顺序容器

vector	**可变数组，支持快速随机访问，在尾部之外插入、删除很慢**
deque	双端队列，支持随机访问，在头尾插入删除很快
list	双向链表，支持顺序访问，在任何位置插入删除很快
forward_list	单向链表……
array	固定大小数组，支持快速访问，不能添加、删除元素。array<int,10> a;
string	与vector相似的容器，保存字符，随机访问，在尾部插入、删除很快

公共接口：

类型别名	
iterator	此容器的迭代器类型
构造函数	
C c;	默认构造函数
C c(c0);	通过c0构造c
C c(b,e)	构造c，将迭代器b和e指定范围内的元素拷贝到c
只有特定容器特有的初始化	
C seq(n)	包含n个元素，并初始化，不适用于string
C seq(n,t)	包含n个初始化为t的元素
赋值与替换	
c1 = c2	将c1 中的元素替换成c2中的元素
c1 = {a, b, c ...}	将c1中的元素替换为列表中的元素
a.swap(b)	a，b相互交换元素
添加/删除元素	不适用于array
c.insert()	插入元素，参数自己看
c.emplace(inits)	使用inits构造c中的一个元素
c.erase(args)	删除args
c.clear()	删除容器中所有元素
关系运算符	
==, !=	所有容器都支持
<, <=, >, >=	无序关联容器不支持
获取迭代器	
c.begin(),c.end()	
c.cbegin(), c.cend()	
反向容器额外成员	不支持forward_list
c.rbegin(), c.rend()	
c.crbegin(), c.crend()	

```
使用assign(仅顺序容器):允许我们从一个不同但相容的类型赋值
list<string> names;
vector<const char* > oldstyle;
names = oldstyle;   //报错
names.assign(oldstyle.cbegin(),oldstyle.cend());    //可以
```

改变容器的大小-> resize()

注意：当我们删除元素时，尾部迭代器会失效

- [x] 在使用erase，insert删除和插入后，会自动跟新迭代器么，没确认，已确认，更新

管理容量的成员函数：

c.shrink_to_fit()	**退回不需要的空间，适用于vector、deque、string**
c.capacity()	不从新分配内存，c可以保存多少元素
c.reserve(n)	分配至少能容纳n个元素的空间

额外string操作

s.substr(pos,n)	**返回要给string，pos开始位置，拷贝n个**
s.append("+str")	尾部追加
s.replace(2,3,"newstr")	从位置2开始，删除3个字符串，并插入“newstr”
s.find(args)	查找args第一次出现的位置
s.rfind(args)	查找args最后一次出现的位置
to_string(val)	将val转换为字符串
stoi(str)	将str转换为int类型
stol(str)	……long
stod(str)	double

容器适配器（使用容器来编写）：

栈	**队列**	**优先队列**
stack	queue	priority_queue(排序了，估计)