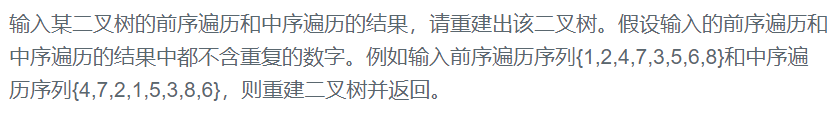
* 第一题
* 题目描述



* 相关知识补充

C++ iterator

1. 概念

迭代器提供对一个容器中的对象的访问方法，并且定义了容器中对象的范围。迭代器就如同一个指针。检查容器内元素并遍历元素的数据类型。每种容器都定义了自己的迭代器类型。

1. 相关操作
2. begin和end操作

如果有元素的话，由begin返回迭代器指向的第一个元素.下列例子：如果vector不空，初始化后，iter即指向钙元素为iver[0]

Vector<int>::iterator iter = iver.begin();

由end操作返回的迭代器指向vector的“末端元素的下一个”由end操作返回的迭代器并不指向vector中任何实际的元素，相反，他只是起一个哨兵的作用，表示我们已经处理完vector中所有元素。

1. vector迭代器的自增和解引用运算

使用\*访问迭代器所指向的元素，解引用操作符返回值是一个左值，因此可以对它进行赋值来改变他的值。\*iter=2。

迭代器的自增操作和int型对象的自增操作类似。相当于“向前移动一个位置“””

1. 迭代器编写循环

vector<int>::iterator it;

for(it = in.begin();it != in.end();it++){

\*it = 0;//将每一个元素置为0

}

1. iterator的算术操作：

iterator除了进行++,–操作，可以将iter+n,iter-n赋给一个新的iteraor对象。还可以使用一个iterator减去另外一个iterator.

const vector<int>::iterator newiter = ivec.begin();

vector<int>::iterator newiter2 = ivec.end();

cout<<"\n"<<newiter2 - newiter;

3、vector<int>::const\_iterator 和 const vector<int>::iterator的区别

const vector<int>::iterator newiter=ivec.begin();

\*newiter=11; //可以修改指向容器的元素

newiter++; //迭代器本身不能被修改

1. **Iterator（迭代器）模式 具体看博文**

<https://blog.csdn.net/qq_23845067/article/details/51286852?ops_request_misc=%7B%22request%5Fid%22%3A%22158282279019725256716232%22%2C%22scm%22%3A%2220140713.130056874..%22%7D&request_id=158282279019725256716232&biz_id=0&utm_source=distribute.pc_search_result.none-task>

* 题目求解
* 代码实现
* 第二题
* 题目描述
* 相关知识补充
* 题目求解
* 代码实现