项目说明文档

数据结构课程设计

——两有序链表序列交集

作 者 姓 名： 罗佳瑞

学 号： 1952528

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](#_Toc16091)

[1.1 背景分析 1](#_Toc24450)

[1.2 功能分析 1](#_Toc17655)

[2 设计 2](#_Toc1478)

[2.1 数据结构设计 2](#_Toc22910)

[2.2 类结构设计 2](#_Toc7900)

[2.3 成员与操作设计 2](#_Toc17510)

[2.4 系统设计 3](#_Toc3404)

[3 实现 4](#_Toc30395)

[3.1 创建功能的实现 4](#_Toc23544)

[3.1.1 创建功能流程图 4](#_Toc15833)

[3.1.2 创建功能核心代码 4](#_Toc16602)

[3.2 比较功能的实现 6](#_Toc26005)

[3.2.1 比较功能流程图 6](#_Toc14109)

[3.2.2 比较功能核心代码 7](#_Toc26767)

[3.3 打印功能的实现 8](#_Toc29325)

[3.3.1 打印功能流程图 8](#_Toc2118)

[3.3.2 打印功能核心代码 8](#_Toc16367)

[3.4 总体系统的实现 9](#_Toc12969)

[3.4.1 总体系统流程图 9](#_Toc16585)

[3.4.2 总体系统核心代码 9](#_Toc2353)

[3.4.3 总体系统截屏示例 10](#_Toc9876)

[4 测试 11](#_Toc15817)

[4.1 功能测试 11](#_Toc28797)

[4.1.1 一般情况 11](#_Toc23576)

[4.1.2 交集为空 11](#_Toc2070)

[4.1.3 完全相交 12](#_Toc2527)

[4.1.4 一个序列完全属于交集 12](#_Toc15528)

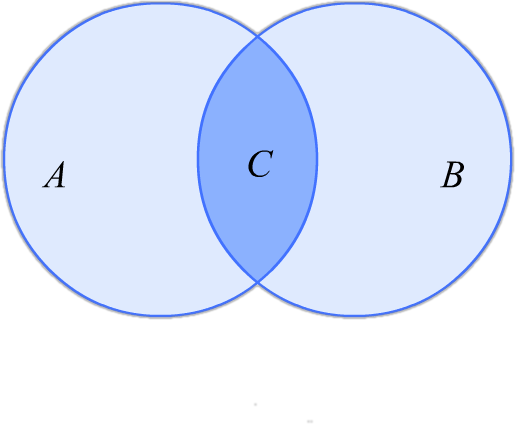
[4.1.5序列为空 13](#_Toc24463)

# 1 分析

## 1.1 背景分析

集合论中，设A，B是两个[集合](https://baike.baidu.com/item/%E9%9B%86%E5%90%88/2908117" \t "https://baike.baidu.com/item/%E4%BA%A4%E9%9B%86/_blank)，由所有属于集合A且属于集合B的[元素](https://baike.baidu.com/item/%E5%85%83%E7%B4%A0/9563210" \t "https://baike.baidu.com/item/%E4%BA%A4%E9%9B%86/_blank)所组成的集合，叫做集合A与集合B的交集（intersection），记作A∩B。

在处理大量数据时，使用计算机求取两非降序序列集合方面，具有人工所无法比拟的优势。因此，编写一个程序自动求取交集具有十分重要的意义。



## 1.2 功能分析

作为一个最简易的求取交集程序，其至少应该具有输入两序列、比较两序列、输出公有元素组成交集等功能。

其中，输入两序列：由若干个正整数构成两个非降序序列，用空格间隔数字输入，-1表示序列的结尾。

比较两序列：将两序列起点对齐，各用一指针依次比较对应元素的大小，若两元素相等将该元素放入结果序列，将两链指针均向后移一位；否则将较小元素所在链的指针后移，直至任意一链结尾。

输出公有元素组成交集：依次输出两个序列的交集序列中的元素，数字间用空格分开；若新链表为空，输出NULL。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求序列的链入、遍历等操作，且其元素个数未知。而链表进行顺序遍历等操作十分简便，且对于动态元素数的存储有着很大优势，因此考虑使用链表数据结构。同时，为了实现简易，在第一个结点之前附加一个头结点，这样就使得链入、取出头结点与处理其他结点方法相同，使得程序更为简洁；在链表尾加入一尾指针，使得链入、取出结点更为方便，避免了遍历。

## 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（Node）与链表类（LinkedList），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class描述链表结点类（Node），这样使得链表结点类（LinkedList）可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类（Node）**

**公有成员：**

int data;//存储该元素数据

Node \*next;//链表结点的指针域

**公有操作：**

Node::Node();//默认构造函数

Node::Node(int d);//构造函数

**链表类（LinkedList）**

**私有成员：**

Node \*first;//链表的头指针

Node \*tail;//链表的辅助尾指针

**公有操作：**

LinkedList::Linkedlist() { first =new Node();tail = first; }//构造函数

void LinkedList::create();//创建链表

**友元函数：**

friend void print(Linkedlist\* s);//打印链表

friend Linkedlist\* compare(Linkedlist&s1,Linkedlist&s2);//比较函数

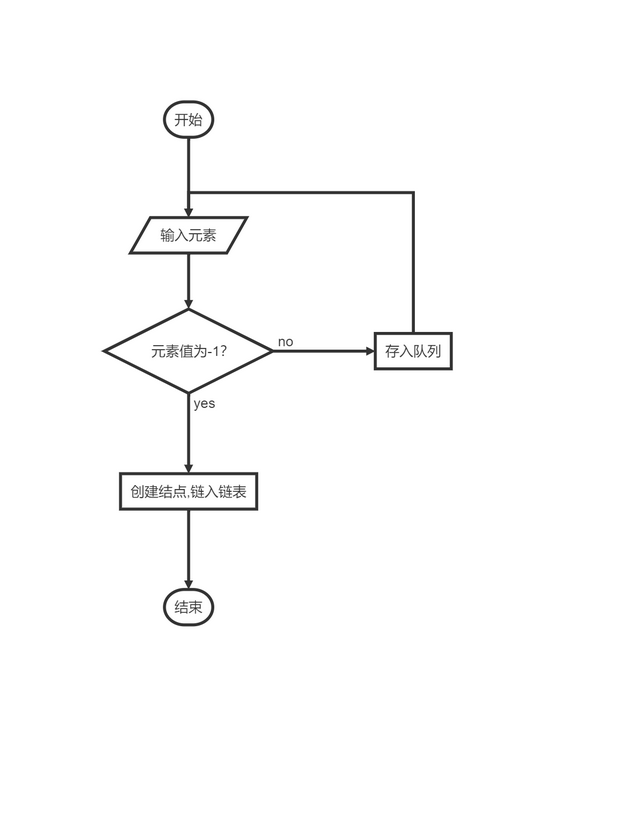
## 2.4 系统设计

系统首先提示用户输入，完成对两个链表s1、s2的创建和输入数据工作，调用比较函数（compare）返回结果链，并调用打印（print）函数打印。

# 3 实现

## 3.1 创建功能的实现

### 3.1.1 创建功能流程图



### 3.1.2 创建功能核心代码

void Linkedlist::create(){//创建链表

int d;

cin >> d;

if (d == -1)

return;

Queue<int> data;

while (d != -1) {//将输入数据存入一队列

data.push(d);

cin >> d;

}

while (data.empty() == false){//从队列中取出数据

Node\* newnode = new Node(data.front());//创建新结点链入链表

data.pop();

tail->next = newnode;

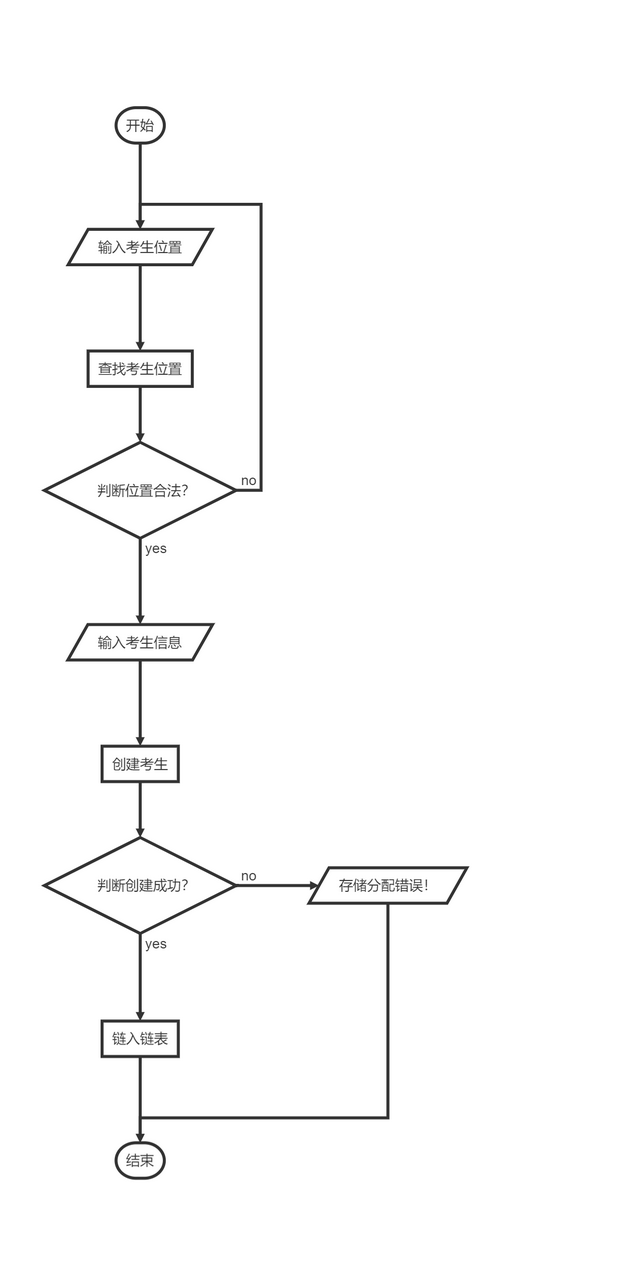
tail = newnode;

}

}

## 3.2 比较功能的实现

### 3.2.1 比较功能流程图



### 3.2.2 比较功能核心代码

Linkedlist\* compare(Linkedlist& s1,Linkedlist& s2) {

Node\* current1 = s1.first->next;

Node\* current2 = s2.first->next;

Linkedlist\* s3=new Linkedlist();

if (current1 == NULL || current2 == NULL)//任一链表为空交集为空

return NULL;

while (current1 != NULL && current2 != NULL)

{

if (current1->data < current2->data)

current1 = current1->next;

else if (current1->data > current2->data)//将较小链沿链后移

current2 = current2->next;

else

{

Node\* newnode = new Node(current1->data);

s3->tail->next = newnode;

s3->tail= newnode;

current1 = current1->next;

current2 = current2->next;

}

}

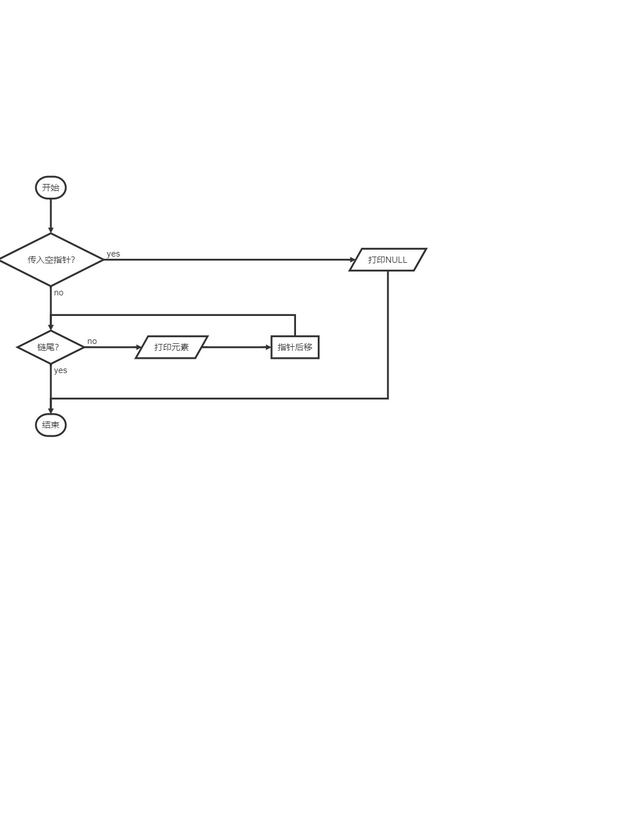
if (s3->first->next == NULL)//交集为空，返回空指针

return NULL;

return s3;}

## 3.3 打印功能的实现

### 3.3.1 打印功能流程图



### 3.3.2 打印功能核心代码

void print(Linkedlist\* s)

{

if (s == NULL)//若传入空指针打印NULL

{

cout << "NULL";

return;

}

Node\* current = s->first->next;

while (current != NULL)

{

cout << current->data;

current = current->next;

if (current != NULL)

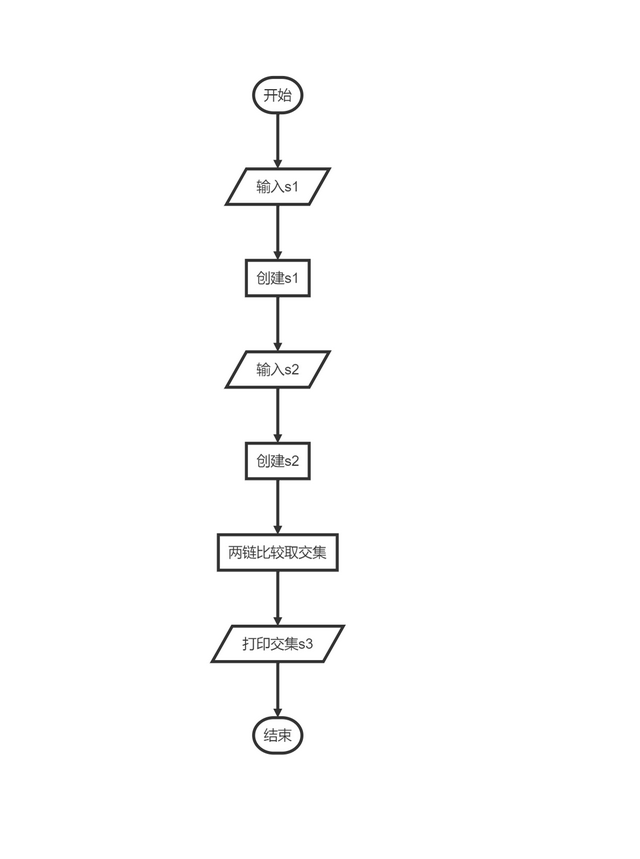
cout << ' ';

}

}

## 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

int main(){

Linkedlist s1,s2;

cout << "请输入序列1（以-1表示序列结尾）：" << endl;

s1.create();

cout << "请输入序列2（以-1表示序列结尾）：" << endl;

s2.create();

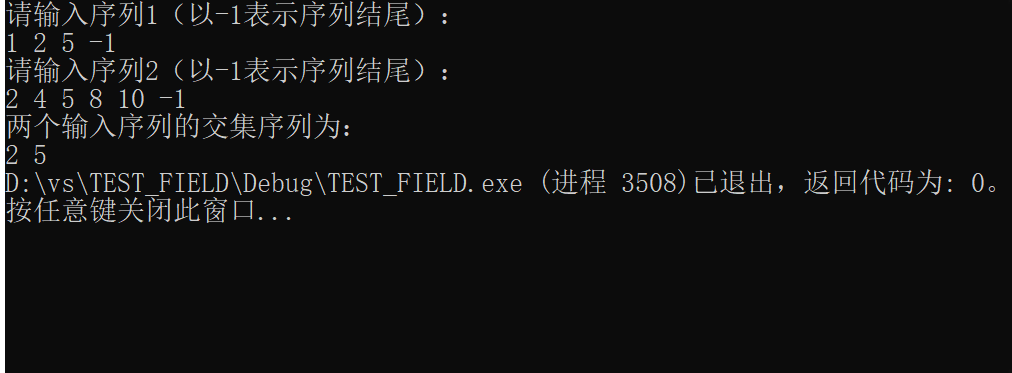
cout << "两个输入序列的交集序列为：" << endl;

print(compare(s1, s2));

return 0;

}

### 3.4.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1 一般情况

**测试用例：**

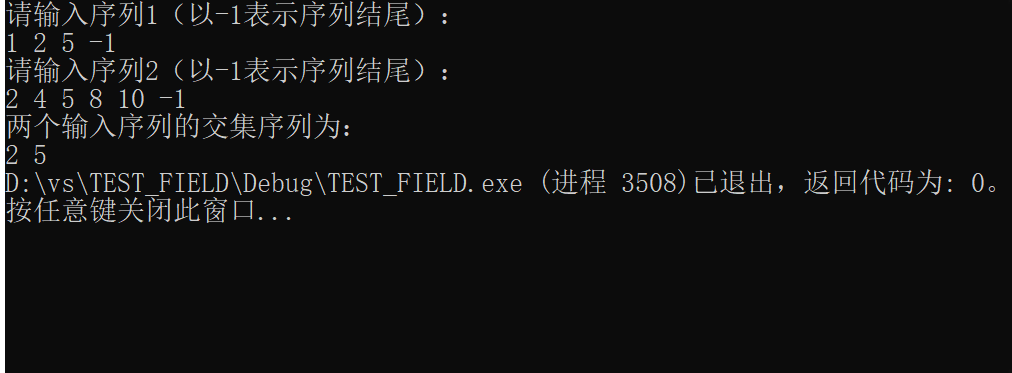
1 2 5 -1

2 4 5 8 10 -1

**预期结果：**

2 5

**实验结果：**



### 4.1.2 交集为空

**测试用例：**

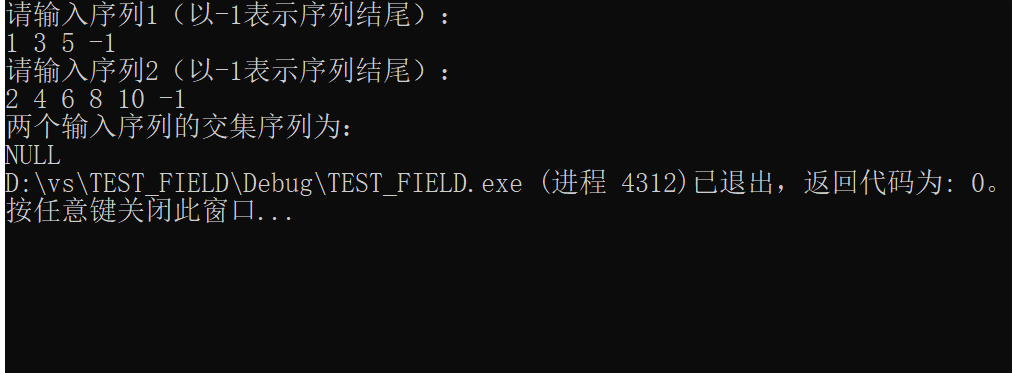
1 3 5 -1

2 4 6 8 10 -1

**预期结果：**

NULL

**实验结果：**



### 4.1.3 完全相交

**测试用例：**

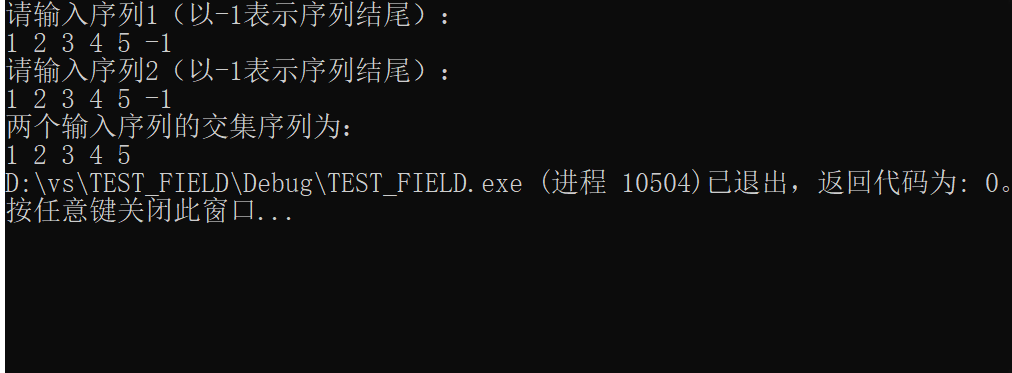
1 2 3 4 5 -1

1 2 3 4 5 -1

**预期结果：**

1 2 3 4 5

**实验结果：**



### 4.1.4 一个序列完全属于交集

**测试用例：**

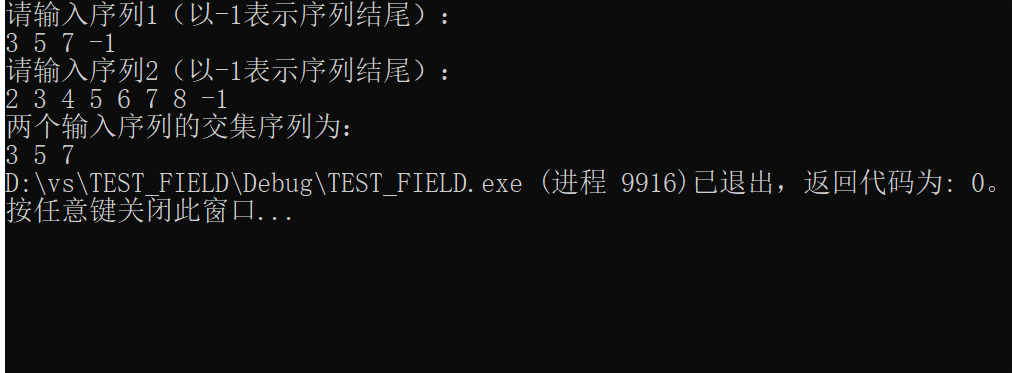
3 5 7 -1

2 3 4 5 6 7 8 -1

**预期结果：**

3 5 7

**实验结果：**

****

### 4.1.5序列为空

**测试用例：**

-1

10 100 1000 -1

**预期结果：**

NULL

**实验结果：**

