项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 任冬晨

学 号： 1852470

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](#_Toc495668153)

[1.1 背景分析 1](#_Toc495668154)

[1.2 功能分析 1](#_Toc495668155)

[2 设计 2](#_Toc495668156)

[2.1 数据结构设计 2](#_Toc495668157)

[2.2 类结构设计 2](#_Toc495668158)

[2.3 成员与操作设计 2](#_Toc495668159)

[2.4 系统设计 4](#_Toc495668160)

[3 实现 4](#_Toc495668161)

[3.1 插入功能的实现 4](#_Toc495668162)

[3.1.1 插入功能流程图 4](#_Toc495668163)

[3.1.2 插入功能核心代码 5](#_Toc495668164)

[3.1.3 插入功能截屏示例 5](#_Toc495668165)

[3.2 删除功能的实现 7](#_Toc495668166)

[3.2.1 删除功能流程图 7](#_Toc495668167)

[3.2.2 删除功能核心代码 7](#_Toc495668168)

[3.2.3 删除功能截屏示例 8](#_Toc495668169)

[3.3 查找功能的实现 10](#_Toc495668170)

[3.3.1 查找功能流程图 10](#_Toc495668171)

[3.3.2 查找功能核心代码 10](#_Toc495668172)

[3.3.3 查找功能截图示例 11](#_Toc495668173)

[3.4 修改功能的实现 13](#_Toc495668174)

[3.4.1 修改功能流程图 13](#_Toc495668175)

[3.4.2 修改功能核心代码 13](#_Toc495668176)

[3.4.3 修改功能截屏示例 14](#_Toc495668177)

[3.5 统计功能的实现 15](#_Toc495668178)

[3.5.1 统计功能流程图 15](#_Toc495668179)

[3.5.2 统计功能核心代码 15](#_Toc495668180)

[3.5.3 统计功能截屏示例 16](#_Toc495668181)

[3.6 总体系统的实现 16](#_Toc495668182)

[3.6.1 总体系统流程图 16](#_Toc495668183)

[3.6.2 总体系统核心代码 17](#_Toc495668184)

[3.6.3 总体系统截屏示例 17](#_Toc495668185)

[4 测试 19](#_Toc495668186)

[4.1 功能测试 19](#_Toc495668187)

[4.1.1 插入功能测试 19](#_Toc495668188)

[4.1.2 删除功能测试 19](#_Toc495668189)

[4.1.3 查找功能测试 20](#_Toc495668190)

[4.1.4 修改功能测试 20](#_Toc495668191)

[4.1.5 统计功能测试 21](#_Toc495668192)

[4.2 边界测试 21](#_Toc495668193)

[4.2.1 初始化无输入数据 21](#_Toc495668194)

[4.2.2 删除头结点 22](#_Toc495668195)

[4.2.3 删除后链表为空 22](#_Toc495668196)

[4.3 出错测试 23](#_Toc495668197)

[4.3.1 考生人数错误 23](#_Toc495668198)

[4.3.2 操作码错误 23](#_Toc495668199)

[4.3.3 插入位置不存在 23](#_Toc495668200)

[4.3.4 删除考号不存在 24](#_Toc495668201)

[4.3.5 查找考号不存在 24](#_Toc495668202)

[4.3.6 修改考号不存在 25](#_Toc495668203)

# 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。

随着计算机科学技术的不断成熟，使用计算机对考试报名系统进行管理，具有手工管理所无法比拟的优势。这些优点能够极大地提高学校和学生的效率，也是学校走向信息化、科学化、国际化的重要条件。因此，开发一套考试报名系统具有十分重要的意义。

## 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。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、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。为了链表结构的可靠性，这里选择使用双向链表结构构建链表。

## 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（LNode）与链表类（LinkList），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为了保证数据的安全性，链表节点内的变量统一定义为private。用将链表类声明为链表节点的友元类的方式，以访问链表节点内声明为private的变量。

## 2.3 成员与操作设计

**链表结点类（Student）**

**私有成员：**

Student\* Last;//该节点的上一个节点

Student\* Next;//该节点的下一个节点

int StudentID;//学生考号

string StudentName;//学生姓名

string StudentSex;//学生性别

int StudentAge;//学生年龄

string TestName;//报考类别

**公有操作：**

Student(int ID,

string Name,

string Sex,

int Age,

string Test);//没有声明上下节点的构造函数

Student(int ID,

string Name,

string Sex,

int Age,

string Test,

Student\* LastStudent);//声明上下节点的构造函数

~Student();//析构函数将所有变量清零

void Print();//输出该节点信息

**链表类（List）**

**私有成员：**

Student\* StartPos;//起始位置

Student\* EndPos;//结束位置

int Lenth;//链表长度

**公有成员：**

Student\* CurrentPos;//用来记录当前链表访问的节点

**公有操作：**

List();//构造函数，将所有变量初始化为零

~List();//析构函数，释放链表占用的空间

void Append(int ID,

string Name,

string Sex,

int Age,

string Test,

Student\* Last);

void Append(Student\* NewStudent);

Student\* Find(int ID);//按学号查找学生

Student\* Find(string name);//按姓名查找学生

void Insert(int Pos, Student\* NewStudent);//插入函数

void Change(int ID);//按学号更改学生信息

void Change(string Name);//按姓名更改学生信息

void Delete(int ID);//按学号删除学生

void Delete(string Name);//按姓名删除学生

void Print();//输出链表所有元素

## 2.4 系统设计

先创建空的链表，根据输入将信息添加至链表中。信息输入完毕后，进入循环体中，根据用户输入的内容确定操作。若用户选择退出程序，则退出循环，并清空链表释放内存。

# 3 实现

## 3.1 插入功能的实现

### 3.1.1 插入功能核心代码

void List::Insert(int Pos,Student\* NewStudent)

{

if (Lenth == 0){

StartPos = NewStudent;

EndPos = NewStudent;

}

else{

if (Pos > Lenth){

cout << "已超过当前列表总长度，该学员已被添加至列表最后一位\n";

this->Append(NewStudent);

}

else if (Pos == 1){

StartPos->Last = NewStudent;

NewStudent->Next = StartPos;

StartPos = NewStudent;

}

else if (Pos < 1) {

cout << "由于插入位置小于等于0该学员已被添加至第一位\n"

<< "请尽量不要输入小于等于0的数\n";

StartPos->Last = NewStudent;

NewStudent->Next = StartPos;

StartPos = NewStudent;

}

else{

CurrentPos = StartPos;

for (int i = 1; i < Pos; i++){

CurrentPos = CurrentPos->Next;

}

NewStudent->Last = CurrentPos->Last;

NewStudent->Next = CurrentPos;

CurrentPos->Last->Next = NewStudent;

CurrentPos->Last = NewStudent;

}

}

Lenth++;

}

3.1.2 插入功能截屏示例

## 3.2 删除功能的实现

### 3.2.1 删除功能核心代码

void List::Delete(int ID)

{

Student\* TempStudent = this->Find(ID);

if (TempStudent == 0)

{

return;

}

else

{

char Juge;

cout << "是否确认删除？ 按y确认 其他键返回\n";

cin >> Juge;

if (Juge == 'Y'||Juge=='y')

{

if (TempStudent == StartPos && TempStudent == EndPos)

{

delete TempStudent;

StartPos = 0;

EndPos = 0;

CurrentPos = 0;

Lenth--;

cout << "已成功删除\n";

}

else if (TempStudent == StartPos)

{

TempStudent->Next->Last = TempStudent->Last;

StartPos = TempStudent->Next;

delete TempStudent;

Lenth--;

cout << "已成功删除\n";

}

else if (TempStudent == EndPos)

{

TempStudent->Last->Next = TempStudent->Next;

EndPos = TempStudent->Last;

delete TempStudent;

Lenth--;

cout << "已成功删除\n";

}

else

{

TempStudent->Last->Next = TempStudent->Next;

TempStudent->Next->Last = TempStudent->Last;

delete TempStudent;

Lenth--;

cout << "已成功删除\n";

}

}

else

{

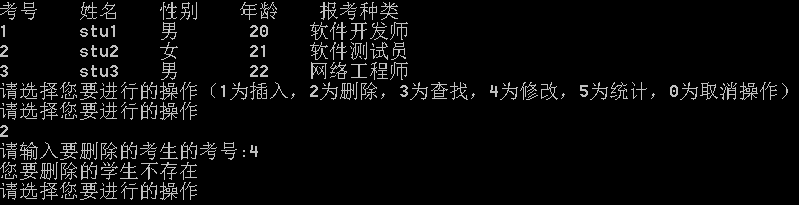
cout << "已终止删除进程\n";

}

}

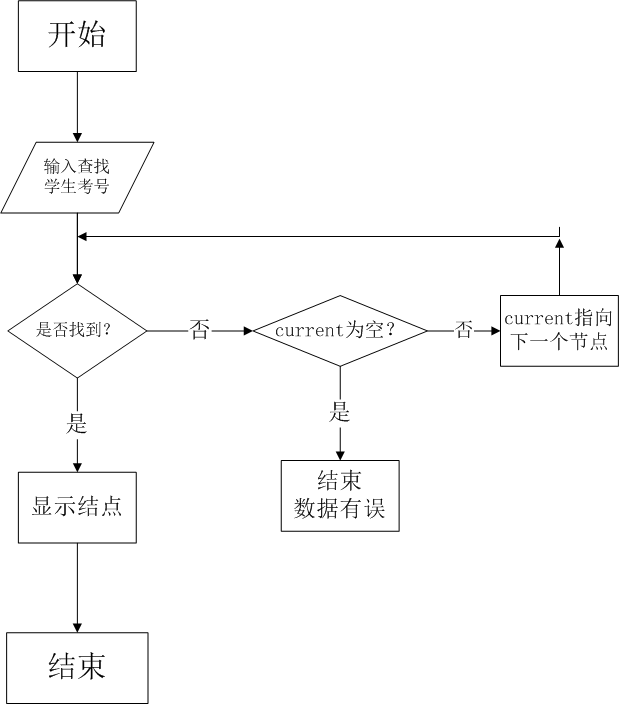
### }3.2.3 删除功能截屏示例





## 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

bool LinkList::Find() const

{

LNode \*current = m\_first->m\_next;//当前寻找的节点

int flag = 0; //找到的标志

while ((flag == 0) && (current != nullptr))

{

if (current->m\_studentID == studentID)

{

flag = 1;

}

else

{

current = current->m\_next;

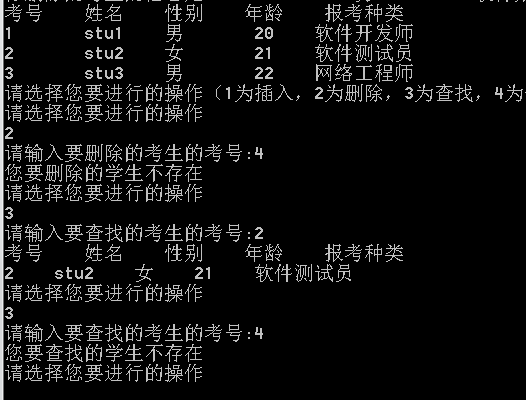
}

}

}

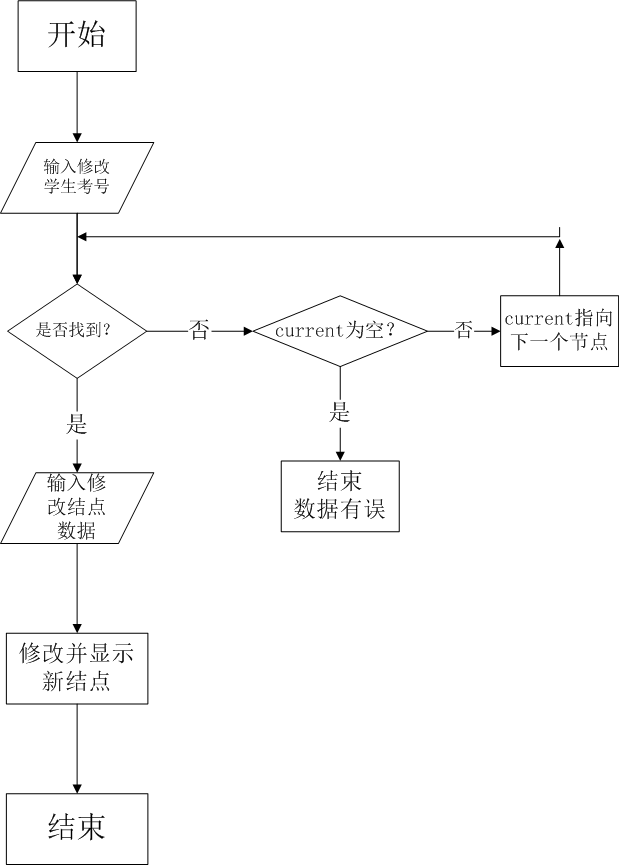
### 3.3.3 查找功能截图示例





## 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

bool LinkList::Change()

{

LNode \*current = m\_first->m\_next;//当前修改的节点

int flag = 0; //找到的标志

while ((flag == 0) && (current != nullptr))

{

if (current->m\_studentID == studentID)

{

flag = 1;

}

else

{

current = current->m\_next;

}

}

{

current->m\_studentID = studentID;

current->m\_studentName = studentName;

current->m\_sex = sex;

current->m\_age = age;

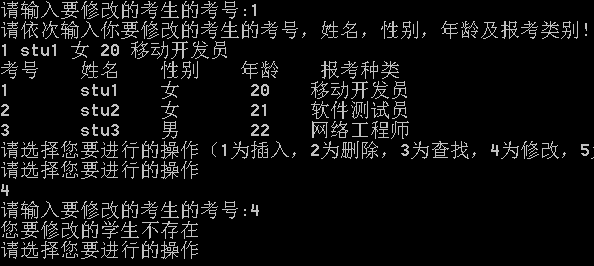
current->m\_category = category;

}

}

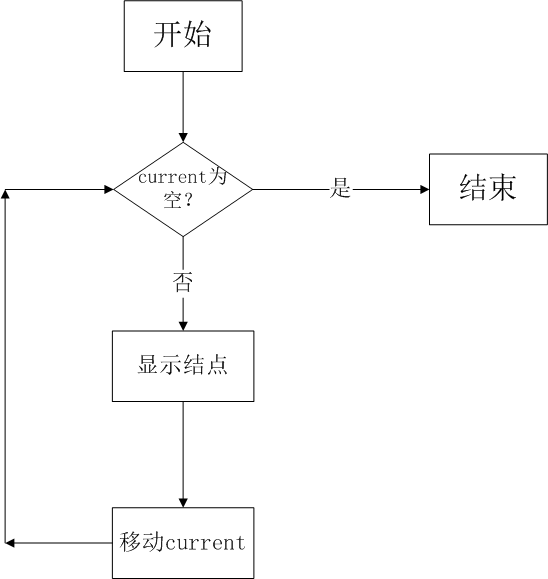
### 3.4.3 修改功能截屏示例





## 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

while (current != nullptr)

{

if (current->m\_studentID != "")

{

cout << current->m\_studentID << " "

<< current->m\_studentName << " "

<< current->m\_sex << " "

<< current->m\_age << " "

<< current->m\_category << " "

<< endl;

}

current = current->m\_next;//移动到下一个节点

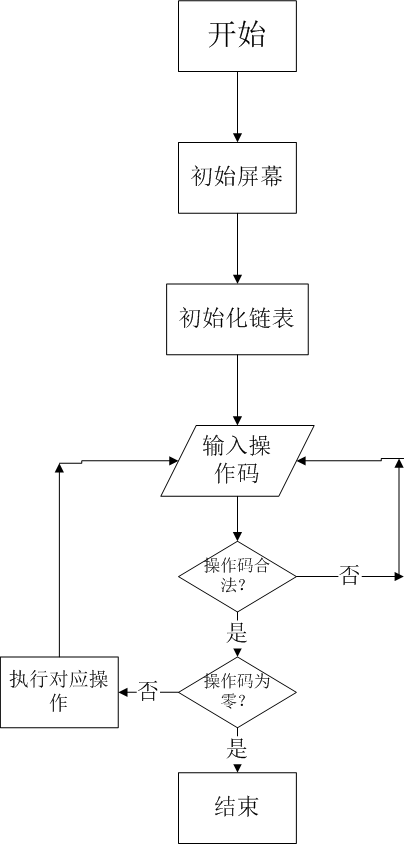
}

### 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 (operateCode != '0')

{

switch (operateCode)

{

case '1': L.Insert(); break;

case '2': L.Delete(); break;

case '3': L.Find(); break;

case '4': L.Change(); break;

case '5': L.Display(); break;

//如果用户输入操作要求错误，提示用户重新输入

default: cout << "抱歉，您想要进行的操作不存在，请重新输入！" << endl; break;

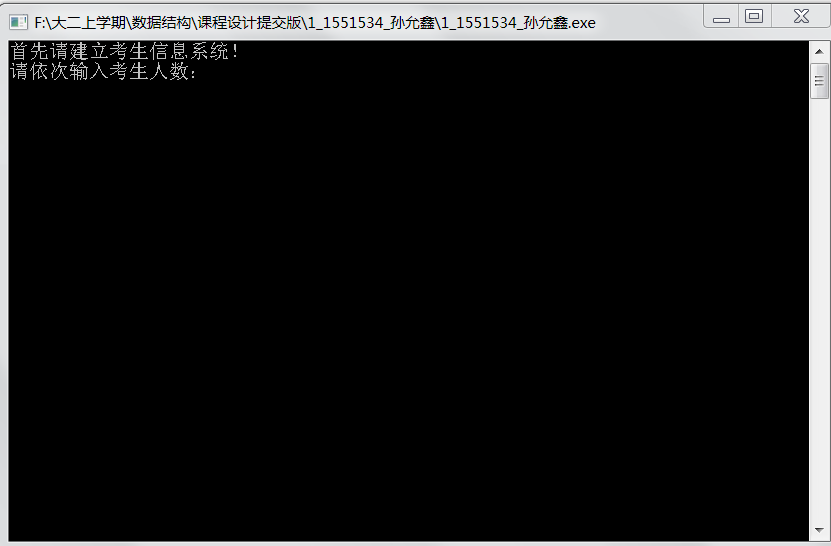
}

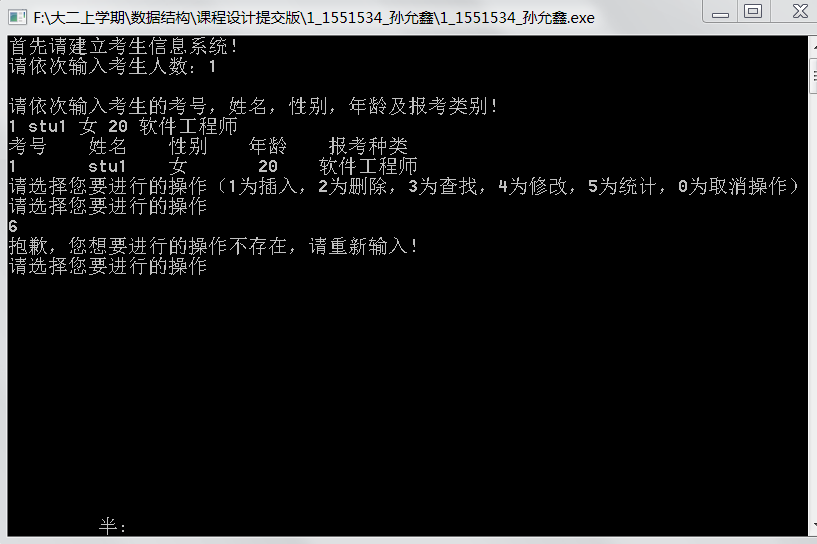
cout << "请选择您要进行的操作" << endl;

cin >> operateCode;

}

### 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 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4的考生

**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：**



### 4.1.3 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 统计功能测试

**测试用例：**统计当前数据

**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：**



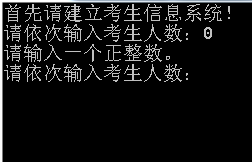
## 4.2 边界测试

### 4.2.1 初始化无输入数据

**测试用例：**初始无输入数据

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

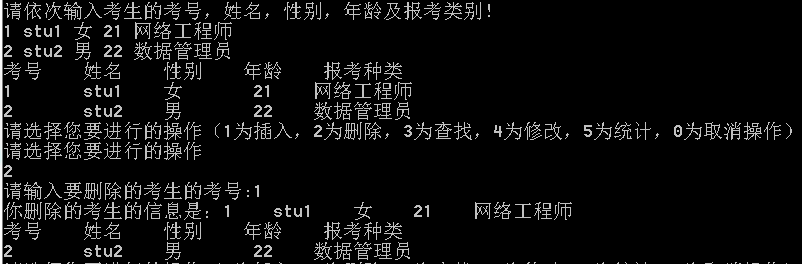


### 4.2.2 删除头结点

**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：**

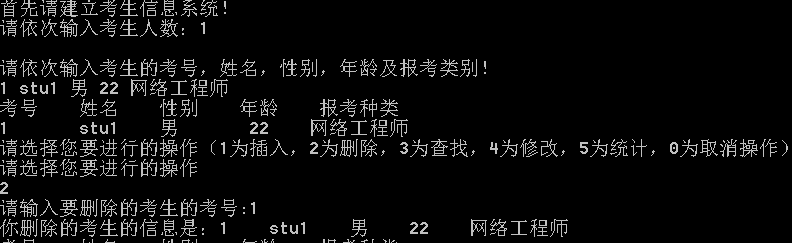


### 4.2.3 删除后链表为空

**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：**



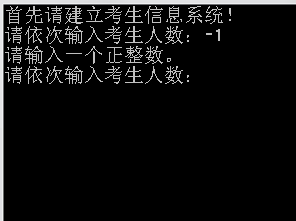
## 4.3 出错测试

### 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：**

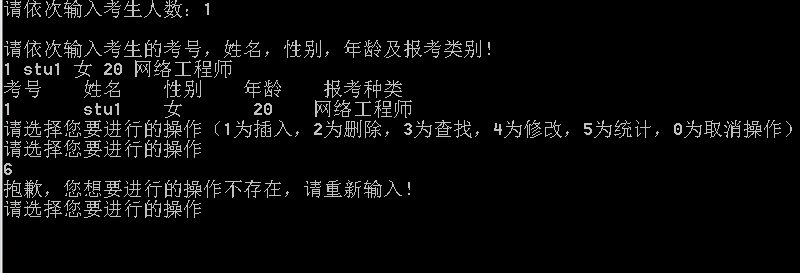
****

### 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：**

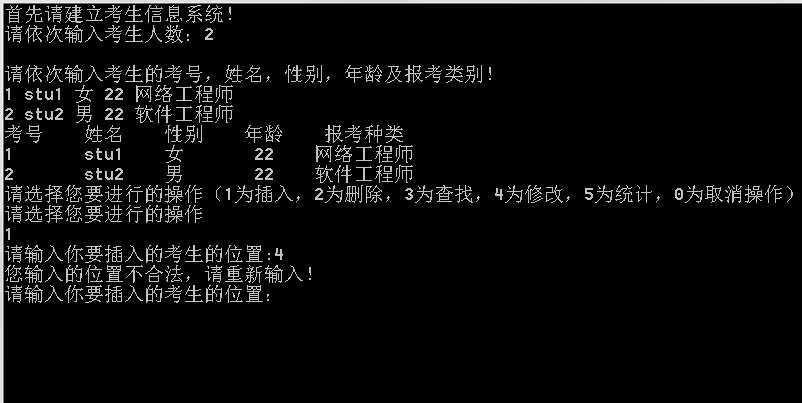
****

### 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：**

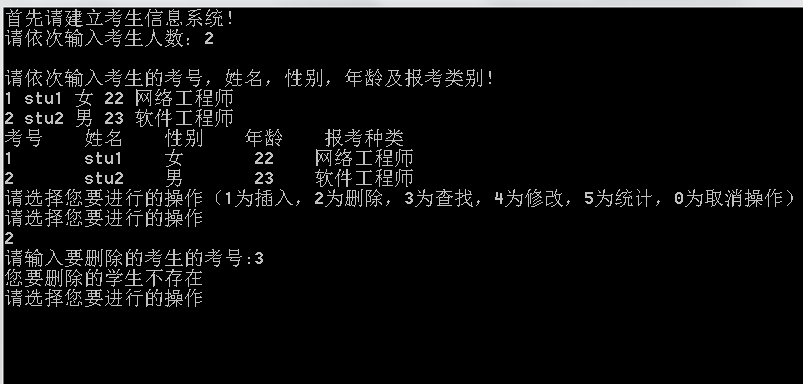


### 4.3.4 删除考号不存在

**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：**

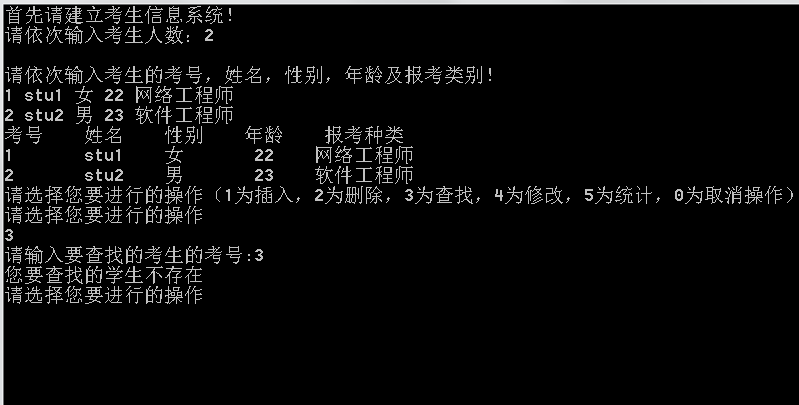
****

### 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：**

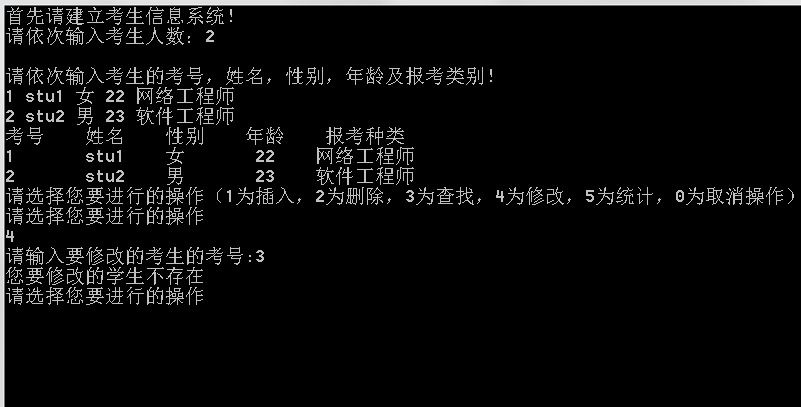
****

### 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：**



# 