# 基础数据结构及STL



## 向量（vector）

叫动态数组或者变长数组应该都可以的。

特点：非常重要的数据结构，即有数组性质，也能很方便的完成离散化等操作

**声明方法**

①vector<数据类型>向量名(向量长度)
②vector<数据类型>向量名
③vector<数据类型>向量名(向量长度,初始化值)

**基本操作**

```
push_back(x)//将x添加至末尾
pop_back()//删除最后一个数
front() //返回最后一个元素
back()//返回最后一个元素
size()//返回向量中元素的个数
insert(x,y)//在x位置插入元素（注意x通常由迭代器指定）
```

# 对（pair）

可以将两个数据组合成一组，实现方式是结构体。

特点：最简单的容器，可以很简单的组合两个元素，在数据类型组合非常多的情况下可以代替struct压行。

```
pair<T1,T2> p1; //声明一个pair对象p1,first类型为T1，second类型为T2
pair<int,double> p2=make_pair(v1,v2) //创建v1和v2组合成的pair对象
pair<string,int>name_age("Tom",18);//第三种创建方法
cout<<name_age.first<<" "<<name_age.second<<endl;//
```

# 链表（list）

**万年没用过了，大家可以跳过这个**

特点：不能随意访问任意元素，只能从表头循环遍历

**声明方式**

①list<数据类型>数据名;
②list<数据类型>数据名(初始长度);
③list<数据类型>数据名(初始长度,初始值);

**基本操作**

```
push_back(x)//表尾添加元素x
pop_back()//删除表尾元素
push_front(x)//表头添加元素x
pop_front()//删除表头元素
erase(x)//删除迭代器x位置的元素
remove(x)//删除值为x的元素
```

# 集合（set）

特点：无重复元素，增删元素后会自动排序

声明方式：set<数据类型>集合名;

**基本操作**

```
clear()//清除所有元素
count(x)//返回集合中x的个数（是否存在）
size()//返回集合中元素个数
empty()//判断set容器是否为空
insert(x)//将x插入集合中
erase(x)//将x从集合中删除
find(x)//返回指向x的迭代器
```

**关于效率**

map和set的插入删除效率比其他序列容器高。set的元素是以节点方式存储的，节点的结构和链表差不多，指向父节点和子节点。set的查找使用的是二分，在logn复杂度内能够找到。

**多重集合(multiset)**

近视理解为元素可重复的set，声明方式:multiset<数据类型>集合名

# 映射（map）

声明方式：map<数据类型1，数据类型2>映射名

特点：红黑树实现，内部元素按字典序，每种元素只能存在一个。对空间的占用较高，但是对查找、插入操作都很高效。

基本操作：

```
map<string,int>mp;
mp["字符串"]=2412;
is=mp.find(x);//返回键为x的地址，若没有，则返回end()即最后一个映射关系的地址
for(map<string,int>::iterator it=mp.begin();it!=mp.end();it++){
    cout<<(*it).first<<" "<<(*it).second<<endl;
}
```

**unordered_map**

效率更高的一种map，记录元素的hash值，根据hash判断元素是否相同。

特点：内部元素无序

# 栈（stack）

声明方法：stack<数据类型>栈名
特点：先进先出
基本操作：

```
push(x)--将x进栈
pop()--删除栈顶元素
top()--返回栈顶元素
size()--返回栈中元素个数
empty()--返回栈是否为空
```

# 队列（queue）

声明方法：queue<数据类型>队列名;
特点：先进后出

**基本操作**

```
push(x)//将x入队
pop()//弹出队列第一个元素
front()//返回队头元素
back()//返回队尾元素
empty()//返回队伍是否为空
size()//返回队伍元素个数
```

**优先队列（priority_queue）**

增删元素后自动排序的队列，默认从大到小

**声明方式**

①从小到大的队列：priority_queue< int,vector <int> ,greater <int> > q1;
①从大到小的队列：priority_queue< int,vector <int> ,less <int> > q2;

**基本操作**

```
push(x)//将x入队
pop()//删除队首元素
top()//返回队顶元素
size()//返回优先队列中元素个数
empty()//返回队列是否为空
```

**双向队列（deque）**

**特点**

双端插入和删除

#####基本操作

```
deque.size()
deque.empty()
deque.push_back() //向后插入
deque.push_front() //向前插入
deque.pop_back() //后部弹出
deque.pop_front() //前部弹出
deque.front()
deque.back()
deque.resize(num, elem); //重新指定容器的长度为num,若容器变长，则以elem值填充新位置,如果容器变短，则末尾超出容器长度的元素被删除。
```

**迭代器**

```
for(deque<int>::const_iterator it=d.begin();it!=d.end();it++){
        cout<<*it<<" ";
}
```

例题：luogu [P1886 滑动窗口 /【模板】单调队列]

## 参考资料


https://blog.csdn.net/yas12345678/article/details/52601454

https://blog.csdn.net/sevenjoin/article/details/81937695

https://www.cnblogs.com/langyao/p/8823092.html