

# NCKU Programming Contest Training Course

## Vector, Map, String

### 2018/02/23

---

**Jheng – Huang Hong**  
*a0987856762@gmail.com*

Department of Computer Science and Information Engineering  
National Cheng Kung University  
Tainan, Taiwan



# Outline

---

- **Vector**
- **Map**
- **String**



# Constructor

```
1 #include <vector>
2 #include <cstdio>
3 #include <cstring>
4
5 struct Point{
6     int x, y;
7 };
8
9 int main(){
10     std::vector<Point> first;           // empty vector of ints
11     std::vector<int> second (4,100);   // four ints with value 100
12     std::vector<int> third (second.begin(),second.begin()+2); // iterating through second
13     std::vector<int> fourth (third);  // a copy of third
14
15     for(int i=0; i<third.size(); i++){
16         printf("%d ",third[i]);
17     }
18     puts("");
19
20     return 0;
21 }
```

outputs: 100 100



# Push, Pop

```
1 #include <vector>
2 #include <cstdio>
3 #include <cstring>
4
5 using namespace std;
6
7 int main(){
8     vector<int> v;
9     v.push_back(1);
10    v.push_back(2);
11    v.push_back(3);
12    v.pop_back();
13
14    printf("outputs: ");
15    for(int i=0; i<v.size(); i++){
16        printf("%d ",v[i]);
17    }
18    puts("");
19
20    return 0;
21 }
```

outputs: 1 2

made by mike199250 & a711186



# Insert ,Erase

```
1 #include <vector>
2 #include <cstdio>
3 #include <cstring>
4
5 using namespace std;
6
7 int main(){
8     vector<int> v(3,1);           // 1 1 1
9     v.insert(v.begin()+1,2,2);  // 1 2 2 1 1
10    v.erase(v.begin()+3);       // 1 2 2 1
11
12    printf("outputs: ");
13    for(int i=0; i<v.size(); i++){
14        printf("%d ",v[i]);
15    }
16    puts("");
17
18    return 0;
19 }
```

outputs: 1 2 2 1



# Clear

- 多筆測資時很好用

```
1 #include <vector>
2 #include <cstdio>
3 #include <cstring>
4
5 using namespace std;
6
7 int main(){
8     vector<int> v(3,1);           // 1 1 1
9     v.clear();
10
11     if(v.empty())
12         printf("empty\n");
13     else
14         printf("not empty\n");
15
16     return 0;
17 }
```



# Combine with Sort

```
1 #include <vector>
2 #include <algorithm>
3 #include <cstdio>
4
5 using namespace std;
6
7 int main(){
8     vector<int> v;
9     v.push_back(3);
10    v.push_back(1);
11    v.push_back(2);
12
13    sort(v.begin(),v.end());
14
15    for(int i=0; i<v.size(); ++i)
16        printf("%d ",v[i]);
17
18    return 0;
19 }
```

output: 1 2 3

code by mike199250 & a711186



# Iterator

```
1 #include <vector>
2 #include <cstdio>
3 #include <cstring>
4
5 using namespace std;
6
7 int main(){
8     vector<int> v(3,2);           // 2 2 2
9
10    for(int i=0; i<v.size(); ++i)
11        printf("%d ",v[i]);
12    puts("");
13
14    vector<int>::iterator vector_it;
15    for(vector_it = v.begin(); vector_it != v.end(); ++vector_it)
16        printf("%d ",*vector_it);
17
18    return 0;
19 }
```

2	2	2
2	2	2



# Vector

- Array
  - 大小開到題目最大範圍
  - 使用`memset`(快)或`for`(慢)初始化
  - 移除陣列中某一元素，需將其後往前搬運



# Vector

- Vector
  - 不需宣告大小
  - Constructor
  - 插入/移除方便



# C++ Reference

---

<http://www.cplusplus.com/reference/>



# Uva - 11462

---

## Problem Description

You are given the ages (in years) of all people of a country with at least 1 year of age. You know that no individual in that country lives for 100 or more years. Now, you are given a very simple task of sorting all the ages in ascending order.



# Uva - 11462

---

## Input

There are multiple test cases in the input file. Each case starts with an integer  $n$  ( $0 < n \leq 2000000$ ), the total number of people. In the next line, there are  $n$  integers indicating the ages. Input is terminated with a case where  $n = 0$ . This case should not be processed.

## Output

For each case, print a line with  $n$  space separated integers. These integers are the ages of that country sorted in ascending order. Warning: Input Data is pretty big ( $\sim 25$  MB) so use faster IO.



# Uva - 11462

---

## Sample Input

5  
3 4 2 1 5  
5  
2 3 2 3 1  
0

## Sample Output

1 2 3 4 5  
1 2 2 3 3



# Outline

---

- **Vector**
- **Map**
- **String**



# Map

- Maps are associative containers that store elements formed by a combination of a key value and a mapped value, following a specific order.





# Constructor, Insert

```
1 #include <map>
2 #include <cstdio>
3
4 using namespace std;
5
6 int main(){
7     map<char, int> m;
8
9     m['a'] = 1;
10    m.insert(map<char, int> :: value_type('b',2));
11
12    map<char, int> m2(m);
13
14    printf("Output: %d",m['b']);
15
16    return 0;
17 }
```

Output: 2



# Find

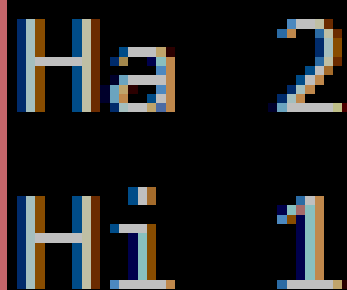
```
1 #include <map>
2 #include <string>
3 #include <cstdio>
4
5 using namespace std;
6
7 int main(){
8     map<string, int> m;
9
10    m["Hi"] = 1;
11    m.insert(map<string, int> :: value_type("Ha",2));
12
13    if(m.find("Hi") != m.end())
14        printf("m[\"Hi\"]: %d\n",m["Hi"]);
15
16    if(m.find("Ha") != m.end())
17        printf("m[\"Ha\"]: %d\n",m["Ha"]);
18
19    return 0;
20 }
```

```
m["Hi"] : 1
m["Ha"] : 2
```



# Iterator

```
1 #include <map>
2 #include <string>
3 #include <iostream>
4
5 using namespace std;
6
7 int main(){
8     map<string, int> m;
9     map<string, int>::iterator map_it;
10
11     m["Hi"] = 1;
12     m.insert(map<string, int> :: value_type("Ha",2));
13
14     for(map_it = m.begin(); map_it != m.end(); ++map_it)
15         cout << map_it->first << " " << map_it->second << endl;
16
17     return 0;
18 }
```



# Practice

---

# Uva - 10420



# Outline

---

- **Vector**
- **Map**
- **String**



# Constructor

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main ()
7 {
8     string s0 ("Initial string");
9     string s1;
10    string s2 (s0);
11    string s3 (s0, 8, 3);
12    string s4 ("A character sequence");
13    string s5 ("Another character sequence", 12);
14    string s6a (10, 'x');
15    string s6b (10, 42); // 42 is the ASCII code for '*'
16    string s7 (s0.begin(), s0.begin()+7);
17
18    return 0;
19 }
```

s1:  
s2: Initial string  
s3: str  
s4: A character sequence  
s5: Another char  
s6a: xxxxxxxxxxxx  
s6b: \*\*\*\*\*  
s7: Initial

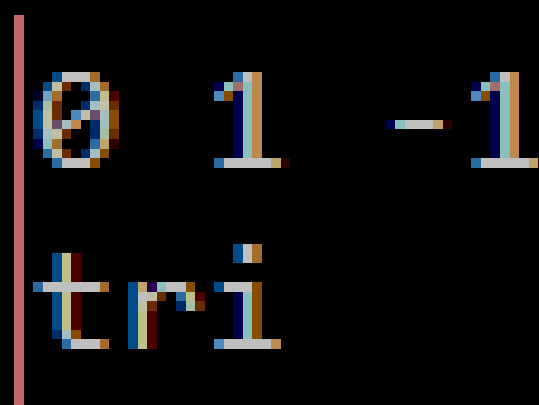
# Operator

```
1 #include <iostream>
2 #include <cstdio>          Initial string
3 #include <string>         Initial string + Append string
4
5 using namespace std;
6
7 int main (){
8     string s0 = "Initial string";
9     cout << s0 << endl;
10
11     s0 += " + Append string";
12     cout << s0 << endl;
13
14     return 0;
15 }
```



# Compare, Substring

```
1 #include <iostream>
2 #include <cstdio>
3 #include <string>
4
5 using namespace std;
6
7 int main (){
8     string s0 = "string2";
9     string s1 = "string2";
10    string s2 = "ring1";
11    string s3 = "string3";
12
13    printf("%d %d %d\n",s0.compare(s1),s0.compare(s2),s0.compare(s3));
14
15    string s4 = s0.substr(1,3);
16    printf("%s",s4.c_str());
17
18    return 0;
19 }
```





# String <-> Char array

```
1 #include <iostream>
2 #include <cstdio>
3 #include <string>
4
5 using namespace std;
6
7 int main (){
8     char c_array[] = "char array";
9     string s0 = "string";
10    string s1(c_array);
11
12    printf("%s\n",s0.c_str());
13    cout << s1 << endl;
14
15    return 0;
16 }
```

string  
char array



# Practice

---

# Uva – 10340

