项目说明文档

数据结构课程设计

——两个有序链表序列的交集

作 者 姓 名： 崔鑫宇

学 号： 1853444

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

## 1.1 背景分析

已知有两个非降序链表序列S1和S2，现要构造出S1、S2这两个链表的交集新链表S3.

## 1.2 功能分析

由项目要求可知，本程序需要可以在两行输入两个由正整数构成的非降序序列，所以该程序必须要具有输入功能。此外，本程序还要求输出功能，以输出序列交集这一新链表。

综上所述，该项目至少应该具有输入、输出功能。

# 2 设计

## 2.1 数据结构设计

如题目——两个有序链表的交集，可以看出本项目需要使用到的是链表这一数据结构。

## 2.2 类结构设计

链表数据结构一般需要设计两个类，分别是链表类（LinkList）和链表结点类（LNode）。本程序为了方便处理数据，特别采用了Node struct来表示节点类，这样使得链表节点类可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类（LNode）**

T data;//节点内所存数据

LNode\* next=NULL;//定义下一个节点

**链表类（LinkList）**

public:

LinkList();

~LinkList() = default;

LinkList<T>& operator=(const LinkList<T>& l);

void append(T d);//添加数据

void print();//打印链表

bool isEmpty() { return head->next == NULL; }//判断链表是否为空

bool isOrdered();//判断链表内所存数据是否为非降序序列

LNode<T>\* FindHead() const { return head; }//将链表指针条回到头节点

int size();//返回链表大小

T top() { return head->data; } //返回头元素

private:

LNode<T>\* head;//头节点

## int length = 0;//长度

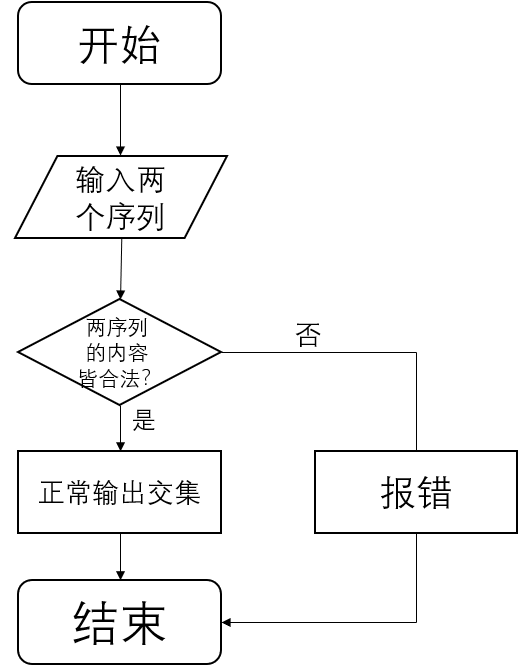
## 

## 2.4 系统设计

系统首先要求用户先后分别输入两段以-1结尾的序列，然后调用ManageList()函数处理两个列表，如果两个序列为非降序正整数序列，则执行FindEqualValue()函数寻找队列间的交集。当其中有一个链表处理完后，程序结束。

# 3 实现

### 3.1 总体系统流程图



### 

### 3.6.2 总体系统核心代码

### ①处理链表序列

//处理链表序列

void ManageList(LinkList<int> list, string str){

stringstream ss(str);

string s;

while (ss >> s) {//当提取到-1时结束提取工作

if (stoi(s) == -1)

break;

else//否则正常进行提取工作

list.append(stoi(s));

}

②找到两个链表中相等的值

//找到两个链表中的相等的值

bool FindEqualValue(LinkList<int> l1, LinkList<int> l2){

int num = 0;

LNode<int>\* p1 = l1.FindHead();

LNode<int>\* p2 = l2.FindHead();

LNode<int>\* pre = p1;

p1 = p1->next;

p2 = p2->next;

if (p1 == NULL || p2 == NULL) {

return false;

}

else {

while (p1 != NULL && p2 != NULL) {

if (p1->data > p2->data) {

//如果p2当前元素小于p1当前元素则让p2指向下一个元素

p2 = p2->next;

}

else if (p1->data < p2->data) {

//如果p1当前元素小于p2当前元素则让p1指向下一个元素

pre->next = p1->next;

p1 = p1->next;

}

else{

p1 = p1->next;

pre = pre->next;

p2 = p2->next;

num++;

}

}

if (p2 == NULL) {

pre->next = NULL;

}

if (num == 0) {

return false;

}

else {

return true;

}

}

③主程序

int main() {

LinkList<int> list1, list2;

string s1, s2;

cout << "请输入第一串序列（以-1结尾！）：";

getline(cin, s1);//输入第一个链表

cout << "请输入第二串序列（以-1结尾！）：";

getline(cin, s2);//输入第二个链表

ManageList(list1, s1);//处理第一个链表

ManageList(list2, s2);//处理第二个链表

//确认两个链表为非降序链表

if (!(list1.isOrdered() && list2.isOrdered())) {

cerr << "输入的序列不符合要求,请检查输入！";

exit(-1);

}

//确认两个链表是否为正整数序列（除标志位-1）

else if ((!isNumber(s1[0])) || (!isNumber(s2[0]))) {

cerr << "请输入正整数序列(除标志位-1)!";

exit(-1);

}

else {

if (FindEqualValue(list1, list2)) {

list1.print();

}

else {

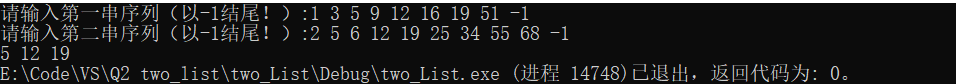
cout << "NULL";

}

}

}

### 3.6.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

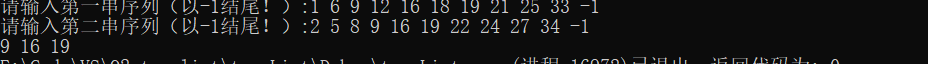
4.1.1 正常数据测试

①**测试用例：**Group1：1 6 9 12 16 18 19 21 25 33 -1

Group2：2 5 8 9 16 19 22 24 27 34 -1

**预期结果：**9 16 19

**实验结果：**

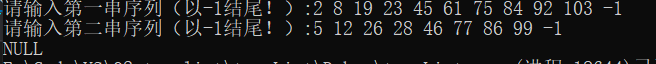
****

②**测试用例：**Group1：2 8 19 23 45 61 75 84 92 103 -1

Group2：5 12 26 28 46 77 86 99 -1

**预期结果：**NULL

**实验结果：**

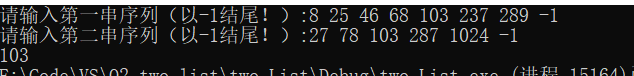


③**测试用例：**Group1：8 25 46 68 103 237 289 -1

Group2：27 78 103 287 1024 -1

**预期结果：**103

**实验结果：**

****

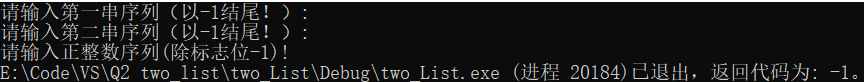
## 4.2 不良输入测试

### 4.2.1 初始化无输入数据

**测试用例：**初始无输入数据

**预期结果：**给出错误提示，程序强行退出。

**实验结果：**



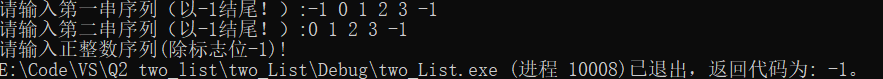
### 4.2.2 首元素为负数

**测试用例：**Group1：-1 0 1 2 3 -1

Group2：0 1 2 3 -1

**预期结果：**给出错误提示，程序强行退出。

**实验结果：**



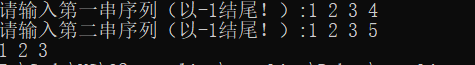
### 4.2.3 没有尾元素标志位-1

**测试用例：**Group1：1 2 3 4

Group2：1 2 3 5

**预期结果：**程序正常运行，无影响。

**实验结果：**



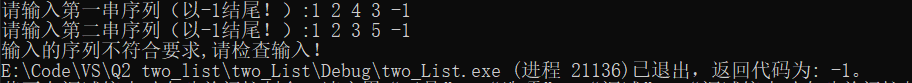
4.2.4 输入序列不为非降序序列

**测试用例：**Group1：1 2 4 3 -1

Group2：1 2 3 5 -1

**预期结果：**程序正常运行，无影响。

**实验结果：**

****