

MAP

```
#include <map>
```

Map

Khái niệm

Map là container giúp lưu các phần tử theo **cặp key, value** (khóa - giá trị). **Mỗi giá trị của key sẽ ánh xạ sang một value tương ứng**. So với Set thì Map thậm chí còn mạnh mẽ và giải quyết được nhiều vấn đề hơn.

Tính chất



Các **key** trong map là **những giá trị riêng biệt**, không có 2 key nào có giá trị giống nhau, **value** thì có thể trùng nhau.



Các **cặp phần tử** trong map được sắp xếp theo **thứ tự tăng dần của key**.



Mỗi phần tử trong map thực chất là **một pair**, với first lưu key và second lưu value.

Map

CÚ PHÁP

```
map <key_data_type, value_data_type> map_name;
```

SỬ DỤNG MAP

- /01** Các bài toán liên quan tới tần suất của các phần tử.
- /02** Các bài toán cần tìm kiếm, thêm, xóa một cách nhanh chóng.
- /03** Dùng map thay cho các bài toán sử dụng mảng đánh dấu khi dữ liệu không đẹp.



Một số hàm trong map

Thêm một phần tử vào trong map:

- Dùng **hàm insert**
- Dùng cú pháp **map[key] = value** nếu key chưa tồn tại trong map, hoặc sẽ thay đổi value nếu key đã tồn tại.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    map<int, int> mp;
    mp.insert({1, 2}); //Thêm cặp (1,2)
    mp.insert({2, 4}); //Thêm cặp (2,4)
    mp.insert({1, 3}); //Không thêm được cặp (1,3)
    mp[3] = 10; // thêm cặp (3,10)
    mp[2] = 5; //Thay đổi cặp (2,4) thành (2,5)
}
```

● mp = {(1,2), (2,5), (3,10)}

Một số hàm trong map

Hàm size: trả về số lượng phần tử trong map.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    map<int, int> mp;
    mp.insert({1, 2}); //Thêm cặp (1, 2)
    mp.insert({2, 4}); //Thêm cặp (2, 4)
    mp.insert({1, 3}); //Thêm cặp (1, 3)
    //nhưng không thêm được
    mp[3] = 10; // thêm cặp (3, 10)
    mp[2] = 5; //Thay đổi cặp (2, 4)
    //thành (2, 5)
    cout << mp.size() << endl;
}
```

OUTPUT: 3

Một số hàm trong map

Hàm empty: kiểm tra map rỗng, nếu rỗng trả về true, ngược lại trả về false.

Hàm clear: xóa mọi phần tử trong map.



```
#include <bits/stdc++.h>
using namespace std;

int main(){
    map<int, int> mp;
    mp.insert({1, 2}); //Thêm cặp (1, 2)
    mp.insert({2, 4}); //Thêm cặp (2, 4)
    mp.insert({1, 3}); //Thêm cặp (1, 3)
    //nhưng không thêm được
    mp.insert({2, 5});
    mp.insert({3, 6});
    cout << mp.size() << endl;
    if(mp.empty()) cout << "Empty\n";
    else cout << "Not empty\n";
    mp.clear(); // xóa hết mọi phần tử
    if(mp.empty()) cout << "Empty\n";
    else cout << "Not empty\n";
}
```

OUTPUT: 3

Not empty !

Empty !

Một số hàm trong map

Hàm find: Tìm kiếm sự xuất hiện của một key nào đó trong map. Độ phức tạp là $O(\log N)$.

Hàm này trả về iterator tới cặp phần tử nếu nó tìm thấy, ngược lại nó trả về iterator `end()` của map khi giá trị key tìm kiếm không tồn tại trong map. ●●●

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    map<int, int> mp;
    mp.insert({1, 2}); //Thêm cặp (1, 2)
    mp.insert({2, 4}); //Thêm cặp (2, 4)
    mp.insert({3, 6}); //Thêm cặp (3, 6)
    auto it = mp.find(1);
    if(it == mp.end()){
        cout << "NOT FOUND KEY\n";
    }
    else{
        cout << (*it).first << ' ' <<
        (*it).second << endl;
    }
}
```

OUTPUT: 1 2

Một số hàm trong map

Hàm count: Hàm này dùng để đếm số lần xuất hiện của 1 key nào đó trong map. Đối với map hàm count trả về 0 hoặc 1, có thể sử dụng hàm này để thay cho hàm find. Độ phức tạp $O(\log N)$. ●●●

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    map<int, int> mp;
    mp.insert({1, 2}); //Thêm cặp (1, 2)
    mp.insert({2, 4}); //Thêm cặp (2, 4)
    mp.insert({3, 6}); //Thêm cặp (3, 6)
    cout << mp.count(1) << endl;
    cout << mp.count(5) << endl;
}
```

OUTPUT: 1
0

Một số hàm trong map

Hàm erase: Xóa một phần tử khỏi map với độ phức tạp là $O(\log N)$, trước khi sử dụng hàm erase hãy đảm bảo phần tử bạn cần xóa tồn tại trong map nếu không sẽ xảy ra lỗi runtime error. ●●●

Xóa thông qua giá trị của key

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    map<int, int> mp;
    mp.insert({1, 2}); //Thêm cặp (1, 2)
    mp.insert({2, 4}); //Thêm cặp (2, 4)
    mp.insert({3, 6}); //Thêm cặp (3, 6)
    mp.erase(1);
    for(auto it : mp){
        cout << it.first << ' ' <<
it.second << endl;
    }
}
```

OUTPUT: 2 4
3 6

Một số hàm trong map

Hàm erase: Xóa một phần tử khỏi map với độ phức tạp là $O(\log N)$, trước khi sử dụng hàm erase hãy đảm bảo phần tử bạn cần xóa tồn tại trong map nếu không sẽ xảy ra lỗi runtime error. ●●●

Xóa thông qua iterator

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    map<int, int> mp;
    mp.insert({1, 2}); //Thêm cặp (1, 2)
    mp.insert({2, 4}); //Thêm cặp (2, 4)
    mp.insert({3, 6}); //Thêm cặp (3, 6)
    auto it = mp.find(3);
    if(it != mp.end()){
        mp.erase(it);
    }
    for(auto it : mp){
        cout << it.first << ' ' <<
it.second << endl;
    }
}
```

OUTPUT: 1 2
2 4

Duyệt map

Duyệt map

● mp = {(1,2),(2,4),(3,6)}

Duyệt bằng for each:

```
//for each
for(pair<int, int> it : mp){
    cout << it.first << ' ' << it.second << endl;
}
//for each dùng auto thay cho pair
for(auto it : mp){
    cout << it.first << ' ' << it.second << endl;
}
//Từ key suy ra value
for(auto it : mp){
    int key = it.first;
    cout << key << ' ' << mp[key] << endl;
}
```

Duyệt map

Duyệt map

● mp = {(1,2),(2,4),(3,6)}

Duyệt bằng iterator:

```
//Dùng iterator
for(map<int, int>::iterator it = mp.begin(); it !=
mp.end(); ++it){
    cout << (*it).first << ' ' << (*it).second << endl;
}
//Thay bằng auto cho tiện
for(auto it = mp.begin(); it != mp.end(); ++it){
    cout << (*it).first << ' ' << (*it).second << endl;
}
```