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：翟晨昊

学　　号：1952216

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

1.1 背景分析

在两串数据中找到它们两个共有的部分（交集）是我们在编写程序时经常用到的操作。如果两串数据保存在两个数组中，那么寻找交集需要每次拿出一个数组中的元素，在另一个数组中遍历寻找是否存在；如果两串数据保存在两个链表中，那么在比对成功后可以删除该结点，使得比对总次数会略微减少。现在，如果给出了两串非降序正整数数据保存在两个链表中，那么会不会存在有更好的求交集办法？

1.2 功能分析

项目要求采用链表进行操作，首先需要建立链表，并将数据输入进链表中，同时最好也可以检查输入的是否是非降序正整数序列。这就需要插入并检查功能。其次，还需要将生成的交集链表显示，需要输出功能。执行完操作后，还需要将链表都清空，回收内存，需要清空功能。

综上所述，一个求非降序正整数链表序列交集的程序需要有输入、输出、插入并检查、清空的功能。

2 设计

2.1 数据结构设计

如上功能分析所述，该系统使用了链表数据结构，由于比对时只需要朝一个方向进行，因此最后选择使用单向链表来存储信息。同时在链表前附加了一个头结点，这样使得增加和删除头结点时与处理其它结点方法相同，简化了代码。

2.2 类结构设计

链表一般包括两个抽象数据类型——链表结点类（Node 类）与链表类（LinkList 类），Node 类来储存每一个数据，LinkList 类来将数据形成一个序列，并提供插入并检查，清空等操作。这两个类通过友元来建立起联系，这样使得 LinkList 类可以访问 Node 类。为了使链表更加具有泛用性，本系统将 Node 类与 LinkList 类都设计为了模板类。

2.3 成员与操作设计

链表结点类（Node）：

```
template <typename ElementType>
class Node {
public:
    friend class LinkList<ElementType>;
    Node() = default;
    Node(ElementType& inputData) :data(inputData) {}
    ElementType getData();
private:
    ElementType data;
    Node<ElementType>* next = nullptr;
};
```

私有成员：

ElementType data;//链表结点中的数据
Node<ElementType>* next;//指针域，表示该结点的下一个结点

公有操作：

```
friend class LinkList<ElementType>;
//将 LinkList 声明为友元

Node () = default;
//默认构造函数

Node(ElementType& inputData);
//含输入信息的构造函数
```

ElementType getData();
//获取链表结点中的数据

链表类（LinkList）：

```
template <typename ElementType>
class LinkList {
public:
    LinkList();
    ~LinkList();
    void append(ElementType inputData);
    void deleteList();
    void goToNext();
    bool readAndCheck();
    Node<ElementType>* getCurrent();
    void display();
private:
    Node<ElementType>* head;
    Node<ElementType>* current;
    int size;
}
```

私有成员：

Node<ElementType>* head;//指针域，指向链表的头结点
Node<ElementType>* current;//指针域，指向当前正在访问的结点
int size;//链表中的结点个数

公有操作：

LinkList();
    //构造函数，开辟一个附加头结点并将链表结点个数设为 0，current 指向头结点

~LinkList();
//析构函数，调用 deleteList()将链表中的结点删除，实现对内存的回收

void append(ElementType inputData);
//向链表中插入新结点

void deleteList();
//清空链表并释放内存

void goToNext();
//访问当前结点的下一个结点

```
bool readAndCheck();
//将数据输入链表并检查输入后链表是否还为非降序序列

Node<ElementType>* getCurrent();
//访问当前结点

void display();
//展示链表中的所有数据
```

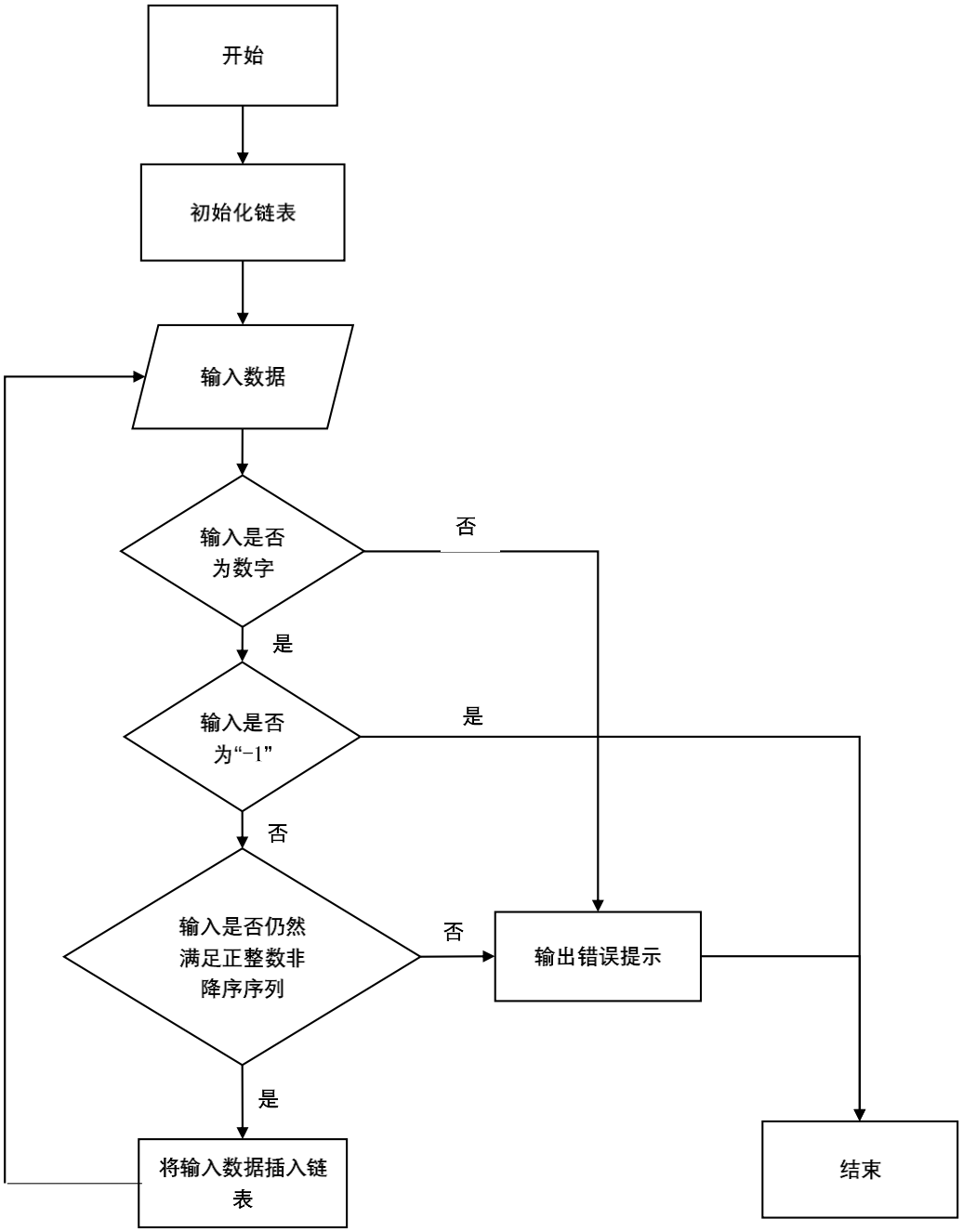
2.4 系统设计

系统会首先询问用户是否开始求交集，输入 Y 进入程序后，用户输入两行以‘-1’结尾的非降序正整数序列，调用 readAndCheck()函数来将两行数据检查并读入两个链表中，随后使用 takeIntersection()函数求出两个序列的交集，并将生成的新序列输出，随后继续询问用户是否要再进行一组求解。

3 实现

3.1 输入序列功能的实现

3.1.1 输入序列功能流程图

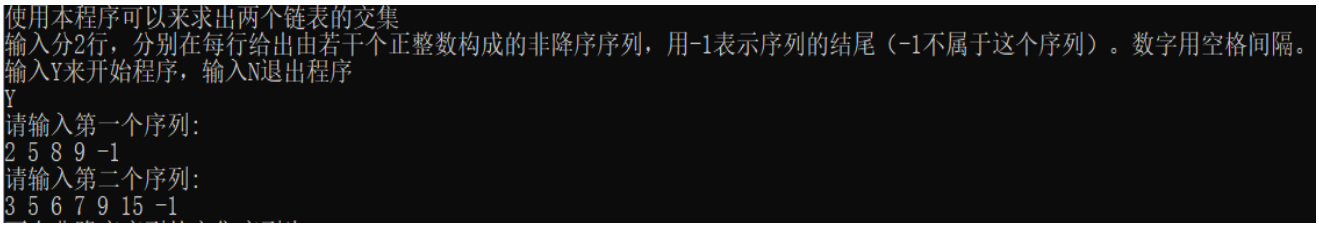


3.1.2 输入序列功能核心代码

Linklist 类中：

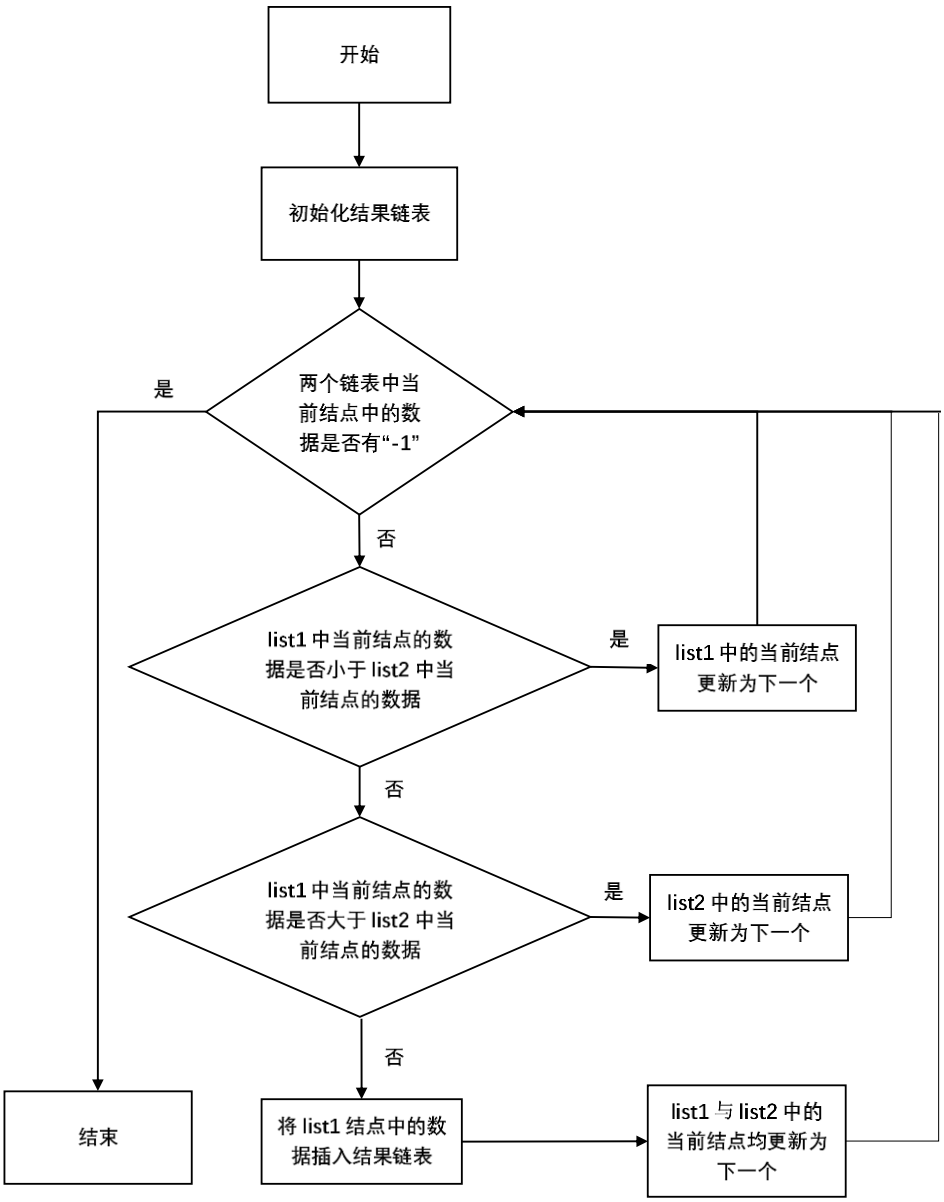
```
template <typename ElementType>
bool LinkList<ElementType>::readAndCheck()
{
    int prevNum = 1;
    int curNum = 0;
    while (cin >> curNum)
    {
        if (curNum == -1)
        {
            append(curNum);
            current = head->next;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            break;
        }
        else if (curNum < prevNum)
        {
            cout << "输入不满足是正整数非降序序列" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            return false;
        }
        else
        {
            append(curNum);
            prevNum = curNum;
        }
    }
    if (curNum == 0 || curNum == prevNum || cin.fail())
    {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(),'\n');
        cout << "输入不合法！！！" << endl;
        return false;
    }
    return true;
}
```

3.1.3 输入序列功能截屏示例



3.2 求交集功能的实现

3.2.1 求交集功能流程图



3.2.2 求交集功能核心代码

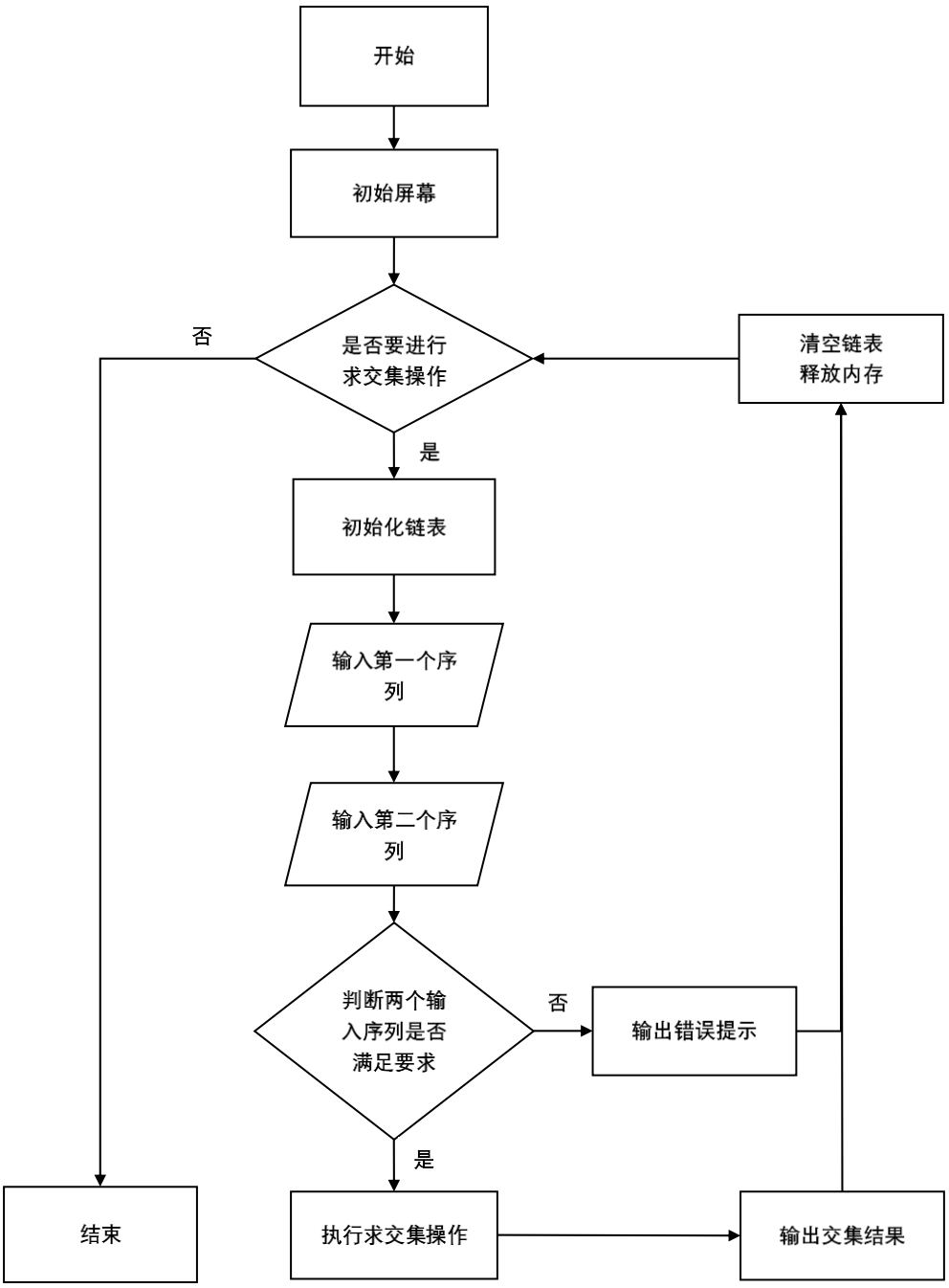
```
template <typename ElementType>
LinkList<ElementType>* takeIntersection(LinkList<ElementType>* list1, L
inkList<ElementType>* list2)
{
    LinkList<ElementType>* intersectionList = new LinkList<ElementType>
;
    while (list1->getCurrent()->getData() != -
1 && list2->getCurrent()->getData() != -1)
    {
        if (list1->getCurrent()->getData() < list2->getCurrent()->getDa
ta())
        {
            list1->goToNext();
        }
        else if (list1->getCurrent()->getData() > list2->getCurrent()->
getData())
        {
            list2->goToNext();
        }
        else if (list1->getCurrent()->getData() == list2->getCurrent()-
>getData())
        {
            intersectionList->append(list1->getCurrent()->getData());
            list1->goToNext();
            list2->goToNext();
        }
    }
    return intersectionList;
}
```

3.2.3 求交集功能截屏示例

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
2 5 8 9 -1
请输入第二个序列:
3 5 6 7 9 15 -1
两个非降序序列的交集序列为:
5 9
是否要再进行一组求链表的交集（Y为是，N为否）：Y
请输入第一个序列:
3 4 5 -1
请输入第二个序列:
1 2 -1
两个非降序序列的交集序列为:
NULL
```

3.3 总体功能的实现

3.3.1 总体功能流程图



3.3.2 总体功能核心代码

```
int main()
{
    cout << "使用本程序可以来求出两个链表的交集" << endl;
    cout << "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。"
         << endl;
    cout << "输入 Y 来开始程序，输入 N 退出程序" << endl;
    bool isRead1 = false;
    bool isRead2 = false;
    string judge = "";
    cin >> judge;
    while (judge == "Y")
    {

        LinkList<int> list1, list2;
        LinkList<int>* intersection;
        cout << "请输入第一个序列:" << endl;
        isRead1 = list1.readAndCheck();
        cout << "请输入第二个序列:" << endl;
        isRead2 = list2.readAndCheck();
        if (isRead1 && isRead2)
        {
            intersection = takeIntersection(&list1, &list2);
            cout << "两个非降序序列的交集序列为:" << endl;
            intersection->display();
            cout << endl;
            intersection->deleteList();
        }
        else
        {
            cout << "两个序列并未全部合法输入！" << endl;
        }
        cout << "是否要再进行一组求链表的交集（Y 为是，N 为否）：";
        cin >> judge;
        list1.deleteList();
        list2.deleteList();
    }
    if (judge != "Y" && judge != "N")
    {
        cout << "输入不符合要求，将会退出程序！" << endl;
    }
    cout << "成功退出程序！" << endl;
```

```
    return 0;
}
```

Linklist 类中：

```
template <typename ElementType>
void LinkList<ElementType>::display()
{
    if (size == 0)
    {
        cout << "NULL" << endl;
    }
    else
    {
        current = head;
        while (current->next != nullptr)
        {
            goToNext();
            cout << current->getData() << ' ';
        }
    }

}
```

3.3.3 总体功能截屏示例

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
2 5 8 9 -1
请输入第二个序列:
3 5 6 7 9 15 -1
两个非降序序列的交集序列为:
5 9
是否要再进行一组求链表的交集（Y为是，N为否）：Y
请输入第一个序列:
4 5 6 -1
请输入第二个序列:
1 2 3 -1
两个非降序序列的交集序列为:
NULL

是否要再进行一组求链表的交集（Y为是，N为否）：N
成功退出程序!
```

4 测试

4.1 功能测试

4.1.1 一般情况测试

测试用例：

1 2 5 -1

2 4 5 8 10 -1

预期结果：

2 5

实验结果：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
1 2 5 -1
请输入第二个序列:
2 4 5 8 10 -1
两个非降序序列的交集序列为:
2 5
```

4.1.2 交集为空情况测试

测试用例：

1 3 5 -1

2 4 6 8 10 -1

预期结果：

NULL

实验结果：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
1 3 5 -1
请输入第二个序列:
2 4 6 8 10 -1
两个非降序序列的交集序列为:
NULL
```

4.1.3 完全相交情况测试

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：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
1 2 3 4 5 -1
请输入第二个序列:
1 2 3 4 5 -1
两个非降序序列的交集序列为:
1 2 3 4 5
```

4.1.4 其中一个序列完全属于交集情况测试

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：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
3 5 7 -1
请输入第二个序列:
2 3 4 5 6 7 8 -1
两个非降序序列的交集序列为:
3 5 7
```

4.1.5 其中一个序列为空情况测试

测试用例：

-1

10 100 1000 -1

预期结果：

NULL

实验结果：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
-1
请输入第二个序列:
10 100 1000 -1
两个非降序序列的交集序列为:
NULL
```

4.2 边界测试

4.2.1 两个序列均为空

测试用例：

-1

-1

预期结果：

NULL

实验结果：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
-1
请输入第二个序列:
-1
两个非降序序列的交集序列为:
NULL
```

4.3 出错测试

4.3.1 操作码错误

测试用例：输入操作码错误
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：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
sadsafsafs
输入不符合要求，将会退出程序！
成功退出程序！
```

4.3.2 输入序列中含有非数字数据

测试用例：
2 r efwe 测试 5 6 8 -1
3 4 5 9 -1
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：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
2 r efwe 测试 5 6 8 -1
输入不合法！！！
请输入第二个序列:
3 4 5 9 -1
两个序列并未全部合法输入！
```

4.3.3 输入序列中含有降序部分

测试用例：
3 5 7 9 2 6 -1
1 5 8 3 9 -1
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：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
3 5 7 9 2 6 -1
输入不满足是正整数非降序序列
请输入第二个序列:
1 5 8 3 9 -1
输入不满足是正整数非降序序列
两个序列并未全部合法输入！
```

4.3.4 输入序列中存在非-1 的非正整数

测试用例：

2 5 8 -3 9 -1

-4 4 7 8 9 -1

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：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
2 5 8 -3 9 -1
输入不满足是正整数非降序序列
请输入第二个序列:
-4 4 7 8 9 -1
输入不满足是正整数非降序序列
两个序列并未全部合法输入！
```

4.3.5 输入序列中-1 不在序列中的最后

测试用例：

2 5 8 9 -1 11 10

3 4 5 7 9 -1 10

预期结果：程序忽略每一个序列中-1 之后的部分，正常求解，程序正常运行不崩溃。输出：5 9

实验结果：

```
使用本程序可以来求出两个链表的交集
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。
输入Y来开始程序，输入N退出程序
Y
请输入第一个序列:
2 5 8 9 -1 11 10
请输入第二个序列:
3 4 5 7 9 -1 10
两个非降序序列的交集序列为:
5 9
```