps_40717). The main content area is titled 'Practice > C++' and shows a progress bar indicating '20 more points to get your next star!' with a current rank of 124655 and 50/70 points. Below this, a list of topics is shown, each with a 'Solve Challenge' button and a star icon:

- 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!**

On the right side, there are filters for STATUS (Solved, Unsolved), DIFFICULTY (Easy, Medium, Hard), and SUBDOMAINS (Introduction, Strings, **Classes**, STL). The 'Classes' subdomain is currently selected.

下週課程預告

- 期末專案demo（如果有人有做的話）
- 專案管理：Sourcetree + github
- 物件導向補充 – 靜態成員(static member)、friend函數
- 多載函數（Ch12）
- 進階演算法
 - 更多的排序演算法：Insertion sort、Quick sort
 - Dynamic programming
- 其他程式語言簡介
- 課程回顧

中英名詞對照

- 類別：Class
- 物件：Object
- 物件導向程式設計：Object-Oriented Programming
- 資料成員：Data member
- 成員函數：Member function
- 靜態成員：Static member
- 建立者：Constructor
- 破壞者：Destructor

作業

- 從以下題目任選兩題完成，下次上課時找助教檢查。
 - c665：進制轉換
 - d092：算式也可以比大小！？
 - d625：踩地雷真好玩
 - a218：連猴子都會的小case
 - c638：天干地支
 - d098：Stringstream運用練習(C++)
 - a982：迷宮問題#1
- Reading: 課本 Ch11.1 ~ 11.4
- 若遇到作業問題，歡迎隨時寄信至：r07922059@ntu.edu.tw