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： 杨煜

学 号： 1850217

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](#_Toc26871290)

[1.1 项目内容 1](#_Toc26871291)

[1.2 项目要求 1](#_Toc26871292)

[2 设计 2](#_Toc26871293)

[2.1 数据结构设计 2](#_Toc26871294)

[2.2 类结构设计 2](#_Toc26871295)

[2.3 成员与操作设计 2](#_Toc26871296)

[3 实现 3](#_Toc26871297)

[3.1 构造函数的实现 3](#_Toc26871298)

[3.1.1 构造函数 3](#_Toc26871299)

[3.1.2 构造函数核心代码 3](#_Toc26871300)

[3.2 析构函数的实现 3](#_Toc26871301)

[3.2.1 析构函数 3](#_Toc26871302)

[3.2.2 析构函数核心代码 3](#_Toc26871303)

[3.3 建立链表的实现 4](#_Toc26871304)

[3.3.1 建立链表 4](#_Toc26871305)

[3.3.2 建立链表核心代码 4](#_Toc26871306)

[3.4 获取头节点的实现 5](#_Toc26871307)

[3.4.1 获取头节点 5](#_Toc26871308)

[3.4.2 获取头节点核心代码 5](#_Toc26871309)

[3.5 交集函数的实现 5](#_Toc26871310)

[3.5.1 交集函数 5](#_Toc26871311)

[3.5.2 交集函数核心代码 5](#_Toc26871312)

[3.6输出功能的实现 6](#_Toc26871313)

[3.6.1 输出功能 6](#_Toc26871314)

[3.6.2 输出功能核心代码 7](#_Toc26871315)

[3.7 总体流程的实现 7](#_Toc26871316)

[3.7.1 总体流程 7](#_Toc26871317)

[3.7.2 总体系统核心代码 7](#_Toc26871318)

[3.7.3 总体系统核心代码 8](#_Toc26871319)

[4 测试 8](#_Toc26871320)

[4.1 功能测试 8](#_Toc26871321)

[4.1.1 一般情况 8](#_Toc26871322)

[4.1.2 交集为空的情况 9](#_Toc26871323)

[4.1.3 完全相交的情况 10](#_Toc26871324)

[4.1.4 其中一个序列完全属于交集的情况 11](#_Toc26871325)

[4.1.5 其中一个序列为空的情况 12](#_Toc26871326)

[4.2 出错测试 13](#_Toc26871327)

[4.2.1 含有负整数 13](#_Toc26871328)

[4.2.2 非有序序列 14](#_Toc26871329)

[4.2.3 既非有序又有负整数 15](#_Toc26871330)

# 1 分析

## 1.1 项目内容

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。

## 项目要求

1、本项目的实质是完成输入说明：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。

2、输出说明：在一行中输出两个输入序列的交集序列，数字间用空格分开，结尾不能有多余空格；若新链表为空，输出NULL。

3、测试用例：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 输入 | 输出 | 说明 |
| 1 | 1 2 5 -1  2 4 5 8 10 -1 | 2 5 | 一般情况 |
| 2 | 1 3 5 -1  2 4 6 8 10 -1 | NULL | 交集为空的情况 |
| 3 | 1 2 3 4 5 -1  1 2 3 4 5 -1 | 1 2 3 4 5 | 完全相交的情况 |
| 4 | 3 5 7 -1  2 3 4 5 6 7 8 -1 | 3 5 7 | 其中一个序列完全属于交集的情况 |
| 5 | -1  10 100 1000 -1 | NULL | 其中一个序列为空的情况 |

# 2 设计

## 2.1 数据结构设计

如上述项目要求所述，该程序需要对于两个非降序列S1和S2进行求交集的行为。毫无疑问，使用链表来实现该功能是合理的，且效率较高的。只需要通过比较两链表中元素是否相等，将相等元素链入S3即可。该程序中使用到了删除和依次比较功能，故而，链表来实现是十分理想的数据结构。

## 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（LinkNode）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，本程序采用友元的方式使得链表结点类以及链表类可以互相访问。

## 2.3 成员与操作设计

**链表结点类（LinkNode）**

class LinkNode

{

friend class List;//申明友元

private:

int number;//节点中数字

LinkNode\* link;//节点的下一个元素

};

**链表类（List）**

class List

{

public:

List();//构造函数

~List();//析构函数

bool Construction();//建立链表

void Union(List&a,List&b);//求交集运算

void Display();//输出

LinkNode\* Gethead();//取头节点

private:

LinkNode\* first;//链表头节点

};

# 3 实现

## 3.1 构造函数的实现

### 3.1.1 构造函数

构造函数构造了一个空的头节点放在节点的第一个位置使得链表成为了一个，带附加头结点的链表，方便进行查找，插入等一系列的操作，使得程序更加简洁。

### 3.1.2 构造函数核心代码

List::List()//构造函数

{

first = new LinkNode;

}

## 3.2 析构函数的实现

### 3.2.1 析构函数

析构函数通过循环查找下一个不为空节点的方式，然后一个一个删除掉链表中的节点，清空除去头节点以外的链表内容，最后则删除头节点，链表空间全部释放，防止出现碎片化的存储空间。

### 3.2.2 析构函数核心代码

List::~List()

{

LinkNode\* q;

while (first->link != NULL)//查找到尾结点

{

q = first->link;

if (q->link != NULL)

{

first->link = q->link;

delete q;

}

}

delete first;

}

## 3.3 建立链表的实现

### 3.3.1 建立链表

根据输入内容创建非降序的序列，收到输入后，如果这个输入的值大于上一个输入的值（开头默认为0）且不是非正整数时，开辟一个节点空间，将该数字链入链表之中。否则，停止输入并报出错误信息。当输入的数字为-1时，则认为该序列输入结束。当构建不成功时，输出0，成功时输出1。

### 3.3.2 建立链表核心代码

bool List::Construction()

{

LinkNode\* newNode;

LinkNode\* last;

last = first;

int n,k=0;

cin >> n;

while (n != -1)

{

if (k > n&&k>=0)//错误判断

{

cout << "序列有误，非有序序列" << endl;

return 0;

}

k = n;

newNode = new LinkNode;//建立新节点

last->link = newNode;

last = newNode;

newNode->number = n;

cin >> n;

}

last->link = nullptr;

return 1;

}

## 3.4 获取头节点的实现

### 3.4.1 获取头节点

函数返回值为附加头节点的下一个节点地址。

### 3.4.2 获取头节点核心代码

LinkNode\* List::Gethead()

{

return this->first->link;

}

## 3.5 交集函数的实现

### 3.5.1 交集函数

此为核心功能，交集函数的参数为两个有序链表的地址。首先，通过调用获取头节点的函数得到跳过附加头节点的第一个节点的地址。传进来的两个有序链表都需要这样的一步操作。随后开始比较节点，如果两个链表都没有为空节点，则比较两节点内容，如果相等，则把该内容链入第三个链表中，否则舍弃，比较两者内容大小。如果a链表内容大，则b链表指向下一节点，反之亦然。直到两个链表中有一个为空。

### 3.5.2 交集函数核心代码

void List::Union(List& a, List& b)

{

LinkNode\* current1 = a.Gethead();

LinkNode\* current2 = b.Gethead();

LinkNode\* current3 = first;

LinkNode\* newNode;

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

{

if (current1->number == current2->number)//相等情况

{

newNode = new LinkNode;

newNode->number = current1->number;

current3->link = newNode;

current3 = newNode;

current1 = current1->link;

current2 = current2->link;

}

else if (current1->number < current2->number)//A大

{

current1 = current1 -> link;

}

else if (current1->number > current2->number)//B大

{

current2 = current2 -> link;

}

}

current3->link = nullptr;//末尾链入空

}

## 3.6输出功能的实现

### 3.6.1 输出功能

从第一个节点开始依次输出内容，如果链表为空，则输出NULL，否则输出所有内容。

### 3.6.2 输出功能核心代码

void List:: Display()

{

LinkNode\* current=this->Gethead();

if (current == nullptr)//为空情况

{

cout << "NULL";

}

else//不为空，依次输出

{

while (current != nullptr)

{

cout << current->number << " ";

current = current->link;

}

}

}

## 3.7 总体流程的实现

### 3.7.1 总体流程

首先申明三个链表，a,b,c。然后构建a和b链表，如果构建成功则继续取交集的函数并输出c的内容，否则，输出错误信息后结束该程序

### 3.7.2 总体系统核心代码

int main()

{

List a, b, c;

if (a.Construction() && b.Construction())//判断是否成功建立

{

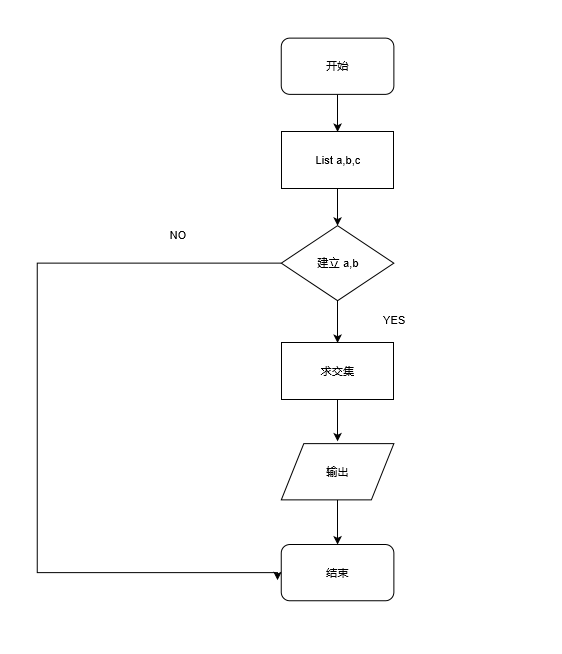
c.Union(a, b);

c.Display();

}

}

### 3.7.3 总体系统核心代码



# 4 测试

## 4.1 功能测试

### 4.1.1 一般情况

**测试用例**：

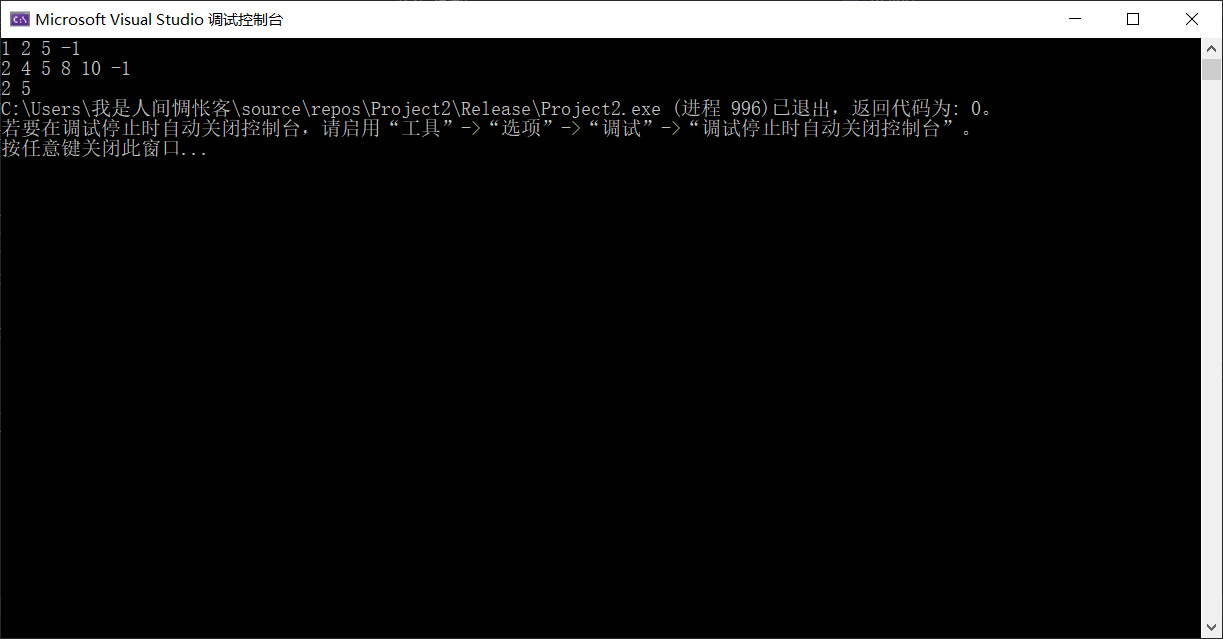
1 2 5 -1

2 4 5 8 10 -1

**预期结果**：

2 5

**实验结果**



### 4.1.2 交集为空的情况

**测试用例：**

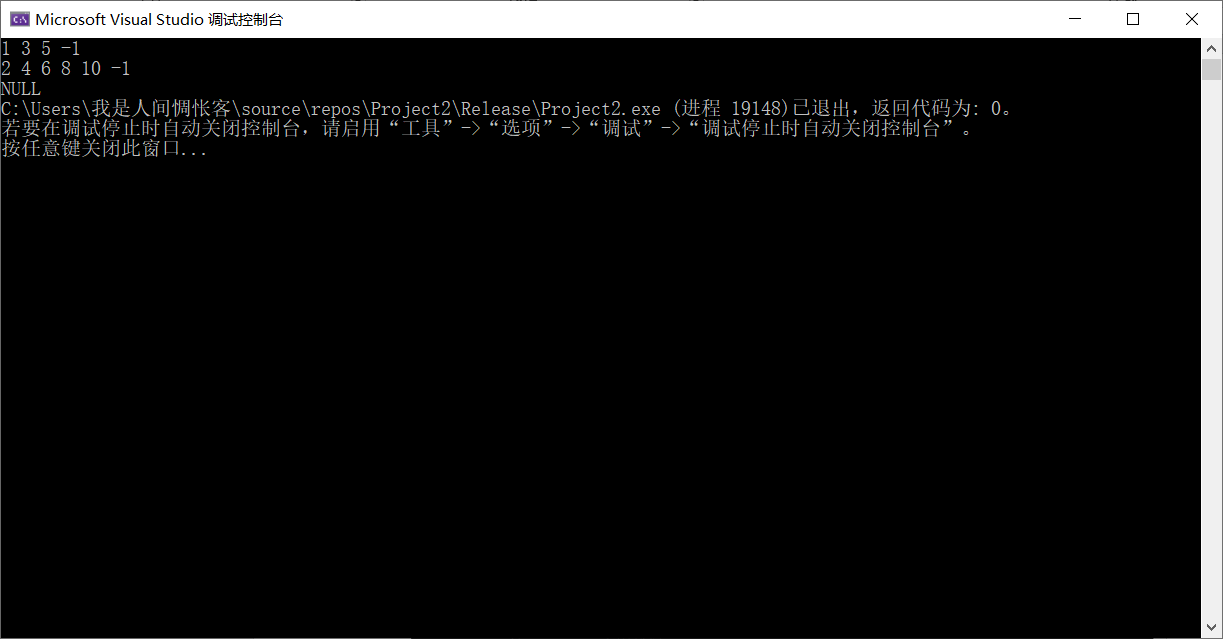
1 3 5 -1

2 4 6 8 10 -1

**预期结果：**

**NULL**

**实验结果：**



### 4.1.3 完全相交的情况

**测试用例：**

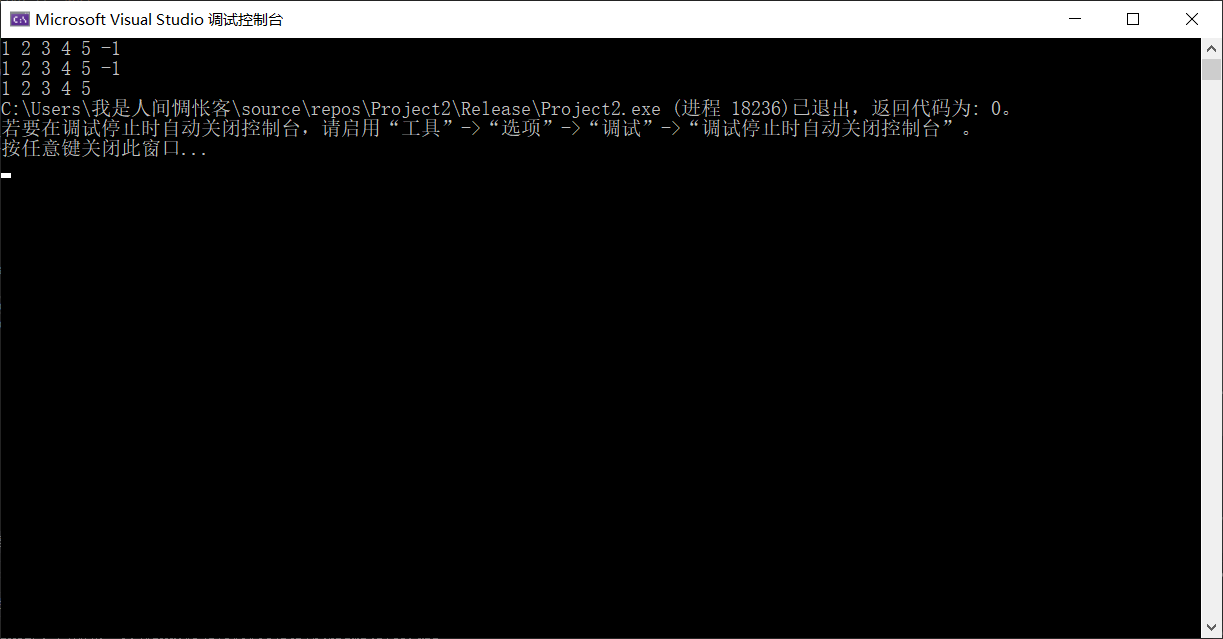
1 2 3 4 5 -1

1 2 3 4 5 -1

**预期结果：**

1 2 3 4 5

**实验结果：**



### 4.1.4 其中一个序列完全属于交集的情况

**测试用例：**

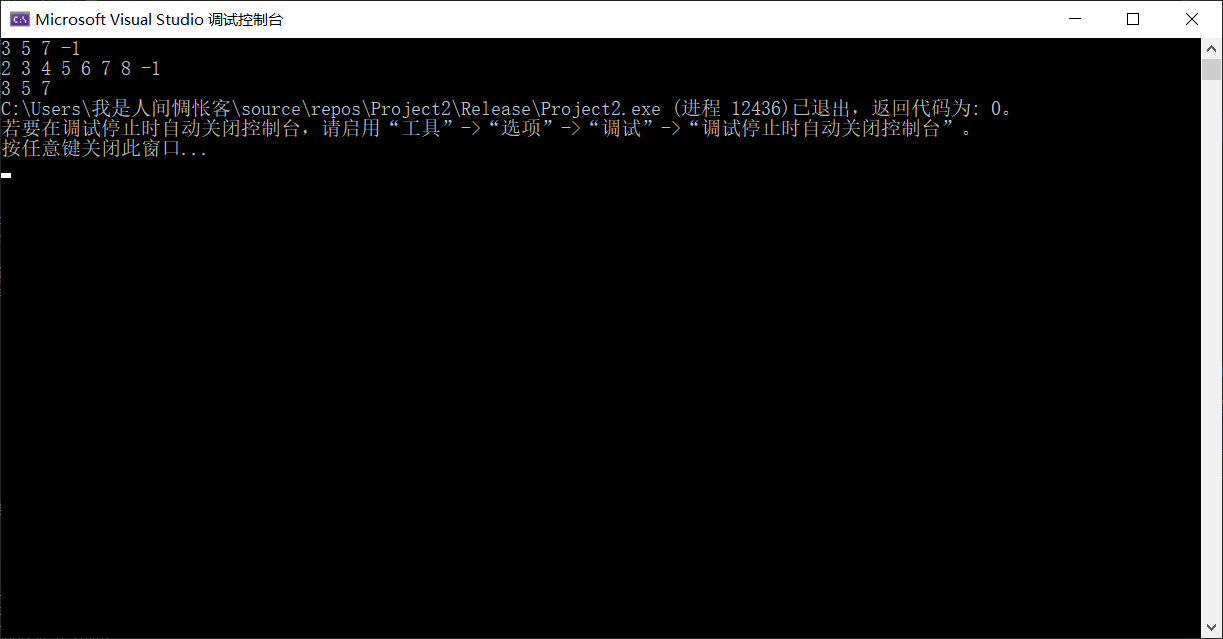
3 5 7 -1

2 3 4 5 6 7 8 -1

**预期结果：**

3 5 7

**实验结果：**



### 4.1.5 其中一个序列为空的情况

**测试用例：**

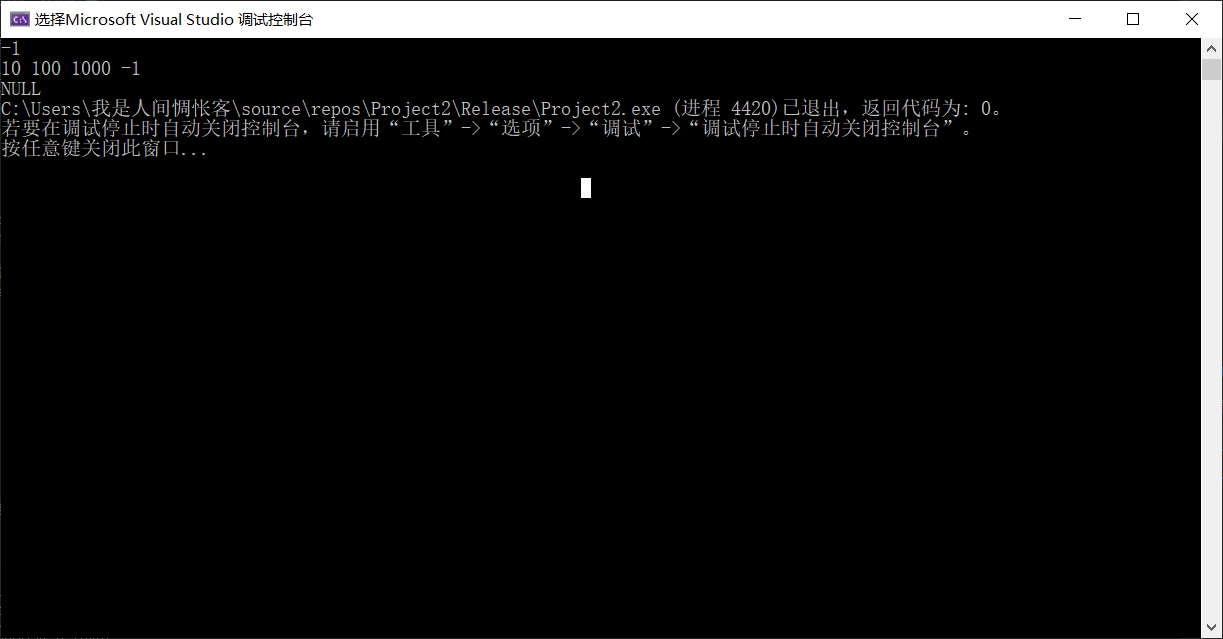
-1

10 100 1000 -1

**预期结果：**

**NULL**

**实验结果：**



## 4.2 出错测试

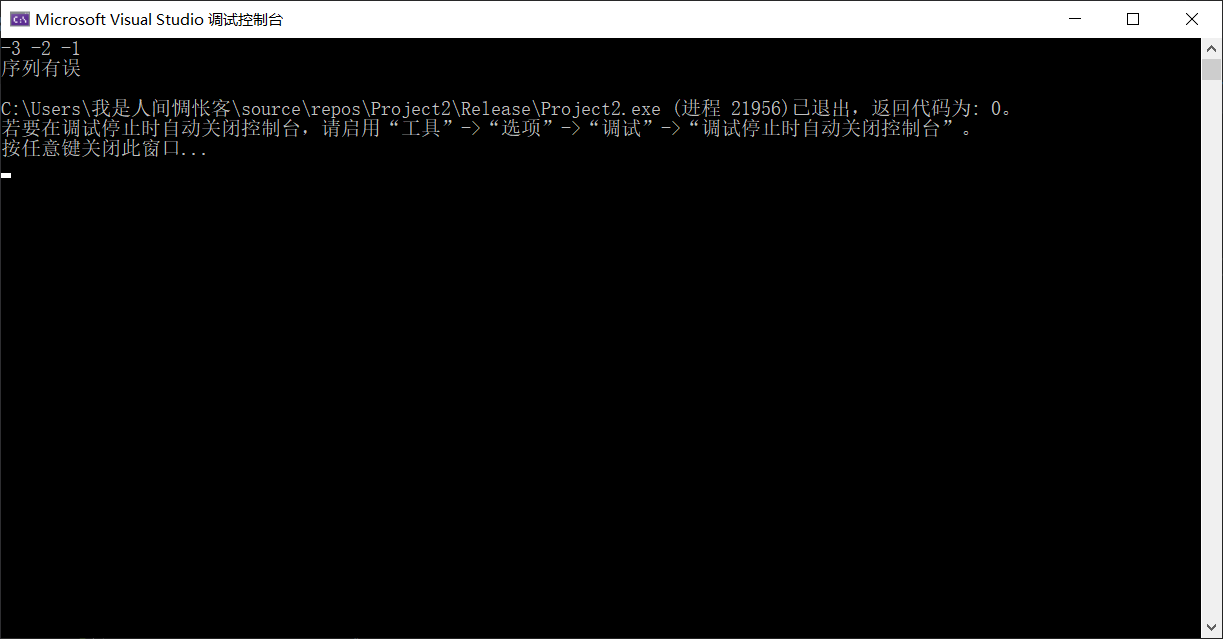
### 4.2.1 含有负整数

**测试用例：**

-3 -2 -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：**



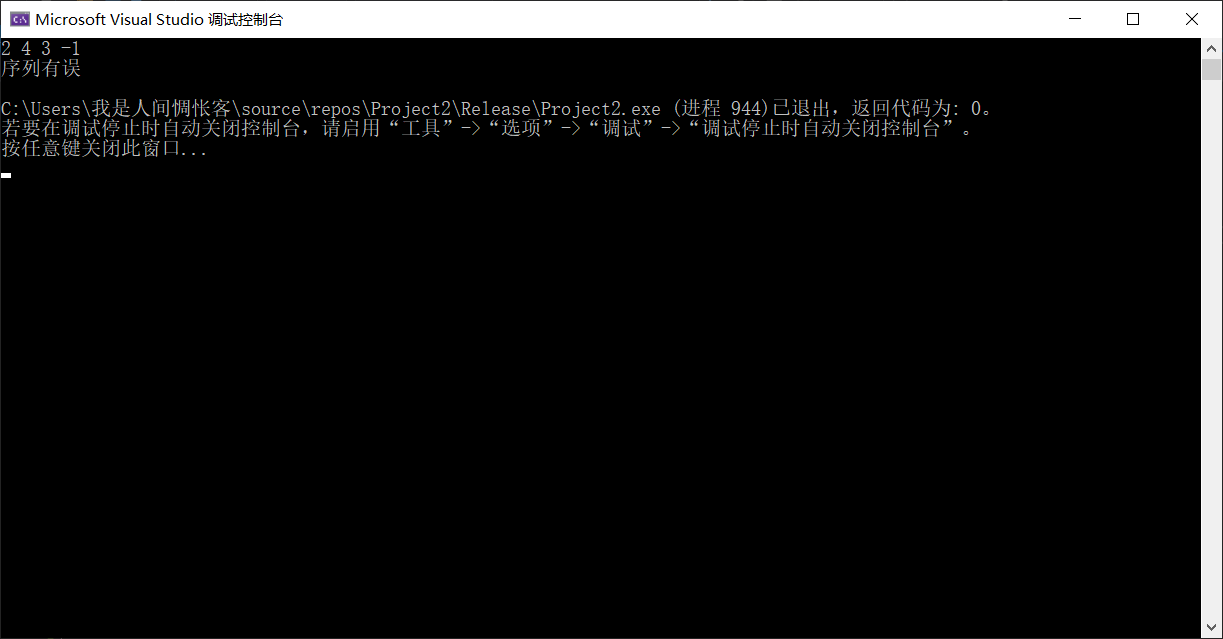
### 4.2.2 非有序序列

**测试用例：**

2 4 3 -1

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**



### 4.2.3 既非有序又有负整数

**测试用例：**

1 2 3 4 5 -1

3 -2 -4 5 -1

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

