C++基礎語法 Unit-11

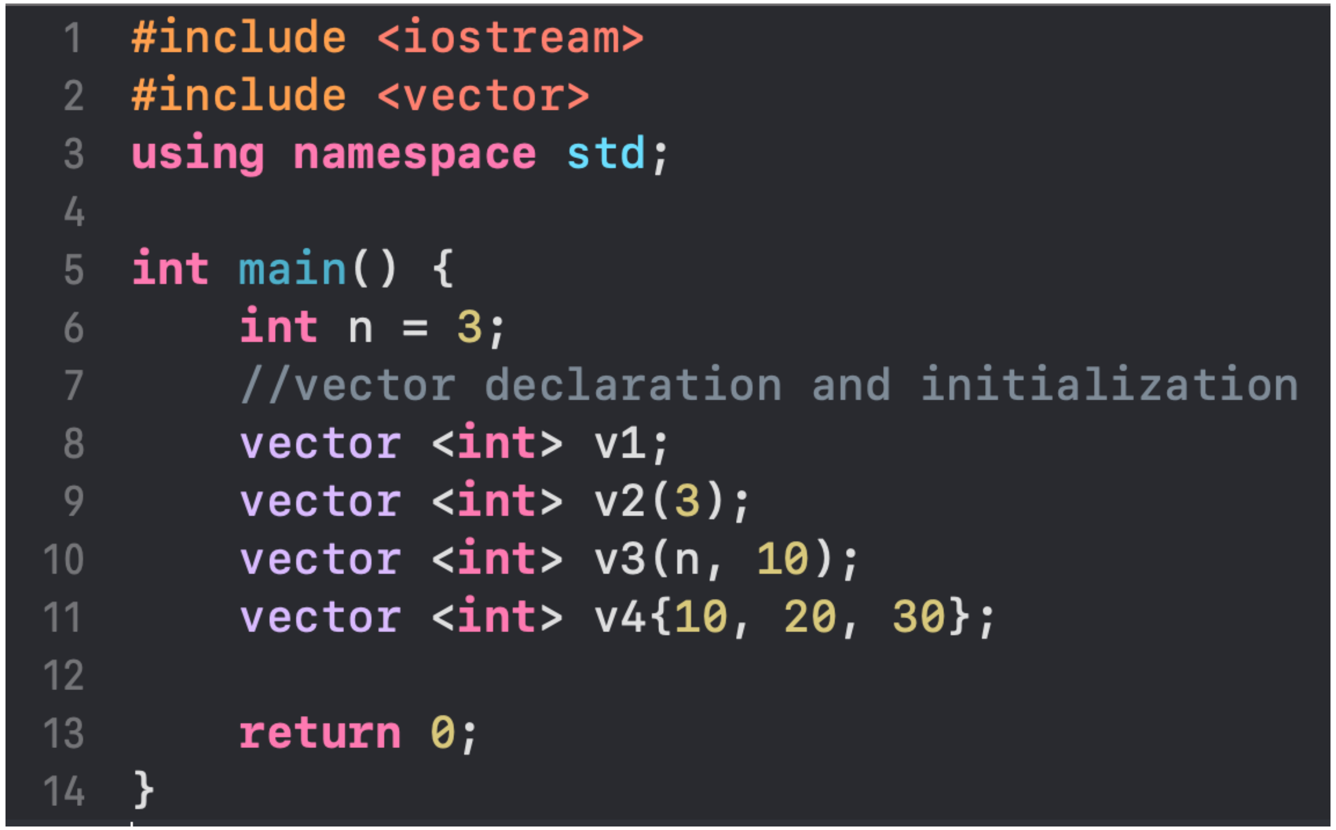
- **STL 標準模板庫 (Standard Template Library)**
- STL 線性資料結構：vector, queue, stack
- STL 非線性資料結構：set, map, priority_queue
- STL algorithm

vector

動態陣列

vector

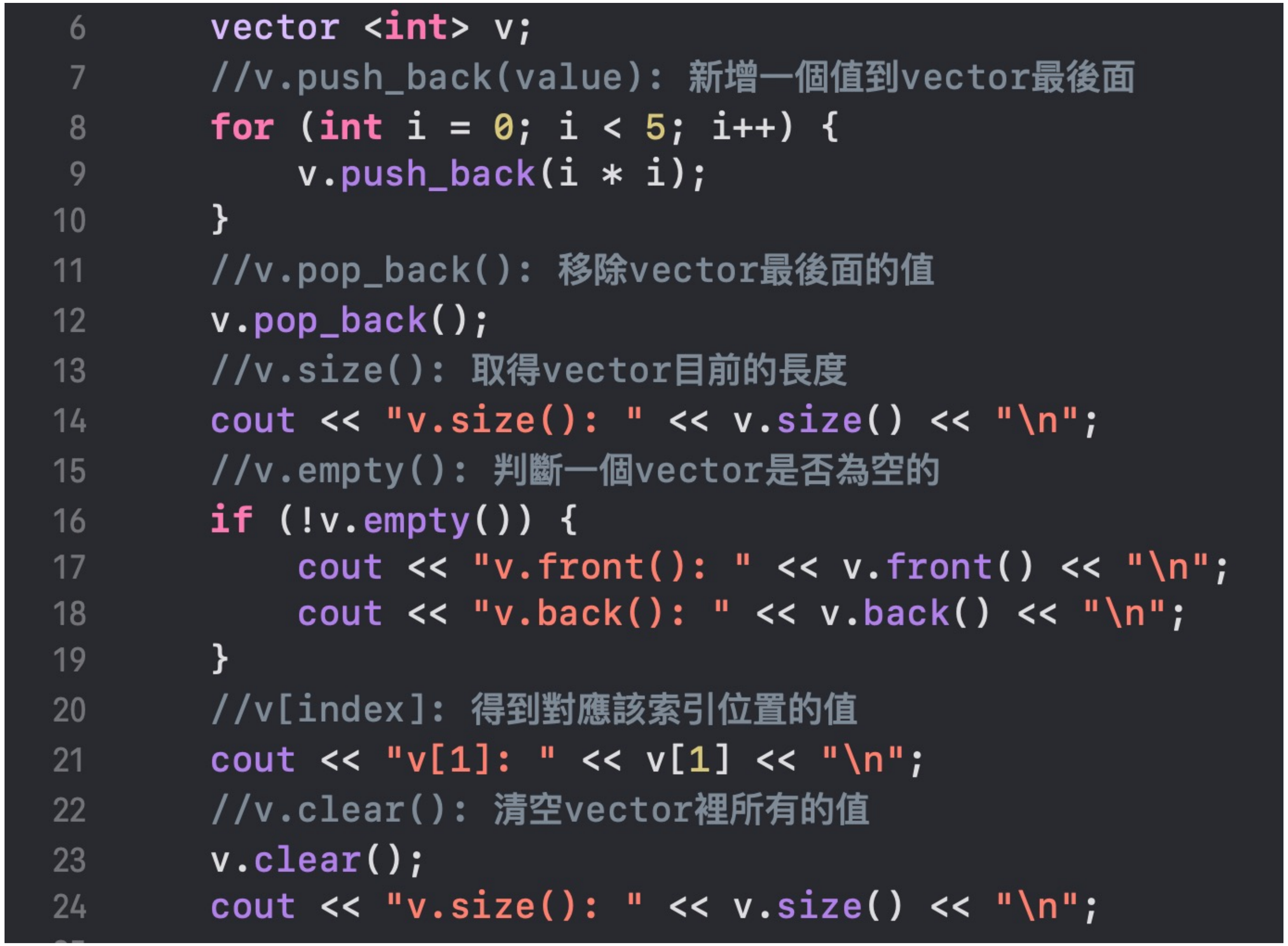
可以想成是個動態陣列，用法跟陣列很像
https://www.cplusplus.com/reference/vector/vector/



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

int main() {
    int n = 3;
    //vector declaration and initialization
    vector <int> v1;
    vector <int> v2(3);
    vector <int> v3(n, 10);
    vector <int> v4{10, 20, 30};

    return 0;
}
```

v.clear()

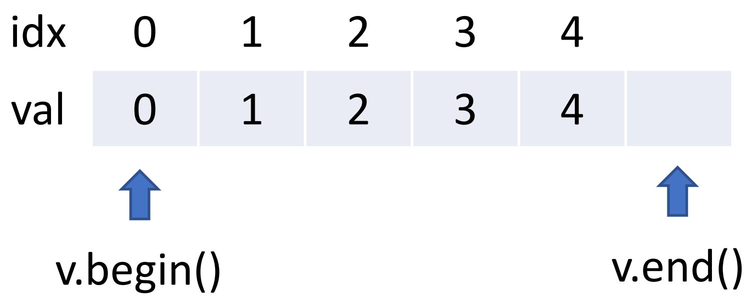
Example 11-2

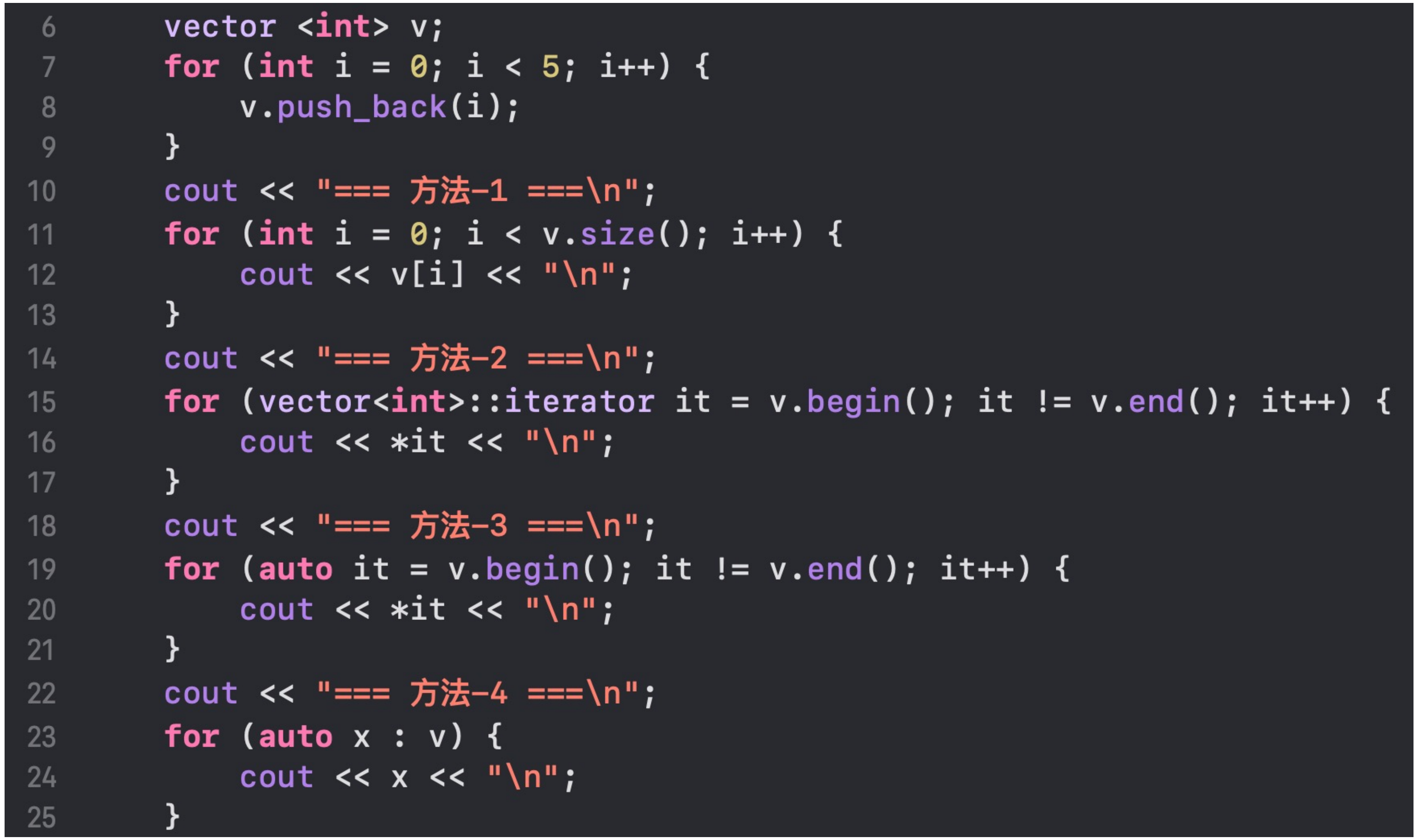


```
    vector <int> v;
    //v.push_back(value): 新增一個值到vector最後面
    for (int i = 0; i < 5; i++) {
        v.push_back(i * i);
    }
    //v.pop_back(): 移除vector最後面的值
    v.pop_back();
    //v.size(): 取得vector目前的長度
    cout << "v.size(): " << v.size() << "\n";
    //v.empty(): 判斷一個vector是否為空的
    if (!v.empty()) {
        cout << "v.front(): " << v.front() << "\n";
        cout << "v.back(): " << v.back() << "\n";
    }
    //v[index]: 得到對應該索引位置的值
    cout << "v[1]: " << v[1] << "\n";
    //v.clear(): 清空vector裡所有的值
    v.clear();
    cout << "v.size(): " << v.size() << "\n";
```

vector遍歷

- iterators
    - v.begin()
    - v.end()

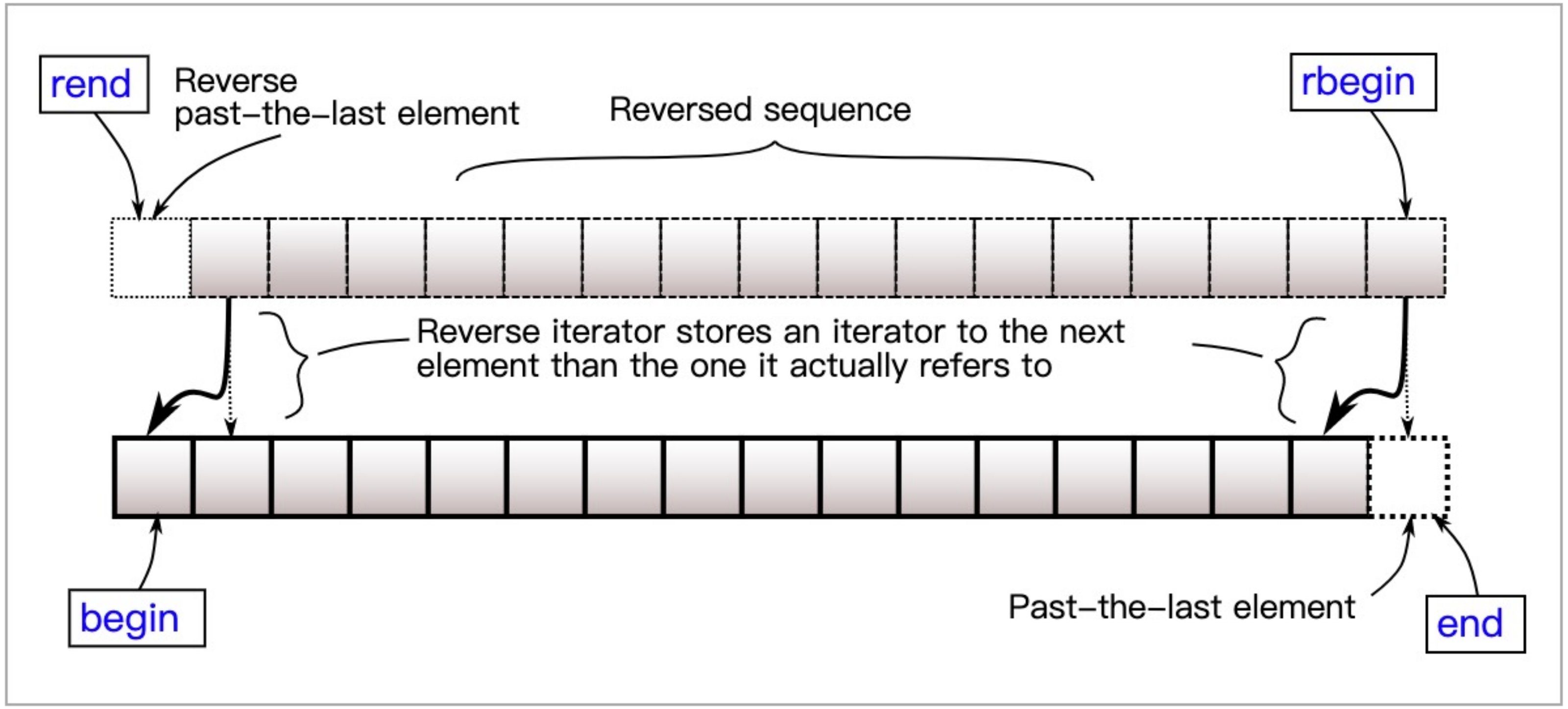




```
    vector <int> v;
    for (int i = 0; i < 5; i++) {
        v.push_back(i);
    }
    cout << "=== 方法-1 ===\n";
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << "\n";
    }
    cout << "=== 方法-2 ===\n";
    for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {
        cout << *it << "\n";
    }
    cout << "=== 方法-3 ===\n";
    for (auto it = v.begin(); it != v.end(); it++) {
        cout << *it << "\n";
    }
    cout << "=== 方法-4 ===\n";
    for (auto x : v) {
        cout << x << "\n";
    }
```

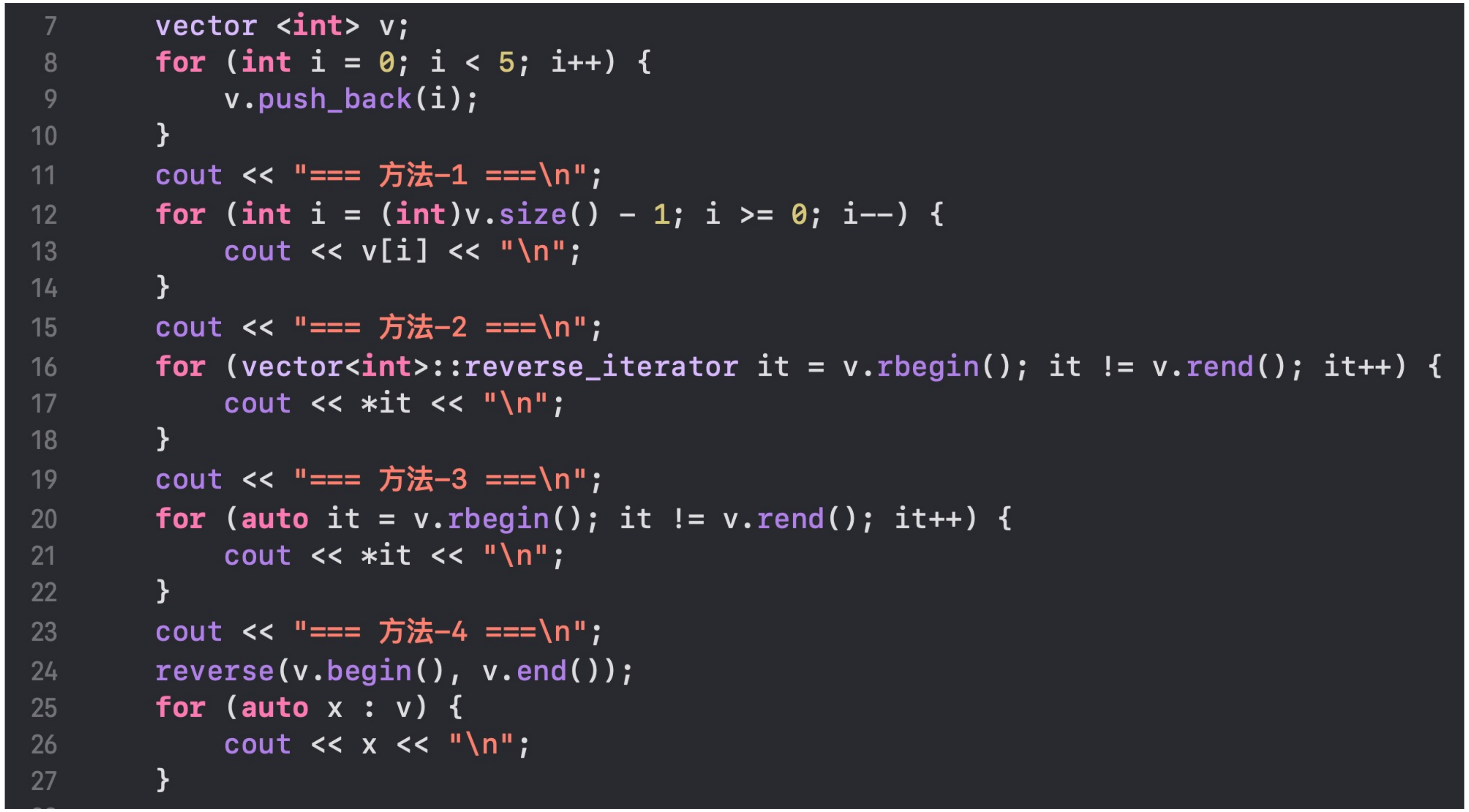
逆向遍歷

- iterators
  - v.rbegin()
  - v.rend()


rend
Reverse past–the–last element
Reversed sequence
rbegin
Reverse iterator stores an iterator to the next element than the one it actually refers to
begin
Past–the–last element
end

https://www.cplusplus.com/reference/vector/vector/

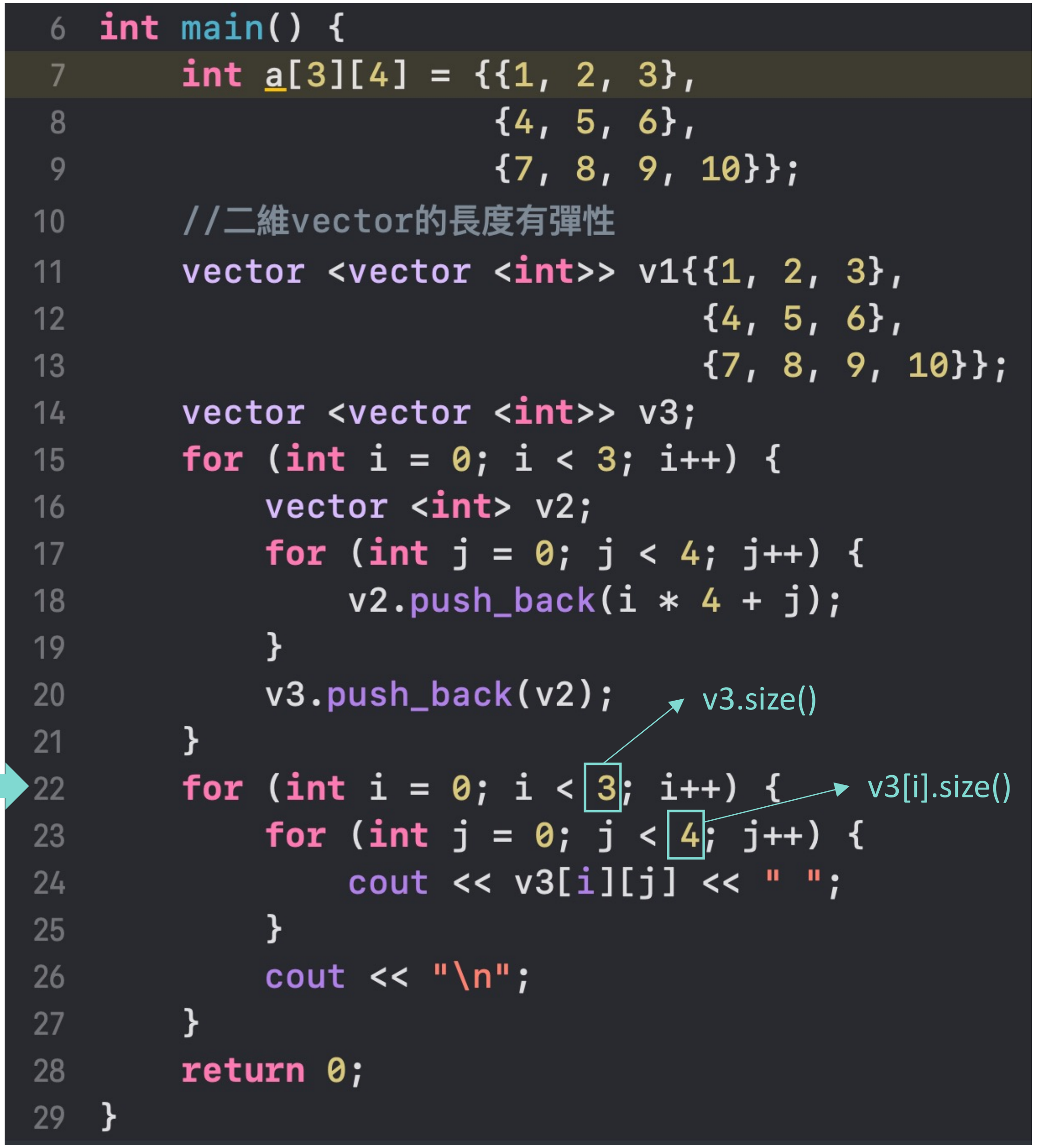
逆向遍歷



```cpp
    vector <int> v;
    for (int i = 0; i < 5; i++) {
        v.push_back(i);
    }
    cout << "=== 方法-1 ===\n";
    for (int i = (int)v.size() - 1; i >= 0; i--) {
        cout << v[i] << "\n";
    }
    cout << "=== 方法-2 ===\n";
    for (vector<int>::reverse_iterator it = v.rbegin(); it != v.rend(); it++) {
        cout << *it << "\n";
    }
    cout << "=== 方法-3 ===\n";
    for (auto it = v.rbegin(); it != v.rend(); it++) {
        cout << *it << "\n";
    }
    cout << "=== 方法-4 ===\n";
    reverse(v.begin(), v.end());
    for (auto x : v) {
        cout << x << "\n";
    }
```

vector of vectors

避免hard-coded values



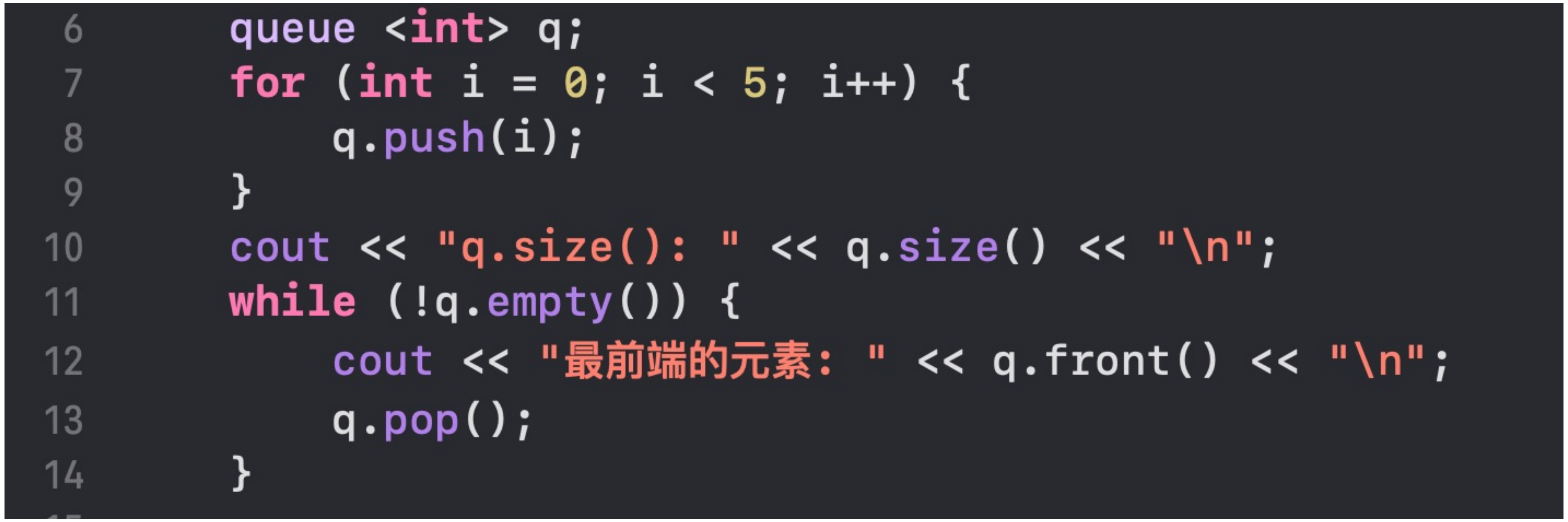
```cpp
int main() {
    int a[3][4] = {{1, 2, 3},
                   {4, 5, 6},
                   {7, 8, 9, 10}};
    //二維vector的長度有彈性
    vector <vector <int>> v1{{1, 2, 3},
                             {4, 5, 6},
                             {7, 8, 9, 10}};
    vector <vector <int>> v3;
    for (int i = 0; i < 3; i++) {
        vector <int> v2;
        for (int j = 0; j < 4; j++) {
            v2.push_back(i * 4 + j);
        }
        v3.push_back(v2);
    }
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            cout << v3[i][j] << " ";
        }
        cout << "\n";
    }
    return 0;
}
```

queue

FIFO (first in first out)

queue

- #include <queue>
- 沒有 .clear()



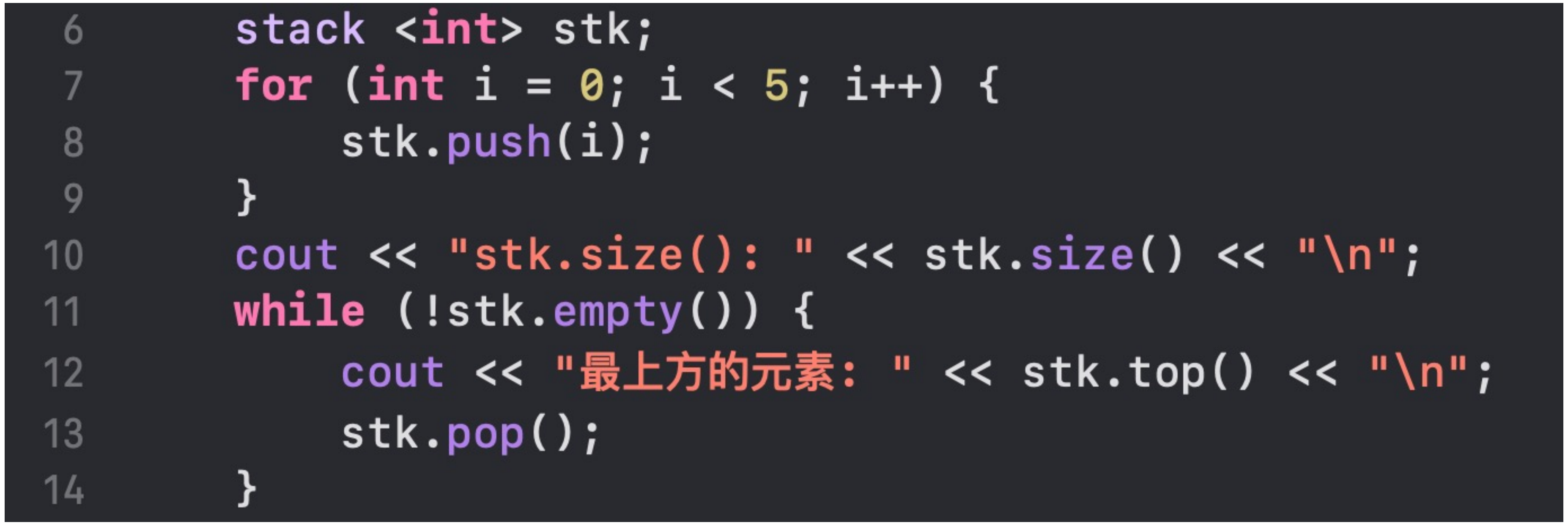
```
queue <int> q;
for (int i = 0; i < 5; i++) {
    q.push(i);
}
cout << "q.size(): " << q.size() << "\n";
while (!q.empty()) {
    cout << "最前端的元素: " << q.front() << "\n";
    q.pop();
}
```

stack

LIFO (last in first out)

stack

- #include <stack>
- 沒有 .clear()



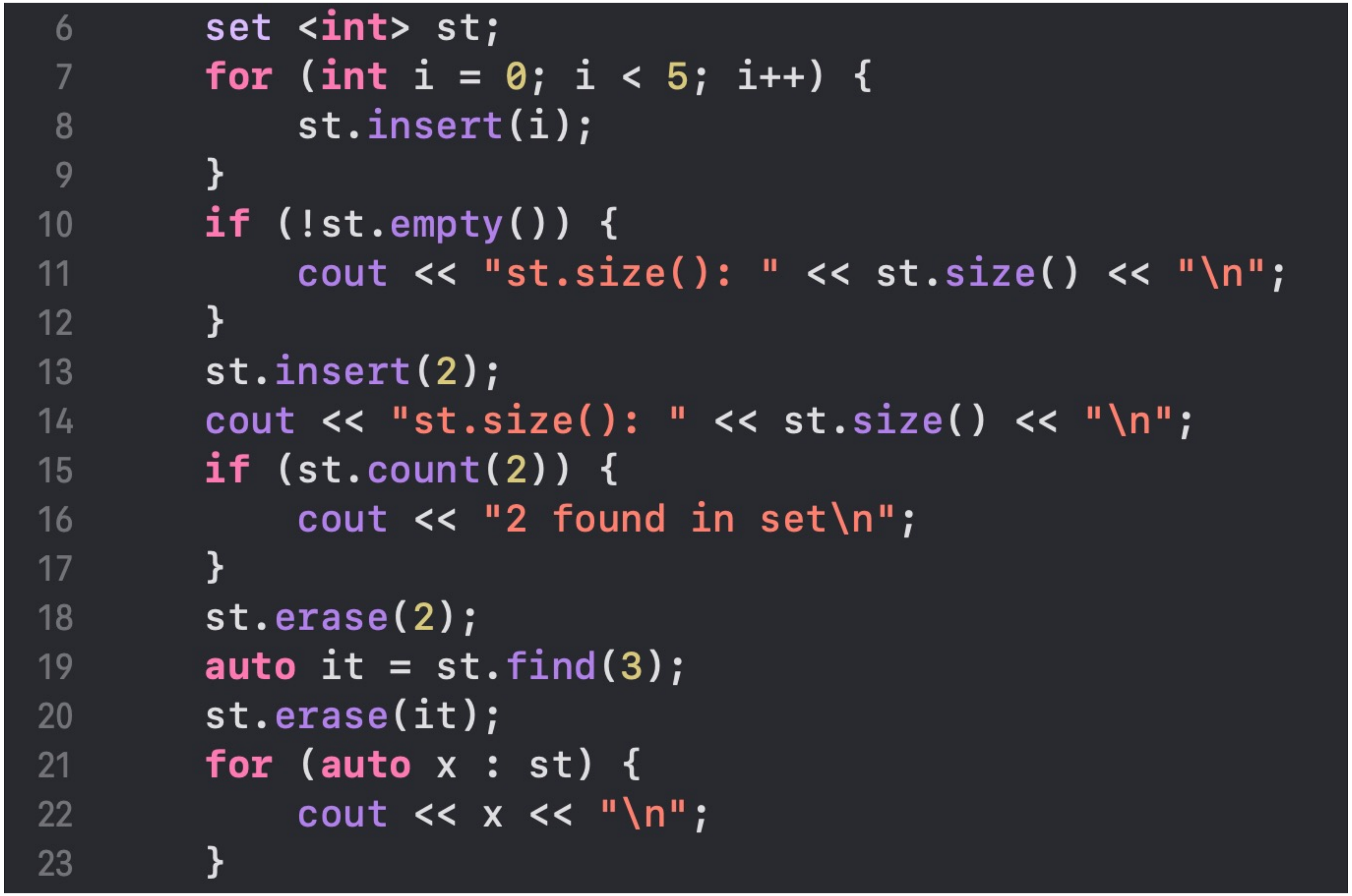
```
    stack <int> stk;
    for (int i = 0; i < 5; i++) {
        stk.push(i);
    }
    cout << "stk.size(): " << stk.size() << "\n";
    while (!stk.empty()) {
        cout << "最上方的元素: " << stk.top() << "\n";
        stk.pop();
    }
```

set

- 去除重複的元素（去重）
- 自動排序（由小到大）

set

- #include <set>
- st.clear()

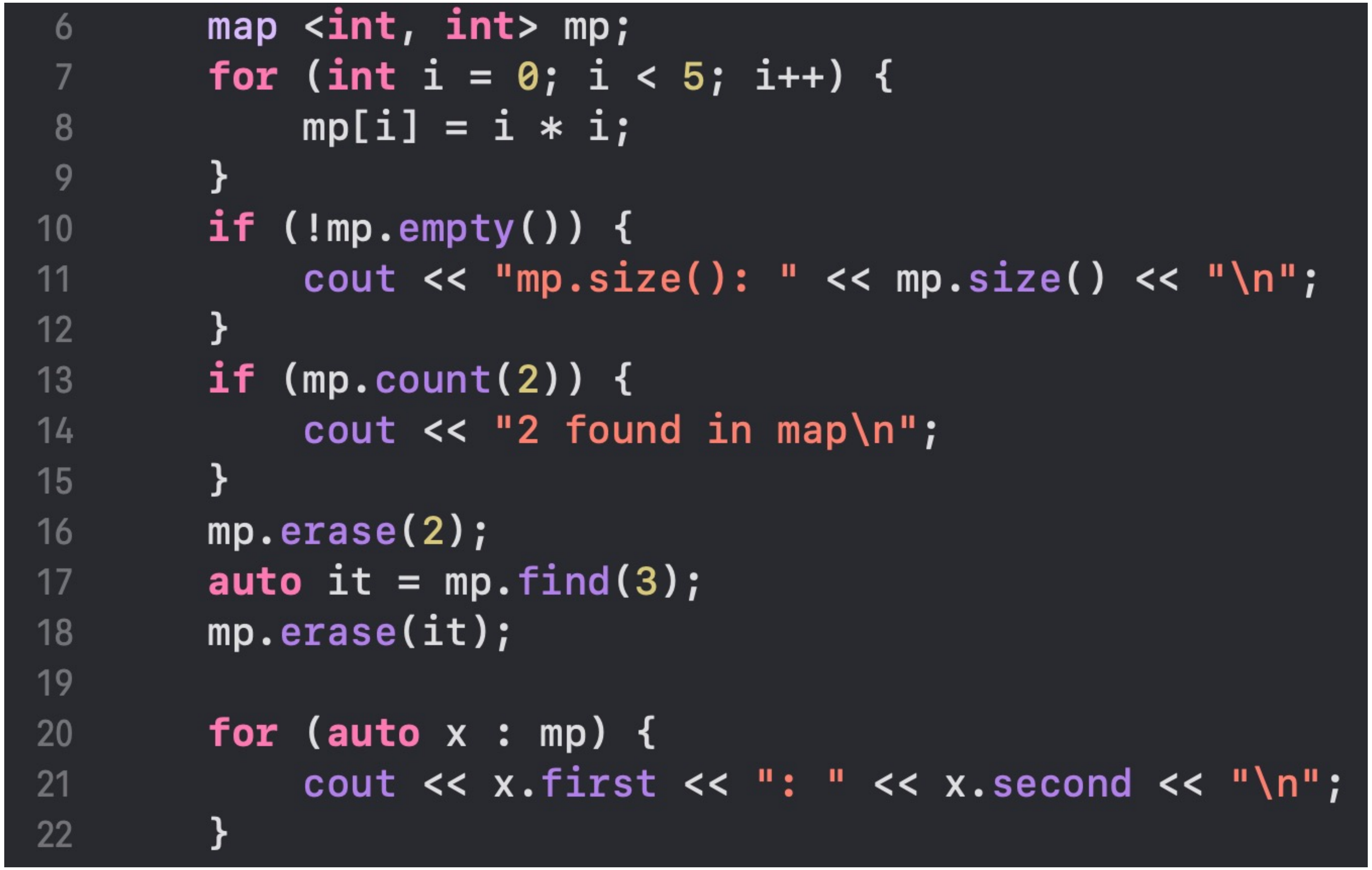


```
    set <int> st;
    for (int i = 0; i < 5; i++) {
        st.insert(i);
    }
    if (!st.empty()) {
        cout << "st.size(): " << st.size() << "\n";
    }
    st.insert(2);
    cout << "st.size(): " << st.size() << "\n";
    if (st.count(2)) {
        cout << "2 found in set\n";
    }
    st.erase(2);
    auto it = st.find(3);
    st.erase(it);
    for (auto x : st) {
        cout << x << "\n";
    }
```

map

- key-value map 速查
- key不重複
- key自動排序（由小到大）

- #include <map>
- mp.clear()



```
    map <int, int> mp;
    for (int i = 0; i < 5; i++) {
        mp[i] = i * i;
    }
    if (!mp.empty()) {
        cout << "mp.size(): " << mp.size() << "\n";
    }
    if (mp.count(2)) {
        cout << "2 found in map\n";
    }
    mp.erase(2);
    auto it = mp.find(3);
    mp.erase(it);

    for (auto x : mp) {
        cout << x.first << ": " << x.second << "\n";
    }
```

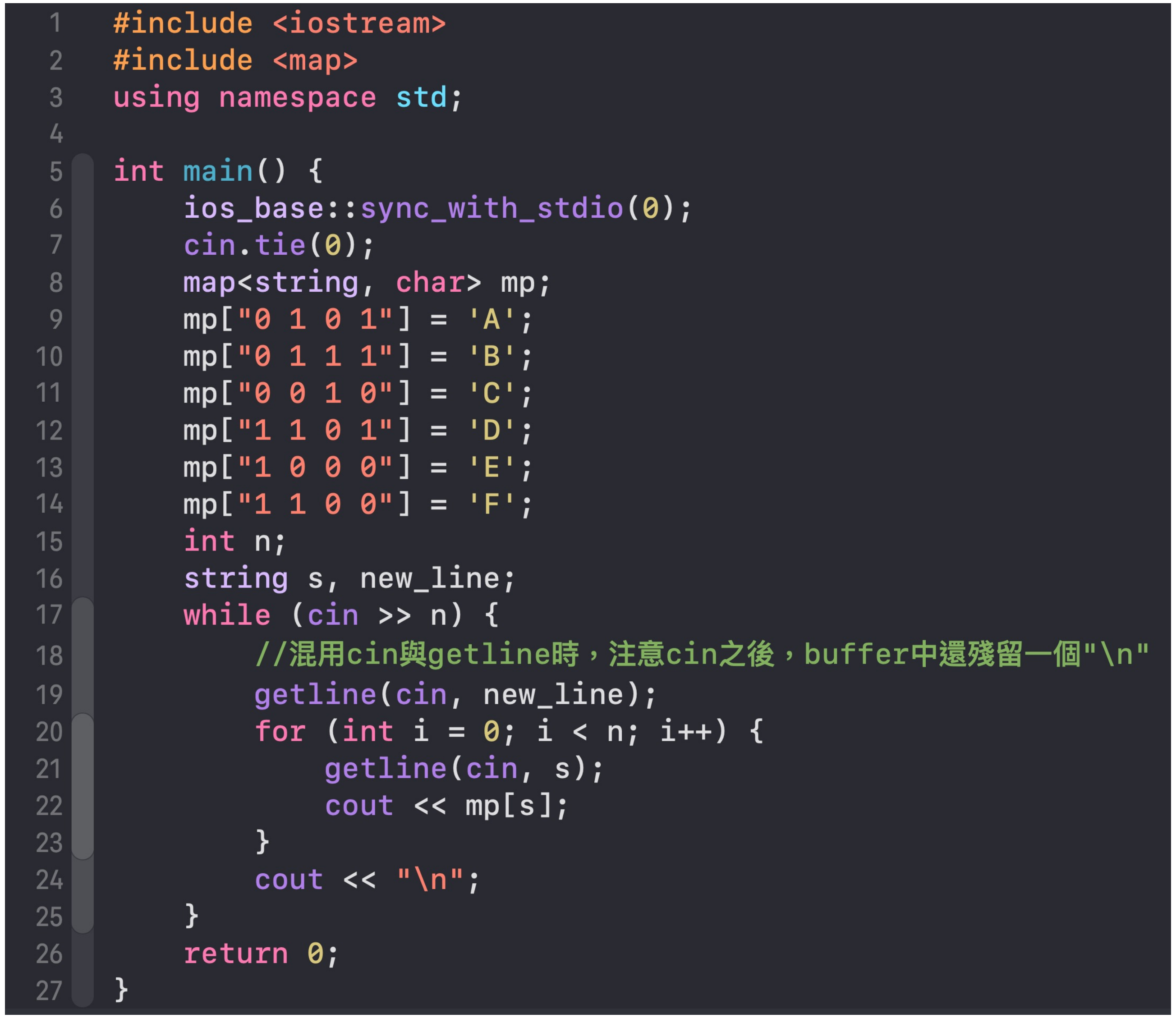
map的初始化

- 宣告同時初始化

```
map <char, int> mp = {{'A', 0}, {'U', 1}, {'C', 2}, {'G', 3}};
```

- 比較常用的場景：在程式執行的過程中，再根據需求添加元素進 map

【範例】e283: APCS 類似題 - 小歲的特殊編碼



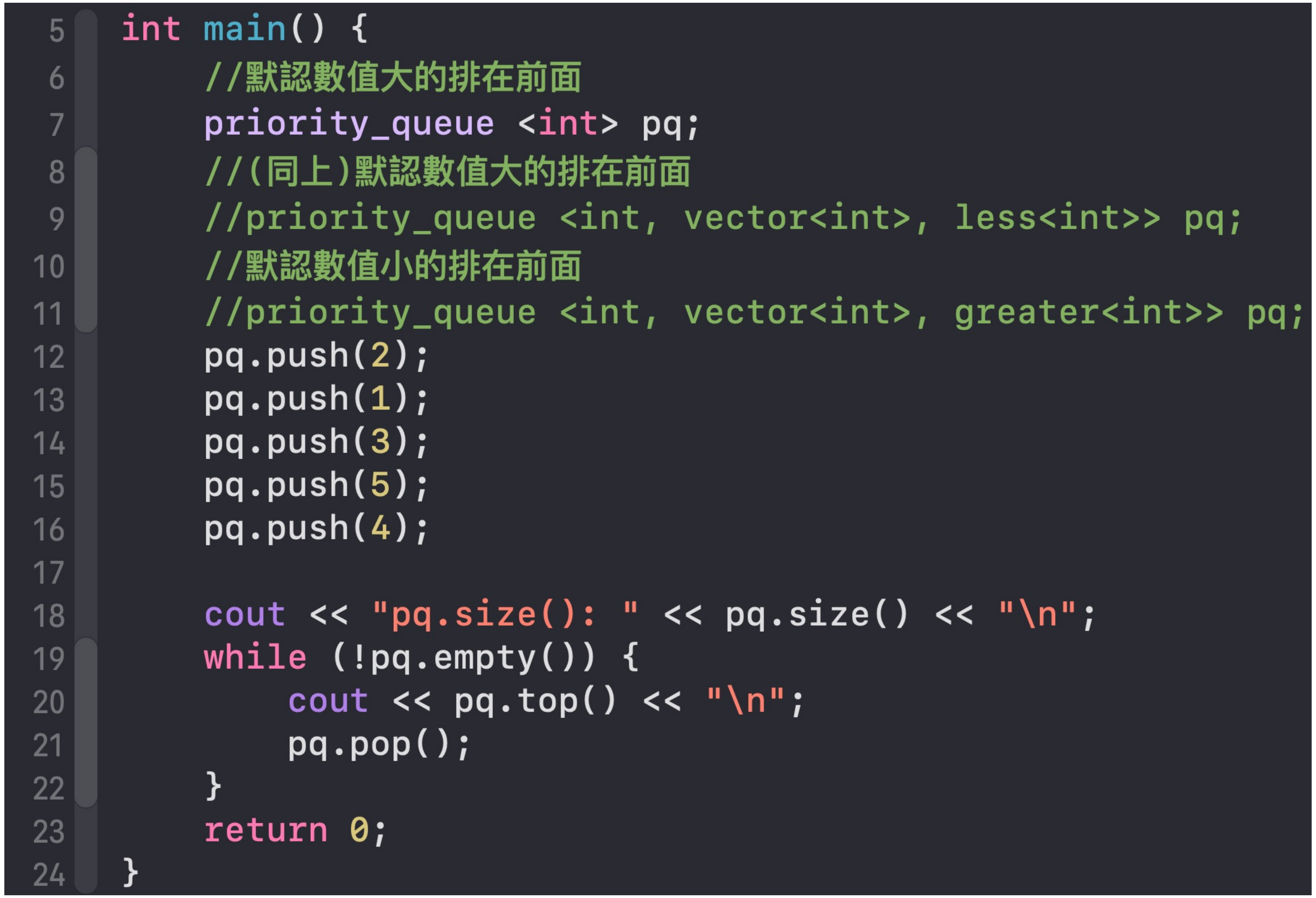
```cpp
#include <iostream>
#include <map>
using namespace std;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    map<string, char> mp;
    mp["0 1 0 1"] = 'A';
    mp["0 1 1 1"] = 'B';
    mp["0 0 1 0"] = 'C';
    mp["1 1 0 1"] = 'D';
    mp["1 0 0 0"] = 'E';
    mp["1 1 0 0"] = 'F';
    int n;
    string s, new_line;
    while (cin >> n) {
        //混用cin與getline時，注意cin之後，buffer中還殘留一個"\n"
        getline(cin, new_line);
        for (int i = 0; i < n; i++) {
            getline(cin, s);
            cout << mp[s];
        }
        cout << "\n";
    }
    return 0;
}
```

priority_queue

可以自定義數據的優先級, 讓優先級高的排在queue的前面

priority_queue

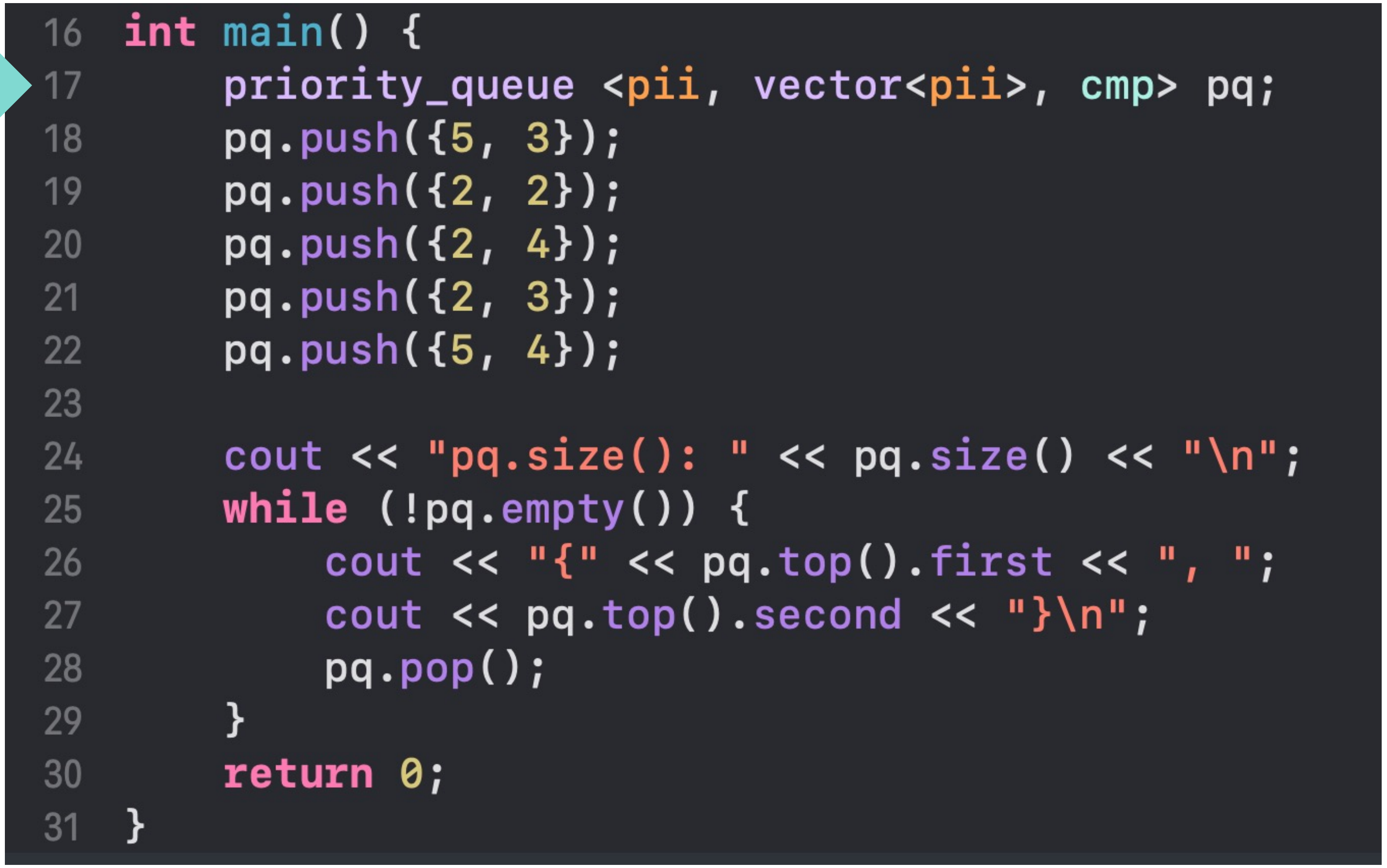
- #include <queue>
- 沒有 .clear()



```
int main() {
    //默認數值大的排在前面
    priority_queue <int> pq;
    //(同上)默認數值大的排在前面
    //priority_queue <int, vector<int>, less<int>> pq;
    //默認數值小的排在前面
    //priority_queue <int, vector<int>, greater<int>> pq;
    pq.push(2);
    pq.push(1);
    pq.push(3);
    pq.push(5);
    pq.push(4);

    cout << "pq.size(): " << pq.size() << "\n";
    while (!pq.empty()) {
        cout << pq.top() << "\n";
        pq.pop();
    }
    return 0;
}
```

改變 priority_queue 排序的方式 (1)

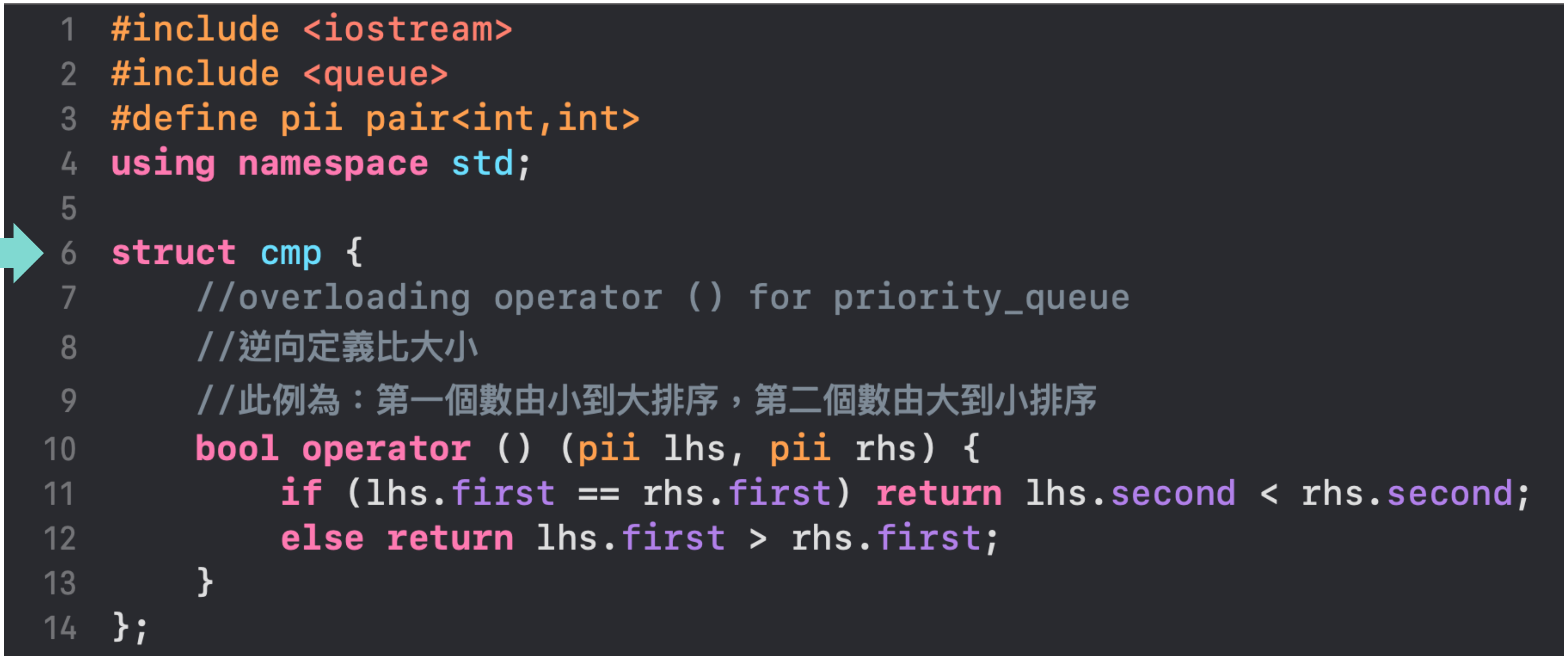
cmp is a struct
(defined on the next page)



```cpp
int main() {
    priority_queue <pii, vector<pii>, cmp> pq;
    pq.push({5, 3});
    pq.push({2, 2});
    pq.push({2, 4});
    pq.push({2, 3});
    pq.push({5, 4});

    cout << "pq.size(): " << pq.size() << "\n";
    while (!pq.empty()) {
        cout << "{" << pq.top().first << ", ";
        cout << pq.top().second << "}\n";
        pq.pop();
    }
    return 0;
}
```

改變 priority_queue 排序的方式 (2)

Example 11-11



```cpp
#include <iostream>
#include <queue>
#define pii pair<int,int>
using namespace std;

struct cmp {
    //overloading operator () for priority_queue
    //逆向定義比大小
    //此例為：第一個數由小到大排序，第二個數由大到小排序
    bool operator () (pii lhs, pii rhs) {
        if (lhs.first == rhs.first) return lhs.second < rhs.second;
        else return lhs.first > rhs.first;
    }
};
```

STL algorithm

https://www.cplusplus.com/reference/algorithm/

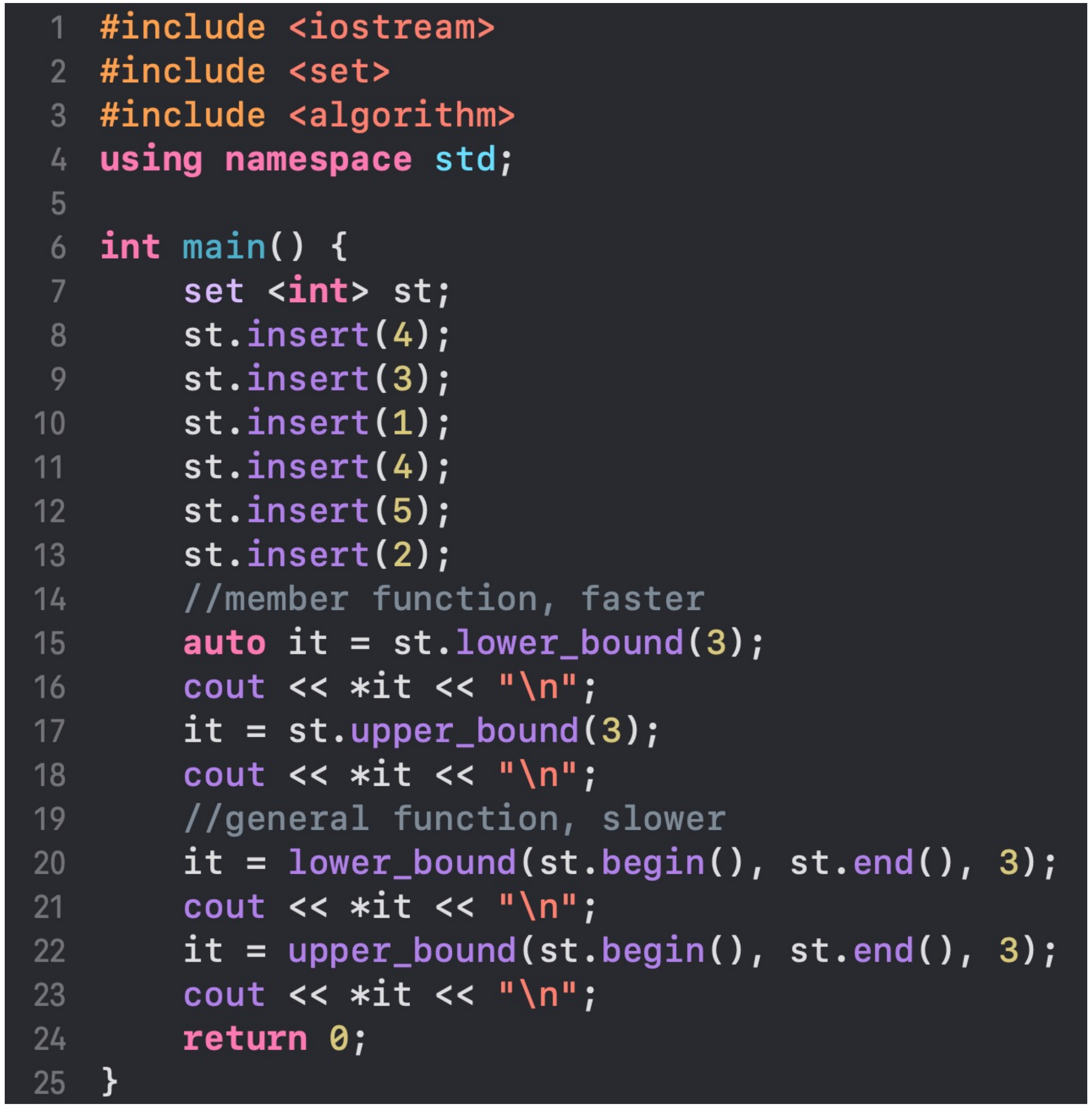
#include <algorithm>

- sort
- reverse
- lower_bound & upper_bound
- next_permutation & prev_permutation
- min_element & max_element

https://www.cplusplus.com/reference/algorithm/

二分搜

- .lower_bound()
- .upper_bound()

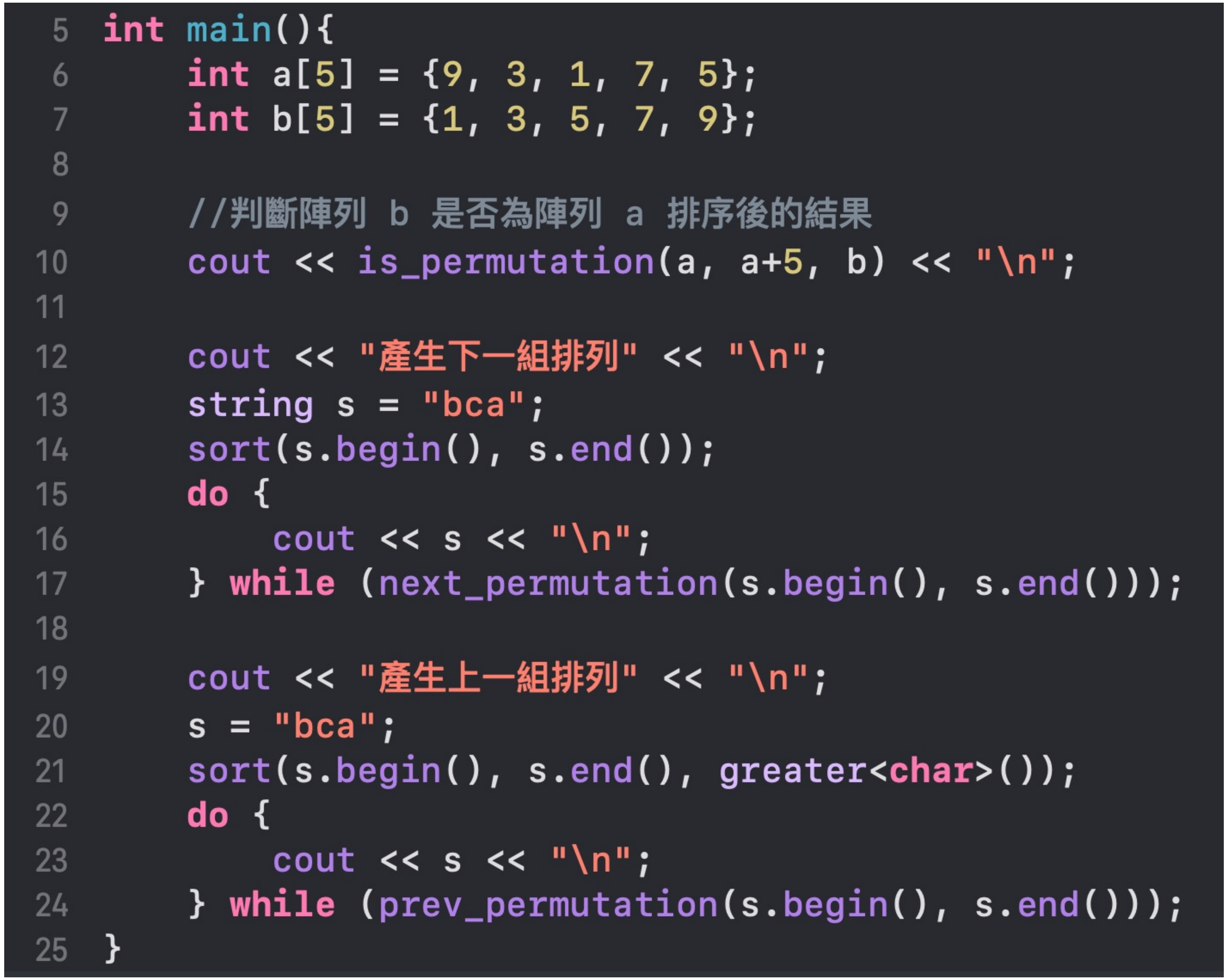


```cpp
#include <iostream>
#include <set>
#include <algorithm>
using namespace std;

int main() {
    set <int> st;
    st.insert(4);
    st.insert(3);
    st.insert(1);
    st.insert(4);
    st.insert(5);
    st.insert(2);
    //member function, faster
    auto it = st.lower_bound(3);
    cout << *it << "\n";
    it = st.upper_bound(3);
    cout << *it << "\n";
    //general function, slower
    it = lower_bound(st.begin(), st.end(), 3);
    cout << *it << "\n";
    it = upper_bound(st.begin(), st.end(), 3);
    cout << *it << "\n";
    return 0;
}
```

permutation

- is_permutation()
- next_permutation()
- prev_permutation()

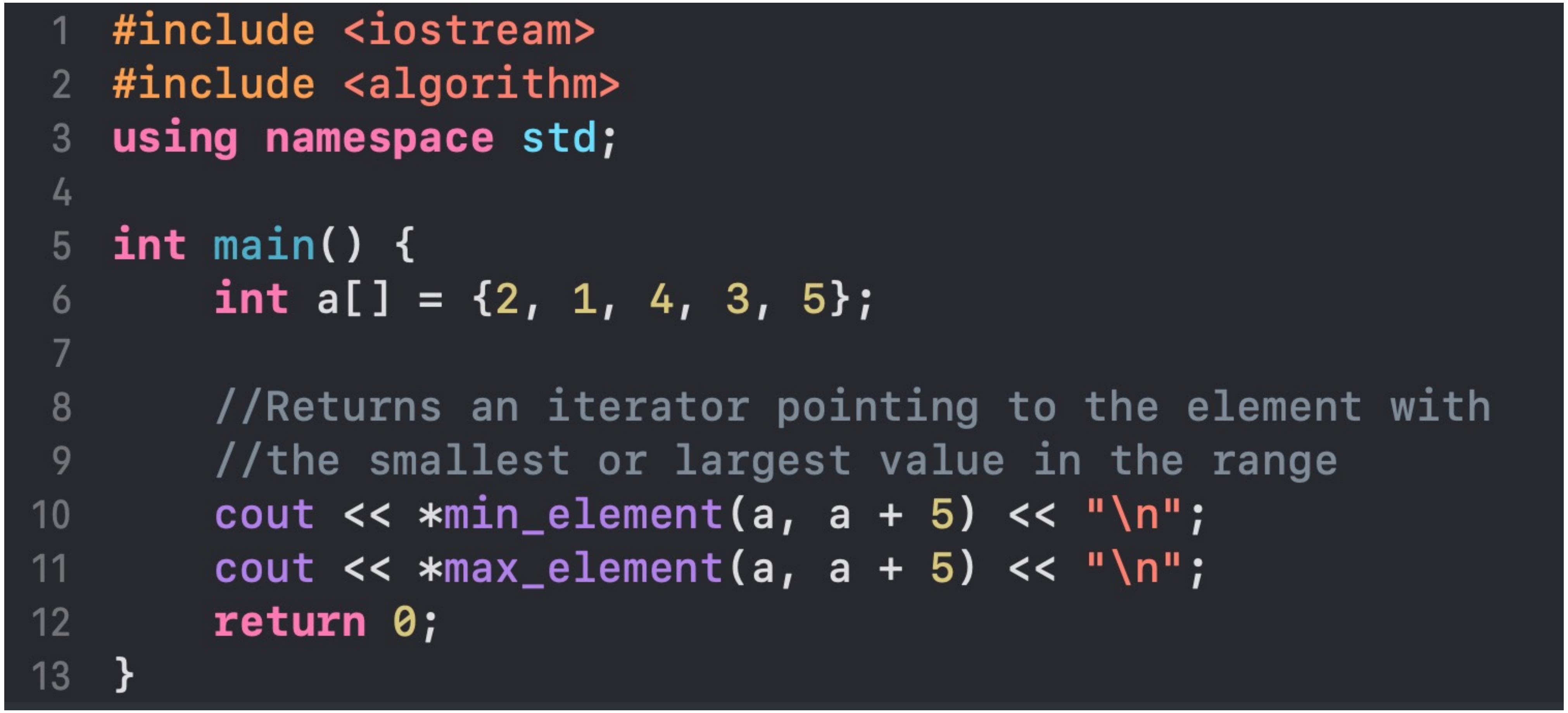
- 【練習】e446: 排列生成



```
int main(){
    int a[5] = {9, 3, 1, 7, 5};
    int b[5] = {1, 3, 5, 7, 9};

    //判斷陣列 b 是否為陣列 a 排序後的結果
    cout << is_permutation(a, a+5, b) << "\n";

    cout << "產生下一組排列" << "\n";
    string s = "bca";
    sort(s.begin(), s.end());
    do {
        cout << s << "\n";
    } while (next_permutation(s.begin(), s.end()));

    cout << "產生上一組排列" << "\n";
    s = "bca";
    sort(s.begin(), s.end(), greater<char>());
    do {
        cout << s << "\n";
    } while (prev_permutation(s.begin(), s.end()));
}
```

min_element() / max_element

Example 11-16



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

int main() {
    int a[] = {2, 1, 4, 3, 5};

    //Returns an iterator pointing to the element with
    //the smallest or largest value in the range
    cout << *min_element(a, a + 5) << "\n";
    cout << *max_element(a, a + 5) << "\n";
    return 0;
}
```