sort:vector deque string [] <
list.sort()<
set<
map
priority_queue 从大到小

string s;
s.substr(pos,n);//copy
s.find()返回 int pos； s.npos没找到 find_first_of（） find_last_of（）
find（pos,elem）

upper_bound(a+i,a+j,x)-a返回的是第一个大于x的数的坐标
upper_bound(a.begin(),a.end(),x)返回的是迭代器
low_bound(first,last,x)返回的是第一个大于等于x的数的指针或者迭代器
不存在返回最后一个迭代器或位置；

pair<t1,t2> x(x1,x2)//初始化；
赋值：
1.x=y;变量间复制
2.x.first= x.second=;
3.x=make_pair(x1,x2);
访问 用 . first second;

vector： （下标赋值必须先对vector初始化一个值或者长度）
修改性：
c.pop_back() 删除最后一个数据。
c.push_back(elem) 在尾部加入一个数据。
c.insert(pos,elem) 在pos位置插入一个elem拷贝
c.erase(pos) 删除pos位置的数据
c.erase(beg,end) 删除[beg,end)区间的数据
c.clear() 移除容器中所有数据。
c.resize(num) 重新设置该容器的大小
查看性：

下标直接访问；
c.front()　　传回第一个数据。
c.back()：传回最后一个数据，不检查这个数据是否存在。
c.begin()　　返回指向容器第一个元素的迭代器
c.end()　　返回指向容器最后一个元素的迭代器
c.empty()　　判断容器是否为空。
c.size()　　回容器中实际数据的个数。

遍历法：
下标遍历
迭代器遍历

list （双向链表）
直接用find(gedin(,ebd(,elem)返回迭代器
迭代器:
begin() 返回迭代器开始(公共成员函数)
end() 返回迭代器结束(公共成员函数)
rbegin() 反向迭代器返回开始反向(公共成员函数)
rend() 反向迭代器返回反向结束(公共成员函数)
cbegin() const_iterator回到开始(公共成员函数)
cend() 返回const_iterator结束(公共成员函数)
crbegin() 返回const_reverse_iterator逆转开始(公共成员函数)
crend() 返回const_reverse_iterator扭转结束(公共成员函数)

容量capacity:
empty() 测试容器是否为空(公共成员函数)
size() 回的大小(公共成员函数)
返
max_size 返回最大大小(公共成员函数)

元素访问:
front() 访问第一个元素(公共成员函数)
back()访问的最后一个元素 (公共成员函数)

修饰符**Modifiers**:
push_front() 插入元素开始(公共成员函数)
pop_front() 删除第一个元素(公共成员函数)
push_back() 末尾添加元素(公共成员函数)
pop_back() 删除最后一个元素(公共成员函数)
insert() 插入元素(公共成员函数)
erase() 删除元素(公共成员函数)
swap() 交换内容(公共成员函数)
resize() 改变大小(公共成员函数)
clear() 清空内容(公共成员函数)
操作:
remove() 删除元素与特定的值(公共成员函数)
remove_if() 删除元素满足条件(公共成员函数模板)
unique() 删除重复的值(公共成员函数)
merge() 合并排序的列表(公共成员函数)
sort() 排序元素的容器(公共成员函数)
reserve() 改变元素的顺序(公共成员函数)
遍历法：迭代器

deque： (可直接下表初始化赋值)

非变动性操作
```
c.size();          //返回当前的元素数量
c.empty();         //判断大小是否为零。等同于c.size() == 0,但可能更快
c.max_size();      //可容纳元素的最大数量
c.front() ;        //返回第一个元素,不检查元素是否存在
c.back();          //返回最后一个元素
c.begin();         //返回一个随机迭代器，指向第一个元素
```

c.end();　　//返回一个随机迭代器，指向最后元素的下一位置

变动性操作：
c1 = c2 ；　　//将c2的所有元素赋值给c1;
c.push_back(elem);　//在尾部添加元素elem
c.pop_back()　；　//移除最后一个元素（但不回传）
c.push_front()　；　//在头部添加元素elem
c.pop_front()　；　//移除头部一个元素（但不回传）
c.erase(pos)　；　//移除pos位置上的元素，返回一元素位置　//如 c.erase(
c.begin() + 5）//移除第五个元素
c.insert(pos , elem); //在pos位置插入一个元素elem，并返回新元素的位置
c.insert(pos , n , elem); //在pos位置插入n个元素elem，无返回值
c.insert(pos , beg , end);
c.resize(num);　　//将容器大小改为num。可更大或更小。
c.resize(num , elem);　//将容器大小改为num，新增元素都为 elem
c.clear();　　//移除所有元素，将容器清空
遍历：下标 迭代器

set：
查看性：
s.size()　　// 集合中元素的数目
s.find()　　// 返回一个指向被查找到元素的迭代器
s.lower_bound() // 返回指向大于（或等于）某值的第一个元素的迭代器
s.upper_bound() // 返回大于某个值元素的迭代器
s.max_size()　// 返回集合能容纳的元素的最大限值
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.count()　　// 返回某个值元素的个数
s.empty()　　// 如果集合为空，返回true(真)
s.equal_range() // 返回集合中与给定值相等的上下限的两个迭代器
修改性：
s.erase()　　// 删除集合中的元素

s.clear()　　// 清除所有元素

s.insert()　　// 在集合中插入元素

s.swap()　　// 交换两个集合变量

遍历：迭代器

map：

at	查找具有指定键值的元素。
begin	返回一个迭代器，此迭代器指向映射中的第一个元素。
cbegin	返回一个常量迭代器，此迭代器指向映射中的第一个元素。
cend	返回一个超过末尾常量迭代器。
clear	清除映射的所有元素。
count	返回映射中其键与参数中指定的键匹配的元素数量。
crbegin	返回一个常量迭代器，此迭代器指向反向映射中的第一个元素。
crend	返回一个常量迭代器，此迭代器指向反向映射中最后一个元素之后的位置。
emplace	将就地构造的元素插入到映射。
emplace_hint	将就地构造的元素插入到映射，附带位置提示。
empty	如果映射为空，则返回 true。
end	返回超过末尾迭代器。
equal_range	返回一对迭代器。 此迭代器对中的第一个迭代器指向 map 中其键大于指定素。 此迭代器对中的第二个迭代器指向 map 中其键等于或大于指定键的第
erase	从指定位置移除映射中的元素或元素范围。
find	返回一个迭代器，此迭代器指向映射中其键与指定键相等的元素的位置。
get_allocator	返回用于构造映射的 allocator 对象的副本。
insert	将元素或元素范围插入到映射中的指定位置。
key_comp	返回用于对映射中的键进行排序的比较对象副本。
lower_bound	返回一个迭代器，此迭代器指向映射中其键值等于或大于指定键的键值的第
max_size	返回映射的最大长度。

rbegin	返回一个迭代器，此迭代器指向反向映射中的第一个元素。
rend	返回一个迭代器，此迭代器指向反向映射中最后一个元素之后的位置。
size	返回映射中的元素数量。
swap	交换两个映射的元素。
upper_bound	返回一个迭代器，此迭代器指向映射中其键值大于指定键的键值的第一个元
value_comp	检索用于对映射中的元素值进行排序的比较对象副本。
shrink_to_fit	放弃额外容量。
size	返回vector元素个数
swap	交换两个向量的元素。

遍历：迭代器

1.stack （不支持遍历，没有迭代器）

```
stack<int> s;
stack< int, vector<int> > stk; //覆盖基础容器类型，使用vector实现stk
s.empty(); //判断stack是否为空，为空返回true，否则返回false
s.size(); //返回stack中元素的个数

s.top(); //返回栈顶元素的值，但不删除此元素
s.pop(); //删除栈顶元素，但不返回其值
s.push(item); //在栈顶压入新元素item
```

2.queue & priority_queue

```
queue<int> q; //priority_queue<int> q;
q.empty(); //判断队列是否为空
q.size(); //返回队列长度
```

```
q.pop(); //删除队首一个元素
q.push(item);  //对于queue，在队尾压入一个新元素
               //对于priority_queue，在基于优先级的适当位置插入新元素

//queue only:
q.front();  //返回队首元素的值，但不删除该元素
q.back();  //返回队尾元素的值，但不删除该元素

//priority_queue only:
q.top();   //返回具有最高优先级的元素值，但不删除该元素
```