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： 伊啸

学 号： 1951220

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc59472256)

[1.1 项目介绍 1](#_Toc59472257)

[1.2 基本思路 1](#_Toc59472258)

[2 设计 2](#_Toc59472259)

[2.1 类结构设计 2](#_Toc59472260)

[2.2 成员与操作设计 2](#_Toc59472261)

[3 实现 3](#_Toc59472269)

[3.1 输入函数的实现 3](#_Toc59472270)

[3.1.1 输入功能核心代码 3](#_Toc59472271)

[3.1.2 输入功能截屏示例 4](#_Toc59472272)

[3.2 输出功能的实现 4](#_Toc59472273)

[3.2.1 输出功能核心代码 4](#_Toc59472274)

[3.2.2 输出功能截屏示例 4](#_Toc59472285)

[3.3 求两链表交集 5](#_Toc59472286)

[3.3.1 求交集过程流程图 5](#_Toc59472287)

[3.3.2 求交集功能核心代码 6](#_Toc59472288)

[3.3.3 求交集功能截图示例 7](#_Toc59472289)

[3.4 总体系统的实现 8](#_Toc59472290)

[3.4.1 总体系统流程图 8](#_Toc59472291)

[3.4.2 总体系统核心代码 10](#_Toc59472292)

[3.4.3 总体系统截屏示例 10](#_Toc59472293)

[4 测试 10](#_Toc59472294)

[4.1 功能测试 10](#_Toc59472295)

[4.2 边界测试 12](#_Toc59472296)

[4.2.1 输入其他回答，而非y或者n 12](#_Toc59472297)

[4.2.2 输入的数据中含有负数 13](#_Toc59472298)

[4.2.3 输入的数据不为正整数，为浮点数或者字符 13](#_Toc59472299)

# 1 分析

## 1.1 项目介绍

已知两个非降序链表序列S1和S2，设计函数构造出S1和S2的交集新链表S3。

（**采用链表实现**）

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。

3. 测试用例：

序号 输入 输出 说明

1 1 2 5 -1

2 4 5 8 10 -1 2 5 一般情况

2 1 3 5 -1

2 4 6 8 10 -1 NULL 交集为空的情况

3 1 2 3 4 5 -1

1 2 3 4 5 -1 1 2 3 4 5 完全相交的情况

4 3 5 7 -1

2 3 4 5 6 7 8 -1 3 5 7 其中一个序列完全属于交集的情况

5 -1

10 100 1000 -1 NULL 其中一个序列为空的情况

## 1.2 基本思路

建立两个链表A、B，将输入的数据逐一插入链表中，从头开始比较链表的各个数据，比如a是A中的元素，b是B中的元素，若ab相等则将数据插入到一个新链表中，此时A、B链表的指针都指向下一个节点；若a>b,则B链表的指针指向下一个节点；若a<b,则A链表的指针指向下一个节点。当某一个链表到达终点时就终止程序，即此时的节点数据为-1。

# 2 设计

## 2.1 类结构设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（LinkNode）与链表类（LinkList），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为方便处理，本系统采用struct描述链表结点类（LinkNode），且使LinkNode中的数据和函数都为公有的，这样使得链表结点类（LinkList）可以访问链表结点。

## 2.2 成员与操作设计

**链表结点类（LinkNode）**

**公有成员：**

int data;//结点中的数据

LinkNode\* link;//指向下一个结点的指针域

**公有操作：**

LinkNode(int num): data(num),link(NULL){ }//具有默认参数的构造函数

~LinkNode() { }//析构函数

**链表类（LinkList）**

**私有成员：**

LinkNode\* head;//头指针

**公有操作：**

## LinkList()；//构造函数

## ~LinkList() { delete[]head; } //析构函数

## LinkNode\* rhead() { return head; }//返回指向头结点的指针

## LinkList\* Intersection(LinkList\*, LinkList\*);

## //求两个链表交集的操作

## void input()；//输入函数

## void display()； //输出函数

# 3 实现

## 3.1 输入函数的实现

## 3.1.1 输入功能核心代码

while ( num != -1) {

if (num <= 0) **//判断输入的数据是否存在负数**

judge\_negative = true;

LinkNode\* q = new LinkNode(num);**//创建一个数据为num的新结点**

p->link = q;

p = p->link;**//将该结点插入链表中**

while (1) {

cin >> num;**//判断输入的是否是一个整数**

if (cin.good() == false) {

cout << "Input error!" << endl;

while (getchar() != '\n')

;

cin.clear();**//清空输入缓存区**

cout << "Please enter this list again:";

}

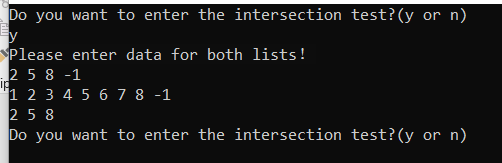
else

break;

}

}

## 3.1.2 输入功能截屏示例



## 3.2 输出功能的实现

### 3.2.1 输出功能核心代码

### LinkNode\* p = head;

### if (p == NULL)

### cout << "NULL" << endl;

### while (p != NULL && p->link != NULL) {

### //实现最后一个输出后没有空格

### cout << p->data << " ";

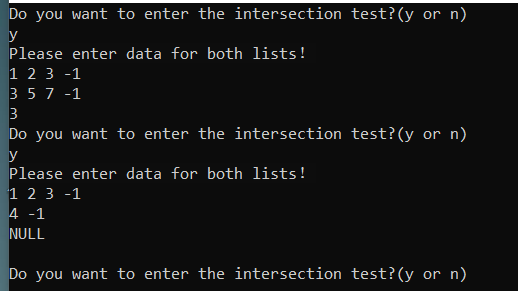
### p = p->link;

### }

### if (p && p->link == NULL)

### cout << p->data;

### 3.2.2 输出功能截屏示例



## 3.3 求两链表交集

### 3.3.1 求交集过程流程图



### 3.3.2 求交集功能核心代码

while (p != NULL && q != NULL) {

if (p->data > q->data)

q = q->link;

else if (q->data == p->data) {

LinkNode\* newnode = new LinkNode(p->data);

//使用p指向的数据创建一个新节点，并将r链接指向它，以将该节点放入列表中

r->link = newnode;

r = r->link;

p = p->link;

q = q->link;

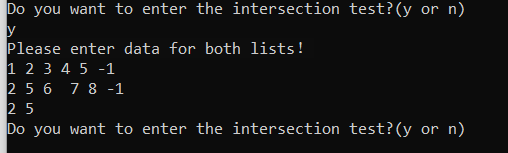
}

else

p = p->link;

}

### 3.3.3 求交集功能截图示例



## 3.4 总体系统的实现

### 3.4.1 总体系统流程图



### 3.4.2 总体系统核心代码

LinkList\* List1 = new LinkList;

List1->input(); //输入第一个链表数据

while (judge\_negative) {

//判断是否输入了负数

cout << "Illegal input, negative number exists, please re-enter this list!" << endl;

judge\_negative = false;

List1->input();

}

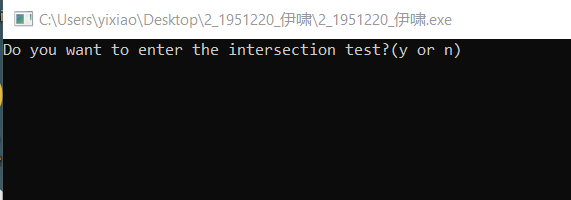
LinkList\* inter\_list = List1->Intersection(List1, List2);

//求两链表交集

inter\_list->display();

cout << endl;

### 3.4.3 总体系统截屏示例



# 4 测试

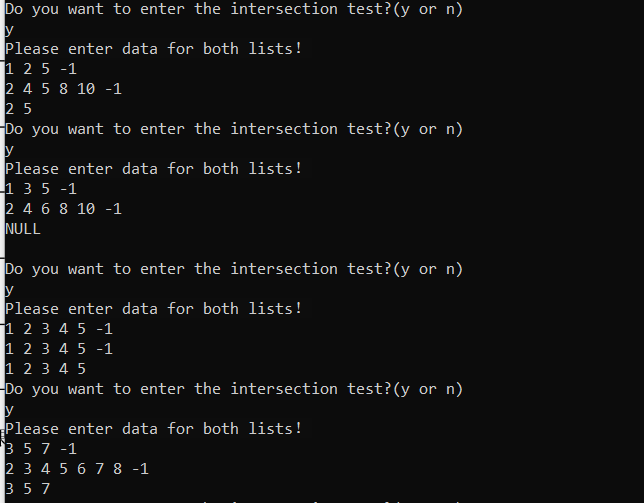
## 4.1 功能测试

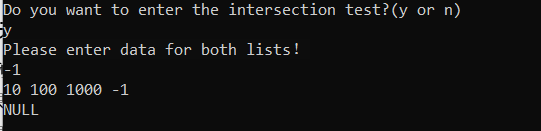
求交集功能测试：

**测试用例：**

****

**实验结果：**

****

****

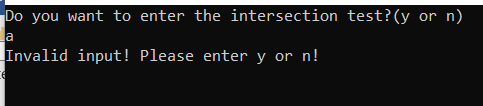
## 4.2 边界测试

### 4.2.1 输入其他回答，而非y或者n

**测试用例：**输入a

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

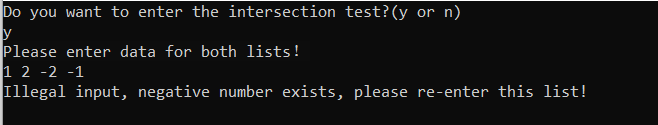


### 4.2.2 输入的数据中含有负数

**测试用例：**1 2 -2 -1

**预期结果：**程序正常运行，不崩溃。

**实验结果：**



### 4.2.3 输入的数据不为正整数，为浮点数或者字符

**测试用例：输入含有浮点数或者字符的数据**

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

