项目说明文档

数据结构课程设计

——双链表求交集

作 者 姓 名： 王加炜

学 号： 2150265

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 分析

## 1.1功能分析

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。

# 设计

## 2.1数据结构设计

如上功能分析所述，该系统需要进行取交集的功能实现。而用链表存储数据信息相比用数组存储处理该类问题可以大大降低时间复杂度，因此决定采用链表来存储数据。

## 2.2类结构设计

数据存储在一个具有link域的结构体中。通过链表，可以将数据链式存储在内存单元中。

## 2.3成员与操作设计

newnode类：

struct newnode

{

int data;

newnode\* link;

newnode(newnode\* ptr = NULL)

{

link = ptr;

}

newnode( int value, newnode\* ptr = NULL)

{

data = value;

link = ptr;

}

};

list类：

class list

{

newnode\* first;

public:

newnode\* gethead()//获取头指针

{

return first;

}

list()//初始化函数

{

first=new newnode;

}

~list()//析构函数

{

empty();

}

void input();//输入函数

void empty();

void output();//输出函数

};

## 2.4其他函数

求交集函数：

void jiaoji(list a,list b,list&c)

{

//求交集操作,时间复杂度为一阶

newnode\* a2 = a.gethead();

newnode\* b2 = b.gethead();

newnode\* c1 = c.gethead();

while (a2->link != NULL && b2->link != NULL)

{

if (a2->link->data == b2->link->data)

{

c1 = c1->link = new newnode(a2->link->data);

a2 = a2->link;

b2 = b2->link;

}

else if (a2->link->data < b2->link->data)

{

a2 = a2->link;

}

else

{

b2 = b2->link;

}

}

}

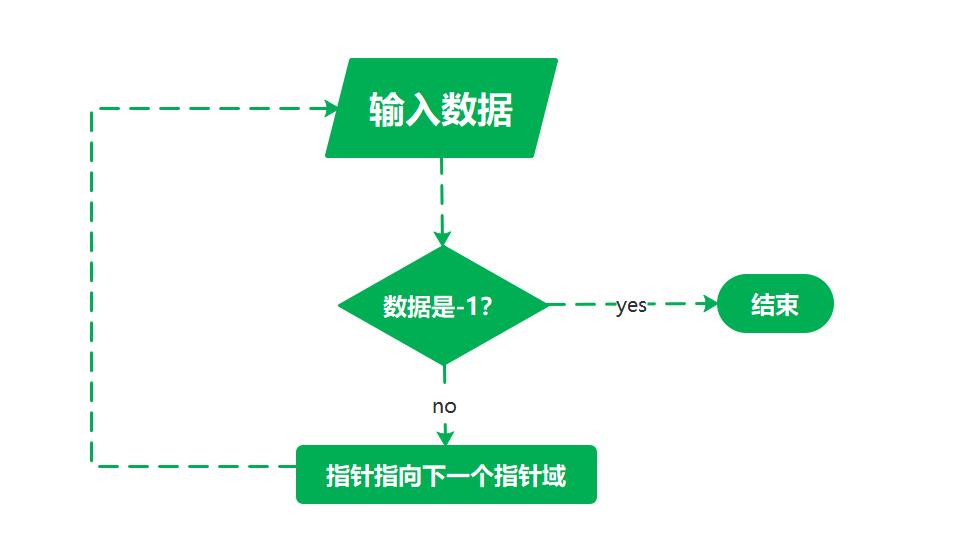
## 2.5系统设计

系统首先创建两个新链表，同时创建另一个链表用来保存求交集运算的结果；系统输入两个集合中的数据，通过求交集函数求出两个集合的交，最后将结果输出。

# 三．实现

## 3.1数据按顺序输入：

### 3.1.1流程图



### 3.1.2代码实现

void list::input()

{

cout << "请输入链表中的数据（从小到大，以-1结束）";

int val;

newnode\* a1 = first;

cin >> val;

while (val != -1)

{

a1 = a1->link = new newnode(val);

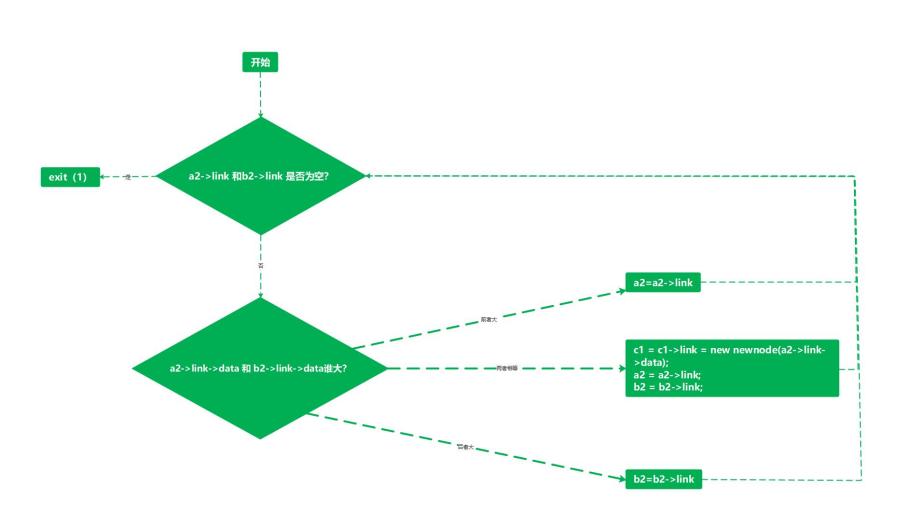
cin >> val;

}

}

## 3.2取交集：

### 3.2.1流程图：



### 3.2.2文字说明

首先建立两个数组的头指针，然后从两个数组的头指针开始分别向前遍历数据。如果第一个链表的结点中的数据大于或小于第二个链表中结点的数据，则将那个较小数据的结点指向它下一个结点。原因是较小的数据不会有后续数据和它相等，所以可以直接跳过它从下一个开始继续搜索。如果两个链表的结点数据相等，则将该数据存储在新的链表中，并将该链表指针指向下一个地址。重复该操作，直到有一个链表结点指向NULL时停止执行。本操作最多需要len（a+b）步，因此时间复杂度是一阶的，可以提升运行效率。

### 3.2.3代码实现

void jiaoji(list a,list b,list&c)

{

//求交集操作,时间复杂度为一阶

newnode\* a2 = a.gethead();

newnode\* b2 = b.gethead();

newnode\* c1 = c.gethead();

while (a2->link != NULL && b2->link != NULL)

{

if (a2->link->data == b2->link->data)

{

c1 = c1->link = new newnode(a2->link->data);

a2 = a2->link;

b2 = b2->link;

}

else if (a2->link->data < b2->link->data)

{

a2 = a2->link;

}

else

{

b2 = b2->link;

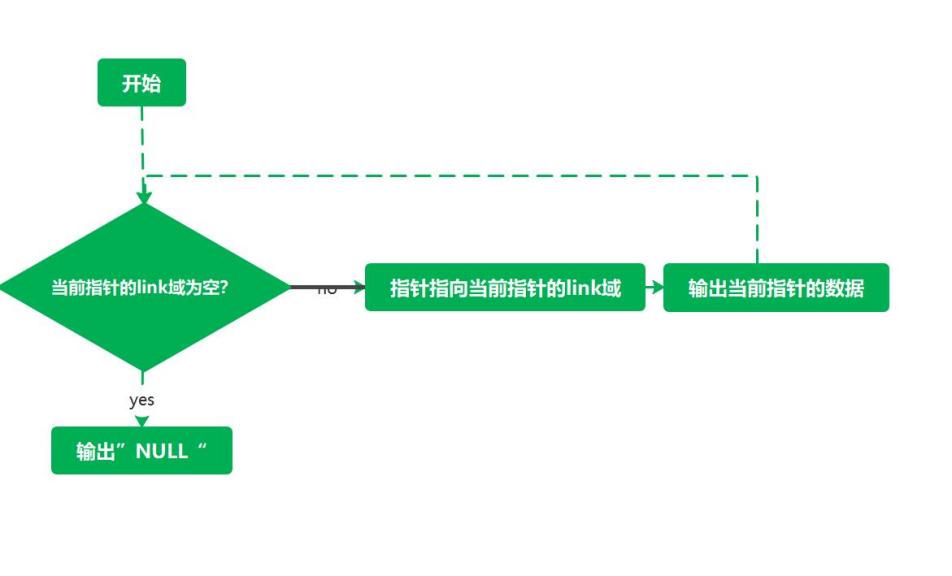
}

}

}

## 3.3输出交集：

### 3.3.1流程图



### 3.3.2代码实现

void jiaoji(list a,list b,list&c)

{

//求交集操作,时间复杂度为一阶

newnode\* a2 = a.gethead();

newnode\* b2 = b.gethead();

newnode\* c1 = c.gethead();

while (a2->link != NULL && b2->link != NULL)

{

if (a2->link->data == b2->link->data)

{

c1 = c1->link = new newnode(a2->link->data);

a2 = a2->link;

b2 = b2->link;

}

else if (a2->link->data < b2->link->data)

{

a2 = a2->link;

}

else

{

b2 = b2->link;

}

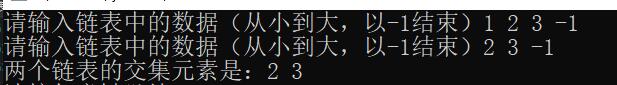
}

}

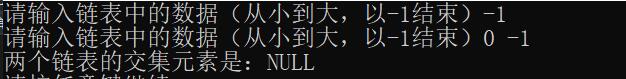
# 四．测试

## 4.1功能测试及边界测试

## 4.1.1一般情况：



## 4.1.2序列为空：



## 4.1.3完全相交：

