逻辑航线信息学奥赛系列教程

map容器

概述

map是STL的一个关联容器，以键值对存储的数据，其类型可以自己定义，每个关键字在map中只能出现一次，关键字不能修改，值可以修改；map同set、multiset、multimap（与map的差别仅在于multimap允许一个键对应多个值）内部数据结构都是红黑树，而java中的hashmap是以hash table实现的。所以map内部有序（自动排序，单词时按照字母序排序），查找时间复杂度为O(logn)。

基本用法

1、头文件

```
#include<map>
```

2、定义

```
//方法一：
//以string类型为键
//以int类型为值
map<string,int> myMap1;

//方法二：
//为字典定义一个别名
typedef map<string,int> MyMap;
MyMap myMap2;
```

3、常用方法

函数名	功能
map.insert()	插入
map.find()	擦查找一个元素
map.count()	判断关键字是否存在
map.clear()	清空
map.erase()	删除一个元素
map.size()	map的长度大小
map.begin()	返回指向map头部的迭代器
map.end()	返回指向map末尾的迭代器
map.rbegin()	返回一个指向map尾部的逆向迭代器
map.rend()	返回一个指向map头部的逆向迭代器
map.empty()	map为空时返回true
swap()	交换两个map,两个map中所有元素都交换

4、代码示例

4-1 基本的读写，以及迭代器的使用

```
map<int, string> myMap;
```

```
//第一种：直接使用键值对进行赋值
myMap[1] = "swh";
//第二种方法，用insert函数插入pair数据：
myMap.insert(pair<int, string>(2, "first"));

//使用迭代器对字典进行读取
map<int, string>::iterator it;
for (it = myMap.begin(); it != myMap.end(); it++) {
    cout << it->first << " " << it->second << endl;
}
```

输出结果为：

```
1 swh
2 first
```

4-2 查找元素

```
map<string, int> myMap;
myMap["swh"] = 1;
myMap.insert(pair<string, int>("first", 2));

//方法1：用count函数来判断关键字是否出现，其缺点是无法定位元素出现的位置。
//      由于map一对一的映射关系，count函数的返回值要么是0，要么是1。
cout << myMap.count("first") << endl;

//方法2：用find函数来定位元素出现的位置，它返回一个迭代器，当数据出现时，
//      返回的是数据所在位置的迭代器；
//      若map中没有要查找的数据，返回的迭代器等于end函数返回的迭代器。
map<string, int>::iterator it;
it = myMap.find("1");
if (it != myMap.end()) {
    cout << "Find, the value is " << it->second << endl;
} else {
    cout << "Do not Find" << endl;
}
```

4-3 删除元素

```
map<string, int> myMap;
myMap["swh"] = 1;
myMap.insert(pair<string, int>("first", 2));
map<string, int>::iterator it;
it = myMap.find("1");
//删除成功返回1，失败返回0
int n = myMap.erase("1");
cout << n;
```

4-4 排序

map中元素是自动按key升序排序（从小到大）的

```
map<string, int> myMap;
//先存入b
myMap["b"] = 1;
//再存入a
myMap.insert(pair<string, int>("a", 2));
//遍历打印
map<string,int>::iterator it;
for (it = myMap.begin(); it != myMap.end(); it++) {
    cout << it->first << " " << it->second << endl;
}
```

输出结果如下：

```
a 2
b 1
```