项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 张诚睿

学 号： 2150998

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](#_Toc122643524)

[1.1 项目背景分析 1](#_Toc122643525)

[1.2 功能分析 1](#_Toc122643526)

[2 设计 1](#_Toc122643527)

[2.1 数据结构设计 1](#_Toc122643528)

[2.2 类结构设计 2](#_Toc122643529)

[2.3 成员与操作设计 2](#_Toc122643530)

[2.4 系统设计 4](#_Toc122643531)

[3 实现 4](#_Toc122643532)

[3.1 插入功能的实现 4](#_Toc122643533)

[3.1.1 插入功能流程图 4](#_Toc122643534)

[3.1.2 插入功能核心代码 4](#_Toc122643535)

[3.1.3 插入功能截屏示例 5](#_Toc122643536)

[3.2 删除功能的实现 6](#_Toc122643537)

[3.2.1 删除功能流程图 6](#_Toc122643538)

[3.2.2 删除功能核心代码 6](#_Toc122643539)

[3.2.3 删除功能截屏示例 8](#_Toc122643540)

[3.3 查找功能的实现 9](#_Toc122643541)

[3.3.1 查找功能流程图 9](#_Toc122643542)

[3.3.2 查找功能核心代码 9](#_Toc122643543)

[3.3.3 查找功能截图示例 10](#_Toc122643544)

[3.4 修改功能的实现 11](#_Toc122643545)

[3.4.1 修改功能流程图 11](#_Toc122643546)

[3.4.2 修改功能核心代码 11](#_Toc122643547)

[3.4.3 修改功能截屏示例 12](#_Toc122643548)

[3.5 统计功能的实现 13](#_Toc122643549)

[3.5.1 统计功能流程图 13](#_Toc122643550)

[3.5.2 统计功能核心代码 13](#_Toc122643551)

[3.5.3 统计功能截屏示例 14](#_Toc122643552)

[3.6 总体系统的实现 15](#_Toc122643553)

[3.6.1 总体系统流程图 15](#_Toc122643554)

[3.6.2 总体系统核心代码 15](#_Toc122643555)

[3.6.3 总体系统截屏示例 16](#_Toc122643556)

[4 测试 17](#_Toc122643557)

[4.1 功能测试 17](#_Toc122643558)

[4.1.1 插入功能测试 17](#_Toc122643559)

[4.1.2 删除功能测试 17](#_Toc122643560)

[4.1.3 查找功能测试 18](#_Toc122643561)

[4.1.4 修改功能测试 19](#_Toc122643562)

[4.1.5 统计功能测试 19](#_Toc122643563)

[4.2 边界测试 20](#_Toc122643564)

[4.2.1 初始化无输入数据 20](#_Toc122643565)

[4.2.2 删除头结点 20](#_Toc122643566)

[4.2.3 删除后链表为空 21](#_Toc122643567)

[4.2.4 考号、姓名、报考类型中英混杂 22](#_Toc122643568)

[4.3 出错测试 23](#_Toc122643569)

[4.3.1 考生人数错误 23](#_Toc122643570)

[4.3.2 操作码错误 23](#_Toc122643571)

[4.3.3 插入位置不存在 24](#_Toc122643572)

[4.3.4 删除考号不存在 24](#_Toc122643573)

[4.3.5 查找考号不存在 25](#_Toc122643574)

[4.3.6 修改考号不存在 25](#_Toc122643575)

# 1 分析

## 1.1 项目背景分析

考试作为客观衡量个体在某一方面能力的活动，如今已被学校、机构等大量运用，庞杂的参考人群对传统手记名单信息的方式提出了挑战，市场需要一款更加快速、准确、便捷的产品对考试人员信息进行管理。

在计算机技术广泛运用的今天，开发一个能够在电脑端简单管理考试报名信息的系统显得尤为重要，也更佳符合管理者日常工作习惯。由此考试报名系统孕育而生。

## 1.2 功能分析

本项的目的在于简单实现考试报名信息的管理功能，因此必须包含初始化建立全部考生信息、显示全部考生信息、添加新考生、删除原有考生的功能。

在此之外，为了方便寻找某一特定考生，项目还设计了根据学号查找考生的功能。同理，也可以修改任意考号考生的信息。

综上所述，项目可以实现对全体考生的信息输入输出、对某一特定考生的查找、修改、删除、添加功能，既做到了总览，也完成了个体操作，极大提高使用者的工作效率。

# 2 设计

## 2.1 数据结构设计

每个考生信息所包含的内容有学号、姓名、性别、年龄、报考类别五项，考生数量并不确定。因此，将该五个内容定义在同一个结构体LinkNode\_StuInfo中，并在结构体中添加指向该结构体的指针，作为链表的结点，再设计一个类ManageSystem，利用上述结点构建链表，将头结点记录在私有成员中，就可实现对整个链表的操作。

需要注意的是，姓名、报考类别是字符串，且长度不定，对有的考试来说，考号也不单纯是数字，因此需要引入字符串来记录。在此设计一个MyString类，用动态申请字符数组的方式实现变长度操作字符串，并重载部分运算符来简化使用操作。

## 2.2 类结构设计

如上所述，本项目采用struct描述链表结点类（LNode），这样使得链表结点类（LinkList）可以访问链表结点。

MyString利用动态申请数组方式管理字符串，用char\*类型的私有成员访问数组。

## 2.3 成员与操作设计

**链表结点类**

struct LinkNode\_StuInfo {

MyString num; // 考号

MyString name; // 姓名

bool sex = 0; // 0女1男

int age = 0; // 年龄

MyString category; // 报考类别

LinkNode\_StuInfo\* next = NULL; // 下一结点

};

利用编写的MyString类实现对字符串的管理，使得学生信息结点中考号支持字符（中英均可）+数字，姓名支持任意长度的中英文，性别用最节省空间的bool类型管理，年龄用整型记录，报考类别支持任意长度中英文。最后，有next指针用于连接节点

**链表类**

class ManageSystem {

private:

int StuNum; // 学生总数

LinkNode\_StuInfo\* head; // 记录首个结点

void init(); // 生成链表

public:

ManageSystem() :StuNum(0), head(NULL) {};

void GetInfo(); // 获取初始信息

int GetChoice(); // 获取操作选项

void Exit(); // 退出系统

void Insert(); // 插入信息

void Delete(); // 删除信息

void Seek(); // 查找信息

void Change(); // 更改信息

void Statistic(); // 统计信息

void show(); // 展示信息

};

用head指针记录头结点，从而对整个链表进行操作

**MyString字符串管理类**

class MyString {

private:

MyString& operator+=(char ch); // 末尾添加一个字符

int size; // 字符串长度

char\* content; // 内容指针

public:

MyString(const char\* str);

MyString(); // 无参

~MyString();

friend ostream& operator<<(ostream& os, MyString& target);

// 输出内容

friend istream& operator>>(istream& is, MyString& target);

// 读取内容，遇空格、回车停止

bool operator==(const MyString& matched);

// 判断两字符串是否一致

}；

MyString实现对字符串的管理，重载了流运算符以方便输入输出。读入内容时，每次size++，重新申请一块新空间并添加一个字符进content数组，这样操作虽然效率降低，但相比于每次多开辟固定长度的方式节省了空间，达到100%空间利用率，同时考虑到一般情况下考号、姓名等信息不会过长，对效率的影响几乎可以不计，所以采用了最大化利用空间的设计方式。

## 2.4 系统设计

首先建立ManageSystem对象，调用其GetInfo()函数获取初始信息，再调用其GetChoice()函数获取操作码，根据对应操作码执行插入、查找等一系列功能。若用户输入操作码0，退出系统。

# 3 实现

## 3.1 插入功能的实现

### 3.1.1 插入功能流程图

### 3.1.2 插入功能核心代码

// 获取新成员数据

LinkNode\_StuInfo\* p\_Work = head; // 工作指针

for (int i = 0; i < address - 2; i++)

{

p\_Work = p\_Work->next;

}

if (p\_Work == head)

{ // 在开头插入

head = NewMember;

NewMember->next = p\_Work;

}

else

{

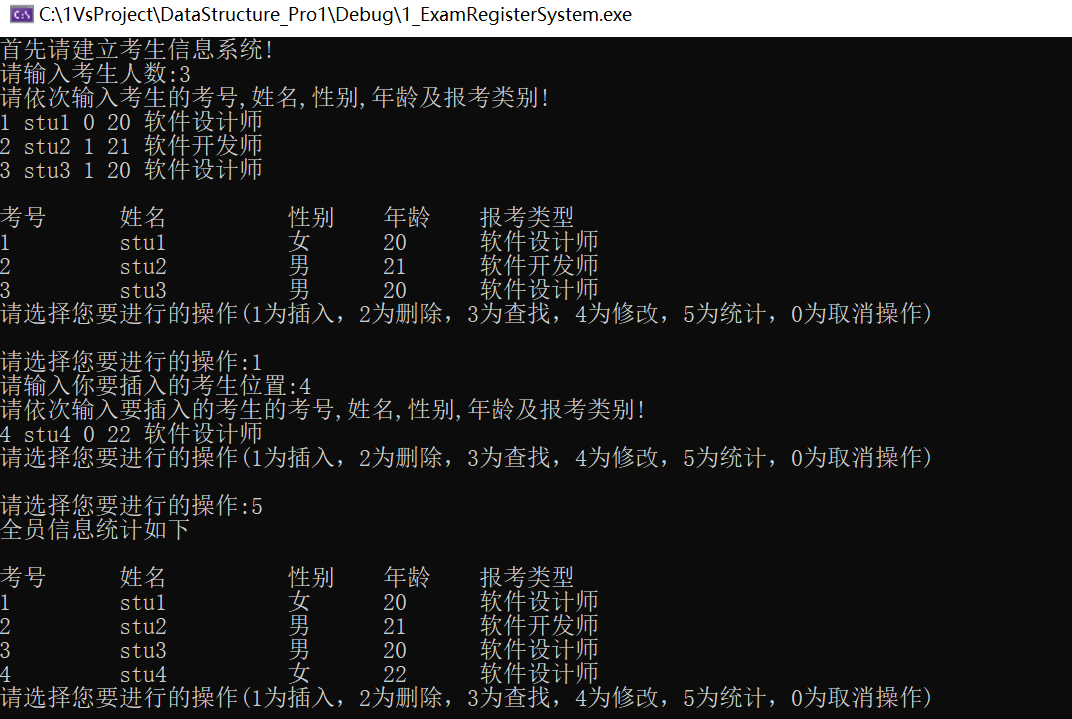
NewMember->next = p\_Work->next;

p\_Work->next = NewMember;

}

StuNum++;

### 3.1.3 插入功能截屏示例



此次插入在原有三人信息之后第四个位置插入新考生，并用5-统计指令展示结果

## 3.2 删除功能的实现

### 3.2.1 删除功能流程图

### 3.2.2 删除功能核心代码

void ManageSystem::Delete()

{

cout << "请输入要删除的考生的考号:";

MyString Delete\_Query;

cin >> Delete\_Query;

// 删除

LinkNode\_StuInfo\* p\_Work = head; // 工作指针

if (p\_Work->num == Delete\_Query)

{ // 首项即为目标

head = p\_Work->next;

delete p\_Work;

cout << "已成功删除该考生信息" << endl;

StuNum--;

if (StuNum == 0)

{

cout << "已无考生待处理，自动退出" << endl;

exit(EXIT\_SUCCESS);

}

return;

}

else

{

while (p\_Work->next != NULL)

{

if (p\_Work->next->num == Delete\_Query)

{

break;

} // 查找成功，位置为p->next

p\_Work = p\_Work->next;

}

}

if (p\_Work->next == NULL)

{ // 查找失败

cout << "查无此人!" << endl;

}

else

{ // 查找成功且不为首项

LinkNode\_StuInfo\* p\_Record = p\_Work->next;

p\_Work->next = p\_Record->next;

delete p\_Record;

cout << "已成功删除该考生信息" << endl;

StuNum--;

}

if (StuNum == 0)

{

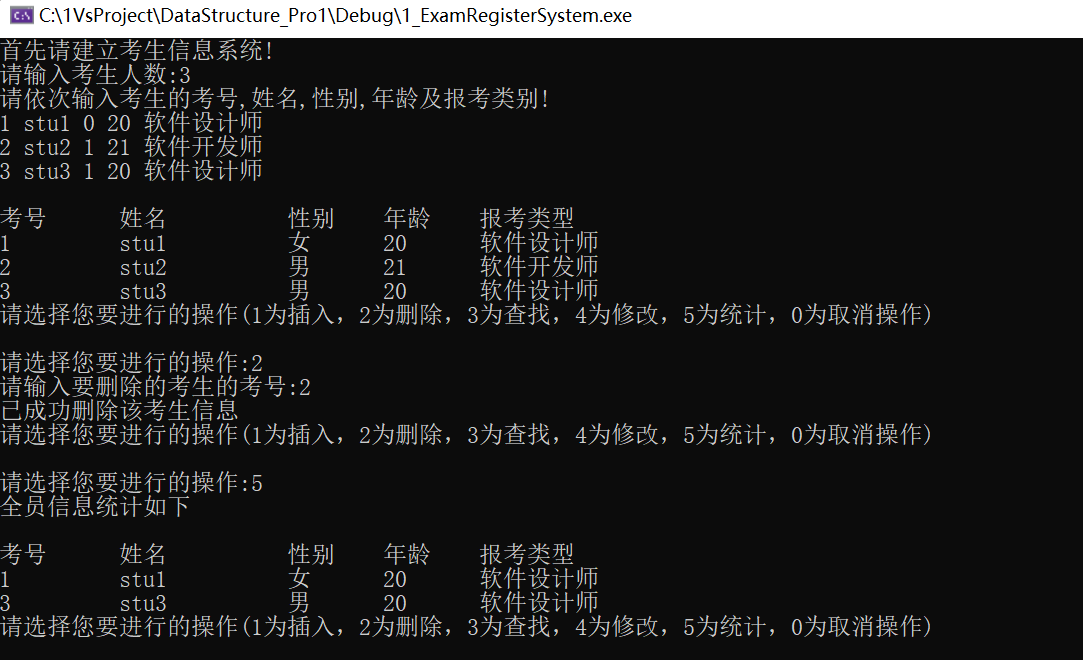
cout << "已无考生待处理，自动退出" << endl;

exit(EXIT\_SUCCESS);

}

}

### 3.2.3 删除功能截屏示例



此次在原有三人信息中删除考号为2的考生信息，并用指令5-统计展示结果

## 3.3 查找功能的实现

### 3.3.1 查找功能流程图

### 3.3.2 查找功能核心代码

void ManageSystem::Seek()

{

cout << "请输入要查找的考生的考号:";

MyString query;

cin >> query;

// 查找

LinkNode\_StuInfo\* p\_Work = head;

while (p\_Work != NULL)

{

if (p\_Work->num == query)

{

break;

} // 查找成功

p\_Work = p\_Work->next;

}

if (p\_Work == NULL)

{ // 查找失败

cout << "查无此人!" << endl;

}

else

{ // 查找成功

cout << endl;

cout << setiosflags(ios::left); // 右对齐模式

cout << "考号 姓名 性别 年龄 报考类型" << endl;

cout << setfill(' ') << setw(10) << p\_Work->num <<

setw(14) << p\_Work->name <<

setw(8) << (p\_Work->sex ? "男" : "女") <<

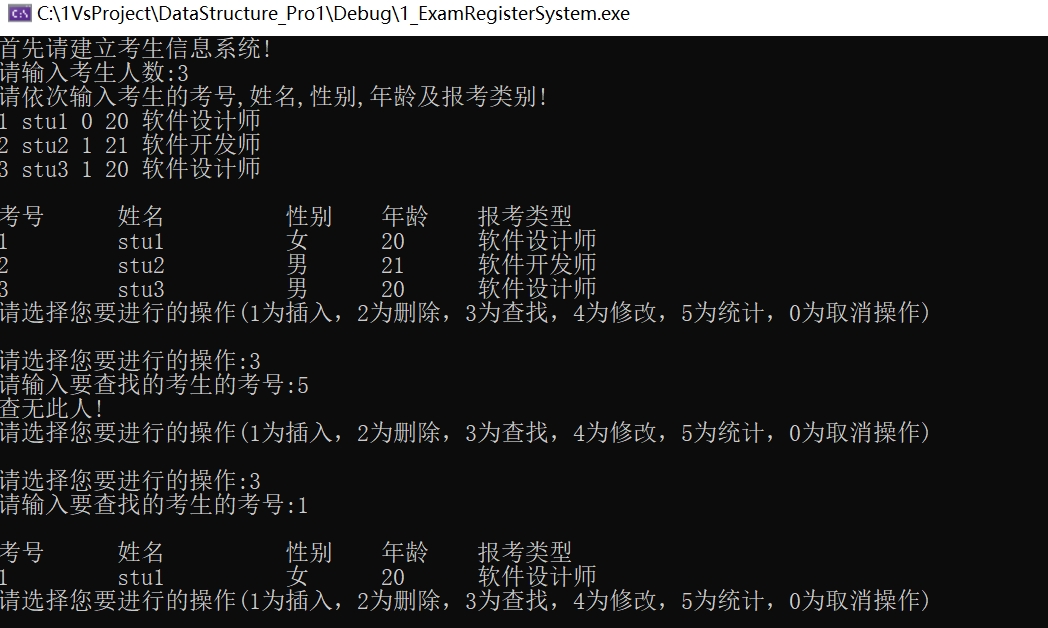
setw(8) << p\_Work->age <<

p\_Work->category << endl;

}

}

### 3.3.3 查找功能截图示例



此次在三个考生信息中首先查找了不存在的考号，显示“查无此人！”；再查找了考号为1的考生，打印出其信息

## 3.4 修改功能的实现

### 3.4.1 修改功能流程图

### 3.4.2 修改功能核心代码

void ManageSystem::Change()

{

cout << "请输入要修改的考生的考号:";

MyString query;

cin >> query;

// 查找

LinkNode\_StuInfo\* p\_Work = head;

while (p\_Work != NULL)

{

if (p\_Work->num == query)

{

break;

} // 查找成功

p\_Work = p\_Work->next;

}

if (p\_Work == NULL)

{ // 查找失败

cout << "查无此人!" << endl;

}

else

{ // 查找成功

cout << "请依次输入要修改的考生的考号,姓名,性别,年龄及报考类别!" << endl;

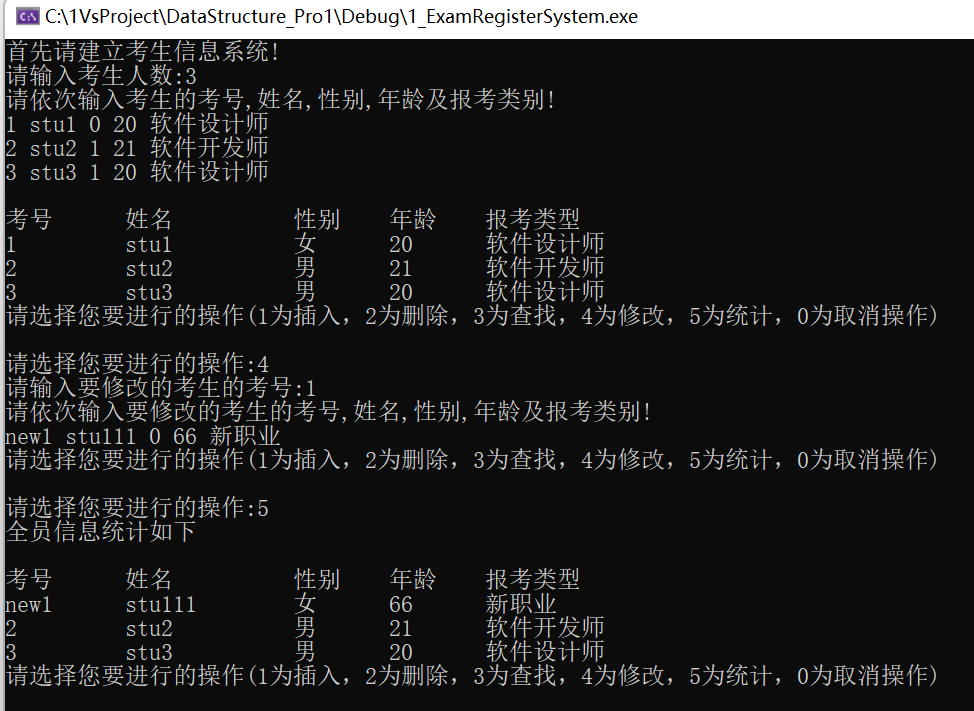
cin >> p\_Work->num >> p\_Work->name >> p\_Work->sex >> p\_Work->age >> p\_Work->category;

// 获取修改后数据

}

}

### 3.4.3 修改功能截屏示例



此次在三个考生信息中修改考号为1的考生信息，并用指令5-统计来展示修改后结果

## 3.5 统计功能的实现

### 3.5.1 统计功能流程图

### 3.5.2 统计功能核心代码

void ManageSystem::show()

{

cout << endl;

cout << setiosflags(ios::left); // 右对齐模式

cout << "考号 姓名 性别 年龄 报考类型" << endl;

LinkNode\_StuInfo\* p = head;

while (p != NULL)

{

cout <<setfill(' ')<<setw(10)<<p->num <<

setw(14) << p->name <<

setw(8) << (p->sex ? "男" : "女") <<

setw(8) << p->age <<

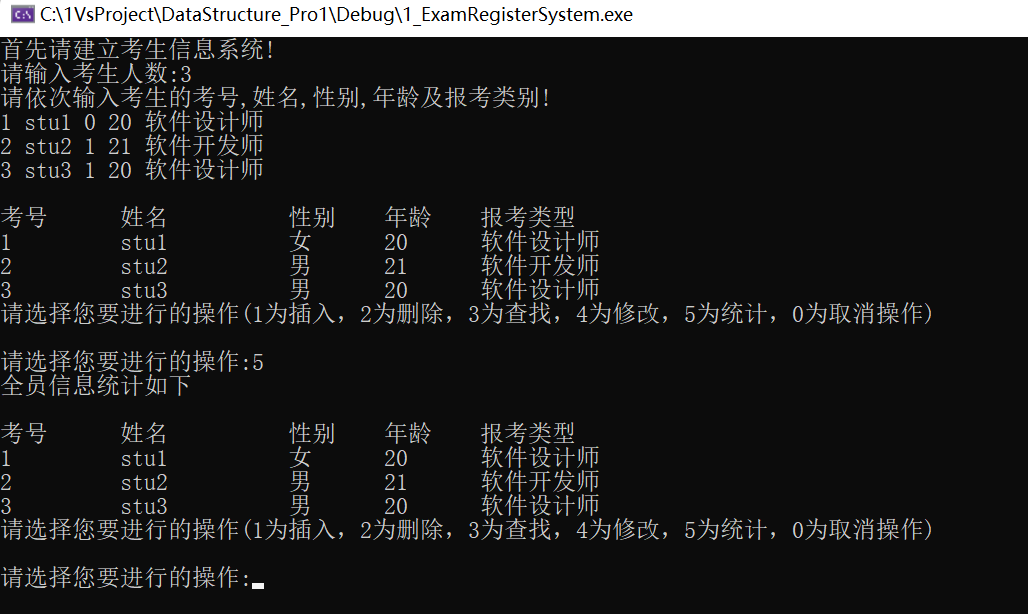
p->category << endl;

p = p->next;

}

}

### 3.5.3 统计功能截屏示例



## 3.6 总体系统的实现

### 3.6.1 总体系统流程图

### 3.6.2 总体系统核心代码

while (choice = ExamRegister.GetChoice())

{

switch (choice){

case 1:

ExamRegister.Insert();

break;

case 2:

ExamRegister.Delete();

break;

case 3:

ExamRegister.Seek();

break;

case 4:

ExamRegister.Change();

break;

case 5:

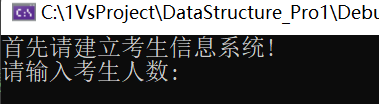
ExamRegister.Statistic();

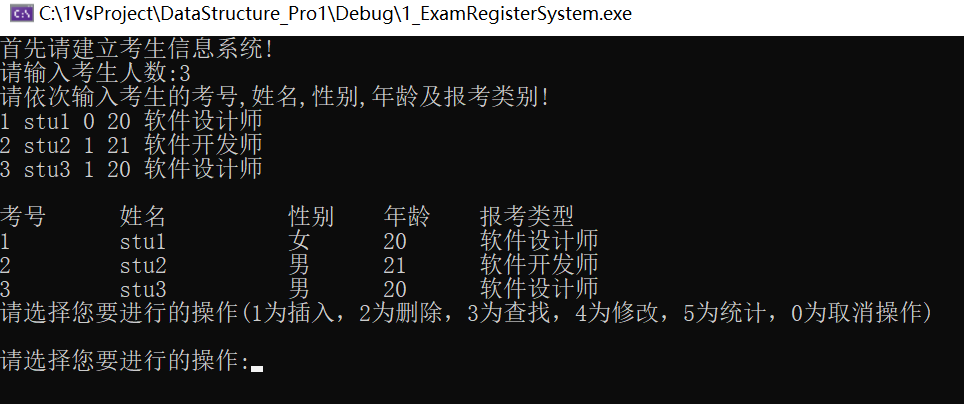
break;

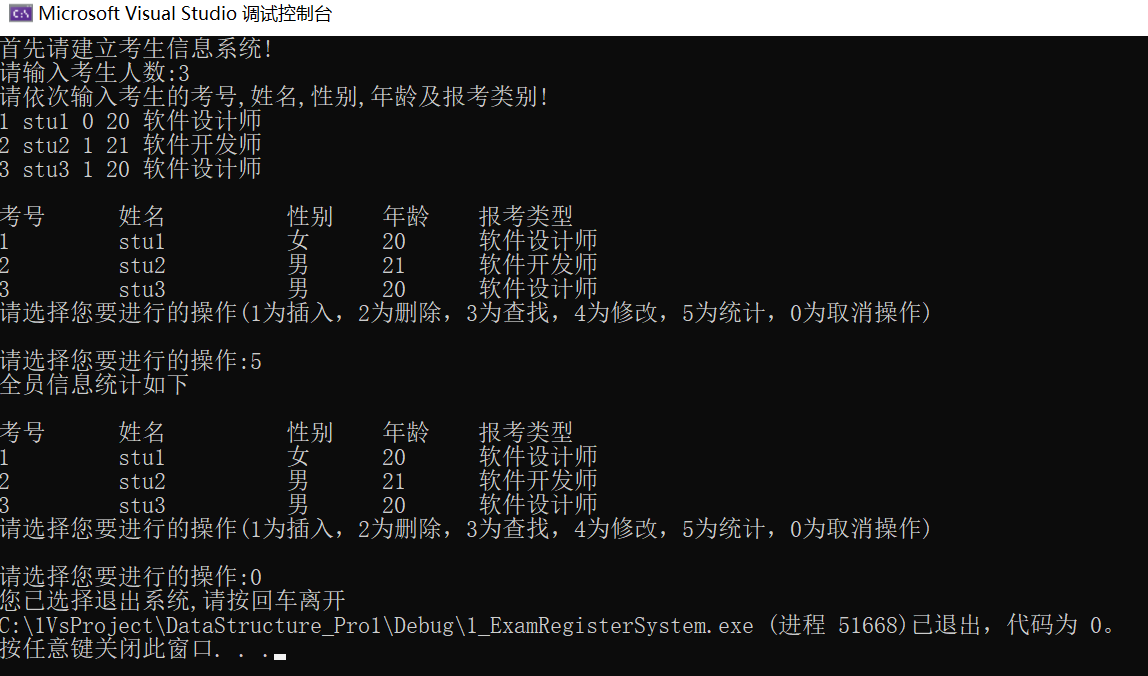
}

}

### 3.6.3 总体系统截屏示例







# 4 测试

## 4.1 功能测试

### 4.1.1 插入功能测试

**测试用例**：3 stu3 1 20 软件设计师

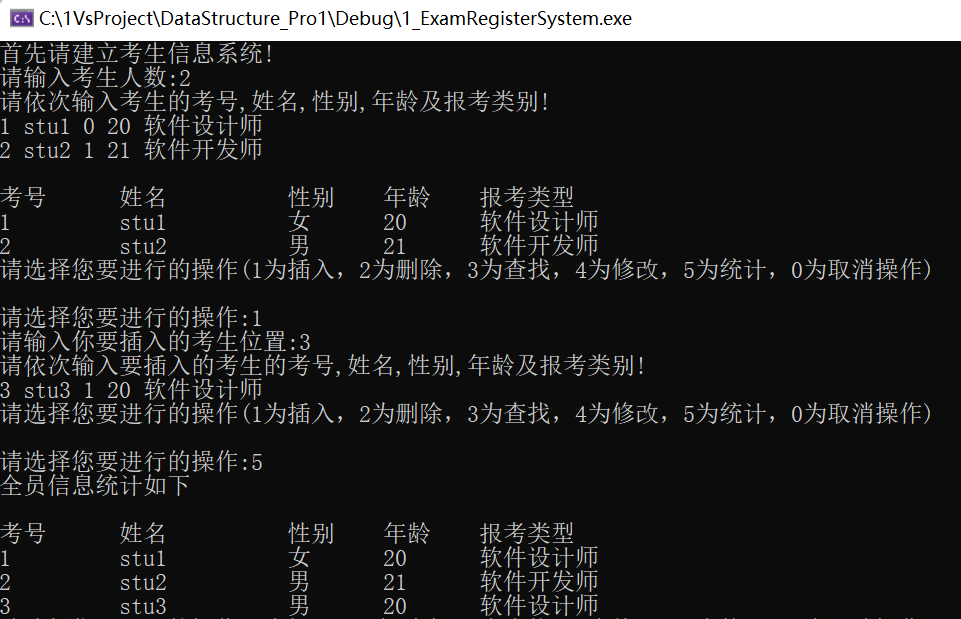
**预期结果**：

1 stu1 女 20 软件设计师

2 stu2 男 21 软件开发师

3 stu3 男 20 软件设计师

**实验结果**



### 4.1.2 删除功能测试

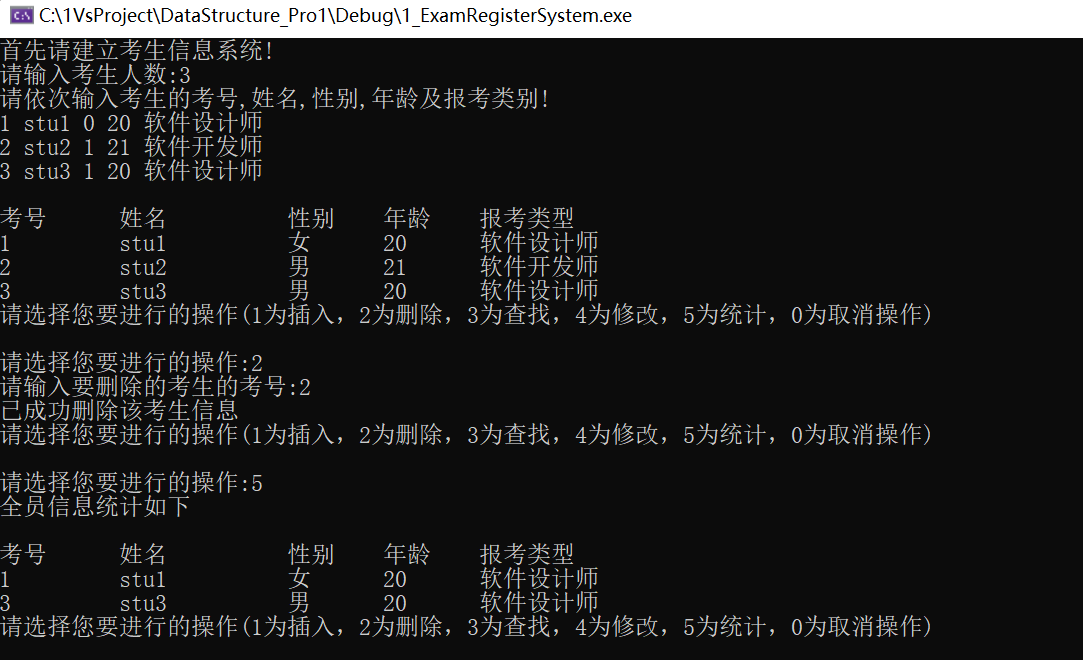
**测试用例：**删除考号为2的考生

**预期结果：**

1 stu1 女 20 软件设计师

3 stu3 男 20 软件设计师

**实验结果：**



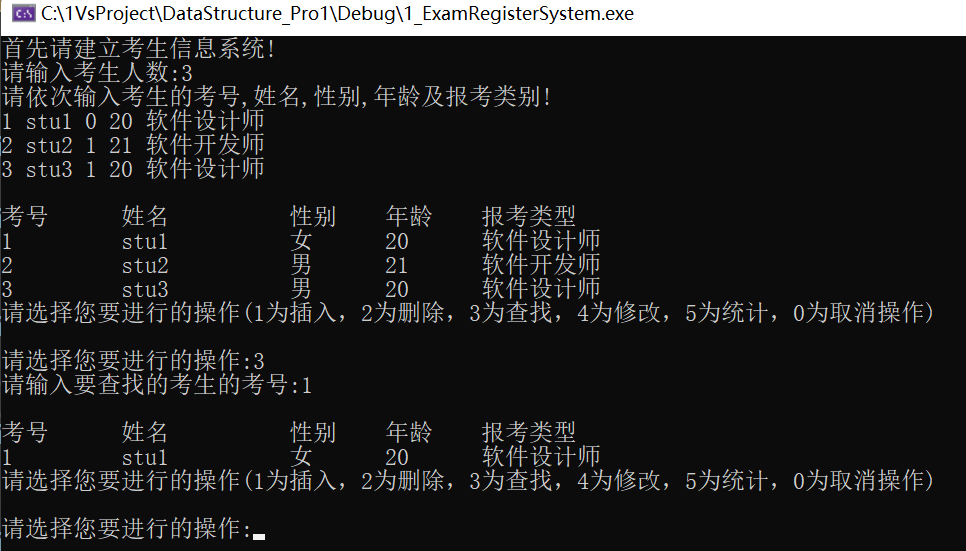
### 4.1.3 查找功能测试

**测试用例：**查找考号为1的考生

**预期结果：**

1 stu1 女 20 软件设计师

**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66，报考种类新职业，考号new111，姓名stu111。

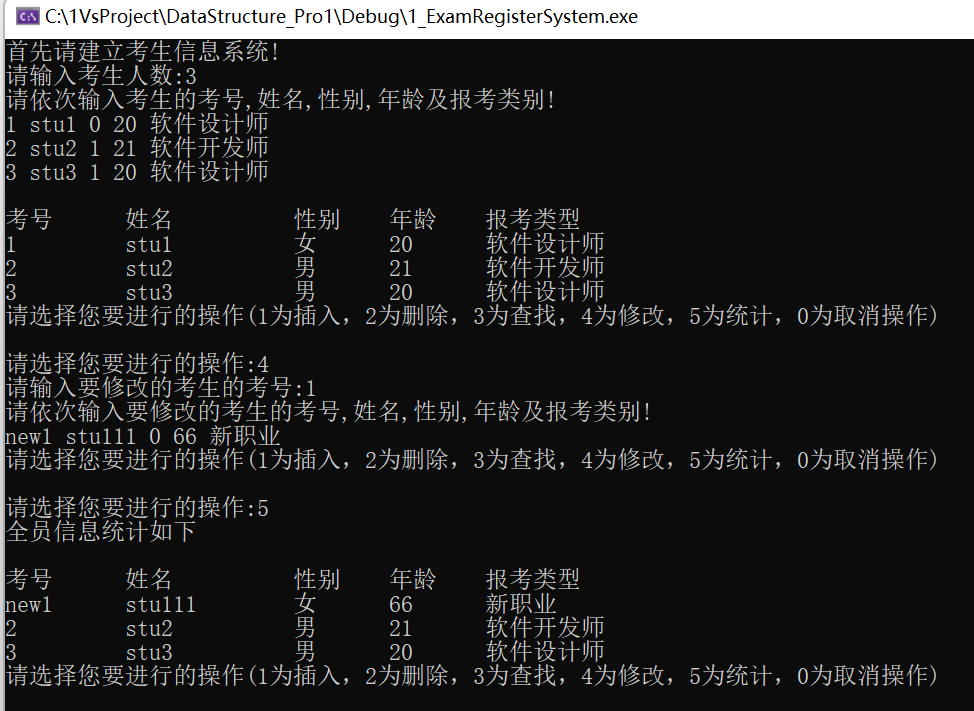
**预期结果：**

new111 stu111 女 66 新职业

2 stu2 男 21 软件开发师

3 stu3 男 20 软件设计师

**实验结果：**



### 4.1.5 统计功能测试

**测试用例：**统计当前数据

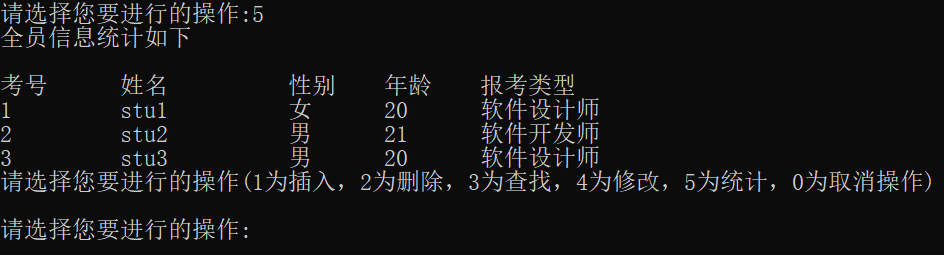
**预期结果：**

1 stu1 女 20 软件设计师

2 stu2 男 21 软件开发师

3 stu3 男 20 软件设计师

**实验结果：**



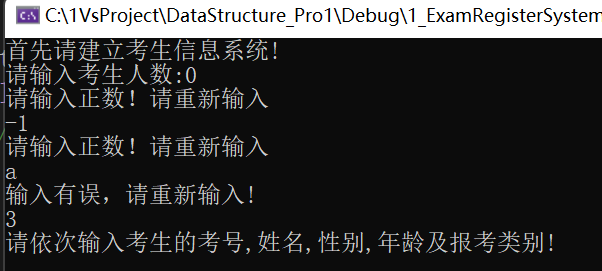
## 4.2 边界测试

### 4.2.1 初始化无输入数据

**测试用例：**初始无输入数据，或考生人数为字符、负数

**预期结果：**给出错误信息，重新输入直到输入正整数

**实验结果：**

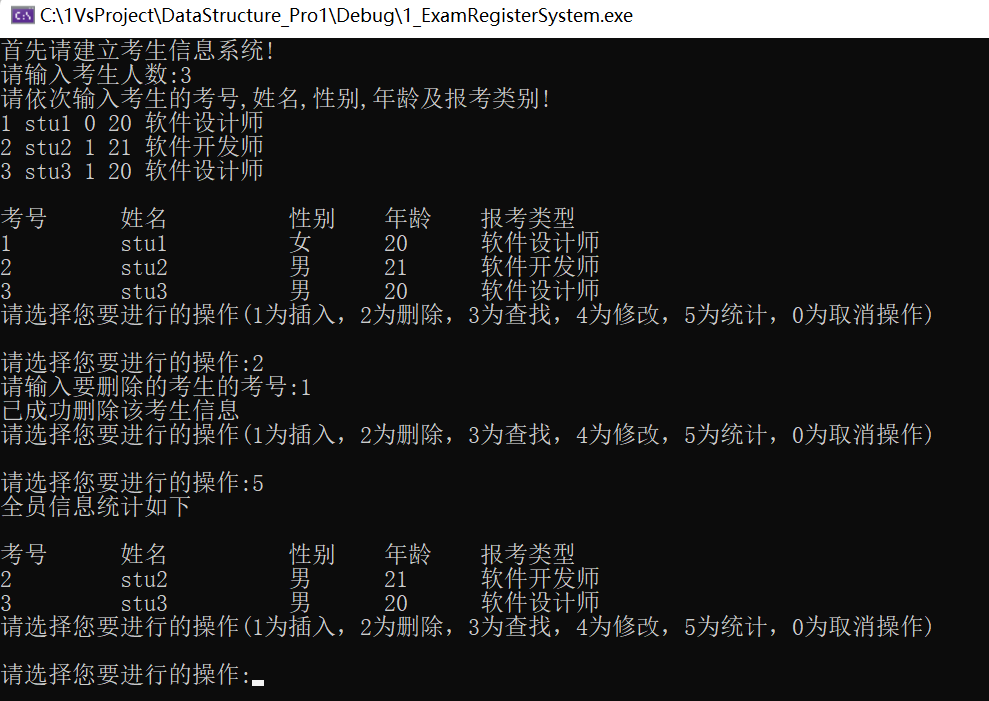


### 4.2.2 删除头结点

**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：**

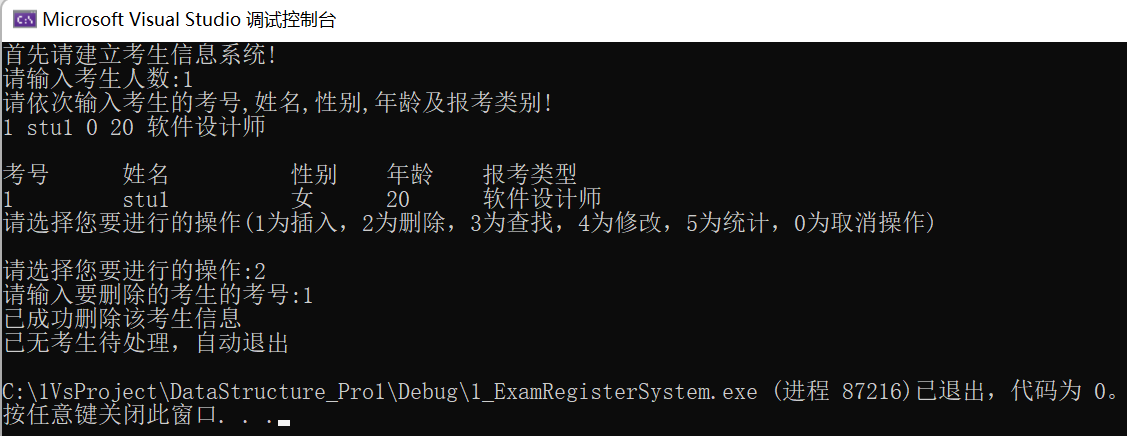


### 4.2.3 删除后链表为空

**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，提示已无考生信息需要处理并正常退出程序。

**实验结果：**



### 4.2.4 考号、姓名、报考类型中英混杂

测试用例：

test 张三 0 19 高级Engineer

509X Sam王 1 20 特级工程师

7/na Bob 1 90 Special职位

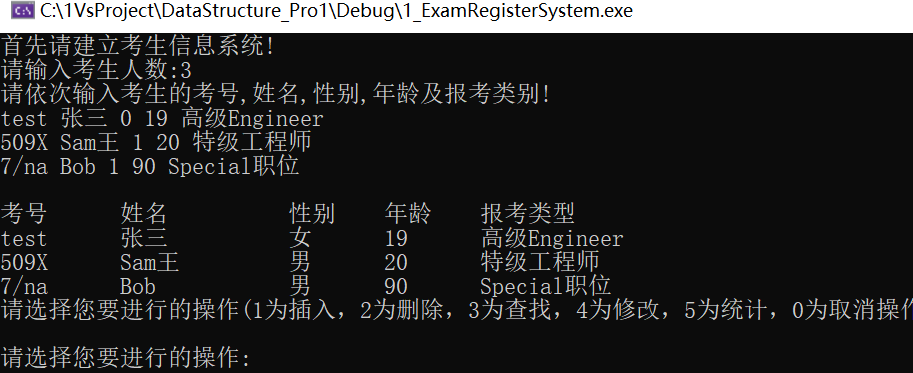
预期结果：

test 张三 0 19 高级Engineer

509X Sam王 1 20 特级工程师

7/na Bob 1 90 Special职位

实验结果：



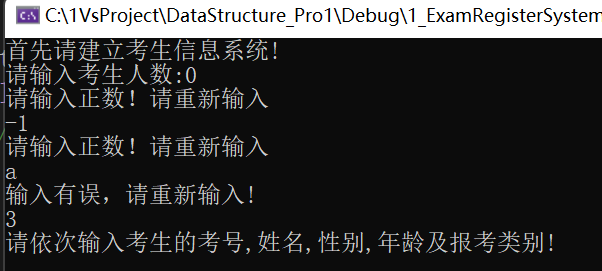
## 4.3 出错测试

### 4.3.1 考生人数错误

**测试用例：**输入考生人数为负数或字符

**预期结果：**程序给出错误提示，重新输入直至输入正确

**实验结果：**

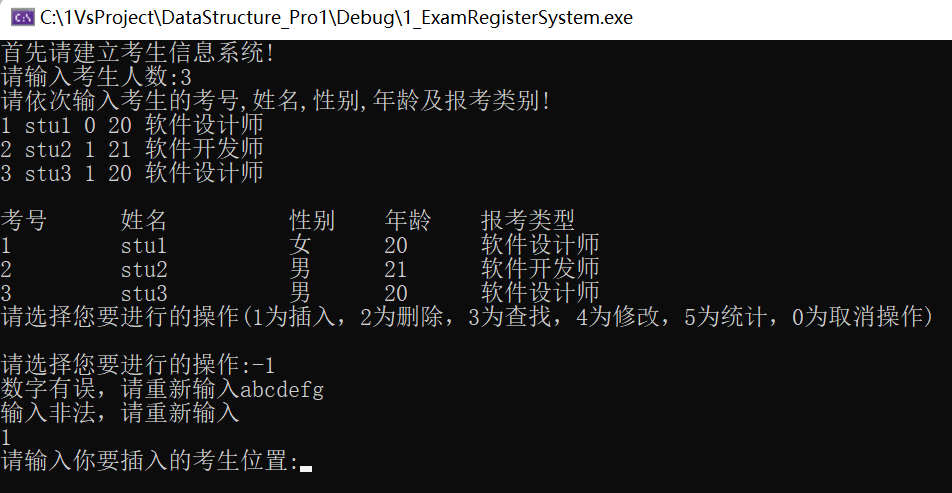


### 4.3.2 操作码错误

**测试用例：**输入操作码不为0-5的数字或是字符

**预期结果：**程序给出提示信息，重新输入直至输入正确。

**实验结果：**

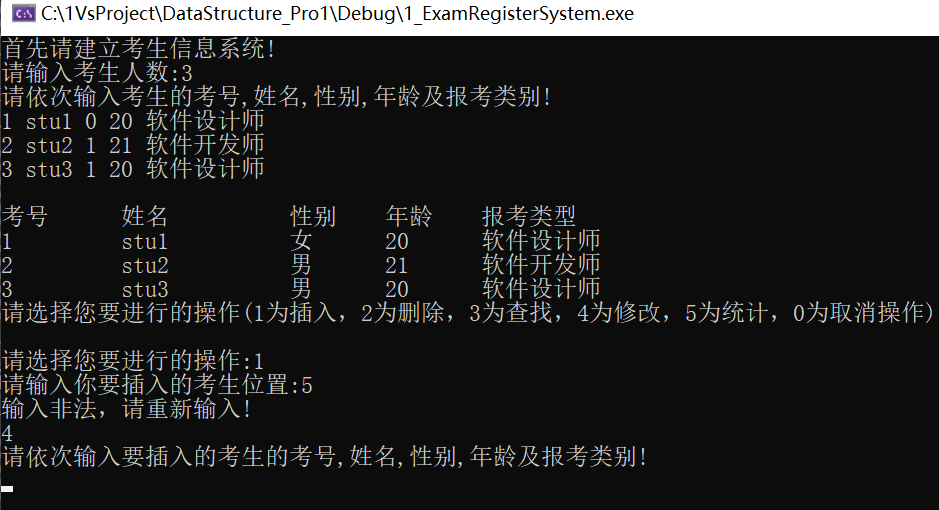


### 4.3.3 插入位置不存在

**测试用例：**链表里有三条记录，向链表的第五个位置插入结点

**预期结果：**程序给出错误信息，重新输入插入位置直至正确

**实验结果：**

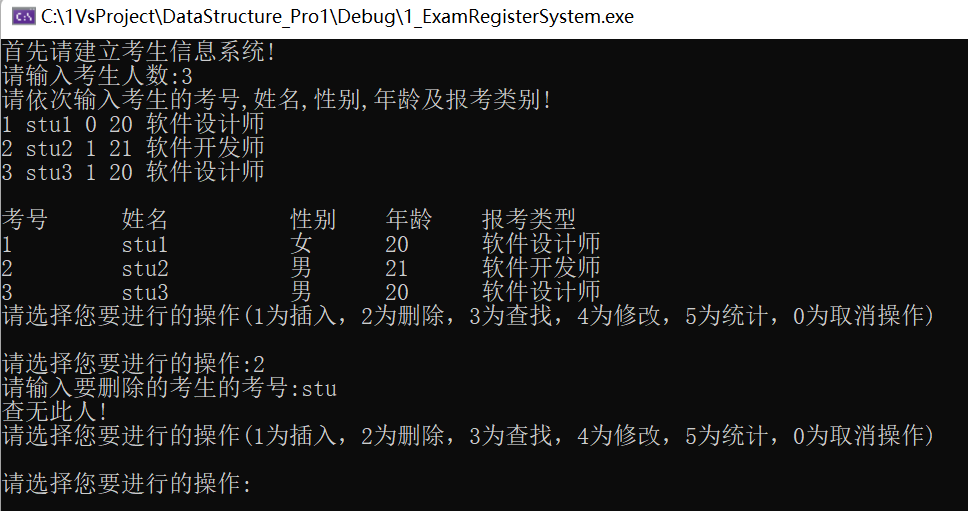


### 4.3.4 删除考号不存在

**测试用例：**要删除的考号不存在

**预期结果：**程序给出错误提示（由于MyString类重载==的加入，可以对任意内容的考号进行匹配）

**实验结果：**

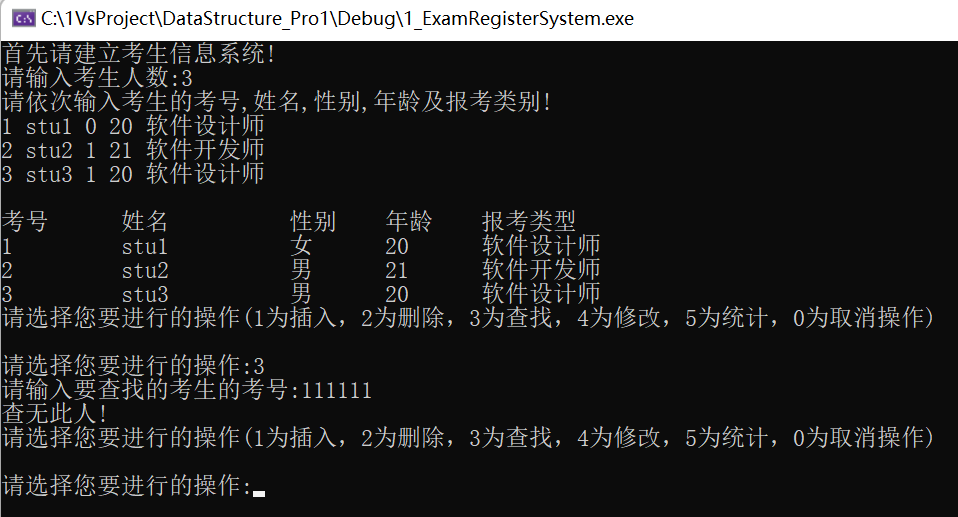


### 4.3.5 查找考号不存在

**测试用例：**要查找的考号不存在

**预期结果：**程序给出错误提示（由于MyString类重载==的加入，可以对任意内容的考号进行匹配）

**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：**程序给出错误提示（由于MyString类重载==的加入，可以对任意内容的考号进行匹配）

**实验结果：**

