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： 何慧琳

学 号： 2152343

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](#_Toc495668153)

[1.1 背景分析 1](#_Toc495668154)

[1.2 功能分析 1](#_Toc495668155)

[2 设计 1](#_Toc495668156)

[2.1 数据结构设计 1](#_Toc495668157)

[2.2 类结构设计 1](#_Toc495668158)

[2.3 成员与操作设计 2](#_Toc495668159)

[2.4 系统设计 2](#_Toc495668160)

[3 实现 3](#_Toc495668161)

3.1 求交集功能的实现……………………………………………………………..3

[3.1.1 求交集功能流程图 3](file:///\\10.60.41.1\Publicfiles\CourseDocuments_课程文档\数据结构课程设计\文档模板.docx#_Toc495668163)

[3.1.2 求交集功能核心代码 3](file:///\\10.60.41.1\Publicfiles\CourseDocuments_课程文档\数据结构课程设计\文档模板.docx#_Toc495668164)

[3.1.3 求交集功能截屏示例 4](file:///\\10.60.41.1\Publicfiles\CourseDocuments_课程文档\数据结构课程设计\文档模板.docx#_Toc495668165)

[3.2 总体系统的实现 5](file:///\\10.60.41.1\Publicfiles\CourseDocuments_课程文档\数据结构课程设计\文档模板.docx#_Toc495668166)

[3.2.1 总体系统流程图 5](file:///\\10.60.41.1\Publicfiles\CourseDocuments_课程文档\数据结构课程设计\文档模板.docx#_Toc495668167)

[3.2.2 总体系统核心代码 5](file:///\\10.60.41.1\Publicfiles\CourseDocuments_课程文档\数据结构课程设计\文档模板.docx#_Toc495668168)

[3.2.3 总体系统截屏示例 6](file:///\\10.60.41.1\Publicfiles\CourseDocuments_课程文档\数据结构课程设计\文档模板.docx#_Toc495668169)

[4 测试 6](#_Toc495668186)

[4.1 功能测试 6](#_Toc495668187)

[4.2 边界测试 7](#_Toc495668193)

[4.3 出错测试 8](#_Toc495668197)

[4.3.1 输入序列错误 8](#_Toc495668198)

[4.3.2 选择继续与否的输入错误 8](#_Toc495668199)

# 1 分析

## 1.1 背景分析

求交集是对数据进行分析处理的一种常见的重要操作，对有序序列求交集是将求交集问题简化后的问题，对交集算法的掌握有重要意义。

## 1.2 功能分析

在此项目中为了简化问题，直击交集算法的关键，我们规定输入的两个序列均是非降序的。该程序能读入两个以-1结尾的非降序序列并打印出他们的交集，若交集为空则打印NULL，每结束一次求交集都将给出“是否继续”的提示，因此也可以实现循环求多对序列交集的操作。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求对有序序列求交集操作，而链表进行求交集操作十分简便，因此考虑使用链表数据结构。

## 2.2 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（Listnode）与链表类（List），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，本系统采用继承关系，即将List声明为Listnode的友元，这样List就可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类（Listnode）**

**私有成员：**

int data;//结点值

Listnode\* link;//指向当前节点的下一个结点

**公有操作：**

Listnode() :link(NULL) {};//链表结点构造函数

Listnode(int i) :data(i), link(NULL) {};//链表结点构造函数

**链表类（LinkList）**

**私有成员：**

Listnode\* first;//头指针

**公有操作：**

List() { first = new Listnode(); };//链表构造函数

~List();//析构函数

void read();//将输入数据读入链表的函数

void AND(List &list2);//取交集函数并输出

void Makeempty();//删除除表头外的所有结点

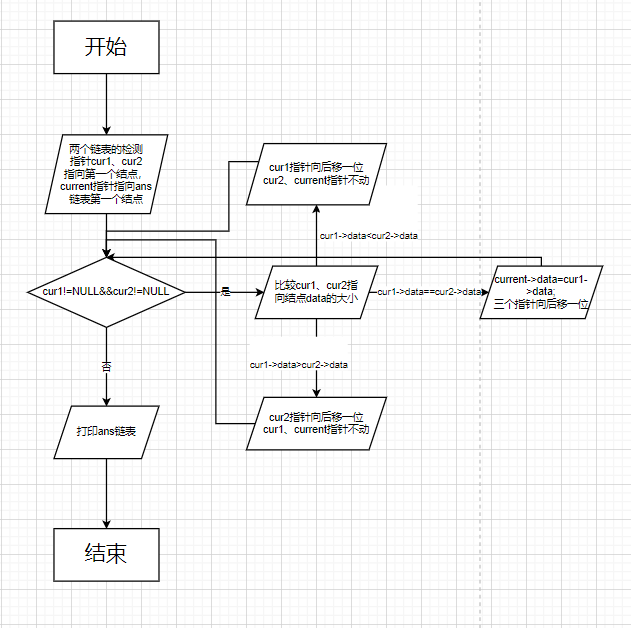
## 2.4 系统设计

系统首先在main（）函数中创建链表S1、S2，然后进行一个while循环，该循环中将通过调用List类中的函数，完成对S1、S2的输入数据、求交集并打印交集、求交集完成后的清空链表工作，最后根据用户所输入的y/n选择是否继续该循环。

# 3 实现

## 3.1 求交集功能的实现

### 3.1.1 求交集功能流程图



### 3.1.2 求交集功能核心代码

void List::AND(List& list2)

{//取交集函数并输出

List ans;//存储输出序列

Listnode\* current = ans.first;//指向ans链表的指针

Listnode\* cur1=first->link,\* cur2=list2.first->link;//this链表和list2链表的检测指针

while (cur1 != NULL && cur2 != NULL) {//两个序列均未读完

if (cur1->data == cur2->data) {//一个数在两个链表中都出现则存入输出链表

current->link = new Listnode(cur1->data);

current = current-> link;//ans指针向后移一位

cur1 = cur1->link;//cur1指针向后移一位

cur2 = cur2->link;//cur2指针向后移一位

}

else if (cur1->data < cur2->data) {//cur1指向的数小于cur2指向的数，cur1指针移一位cur2不动

cur1 = cur1->link;

}

else if (cur1->data > cur2->data) //cur1指向的数小于cur2指向的数，cur2指针移一位cur1不动

cur2 = cur2->link;

}

cout << "交集S3：";

current = ans.first->link;//current再次指向第一个结点

if (current == NULL) {//ans为空表

cout << "NULL" << endl;

}

while (current != NULL) {

cout << current->data;

if ((current = current->link) != NULL)//current指向下一个结点，下一个结点非空则输出一个空格

cout << " ";

else

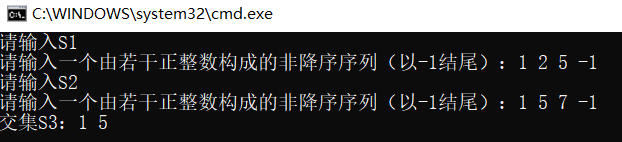
cout << endl;

}

cout << endl;

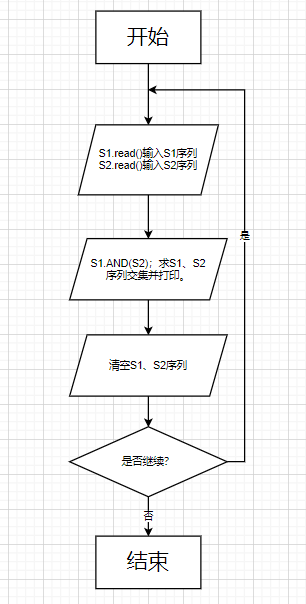
}

### 3.1.3 求交集功能截屏示例



## 3.1 总体系统的实现

### 3.6.1 总体系统流程图



### 3.6.2 总体系统核心代码

while (1) {

cout << "请输入S1" << endl;

S1.read();

cout << "请输入S2" << endl;

S2.read();

S1.AND(S2);

cout << "是否继续？（y/n）";

char go\_on;

cin >> go\_on;

while (go\_on != 'y' && go\_on != 'n') {//输入错误

cout << "输入错误，请重新输入（y/n）";

cin.ignore(100, '\n');

cin.clear();

cin >> go\_on;

}

if (go\_on == 'n')

break;

else {//继续则置空S1、S2

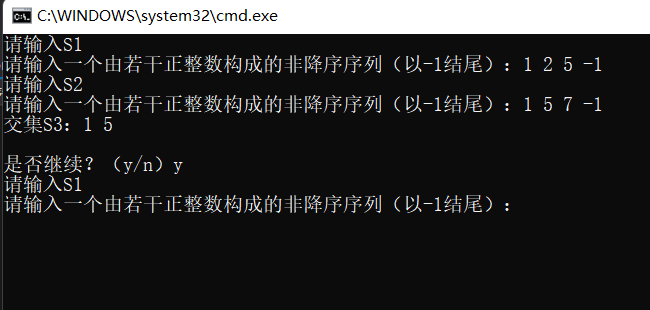
S1.Makeempty();

S2.Makeempty();

}

### }

### 3.6.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

**测试用例**：1 2 5 -1

1 5 7 -1

y

1 2 3 4 5 -1

1 2 3 4 5 -1

y

1 3 -1

2 4 -1

n

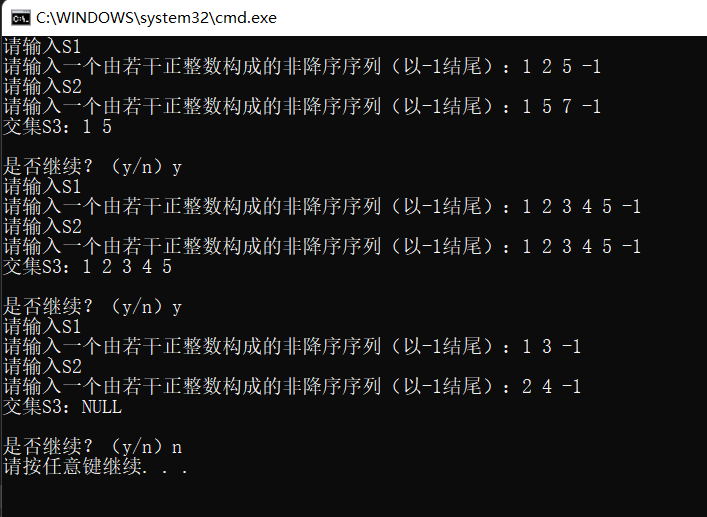
**预期结果**：

1 5

1 2 3 4 5

NULL

**实验结果**



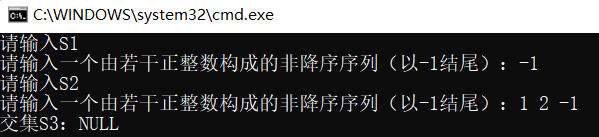
## 4.2 边界测试

**测试用例：**-1

1 2 -1

**预期结果：**程序运行正常不崩溃，输出交集为NULL。

**实验结果：**



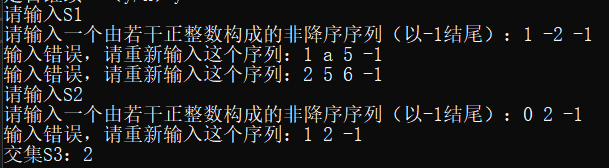
## 4.3 出错测试

### 4.3.1 序列数据输入错误

**测试用例：**输入考生人数为负数或非数

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

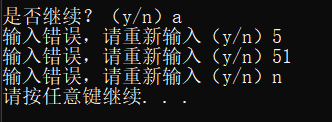
****

### 4.3.2 是否继续的选择符输入错误

**测试用例：**输入y/n错误

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

****