NGÔN NGỮ LẬP TRÌNH C++

SLIDE 5: Set, Map in STL

Giảng viên: Th.S Bùi Văn Kiên

NỘI DUNG

- 1. Lớp Container
- 2. Lớp Set
- 3. Lớp Map

Ví dụ về Set:

- Cho mảng A[] gồm N ($1 \le N \le 10^6$) phần tử,

Có Q ($Q \le 10^6$) truy vấn mỗi truy vấn yêu cầu bạn kiểm tra xem giá trị X có xuất hiện trong mảng hay không?

- Giới hạn:
  - $A[i] \le 10^6$
  - $A[i] \le 10^9$

1. Lớp Container

- Lớp Container bao gồm nhiều lớp cơ bản của C++:
  - lớp Vector
  - lớp danh sách (List)
  - lớp Stack và Queue
  - lớp tập hợp (Set)
  - lớp ánh xạ (Map).

- Container quản lý không gian lưu trữ cho các phần tử của nó và cung cấp các hàm thành viên để truy cập chúng, trực tiếp hoặc thông qua các trình lặp (tham chiếu các đối tượng có thuộc tính tương tự như con trỏ).

1. Lớp Container

Tùy thuộc vào mỗi kiểu container, chúng sẽ có một vài phương thức và toán tử phổ biến như sau:

- ==: Toán tử so sánh bằng
- <: Toán tử so sánh nhỏ hơn
- begin(): Giá trị khởi đầu của con chạy iterator
- end(): Giá trị kết thúc của con chạy iterator
- size(): Số lượng phần tử đối tượng của container
- empty(): kiểm tra container rỗng

1. Lớp Container

Phương thức và toán tử phổ biến

- front() →Phần tử thứ nhất của container
- back() →Phần tử cuối cùng của container
- [] →Toán tử truy nhập đến phần tử của container
- insert() →Thêm vào container một (hoặc một số) phần tử
- push_back()→ Thêm một phần tử vào cuối container
- push_front()→ Thêm một phần tử vào đầu container
- erase() → Loại bỏ một phần tử khỏi container
- pop_back() → Loại bỏ phần tử cuối của container
- pop_front() → Loại bỏ phần tử đầu của container

1. Lớp Container

Con chạy Iterator

- Cú pháp: Tên_lớp<T>::iterator Tên_con_chạy;

Trong đó:

  - Tên lớp: là tên của lớp cơ bản ta đang dùng, ví dụ lớp Set, lớp List…
  - T: là tên kiểu lớp của các phần tử chứa trong container.
  - Tên con chạy: là tên biến sẽ được sử dụng làm biến chạy
- Ví dụ:
  - Set<int>::iterator iter;
  - Hoặc: List<Person>::iterator iter;

1. Lớp Container

Con chạy Iterator

- Con chạy được sử dụng khi cần duyệt lần lượt các phần tử có mặt trong container.

Ví dụ:

```
set<int> mySet;              // Khai báo một đối tượng của lớp Set,
set<int>::iterator i;        // Khai báo con chạy của lớp Set,
mySet.insert(1);
mySet.insert(2);
mySet.insert(3);
for(i=mySet.begin(); i<mySet.end(); i++)
        cout << mySet[i] << " "
```

2. Lớp tập hợp Set

- Trong C++, set là một cấu trúc dữ liệu thuộc thư viện STL (Standard Template Library) xây dựng dựa trên cây đỏ đen (Red black tree).
- Std::set cho phép lưu trữ các phần tử không trùng lặp và được sắp xếp tự động theo thứ tự tăng dần (mặc định). Std::set hỗ trợ các thao tác như thêm, tìm kiếm, xóa phần tử và có hiệu suất khá cao trong việc xử lý các tập hợp dữ liệu.

2. Lớp tập hợp Set

- Thư viện: #include <set>
- Khởi tạo không tham số:

set<T> Tên_đối_tượng;

- Thao tác:
  - mySet.insert()
  - mySet.erase()
  - mySet.size()
  - mySet.empty()
  - mySet.clear()
  - mySet.lower_bound()
  - mySet.upper_bound()

```
#include <set>
#include <iostream>

int main() {
    std::set<int> mySet; // khai báo một set chứa các số nguyên
    mySet.insert(1);     // thêm phần tử 1
    mySet.insert(2);     // thêm phần tử 2
    mySet.insert(1);     // không thêm được vì 1 đã tồn tại

    for (int element : mySet) {
        std::cout << element << " "; // sẽ in ra: 1 2
    }
    return 0;
}
```

2. Lớp tập hợp Set

- Duyệt phần tử trong Set

Sử dụng range-based for loop

```cpp
#include <set>
#include <iostream>

int main() {
    std::set<int> mySet = {1, 2, 3, 4, 5};

    for (int element : mySet) {
        std::cout << element << " "; // In ra: 1 2 3 4 5
    }
    return 0;
}
```

Sử dụng iterator

```cpp
int main() {
    std::set<int> mySet = {1, 2, 3, 4, 5};

    for (std::set<int>::iterator it = mySet.begin(); it != mySet.end(); ++it) {
        std::cout << *it << " "; // In ra: 1 2 3 4 5
    }
    return 0;
}
```

2. Lớp tập hợp Set

- multiset<int> S

Cho phép lưu trữ các phần tử có giá trị bằng nhau, tự duy trì tính sắp xếp trong tập hợp.

```cpp
#include <iostream>
#include <set>

int main() {
    std::multiset<int> myMultiset = {3, 1, 4, 1, 5, 9, 2, 6, 5};

    // In các phần tử của multiset
    for (int element : myMultiset) {
        std::cout << element << " "; // In ra: 1 1 2 3 4 5 5 6 9
    }
    return 0;
}
```

2. Lớp tập hợp Set

- Xóa phần tử: S.erase(value)

Xóa theo value

```
#include <set>
#include <iostream>

int main() {
    std::set<int> mySet = {1, 2, 3, 4, 5};

    mySet.erase(3); // Xóa phần tử có giá trị là 3

    for (int element : mySet) {
        std::cout << element << " "; // In ra: 1 2 4 5
    }
    return 0;
}
```

Tìm kiếm và xóa theo iterator

```
#include <set>
#include <iostream>

int main() {
    std::set<int> mySet = {1, 2, 3, 4, 5};
    auto it = mySet.find(4); // Tìm iterator của phần tử 4

    if (it != mySet.end()) {
        mySet.erase(it); // Xóa phần tử tại iterator `it`
    }

    for (int element : mySet) {
        std::cout << element << " "; // In ra: 1 2 3 5
    }
    return 0;
}
```

2. Lớp tập hợp Set

- Sử dụng set với struct: cần khai báo toán tử < giữa 2 phần tử của struct.

```
Struct phanso{
        int a, b;
        friend bool operator < (phanso A, phansoB) {

        }
}
```

Ví dụ về Map

- Nhập N xâu đầu vào. Yêu cầu in ra xâu có tần số xuất hiện nhiều nhất?
- Input

AAA

AAA

BBB

CCC

ABC

AAA

- Output = AAA

3. Lớp ánh xạ Map

- Map cũng là 1 CTDL của thư viện STL được xây dựng dựa trên cây đỏ đen (một loại cây tìm kiếm nhị phân tự cân bằng).
- Mỗi phần tử của Map là ánh xạ giữa yếu tố key (khóa) với giá trị (value) của nó.
- Map không chứa hai phần tử nào giống nhau và các phần tử trong map cũng được sắp xếp theo một thứ tự nào đó giống như Set.
- Key và value có thể có kiểu khác nhau.

- Ví dụ:
  - map<int, int> a;
  - map<char, int> b;

3. Lớp ánh xạ Map

- Ứng dụng của Map:
  - Map có thể hoàn toàn thay thế cho Set (gán value = 1)
  - Sử dụng Map là mảng đếm (counting) khi giá trị số lớn (10^9) hay phần tử là kiểu dạng string

- Có thể thay đổi thứ tự sắp xếp các phần tử trong map bằng cách sử dụng con trỏ hàm.

```
struct cmp{
    bool operator() (char a, char b) {return a>b;}
};
// khai báo thêm biến cmp
map<char, int, cmp> m;
```

3. Lớp ánh xạ Map

Các phương thức:

- Thêm phần tử
- Kiểm tra sự tồn tại của phần tử
- Truy cập phần tử: [] hoặc Map.at

```cpp
#include <iostream>
#include <map>
#include <string>

int main() {
    std::map<std::string, int> myMap = {{"Alice", 25}, {"Bob", 30}, {"Charlie", 35}};

    // Thêm phần tử
    myMap["David"] = 40;

    // Cập nhật giá trị của "Alice"
    myMap["Alice"] = 26;

    // Kiểm tra sự tồn tại của một key
    if (myMap.find("Bob") != myMap.end()) {
        std::cout << "Bob tồn tại trong map" << std::endl;
    }
```

3. Lớp ánh xạ Map

- Xóa phần tử
- Duyệt các phần tử
  - Sử dụng con chạy iterator
  - Sử dụng (pair<int, int> x : myMap)
  - Sử dụng (auto x : myMap)

```
// Xóa một phần tử
myMap.erase("Charlie");

// In map sau khi xóa
for (const auto& pair : myMap) {
    std::cout << pair.first << ": " << pair.second << std::endl;
}
```

Alice: 26

Bob: 30

David: 40

3. Lớp ánh xạ Map

- Nhập N xâu đầu vào. Yêu cầu in ra xâu có tần số xuất hiện nhiều nhất?
- Input

AAA

AAA

BBB

CCC

ABC

AAA

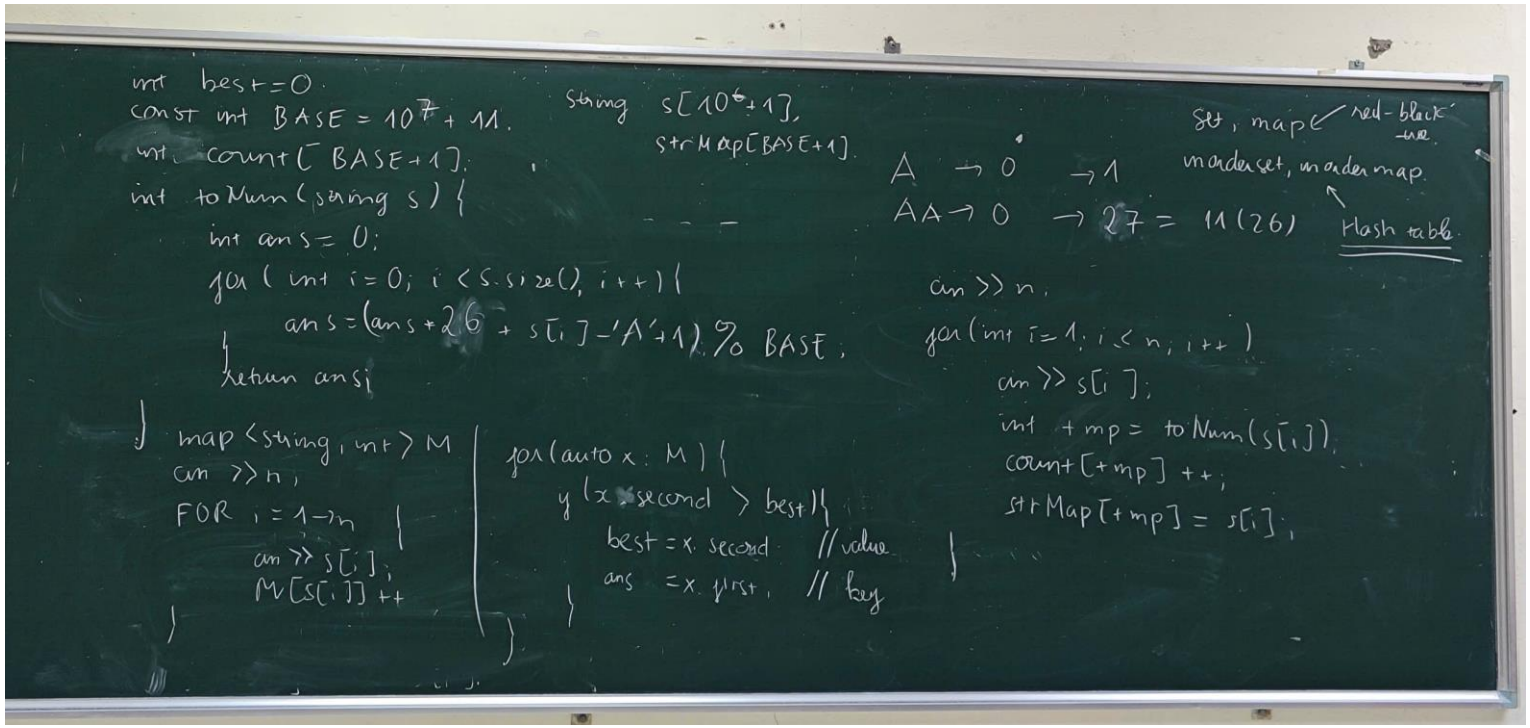
- Output = AAA

```
int best = 0;
string ans = “”;
map<string, int> M;
cin >> n;
for(int i = 1; i <= n; i++) {
        cin >> s[i];
        M[s[i]]++;
}
for(auto x : M) {
        if(x.second > best) {
                best = x.second;
                ans = x.first;
        }
}
```

3. Lớp ánh xạ Map

- unordered_set, unordered_map: có các phương thức tương tự set, map, nhưng được xây dựng trên HashTable

(vai trò tương tự mảng count[] ở dưới đây)



QUESTIONS & ANSWERS