项目说明文档

数据结构课程设计

——两个有序链表的交集

作 者 姓 名： 沈星宇

学 号： 1951576

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

1. **分析**
2. **应用背景**

对两个集合取交集是数学里面常用的基本操作之一，在交集的基础之上可以延伸出更多数学内容。本项目通过链表构造实现取交集的目标。

本项目对两个非降序链表序列list1和list2进行取交集的操作，设计函数构造出二者公共的部分resultList。

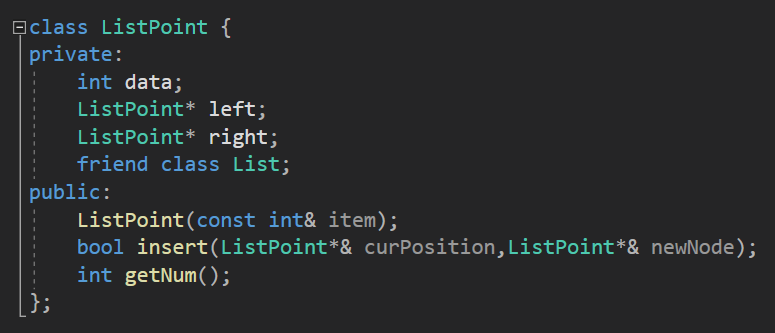
而对于用户输入的非升序链表会进行提示报错，要求重新输入，并且要能够对用户的不合法输入（例如字母等）进行报错处理，直到用户输入符合要求的序列为止。

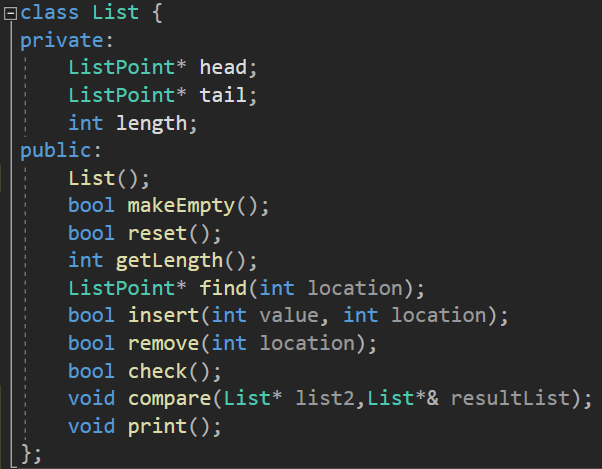
1. **项目功能要求**

1.输入说明：输入分2行，分别在每行给出由若干个正整数构成的非降序序列，用-1表示序列的结尾（-1不属于这个序列）。数字用空格间隔。

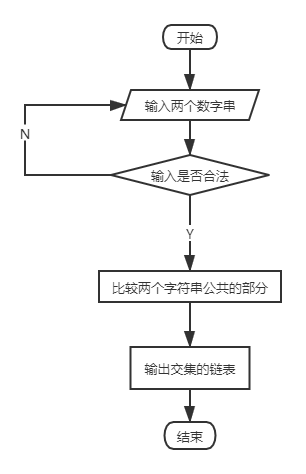
2.输出说明：在一行中输出两个输入序列的交集序列，数字间用空格分开，结尾不能有多余空格；若新链表为空，输出NULL。

1. **设计**
2. **数据结构设计**





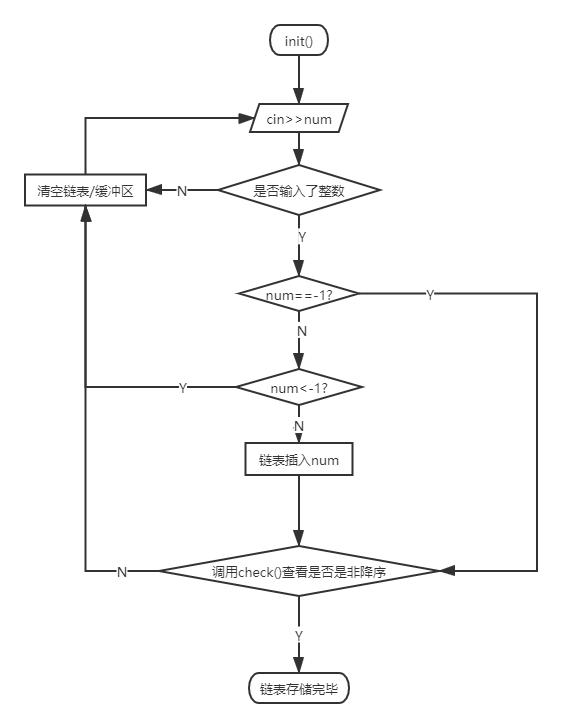
1. ListPoint是结点的结构，内部包含了存储的数字data和指针left、right，为了锻炼写双链表的能力才写成了双链表，其实本项目单链表即可实现
2. List是链表的结构，里面包含了常规的操作的函数，为了使List能够访问ListPoint的成员，故将其在ListPoint类里写作了友元。
3. **程序流程设计**



1. 首先对链表进行初始化，初始输入一些数据，此处对合法性进行判断，如果输入不合法则会请求重新输入；输入合法数据后会告诉你输入正确
2. 调用比较函数compare()，同时生成新的链表，将新的链表打印输出
3. 最后释放动态分配的内存
4. **实现**
5. **main()函数的内部逻辑**
6. main()函数会首先用init()函数对链表进行初始化，在init()函数内部对输入合法性进行检查。
7. 接着调用list1的compare函数，传入的引用resultPointer是指向resultList的指针，该操作使得调用完比较函数后能使resultList将交集的部分存储起来。
8. 调用resultList的print()函数将结果打印出来
9. 释放链表的内存



1. **init()的实现**



1. 对于输入的一串数字，用cin>>num对单独的数字进行处理，首先用cin.fail()检测是否输入了不合法的类型
2. 然后判断输入的数字是否是-1，读取到-1则停止插入；如果输入的整数是除-1以外的其他负数，那么请求从头重新输入链表；输入合法时则将其用尾插法插入链表
3. 用check()函数查看是否输入的序列为非降序，如果是不和要求，要请求重新输入新的数据



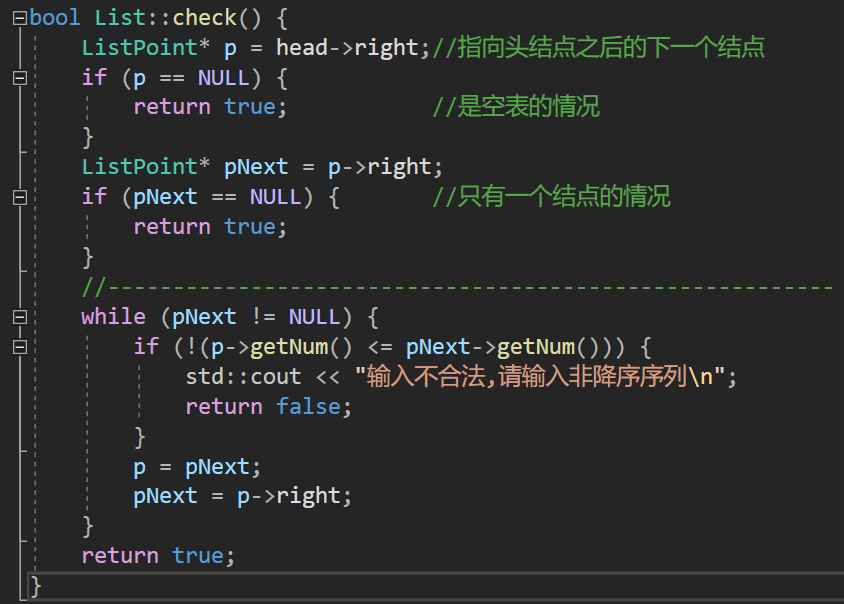
1. **List::check()的实现**

用于检测输入的序列是否的非降序的序列。

首先考虑链表是空表的情况，即头结点的right指向的是NULL，空表满足非降序的要求所以返回true。

然后用两个ListPoint\*类型的指针p和pNext来读取p指向的结点和p指向的结点的下一个结点的数据，进行比较，如果非降序则直接返回false。

循环终止的条件是pNext指针指向了NULL，即比较到了链表的结尾。



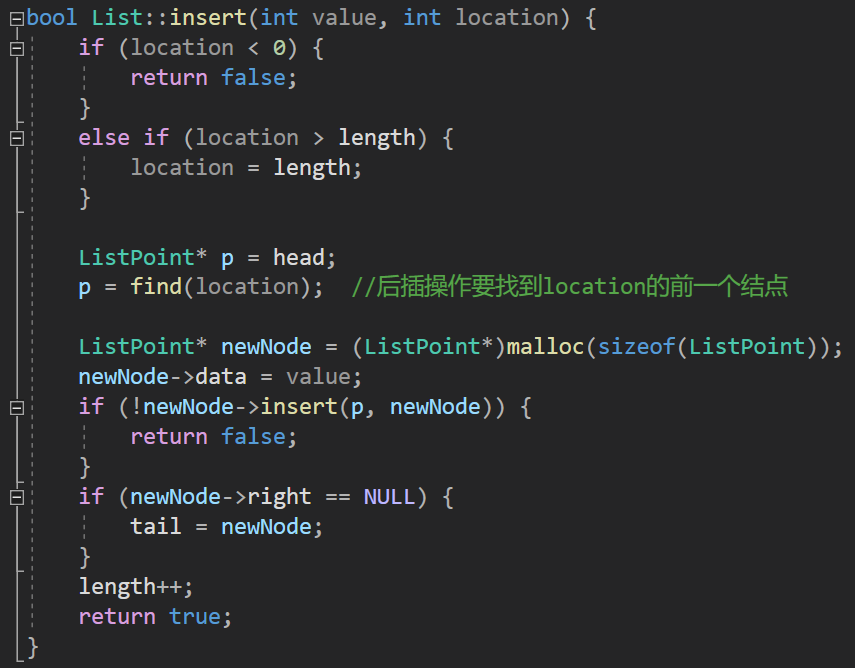
1. **List::insert()的实现**

insert()函数其实是按位后插，即在location的后面插入一个结点。

（1） 首先会对location的合法性进行检测（虽然本项目按照顺序插入在队尾不会涉及到这个内容，但是为了安全性和整体性仍然进行检测）。禁止在第0位及第0位之前插入结点；在超出链表的最后一个结点的时候在最后一个结点插入。

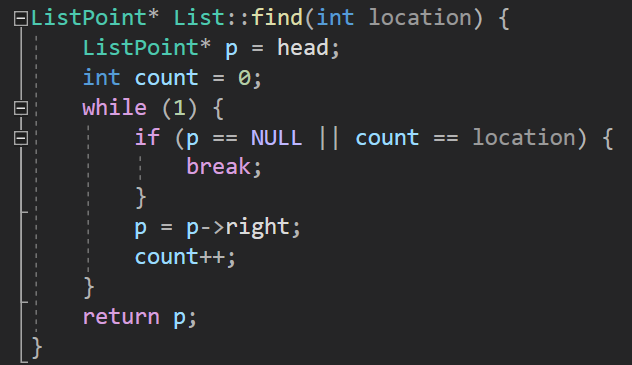
（2）然后调用find()函数去查找到location位置的结点，用malloc为新的结点动态分配一个内存，修改新节点newNode的data并且调用newNode的insert()函数将它的指针链接上去。

（3）调整尾指针，使链表的长度增加。



1. **List::find()的实现**

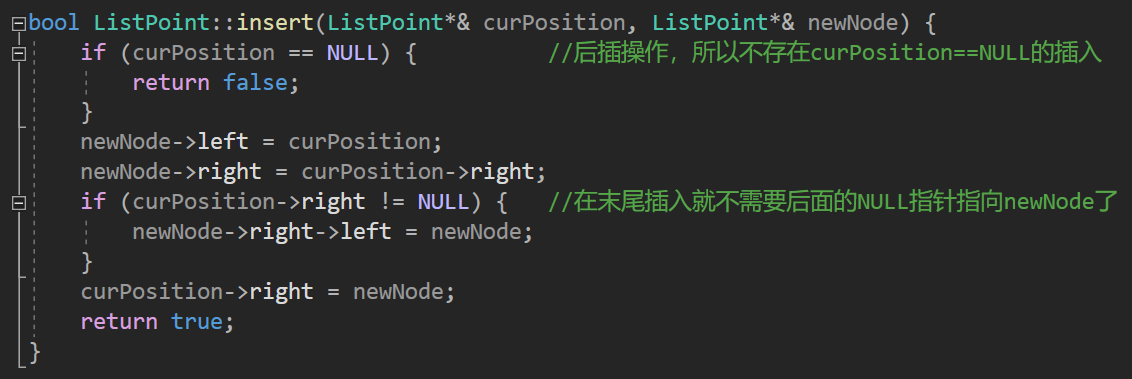
按位查找，用一个指针从第一个结点指向最后一个结点，该函数返回所寻找位置的结点指针。



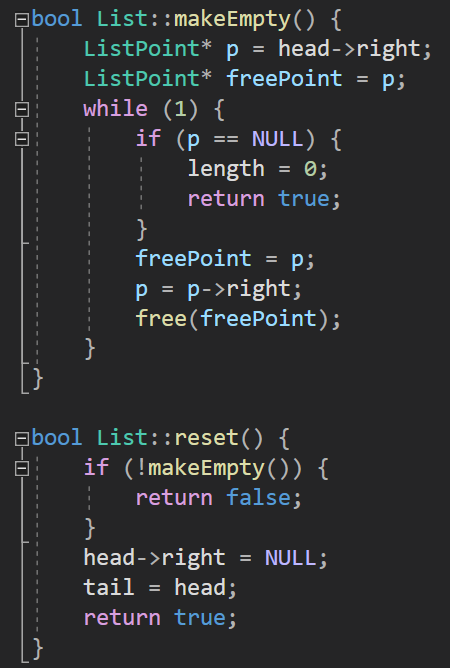
1. **ListPoint::insert()的实现**

后插操作要不存在curPositon==NULL的情况，即不在空指针后面插入内容。

把结点用指针链接起来。因为最后一个结点的right指针指向的是NULL，如果是在链表的末尾插入结点，那么不需要将newNode之后的结点（其实是空指针）的left指针指向newNode，因为空指针不存在left指针。



1. **List::reset()和List::makeEmpty()的实现**

（1）在程序接收到不合法的输入时，程序会将链表清空，并且请求用户重新输入数字序列，直到输入的序列合法为止。实现这一部分功能的是reset()和makeEmpty()函数。

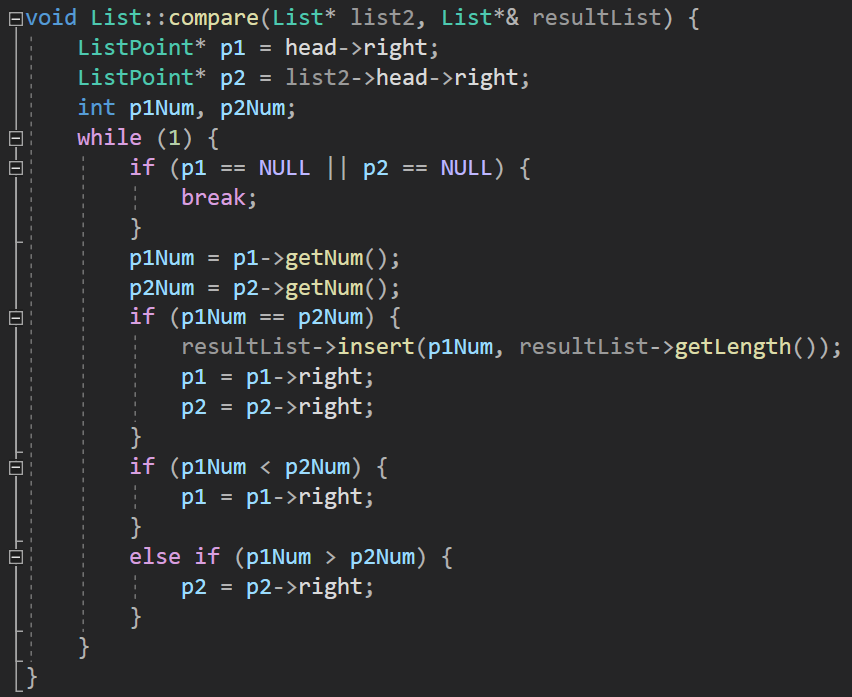
（2）makeEmpty()是释放内存的作用，在main()函数的末尾、程序结束时也会被调用，防止碎片化内存块的出现。它通过指针p去访问链表的每一个结点，将每一个结点的内存释放，然后重置链表的长度length = 0。

（3）对于reset()函数，用于清空重置链表，内部会先调用makeEmpty()释放内存，然后将头指针的right指针指向NULL（防止访问脏数据），将尾指针指向头结点。

1. **List::compare()的实现**

因为序列是非降序的，那么用两个指针指向两个链表的结点，比较结点的data值，将共有的data存入新的链表resultList即可。

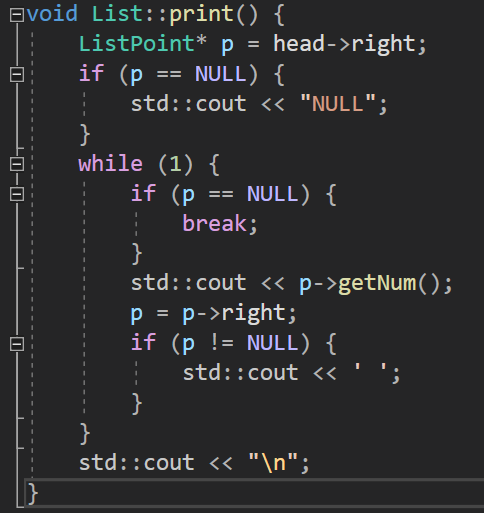
循环停止的条件是任意一个指针到达了链表的结尾。



1. **List::print()的实现**

同样用指针p去访问结果链表的每个结点，输出内部的值。

注意在最后一个元素的后面不能多空格。



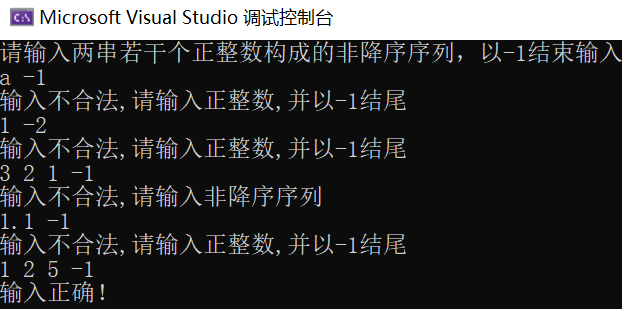
1. **测试**

1、 **初始化表格的合法性检测**

（1） 输入非-1的负数、浮点数和字符要求重新输入

（2） 输入非降序序列要求重新输入

（3） 输入人数为非降序正整数序列则进行下一步

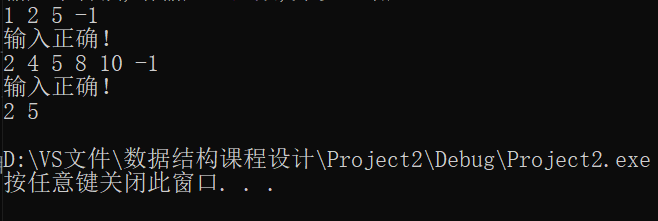


**2、一般情况**

（1） 两个序列为一般的非空非降序序列

（2） 两个序列有共有元素

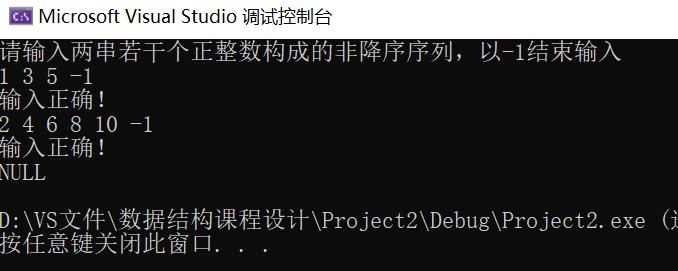
（3） 输出两个序列的交集



**3、 交集为空的情况**

（1）两个序列为一般的非空非降序序列

（2）两个序列有共有元素

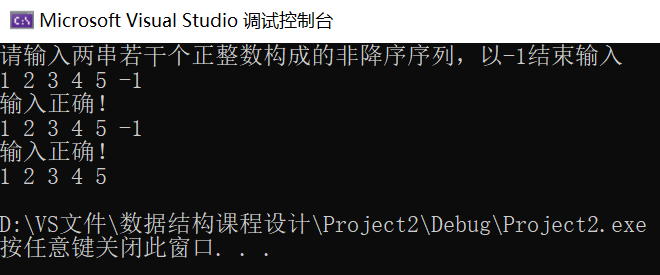
（3）输出NULL

**4、 完全相交的情况**

（1）两个序列为一般的非空非降序序列

（2）两个序列完全相交

（3）输出交集

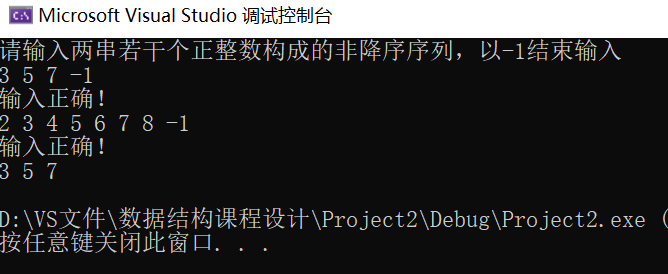


**5、其中一个序列完全属于交集的情况**

（1）两个序列为一般的非空非降序序列

（2）其中一个序列完全属于交集

（3）输出交集



**6、其中一个序列为空的情况**

（1）其中一个序列为空

（2）输出NULL

