链滴

C++ vector 泛型算法笔记

作者：heyang5188

原文链接：https://ld246.com/article/1503200827049

来源网站：链滴

泛型算法前三节

算法是操作迭代器，算法永远不会执行容器的操作，只会在容器中移动元素但永远不会直接添加和删元素。

简单的算法复习：
count 查找vector里面的 数字多少次 返回
auto result = count(vec.begin(),vec.end(),8);

accumulate 求一个vector<int >中的元素之和
int sum = accumulate(vec.begin(),vec.end(),0);
第三个参数说明 加的东西是什么类型，如果vector里面的是string 对象，那么在第三个参数填上""就了
在泛型算法中，参数很重要，需要明确知道参数的作用，还有重载的参数也好多，需要特别注意。
另外，算法的迭代器参数有可能不是同一个容器的。

特殊的迭代器，插入，流，反向

插入迭代器 insert iterator，可以向一个容器中插入数据

```
auto it = front_inserter(vec);//从前面插入，必须要支持push_front
auto it = back_inserter(vec); //从后面插入
auto it = inserter(vec,vec,begin());//从迭代器位置开始插入

*it = val ; //这样就插入了
等于如下操作：
it = vec.insert(it,vec.begin());//插入
it++; //递增it使它指向原来的元素

vector<string> vec{ "happynery","some","fox","school","parsongers for","fox","the","pasenger"
"sanyuan" };
    list<string> lis;
    sort(vec.begin(), vec.end());
    unique_copy(vec.begin(), vec.end(), back_inserter(lis));
    for (auto &m : lis)
        cout << m << endl;
```

流迭代器：绑定在输入或者输出流上的迭代器；

```
istream_iterator<int> in(cin); //从CIN读取int
    istream_iterator<int> eof; //尾部迭代器
    vector<int>vec(in,eof); //从流迭代器初始化vector
    ostream_iterator<int> out_iter(cout, " ");
    for (auto &e : vec)
        *out_iter++ = e; // *和++运算符其实对输出流迭代器不做任何事情，这里是为了使其他的使
保持一致；
    cout << endl;
    可以通过调用 copy 来打印vec的内容；
    copy(vec.begin(),vec.end(),out_iter);
    cout<<endl;

// 使用流迭代器读取一个文本文件，存入一个vector中的string里
```

```
    ifstream in("shader.vs");
    if (!in) {
        cout << "读取文件失败" << endl;
    }
    istream_iterator<string> in_it(in);
    istream_iterator<string> eof;
    //高能预警，高级初始化
    vector<string> vecStr(in_it,eof);
    //版本2，循环push_back
    while(in_it !=  eof){
      vecStr.push_back(*in_it++);
    }
    //华丽的分界线
    for (auto &elem : vecStr)
        cout << elem << endl;

//使用流迭代器，sort copy 从标准输入读取一个整数序列，将其排序，打印不重复的元素
    istream_iterator<int> in_it(cin);
    istream_iterator<int> eof;
    vector<int> vecInt(in_it,eof);
    sort(vecInt.begin(),vecInt.end());
    ostream_iterator<int> out(cout, "\n");
    unique_copy(vecInt.begin(), vecInt.end(), out);

//自己看得懂
int  main(int argc,char**argv) {
    if (argc != 4) {
        cout << "please in put file name out filename :";
        return -1;
    }
    ifstream in(argv[1]);
    istream_iterator<int> in_it(in);
    istream_iterator<int> eof;
    ofstream out1(argv[2]);
    ofstream out2(argv[3]);
    ostream_iterator<int> out_it1(out1, "\n");
    ostream_iterator<int> out_it2(out2, " ");
    while (in_it != eof)
    {
        if (*in_it % 2 == 0)
        {
            *out_it1++ = *in_it++;
        }
        else
        {
            *out_it2++ = *in_it++;
        }
    }
    system("pause");
    return 0;
}
```

反向迭代器：向后移动的迭代器；

移动迭代器：专门移动迭代器的迭代器；

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sstream>
#include <list>
#include <deque>
#include <forward_list>
#include <stack>
#include <algorithm>
#include <functional>
using namespace std;
void elimDups(vector<string> &words) {
    //按字典排序
    sort(words.begin(), words.end());
    //unique 重排输入范围，使得每个单词只出现一次
    //排列在范围的前部，返回指向不重复区域之后的一个位置的迭代器
    auto end_unique = unique(words.begin(), words.end());
    //使用向量操作来删除重复的单词
    words.erase(end_unique, words.end());
}
bool compare(const string &str1, const string &str2) { return str1.size() > str2.size(); }
bool bigsize(const string &str1, string::size_type sz) {
    return str1.size() >= sz;
}
string make_plural(size_t ctr, const string &word, const string &ending) {
    return (ctr > 1) ? word + ending : word;
}
void biggise(vector<string> &words, vector<string>::size_type sz) {
    elimDups(words);
    stable_sort(words.begin(), words.end(), compare);
    auto wc = find_if(words.begin(), words.end(),bind(bigsize,std::placeholders::_1,sz));
    auto count = words.end() - wc;
    cout << count << " " << make_plural(count, "word", "s")
        << "of length" << sz << "or longer" << endl;
    for_each(wc, words.end(),
        [](const string &s) {cout << s << " "; });
        cout << endl;
}
int  main() {

    vector<string> vec{ "happynery","some","fox","school","parsongers for","fox","the","pasen
er","sanyuan" };
    biggise(vec,5);
    system("pause");
    return 0;
}
```