

Lab of Object-Oriented Programming: STL

黃威、陳岳紘、邱彥翔
2022

使用 moodle 點名

請登入實習課的 moodle 課程


點擊出缺席並完成今日的點名

- 邱彥翔 - 108703017@nccu.edu.tw

E-mail 格式

- 標題：[OOP111] + 問題
- 必須包含系級學號姓名
- 請附上有問題的**部分**程式碼或截圖

 討論區

 出缺席

Outline

- Container - 容器
- Iterator - 迭代器
- Algorithms - 演算法

<https://onlinegdb.com/Rt42oV5mQ>

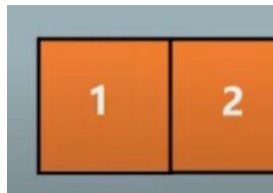
Container 容器

- 循序式容器 (Sequence containers)
 - vector, linked list
- 關聯式容器 (Associative containers)
 - set, map
- 容器配接器 (Container adapter)
 - stack, queue, priority_queue

循序式容器 (Sequence containers)

有序串列兩種製作方式之比較

- < Vector > - 連續記憶體



- < List >

sequential List (array)

V.S

Linked List

① 節點空間小 □ 1个

① 節點空間大 □ 2个

② 空間利用低 50%

② 空間利用高 100%

③ 插入 $O(n)$

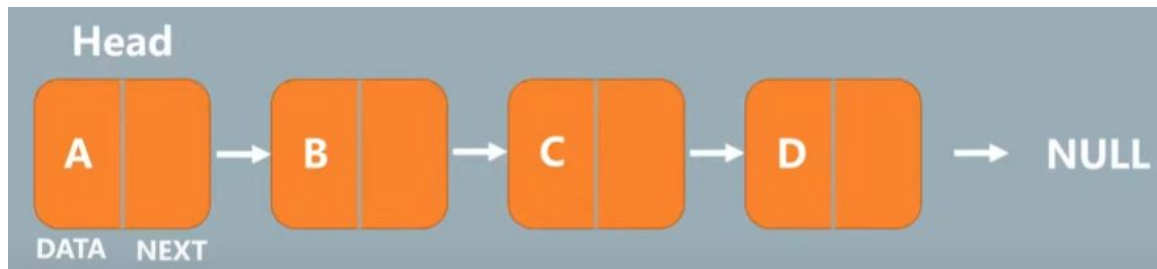
③ 插入 $O(1)$

④ 合併 $O(n)$

④ 合併 $O(1)$ 改尾指標指向另一個首指標

⑤ 隨機存取 $O(1)$

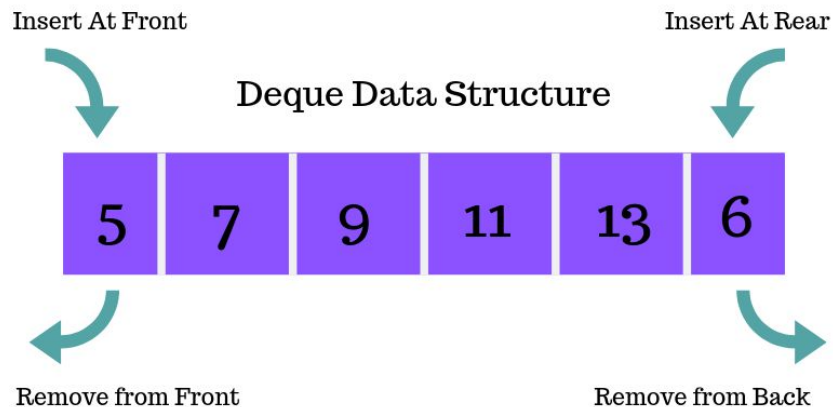
⑤ 隨機存取 $O(n)$: 需從開頭 search



循序式容器 (Sequence containers)

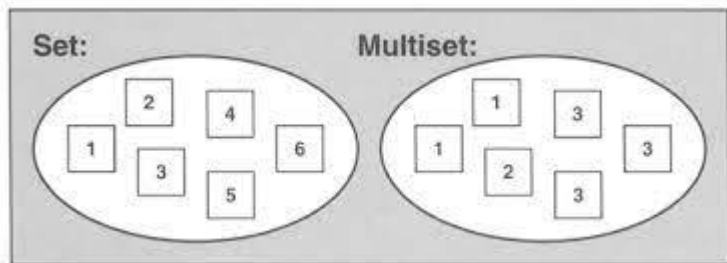
- < deque >

大致跟Vector相同且整合 List & Vector 的優點



關聯式容器 (Associative containers)

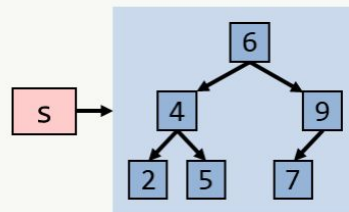
- < Set >
- < Multiset >



```
#include <set>
```

```
set<Key>
```

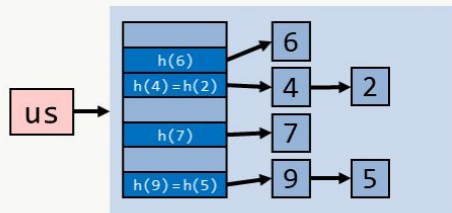
```
multiset<K>
```



```
#include <unordered_set>
```

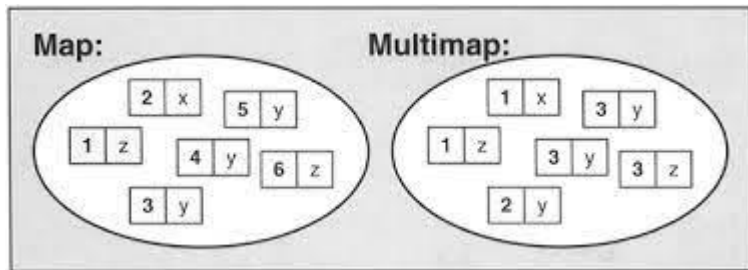
```
unordered_set<Key>
```

```
unordered_multiset<Key>
```



關聯式容器 (Associative containers)

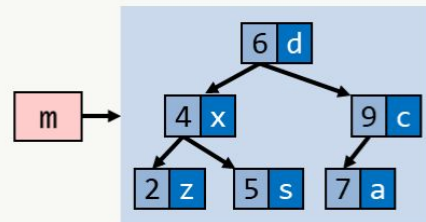
- `< Map >`
- `< Multimap >`



```
#include <map> // balance tree -- red_black tree
```

`map<Key, Value>`

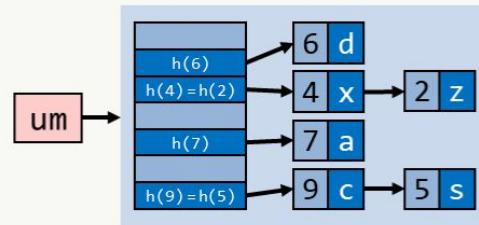
`multimap<K, V>`



```
#include <unordered_map> // hash table
```

`unordered_map<Key, Value>`

`unordered_multimap<Key, Value>`



關聯式容器 (Associative containers)

	map	unordered_map
優點	平衡樹 大部分的操作都可以在logn內完成	雜湊表 查找速度快
缺點	空間占用率高 需額外紀錄父節點、子節點及紅黑性質	建表較耗時
適用	有順序要求問題	查找問題

Set & Multiset

- Set為一個Value, 是獨立存在的
- Multiset允許相同Value存取

```
set<int> numSet;  
multiset<int> numSet;
```

Map & Multimap

- Map第一個為關鍵字(key), 是獨立存在的, 第二個為該關鍵字的值(value)
- Multimap一樣為允許相同關鍵字的Value存取

Map key-value 為 (one-to-one)的情況, 如學生的學號與其姓名

```
map<string, string> mapStudent;  
multimap<string, string> mapStudent;
```

Map 的特色

- 資料結構為red-black tree
 - balence tree (排序資料)
 - find(), insert() 的 time complex 皆為 $O(\log n)$

Iterator

```
int main(){
    vector<int> values = {1, 2, 3, 4, 5};

    for (int i = 0; i < values.size(); i++){
        cout << values[i] << " ";
    }
    cout << endl;

    for (vector<int>::iterator it = values.begin(); it != values.end(); it++){
        cout << *it << " ";
    }
    cout << endl;

    for (auto it = values.begin(); it != values.end(); it++){
        cout << *it << " ";
    }
    cout << endl;

    for (int value : values){
        cout << value << " ";
    }
    cout << endl;
```

Container member function

```
c.insert(pos,elem);  
c.insert(pos,n.elem); 插入n個elem  
c.insert(pos,beg,end); 在pos出插入beg，end區間內的所有元素。  
c.push_back(elem);  
c.pop_back();  
c.erase(pos); 刪除pos上的元素，返回下一個元素  
c.erase(beg,end);  
c.resize(num);將元素數量改為num，如果size變大了，多出來的新元素都要一default方式構建。  
c.resize(num,elem);將元素數量改為num，如果size變大了，多出來的新元素是elem的副本。  
c.clear();刪除所有。
```

Exercise 10

內容：

設計一個編碼轉換程式，輸入相對的指令，轉成先對應的數字與結果，印出來。

- step 1: 從 a-z 遍歷，先對每個字母建立一個array，array的建立方式為：將小寫字母轉成int，減去96(按照編碼表)後，從0開始將每個element放入array，最後將該字母與該字母生成的array組成一組，放入map。
 - 如下：

```
map<char, vector<int>> azmap;
char c;
for (c='a'; c<='z'; c++){
    int now = (int)c - 96;
    vector<int> vect;
    for(int i = 0; i < now; i++){
        cout << i << " ";
        vect.push_back(i);
    }
    cout << endl;
    azmap[c] = vect;
}
```
- step 2: 指令共有n行，每一行有三個元素：token, command, element。token範圍為a-z，決定該行的結果產生要調用哪一個字母的array，command為指令種類，共三種0,1,2分別對應不同的指令，element為command搭配需要搜尋的元素。
 - command == 0: 代表要回傳element在array存在的位置（從0開始算的位置）。
 - command == 1: 要先reverse array，再回傳element在array存在的位置（從0開始算的位置）。
 - command == 2: 要先將array的偶數部分找出來，自成一陣列，並排序，最後回傳「第element個」元素本人（不是位置），假設找不到，或者element超過array的size，則回傳0。
- step 3: 將所有程序的結果，用空格連成一串，印出來。

Exercise 10

輸入說明

一行指令，以三個元素組成每一行。

- input解釋：z的array，0 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25

範例輸入

```
z 0 6
z 1 6
z 2 6
```

輸出說明

每一行產生的結果，依序print出來，每一行的結果間以“空格”串連。

- output解釋：
 - z 0 6 --> 6在第6個元素
 - z 1 6 --> 反轉後，6在第19個元素
 - z 2 6 --> 擷取出偶數陣列後，第6個元素為10

範例輸出

```
6 19 10
```


**Any
questions?**

Reference

➤ STL:

<https://www.youtube.com/watch?v=HHzEKQzfU3s>

<https://learnersbucket.com/tutorials/data-structures/implement-deque-data-structure-in-javascript/>

➤ Map:

<https://www.scaler.com/topics/cpp/multiset-in-cpp/>

https://hackingcpp.com/cpp/std/associative_containers.html

➤