泛型算法：介绍了一些泛型算法&lambda，并没有说如何定义泛型算法

只读算法有：

```
find(c.begin(),c.end(),value);  // 返回跟跟value匹配的指针
accumulate(c.begin(),c.end(),defaultvalue); // 累加，初始值为defaultvalue
fill(c.begin(),c.end(), 10);     // 将指定范围内值置为10
fill_n(c.begin(),c.size(),10);  // 从c.begin()开始，共c.size()个元素置为10

back_inserter:插入迭代器
    vector<int> vec;
    auto it = back_inserter(vect);  // 返回一个插入迭代
    *it = 42;    // 赋值运算符会调用push_back将一个具有给定值的元素添加到容器中
    fill_n(back_inserter(vec), 10,0); // 不用担心越界了

auto b = copy(c.begin(),c.end(),a); //把c的内容拷贝给a，b指向a尾元素之后的位置
repace(c.begin(),c.end(),0,33); // 将范围内为0的值替换为33
repace_copy(c.cbegin(), c.cend(), back_inserter(a),0,33);    // 替换并将复制给a容器，不改变c容器。
sort(c.begin(),c.end()); // 将c排序
auto end_unique = unique(c.begin(),c.end());     //把重复项移到最后，并返回指向重复项一个迭代指针，然后你就可以通过此指针删除（erase)后面元素。

```

lambda表达式

$$[capture](parameterlist)->returntype\,functionbody \tag{1}$$

```
capture:时lambda所在函数中定义的局部变量列表
int a = 100;
auto f = [a]()->void {cout << a << endl;}
f();
```

```
auto wc = find_if(words.begin(), words.end(), [](const string& a)->bool{return a.size()>6});
    //返回第一个字符串大小大于6的迭代器
for_each(words.begin(),words.end(),[](const string& s){cout << s << '\t';})  //作用于每个元素上
```

capture 变量前加&为引用，不加为赋值————在lambda创建时就拷贝了，后期外部再改变此值无效。

bind(绑定参数,在functional中)

```
auto newCallable = bind(callable,arg_list);
ps:
auto f = [](int a, int b){cout << a+b << endl;}
auto newf = bind(f,placeholders::_1,2);
newf(3); <=> f(3,2);
```

注：placeholders::_1 代表第一个参数，placeholders::_2 代表第二个参数.....

注：bind只是拷贝参数，如果参数时i/ostream，则不能拷贝，会报错

再探迭代器

插入迭代器

back_inserter	创建一个使用push_back的迭代器
front_inserter	创建一个使用push_front的迭代器
inserter	插入呗，并返回新元素的位置（迭代器）

iostream迭代器

```
vector<int> vec;
istream_iterator<int> in_iter(cin); //从cin中读取int
istream_iterator<int> eof;  //istream尾后迭代器
while(in_iter != eof)
    vec.push_back(*in_iter++);
ostream_iterator<int> out_list(cout,'\t');
for (auto e : vec){
  *out_list++ = e;  // 可以忽略*,++的影响
}
```

list和forward_list一些特有的操作

```
lst.merge(lst2); //lst合并lst2
lst.merge(...)
lst.remove(val);    // 删除值为val的元素
lst.reverse();  //反转lst中元素顺序
lst.sort()
lst.unique()
lst.slice(args)
```

函数指针

形式：返回类型(* 函数名) (参数);

```
char (*pfun)(int);  //定义了一个函数指针
char glfun(int a){ return ;}    // 定义了一个函数
pfun = glfun;
```

注意：accumulate(a.cbegin(), a.cend(), string("")）；对。accumulate(a.cbegin(), a.cend(), “ ”）；错