实验舱蛟龙三班

Day13 set map应用

zlj

2022.12

Stl 介绍

部分数据结构与算法的封装，统一接口（命令方式），简化代码，NOIP NOI 都开放。(基本封装成容器）

Stl容器:

vector 、map、set、string

pair

一、pair< T1,T2> a

功能：pair 将一对值组合成一个值，这一对值可以具有不同的数据类型（T1和T2），两个值可以分别用pair的两个公有函数first和second访问

类似于结构体 ， 常用的函数 **make_pair(x,y)**

Pair 初始化：

1、pair<int, double> p1; //使用默认构造函数

2、pair<int, double> p2(1, 2.4); //用给定值初始化

数据访问：

```
pair<int, double> p;

p.first=1;

p.second=2.8;

cout<<p.first<<“  “ <<p.second;
```

数据输入：

pair<int, double> p[100];

1、 p[1]=make_pair(1,2.8);

2、 pair<int, double> p(2 ,2.8);

3、 pair<int, double> p2＝p;

pair排序规则是先 first ，后second

例：《坐标排序》

输入N个不同的坐标，请你按从左到右，从上到下的次序重新排序并输出。

输入格式：

第一行一个整数N（N<10000），表示有N个不同的坐标。
以下N行，每行两个整数x,y(x,y<=1000),表示坐标位置。

输出格式：

N行，每行两个整数x,y；

输入样例：

```
5
1 2
2 3
2 1
1 1
5 9
```

样例输出：

```
1 1
1 2
2 1
2 3
5 9
```

二、set（集合关联容器)

set容器是一个关联容器，容器中的元素互不相同，称其为主键（或者关键字），并且容器中的元素自动排好序。每当插入或者删除一个元素时，容器都会重新排序，排序的复杂度为O(logn)。

set容器有两大特点，一个是元素唯一性，另一个有序，查询复杂度为O(logn)（没有vector快，复杂度为O(1)）。

set 初始化：

```
set<int>s;      //定义int类型的set
set<int>::iterator it; //定义int类型set的迭代器
```

由于set是关联容器，内部元素不是按顺序存放，所以只能用迭代器访问set容器中的元素。

```
auto
```

Set（常用函数）

s.insert(x) 添加集合元素
s.begin () 返回指向第一个元素的迭代器
s.end () 返回指向最后一个元素之后的迭代器，不是最后一个元素
s.clear () 清除所有元素
s.count () 返回**bool**型,有返回**1**,无返回**0**
s.empty() 如果集合为空，返回**true**
s.erase( ) 删除集合中的元素
s.find() 返回一个指向被查找到元素的迭代器，如果没找到则返回**end()**
s.size() 集合中元素的数目
s.swap() 交换两个集合变量

set的应用

set的使用举例：

```
set<int>s;      //定义int类型的set
set<int>::iterator it; //定义int类型set的迭代器，set
for(int i=0;i<10;i++){
    cin>>x;
    s.insert(x); //往set里插入数据
}
for(it=s.begin();it!=s.end();it++)cout<< *it<<" ";
```

insert（x）函数会返回一个pair类型的值，其first是指向刚插入元素的迭代器。second是bool型表示插入是否成功。

思考：set元素是唯一的不重复，如果插入值x不在set中insert返回值是什么？如果x已经在set中返回值又是什么？

set的应用

```
set<int>s;
set<int>::iterator it;
pair< set<int>::iterator ,bool>rst; //定义pair类型的的变量
for(int i=0;i<10;i++){
    cin>>x;
    rst=s.insert(x); //往set里插入数据
    if(rst.second==true) cout<<x<<" Insert success"<<endl;
    else cout<<x<<" Insert fail"<<endl;
}
for(it=s.begin();it!=s.end();it++)cout<< *it<<" ";
```

我们可以根据second的值判断x是否已经存在,达到搜索中判重的目的。另外，插入重复值自动忽略的特点，可以用于去除重复的目的。

练习:

输入N个数，去重，并按升序输出。

【输入样例】

10

20 40 32 67 40 20 89 300 400 15

【输出样例】

8

15 20 32 40 67 89 300 400

例 ：

输入若干个单词，以空格分隔，统计有多少个不同的单词。

1、按字典序输出不同的单词

2、按原序输出不同的单词

样例输入：

no pain no gain

样例输出：

3
gain
no
pain

集合

类型名	作用
set<elemType>name	去重并且排序
multiset<elemType>name;	只排序不去重
unordered_set<elemType> name;	只去重不排序

set的常用函数总结：

set主要用于判重和去重，去重很简单不管什么往set里扔就好，反正它只保存一份，重复的值插不进去。判重呢？判重是在set中查找x，看x是否在set中，为此set给我们提供好几种查询的方法：

1、insert（key）：往set中插入元素，返回pair类型值，first指向插入关键字的迭代器，second表示是否成功。若key已经在set中则second返回false。

适用于：判断是否存在，如果不存在则插入

我们经常利用这个特点用于搜索状态的判重：如果当前状态之前没有出现，则插入set中，并继续搜索下去；如果之前出现过，则跳过不用继续搜索。

Set 作为集合的应用：

1、查询是否在集合？

count(key)：

返回key的数量 ，1或0，

```
set<int>s;
for(int i=0;i<10;i++){cin>>x;  s.insert(x);}
cin>>x;
if(s.count(x)==1) cout<<"Yes"<<endl;
else cout<<"No"<<endl;
```

适用于：只需判断元素是否存在，不存在也不用插入。

2、find(key)：返回指向key的迭代器，不存在返回end()；

例如：set<int>s;

```
set<int>::iterator it;
for(int i=0;i<10;i++){cin>>x;s.insert(x);}
cin>>x;
it=s.find(x);
if(it!=s.end()) cout<<"Yes"<<endl;
else cout<<"No"<<endl;
```

适用于：只需判断元素是否存在，不存在也不用插入。

3、lower_bound（key）：返回首个不小于（大于等于）key的元素的迭代器

当key存在，返回指向key的迭代器;

若大于key的元素也不存在，则返回end();

例如：2 3 6 8 9 10

it=s.lower_bound(3);it指向谁？

it=s.lower_bound(4);it又指向谁？

it=s.lower_bound(11);呢？

4、upper_bound(key)：返回首个大于key的元素的迭代器；若不存在，返回end()。例如：2 3 6 8 10 13

it=s.upper_bound(3);it指向谁？

it=s.upper_bound(4);it又指向谁？

思考：it1=s.lower_bound(3);

it2=s.upper_bound(8);

区间[it1,it2)表示了哪些元素？

it1=s.lower_bound(4);

it2=s.upper_bound(9); [it1,it2)区间内又有哪些元素？

可以利用lower_bound和upper_bound组合起来表示某个区间内的元素

set的常用函数

1、erase 可以删除单个元素，参数可以是删除元素的值；也可以是指向删除元素的迭代器；还可以是指向某个左闭右开的区间，区间要用相应的迭代器，表示删除对应区间内的元素。

例如　set<int>s; 假设s中有：2 3 6 8 9 10 13 16 23

```
set<int>::iterator it;
s.earse(6); //元素6被删除
it=s.begin();
s.erase(it); //删除元素2。
s.erase(s.lower_bound(3),s.upper_bound(9));
```

思考：这里删除了哪些元素？

注意：由于插入、删除元素后，set会重新排序，各元素位置可能发生变化，所以已经赋值的迭代器变量会失效，需要重新赋值才能使用。

集合

类型名	作用
set<elemType>name	去重并且排序
multiset<elemType>name;	只排序不去重
unordered_set<elemType> name;	只去重不排序

三、map（映射关联容器)

映射：就是两个元素的对应关系，从一个可以找到另外一个），如：

一个数字映射一个字符串，例如：学号——姓名；

字符串对应一个数字：例如：为一个字符串生成一串数字编码。

我们学过的数组其实就是一种映射，a[5]="Chinese";这里是下标数字映射为字符串。

STL提供了专门处理这种映射关系的容器——map

map的定义

map是一种关联容器，它提供一对一的数据处理能力（其中第一个可以称为关键字，每个关键字只能在map中出现一次，第二个可以称为该关键字对应的值）。

map数据元素是pair类型，其中first是关键字，seond是关键字对应的值。first是关键字不能修改，关键字对应的值second可以修改。

map容器中的元素是根据关键字first大小排好序的，每当插入或者删除一个元素，容器都会重新排序，复杂度为O(logn)。

一、map定义

```
map<int,string>p; //int为关键字，string为该关键字对应的值。
map<string,string>q；
```

Map容器数据装入：

二、数据插入

1、用insert函数插入pair类型数据（3种方式）

```
map<int,string>p
map<int,string>::iterator it;
p.insert(make_pair(1,"China"));
p.insert(pair<int,string>(5,"Japan")); //尖括号类变量类型要与map一致
p.insert(map<int,string>::value_type(3,"U.S.A") ) ;
for(it=p.begin();it!=p.end();it++)
        cout<< it->first <<" "<< it->second <<endl;
```

map数据插入

当map中有这个关键字时，insert操作是插入不了数据的，直接忽视这个insert操作，跟set类似这里insert也返回一个pair类型的值，first指向插入关键字的迭代器，second表示插入是否成功，为bool类型；可以检查second的值判断插入是否成功。

例如：map<int,string>p

```
pair< map<int,string>::iterator ,bool>rst;
rst=p.insert(make_pair(1,"China"));
if( rst.second==true )cout<<"1 Success"<<endl;
else cout<<"1 Fail"<<endl;
rst=p.insert(make_pair(1,"Japan"));
if(rst.second==true)cout<<"2 Success"<<endl;
else cout<<"2 Fail"<<endl; //思考：屏幕输出的内容是什么？
```

map数据插入

2、用下标[ ]方式：若关键字不存在，则添加；若存在则修改对应的值。

例1：
```
map<int,string>p;
map<int,string>::iterator it;
p[1]="China"; //这里下标1表示关键字，p[1]是关键字1对应的值
p[5]="Japan"; //下标不一定是连续的，与数组不同，仅表示关键字
p[3]="U.S.A";
for(it=p.begin();it!=p.end();it++)
    cout<<it->first<<" "<<it->second<<endl;
```

例2：
```
map<string,int>p;
p["China"]=1; //这里“China”就是关键字，1就是它对应的值
p["Japan"]=5;
```

map数据插入

注意：用下标[]引用map元素，如果对应关键字不存在，则会插入该关键字，其second赋值为默认值。

例如：map<int,int>p;

```
cout<<p.size()<<endl;
cout<<p[2]<<" "<<p.size()<<endl;
```

输出显示：0

```
0 1
```

因为2不存在，所以把2作为关键字插入map中。所以为了统计某关键字出现的次数，可以直接用p[2]++;

map数据插入

insert和[]法都可以向map中插入数据，但二者还是有区别的。insert遇到重复关键字，忽略插入操作。但是用[]方式就不同了，它是覆盖原来的值。

```
map<string,int>p;
p["China"]=2;
p["China"]=3;
cout<<p["China"]<<endl;
```

输出：3

map的遍历

关联容器我们一般用迭代器进行遍历访问。

```
map<string,int>::iterator it;
for(it=p.begin();it!=p.end();it++)
        cout<<it->first<<" "<<it->second<<endl;
```

map元素是pair类型，it迭代器相当于指针

map实战应用

例6、统计英文文章中出现了多少个不同的单词，以及这些单词出现的频率。

样例输入：

```
no pain no gain
```

样例输出：

```
3
gain 1
no 2
pain 1
```

map数据元素查找

判断一个数据（关键字）是否在map中

1、count（key）：返回0或1

```
map<int,int>p;
p[2]=3;
cout<<p.count(2)<<" "<<p.count(3)<<endl;
```

输出：1 0

2、find(key) ：若找到返回指向key的迭代器，否则返回end();

3、insert：检查返回值的second

map常用函数

1、insert 返回pair类型值，前一个指向关键字的迭代器，后一个表示是否成功

2、count(key)：返回key的数量，1或0，表示存在或不存在

3、find(key)：返回指向key的迭代器，不存在返回end()

4、clear()：清空map，删除所有元素

5、empty()：

6、lower_bound（key）：返回首个不小于（大于等于）key的元素的迭代器

7、upper_bound(key)：返回首个大于key的元素的迭代器，若不存在，返回end()

8、erase 删除元素，可以删除单个元素，参数是关键字；也可以删除某个左闭右开区间的元素，参数是迭代器

例：统计单词后缀数

给你若干单词，请统计出以某个字符串为前缀的单词数量(单词本身也是自己的前缀).

Input

输入数据的第一部分是一张单词表,每行一个单词,单词的长度不超过10,一个空行代表单词表的结束.第二部分是一连串的提问,每个提问都是一个字符串.(提问不超过100)

注意:本题只有一组测试数据,处理到文件结束.

Output

对于每个提问,给出以该字符串为前缀的单词的数量.

Sample Input

```
banana
band
bee
absolute
acm

ba
b
band
abc
```

Sample Output

```
2
3
1
0
```

```cpp
using namespace std;//统计单词前缀
char a;
map<string, int>st;
int main() {
    string sum="";
    while (1) {
        scanf("%c",&a);
        if (a=='\n') {//读到行末
            scanf("%c",&a);
            sum="";
        }
        if (a=='\n') {//读行末的下一位还是‘ \n’
            break;    //就退出
        }
        sum+=a;
        st[sum]++;
    }
    while (cin>>sum) //输出
        cout<<st[sum]<<endl;
    return 0;
```

例题讲解：
〈DD老师的通话记录〉

分析:将电话号码当作关键字，用map统计即可

〈最懒的小朋友〉

分析:数据给出了没迟到的小朋友，所以迟到的次数就是总次数减不迟到次数得到，打擂台找答案即可

map、set 区别

1、如果你想保证插入元素的唯一性，也就是你不想有重复值的出现，那么可以选择一个 **Set** 的实现类
2、如果你以键和值的形式进行数据存储那么 **Map** 是你正确的选择