C++primer学习笔记（9–10章）

第九章 顺序容器

· 顺序容器：容器即一种特定元素的集合，顺序容器提供了控制元素存储和访问元素的能力

顺序容器类型

- vector　　可变大小数组，支持随机访问
- deque　　双端队列，支持随机访问，但在中间位置添加和删除元素代价很高
- list　　双向链表，支持双向顺序访问
- forword_list　　单向列表，支持单向顺序访问
- array　　固定大小数组，支持随机访问
- string　　与 vector 相似，用于保存字符，支持随机访问

迭代器

forward_list（单向链表）不支持递减运算符

迭代器范围

- begin 指向首元素
- end 是指向最后一个元素的后一个位置

```C++
list<srting> a={"test","demo"};
auto it1=a.begin();//list<string>::iterator
auto it2=a.rbegin();//list<string>::reverse_iterator
auto it3=a.cbegin();//list<string>::const_iterator
auto it4=a.crbegin();//list<string>::const_reverse_iterator
```

容器定义与初始化

```C++
//初始化
vector<int> vec1={1,2,3,4,5,6};//列表初始化
vector<string>svec(10,"hi");//仅顺序容器支持大小相关的构造函数
//拷贝
vector<int> vec2(vec);
vector<int> vec3(vec1.begin(),vec2.end());
```

标准库 array

- 与内置数组一样具有固定大小，定义时需指定数组大小
- 由于大小是 arr 类型的一部分，故其不支持普通容器的构造函数
- array 支持拷贝，要求类型与大小一样
- 不支持 assign 操作

```C++
//初始化
array<int,3> arr1;//垃圾值
```

```
array<int,3> arr2={1};//arr[0]被初始化为1，其余为0
arr1=arr2;//拷贝
```

容器赋值运算

- 赋值相关运算会导致左边容器的迭代器、引用、指针失效，而 swap 不会

c1=c2	拷贝
c2={a,b,c}	初始化列表
swap(c1,c2)　c1.swap(c2) 可以保证在常数时间内完成	交换 ,除 array 外，swap 只是交换了两个容器的内部数据结构，因此
seq.assign(b,e)	将 seq 中的元素替换为 b 和 e 范围中的元素
seq.assign(s)	将 seq 中的元素替换为 s 中的元素
seq.assign(n,t)	将 seq 中的元素替换为 n 个值为 t 的元素

容器大小比较

- 若容器大小相同且元素两两对应则两容器相等
- 其他情况从首元素逐一比较

元素插入

C++

```
vector<int> vec1={1,2,3};
vector<int> vec2={4,5,6};
vec2.insert(vec2.begin(),0);//在向量头部插入0
vec2.insert(vec2.end(),10,0);//在向量尾部插入10个0
vec2.insert(vec2.begin(),vec1.end()-2,vec1.end())//插入一个范围
```

emplace 操作

- 包括 emplace_front、empace、emplace_back
- 构造元素而不是拷贝元素,emplace 成员传递给相应构造函数，使用参数在内存空间中直接构造元素

访问元素

- 在容器中访问成员函数返回的都是引用,故只要不是 const 引用，都能用来改变元素的值

删除元素

c.pop_back()	删除尾元素
c.pop_front()	删除首元素
c.erase(it)	删除迭代器指定位置元素，返回被删除元素之后迭代器位置,若 it 指向尾元素，则返回尾后迭代器
c.erase(b,e)	删除范围内元素
c.clear()	删除所有元素，返回 void

vector 对象的增长

- 通常会分配比实际情况更大的内存空间以防止频繁移动元素
- capacity 是在不重新分配内存空间的情况下最多可以保存多少个元素，size 是指已经保存的元素个数

构造 string

string s(arr,n)	拷贝数组 arr 的前 n 个字符
string s(str,pos)	拷贝 str 字符串，从 pos 开始

string s(str,pos,n)　拷贝 str 字符串，从 pos 开始拷贝 n 个

第十章 泛型算法

只读算法：只读取输入范围里的元素，而不改变元素

- find　查找元素
- accumulate　数值求和，字符串连接
- equal　用于确认两个序列中是否保存相同的值

```C++
vector<int> vec={1,2,3};
vector<int>::iterator it=find(vec.begin(),vec.end(),1);//查找元素1
int sum=accumulate(vec.begin(),vec.end(),0);//累加，以0为初值
vector<int> vec2={1}
bool res=(vec.begin(),vec.end(),vec2.begin());//返回bool值，判断在范围内是否具有相同值
```

写容器元素的值，

- fill　fill_n　对范围内进行赋值
- back_insert_iterator　接受一个指向容器的引用
- copy　拷贝算法
- replace　替换算法
- sort　排序函数
- uniqe　返回一个指向不重复值范围末尾的元素

```C++
vector<int> vec={1,1,2,3};
fill(vec.begin(),vec.end(),0);//将范围内元素置0
fill_n(vec.begin(),vec.size(),0);//将指定位置开始的一定数目的元素赋值为0
auto it=back_inserter_iterator<vector<int>>(vec);
*it=14;//push_back(14)
int arr1[]={1,2};
int arr2[2];
copy(begin(arr1),end(arr2),arr2);//内置数组拷贝
replace(vec.begin(),vec.end(),1,5);//将范围内所有1替换为5
sort(vec.begin(),vec.end(),less<int>());//升序排序
uniqe(vec.begin(),vec.end());//size不变，向量变为{1,2,3,1}
```

谓词：谓词是一个可调用的表达式，返回结果是一个能用作条件的词

lambda 表达式

- 当定义一个 lambda 时，编译器生成一个与 lambda 对应的新的（未命名）类类型
- [capture list](parameter list) – > return type {function boby}
- 隐式捕获采用&和=操作符。&表示捕获引用，=捕获值
- 一个值被拷贝的元素，lambda 改变其值需在参数列表首部加上关键字 mutable

```C++
int s=4;
auto fn=[s]{return s;};
//遍历
vector<int> vec{1,2,3};
for_each(vec.begin(),vec.end(),[](int s) 777+{cout<<s<<endl;});
```

标准库 bind 函数

- 用以解决参数长度问题
- auto newCallable = bind(callable,arg_list) 其中 arg_list 表示参数列表，callable 表示一个函数
- 命名空间 placeholders：名字_n 都定义在这个空间，而这个命名空间本身定义在 std 命名空间中

插入迭代器

back_inserter	创建的一个使用 push_back 的迭代器
front_inserter	创建的一个使用 push_front 的迭代器
inserter	创建的一个使用 insert 的迭代器

iostream 迭代器

- istream_iterator 读取输入流 ostream_iterator 读取输出流

```C++
//输入流运用
vector<int> vec;
istream_iterator<int> int_it(cin);//从cin读取int
istream_iterator<int> eof;//尾后迭代器
while(int_it!=eof)//当未遇到文件尾部或者IO错误
{
    vec.push_back(*int_it++);
}
//输出流运用
ostream_iterator<int> out_iter(cout," ");
for(auto e:vec)
{
    *out_iter++=e;
}
cout<<endl;
copy(vec.begin(),vec.end(),out_iter);//一种便捷打印
```

五类迭代器

- 输入迭代器　　可以读取序列中的元素，只能用于单向扫描算法
- 输出迭代器　　只写而不读元素，只能用于单向扫描算法
- 前向迭代器　　可以读写元素，只能沿着序列的一个方向移动
- 双向迭代器　　支持双向读写元素
- 随机访问迭代器　　提供常量时间内访问元素的能力