项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 邓若涛

学 号： 1951725

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](#_Toc7284)

[1.1 背景分析 1](#_Toc23142)

[1.2 功能分析 1](#_Toc4794)

[2 设计 1](#_Toc29874)

[2.1 数据结构设计 1](#_Toc18850)

[2.2 类结构设计 2](#_Toc18238)

[2.3 成员与操作设计 2](#_Toc21076)

[2.4 系统设计 3](#_Toc27672)

[3 实现 3](#_Toc21291)

[3.1 输入功能的实现 3](#_Toc15322)

[3.1.1 输入功能流程图 3](#_Toc24456)

[3.1.2 输入功能核心代码 3](#_Toc29950)

[3.1.3 输入功能截屏示例 4](#_Toc9739)

[3.2 插入功能的实现 5](#_Toc9670)

[3.2.1 插入功能流程图 5](#_Toc30820)

[3.2.2 插入功能核心代码 6](#_Toc28110)

[3.2.3 插入功能截屏示例 7](#_Toc28531)

[3.3 删除功能的实现 7](#_Toc23215)

[3.3.1 删除功能流程图 7](#_Toc10527)

[3.3.2 删除功能核心代码 8](#_Toc8852)

[3.3.3 删除功能截图示例 9](#_Toc8030)

[3.4 查找功能的实现 10](#_Toc1351)

[3.4.1 查找功能流程图 10](#_Toc1725)

[3.4.2 查找功能核心代码 10](#_Toc3126)

[3.4.3 查找功能截屏示例 11](#_Toc21327)

[3.5 修改功能的实现 12](#_Toc13508)

[3.5.1 修改功能流程图 12](#_Toc2053)

[3.5.2 统计功能核心代码 12](#_Toc17427)

[3.5.3 修改功能截屏示例 13](#_Toc17453)

[3.6 排序功能的实现 13](#_Toc16391)

[3.6.1 排序功能流程图 13](#_Toc8855)

[3.6.2 排序功能核心代码 13](#_Toc32130)

[3.6.3 排序功能截屏示例 14](#_Toc31888)

[3.7总体系统实现 15](#_Toc8777)

[3.7.1 总体系统流程图 15](#_Toc27842)

[3.7.2总体系统核心代码 16](#_Toc5135)

[4 测试 18](#_Toc7085)

[4.1 功能测试 18](#_Toc5937)

[4.1.1 插入功能测试 18](#_Toc3259)

[4.1.2 删除功能测试 19](#_Toc22091)

[4.1.3 查找功能测试 20](#_Toc5055)

[4.1.4 修改功能测试 20](#_Toc742)

[4.1.5 排序功能测试 21](#_Toc7451)

[4.2 边界测试 22](#_Toc15597)

[4.2.1 初始化人数小于1 22](#_Toc25046)

[4.2.2 删除头结点 22](#_Toc14061)

[4.2.3 删除后链表为空 23](#_Toc14691)

[4.2.4 排序时链表为空 24](#_Toc1343)

[4.3 出错测试 24](#_Toc24225)

[4.3.1 考生人数错误 24](#_Toc14215)

[4.3.2 操作码错误 25](#_Toc27101)

[4.3.3 插入位置不存在 25](#_Toc5939)

[4.3.4 删除考号不存在 26](#_Toc4059)

[4.3.5 查找考号不存在 27](#_Toc31449)

[4.3.6 修改考号不存在 27](#_Toc30256)

[4.3.7 排序中表有相同考号存在 29](#_Toc23975)

# 1 分析

## 1.1 背景分析

考试报名系统是一个学校不可缺少的部分，它对于学校的管理者和学生来说都至关重要，所以一个良好的考试报名系统应该能够为用户提供充足的信息和功能。考试报名系统对于学校加强考试管理有极其重要的作用。随着学生数量和考试数量的日益庞大，如何管理如此庞大的数据显得极为复杂，传统的手工管理工作量大且容易出错。

## 1.2 功能分析

作为一个最简易的考试报名系统，首先应该有的功能就是输入同学们的考试报名情况并且可以予以显示。其次，考试报名系统还应该具有插入、删除、修改功能，以保证同学可以随时更改自己的考试报名情况。同时，为了方便纪录按考号的升序进行排列，并可以检查时否有相同的考号出现。最后，考试报名系统软件还应该确保软件可以正常关闭。

综上所述，一个考试报名系统至少应该具有输入、输出、插入、删除、修改、排列检查，退出的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求大量的增加、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

**链表结点类（date）**

**公有成员：**

int number; //表示学生学号

char name[20]; //姓名

char gender[20];//性别

int age;//年龄

char job[20];//报考类别

date\*link;//指针部分

**链表类（list）**

**私有成员：**

date\* head; //头指针

date\* end; //尾指针

date\* current; //现指针

date\* temp; //临时指针

int firstsetnumber; //第一次建立的人数

int nowsetnumber; //现在表内的人数

**公有操作：**

list(int n);//构造函数

bool input();//输入函数

bool output();//输出函数

void relist();//按学号重排列函数

bool insert(int n);//插入到n位函数

bool remove(int n);//删去n位函数

bool seek(int n);//寻找学号为n的函数

bool change(int n);//修改学号为n的函数

## void check();//判断是否有重复学号出现

## 2.4 系统设计

系统首先调用input（）函数实现对屏幕的初始化，完成对链表L的创建和输入数据工作，然后根据用户所输入的操作码执行链表L对应的成员函数。

# 3 实现

## 3.1 输入功能的实现

### 3.1.1 输入功能流程图

判断输入的个数是否大于等于1 否 退出

是

构建头指针并依次建立对应个数

的数据组

### 3.1.2 输入功能核心代码

bool list::input()

{

if (nowsetnumber >= 1)

{

current = new date;

cin >> current->number;

cin >> current->name;

cin >> current->gender;

cin >> current->age;

cin >> current->job;

head->link = current;

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

{

temp = new date;

cin >> temp->number;

cin >> temp->name;

cin >> temp->gender;

cin >> temp->age;

cin >> temp->job;

current->link = temp;

current = temp;

if (n == nowsetnumber) { current->link = NULL; end = current; current = head->link; }

}

return true;

}

else

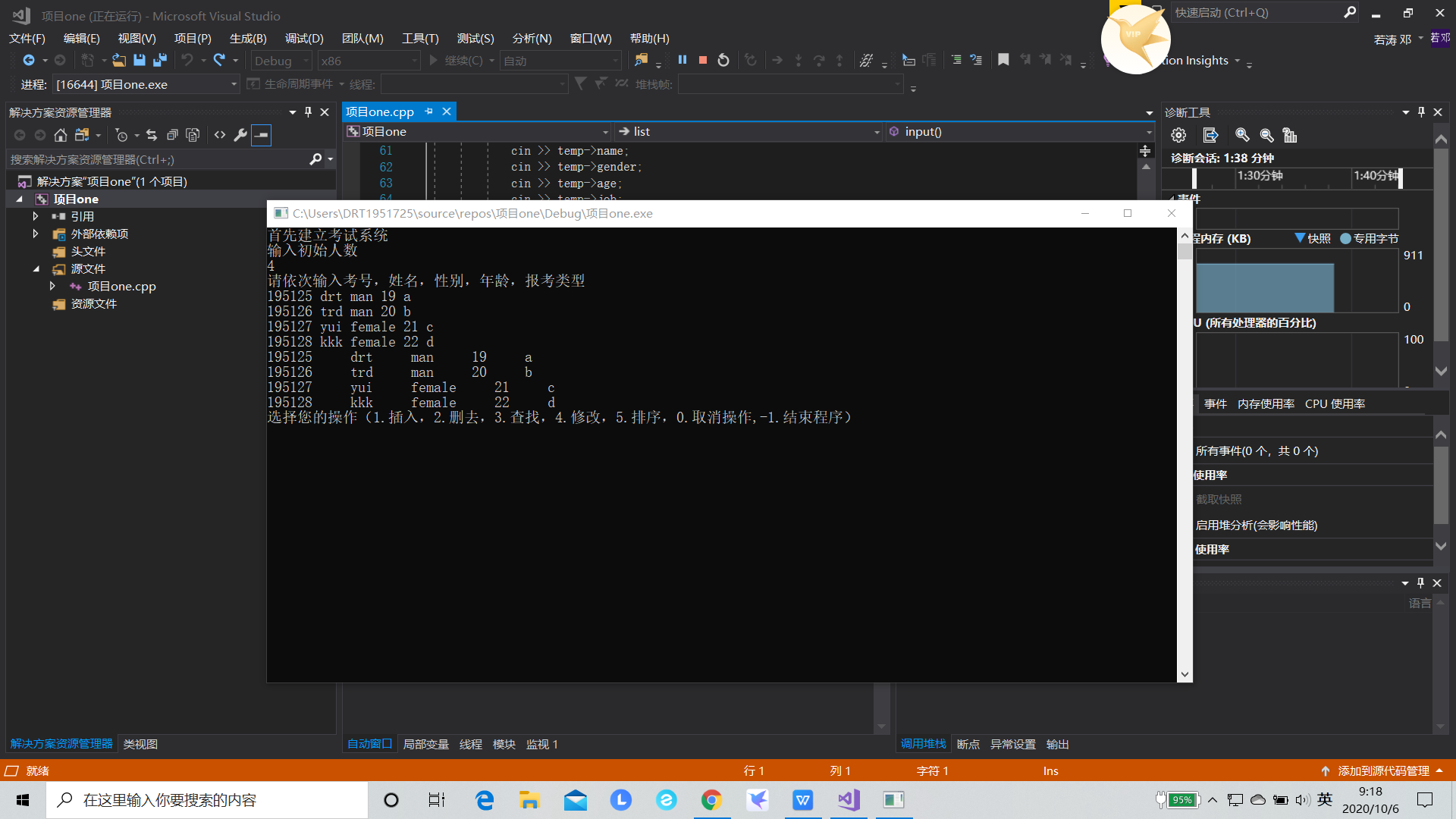
{

return false;

}

}

### 3.1.3 输入功能截屏示例



## 3.2 插入功能的实现

### 3.2.1 插入功能流程图

### 

### 3.2.2 插入功能核心代码

bool list::insert(int n)

{

current = head;

temp = new date;

if (temp == NULL) { cout << "插入时，空间不足" << endl; return false; }

cin >> temp->number;

cin >> temp->name;

cin >> temp->gender;

cin >> temp->age;

cin >> temp->job;

nowsetnumber++;

if (n > nowsetnumber) { cout << "插入空间有空隙" << endl; return false; }

if (n <= 0) { cout << "插入位置小于1" << endl; return false; }

for (int m = 0; m <= n - 1; m++)

{

if (m == n - 1)

{

temp->link = current->link;

current->link = temp;

if (temp->link == NULL) { end = temp; }

}

else

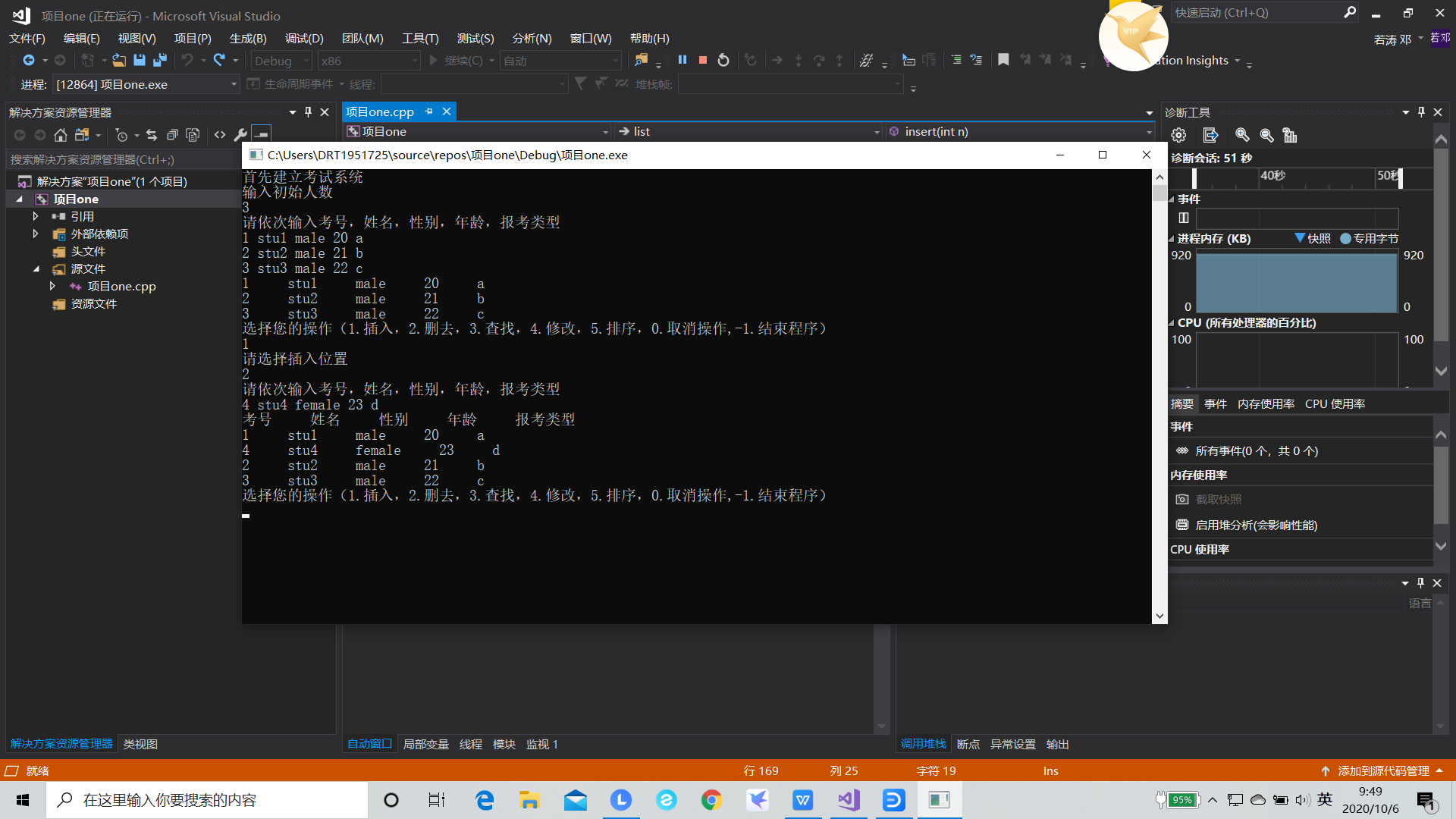
{current = current->link;}

}

return true;

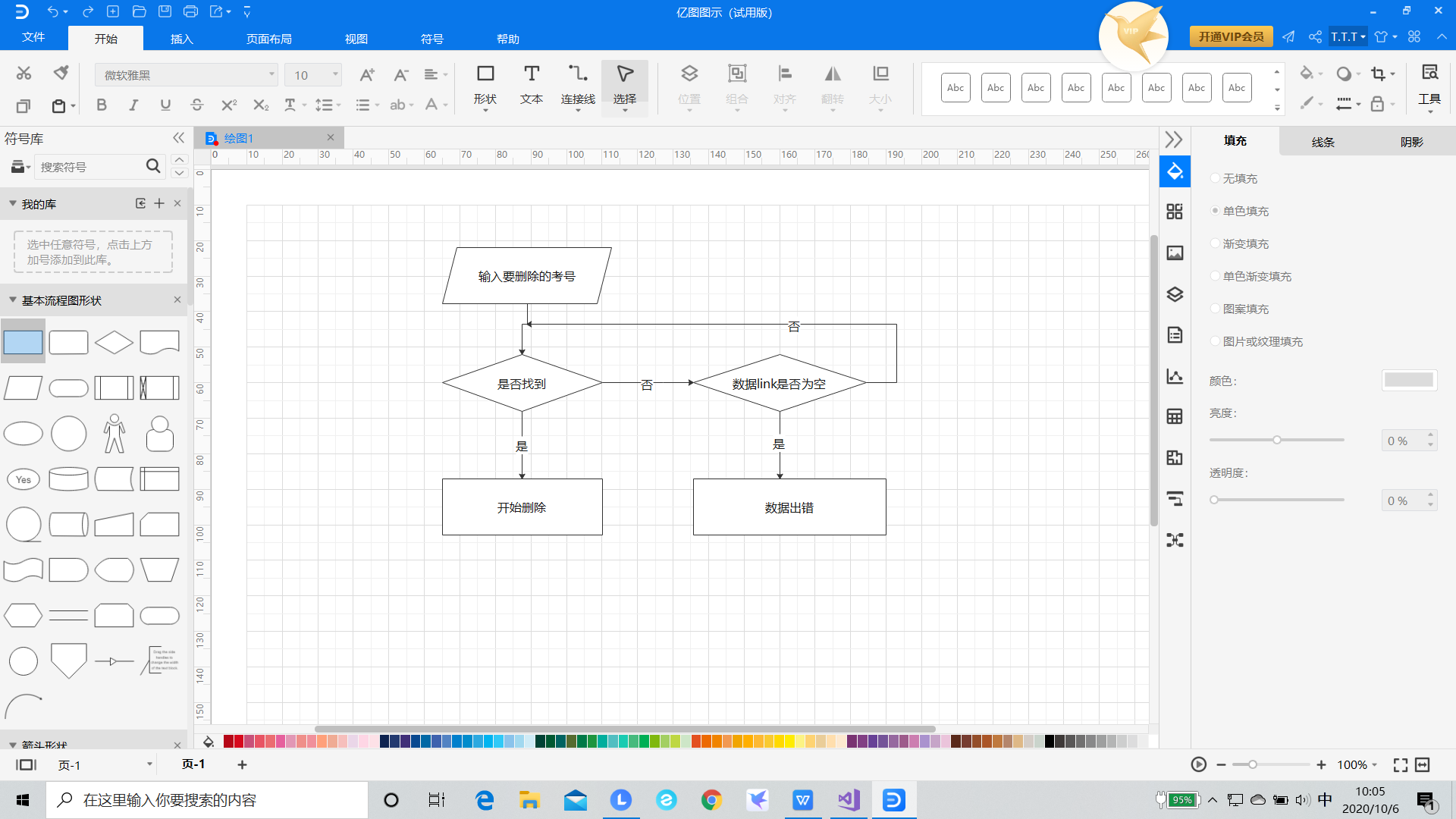
### }

### 3.2.3 插入功能截屏示例



## 3.3 删除功能的实现

### 3.3.1 删除功能流程图



### 3.3.2 删除功能核心代码

bool list::remove(int n)

{

int location=0;

current = head;

temp = new date;

if (temp == NULL) { cout << "删除时，空间不足" << endl; return false; }

for (int m = 1; m <= nowsetnumber; m++)

{

if (current->link == NULL) { break; }

if (current->link->number != n) { current = current->link; }

else

{

location = m;

temp = current->link;

current->link = temp->link;

cout << "您要删除的信息是" << endl;

cout << temp->number;

cout << " ";

cout << temp->name;

cout << " ";

cout << temp->gender;

cout << " ";

cout << temp->age;

cout << " ";

cout << temp->job;

cout<< endl;

delete temp; nowsetnumber--;

if (current->link == NULL) { end = current; }

break;

}

}

//location代表位置，current指向前一个

if (location == 0)

{

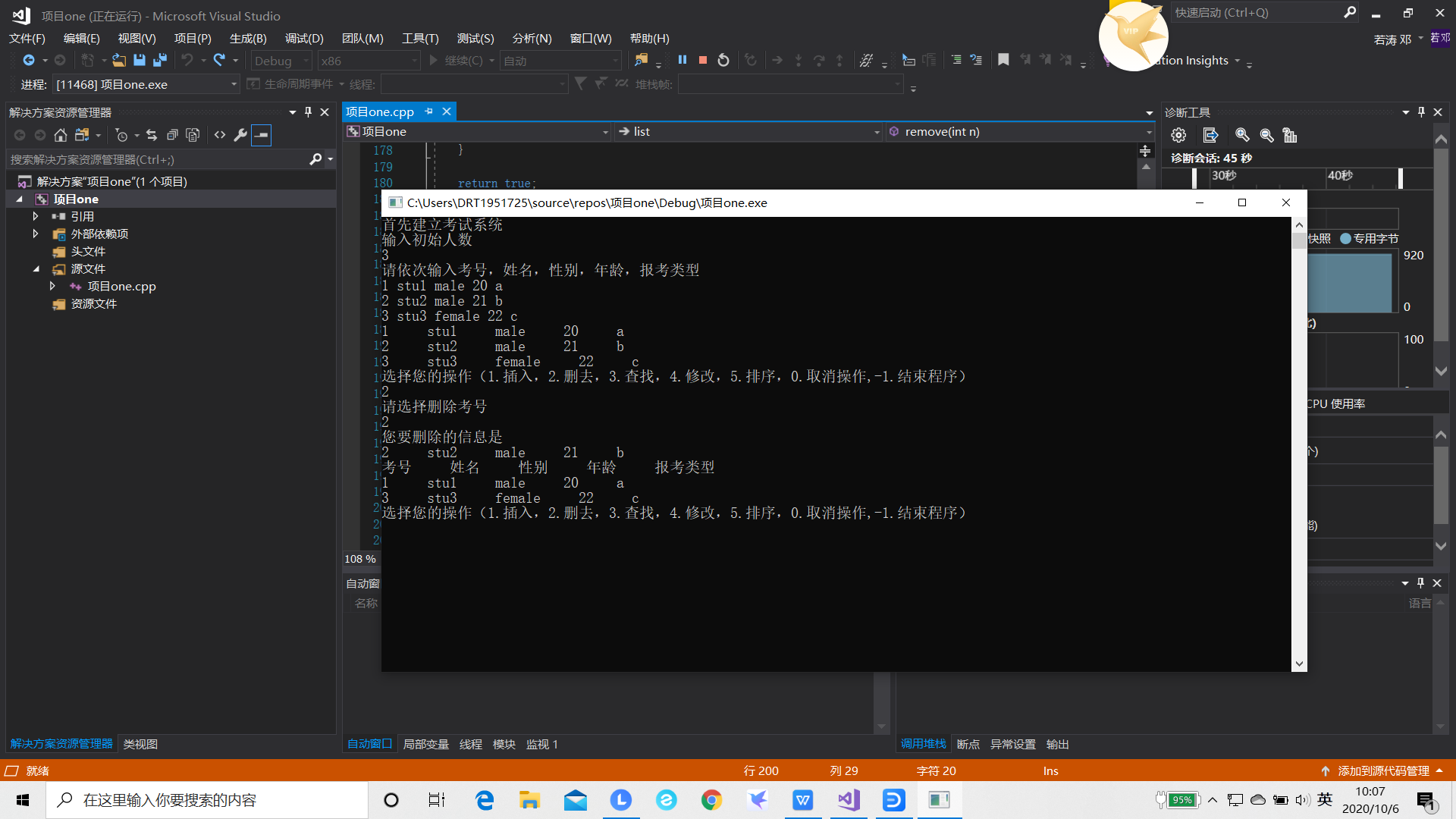
cout << "未找到合适学号" << endl; return false;

}

return true;

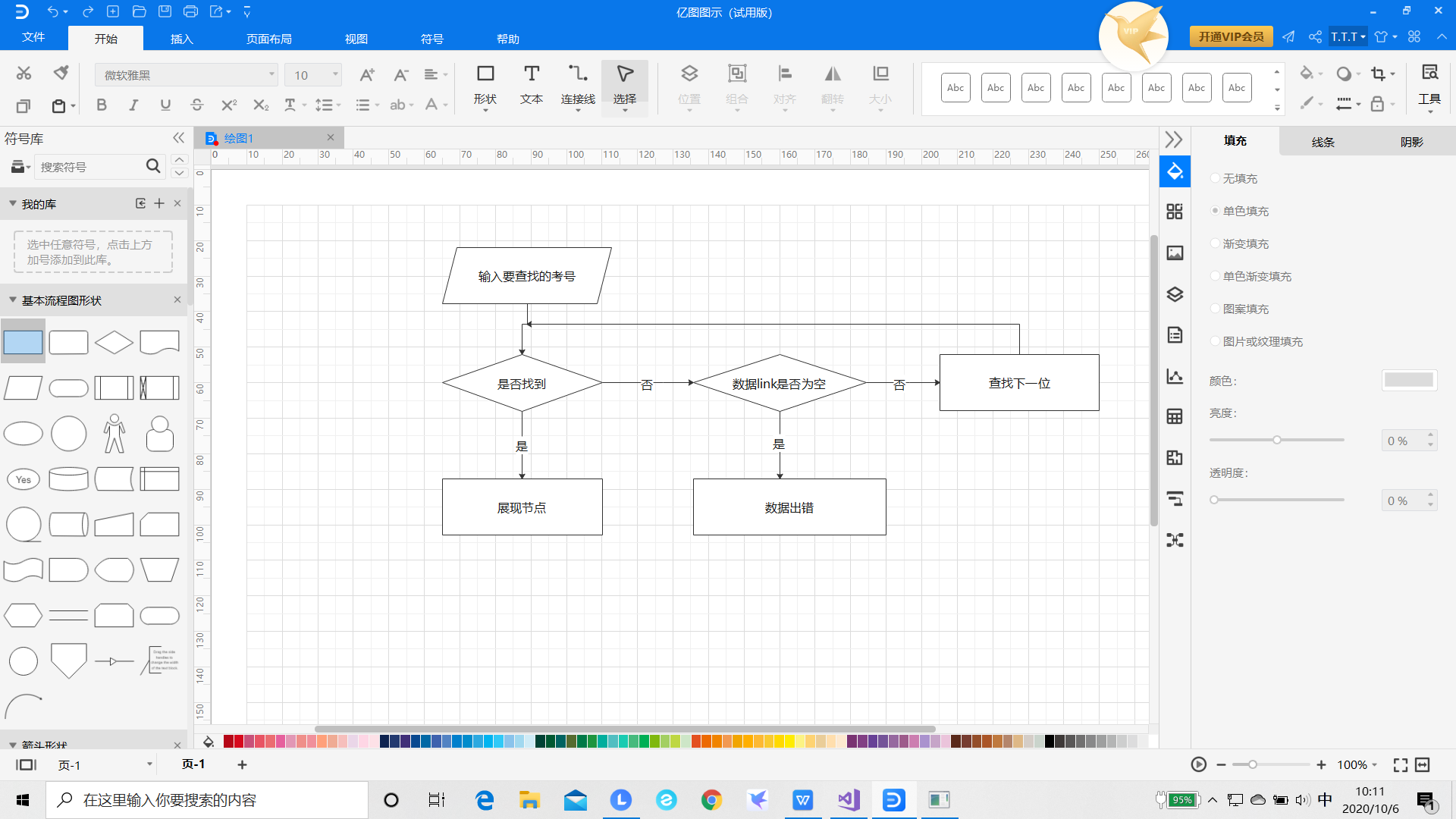
### }

### 3.3.3 删除功能截图示例



## 3.4 查找功能的实现

### 3.4.1 查找功能流程图



### 3.4.2 查找功能核心代码

bool list::seek(int n)

{

current = head->link;

for (int m = 1; m <= nowsetnumber; m++)

{

if (current->number == n)

{

cout << "输出学生";

cout << current->number;

cout << " ";

cout << current->name;

cout << " ";

cout << current->gender;

cout << " ";

cout << current->age;

cout << " ";

cout << current->job;

cout << endl;

return true;

}

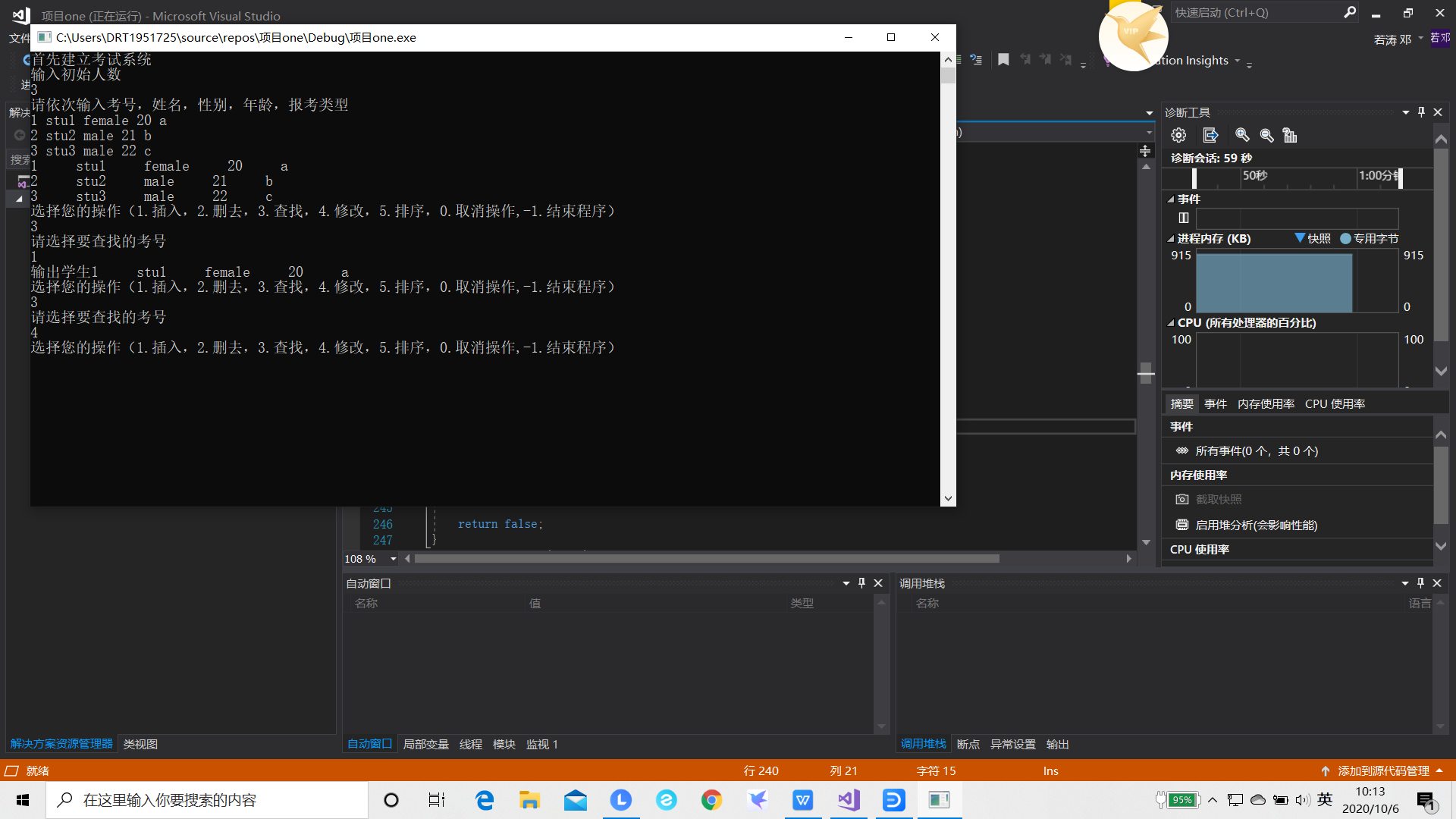
else { current = current->link; }

}

return false;

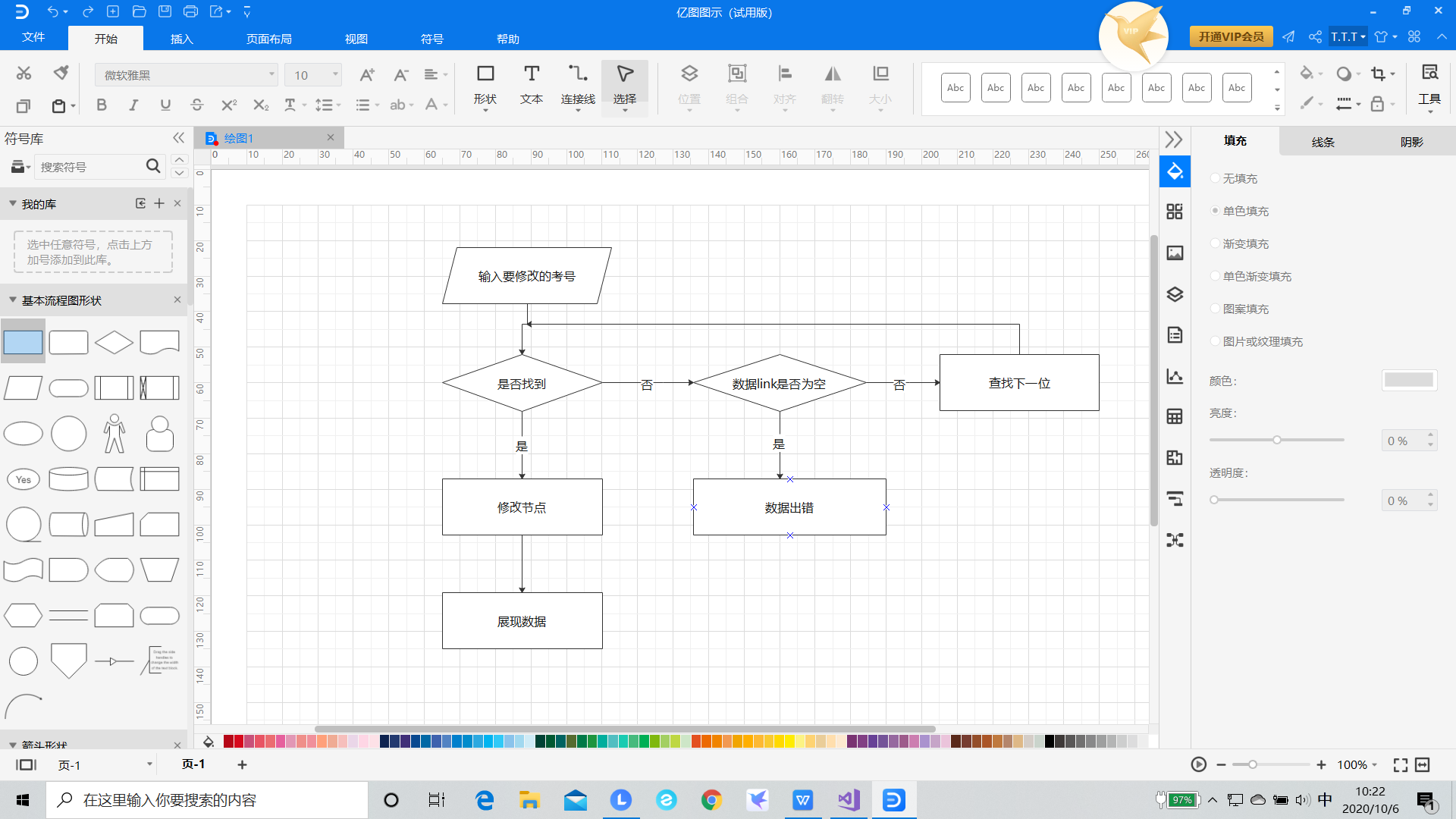
### }

### 3.4.3 查找功能截屏示例



## 3.5 修改功能的实现

### 3.5.1 修改功能流程图



### 3.5.2 统计功能核心代码

bool list::change(int n)

{

current = head->link;

for (int m = 1; m <= nowsetnumber; m++)

{

if (current->number == n)

{

int a; cin >> a; current->number = a;

char b[20]; cin >> b; strcpy(current->name,b);

char c[20]; cin >> c; strcpy(current->gender, c);

int d; cin >> d; current->age = d;

char e[20]; cin >> e; strcpy(current->job, e);

return true;

}

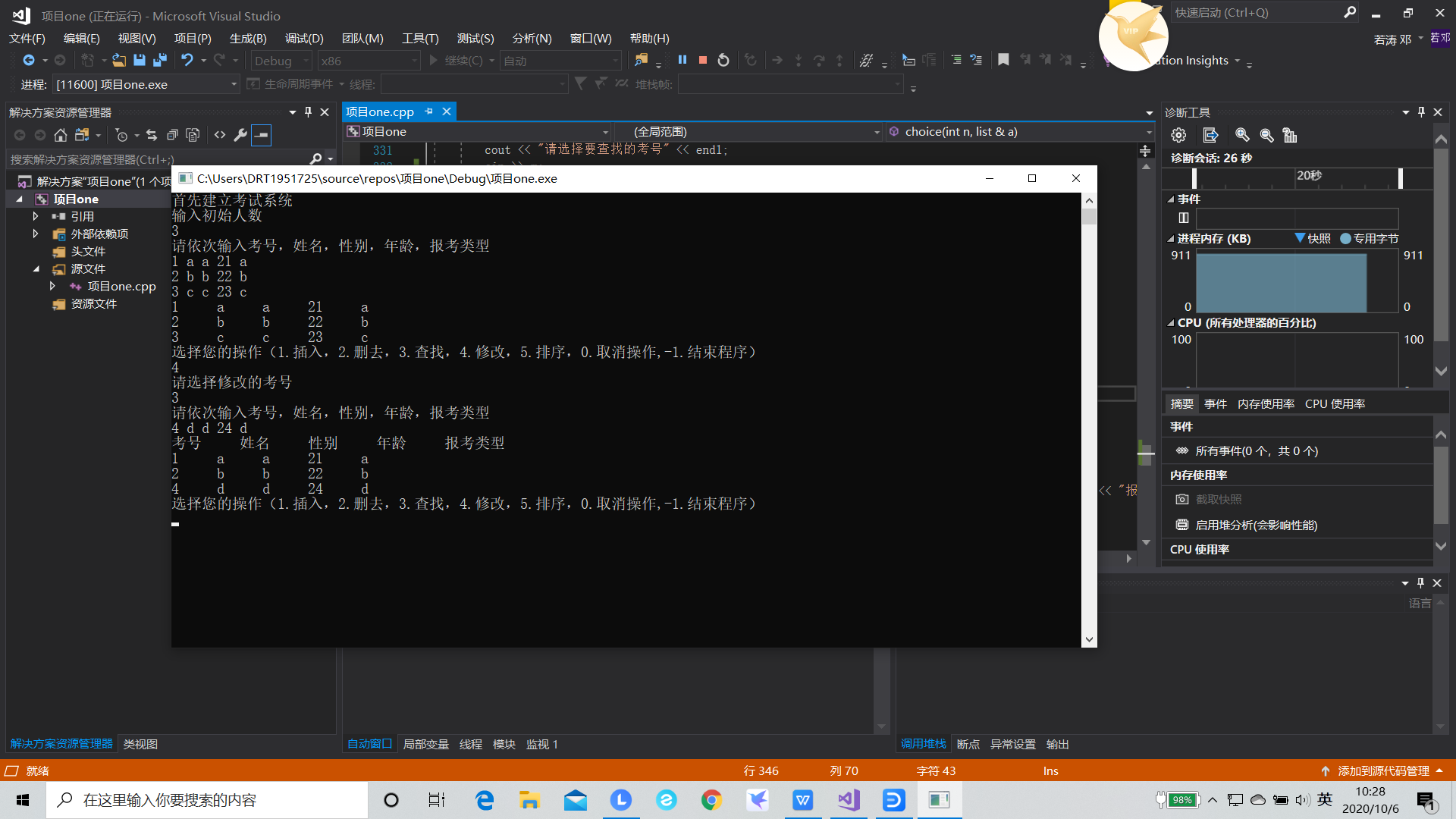
else { current = current->link; }

}

return false;

### }

### 3.5.3 修改功能截屏示例



## 3.6 排序功能的实现

### 3.6.1 排序功能流程图

采用冒泡法

### 3.6.2 排序功能核心代码

void list::relist()

{

if (nowsetnumber < 1) { cout << "人数少于1，错误" << endl; }

else

{

for (int m = 1; m < nowsetnumber; m++)

{

current = head->link;

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

{

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

{

temp = new date;

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

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

current->number = temp->number;

strcpy(temp->name, current->link->name);

strcpy(current->link->name, current->name);

strcpy(current->name, temp->name);

strcpy(temp->gender, current->link->gender);

strcpy(current->link->gender, current->gender);

strcpy(current->gender, temp->gender);

temp->age = current->link->age;

current->link->age = current->age;

current->age = temp->age;

strcpy(temp->job, current->link->job);

strcpy(current->link->job, current->job);

strcpy(current->job, temp->job);

delete temp;

current = current->link;

}

else { current = current->link; }

}

}

}

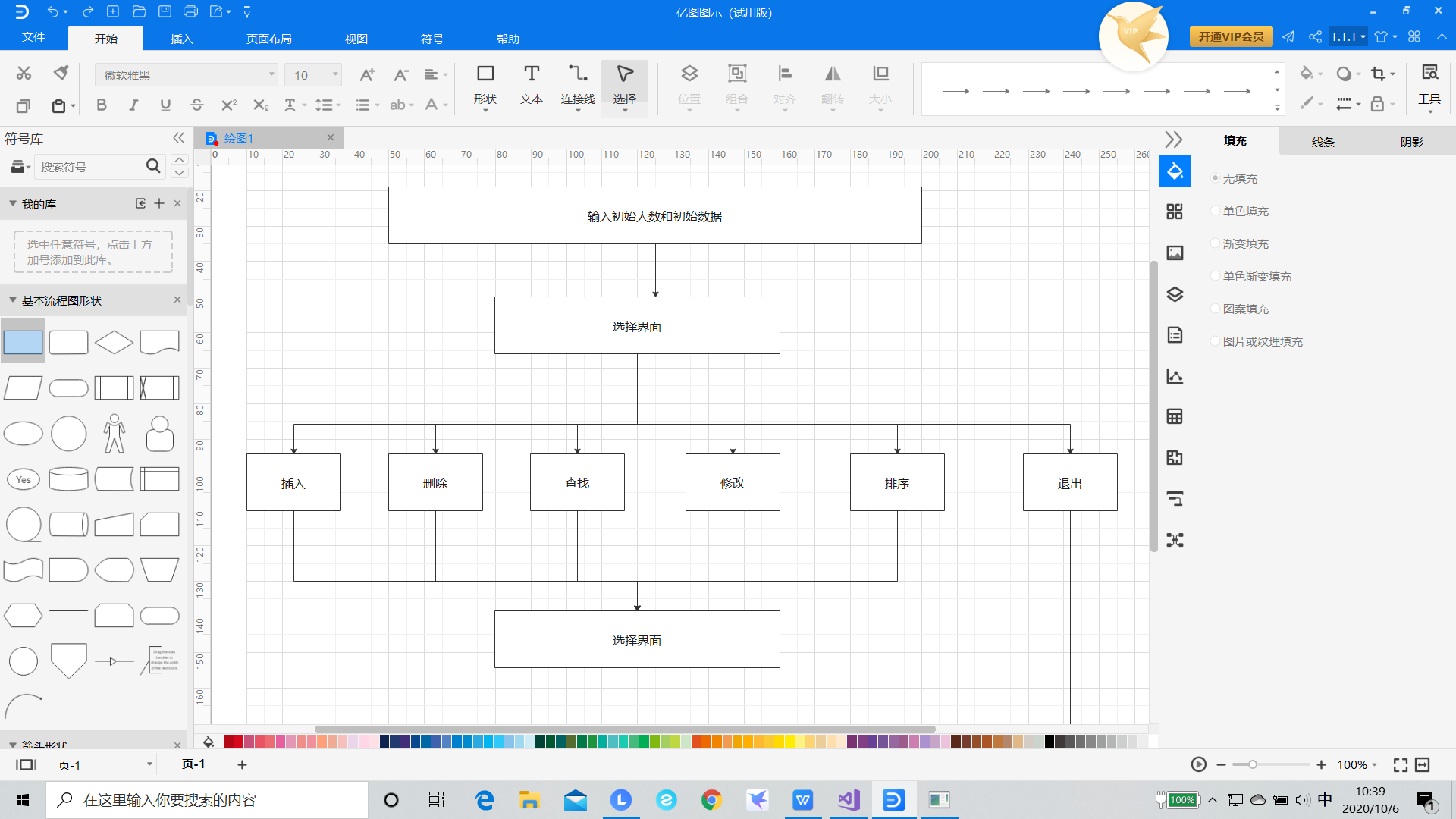
### }

### 3.6.3 排序功能截屏示例

# 

# 3.7总体系统实现

### 3.7.1 总体系统流程图



### 3.7.2总体系统核心代码

void choice(int n,list &a)

{

switch (n)

{

case 1:

{

int m;

cout << "请选择插入位置" << endl;

cin >> m;

cout << "请依次输入考号，姓名，性别，年龄，报考类型" << endl;

a.insert(m);

cout << "考号" << " " << "姓名" << " " << "性别" << " " << "年龄" << " " << "报考类型" << " " << endl;

a.output();

break;

}

case 2:

{

int m;

cout << "请选择删除考号" << endl;

cin >> m;

a.remove(m);

cout << "考号" << " " << "姓名" << " " << "性别" << " " << "年龄" << " " << "报考类型" << " " << endl;

a.output();

break;

}

case 3:

{

int m;

cout << "请选择要查找的考号" << endl;

cin >> m;

int b;

b = a.seek(m);

if (b == 0)

{

cout << "表中无该考生数据" << endl;

}

break;

}

case 4:

{

int m;

cout << "请选择修改的考号" << endl;

cin >> m;

cout << "请依次输入考号，姓名，性别，年龄，报考类型" << endl;

int b;

b=a.change(m);

if (b == 1)

{

cout << "考号" << " " << "姓名" << " " << "性别" << " " << "年龄" << " " << "报考类型" << " " << endl;

a.output();

}

else

{

cout << "表中无该考生数据" << endl;

}

break;

}

case 5:

{

a.relist();

a.output();

a.check();

break;

}

case 0:

{

break;

}

}

}

# 4 测试

## 4.1 功能测试

### 4.1.1 插入功能测试

**测试用例**：3 stu3 男 22 网络工程师

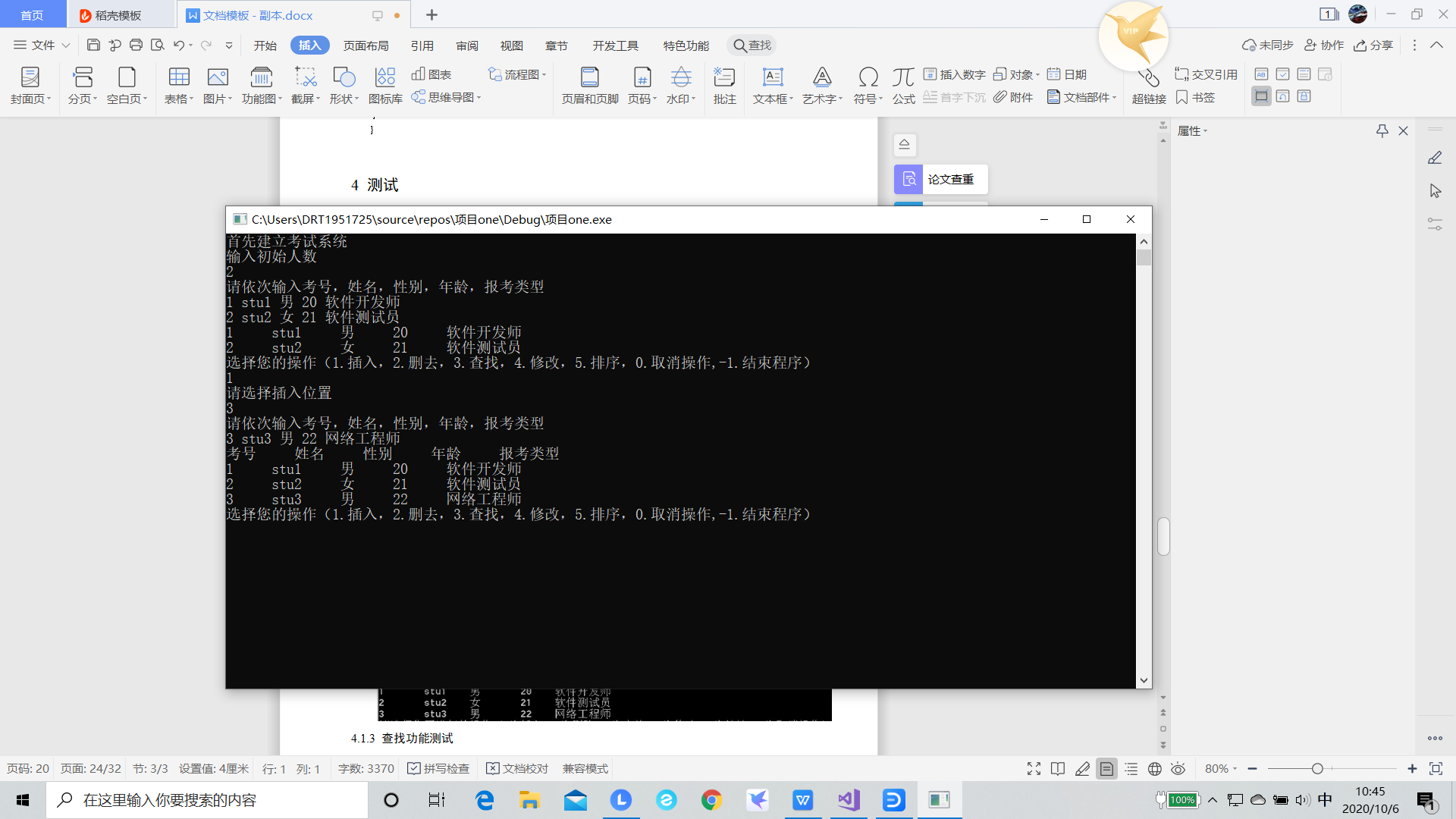
**预期结果**：

1 stu1 男 20 软件开发师

2 stu2 女 21 软件测试员

3 stu3 男 22 网络工程师

**实验结果**



### 4.1.2 删除功能测试

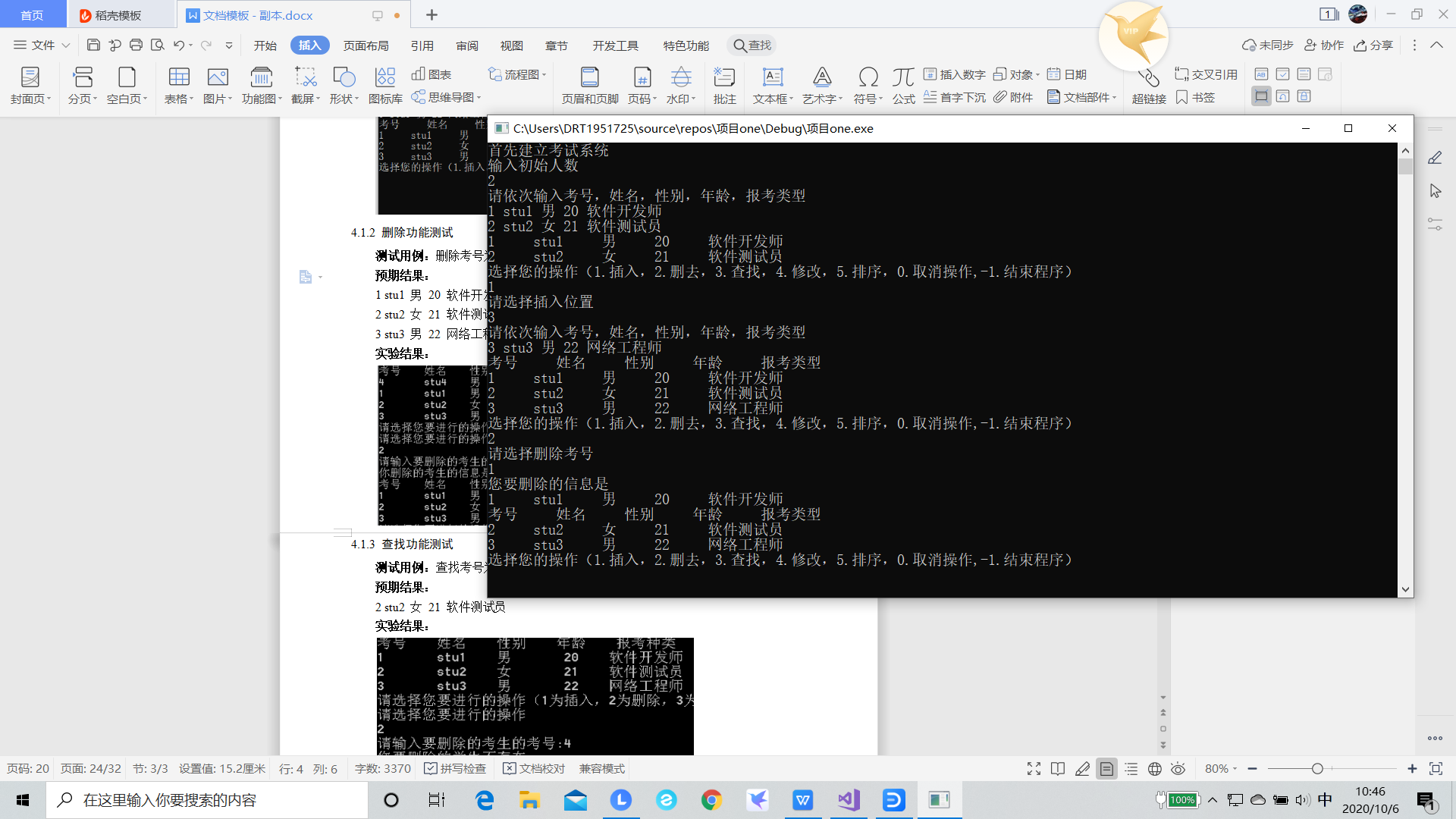
**测试用例：**删除考号为1的考生

**预期结果：**

2 stu2 女 21 软件测试员

3 stu3 男 22 网络工程师

**实验结果：**



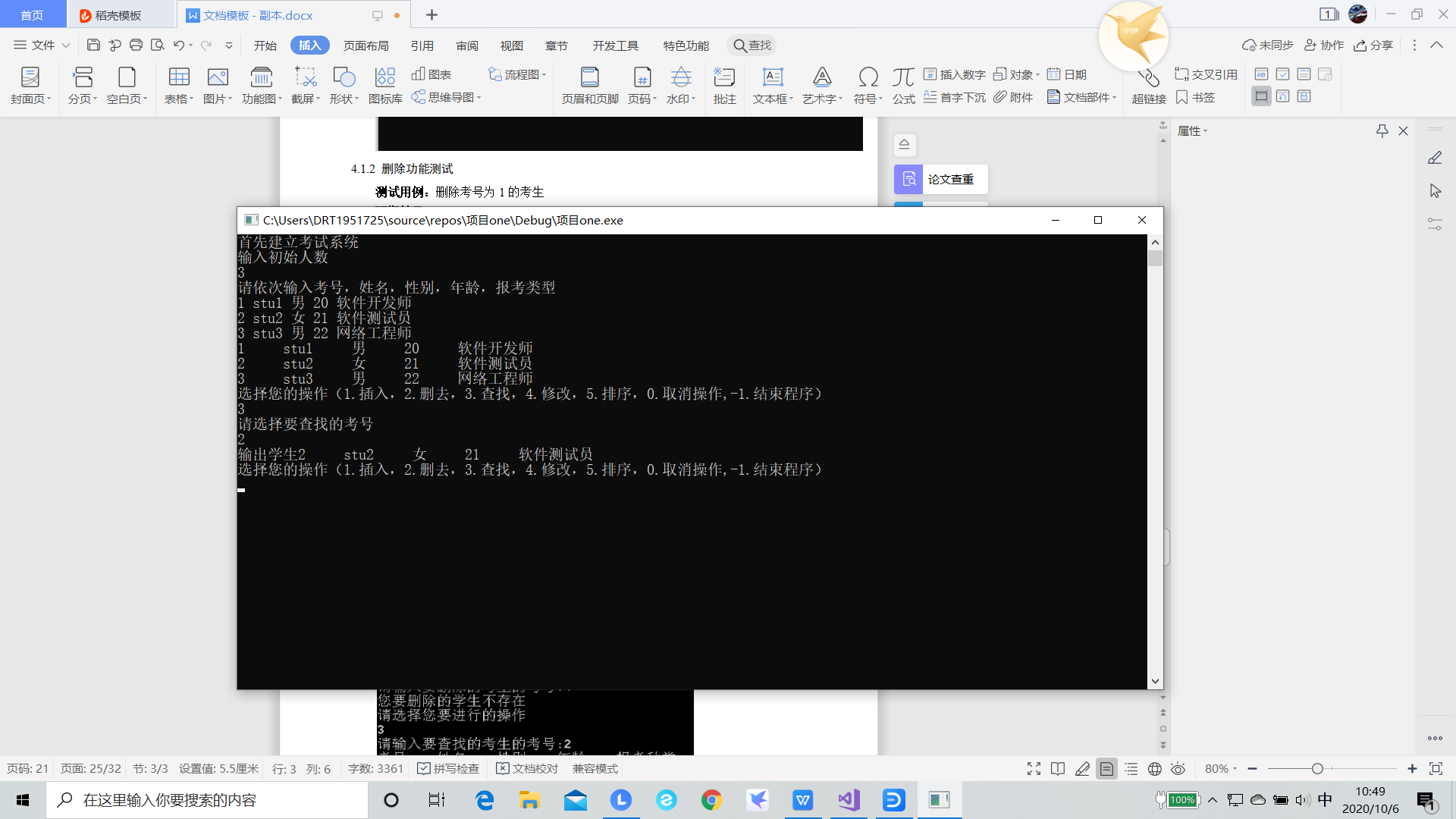
### 4.1.3 查找功能测试

**测试用例：**查找考号为2的考生

**预期结果：**

2 stu2 女 21 软件测试员

**实验结果：**



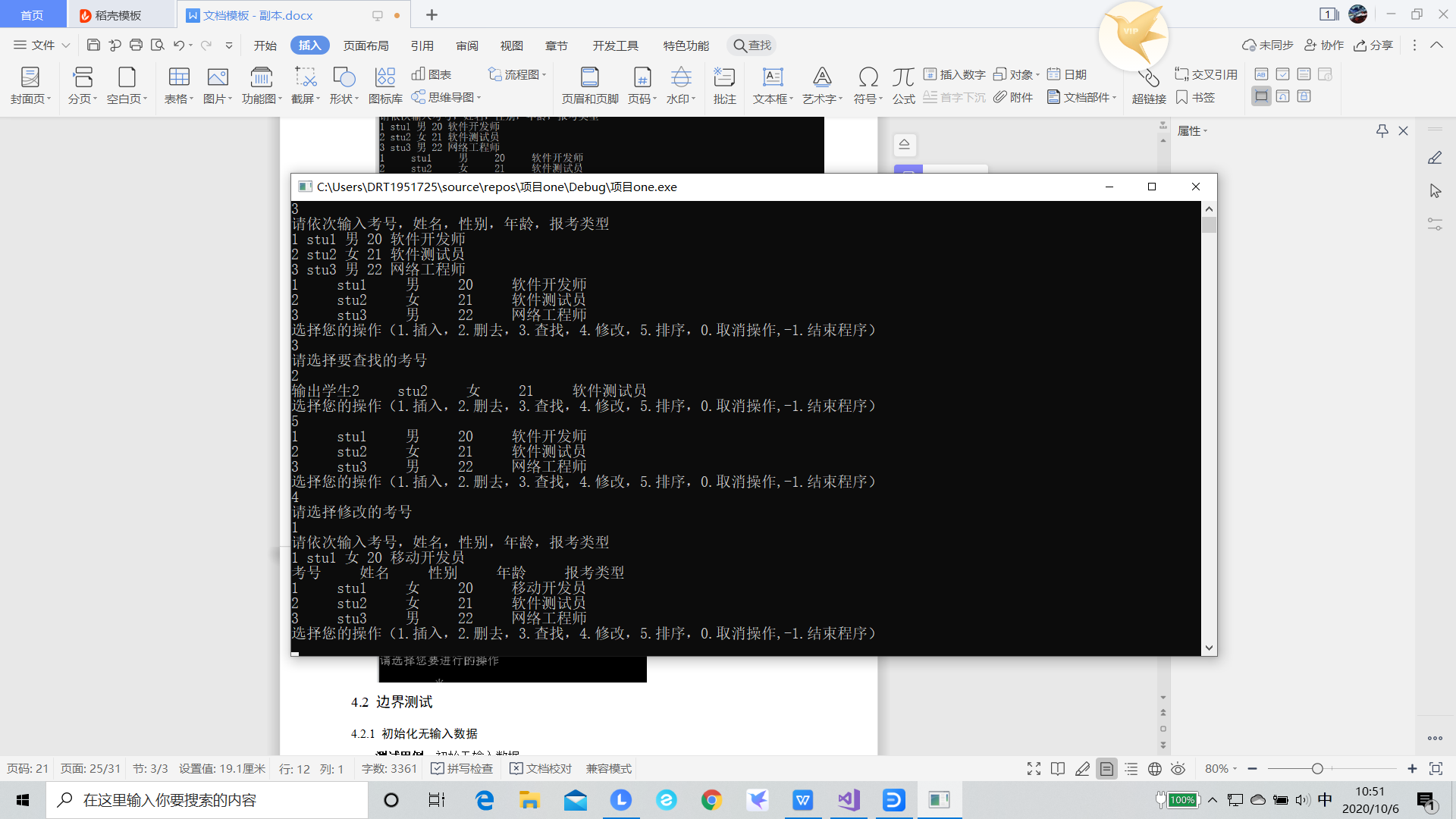
### 4.1.4 修改功能测试

**测试用例：**将考号1修改为性别女，年龄20，报考种类移动开发员。

**预期结果：**

1 stu1 女 20 移动开发员

**实验结果：**



### 4.1.5 排序功能测试

**测试用例：**

3 stu3 男 22 网络工程师

2 stu2 女 21 软件测试员

1 stu1 女 20 移动开发员

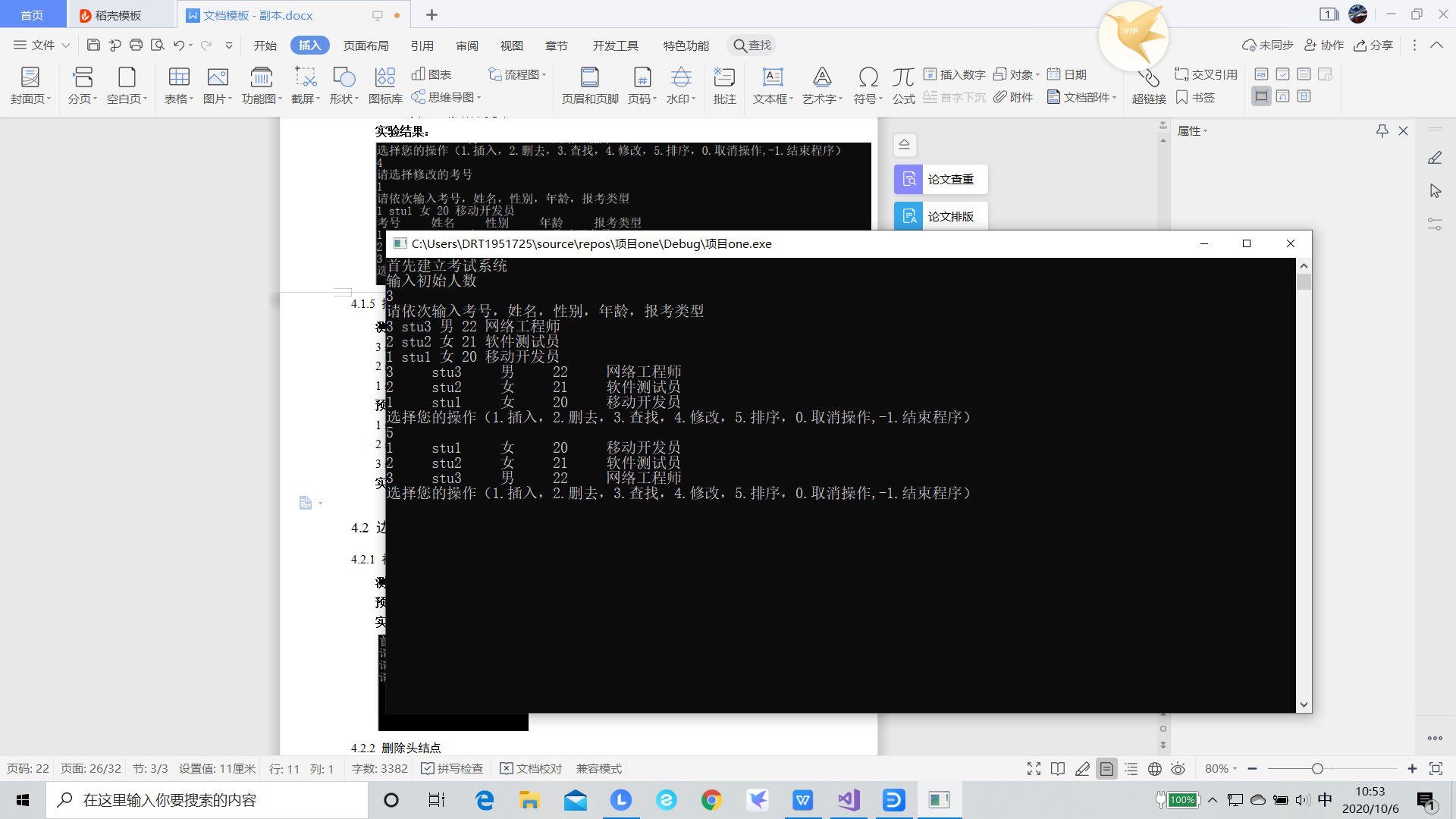
**预期结果：**

1 stu1 女 20 移动开发员

2 stu2 女 21 软件测试员

3 stu3 男 22 网络工程师

**实验结果：**



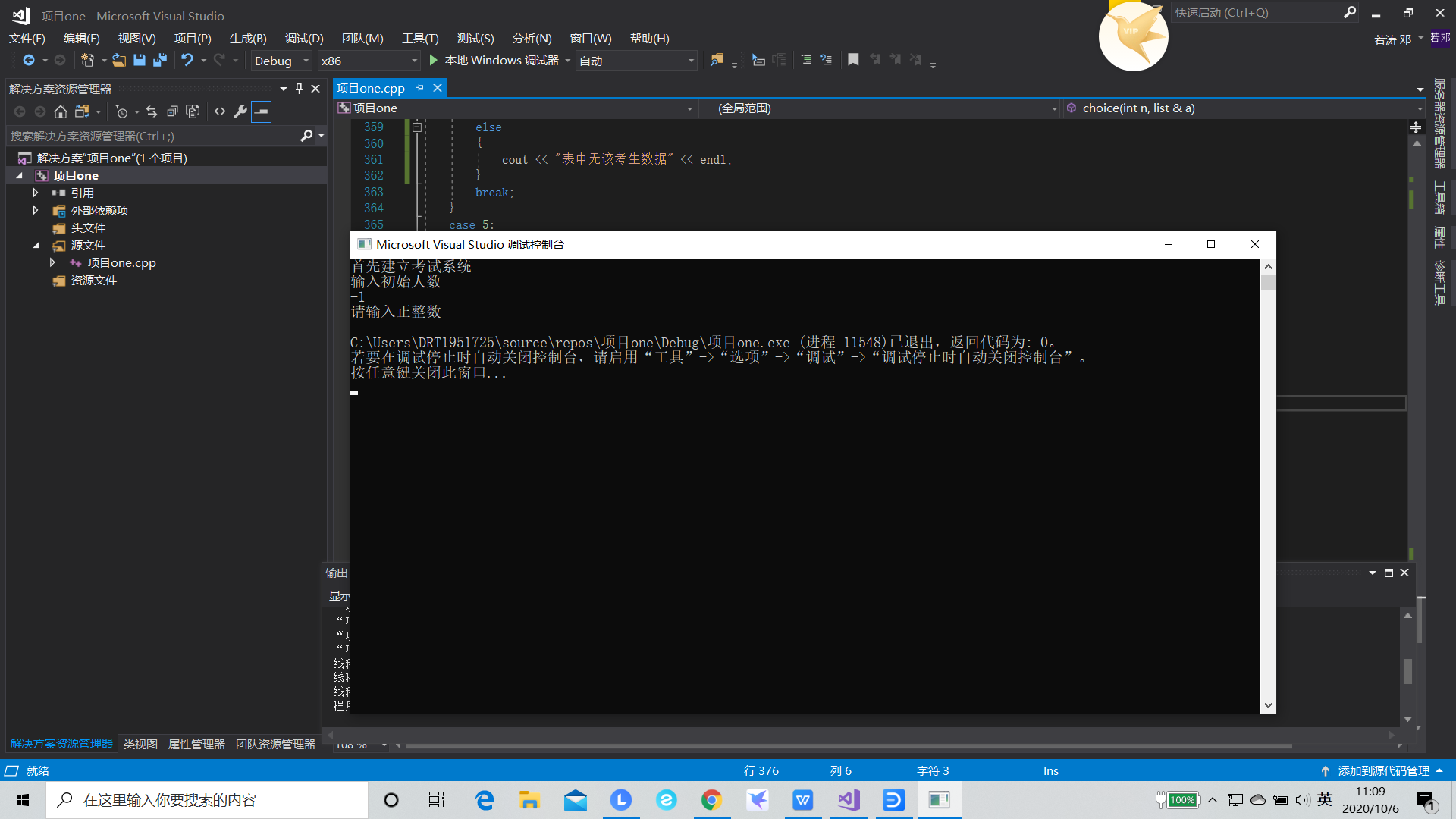
## 4.2 边界测试

### 4.2.1 初始化人数小于1

**测试用例：初始化人数小于1**

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

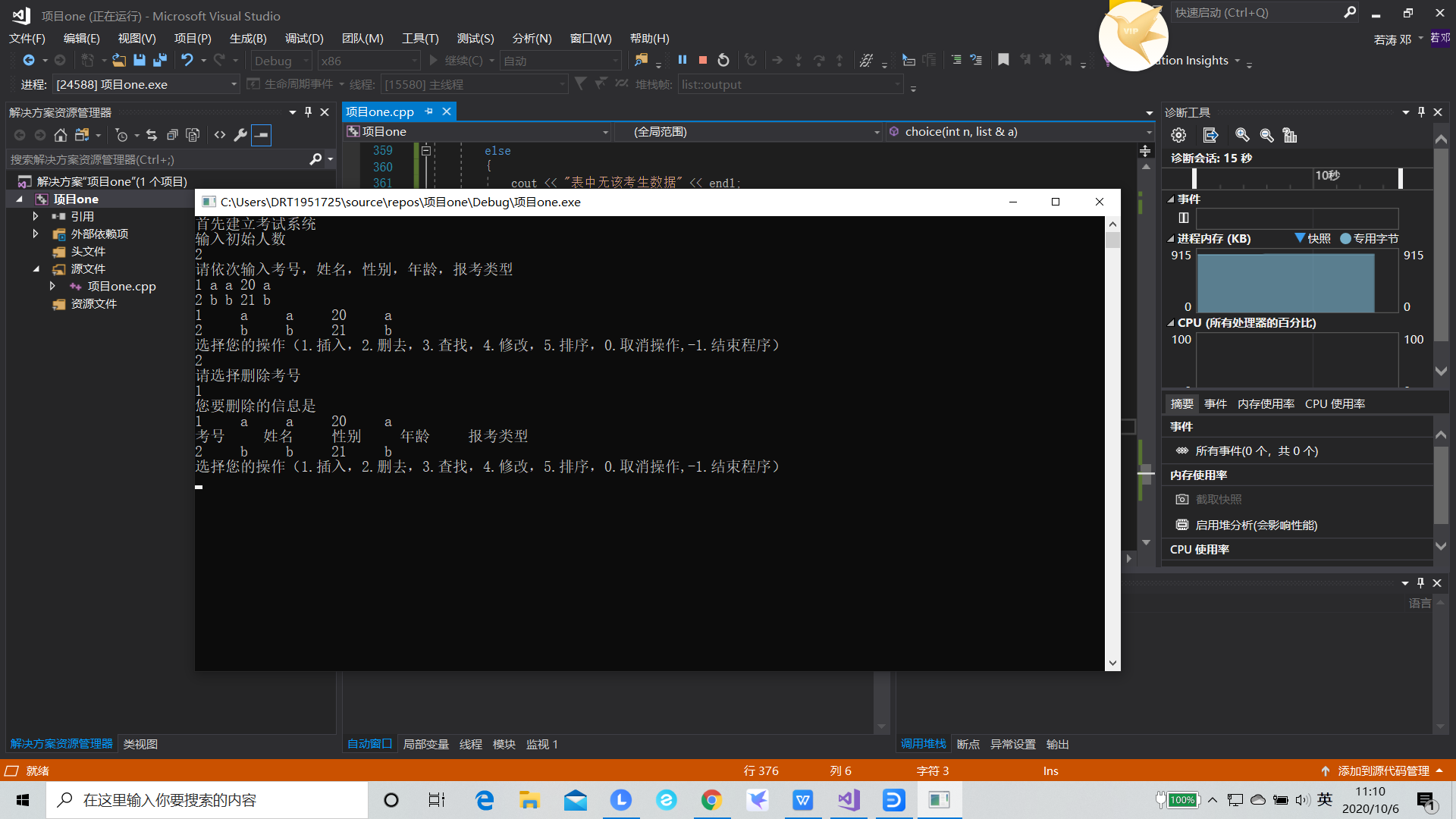


### 4.2.2 删除头结点

**测试用例：**删除头结点

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

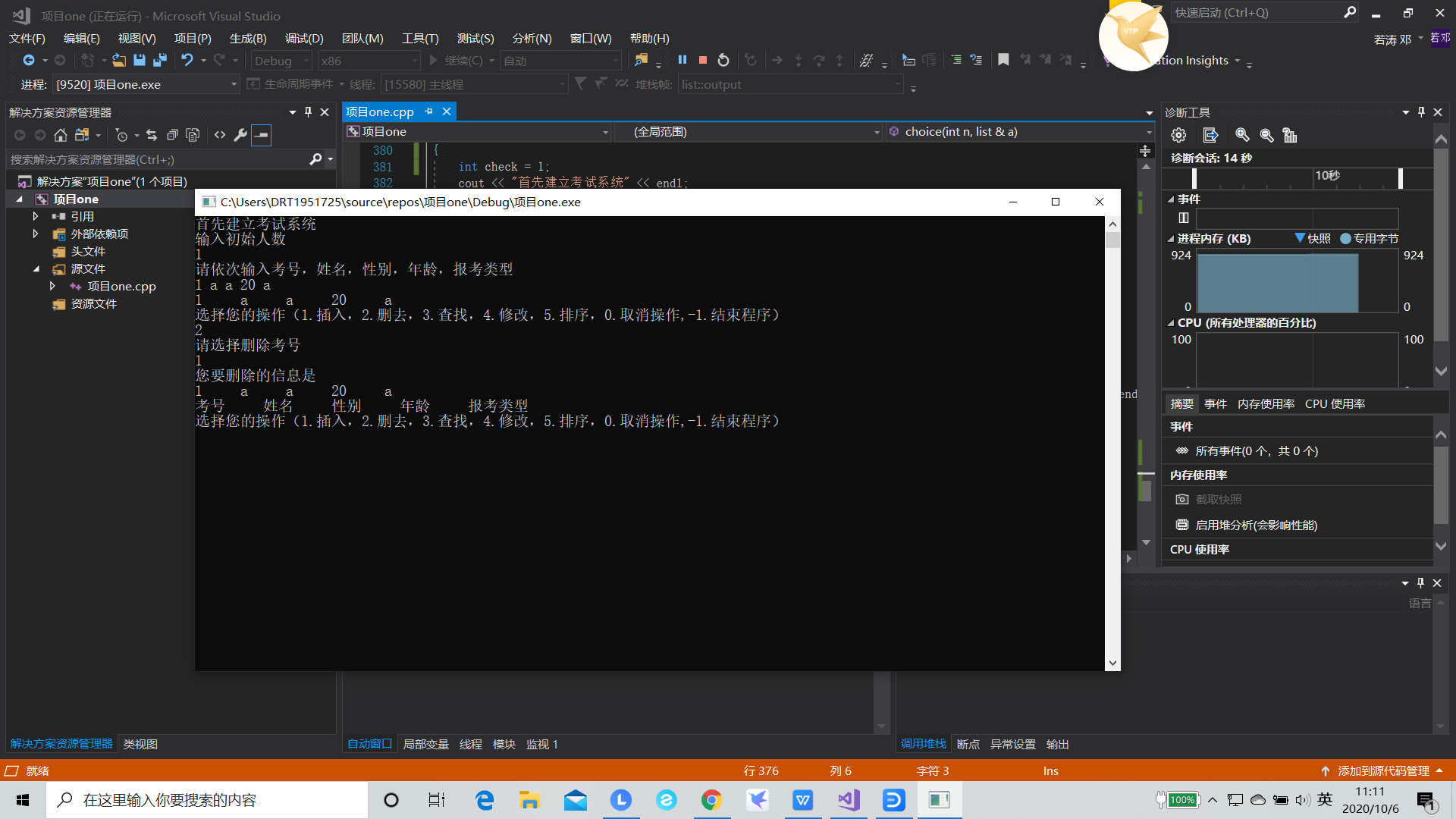


### 4.2.3 删除后链表为空

**测试用例：**删除前链表只有一个结点，删除后链表为空

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

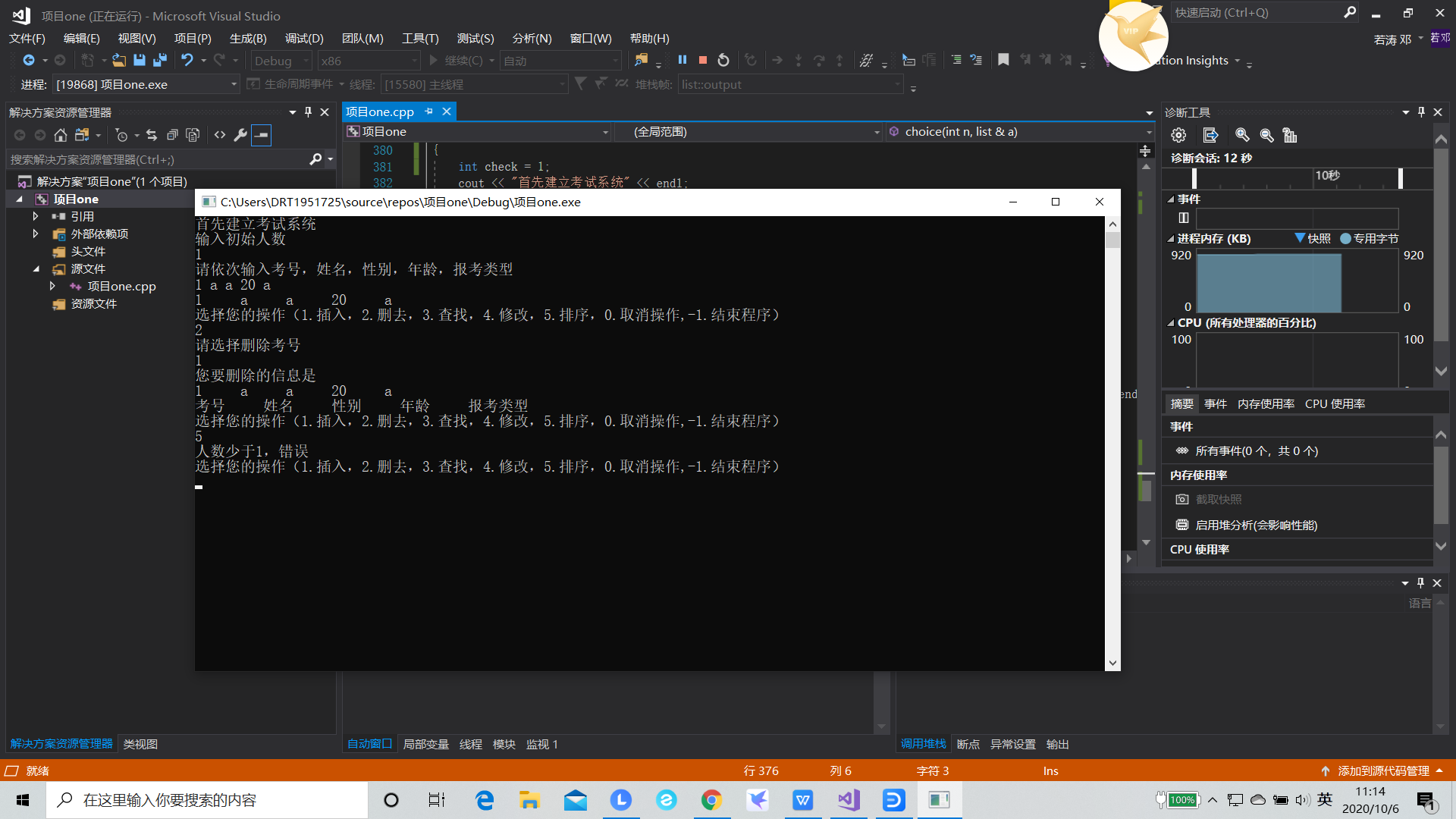


### 4.2.4 排序时链表为空

**测试用例：**删除前链表只有一个结点，删除后链表为空，再进行排序

**预期结果：**程序正常运行，不崩溃。

**实验结果：**



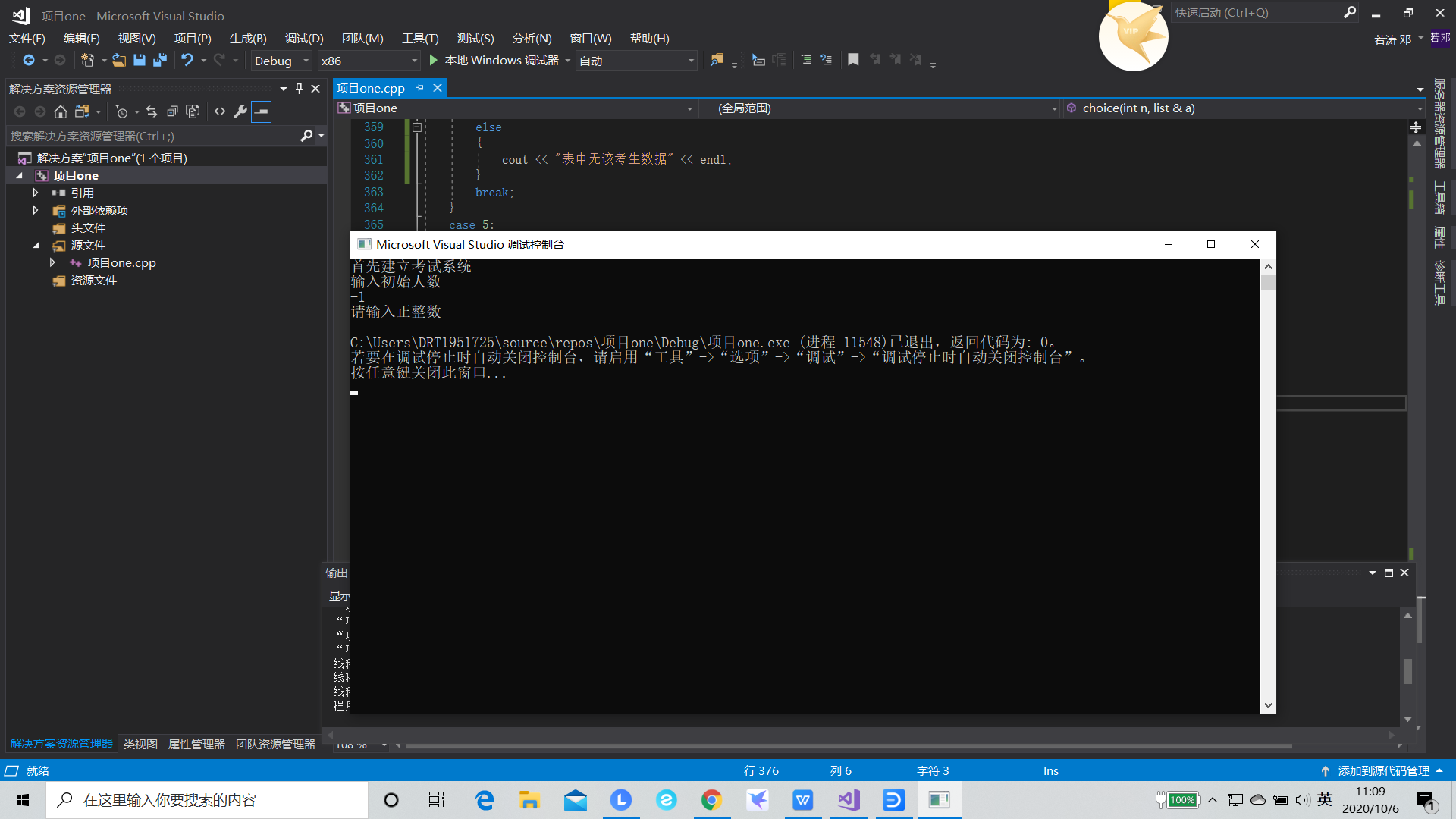
## 4.3 出错测试

### 4.3.1 考生人数错误

**测试用例：**输入考生人数为负数

**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：**

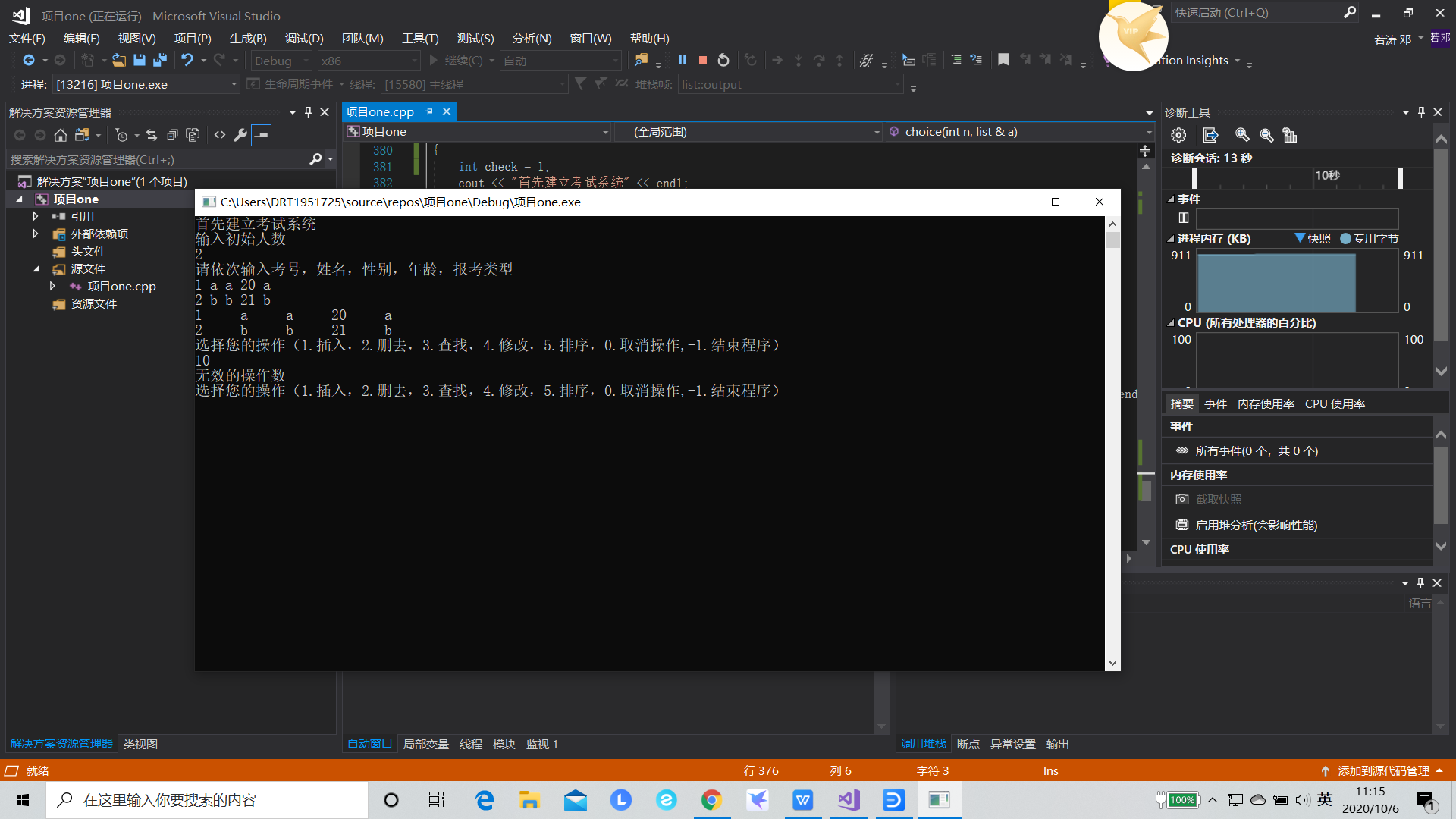


### 4.3.2 操作码错误

**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：**

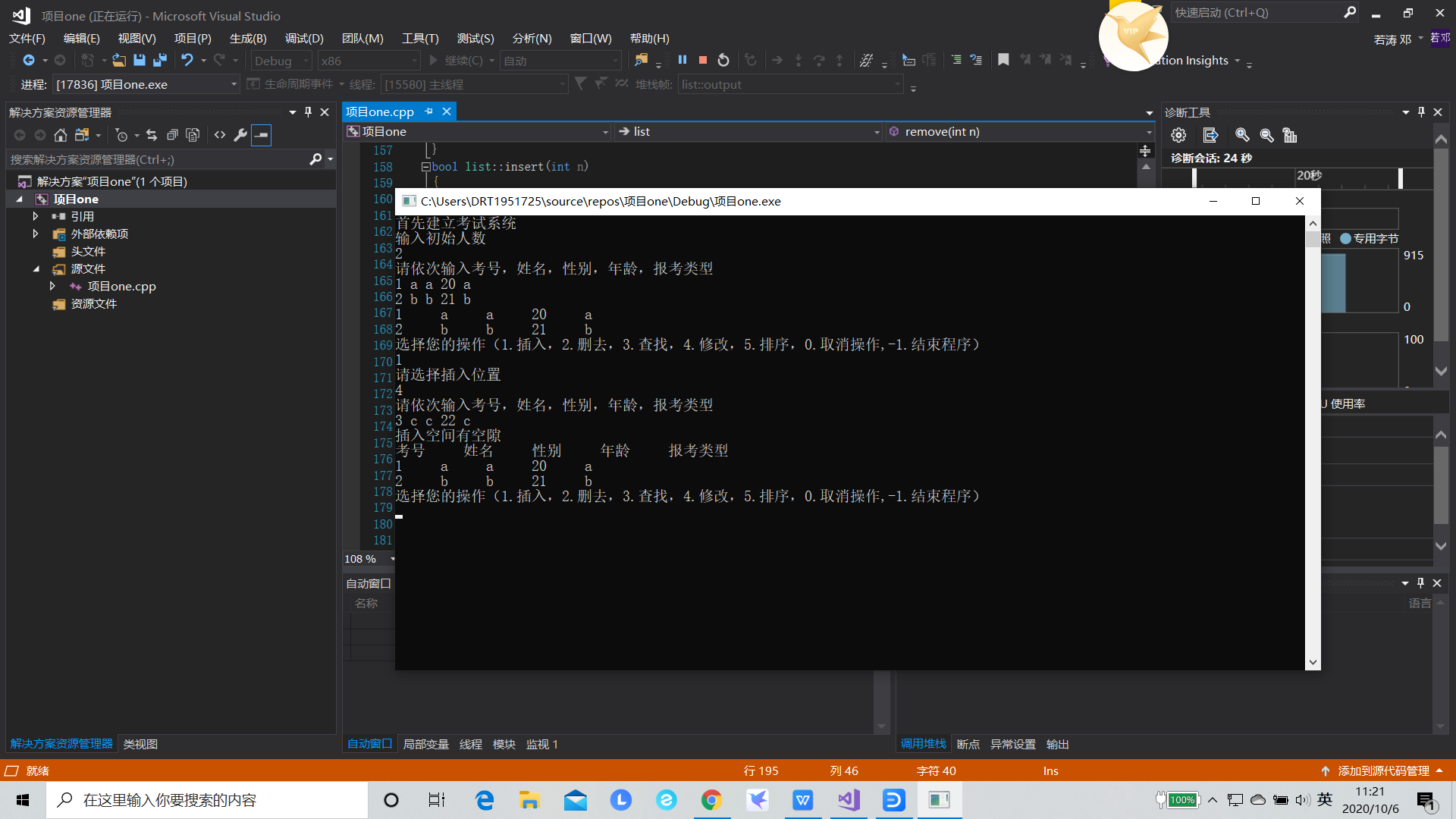


### 4.3.3 插入位置不存在

**测试用例：**链表里有两条记录，向链表的第四个位置插入结点

**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：**

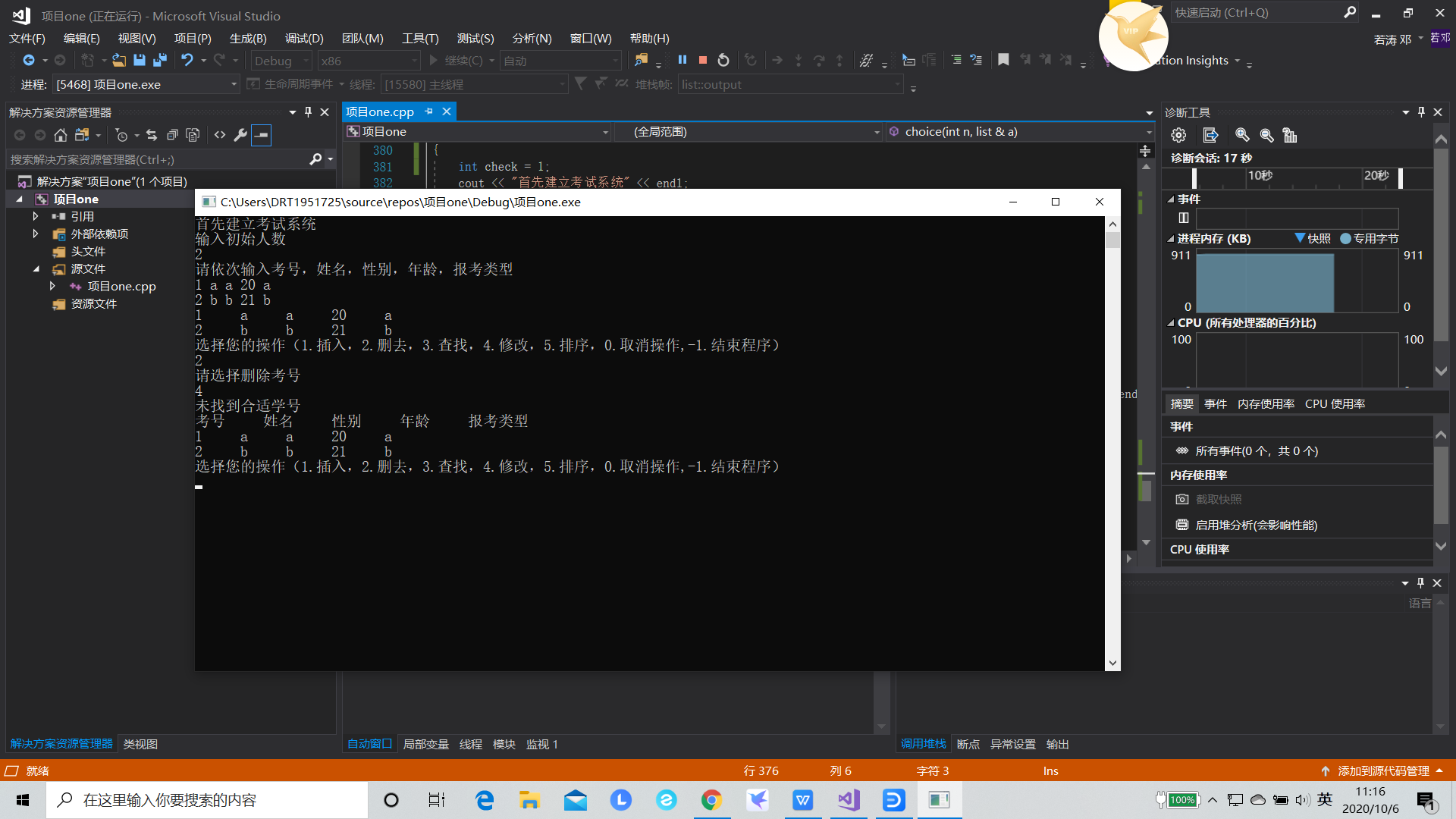


### 4.3.4 删除考号不存在

**测试用例：**要删除的考号不存在

**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：**

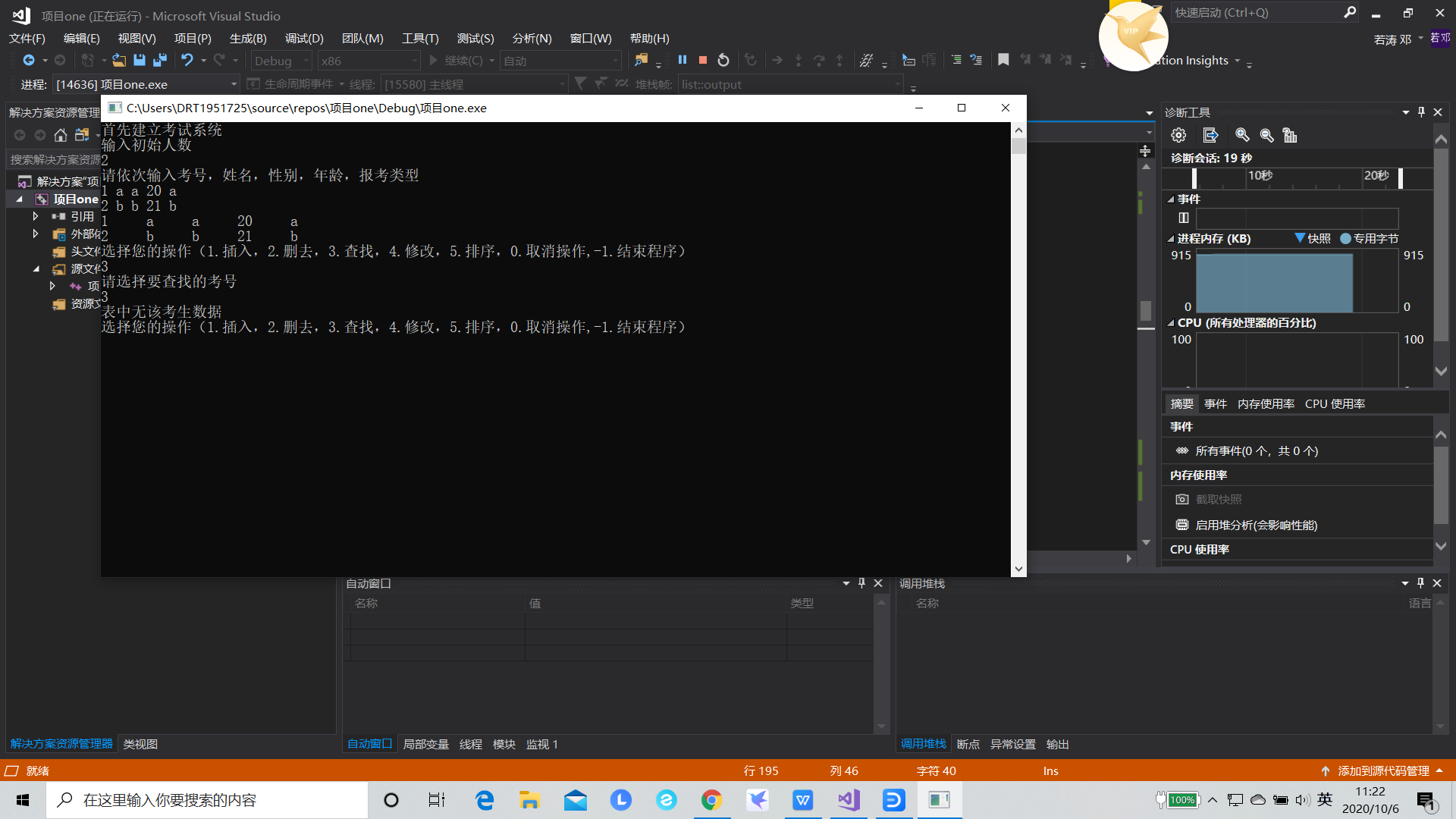


### 4.3.5 查找考号不存在

**测试用例：**要查找的考号不存在

**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：**

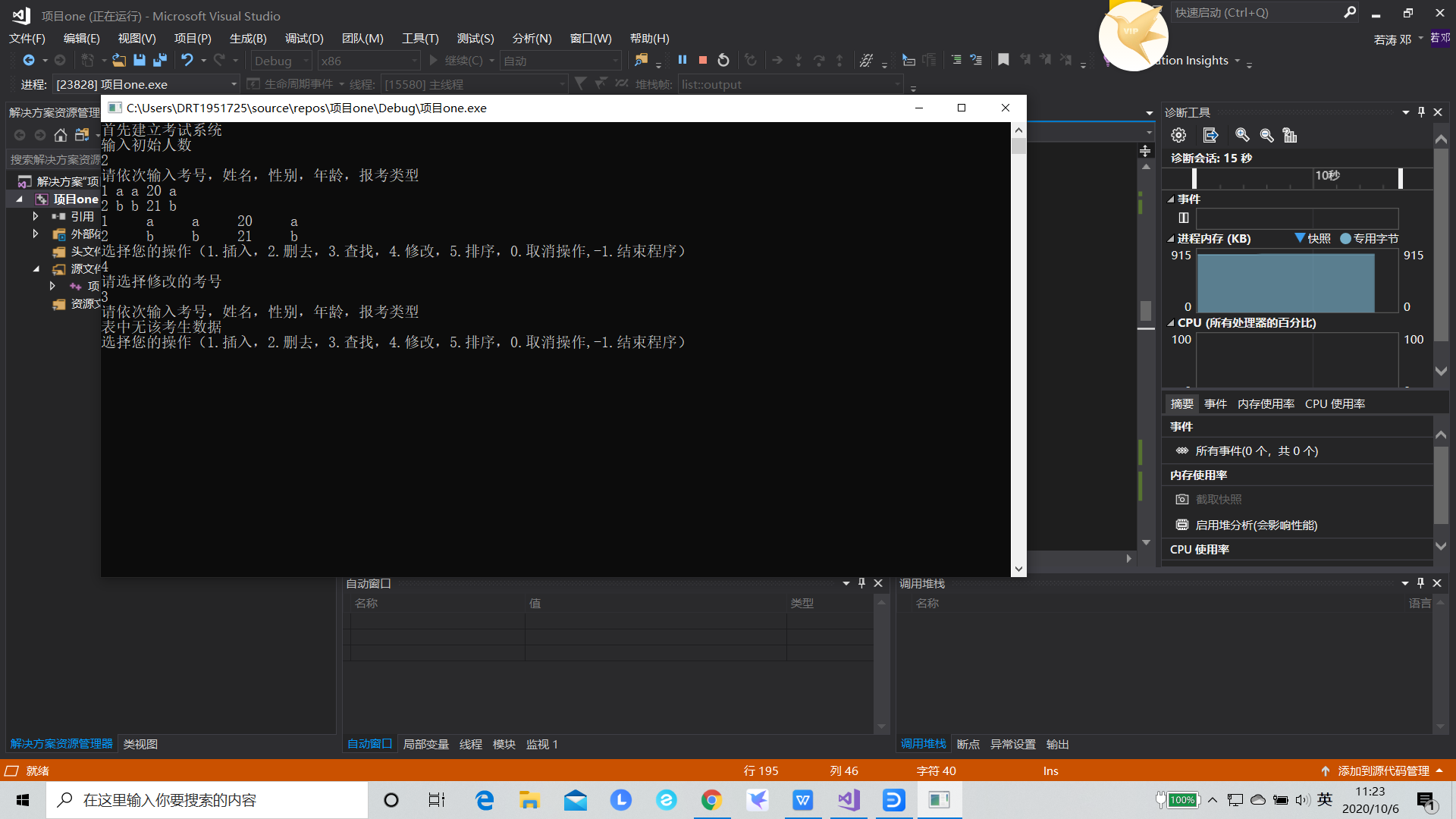


### 4.3.6 修改考号不存在

**测试用例：**要修改的考号不存在

**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：**



# 

### 4.3.7 排序中表有相同考号存在

**测试用例：**表中有两个相同的考号

**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：**

