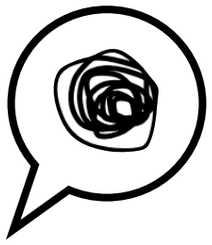
Advanced C++:
More C++ Tools and ideas

Yu-Hsuan Chen

2019 Spring, Introduction to Programming

所以C++到底是什麼？



助教，我到現在也只看過cin、cout啊

C++的真本事

- 強化的型別安全：禁止了void *、加入const的概念，從根本減少C語言中大量的強制轉型
- 集成的STL函式庫：可以快速的做出特定資料結構
- 核心價值：物件導向(Object-Oriented)的程式設計

物件導向？

- 什麼是物件導向？那可以吃嗎？
  →把每個小元件（物件）看作獨立的個體，可以接受、處理、傳達資料

- 物件導向的三大特色
  - 封裝(Encapsulation)
  - 繼承(Inheritance)
  - 多型(Polymorphism)

- 程式碼邁向10000+的關鍵！！

BUT，現在還不是時候

讓你們這麼早接觸OOP是一場巨大的災難
基本的程式還是先打穩再來吧。

期中考之後的實習目標

- 有條理的規劃你的程式——尤其是Function
- STL 函式庫初認識
- 遞迴與動態規劃

STL Introduction

- STL，全名Standard Template Library，中文叫做「標準模板函式庫」

- 透過STL，可以實現大部分的資料結構與型態。

- STL包含了三大要素：容器(container)、演算法(algorithm)、迭代器(iterator)

STL - vector

- 這個vector不是線性代數裡面的那個vector
- 可視作一個動態的陣列，隨時可以對vector增加/刪除內容
- 宣告時不用事先知道大小

```
#include<vector>
std::vector<int> V;   //V目前是空的
```

- 慢，刪除對vector來說效率低落

vector的常用操作

- v.front()　　//回傳v的第一個元素
- v.back()　　//回傳v的最後一個元素的「下一格」
- v.push_back(value)　　//將value加入v的尾巴
- v.pop_back()　　//將v的最後一個取出
- v.size()　　//回傳v裡面的元素數量
- v[i]　　//用陣列的方式存取v

STL - list

- #include<list>
  std::list<int> LL;

- **對的，Linked List是可以用list實作出來的**
  list允許任意位置的插入與刪除

- **但List不能像vector那樣，透過 [] 來存取其中的內容**

list的常用操作

- LL.front()　　//回傳list的第一個元素
- LL.back()　　//回傳list的最後一個元素
- LL.push_front(value)　　//從list的前端插入一個value
- LL.push_back(value)　　//在list的後端插入一個value
- LL.pop_front()　　//把list的第一個元素取出
- LL.pop_back()　　//把list的最後一個元素取出
- LL.insert(position, value)　　//從指定的位置插入一個value

這個position有點特別，他是iterator，細節後面會說

STL - stack

- **想起了什麼嗎？** p與q的邂逅？矩陣的乘法？

- ```
  #include <stack>
  std::stack<int> S;
  ```

- **三種操作**

  ```
  S.pop()          //將S最上面的資料取走(注意沒有回傳值)
  S.top()          //回傳S最上面的資料
  S.push(value)    //將value壓進S的最上面
  ```

- **只能操作最上面的元素（廢話！這就是Stack啊）**
  ```

STL - deque & queue

- 這兩個很類似。
  queue就是一般的佇列（從後面插入、前面離開）
  deque則是雙向的佇列（兩個方向都可以插入刪除）

- ```
  #include<deque>
  #include<queue>
  std::deque <int> DQ;
  std::queue <int> QUE;
  ```
  ```

queue/deque

- std::queue<int> QUE;
- QUE.front() //回傳第一個
- QUE.back()
  //回傳最後一個
- QUE.push(value)
  //從尾巴加入一個value
- QUE.pop()
  //把QUE的第一個取出

- std::deque<int> DQ;
- DQ.push_back(value);
- DQ.push_front(value);
- DQ.pop_front();
- DQ.pop_back();
- DQ.insert(pos,value);
  //在pos位置插入value
- DQ.erase(pos);
  //把pos位置的元素刪掉

STL - map

- map的構成：<Key, Value>，其中Key是唯一的
- map可以應用在Hash Table（雜湊表）

迭代器 iterator

- 也被稱作cursor（游標）。
- 可以直接在container上遍巡整個介面，而且不用關心底層的實作細節
- 迭代器的概念在很多語言都有實現：Java,C#,Python...
- C++是透過STL與template做出iterator。

迭代器 iterator (cont.)

- 幾乎所有的STL容器都可以用iterator
- iterator的宣告

```
vector<int>::iterator it;
```

- 使用iterator

```
std::vector<int> V = {1, 2, 3, 4, 5};
it = V.begin();        //it指向V的第一個元素
std::cout << *it;      //印出1
it = V.end() - 1;      //end指向的是V的最後一個的「下一個」空白
std::cout << *it;      //所以V.end()-1才會是V最後一個元素
```

STL - find

- find不是一個容器，而是一個找尋特定資料是否存在的方法
- 被定義在<algorithm>裡面
- 函式原型：

```
Iterator find(Iterator first, Iterator last, const T& val);
```

- 回傳值：如果有找到就回傳目標所在的位置，找不到就回傳last（指定找尋範圍的尾巴）

STL - find (cont.)

```
std::vector<int> V= {10, 20, 30, 40, 50};
std::vector<int>::iterator it;

it = std::find(V.begin(), V.end(), 30);
        //找尋整個V，確認30是否在V裡面
if(it != V.end())
    std::cout << *it;
else
    std::cout <<  “Not Found!” ;
```

new & delete

- 可以當成malloc跟free的加強版
- 使用方式

```
int *ptr = new int;          //取得一個int大小的空間
int *ptr = new int(100);     //取得空間，並初始化為100
int *ptr2 = new int[10];     //ptr2是一個有10格的陣列

delete ptr;                  //回收單一空間
delete [] ptr2;              //回收陣列
```

string

- **在你只會C的時候，你會用char陣列來當作字串...**

  char c[] = “NTOUCS”;

- **但是這個字串操作很麻煩...**

  需要strcpy、strcmp等等的函式輔助....

  char c2[7] = c; //Error

  strcpy(c2,c); //OK

- **有時候會人為操作失誤而不自覺**

  char a[7] = “1234567”; // ‘\0’因為空間不夠而不見了

- **現在你可以用C++的string類別解決這一切了!**

string (cont.)

- 這三個標頭檔的差異

  #include<string.h>

  #include<cstring>

  #include<string>

  string.h跟cstring是同樣的東西，處理的層級只有char *的內容，要用C++的string類別需要的是第三行的<string>

- #include<string>

  std::string SS;

string (cont.)

- **string可以是任意的大小!!**

```
string s1 = “” ;          //空字串
string s2 = “NTOUCSE” ;   //長度隨意
```

- **string可以拼接組合、直接比較是否相等**

```
string s3 = “Apple” ;
string s4 = “Pen” ;
string s5 = s3+s4;   //ApplePen，string裡面有定義+的運算
cout << s3 == s4;    //False
```

- **取得string的長度**

```
s3.length();   //5
```

注意這個東西是不被允許的

```
string s = “NTOU” ;
char a[] = s; //NO!!
```

字串處理：如何應對不一樣的Input Format

C/C++ 串流比較表

輸入種類\語言	C <stdio.h>	C++ <iostream>
數字 int, double, float	scanf %d %i, %lf, %f	std::cin
單一字元	scanf %c	std::cin cin.get()
字串 (沒有空白)	scanf %s	std::cin
讀取一整行	gets() fgets()	getline()

傳統的gets

- 大一上剛學C的時候，會用gets(char *str)

- 但你開始用Visual Studio 2015之後就不能用了
  實際上gets是不安全的（無法控制輸入Buffer的大小）

控制最大輸入長度
超過就忽略

- 替代品：fgets(char *str, int size, FILE *stream)

標準輸入是stdin

getline：一次讀入一行

- cin.getline(char* s, int size, char delim);
  接受一個char字串，必須傳入大小(避免overflow)
  會把delim從buffer取走，但不放入str

結束字元（讀到這個字元就截斷）
此參數可以省略，預設是'\0'

- getline(istream& is, string& str, char delim);
  接受string作為輸入
  會把delim從buffer取走，但一樣不放入str

Sample Code

```
#include<iostream>
#include<string>
using namespace std;

int main()
{
    string name;
    getline(cin, name);
    cout << name;
}
```

```
#include<iostream>
using namespace std;

int main()
{
    char name[256];
    cin.getline(name,256);
    cout << name;
}
```