Map Trong C++

MAP, MULTIMAP, UNORDERED_MAP

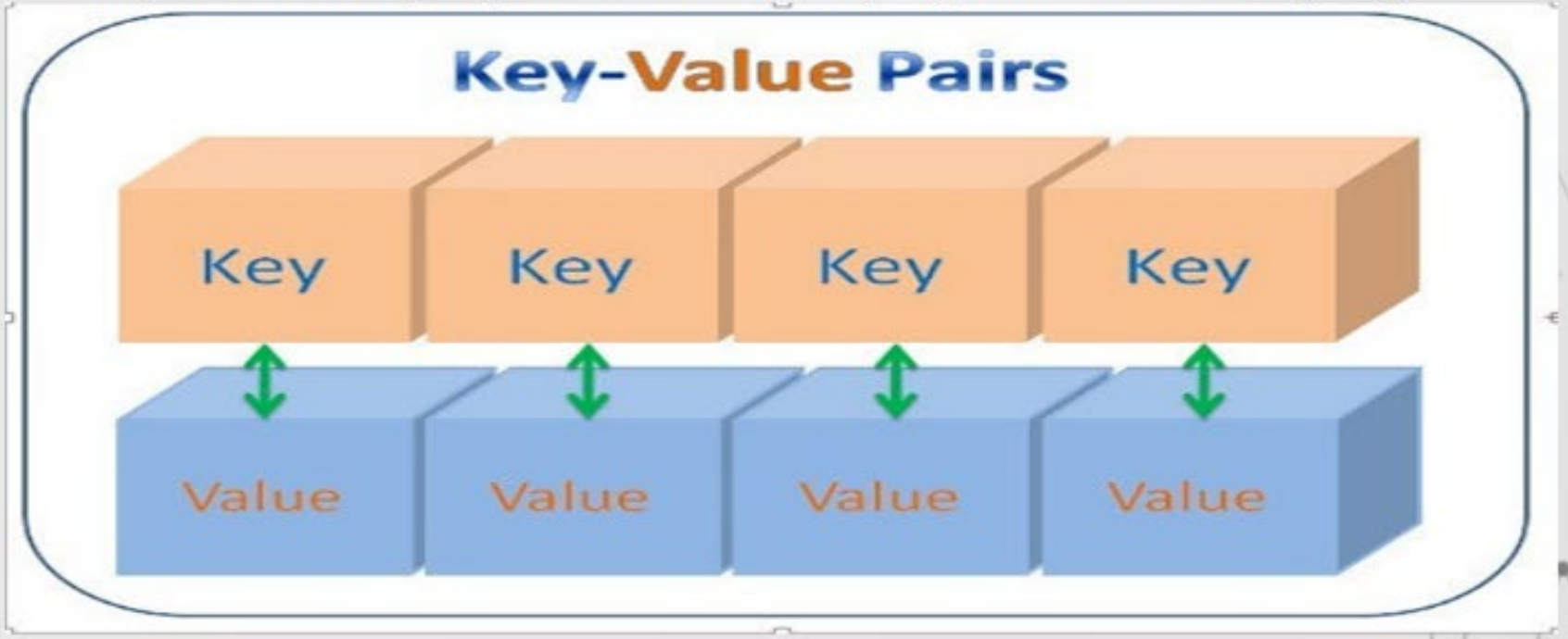
Map là gì?

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.

Ứng dụng:

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.

- Map có thể tìm kiếm nhanh.
- Map có thể dùng key làm index để truy cập vào value tương ứng.



1

Map.

Map

Tính Chất:

- Các key trong map là những giá trị riêng bi ệt, 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

Khai Báo:

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

Một số hàm thông dụng trong Map:

- Hàm **insert :** Thêm một phần tử vào trong map. **cú pháp** : map[key] = value hoặc map.insert({key,value})
- Hàm **size** : Trả về số lượng phần tử trong map. **cú pháp** : map.size() .
- 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. **cú pháp** : map.erase(value) .
- Hàm **clear** : xóa mọi phần tử trong map. **cú pháp** : map.clear() .
- Hàm **empty**: kiểm tra map rỗng, nếu rỗng trả về true, ngược lại trả về false. **cú pháp** : map.empty()
- 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. **cú pháp** : map.find(key) .
- 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). **cú pháp** : map.count(key) .

Cách Duyệt Map.

```cpp
map<int,int> mp;
mp.insert({1,2});
mp.insert({3,4});
mp.insert({2,1});
mp.insert({2,8});


//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;
}
```

output.out

```
1 2
2 1
3 4
```

2

MULTIMAP

MULTIMAP

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

-Khai Báo:

```
multimap<key_data_type, value_data_type> multimap_name;
```

MỘT SỐ HÀM TRONG MULTIMAP

- Hàm **find**: Vì trong multimap có nhiều key có cùng giá trị nên n ếu ta tìm kiếm giá trị của 1 key nào đó nó sẽ trả về vị trí xuất hi ện đầu tiên của key đó trong multimap. **cú pháp** : map.find(key) .
- Hàm **erase** : Xóa thông qua giá trị sẽ xóa tất cả các key tương ứng **cú pháp** : map.erase(value) .

3

UNORDERED_MAP.

UNORDERED_MAP

-Tính Chất:
Unordered_map cũng tương tự như map nhưng các key trong unordered_map không có thứ tự như map và multimap.
-Chú ý:
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 bi ệt so với map.
Ở map và multimap các hàm có độ phức t ạp là O(logN).
Ở unordered_map tốc độ của các hàm trong trường hợp tốt nhất là O(1) còn t ệ nhất có thể lên đến O(N).

-Khai Báo:

```
unordered_map<key_data_type, value_data_type> unordered_map_name;
```

THANKS

Full House Tất Cả Là Một Nhà!