项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 伊啸

学 号： 1951220

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 3](#_Toc53078146)

[1.1 背景分析 3](#_Toc53078147)

[1.2 功能分析 3](#_Toc53078148)

[2 设计 4](#_Toc53078149)

[2.1 数据结构设计 4](#_Toc53078150)

[2.2 类结构设计 4](#_Toc53078151)

[2.3 成员与操作设计 4](#_Toc53078152)

[3 实现 6](#_Toc53078153)

[3.1 插入功能的实现 6](#_Toc53078154)

[3.1.1 插入功能流程图 6](#_Toc53078155)

[3.1.2 插入功能核心代码 8](#_Toc53078156)

[3.1.3 插入功能截屏示例 8](#_Toc53078157)

[3.2 删除功能的实现 10](#_Toc53078158)

[3.2.1 删除功能流程图 10](#_Toc53078159)

[3.2.2 删除功能核心代码 10](#_Toc53078160)

[3.2.3 删除功能截屏示例 11](#_Toc53078161)

[3.3 查找功能的实现 11](#_Toc53078162)

[3.3.1 查找功能流程图 11](#_Toc53078163)

[3.3.2 查找功能核心代码 11](#_Toc53078164)

[3.3.3 查找功能截图示例 12](#_Toc53078165)

[3.4 修改功能的实现 13](#_Toc53078166)

[3.4.1 修改功能流程图 13](#_Toc53078167)

[3.4.2 修改功能核心代码 13](#_Toc53078168)

[3.4.3 修改功能截屏示例 15](#_Toc53078169)

[3.5 统计功能的实现 16](#_Toc53078170)

[3.5.1 统计功能流程图 16](#_Toc53078171)

[3.5.2 统计功能核心代码 16](#_Toc53078172)

[3.5.3 统计功能截屏示例 17](#_Toc53078173)

[4 测试 18](#_Toc53078174)

[4.1 功能测试 18](#_Toc53078175)

[4.1.1 插入功能测试 18](#_Toc53078176)

[4.1.2 删除功能测试 18](#_Toc53078177)

[4.1.3 查找功能测试 19](#_Toc53078178)

[4.1.4 修改功能测试 19](#_Toc53078179)

[4.1.5 统计功能测试 20](#_Toc53078180)

[4.2 边界测试 20](#_Toc53078181)

[4.2.1 初始化无输入数据 20](#_Toc53078182)

[4.2.2 删除头结点 21](#_Toc53078183)

[4.2.3 删除后链表为空 21](#_Toc53078184)

[4.3 出错测试 22](#_Toc53078185)

[4.3.1 考生人数错误 22](#_Toc53078186)

[4.3.2 操作码错误 22](#_Toc53078187)

[4.3.3 插入位置不存在 23](#_Toc53078188)

[4.3.4 删除考号不存在 23](#_Toc53078189)

[4.3.5 查找考号不存在 23](#_Toc53078190)

[4.3.6 修改考号不存在 24](#_Toc53078191)

1 分析

## 背景分析

随着信息时代的发展以及学生人数的急剧增加，以往传统的考试报名耗费了大量人力物力，在逐渐被淘汰，取而代之的是通过计算机系统来构建一个考试报名系统，在节省大量人力物力的同时，出错率也大大降低。与此同时，建立一个考试报名系统也能够为高校加强考试管理，对学生的个人信息也能够起到有效的保护作用。因此，建立一个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，首先应该具备的功能就是能够输入同学们的考试报名情况并且可以显示出来。其次，考试报名系统还应该具有插入、删除、修改等功能，以保证同学可以随时更改自己的个人信息。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。

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（InfoNode）与链表类（NodeList）。为方便处理，本系统采用struct描述链表结点类（InfoNode），这样使得链表结点类（NodeList）可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类（InfoNode）**

公有成员：

string num;//表示学生学号

string name;//表示学生姓名

string sex;//表示学生性别

string age;//表示学生年龄

string category;//表示学生报考类别

InfoNode\* next;

公有操作：

InfoNode::InfoNode() {

num = "";

name = "";

age = "";

sex = "";

category = "";

next = NULL;

}

**链表类（NodeList）**

私有成员：

int len;//链表的长度

InfoNode\* head;//链表的头结点

公有操作：

NodeList();//构造函数

~NodeList() { };//析构函数

int rlen();//返回len值

InfoNode\* Insert(InfoNode\* infonode,int position);//实现插入功能

InfoNode\* Delete(string student\_number);//实现删除功能

InfoNode\* Search(string student\_number);//实现查找功能

InfoNode\* Modify(string student\_number);//实现修改功能

void show(const InfoNode\* infonode);//输出单个节点

void showlist(const NodeList& nodelist);//输出整个链表

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

InfoNode\* p = head;

for (int i = 0; i < position - 1; i++) {

p = p->next;

}//找到要插入节点的前一个节点

if (p != nullptr && p->next != nullptr) {//在中间插入一个节点

infonode->next = p->next;

p->next = infonode;

len++;

return head;

}

else if (p != nullptr) {//在最后一个节点后插入一个节点

infonode->next = nullptr;

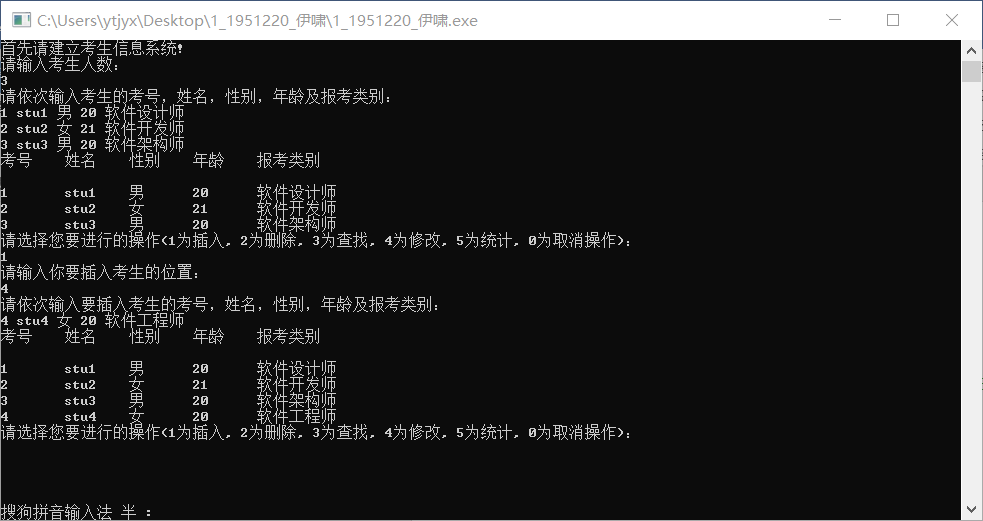
p->next = infonode;

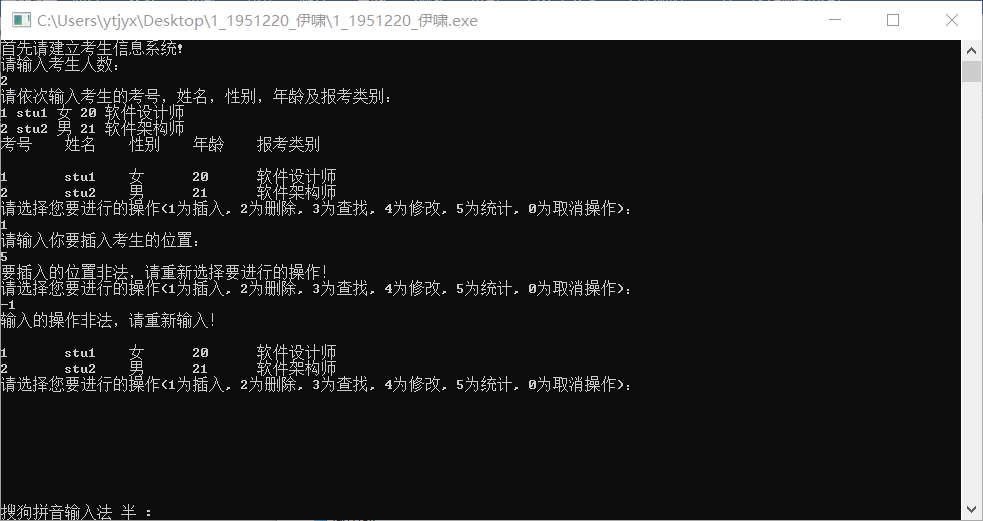
len++;

return head;

}

3.1.3 插入功能截屏示例





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

while (q) {

if (student\_number == q->num) {

p->next = q->next;

len--;

return head;

}

else {

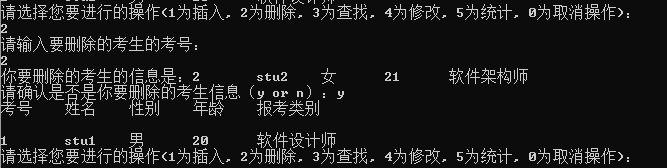
p = p->next;

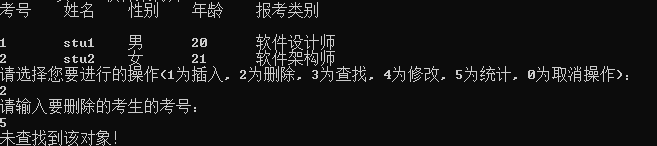
q = q->next;

}

}

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

InfoNode\* p = head;

while (p) {

if (student\_number == p->num)

return p;

else

p = p->next;

}

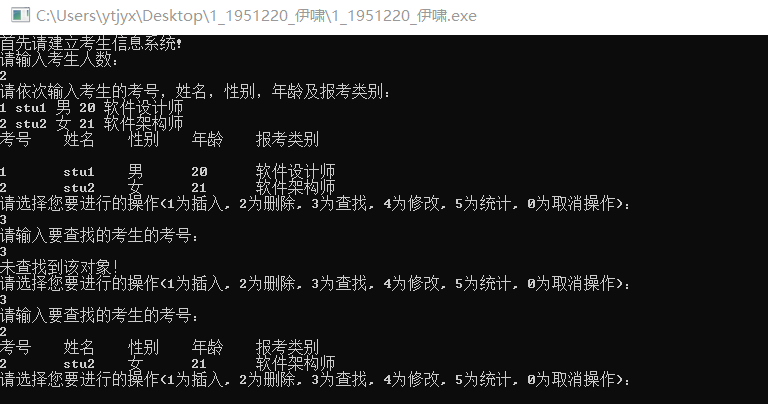
if (p != nullptr && p->next == nullptr) {

return nullptr;

}

return nullptr;

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

while (p) {

if (p->num == student\_number) {

string information;

cout << "请输入要修改考生的信息：（考号，姓名，性别，年龄，报考类别）" << endl;

while (cin >> information) {

if (information == "考号") {

string new\_num;

cout << "请输入修改后的新考号：" << endl;

cin >> new\_num;

p->num = new\_num;

return head;

}

else if (information == "姓名") {

string new\_name;

cout << "请输入修改后的新姓名：" << endl;

cin >> new\_name;

p->name = new\_name;

return head;

}

else if (information == "性别") {

string new\_sex;

cout << "请输入修改后的新性别：" << endl;

cin >> new\_sex;

p->sex = new\_sex;

return head;

}

else if (information == "年龄") {

string new\_age;

cout << "请输入修改后的新年龄：" << endl;

cin >> new\_age;

p->age = new\_age;

return head;

}

else if (information == "报考类别") {

string new\_category;

cout << "请输入修改后的新报考类别：" << endl;

cin >> new\_category;

p->category = new\_category;

return head;

}

else {

cout << "不存在这样的考生信息!" << endl;

cout << "请输入要修改考生的信息：（考号，姓名，性别，年龄，报考类别）" << endl;

}

}

}

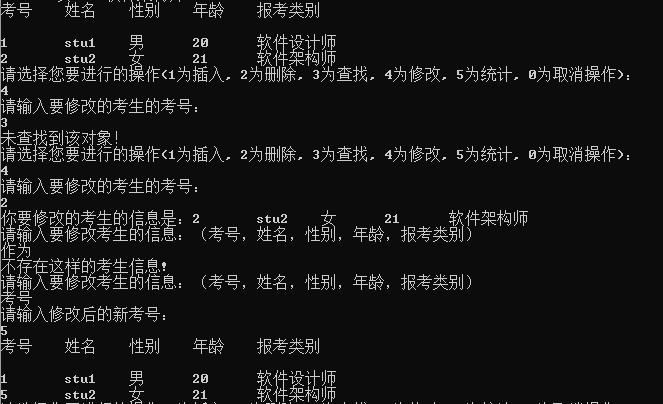
else

p = p->next;

}

cout << "未找到要修改的对象！" << endl;

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

void NodeList::show(const InfoNode\* infonode) {

if (infonode != nullptr)

cout << infonode->num << " " << infonode->name << " " << infonode->sex << " " << infonode->age << " " << infonode->category << endl;

}//输出单个节点

void NodeList::showlist(const NodeList& nodelist) {

InfoNode\* p = nodelist.head;

while (p) {

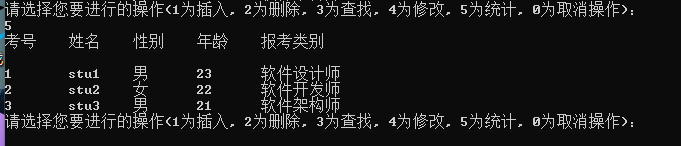
show(p);

p = p->next;

}

}//输出整个链表

3.5.3 统计功能截屏示例



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

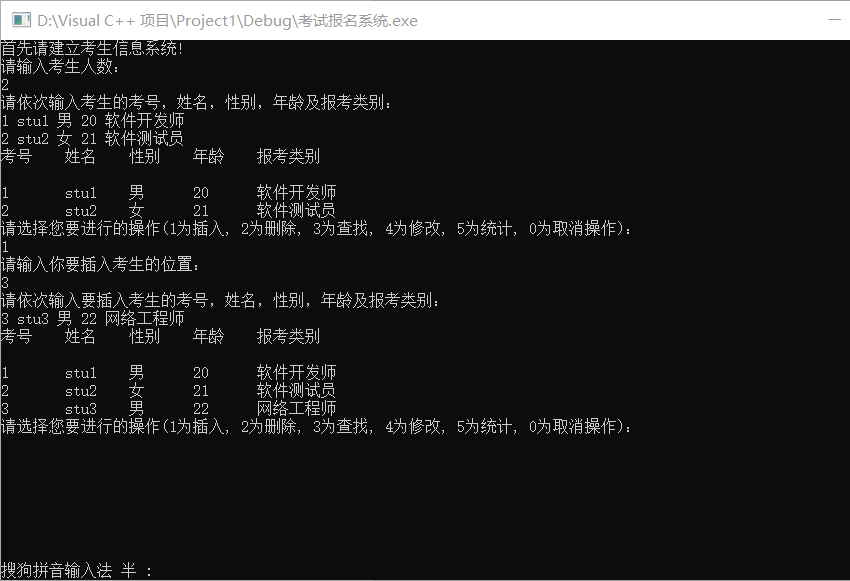
**预期结果**：

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

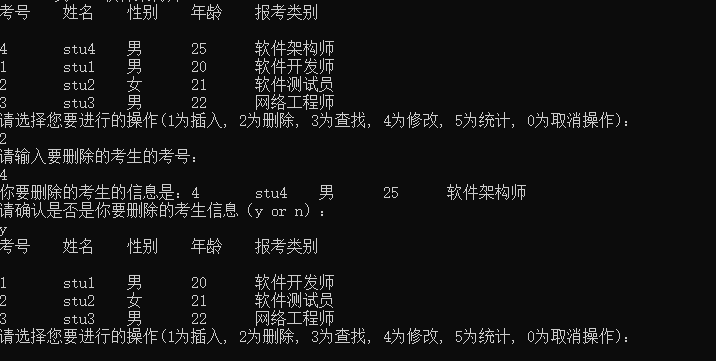
**测试用例：**删除考号为4的考生

**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



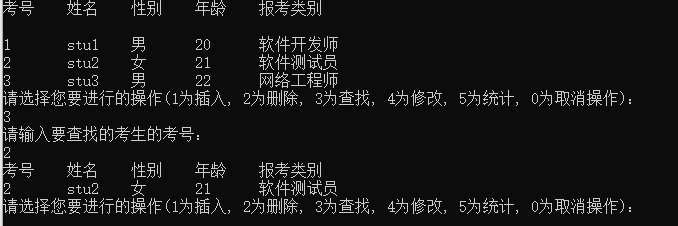
4.1.3 查找功能测试

**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：**



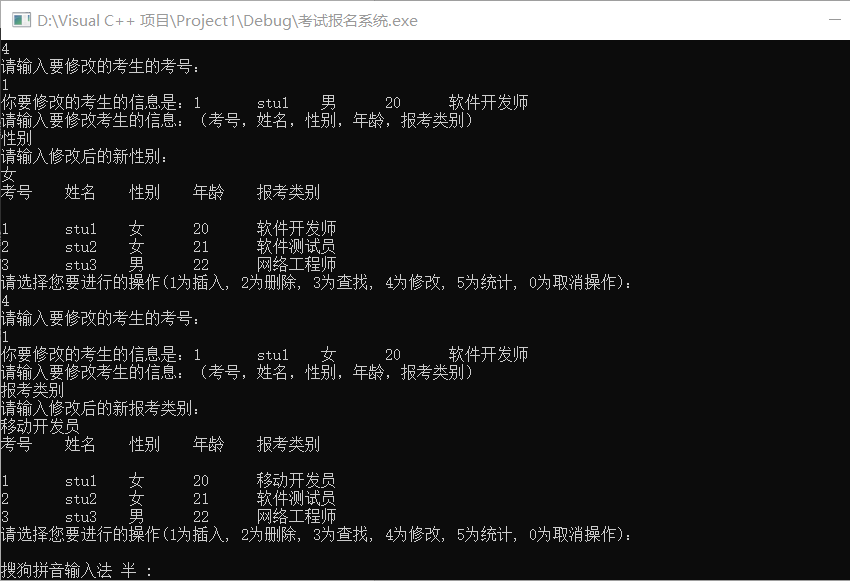
4.1.4 修改功能测试

**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

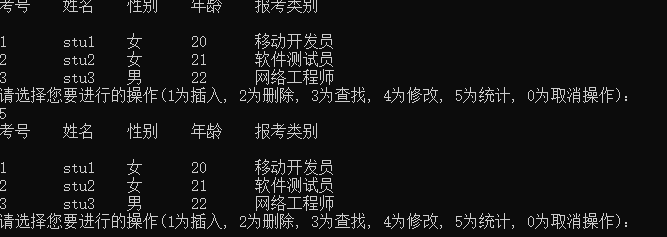
**预期结果：**

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：**



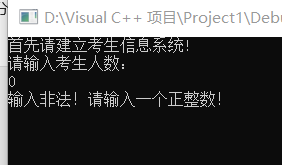
4.2 边界测试

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：**

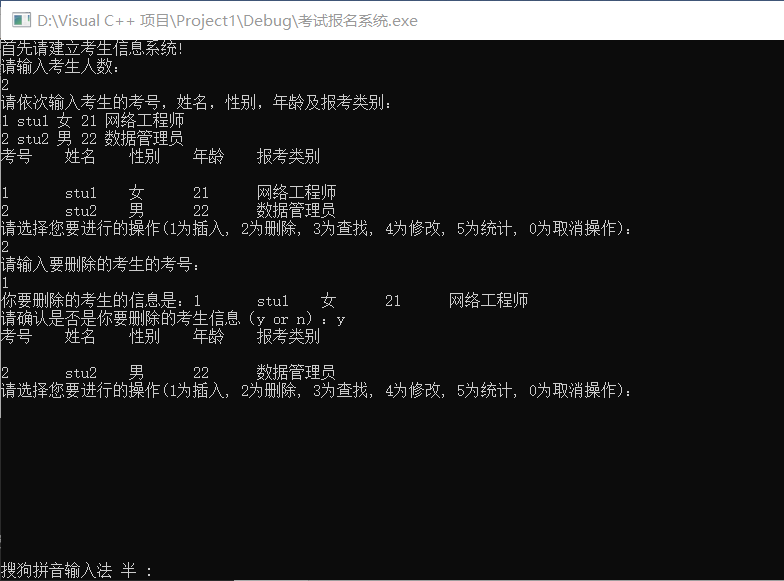
****

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：**

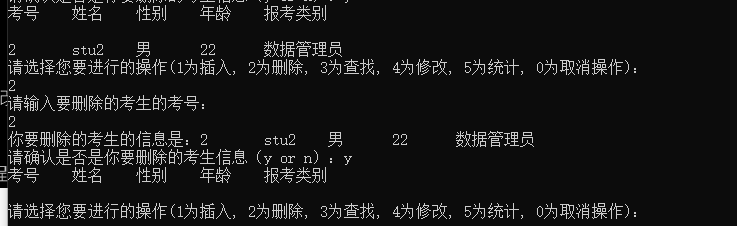
****

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：**

****

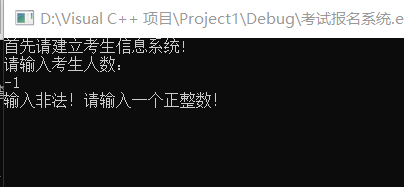
4.3 出错测试

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：**

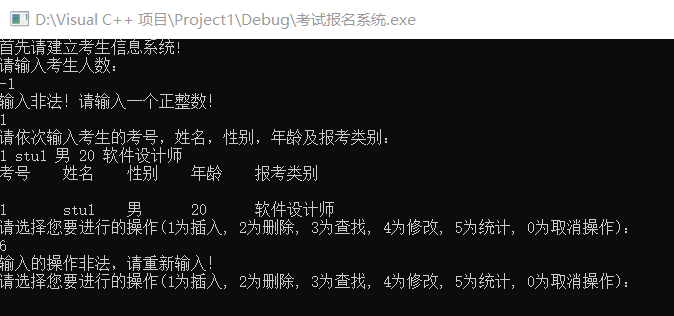


4.3.2 操作码错误

**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：**

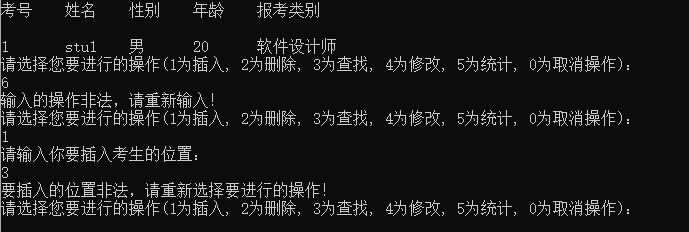


4.3.3 插入位置不存在

**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：**

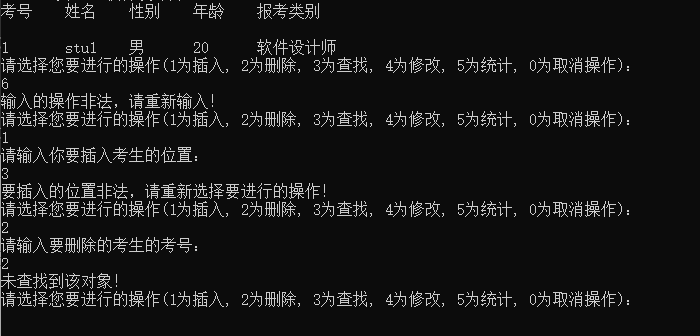
****

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：**

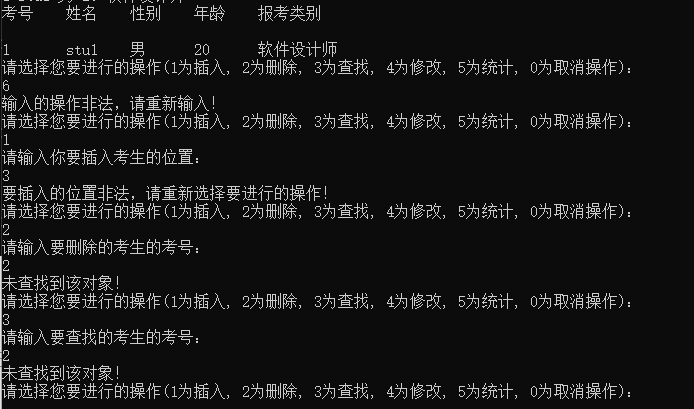
****

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：**

****