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

同济大学

Tongji University

目 录

[1 分析 1](#_Toc3017)

[1.1 背景分析 1](#_Toc20930)

[1.2 功能分析 1](#_Toc28922)

[2 设计 1](#_Toc20440)

[2.1 数据结构设计 1](#_Toc3546)

[2.2 类结构设计 1](#_Toc21357)

[2.3 成员与操作设计 1](#_Toc5530)

[3 实现 3](#_Toc9887)

[3.1 输入功能的实现 3](#_Toc32712)

[3.1.1 输入功能流程图 3](#_Toc20944)

[3.1.2 输入功能核心代码 3](#_Toc14619)

[3.2 排序功能的实现 4](#_Toc27675)

[3.2.1 排序功能流程图 4](#_Toc22869)

[3.2.2 排序功能核心代码 4](#_Toc10091)

[3.3 取并集功能的实现 5](#_Toc19449)

[3.3.1 取并集功能流程图 5](#_Toc18666)

[3.3.2 取并集功能核心代码 5](#_Toc18814)

[3.4总体系统实现 7](#_Toc12436)

[3.4.1 总体系统流程图 7](#_Toc25237)

[3.4.2总体系统核心代码 7](#_Toc324)

[4 测试 8](#_Toc5523)

[4.1 功能测试 8](#_Toc7232)

[4.2 边界测试 8](#_Toc16225)

[4.2.1 交集为空的情况 8](#_Toc27315)

[4.2.2 完全相交的情况 9](#_Toc15371)

[4.2.3 其中一个序列完全属于交集的情况 9](#_Toc22981)

[4.2.4 其中一个序列为空的情况 10](#_Toc26688)

# 1 分析

## 1.1 背景分析

单链表是数据结构中的重要的一个组成部分之一，单链表也可以如同数组一般，进行数据的存储，可以利用两个单链表进行两个数列的求并集

## 1.2 功能分析

作为一个最简易的求并集的程序，首先应该有输入模块，排序模块，取并集模块，输出模块。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求有增加、删除、修改操作，而链表进行增加、删除等操作十分简便，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，这样就使得增加或者删除头结点与处理其他结点方法相同，使得程序简洁。

## 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（date）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，本系统采用struct描述链表结点类(date)，这样使得链表结点类（list）可以访问链表结点。

## 2.3 成员与操作设计

template<class T>

class date

{

public:

T number;

date\* link;

};

template <class T>

class list

{

private:

date<T>\* head;

date<T>\* current;

int nowsetnumber = 0;

public:

list(int a);

list(); //初始化函数

void output();//输出函数

void relist(); //排序函数

template<class T>

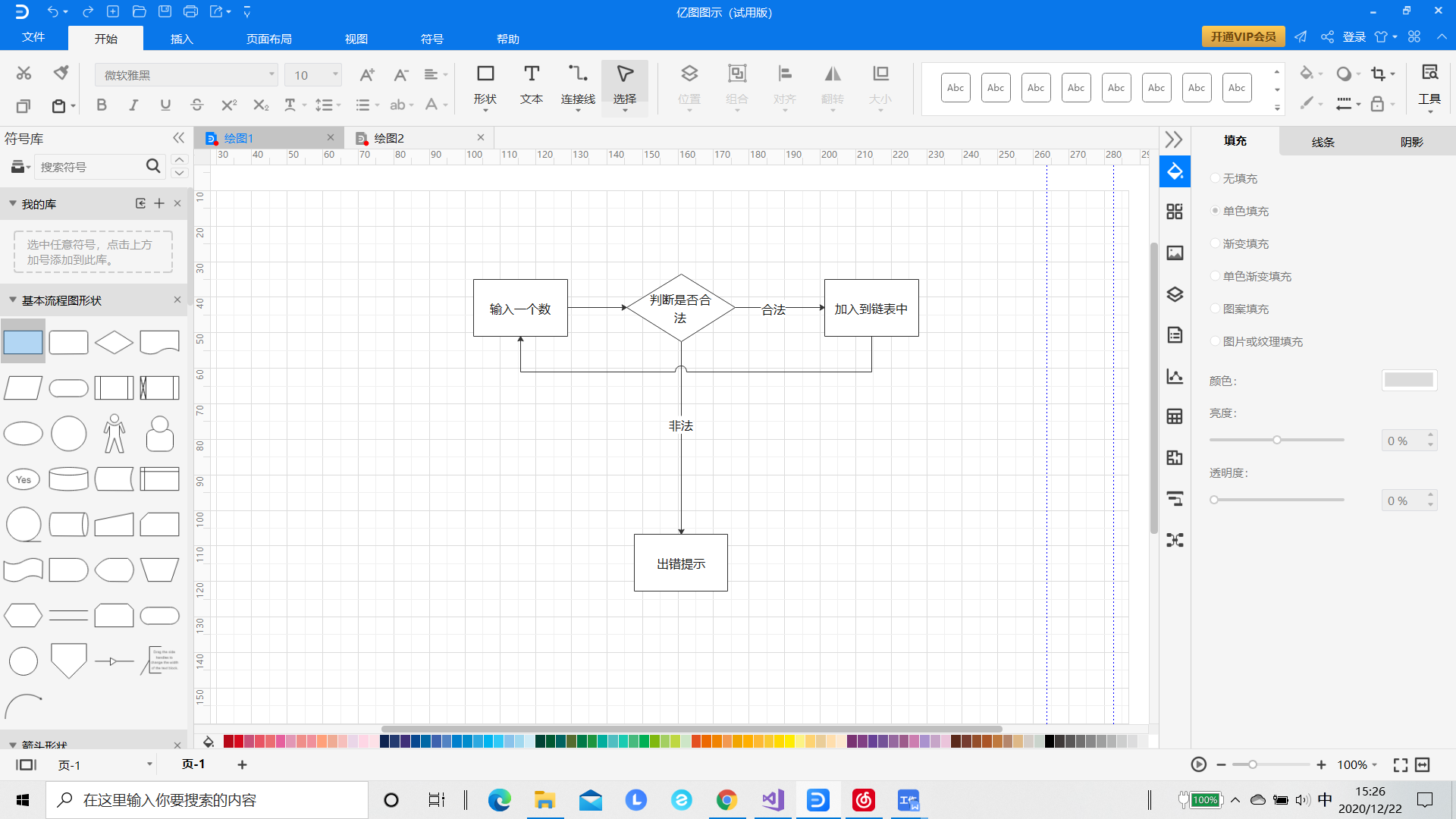
friend bool change(list<T> &a, list<T> &b, list<T> &c); //求并集函数

## };

# 3 实现

## 3.1 输入功能的实现

### 3.1.1 输入功能流程图



### 3.1.2 输入功能核心代码

类函数中的构造函数

template<class T>

list<T>::list()

{

head = new date<T>;

if (head == NULL) { cout << "头指针分配失败" << endl; }

for (int n=1;;n++)

{

T input;

cin >> input;

if (input < -1) { cout << "输入了非负1以外的负数,请重试" << endl; n--; continue; }

if (n == 1)

{

nowsetnumber++;

current = new date<T>;

current->number = input;

head->link = current;

current->link = NULL;

if (current->number == -1) { break; }

}

else

{

nowsetnumber++;

current->link = new date<T>;

current = current->link;

current->number = input;

current->link = NULL;

if (current->number == -1) { break; }

}

}

### }

## 3.2 排序功能的实现

### 3.2.1 排序功能流程图

### 采用冒泡法

### 3.2.2 排序功能核心代码

template<class T>

void list<T>::relist()

{

current = head->link;

for (int n = 1; n <= nowsetnumber-2; n++)

{

current = head->link;

for (int m = n; m <= nowsetnumber-2; m++)

{

if (current->number > current->link->number)

{

T a;

a = current->number;

current->number = current->link->number;

current->link->number = a;

current = current->link;

}

else

{ current = current->link; }

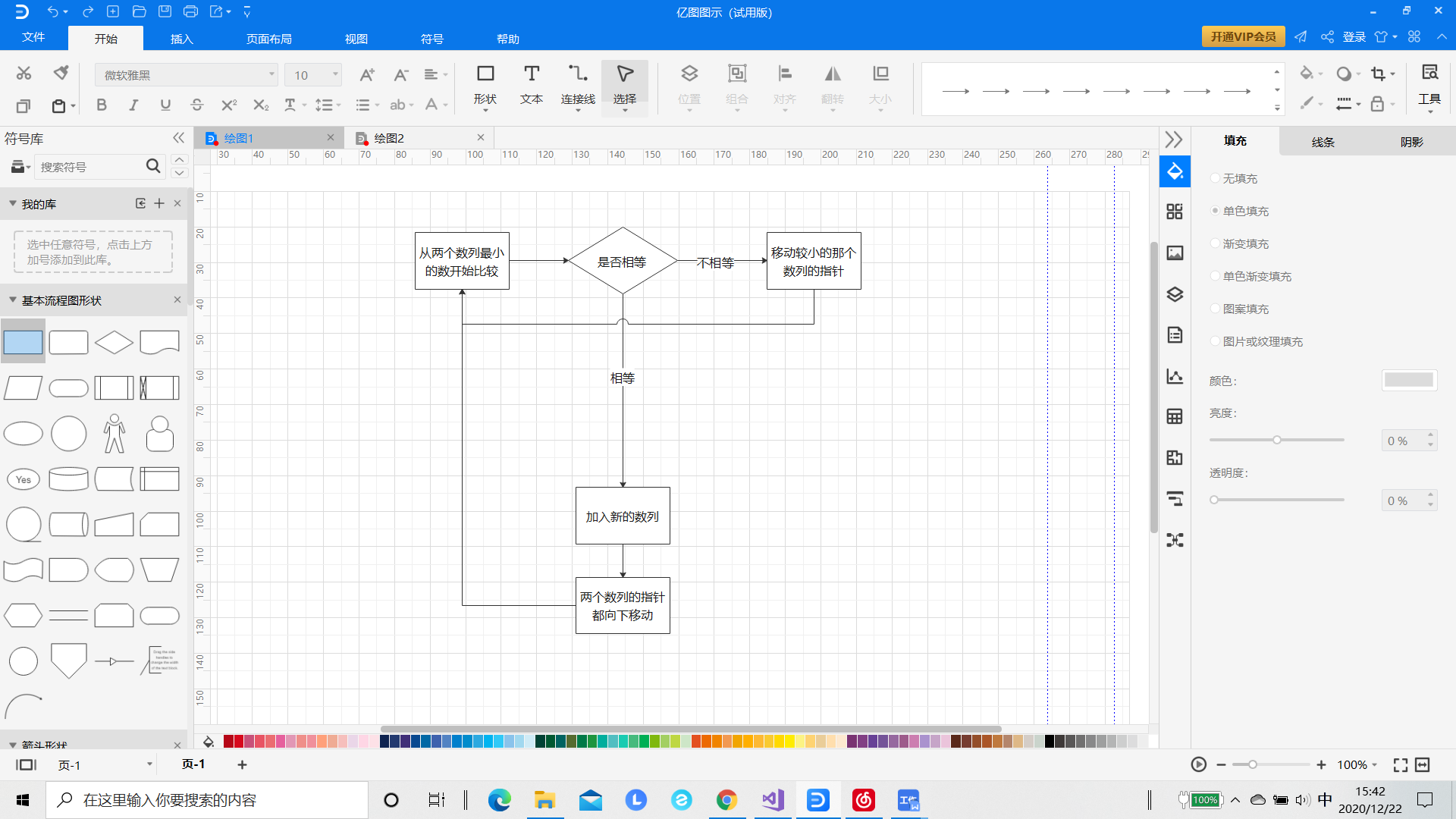
}

}

### }

## 3.3 取并集功能的实现

### 3.3.1 取并集功能流程图



### 3.3.2 取并集功能核心代码

template<class T>

bool change(list<T>&a, list<T>&b, list<T>&c)

{

a.current = a.head->link;

b.current = b.head->link;

if (a.nowsetnumber <= 1 || b.nowsetnumber <= 1)

{

return false;

}

else

{

for (;;)

{

if (a.current->number == -1 || b.current->number == -1) { break; }

if (a.current->number < b.current->number)

{

a.current = a.current->link;

}

else

{

if (a.current->number >b.current->number)

{

b.current = b.current->link;

}

else

{

c.current->number = a.current->number;

c.current->link = new date<T>;

c.current = c.current->link;

c.current->number = -1;

c.current->link = NULL;

c.nowsetnumber++;

a.current = a.current->link;

b.current = b.current->link;

}

}

}

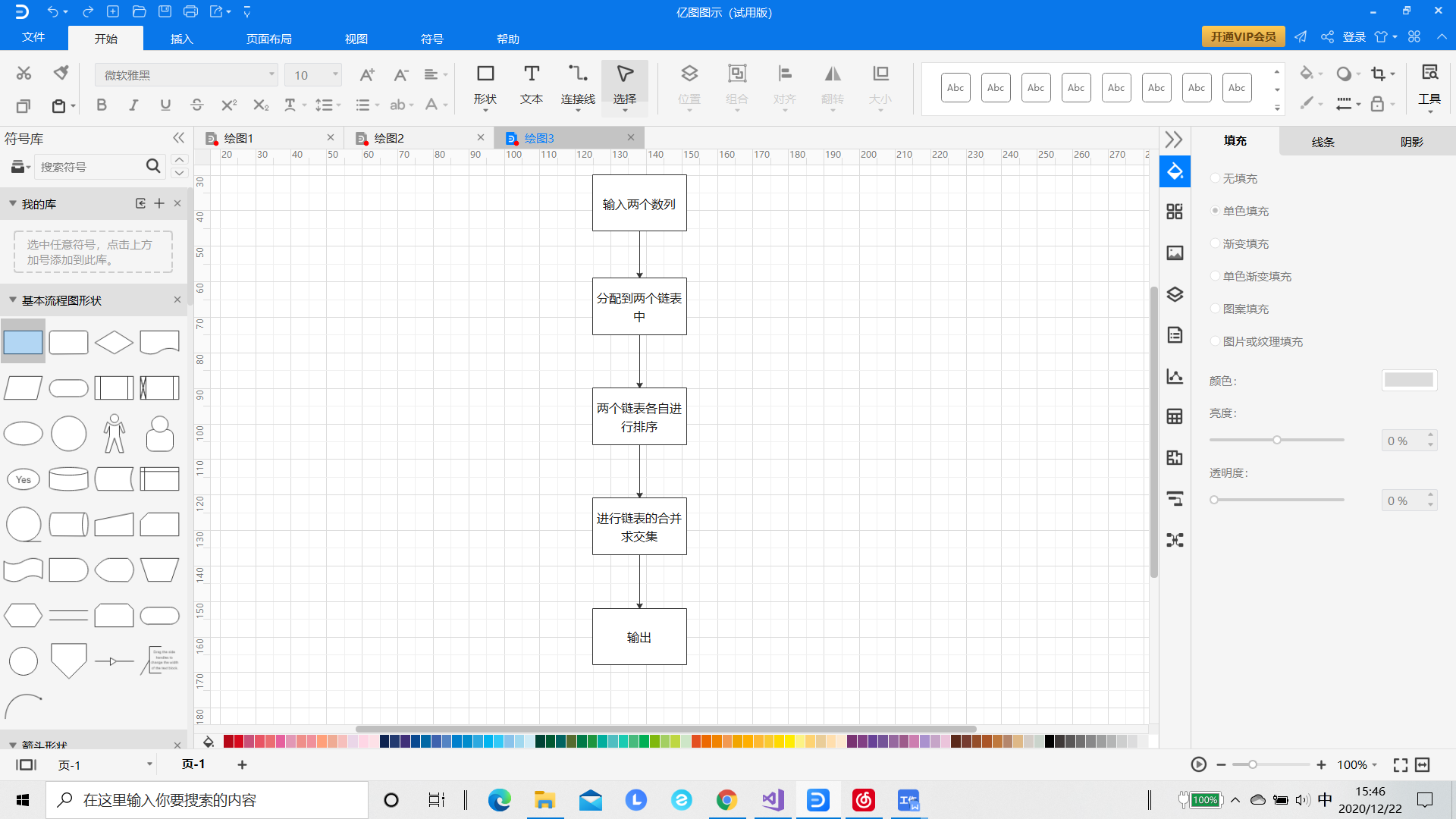
return true;

}

## }

# 3.4总体系统实现

### 3.4.1 总体系统流程图



### 3.4.2总体系统核心代码

int main()

{

list<int> a;

a.relist();

list<int> b;

b.relist();

list<int> c(-1);

change(a, b, c);

c.output();

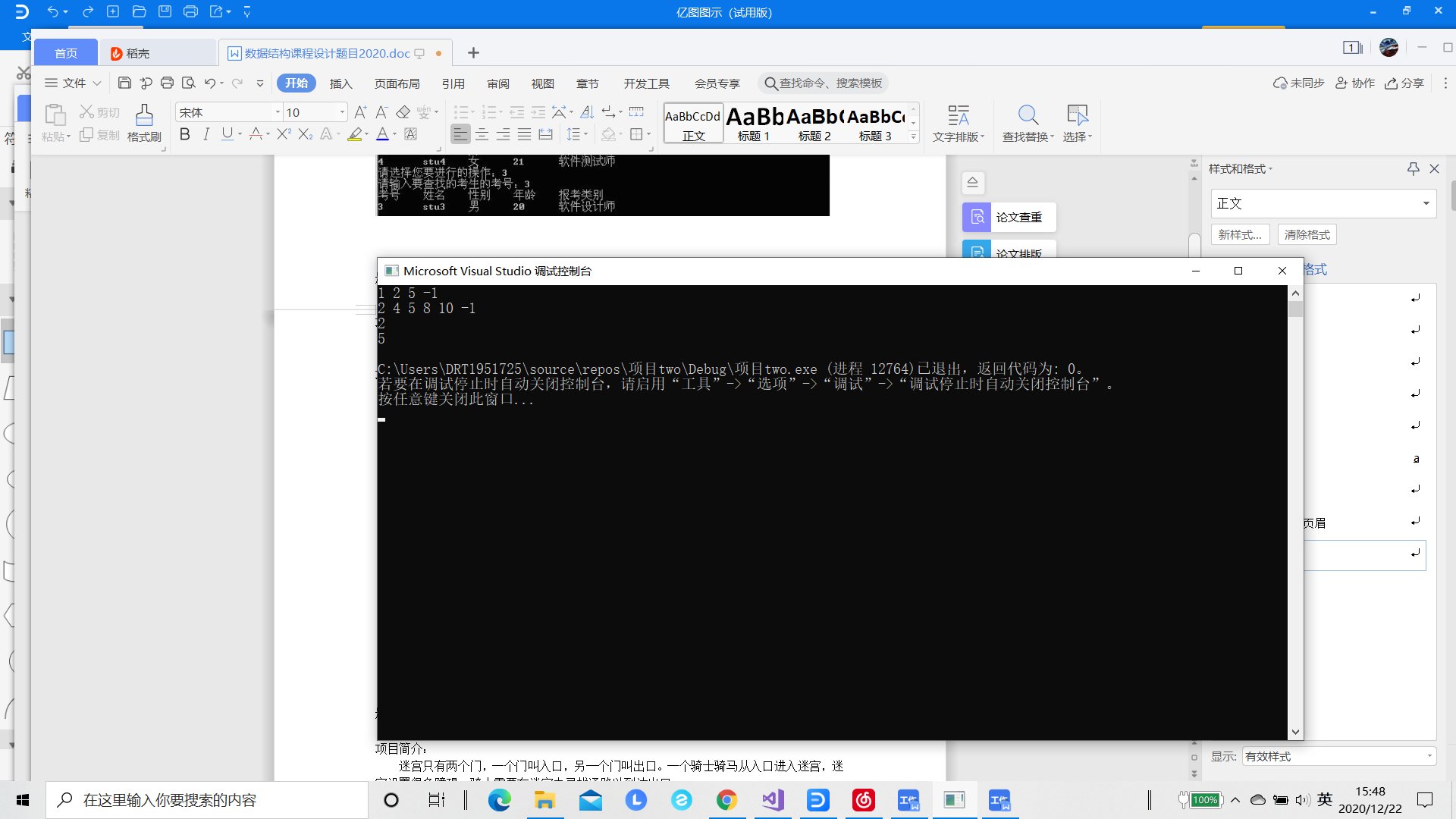
return 0;

}

# 4 测试

## 4.1 功能测试

一般情况：



## 4.2 边界测试

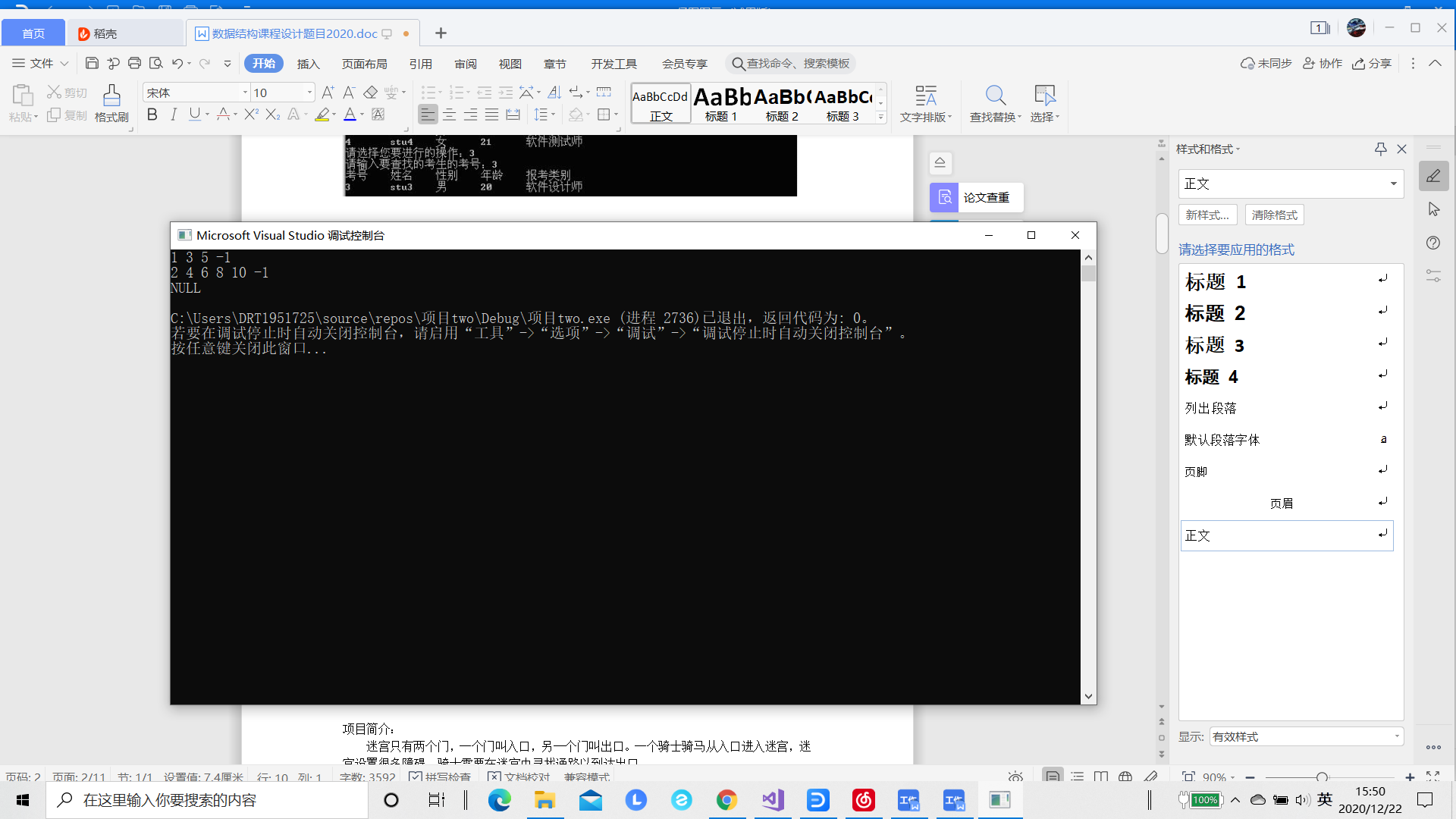
### 4.2.1 交集为空的情况

**测试用例：1 3 5 -1**

**2 4 6 8 10 -1**

**预期结果：NULL**。

**实验结果：**



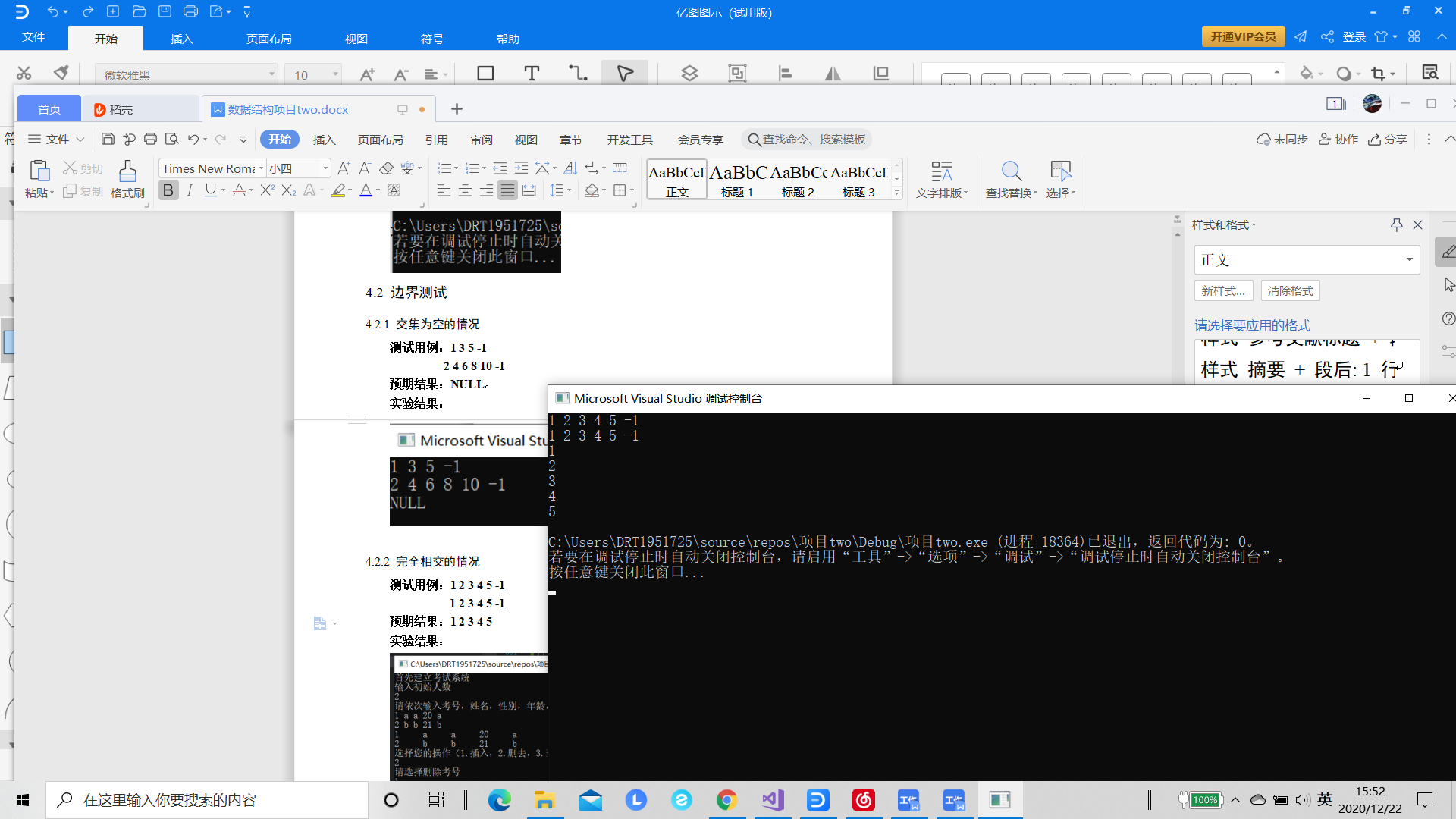
### 4.2.2 完全相交的情况

**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：**



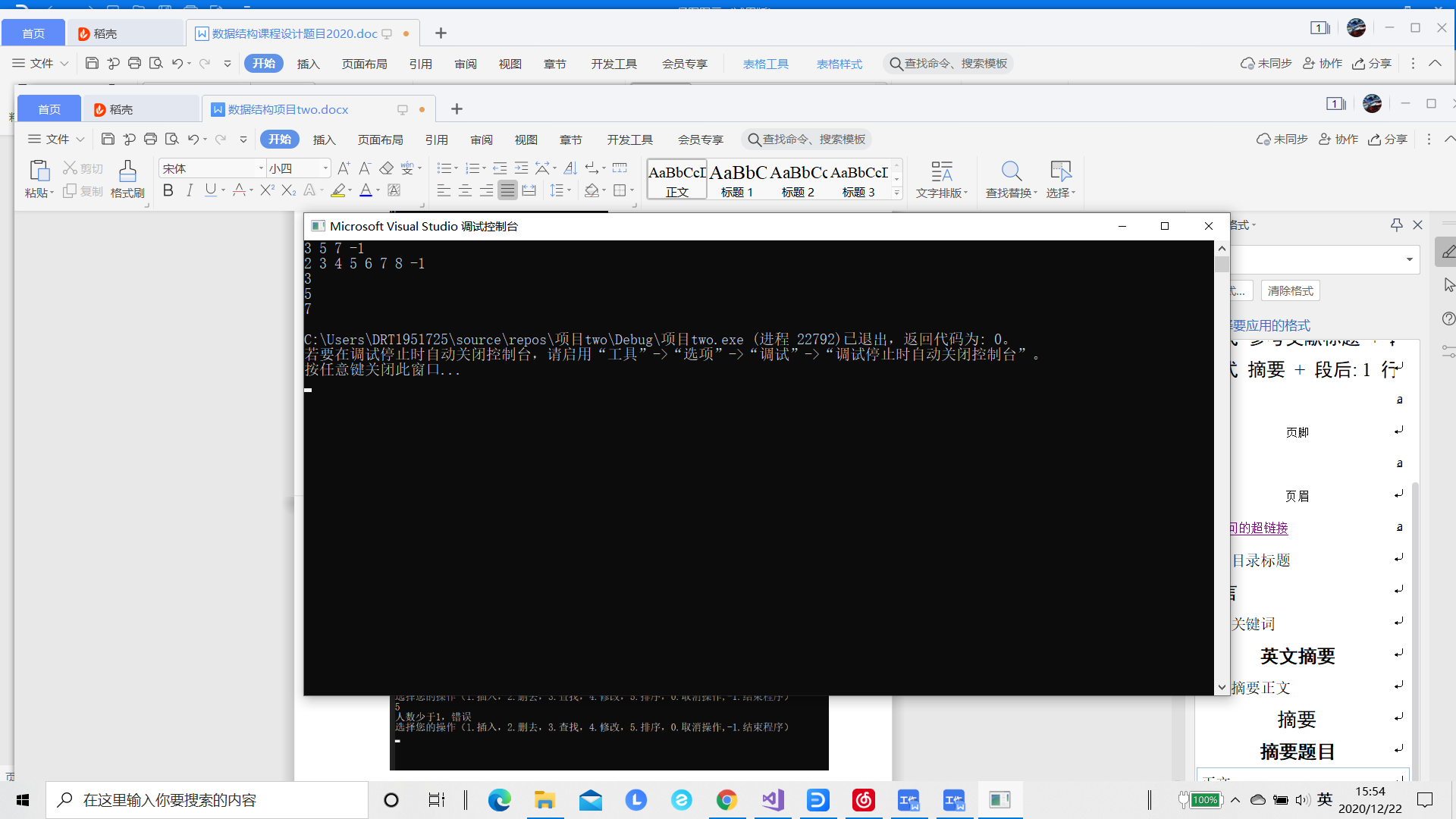
### 4.2.3 其中一个序列完全属于交集的情况

**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：**



### 4.2.4 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：**

