1. multiset 基于红黑树实现的排序

template<class T, 数据类型

class Compare=less<T>, 比较类型（less，greater，默认less）

class Alloc=allocator<T>> 内存分配器

int myints[]= {2,16,30,30,30,40,77};

初始化：multiset<int,greater<int>> MS(myints, myints+6);//会自动排序

multiset<int,greater<int>>::iterator iter=MS.begin();

插入数据：insert MS.insert(50)

删除数据：erase MS.erase(iter) MS.erase(30) MS.erase(MS.begin(),MS.begin()+2)范围不包含后面一个迭代器

查找数据：find iter=MS.find(2) 返回迭代器，如果没找到，返回MS.end()

查找相同数据范围（因为是基于红黑树，排好序的，所以相同的数是挨着的）：equal\_range

std::pair<iter1,iter2> ret = mymultiset.equal\_range(30);返回的迭代器范围，数据不包含后一个迭代器

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

itlow = mymultiset.lower\_bound (30);//指向第一个与30相等的数

itup = mymultiset.upper\_bound (40); //指向最后一个与40相等的数的后一个数

mymultiset.erase(itlow,itup);//删除所有的30和40

遍历数据：for(auto iter=MS.begin();iter!=MS.end();++iter)

cout<<\*iter<<” “;

清除数据:clear MS.clear()

1. multimap（一对多索引）（基于红黑树实现set, multiset, map, multimap）

template<class Key, //键类型

class T, //映射类型

class Compare=less<Key>, //键比较

class Alloc=allocator<pair<const Key,T>> //内存器分配

> class multimap

multimap<int, string> mapStudent;

multimap<int, string>::iterator iter, beg, end;

插入操作：mapStudent.insert(std::pair<int,string>(0,”gxf”); 不支持下标操作

遍历操作：for(iter=mapStudent.begin();iter!=mapStudent.end();++iter)

Cout<<iter->first<<” “<<iter->second;

单键查询: int count=mapStudent.find(0);

Iter=mapStudent.find(0);

For(int i=0;i<count;++I,++iter)

Cout<<iter->first<<” “<<iter->second;

范围查询：beg=mapStudent.lower\_bound(0);//>=0

End=mapStudent.upper\_bound(0);//<=0

For(;beg!=end;++beg)

Cout<<iter->first<<” “<<iter->second;

删除：iter=mapStudent.find(0);

mapStudent.erase(iter);

清空: mapStudent.clear()

1. unordered\_map<char, string>{{'2', "abc"}, {'3', "def"}, {'4', "ghi"} }; 初始化

1. string

string str=to\_string(50);//数字转string

int i=stoi(str)

1. INT\_MAX +2147483647. 10位数

INT\_MIN -2147483648

所以INT\_MIN的绝对值会溢出

1. 小写字母的ASCII码比大写字母大，同样字母相差32

a~z 97~122

A~Z 65~90