queue容器

queue就是队列（先进先出）

首先得包含头文件#include<queue>

然后定义queue<typename> name;

typename可以是int、float、还可以是自己定义的结构体。

例如：

```
struct factor
{
    int a;
    int b;
}
queue<factor> q;
```

queue的常用函数

- push(): push(x)实现对x进行入队
- front（）、back（）：分别获得队首和队尾元素
- pop（）：令队首元素进行出队
- empty（）：检测队列queue是否为空，如果为空则返回true，不为空则返回false
- size（）：返回queue内元素的个数

优先队列——堆排序

操作和队列的基本一致，只是这个优先队列对将队列中的数据进行自动排序。

```
//升序队列，小顶堆
priority_queue <int,vector<int>,greater<int> > q;
//降序队列，大顶堆
priority_queue <int,vector<int>,less<int> >q;

greater和less是std实现的两个仿函数（就是使一个类的
使用看上去像一个函数。其实现就是类中实现一个
operator()，这个类就有了类似函数的行为，就是一个仿
函数类了）？
```

下面是基本操作：

和队列基本操作相同:

q.top()访问队头元素
q.empty()队列是否为空
q.size()返回队列内元素个数
q.push()插入元素到队尾（并排序）
q.emplace()原地构造一个元素并插入队列
q.pop()弹出队头元素
q.swap()交换内容

deque容器（双向队列）

容器deque和vector非常相似，属于序列式容器。都是采用动态数组来管理元素，提供随机存取，并且有着和vector一样的接口。不同的是deque具有首尾两端进行快速插入、删除的能力。

创建deque

- daque<Elem>c;　　创建一个空的deque
- deque<Elem>c1(c2);　　复制deque,复制跟c1一模一样的队列c2
- deque<Elem>c(n);　　创建一个deque,元素个数为n,且值均为0
- deque<Elem>c(n,num);　创建一个deque,元素个数为n,且值均为num

deque数据访问

- c.at(idx); 返回索引下标idx所指的数据（从0开始）
- c.front(); 返回第一个数据
- c.back(); 返回最后一个数据
- c.begin(); 返回指向第一个数据的迭代器
- c.end(); 返回指向最后一个数据的下一个位置的迭代器
- c.rbegin(); 返回逆向队列的第一个数据

deque插入数据

- c.push_back(num); 在尾部加入一个数据num
- c.push_front(num); 在头部插入一个数据num
- c.insert(pos,num); 在该pos位置的数前面插入一个num

- c.insert(pos,n,num); 在该pos位置的数前面插入n个num

deque删除数据

- c.pop_back(); 删除最后一个数据
- c.pop_front(); 删除头部数据
- c.erase(pos); 删除pos位置的数据