1. Map

a. Khai báo:

map<kiểu dữ liệu, kiểu dữ liệu> tên\_map;

int main ()

{

map<string,int> mp = {

{ "alpha", 20 },

{ "beta", 20 },

{ "alpha", 10 },

{ "gamma", 30 },

{ "alpha", 10 },

};

for (auto x: mp) {

cout << x.first << ": " << x.second << endl;

}

cout<< mp.size() <<endl;

return 0;

}

Kq:

alpha: 20

beta: 20

gamma: 30

3

**b. Tìm phần tử trong map C++ bằng hàm find**

**Hàm count** là một hàm thành viên trong class std:map, có tác dụng đếm số lần xuất hiện của phần tử trong map C++ thông qua khóa của nó.

Chúng ta sử dụng hàm count trong C++ với cú pháp sau đây:

mp.count(key);

#include <iostream>

#include <map>

using namespace std;

int main() {

map<char,int> mp;

mp['a']=100;

mp['b']=200;

mp['c']=300;

//Đếm số lần xuất hiện của phần tử tồn tại trong map

cout << mp.count('b') <<endl;

//Đếm số lần xuất hiện của phần tử không tồn tại trong map

cout << mp.count('f') <<endl;

return 0;

}

Kq:

1

0

**Hàm equal\_range** là một hàm thành viên trong class std:map, có tác dụng tìm phạm vi của tất cả các phần tử có khóa giống với khóa chỉ định trong map.

Chúng ta sử dụng hàm equal\_range trong C++ với cú pháp sau đây:

mp.find(key);

#include <iostream>

#include <map>

using namespace std;

//Tạo hàm xuất map

void dump(map<char,int>& mp)

{

for (auto x: mp) {

cout << x.first << ":" << x.second << " ";

}

cout << endl;

}

int main() {

map<char,int> mp;

mp.insert(make\_pair('a', 1));

mp.insert(make\_pair('b', 2));

mp.insert(make\_pair('c', 3));

mp.insert(make\_pair('d', 1));

mp.insert(make\_pair('e', 2));

dump(mp);

//Tìm phần tử có khóa bằng 'c' trong map

auto ret = mp.equal\_range('c');

//Xóa phần tử vừa tìm thấy

mp.erase (ret.first,ret.second);

dump(mp);

return 0;

}

Kq:

a:1 b:2 c:3 d:1 e:2

a:1 b:2 d:1 e:2

**Hàm lower\_bound** là một hàm thành viên trong class std::map, có tác dụng tìm vị trí **phần tử đầu tiên** trong map có khóa lớn hơn hoặc bằng với khóa chỉ định.

Chúng ta sử dụng hàm lower\_bound trong C++ với cú pháp sau đây:

mp.lower\_bound(key);

Hàm lower\_bound() sẽ trả về trình lặp trỏ đến vị trí **phần tử đầu tiên** có khóa lớn hơn hoặc bằng với khóa chỉ định. Và nếu không tìm thấy, hàm sẽ trả về trình lặp trỏ đến vị trí cuối cùng trong map.

#include <iostream>

#include <map>

using namespace std;

//Tạo hàm xuất map

void dump(map<char,int>& mp)

{

for (auto x: mp) {

cout << x.first << ":" << x.second << " ";

}

cout << endl;

}

int main() {

map<char,int> mp;

mp.insert(make\_pair('a', 1));

mp.insert(make\_pair('b', 2));

mp.insert(make\_pair('c', 3));

mp.insert(make\_pair('d', 1));

mp.insert(make\_pair('e', 2));

mp.insert(make\_pair('f', 3));

dump(mp);

/\*Tìm vị trí phần tử đầu tiên có khóa lớn hơn hoặc bằng 'b' trong map\*/

auto itr1 = mp.lower\_bound('b'); // itr1 trỏ đến b:2

//Tìm vị trí phần tử đầu tiên có khóa lớn hơn 'e' trong map

auto itr2 = mp.upper\_bound('e'); // itr2 trỏ đến f:3

//Xóa các phần tử trong phạm vi [itr1, itr2)

mp.erase (itr1, itr2);

dump(mp);

return 0;

}

Kết quả, các phần tử trong phạm vi từ b:2 đến trước f:3 đã bị xóa đi.

a:1 b:2 c:3 d:1 e:2 f:3

a:1 f:3

**Hàm upper\_bound** là một hàm thành viên trong class std::map, có tác dụng tìm vị trí **phần tử đầu tiên** trong map có khóa lớn hơn khóa chỉ định.

Chúng ta sử dụng hàm upper\_bound trong C++ với cú pháp sau đây:

**mp.upper\_bound(key);**

#include <iostream>

#include <map>

using namespace std;

int main() {

map<char,int> mp;

mp['a']=100;

mp['b']=200;

mp['c']=300;

mp['d']=300;

mp['e']=300;

mp['f']=300;

/\*Duyệt map mp\*/

for (auto x: mp) {

cout << x.first << ":" << x.second << " ";

}

cout << endl; //a:1 b:2 c:3 d:1 e:2

//Tìm phần tử có khóa bằng 'b' trong map

auto itr1 = mp.lower\_bound('b'); // itr1 trỏ đến b:200

//Tìm vị trí phần tử đầu tiên có khóa lớn hơn 'e' trong map

auto itr2 = mp.upper\_bound('e'); // itr2 trỏ đến f:300

//In các phần tử trong phạm vi (itr1, itr2)

for (auto it=itr1; it!=itr2; ++it)

cout << (\*it).first << ":" << (\*it).second << ' ';

return 0;

}

Kq:

a:100 b:200 c:300 d:300 e:300 f:300

b:200 c:300 d:300 e:300