

C++ 常用 Container

1. Queue

- 就是...queue XD
- 照順序處理資料的時候用
- header: <queue>
- 宣告: `queue<data_type> queue_name;`
- Functions
 - ◆ `push(d)`: 加入 d
 - ◆ `pop()`: 刪除第一項
 - ◆ `size()`: 回傳項目個數
 - ◆ `empty()`: 回傳 boolean, 判斷是否為空
 - ◆ `front()`: 回傳第一項
- 應用: BFS(參考附錄)
- 範例: [queue.cpp](#) [queue.exe](#)

```
1  #include <stdio>
2  #include <queue>
3
4  using namespace std;
5
6  int main()
7  {
8      int inp[9] = {1, 4, 1, 4, 2, 1, 3, 5, 6};
9
10     queue<int> que;
11
12     /*Push in elements*/
13     for(int i = 0; i < 9; i++)
14     {
15         que.push(inp[i]);
16     }
17
18     /*Getting all elements*/
19     while(!que.empty()) //Or while(que.size() > 0)
20     {
21         printf("%d ", que.front());
22
23         que.pop();
24     }
25
26     return 0;
27 }
```

C:\Windows\System32\cmd.exe

D:\Documents\cfTemp>g++ -o test test.cpp

D:\Documents\cfTemp>test

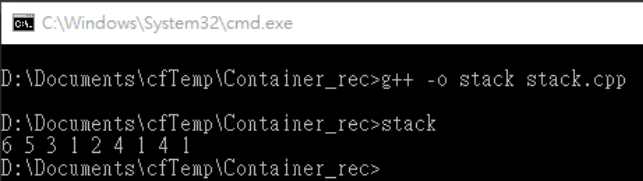
1 4 1 4 2 1 3 5 6

D:\Documents\cfTemp>

2. Stack

- 先進後出
- header: <stack>
- 宣告: `stack<data_type> stack_name;`
- Functions
 - ◆ `push(d)`: 加入 d
 - ◆ `pop()`: 刪除堆疊最上方項
 - ◆ `size()`: 回傳項目個數
 - ◆ `empty()`: 回傳 boolean, 判斷是否為空
 - ◆ `top()`: 取得第一項
- 範例:[stack.cpp](#) [stack.exe](#)

```
1 #include <stdio>
2 #include <stack>
3
4 using namespace std;
5
6 int main()
7 {
8     int inp[9] = {1, 4, 1, 4, 2, 1, 3, 5, 6};
9
10    stack<int> stk;
11
12    /*Push in Elements*/
13    for(int i = 0; i < 9; i++)
14    {
15        stk.push(inp[i]);
16    }
17
18    /*Get All Elements*/
19    while(!stk.empty()) // Or while(stk.size() > 0)
20    {
21        printf("%d ", stk.top());
22        stk.pop();
23    }
24
25    //First In Last Out
26
27    return 0;
28 }
29
```



```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>g++ -o stack stack.cpp
D:\Documents\cfTemp\Container_rec>stack
6 5 3 1 2 4 1 4 1
D:\Documents\cfTemp\Container_rec>
```

3. Vector

- 動態陣列
- header: <vector>
- 宣告: `vector<data_type> vector_name;`
- Functions
 - ◆ `push_back(data_type d):` 加入 d
 - ◆ `size():` 回傳元素個數
 - ◆ `empty():` 回傳 boolean, 判斷是否為空
 - ◆ `clear():` 清除所有元素
 - ◆ `resize`
 - a. `resize(int sz):` 把 size 改成 sz (C++ 11 才有)
 - b. `resize(int sz, data_type d):` 把 size 改成 sz, 多出來的元素填 d
- 其他
 - ◆ 可以用 `vector_name[x]` 存取第 x 個元素 (x 必須 < `size()`)
 - ◆ 如果要像陣列一樣先定義大小再 `vector_name[x] = value` 可以先 `resize`
 - ◆ 應用: 建圖 (參考附錄)
- 範例: [vector.cpp](#) [vector.exe](#)

```
1 #include <cstdio>
2 #include <vector>
3
4 using namespace std;
5
6 int main()
7 {
8     int inp[9] = {1, 4, 1, 4, 2, 1, 3, 5, 6};
9
10    vector<int> v1;
11    vector<int> v2;
12
13    v2.resize(9);
14
15    printf("Before: Size of v1: %d; size of v2: %d\n", v1.size(), v2.size());
16
17    /*Insert Elements*/
18    for(int i = 0; i < 9; i++)
19    {
20        v1.push_back(inp[i]);
21        v2[i] = inp[i];
22    }
23
24    printf("After: Size of v1: %d; size of v2: %d\n\n", v1.size(), v2.size());
25
26    /*Accessing Element*/
27
28    printf("v1=");
29    for(int i = 0; i < v1.size(); i++)
30    {
31        printf("%3d ", v1[i]);
32    }
33    printf("\nv2=");
34    for(int i = 0; i < v2.size(); i++)
35    {
36        printf("%3d ", v2[i]);
37    }
38
39    printf("\n---\n");
40
41    v1[3] = 25;
42    v2[5] = 78;
43    v1[8] = v1[3] + v2[5];
44
45    printf("v1=");
46    for(int i = 0; i < v1.size(); i++)
47    {
48        printf("%3d ", v1[i]);
49    }
50    printf("\nv2=");
51    for(int i = 0; i < v2.size(); i++)
52    {
53        printf("%3d ", v2[i]);
54    }
55    printf("\n");
```

```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>g++ -o vector vector.cpp
D:\Documents\cfTemp\Container_rec>vector
Before: Size of v1: 0; size of v2: 9
After: Size of v1: 9; size of v2: 9
v1= 1  4  1  4  2  1  3  5  6
v2= 1  4  1  4  2  1  3  5  6
---
v1= 1  4  1 25  2  1  3  5 103
v2= 1  4  1  4  2 78  3  5  6
D:\Documents\cfTemp\Container_rec>
```

4. String

- 字串(其實不算是 container, 不過常放在一起用就一起打)
- header: <string>
- 比較方便的字串
- Functions
 - ◆ size(): 回傳字串長度
 - ◆ length(): 同上
 - ◆ clear(): 清除字串
 - ◆ c_str(): 回傳內容相同的字元陣列
 - ◆ compare(str): 跟 strcmp 回傳值相同
- I/O: 要用 cin, cout 或是先輸入字元陣列再 construct/assign
- Operators
 - ◆ 可以直接用=指定(字元陣列、字元、string)
 - ◆ 可以用+, += 來 append(串接)
 - ◆ 可以用 string_name[x]來表示第 x 個字元
 - ◆ 可以用>, <, ==等來判斷大小或相等, 判斷方式為第一個相異字元
- 範例: [string.cpp](#) [string.exe](#)

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main()
7 {
8     /*Append*/
9     cout << "Say something: ";
10
11     string s1 = "What you're gonna say next is \n";
12     string s2;
13
14     cin >> s2;
15
16     s2 += ", \"right?\n\n";
17
18     cout << s1 + s2;
19
20     /*Modifying*/
21     string test = "abcdefg";
22     cout << "string size: " << test.size() << " " << test << "\n---\n";
23
24     test[1] = 'a';
25     test[2] = 'a';
26
27     cout << "string size: " << test.size() << " " << test << "\n\n";
28
29     /*Comparison*/
30     string s3;
31
32     cout << "Say something again: ";
33     cin >> s3;
34     if(s3.compare("meow") == 0) cout << "Yes\n";
35     if(s3 > "aaa") cout << "greater\n";
36
37     return 0;
38 }
39
```

```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>g++ -o string string.cpp
D:\Documents\cfTemp\Container_rec>string
Say something: meow
What you're gonna say next is "meow, "right?

string size: 7:abcdefg
---
string size: 7:aaadefg

Say something again: meow
Yes
greater
D:\Documents\cfTemp\Container_rec>
```

5. Priority Queue(Heap)

- 用來快速取出最大/最小值
- header: <queue>
- 宣告
 - ◆ 最大值: `priority_queue<data_type> heap_name;`
 - ◆ 最小值: `priority_queue<data_type, vector<data_type>, greater<data_type> > heap_name;`
 - ◆ 宣告最小值的方法限有定義"<"的型別(eg. int, double, pair)
- Functions:
 - ◆ `push(d)`: 加入 d
 - ◆ `pop()`: 刪除頂端元素(最大/最小值)
 - ◆ `top()`: 取得頂端元素(最大/最小值)
 - ◆ `size()`: 取得大小
 - ◆ `empty()`: 回傳 boolean, 判斷是否為空
- 範例: [priority_queue.cpp](#) [priority_queue.exe](#)

```
1 #include <iostream>
2 #include <queue>
3
4 using namespace std;
5
6 int main()
7 {
8     int inp[9] = {1, 4, 1, 4, 2, 1, 3, 5, 6};
9
10    priority_queue<int> pq_max; //Max Heap
11    priority_queue<int, vector<int>, greater<int> > pq_min; //Min Heap
12
13    /*Push In Elements*/
14    for(int i = 0; i < 9; i++)
15    {
16        pq_max.push(inp[i]);
17        pq_min.push(inp[i]);
18    }
19
20    /*Pop Out All Elements*/
21    //Max Heap
22    while(!pq_max.empty()) //Or while(pq_max.size() > 0)
23    {
24        printf("%d ", pq_max.top());
25        pq_max.pop();
26    }
27    printf("\n");
28
29    //Min Heap
30    while(!pq_min.empty()) //Or while(pq_min.size() > 0)
31    {
32        printf("%d ", pq_min.top());
33        pq_min.pop();
34    }
35    printf("\n");
36
37
38
39    return 0;
40 }
```

```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>g++ -o priority_q
D:\Documents\cfTemp\Container_rec>priority_queue
6 5 4 4 3 2 1 1 1
1 1 1 2 3 4 4 5 6
D:\Documents\cfTemp\Container_rec>
```

6. Pair

- 一對資料
- 比較時先比第一項，第一項相同時比第二項
- header: <utility>
- 宣告: `pair<data_type_1, data_type_2> pair_name;` 兩個資料形態可以不同
- Members
 - ◆ `first`: 存取第一項
 - ◆ `second`: 存取第二項
- Operators
 - ◆ 可以用 `=` 指定
- 其他
 - ◆ 將兩個元素 `e1`, `e2` 做成 `pair`
 - a. `make_pair(e1, e2);`
 - b. `{e1, e2}` // 只有 C++ 11 後才有，而且有時候不管用:P
- 範例:[pair.cpp](#) [pair.exe](#)

```
1 #include <iostream>
2 #include <utility>
3
4 using namespace std;
5
6 int main()
7 {
8     pair<int, int> p1, p2;
9     pair<char, int> p3;
10
11     /*Accessing*/
12     p1 = make_pair(3, 5);
13     p2 = p1;
14
15     printf("p1: %d %d\n", p1.first, p1.second);
16     printf("p2: %d %d\n", p2.first, p2.second);
17     printf("---\n");
18
19     p1.first = 8;
20     p2.second = 13;
21
22     printf("p1: %d %d\n", p1.first, p1.second);
23     printf("p2: %d %d\n", p2.first, p2.second);
24     printf("\n");
25
26     /*Comparison*/
27     p3 = {'a', 97};
28     pair<char, int> p4 = {'b', 99};
29     pair<char, int> p5 = {'b', 100};
30
31     printf("p3");
32     if(p3 < p4) printf(" < ");
33     else printf(" > ");
34     printf("p4");
35     if(p4 < p5) printf(" < ");
36     else printf(" > ");
37     printf("p5\n");
38
39
40
41     return 0;
42 }
```

```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>g++ -o pair pair.cpp
D:\Documents\cfTemp\Container_rec>pair
p1: 3 5
p2: 3 5
---
p1: 8 5
p2: 3 13
p3 < p4 < p5
D:\Documents\cfTemp\Container_rec>
```

7. Set

- 集合，重覆元素不會重覆出現(另參考 multiset)
- header: <set>
- 宣告: set<data_type> set_name;
- Functions
 - ◆ insert(d): 插入 d
 - ◆ count(d): 回傳元素 d 的個數
 - ◆ clear(): 清除 set
 - ◆ find(d): 在 set 中尋找 d，回傳 iterator，沒找到回傳 end()
 - ◆ empty(): 回傳 boolean，判斷是否為空
- 範例: [set.cpp](#) [set.exe](#)

```
1  #include <iostream>
2  #include <set>
3
4  using namespace std;
5
6  int main()
7  {
8      int inp[9] = {1, 4, 1, 4, 2, 1, 3, 5, 6};
9
10     set<int> st;
11
12     /*Insert Elements*/
13     for(int i = 0; i < 9; i++)
14     {
15         st.insert(inp[i]);
16     }
17
18     /*Count Elements*/
19     for(int i = 0; i <= 9; i++)
20     {
21         printf("%d: %d\n", i, st.count(i));
22     }
23
24     return 0;
25 }
```

```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>.set
0: 0
1: 1
2: 1
3: 1
4: 1
5: 1
6: 1
7: 0
8: 0
9: 0
D:\Documents\cfTemp\Container_rec>
```

8. Map

- 映射，一個 key 對應一個 value(另參考 multimap)
- 宣告：map<key_data_type, value_data_type> map_name;
- Functions:
 - ◆ insert(p): 插入<Key, Value>對 p。若 Key 已存在於該 map 中，則 insert 失敗
 - ◆ find(K): 在 map 中尋找 Key 值為 K 的元素。回傳 iterator，找不到則回傳 end()
 - ◆ count(K): 回傳 Key 值為 K 的元素數量
 - ◆ clear(): 清除 map
 - ◆ size(): 回傳元素個數
 - ◆ empty(): 回傳 boolean，判斷是否為空
- 其他
 - ◆ 可以用 map_name[key] 得到對應 value 值
 - ◆ 可以用 map_name[key] = value; 代替 insert 及重新賦予 key 的 value 值
- 範例：[map.cpp](#) [map.exe](#)

```
1 #include <iostream>
2 #include <map>
3
4 using namespace std;
5
6 int main()
7 {
8     map<int, int> mp1;
9     map<char, int> mp2;
10
11     /*Insert Elements*/
12     mp1.insert(make_pair(1, 2));
13     mp1.insert({2, 3});
14     mp2.insert({'a', 97});
15     mp2['b'] = 98;
16
17     printf("%d:%d\n", 1, mp1[1]);
18     printf("%d:%d\n", 2, mp1[2]);
19     printf("%c:%d\n", 'a', mp2['a']);
20     printf("%c:%d\n", 'b', mp2['b']);
21
22     /*Modifying*/
23     mp1.insert({1, 4}); // Key Already Exists => Nothing Happens
24     mp1[2] = 8;
25
26     printf("%d:%d\n", 1, mp1[1]);
27     printf("%d:%d\n", 2, mp1[2]);
28
29     system("pause");
30     return 0;
31 }
```

```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>g++ -o map map.cpp
D:\Documents\cfTemp\Container_rec>map
1:2
2:3
a:97
b:98

1:2
2:8
請按任意鍵繼續 . . .
D:\Documents\cfTemp\Container_rec>
```


附錄

1. Iterators

- 可以先當作「Container 的指標」
- 宣告: `container::iterator iterator_name;`
 - ◆ 例: `vector<int>::iterator it;`
- Container iterators

container::iterator it;		
Container	Has Iterator	Type of *it
<code>queue<T></code>	X	-
<code>stack<T></code>	X	-
<code>vector<T></code>	O	T
<code>string</code>	O	char
<code>priority_queue<T></code>	X	-
<code>pair<T1, T2></code>	X	-
<code>set<T></code>	O	T
<code>map<T1, T2></code>	O	<code>pair<T1, T2></code>

- `begin()`, `end()` (設 `obj` 是一個 container 的實體)
 - ◆ `obj.begin()`: 指到 `obj` 第一項的 iterator
 - ◆ `obj.end()`: 指到 `obj` 最後一項之後的一項的 iterator
 - ◆ 換句話說, `obj` 中所有元素包含在 `[begin, end)` 中
- Operators
 - ◆ `++/--`: 表示下一個/上一個 iterator
 - ◆ `vector` 的 iterator 可以用 `+=/-=(例 it += 3, it + 3)`
- `erase()`
 - ◆ 有 iterator 的 container 一般都有 `erase(iterator it)` 函式
 - ◆ 回傳值為「刪除 `it` 後的下一個 iterator」
- 其他
 - ◆ 可以用 `it -> member` 代替 `(*it).member` 來存取 member 欄位
 - ◆ 對 container 使用有關連續記憶體空間的函式(eg. `sort`, `lower_bound`)時, 必須使用 iterator
 - eg. `sort(obj.begin(), obj.end());`
 - ◆ 許多尋找類型的成員函式(eg. `set.find()`)找不到時會回傳 `end()`
 - ◆ 一些函式要求 iterator 當作參數或回傳值(eg. `erase`)
- 範例: [iterator.cpp](#) [iterator.exe](#)

```

1  #include <iostream>
2  #include <vector>
3  #include <set>
4  #include <map>
5  #include <algorithm>
6
7  using namespace std;
8
9  int main()
10 {
11     int inp[9] = {1, 4, 1, 4, 2, 1, 3, 5, 6};
12
13     /*Example 1: Vector Iterators*/
14
15     vector<int> vec;
16
17     //Insert Elements
18     for(int i = 0; i < 9; i++)
19     {
20         vec.push_back(inp[i]);
21     }
22
23     sort(vec.begin(), vec.end()); //Using Iterators while Calling Function sort
24
25     //Regular Way to Get All Elements
26     for(int i = 0; i < vec.size(); i++)
27     {
28         printf("%d ", vec[i]);
29     }
30     printf("\n");
31     //Getting All Elements Using Iterators
32     for(vector<int>::iterator it = vec.begin(); it != vec.end(); it++)
33     {
34         printf("%d ", *it);
35     }
36     printf("\n\n");

```

C:\Windows\System32

D:\Documents\cfTemp

D:\Documents\cfTemp

1 1 1 2 3 4 4 5 6

1 1 1 2 3 4 4 5 6

```

43     /*Example 2: Set Iterator*/
44
45     set<int> st;
46
47     //Insert Elements
48     for(int i = 0; i < 9; i++)
49     {
50         st.insert(inp[i]);
51     }
52
53     //Finding All 0 - 9 Numbers in st
54     for(int i = 0; i <= 9; i++)
55     {
56         if(st.find(i) != st.end()) printf("%d:Yes\n", i);
57         else printf("%d:No\n", i);
58     }
59     printf("\n");
60
61

```

0:No

1:Yes

2:Yes

3:Yes

4:Yes

5:Yes

6:Yes

7:No

8:No

9:No

```

61
62     /*Example 3: Map Iterator*/
63
64     map<int, int> mp;
65     for(int i = 0; i < 9; i++)
66     {
67         mp[i] = inp[i];
68     }
69
70     for(map<int, int>::iterator it = mp.begin(); it != mp.end(); it++)
71     {
72         (*it).second -= 2; //Changing Value Is Available
73         printf("%d: %d+2\n", (*it).first, it->second); //(*it).first = it->first
74     }
75

```

0: -1+2

1: 2+2

2: -1+2

3: 2+2

4: 0+2

5: -1+2

6: 1+2

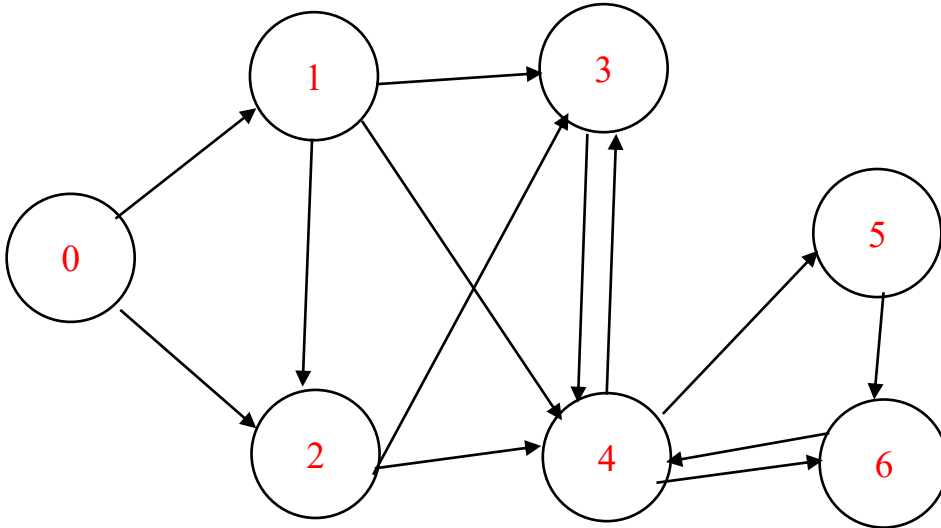
7: 3+2

8: 4+2

D:\Documents\cfTemp\Container_rec>

2. Vector 應用:建圖(adjacency list)

- 建立 vector 陣列，每個 vector 代表一個點(vertex)
- 將每個點的 child 編號 push_back 進該 vector 中
 - ◆ 若是無向圖，則 child 代表的點也要 push_back parent 的點編號
- 範例:[graph.cpp](#) [graph.exe](#)



```
1  #include <iostream>
2  #include <vector>
3  #include <set>
4  #include <map>
5  #include <algorithm>
6
7  using namespace std;
8
9  int main()
10 {
11     int from, to;
12     vector<int> G[7];
13
14     scanf("%d %d", &from, &to);
15
16     while(from + to > 0)
17     {
18         //Input: "v1 v2" represents
19         //v1 has a directed edge to v2, ends at "0 0"
20
21         G[from].push_back(to);
22
23         scanf("%d %d", &from, &to);
24     }
25
26     for(int i = 0; i < 7; i++)
27     {
28         printf("Children of v%d: ", i);
29
30         for(int j : G[i]) // For each integer j in G[i]
31         {
32             printf("%d ", j);
33         }
34
35         printf("\n");
36     }
37
38     return 0;
39 }
```

```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>g++
D:\Documents\cfTemp\Container_rec>graph.exe
0 1
0 2
1 2
1 3
1 4
2 3
2 4
3 4
4 3
4 6
5 6
6 4
0 0
Children of v0: 1 2
Children of v1: 2 3 4
Children of v2: 3 4
Children of v3: 4
Children of v4: 3 6
Children of v5: 6
Children of v6: 4
D:\Documents\cfTemp\Container_rec>
```

3. Queue 應用：BFS

- 將起點 push 入 queue 並將起點標記為已走過
- 將 front 取出，將其 children 推入 queue 並紀錄 children 已被走過
- 重複 b.直到取出終點或 traversal 完畢
- 範例(延續上圖)： [bfs.cpp](#) [bfs.exe](#)

```
6
7 FILE *inp;
8
9 bool vstd[7] = {0}; //Record if vertex is visited
10
11 int main()
12 {
13     inp = fopen("graph.txt", "r");
14
15     int from, to;
16     fscanf(inp, "%d %d", &from, &to);
17     vector<int> G[7];
18
19     while(from + to > 0)
20     { //Input: "v1 v2" represents
21         //v1 has a directed edge to v2, ends at "0 0"
22
23         G[from].push_back(to);
24
25         fscanf(inp, "%d %d", &from, &to);
26     }
27
28     queue<int> que;
29     que.push(0);
30     vstd[0] = true;
31
32     printf("BFS Traversal Order: ");
33     while(!que.empty()) //Traverse for all vertices
34     {
35         int root = que.front();
36         printf("%d ", root);
37
38         for(int i : G[root]) //Pushing in Children
39         {
40             if(!vstd[i])
41             {
42                 que.push(i);
43                 vstd[i] = true; //Record for visited
44             }
45         }
46
47         que.pop();
48     }
```

```
C:\Windows\System32\cmd.exe
D:\Documents\cfTemp\Container_rec>bfs
BFS Traversal Order: 0 1 2 3 4 5 6
D:\Documents\cfTemp\Container_rec>
```