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： 谢宇翔

学 号： 1951708

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](#_Toc19947)

[1.1 背景分析 1](#_Toc898)

[1.2 功能分析 1](#_Toc4217)

[2 设计 1](#_Toc15245)

[2.1 数据结构设计 1](#_Toc29399)

[2.2 类结构设计 1](#_Toc22552)

[2.3 成员与操作设计 1](#_Toc10471)

[2.4 系统设计 2](#_Toc3831)

[3 实现 3](#_Toc1216)

[3.1 功能的实现 3](#_Toc22319)

[3.1.1 功能流程图 3](#_Toc370)

[3.1.2功能核心代码 3](#_Toc10023)

[3.1.3 功能截屏示例 5](#_Toc27900)

[4 测试 6](#_Toc18768)

[4.1 功能测试 6](#_Toc29392)

[4.2 边界测试 7](#_Toc8970)

[4.2.1 无输入数据 7](#_Toc5019)

[4.2.2 两链表相同 8](#_Toc26589)

[4.2.3 交集为空 8](#_Toc15402)

[4.3 出错测试 9](#_Toc17206)

[4.3.1 输入数据错误 9](#_Toc31911)

# 1 分析

## 1.1 背景分析

对2个有序链表的交集可以帮助我们很好掌握对有序链表的操作

## 1.2 功能分析

首先输入2个有序链表，以-1结尾。

对2个链表遍历，相等的记录下来，不等的，操作数小的链表后移，直到其中一个链表空为止。

最后输出交集

# 2 设计

## 2.1 数据结构设计

本次作业较为简单，只需2个链表即可。

## 2.2 类结构设计

本节使用了2个抽象类型---链表结点结构体（node）与链表类（List），list类里包含list型（head和current）指针用于操作。为方便处理，本系统采用struct描述链表结点类（LNode），这样使得链表结点类（LinkList）可以访问链表结点。

## 2.3 成员与操作设计

struct node

{

int data = 0;

node\* next = NULL;

};

class list

{

node\* head = new node, \* current = head;

public:

void get();

friend void intersection(list S1, list S2, list& S3);//交集

friend void show(list s);

};

**公有操作：**

void list::get()

//读入一个链表的所有数据成员

void show(list s);

//输出s链表的所有成员

void intersection(list s1,list s2,list &s3) ;

//对s1,s2链表取交集并存入s3中。

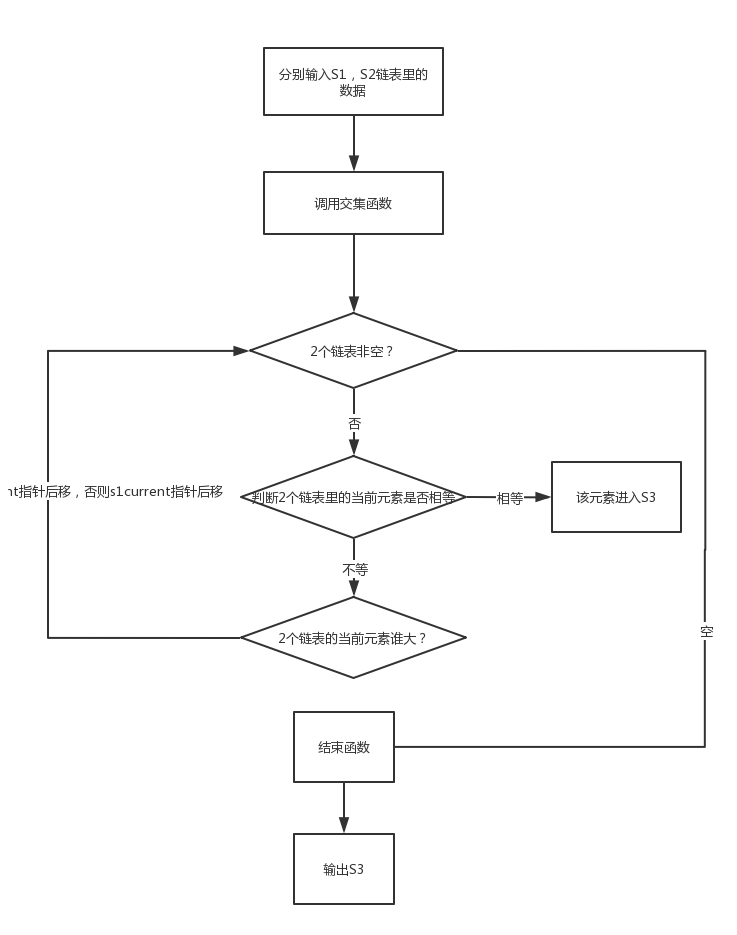
## 2.4 系统设计

首先对2个链表分别赋值，调用intersection函数，将并集存储于s3中，调用show函数输出s3即可。

# 3 实现

## 3.1 功能的实现

### 3.1.1 功能流程图



### 3.1.2功能核心代码

1. 交集函数

void intersection(list s1, list s2,list &s3)

{

s1.current = s1.head->next;

s2.current = s2.head->next;

while (s1.current->data != -1 && s2.current->data != -1)

{

if (s1.current->data < s2.current->data)

{

s1.current = s1.current->next;

}

else if (s1.current->data > s2.current->data)

{

s2.current = s2.current->next;

}

else

{

s3.current->next = new node;

s3.current = s3.current->next;

s3.current->data = s1.current->data;

s1.current = s1.current->next;

s2.current = s2.current->next;

}

}

1. Get函数

void list::get()

{

while (1)

{

int i;

cin >> i;

current->next = new node;

current = current->next;

current->data = i;

if (i == -1)

{

break;

}

}

}

1. show函数

void show(list s)

{

s.current = s.head->next;

if (s.current == NULL)//空集

{

cout << "NULL" << endl;

}

while (s.current != NULL)

{

if (s.current != s.head->next)

{

cout << " ";

}

cout << s.current->data;

s.current = s.current->next;

}

}

1. 主函数

int main()

{

cout << "请输入2个非降序序列:(以-1结束)" << endl;

list S1, S2 ,S3;

S1.get();

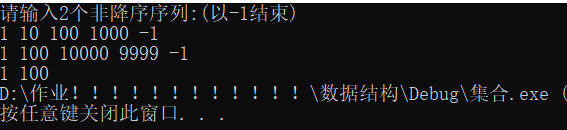
S2.get();

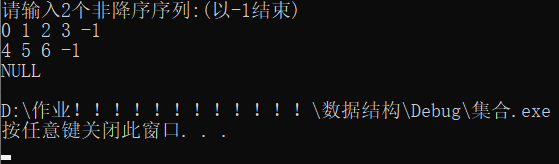
intersection(S1, S2,S3);

show(S3);

}

### 3.1.3 功能截屏示例

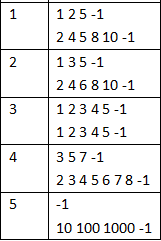




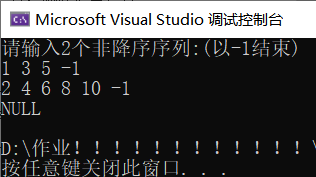
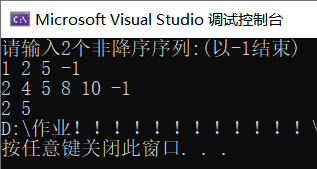
# 4 测试

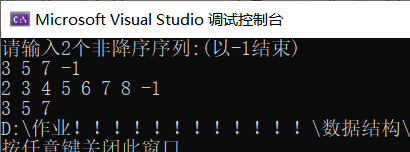
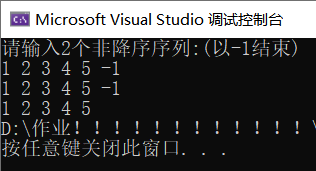
## 4.1 功能测试

测试数据:



输出：





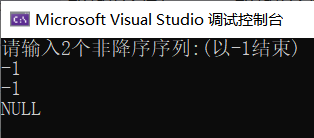
## 4.2 边界测试

### 4.2.1 无输入数据

**测试用例：**初始无输入数据

**预期结果：输出NULL**

**实验结果：**

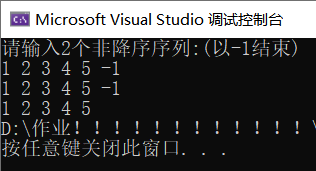


### 4.2.2 两链表相同

**测试用例：**1 2 3 4 5 -1

1 2 3 4 5 -1

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

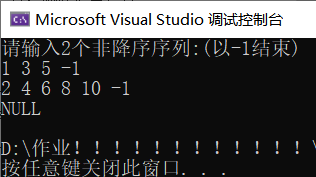
### 4.2.3 交集为空

**测试用例：**

1 3 5 -1

2 4 6 8 10 -1

**预期结果：**输出NULL。

**实验结果：**

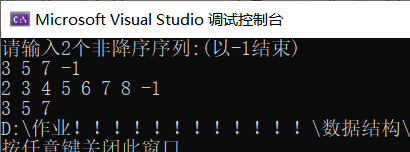
4.2.4一个集合是另一个集合的子集

测试用例:

3 5 7 -1

2 3 4 5 6 7 8 -1

实验结果:



## 4.3 出错测试

### 4.3.1 输入数据错误

**测试用例：**输入数据为非整数

**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：**

