# vector

# map

map内部实现是红黑树（一种二叉查找树），所以对map进行插入、删除后都会自动排序（默认按键值排序），若Key为自定义类型，则自定义类型需要重载operator< 函数。

1、自定义map的排序方式

map类的定义：

template < class Key, class T, class Compare = less<Key>, class Allocator = allocator<pair<const Key, T> > > class map;

重点看第三个参数class Compare = less<Key>，less是一个函数对象（即实现operator()的重载的类或结构体）。下面是less的实现：

template <class T> struct less : binary\_function <T, T, bool> {

bool operator() (const T& x, const T& y) const

{return x<y;}

};

与less相对的还有greater：

template <class T> struct greater: binary\_function <T, T, bool> {

bool operator() (const T& x, const T& y) const

{return x>y;}

};

可以知道map默认使用less函数对象对Key进行排序，即按键值从小到大排序。我们可以在声明map对象时指定函数对象，下面是使用自定义的函数对象对map进行排序。

1. 自定义键值排序的函数对象

struct CmpByKeyLength { //按键值字符串长度排序

bool operator()(const string& k1, const string& k2) {

return k1.length() < k2.length();

}

};

map<string, int, CmpByKeyLength> OrderMap; //指定排序使用的函数对象

接下来再对map进行插入、删除就会自动按照键值字符串长度排序了。

1. 若要希望map按值排序，则借助vector<pair<Key, Value>>对象。将map的键值对放入vector中，再定义一个比较函数传给sort函数，即可对vector排序，相当于对map的键值对按值大小进行排序。如下：

typedef pair<string, int> PAIR;

bool CmpByValue(const PAIR& lhs, const PAIR& rhs) {

return lhs.second < rhs.second;

}

vector<PAIR> OrderVec(OrderMap.begin(), OrderMap.end());

sort(OrderVec.begin(), OrderVec.end(), CmpByValue);

这种情况可以使用hash\_map或C++11的unordered\_map提高执行效率。

# unordered\_map（C++11）

unordered\_map内部使用hash表实现，其定义：

template <class Key,

class Ty,

class Hash = std::hash<Key>,

class Pred = std::equal\_to<Key>,

class Alloc = std::allocator<std::pair<const Key, Ty>>>

class unordered\_map;

1. 用法

* 头文件：#include <unordered\_map>
* 取键值

unordered\_map<Key,T>::iterator it; it->first; it->second;

* 成员函数

begin/cbegin end/cend （c开头表示返回const属性的迭代器）

empty 检查容器是否为空

size 返回容纳的元素数

max\_size 返回可容纳的最大元素数

clear 清空内容

insert 插入元素或节点（C++17）

insert\_or\_assign 插入元素，或若关键已存在则赋值给当前元素