C++程式設計基礎
week 7

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的方法
- 閱讀程式碼的能力
- 獨立完成一個小專案

時間表

- 9/19
- 9/26
- 10/3
- 10/17
- 10/24
- 10/31
- 11/14
- 11/21
- 11/28
- 12/5

- 資料型態、變數、基本輸入輸出
- 流程控制與函式
- 陣列
- 字元與字串
- 複習與練習
- 指標
- C++字串類別 - <string>
- 資料結構（STL container） -<vector>, <map>
- 基礎演算法
- 物件導向程式設計基礎
- 專案實作（踩地雷）

上週回顧

- 指標
  - **動態記憶體**
    - **配置與釋放**
    - **動態陣列**
- C++字串類別：string

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;
```

6-3.cpp, 6-4.cpp

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

6-2.cpp

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;
```

6-5.cpp

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

6-6.cpp, 6-7.cpp

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"};
```

6-8.cpp

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

練習

- (9-13) 寫一C++英打練習程式。
  - 在main函數中，定義**一個字串陣列**，起始資料為英文單字（最少30個單字），再定義**一個字串變數**用來存放串接後的字串。
  - 利用亂數取得陣列中的5個單字，將5個單字串接在一起，單字間以空白隔開。再將此字串存入字串變數後，呼叫並傳遞字串參數給englishTyping函數。
  - 定義一個englishTyping函數，接收呼叫敘述傳遞的字串參數，然後接收鍵盤輸入，按Enter鍵後輸出正確字數、錯誤字數、與正確率。

本週概要

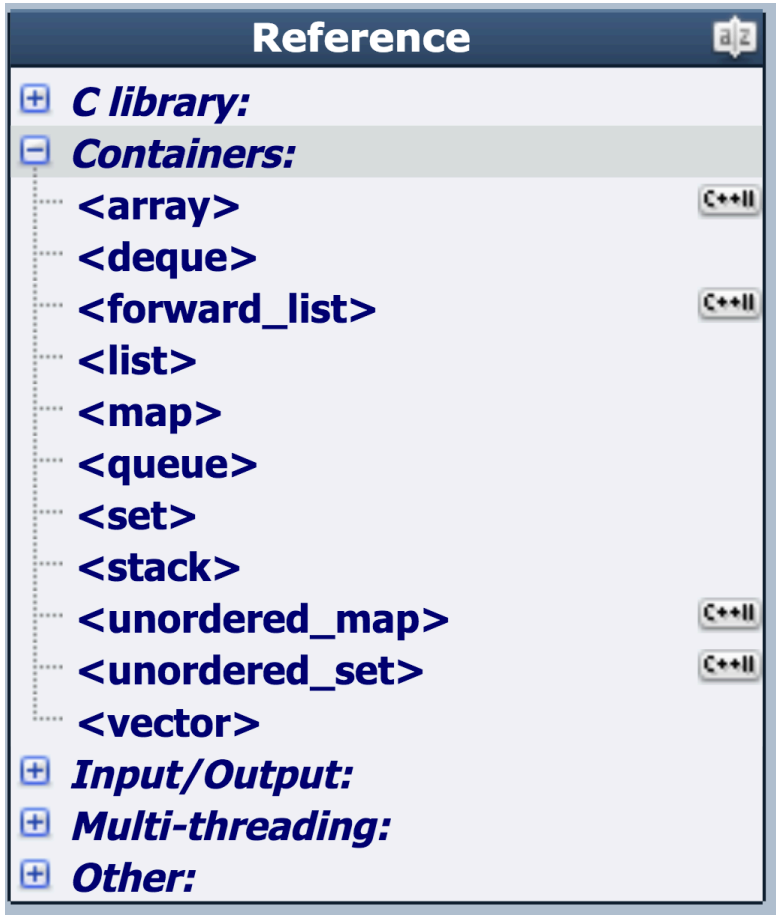
- 資料結構（STL container）
  - vector
  - map
- 「期末專案：踩地雷」實作及討論

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

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)
- 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。

參考資料：http://dangerlover9403.pixnet.net/blog/post/98733652-%5B%E6%95%99%E5%AD%B8%5Dc++-vector%E8%A9%B3%E7%B4%B0%E7%94%A8%E6%B3%95

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

參考資料：http://mropengate.blogspot.com/2015/12/cc-map-stl.html

map成員函數

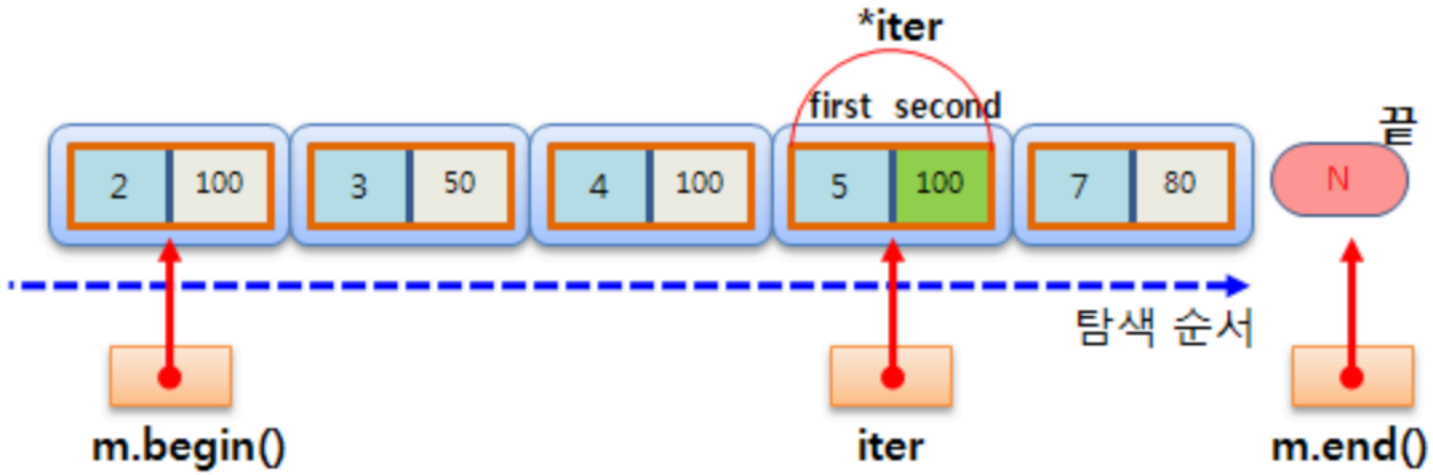
- 插入元素

```
// 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。

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

  - 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？

下週預計課程內容

- 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)
- 進階演算法
  - 進階排序
    - 插入排序法 (insertion sort)
    - 快速排序法 (quick sort)
  - Dynamic programming (DP)

期末專案：踩地雷

實作及討論

- 分組
  - 以小組為單位，一組1至4人，找好請找助教登記。

- 開發專案的流程
  1. 設定目標
  2. 系統規劃
  3. 系統實作
  4. 測試除錯
  5. 發佈

設定目標

- 互動
  - 玩家如何操作？
  - 是否需要圖像化介面？

- 設計
  - 遊戲的模式有幾種？
  - 規則是什麼？

- 系統
  - 是否支援儲存遊戲記錄？

系統規劃－以「遊戲流程」為例

- 遊戲流程
  - 遊戲開始
    - 根據參數初始化遊戲：生成盤面、建立資料結構
  - 遊戲進行中
    - 處理輸入：鍵盤輸入、滑鼠輸入
    - 處理輸出：螢幕輸出
    - 判斷遊戲是否結束
    - 遊戲邏輯
    - 儲存遊戲記錄
  - 遊戲結束
    - 處理輸出：贏家是誰
    - 再來一局

系統實作

- 根據「系統規劃」，逐步實作各項功能。

- 在多人團隊分工的情境中，要跟他人確認好函數的使用方法以及需要傳入的參數，並**盡量不要使用全域變數，以免造成程式碼間過度耦合**。

測試除錯

- 每實作一部分的功能，都需要測試是否與已完成的部分有不相容的狀況或錯誤。

- 在最終發佈前，可請他人協助測試，許多時候，程式設計師都不會發現自己哪裡有問題。

發佈

- 可自行發佈，或是透過各種管道發佈程式執行檔。

- 12/5：Demo

作業

- 從以下題目任選兩題完成，下次上課時找助教檢查。
  - 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>*
- 若遇到作業問題，歡迎隨時寄信至：r07922059@ntu.edu.tw