构造函数（（constructor））

接口说明：

① 空队列

```
explicit deque(const A& al = A());
```

②创建n个value值的队列，第三个参数是具有默认值的空间配置对象

```
explicit deque(size_type n, const T& v = T(), const A& al = A());
```

③用deque对象拷贝构造deque对象

```
deque(const deque& x);
```

④用迭代器［begin，end）区间中的元素构造deque，第三个参数是具有默认值的空间配置对象

```
deque(const_iterator first, const_iterator last,
const A& al = A());
```

⑤用数组元素的地址做为参数创建队列

```
template<class _Iter,
    class = typename enable_if<_Is_iterator<_Iter>::value,
    void>::type>
    deque(_Iter _First, _Iter _Last)
    : _Mybase()
    {   // construct from [_First, _Last)
    _Construct(_First, _Last);
    }
```

① 空队列

```
deque<int> de;
```

②创建n个value值的队列，第三个参数是具有默认值的空间配置对象

```
deque<int> de1(10, 2);
```

③用deque对象拷贝构造deque对象

```
deque<int> de3 = de2;  //deque<int> de3(de2);
```

④用迭代器［begin，end）区间中的元素构造deque，第三个参数是具有默认值的空间配置对象

⑤

```
int main()
{
	int ar[] = {1,2,3,4,5,6,7,8,9,10};
	deque<int> de2(ar, ar+sizeof(ar)/sizeof(int));

	deque<int>::iterator it = de2.begin();
	while(it != de2.end())
	{
		cout<<*it<<" ";
		++it;
	}
	cout<<endl;

	deque<int>::reverse_iterator rit = de2.rbegin();
	while(rit != de2.rend())
	{
		cout<<*rit<<" ";
		++rit;
	}
	cout<<endl;
 return 0;
}
```

queue的遍历

```
deque<int> de1(10, 2);

deque<int>::iterator it = de1.begin();
	while(it != de1.end())
	{
		cout<<*it<<" ";
		++it;
	}
	cout<<endl;

	for(auto e : de1)
		cout<<e<<" ";
```

```
    cout<<endl;

    for(int i=0; i<de1.size(); ++i)
        cout<<de1[i]<<" ";
    cout<<endl;
```

insert 和 erase

```
deque<int> de;
    de.push_back(1);
    de.push_back(2);
    de.push_back(3);
    de.push_front(4);
    de.push_front(5);  //4

    deque<int>::iterator pos = de.begin();
    pos++;
    de.insert(pos, 10);

    pos = find(de.begin(), de.end(), 1);
    de.erase(pos);
```

函数声明	接口说明
begin(), end()	begin：容器起始位置end最后一个元素下一个位置
rbegin(), rend()	反向迭代器rbegin在end位置，rend在begin
cbegin0, cend()	const迭代器，与begin和end位置相同，但不能修改其空间内容
crbegin(), crend()	const反向迭代器，与crbegin在cend位置，crend在cbegin位置

deque的容量操作

函数声明	接口说明
size()	返回deque中有效元素个数
empty()	检测deque是否为空，是返回true，否则返回false
resize(sz, valuel)	将deque中的元素改变到sz，多出的空间用value填充

deque的元素访问操作

函数声明	接口说明
operator[]	返回deque中n位置上元素的引用
front()	返回deque中首元素的引用
back()	返回deque中最后一个元素的引用

deque中修改操作

函数声明	接口说明
push back()和pop_back()	deque的尾插和尾删
push frontQ和pop_front()	deque任意位置插入和删除
insert(pos , value) 和 erase(pos)	删除deque头部元素
swap()	交换两个deque中的内容
clear()	将deque中的元素清空