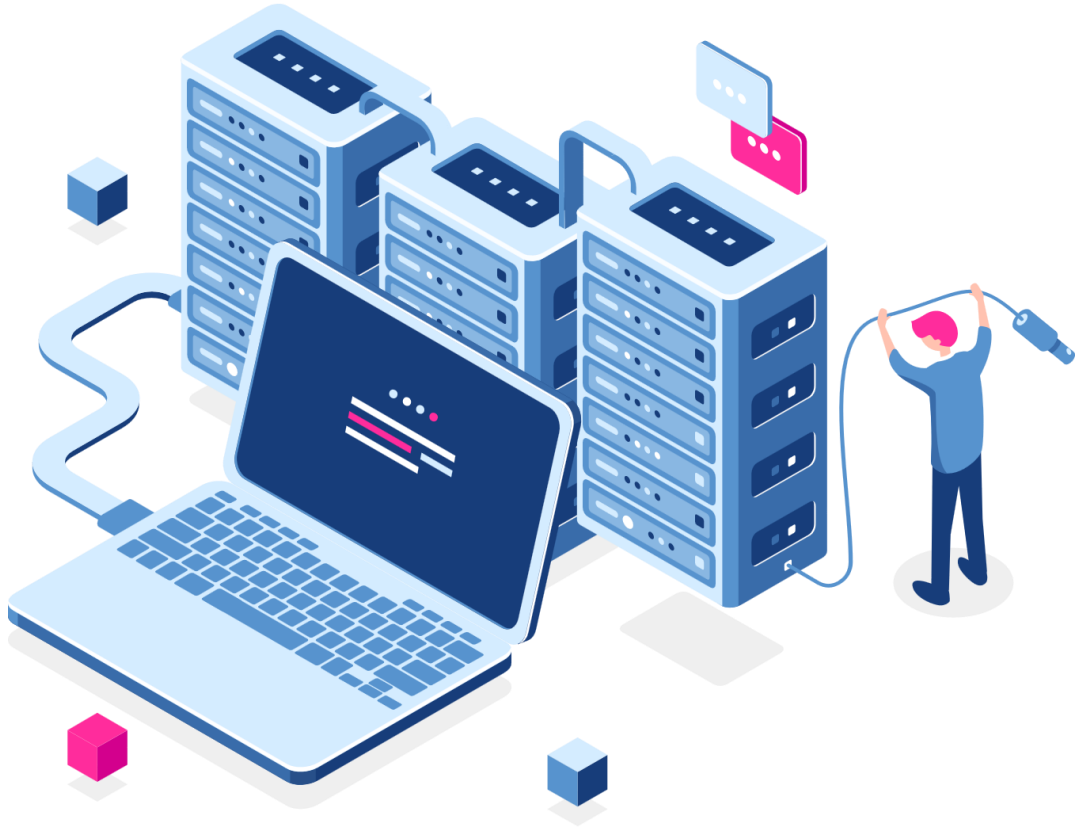
SET, MULTISET, UNORDERED_SET



1. SET:

Khái niệm

Set là một container cực kì mạnh mẽ trong thư viện STL của ngôn ngữ lập trình C++, sử dụng thành thạo Set là một kỹ năng cơ bản mà bạn cần đạt được. Set sẽ giúp code của các bạn trở nên tối ưu và ngắn gọn hơn rất nhiều.

Tính chất

Set là một container mà mỗi phần tử trong đó là duy nhất, tức là sẽ không có 2 phần tử có giá trị giống nhau tồn tại trong set.

Các phần tử trong set được sắp xếp theo thứ tự tăng dần về giá trị số và tăng dần về thứ tự từ điển nếu là xâu kí tự.

1. SET:

CÚ PHÁP

```
set <data_type> set_name;
```

SỬ DỤNG SET

- Các bài toán liên quan tới việc xóa, thêm, tìm kiếm một phần tử nào đó được thực hiện đi thực hiện lại nhiều lần.
- Các bài toán liên quan tới các giá trị khác nhau của mảng

MỘT SỐ HÀM TRONG SET

Hàm insert: thêm một phần tử trong set.

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

int main(){
    set<int> se;
    int a[7] = {1, 1, 2, 1, 3, 4, 5};
    for(int i = 0; i < 7; i++){
        se.insert(a[i]);
    }
}
```

```
se = {1, 2, 3, 4, 5}
```

MỘT SỐ HÀM TRONG SET

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

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

int main(){
    set<int> se;
    int a[7] = {1, 1, 2, 1, 3, 4, 5};
    for(int i = 0; i < 7; i++){
        se.insert(a[i]);
    }
    cout << se.size() << endl;
}
```

OUTPUT: 5

MỘT SỐ HÀM TRONG SET

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

Hàm clear: xóa mọi phần tử trong set.

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

int main(){
    set<int> se;
    int a[7] = {1, 1, 2, 1, 3, 4, 5};
    for(int i = 0; i < 7; i++)
        se.insert(a[i]);
    cout << se.size() << endl;
    if(se.empty())
        cout << "Empty !\n";
    else
        cout << "Not empty !\n";
    se.clear(); // set rỗng
    if(se.empty())
        cout << "Empty !\n";
    else
        cout << "Not empty !\n";
}
```

OUTPUT: 5
Not empty !
Empty !

MỘT SỐ HÀM TRONG SET

Duyệt set

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

int main(){
    set<int> se;
    int a[7] = {1, 1, 2, 1, 3, 4, 5};
    for(int i = 0; i < 7; i++)
        se.insert(a[i]);
    //Cách 1: Dùng For each
    for(int x : se)
        cout << x << ' ';
    cout << endl;
    //Cách 2: Dùng Iterator
    for(set<int>::iterator it = se.begin(); it != se.end(); ++it)
        cout << *it << ' ';
    cout << endl;
    //Cách 3: Dùng Auto
    for(auto it = se.begin(); it != se.end(); ++it)
        cout << *it << ' ';
}
```

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

MỘT SỐ HÀM TRONG SET

Hàm find: kiểm tra sự tồn tại của một phần tử nào đó trong set, đây là một hàm được sử dụng rất nhiều của set vì độ phức tạp của nó là O(logN)

Hàm này trả về iterator tới phần tử nếu nó tìm thấy, ngược lại nó trả về iterator end() của set khi phần tử đó không tồn tại trong set

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

int main(){
    set<int> se;
    int a[7] = {5, 5, 1, 2, 3, 4, 5};
    for(int i = 0; i < 7; i++){
        se.insert(a[i]);
    }
    auto it = se.find(3);
    if(it == se.end()){
        cout << "NOT FOUND\n";
    }
    else{
        cout << "FOUND\n";
    }
}
```

OUTPUT: FOUND

MỘT SỐ HÀM TRONG SET

Hàm count: Hàm này dùng để đếm số lần xuất hiện của 1 phần tử trong set, đối với set thì 1 phần tử sẽ xuất hiện 1 hoặc 0 lần, có thể sử dụng hàm này để thay cho hàm find. Độ phức tạp O(logN).

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

int main(){
    set<int> se;
    int a[7] = {5, 5, 1, 2, 3, 4, 5};
    for(int i = 0; i < 7; i++)
        se.insert(a[i]);
    cout << se.count(1) << endl;
    cout << se.count(6) << endl;
    if(se.count(3) != 0)
        cout << "FOUND\n";
    else
        cout << "NOT FOUND\n";
}
```

OUTPUT:
```
1
0
FOUND
```

MỘT SỐ HÀM TRONG SET

Hàm erase: Xóa 1 phần tử khỏi set với độ phức tạp là O(logN), trước khi sử dụng hàm erase hãy đảm bảo phần tử bạn cần xóa tồn tại trong set nếu không sẽ xảy ra lỗi runtime error.

Xóa bằng giá trị

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

int main(){
    set<int> se;
    int a[7] = {5, 5, 1, 2, 3, 4, 5};
    for(int i = 0; i < 7; i++)
        se.insert(a[i]);
    for(int x : se)
        cout << x << ' ';
    cout << endl;
    se.erase(3);
    for(int x : se)
        cout << x << ' ';
}
```

OUTPUT: 1 2 3 4 5
1 2 4 5

MỘT SỐ HÀM TRONG SET

Hàm erase: Xóa 1 phần tử khỏi set với độ phức tạp là O(logN), trước khi sử dụng hàm erase hãy đảm bảo phần tử bạn cần xóa tồn tại trong set nếu không sẽ xảy ra lỗi runtime error.

Xóa bằng iterator

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

int main(){
    set<int> se;
    int a[7] = {5, 5, 1, 2, 3, 4, 5};
    for(int i = 0; i < 7; i++)
        se.insert(a[i]);
    for(int x : se)
        cout << x << ' ';
    cout << endl;
    auto it = se.find(3);
    if(it != se.end())
        se.erase(it);
    for(int x : se)
        cout << x << ' ';
}
```

OUTPUT: 1 2 3 4 5
1 2 4 5

Dự đoán output của một số chương trình sau

Quiz 1

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

int main(){
    int a[7] = {5, 5, 1, 2, 3, 4, 5};
    set<int> se(a, a + 7);
    auto it = se.find(7);
    --it;
    cout << *it << endl;
}
```

Quiz 2

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

int main(){
    int a[7] = {5, 5, 1, 2, 3, 4, 5};
    set<int> se(a, a + 7);
    auto it = se.find(4);
    it -= 2;
    cout << *it << endl;
}
```

Dự đoán output của một số chương trình sau

Quiz 3

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

int main(){
    int a[7] = {5, 5, 1, 2, 3, 4, 1};
    set<int> se(a, a + 7);

    cout << *se.begin() << endl;

    auto it = se.end(); --it;
    cout << *it << endl;
}
```

Quiz 4

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

int main(){
    int a[7] = {5, 5, 1, 2, 3, 4, 1};
    set<int> se(a, a + 7);

    cout << *se.begin() << endl;
    cout << *se.rbegin() << endl;
}
```

Dự đoán output của một số chương trình sau

Quiz 5

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

int main(){
    int a[7] = {5, 5, 1, 2, 3, 4, 1};
    set<int> se(a, a + 7);
    auto it = se.find(3);
    --it;
    se.erase(it);
    for(int x : se){
        cout << x << ' ';
    }
}
```

Quiz 6

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


int main(){
    int a[7] = {5, 5, 1, 2, 3, 4, 1};
    set<int> se(a, a + 7);
    for(auto it = se.rbegin(); it != se.rend(); ++it){
        cout << *it << ' ';
    }
}
```

2. MULTISET:

Multiset tương tự như set nhưng có thể lưu nhiều phần tử có giá trị giống nhau, các phần tử này cũng được sắp xếp theo thứ tự tăng dần. Các hàm của multiset giống y hệt các hàm của set chỉ khác một chút ở hàm find, count, erase.

MỘT SỐ HÀM TRONG MULTISET

Hàm find: Vì multiset có thể chứa nhiều phần tử giống nhau nên hàm find sẽ trả về iterator đến vị trí đầu tiên của phần tử có giá trị cần tìm kiếm.

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

int main(){
    int a[7] = {5, 5, 1, 1, 2, 2, 2};
    multiset<int> se(a, a + 7);
    //se ={1, 1, 2, 2, 2, 5, 5}
    auto it = se.find(2);
    cout << *it << endl;
    --it;
    cout << *it << endl;
}
```

OUTPUT: 2
1

MỘT SỐ HÀM TRONG MULTISET

Hàm count

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

int main(){
    int a[7] = {5, 5, 1, 1, 2, 2, 2};
    multiset<int> se(a, a + 7);
    //se ={1, 1, 2, 2, 2, 5, 5}
    cout << se.count(1) << endl;
    cout << se.count(2) << endl;
    cout << se.count(4) << endl;
}
```

OUTPUT:
```
2
3
0
```

MỘT SỐ HÀM TRONG MULTISET

Hàm erase: Khi sử dụng hàm erase nếu bạn xóa bằng giá trị multiset sẽ xóa hết mọi phần tử có cùng giá trị bị xóa, vì thế nếu muốn 1 phần tử bạn phải xóa bằng iterator

Xóa bằng giá trị

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

int main(){
    int a[7] = {5, 5, 1, 1, 2, 2, 2};
    multiset<int> se(a, a + 7);
    //se ={1, 1, 2, 2, 2, 5, 5}
    for(int x : se)
        cout << x << ' ';
    cout << endl;
    se.erase(2);
    for(int x : se)
        cout << x << ' ';
}
```

OUTPUT: 1 1 2 2 2 5 5
1 1 5 5

MỘT SỐ HÀM TRONG MULTISET

Hàm erase: Khi sử dụng hàm erase nếu bạn xóa bằng giá trị multiset sẽ xóa hết mọi phần tử có cùng giá trị bị xóa, vì thế nếu muốn 1 phần tử bạn phải xóa bằng iterator

Xóa bằng iterator

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

int main(){
    int a[7] = {5, 5, 1, 1, 2, 2, 2};
    multiset<int> se(a, a + 7);
    //se ={1, 1, 2, 2, 2, 5, 5}
    for(int x : se)
        cout << x << ' ';
    cout << endl;
    auto it = se.find(2);
    se.erase(it);
    for(int x : se)
        cout << x << ' ';
}
```

OUTPUT: 1 1 2 2 2 5 5
1 1 2 2 5 5

3. UNORDERED_SET

Unordered_set tương tự như set, nó chỉ có thể lưu các phần tử đôi một khác nhau nhưng lại không có thứ tự. Không có thứ tự tức là các phần tử ở trong unordered_set sẽ xuất hiện một cách bất kì, không phải là theo thứ tự bạn thêm phần tử vào.

Ví dụ:

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

int main(){
    int a[7] = {5, 5, 1, 1, 2, 2, 2};
    unordered_set<int> se(a, a + 7);
    //se ={1, 1, 2, 2, 2, 5, 5}
    for(int x : se){
        cout << x << ' ';
    }
}
```

OUTPUT: 2 1 5

3. UNORDERED_SET

Sự khác biệt

Unordered_set khác với set và multiset 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 set.

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