**关联容器类型**

// 关联容器支持高效的关键子查找

// 有序容器

map<int, string> map1; // 键值对

set<int> set1; // 键

multimap<int, string> map2; // 键值对，允许键重复

multiset<int> set2; // 键，允许键重复

// 无序容器，使用哈希排序

unordered\_map<int, string> map3;

unordered\_set<int> set3;

unordered\_multimap<int, string> map4;

unordered\_multiset<int> set4;

**初始化**

// 定义并赋值

map<string, string> keyValues { {"key1", "value1"} };

for(auto i : keyValues){

cout << "key:" << i.first << endl;

cout << "value:" << i.second << endl;

}

// 定义空的 map

map<string, string> keyValues1;

keyValues1["key1"] = "value1"; // 通过键赋值，如果对应的键值对不在，则新建

keyValues1["key2"] = "value2";

for(auto i : keyValues1){

cout << "key:" << i.first << endl;

cout << "value:" << i.second << endl;

}

set<string> keys { "key1" };

for(auto i : keys){

cout << "key:" << i << endl;

}

**自定义比较操作**

bool compareOpt(const string &left, const string &right){

return left.size() < right.size();

}

void customizeCompareOptTest(){

// 自定义比较操作，如果key没有 <运算 或 我们希望自定义比较

// decltype(compareOpt)\*中，decltype(compareOpt)等于compareOpt函数类型，语句的意思是 是个指针，指针指向compareOpt函数类型

map<string, string, decltype(compareOpt)\*> map1(compareOpt);

}

**pair**

// 键值，从map循环中取出的项便是pair

pair<string, string> pair1{"key", "value"};

cout << pair1.first << endl; // key

cout << pair1.second << endl; // value

**关联容器操作**

map<string, int> map1;

map<string, int>::key\_type keytype; // string

map<string, int>::mapped\_type maptype; // int

map<string, int>::value\_type valuetype; // pair<string, int>

map1.begin(); // 迭代器

map1.end();

map1.insert({"a", 1}); // 添加元素，若 "a" 已在map中则什么也不做

map1.erase("a"); // 删除 "a" ，返回删除数量（对于 multimap，删除数量可能大于1）

map1["b"] = 3; // 下标操作

auto b = map1["b"];

auto c = map1.at("b"); // 获取值

map1.find("a"); // 查找 "a"， 返回一个迭代器，指向 "a"，若为找到，返回尾后迭代器

map1.count("a"); // 查找 "a" 的数量

map1.lower\_bound("a"); // 返回关键子小于 "a" 的元素列表迭代器，迭代器指向列表最后一个元素的后一位

map1.upper\_bound("a"); // 返回关键子大于 "a" 的元素列表迭代器，迭代器指向列表第一位

**特殊的情况**

std::map<std::string, DocumentTitle>

如上，如果DocumentTitle没有默认的构造函数，那么我们无法使用下标操作

map1["b"] = 3; // 错误

**头文件**

#include <set>

#include <map>

#include <unordered\_map>

#include <unordered\_set>