MAP, MULTIMAP, UNORDERED_MAP

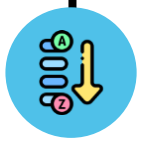
1. MAP:

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.

1. 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.

```cpp
#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.

```cpp
#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.

```cpp
#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

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;
}
```

MỘT SỐ HÀM TRONG 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;
}
```

MỘT SỐ HÀM TRONG MAP

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

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(logN).

```cpp
#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(logN), 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(logN), 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

Dự đoán output của một số chương trình sau

Quiz 1

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

int main(){
    map<int, int> mp;
    mp[1]++; mp[1]++; mp[1]++;
    mp[2] = 3;
    mp[2] = 4;
    for(auto it : mp){
        cout << it.first << ' ' <<
it.second << endl;
    }
}
```

Quiz 2

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

int main(){
    map<int, int> mp;
    cout << mp[1] << endl;
    mp.insert({1, 2});
    mp.insert({1, 3});
    mp.insert({1, 4});
    mp[1] = 5;
    cout << mp[1] << endl;
}
```

Dự đoán output của một số chương trình sau

Quiz 3

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

int main(){
    map<int, int> mp;
    mp.insert({1, 2});
    mp.insert({1, 3});
    mp.insert({1, 4});
    mp[2] = 3; mp[3] = 4; mp[3]++;
    auto it = mp.find(3);
    --it;
    cout << (*it).first << ' ' <<
(*it).second << endl;
}
```

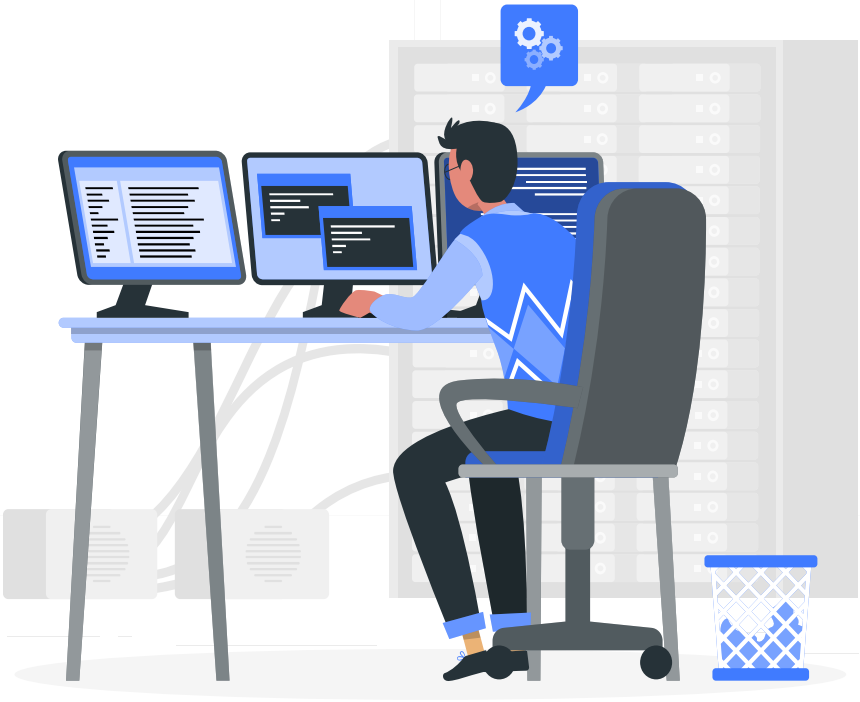
Quiz 4

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

int main(){
    map<int, int> mp;
    int a[] = {1, 2, 3, 4, 1, 2, 3, 4, 5, 5, 1};
    for(int x : a){
        mp[x]++;
    }
    for(auto it : mp){
        cout << it.first << ' ' << it.second <<
endl;
    }
}
```

2. MULTIMAP:

Multimap tương tự như map có điều khác biệt là trong multimap có thể tồn tại nhiều key có cùng giá trị. Các tính chất như key được sắp xếp tăng dần hay độ phức tạp, cách sử dụng hàm thì tương tự như map. Có 3 hàm có sự khác biệt so với map là : find, count, erase.



Chú ý: Trong multimap bạn không thể truy cập value thông qua key, hoặc gán theo cú pháp map[key] = value.

MỘT SỐ HÀM TRONG MULTIMAP

Hàm find: Vì trong multimap có nhiều key có cùng giá trị nên nếu ta tìm kiếm giá trị của 1 key nào đó nó sẽ trả về vị trí xuất hiện đầu tiên của key đó trong multimap.

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

int main(){
    multimap<int, int> mp;
    mp.insert({1, 2});
    mp.insert({1, 3});
    mp.insert({1, 4});
    mp.insert({2, 5});
    auto it = mp.find(1);
    cout << (*it).first << ' ' <<
(*it).second << endl;
}
```

OUTPUT: 1 2

MỘT SỐ HÀM TRONG MULTIMAP

Hàm count

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

int main(){
    multimap<int, int> mp;
    mp.insert({1, 2});
    mp.insert({1, 3});
    mp.insert({1, 4});
    mp.insert({2, 5});
    cout << mp.count(1) << endl;
    cout << mp.count(2) << endl;
}
```

OUTPUT:
```
3
1
```

MỘT SỐ HÀM TRONG MULTIMAP

Hàm erase: Xóa thông qua giá trị sẽ xóa tất cả các key tương ứng.

Xóa bằng giá trị

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

int main(){
    multimap<int, int> mp;
    mp.insert({1, 2});
    mp.insert({1, 3});
    mp.insert({1, 4});
    mp.insert({2, 5});
    mp.insert({2, 6});
    mp.erase(1);
    for(auto it : mp){
        cout << it.first << ' '
<< it.second << endl;
    }
}
```

OUTPUT:
```
2 5
2 6
```

MỘT SỐ HÀM TRONG MULTIMAP

Hàm erase: Xóa thông qua iterator sẽ xóa phần tử mà iterator đó chỉ vào.

Xóa bằng iterator

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

int main(){
    multimap<int, int> mp;
    mp.insert({1, 2});
    mp.insert({1, 3});
    mp.insert({1, 4});
    mp.insert({2, 5});
    mp.insert({2, 6});
    auto it = mp.find(1);
    mp.erase(it);
    for(auto it : mp){
        cout << it.first << ' '
<< it.second << endl;
    }
}
```

OUTPUT:

```
1 3
1 4
2 5
2 6
```

3. UNORDERED_MAP

Unordered_map cũng tương tự như map nhưng các key trong unordered_map không có thứ tự như map và multimap.

Ví dụ:

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

int main(){
    unordered_map<int, int> mp;
    mp.insert({1, 4});
    mp.insert({1, 2});
    mp.insert({1, 3});
    mp.insert({2, 5});
    mp.insert({2, 6});
    mp.insert({3, 5});
    for(auto it : mp){
        cout << it.first << ' '
<< it.second << endl;
    }
}
```

OUTPUT:
```
3 5
2 5
1 4
```

3. UNORDERED_MAP

Sự khác biệt

Unordered_map khác với map và multimap ở tốc độ của các hàm phổ biến như count, find, và erase. Còn cách sử dụng thì không có gì khác biệt so với map.

- Ở map và multimap các hàm có độ phức tạp là O(logN).
- Ở unordered_map tốc độ của các hàm trong trường hợp tốt nhất là O(1) còn tệ nhất có thể lên đến O(N).