【第九講】

排序

講師: 李根逸 (Ken-Yi Lee), E-mail: feis.tw@gmail.com

★ ★ ★ ★

★ ★ ★ ★

課程大綱

- 排序
  - 選擇排序法
  - 插入排序法
  - 對陣列和串列排序
  - 使用 STL 的 **`sort`**

排序

- **排序意指將容器內的元素依照某種規定的順序排列**
  - 在後面投影片我們指的大小表示的是在這個前後順序
- **常見的排序演算法為交換型演算法：**
  - 泡沫排序法 (bubble sort)
  - 選擇排序法 (selection sort)
  - 插入排序法 (insertion sort)
  - 希爾排序法 (Shell sort)
  - 快速排序法 (quicksort)
  - 合併排序法 (merge sort)
- **還有一些使用特殊資料結構的排序演算法在後面會提到**

選擇排序法

- 選擇排序法會每次將目前為止最小的放在最左邊

插入排序法

- 插入排序法會一個一個將元素插入目前已經排好的元素中

陣列和串列如何插入？

9	7	5	10	3	8	4
7	9	5	10	3	8	4
5	7	9	10	3	8	4
5	7	9	10	3	8	4
3	5	7	9	10	8	4
3	5	7	8	9	10	4
3	4	5	7	8	9	10

【範例】對陣列與串列排序

- 陣列（vector）具備快速隨機存取（operator[]）的特性但是雙向串列 （list） 不具備快速隨機存取（operator[]）的特性，但可以迅速新增與刪除
- 選擇排序法：
  - 陣列適合嗎？
  - 串列適合嗎？
- 插入排序法：
  - 陣列適合嗎？
  - 串列適合嗎？

[範例] insertion_sort

[範例] selection_sort

【範例】使用 STL 的 sort

- 具備『隨機存取迭代器』的容器可用 **std::sort**
  - 要相等元素保留在容器中原有的順序時可改用 **std::stable_sort()**
  - 要對自訂類別物件或自訂規則排序主要有三種方法：
    - 多載該類別的 **operator<**
    - 提供比較函式指標 (function pointer) 作為第三個引數
    - 提供比較函式物件 (function object) 作為第三個引數
      - 函式物件是製作一個類別並多載 operator() 來模擬一個函式
- 無法『隨機存取』的容器，要看該容器是否提供排序的成員函式：
  - 例如：
    - **std::list<Type>::sort()**

[範例] sort.cpp

【練習】逆向排序

- 試著用函式物件的方式，作逆向排序

```
struct LargerThan {
 inline bool operator()(int a, int b) const {
   // TODO
 }
};
```

可以使用結構 (struct) 或類別 (class) ，一般在沒有任何私有成員時，我們可以使用 struct 簡化語法

當 a 排在 b 前面時，operator() 回傳 true 否則為 false

inline 關鍵字表示該函式內容可能會被直接複製貼上在呼叫他的地方，而不是像一般函式是用呼叫的。如果函式內容少卻呼叫頻繁的話，效率可能會比較好。

[練習] ex9A.cpp

【練習】資料排序

- 試著用函式物件的方式，由小到大依照年齡排序，年齡相同時使用名字的英文字典順序
  - **string** 預設就可以做比較 (用英文字典順序)：
    - **string("abc") < string("bac")**

```
struct CompPerson {
  inline bool operator()(const Person &a,
                         const Person &b) const {
    // TODO
  }
};
```

[練習] ex9B.cpp

【練習】特定順序排序

- 我們想要排完順序後是單數在前、偶數在後、單數遞增、偶數遞減

範例輸入： 5 8 7 4 8 1 3 0 7 2

範例輸出： 1 3 5 7 7 8 8 4 2 0

[練習] ex9C.cpp