项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 张喆

学 号： 1754060

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 3](#_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 6](#_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 9](#_Toc495668170)

[3.3.1 查找功能流程图 9](#_Toc495668171)

[3.3.2 查找功能核心代码 9](#_Toc495668172)

[3.3.3 查找功能截图示例 10](#_Toc495668173)

[3.4 修改功能的实现 11](#_Toc495668174)

[3.4.1 修改功能流程图 11](#_Toc495668175)

[3.4.2 修改功能核心代码 11](#_Toc495668176)

[3.4.3 修改功能截屏示例 12](#_Toc495668177)

[3.5 统计功能的实现 13](#_Toc495668178)

[3.5.1 统计功能流程图 13](#_Toc495668179)

[3.5.2 统计功能核心代码 13](#_Toc495668180)

[3.5.3 统计功能截屏示例 14](#_Toc495668181)

[3.6 总体系统的实现 14](#_Toc495668182)

[3.6.1 总体系统流程图 14](#_Toc495668183)

[3.6.2 总体系统核心代码 14](#_Toc495668184)

[3.6.3 总体系统截屏示例 16](#_Toc495668185)

[4 测试 1](#_Toc495668186)7

[4.1 功能测试 1](#_Toc495668187)7

[4.1.1 插入功能测试 1](#_Toc495668188)7

[4.1.2 删除功能测试 18](#_Toc495668189)

[4.1.3 查找功能测试 19](#_Toc495668190)

[4.1.4 修改功能测试 19](#_Toc495668191)

[4.1.5 统计功能测试 19](#_Toc495668192)

[4.2 边界测试 2](#_Toc495668193)0

[4.2.1 初始化无输入数据 2](#_Toc495668194)0

[4.2.2 删除头结点 2](#_Toc495668195)0

[4.2.3 删除后链表为空 2](#_Toc495668196)1

[4.3 出错测试 2](#_Toc495668197)2

[4.3.1 考生人数错误 2](#_Toc495668198)2

[4.3.2 操作码错误 2](#_Toc495668199)2

[4.3.3 插入位置不存在 2](#_Toc495668200)2

[4.3.4 删除考号不存在 2](#_Toc495668201)3

[4.3.5 查找考号不存在 2](#_Toc495668202)3

[4.3.6 修改考号不存在 2](#_Toc495668203)3

# 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，以及如果用户输入的数据有问题的时候可以做出简单的检查和提示。

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、健壮性的检测等功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求频繁的插入、删除、修改操作，而链表进行插入、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，并用头指针指向头结点，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，更方便单链表的前插法的设计，也不需要将第一个和最后一个结点做特殊处理。

## 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（NODE）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。为方便处理，本系统将链表类(LinkList)声明为链表结点类(NODE)的友元，这样使得链表结点类（LinkList）可以访问链表结点。

同时将考生的信息封装到Stu结构体中，使得结点类的定义更简洁，更方便插入、删除、修改等功能。

## 2.3 成员与操作设计

**考生信息类（Stu）**

**公有成员：**

int \_ExamNum; //准考证号

string \_name; //姓名

string \_sex; //性别

int \_age; //年龄

string \_AppCate; //报考类别

**考生信息的构造函数：**

Stu() = default; //默认构造函数

Stu(const int ExamNum, const string &name, const string &sex,

const int age, const string &AppCate); //新考生信息的构造函数

Stu::Stu(const int ExamNum, const string &name, const string &sex,

const int age, const string &AppCate)

:\_ExamNum(ExamNum),\_name(name),\_sex(sex),\_age(age),\_AppCate(AppCate){}

**链表结点类（NODE）**

**私有成员：**

Stu data; //封装考生信息于变量data

NODE \*next; //指针域,用于指向下一结点信息

**友元：**

friend class LinkList; //将LinkList类声明为NODE类的友元

//使得LinkList可以访问结点类的私有成员

//而其他地方不可以方位节点类,防止对数据的误处理

**链表类（LinkList）**

**私有成员：**

NODE \* head = NULL; //头指针, 指向头结点

int \_size = 0; //链表中有效信息的数量

**公有操作：**

LinkList() = default; //默认构造函数

LinkList(int size); //指定长度的链表构造函数

~LinkList(); //析构函数: 用于销毁链表中所有结点

void InitLink(); //用于已经构建好的链表的数据填充

void OutputLink(); //打印考生信息表

void DestroyLink(); //从头结点开始销毁链表中所有结点

bool FindLink(int num, Stu &buf);

//寻找考生号为num的考生信息,如果找到储存到buf中, 否则返回false

void InsertLink(int index, bool flag);

//flag用于标识前插法和后插法, 在index位置的(前或后)插入考生信息

bool DeleteLink(int num, Stu &buf);

//删除考生号为num的考生信息, 如果存在该考生则储存到buf中

void ChangeLink(int num);

//修改考生号为num的考生信息, 否则输出相应的信息提示

## 2.4 系统设计

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

# 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

用户输入插入位置部分

while (index<0 || index>this->\_size)

//如果插入的位置非法, 则终止程序

{

cerr << "插入位置非法!\n";

cerr << "请重新输入: ";

cin >> index;

}

初始化move结点部分

NODE \*move = (flag == BACK ? head->next : head);

//后插法初始化为首结点; 前插法初始化为头结点,用move->next指向链表的每个结点

插入部分

for (int i=1; (flag == BACK ? move : move->next) != NULL; move = move->next,++i)

{

if (i == index)

{

NODE \*fresh = new NODE;

if (!fresh)

{

cerr << "Memory Alloction Error!\n";

exit(-1);

}

cout << "请依次输入考生的考号, 姓名, 性别, 年龄及报考类型: " << endl;

cin >> fresh->data.\_ExamNum >> fresh->data.\_name >> fresh->data.\_sex

>> fresh->data.\_age >> fresh->data.\_AppCate;

//将新结点链到链表上

fresh->next = move->next;

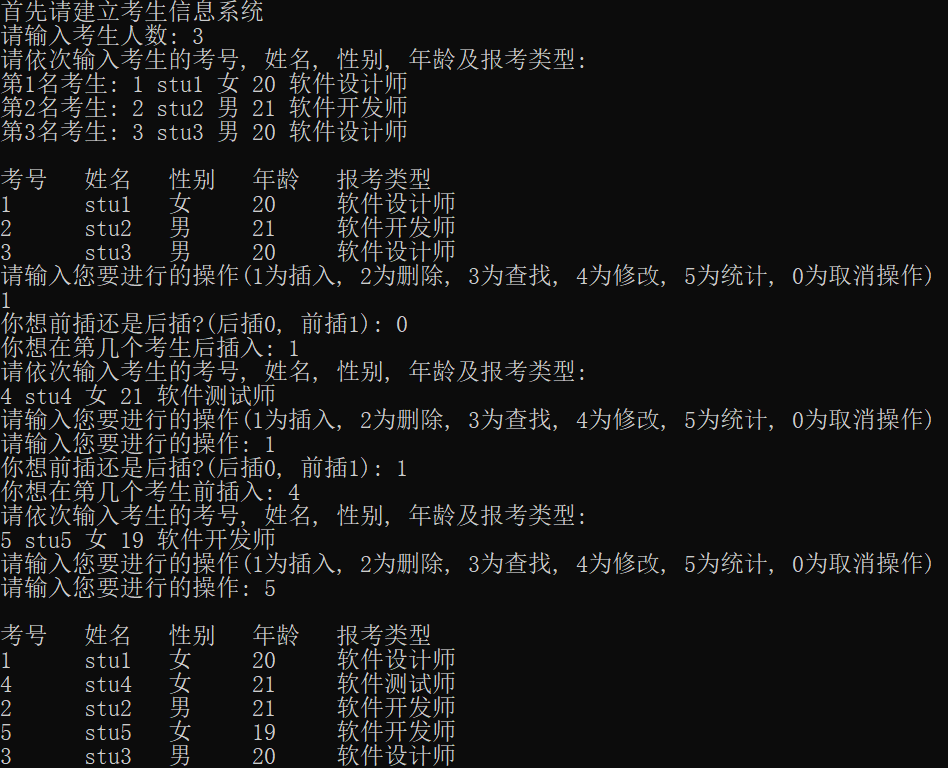
move->next = fresh;

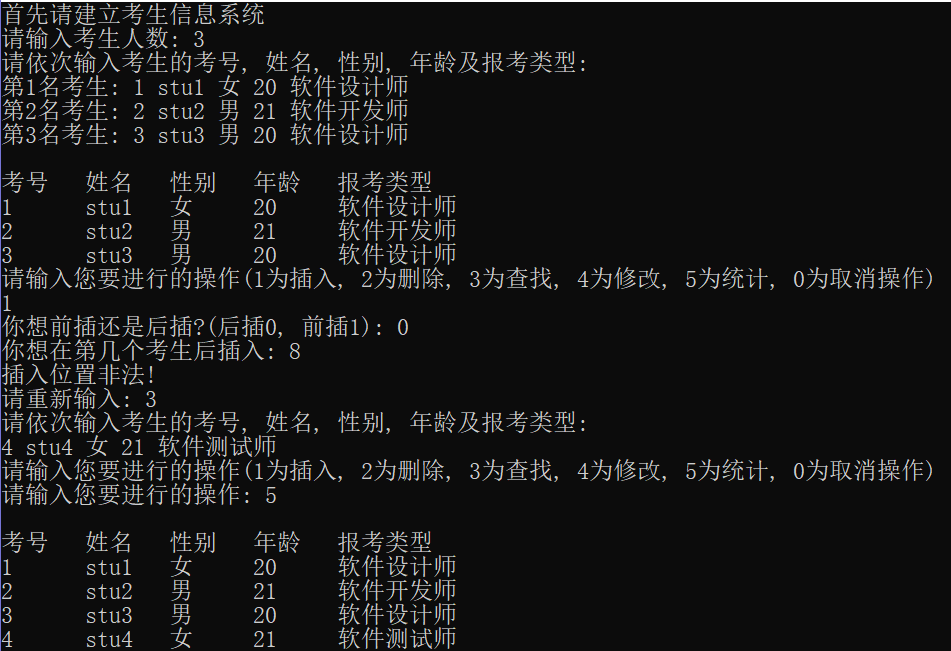
break;

}

}

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

if (!FindLink(num, buf))

//如果不存在该结点, 返回false

{

return false;

}

NODE \*move = head;

//因为需要获取删除结点的前一个结点, 所以move只能指向头结点

NODE \*save;

for (; move->next != NULL; move = move->next)

{

if (move->next->data.\_ExamNum == num)

{

save = move->next; //储存要删除的结点

buf = save->data;

move->next = move->next->next;

//将前一个结点链接到后一个结点上

//释放被删除的空间

delete save;

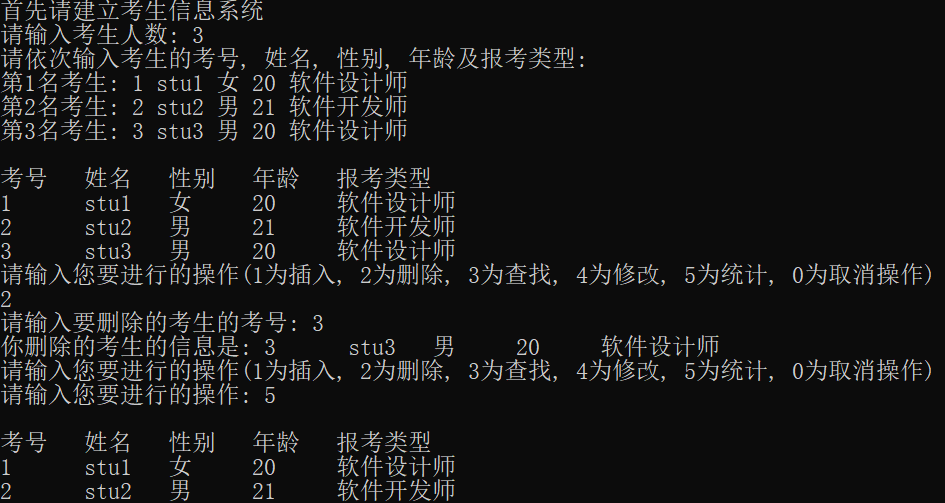
save = NULL;

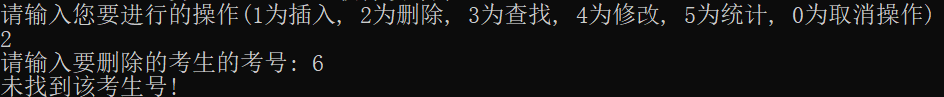
break;

}

### }

### 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

NODE \*move = head->next;

//用move指向每一个链表的结点, 初始化为首结点

for (; move != NULL; move = move->next)

{

if (move->data.\_ExamNum == num)

{

buf = move->data;

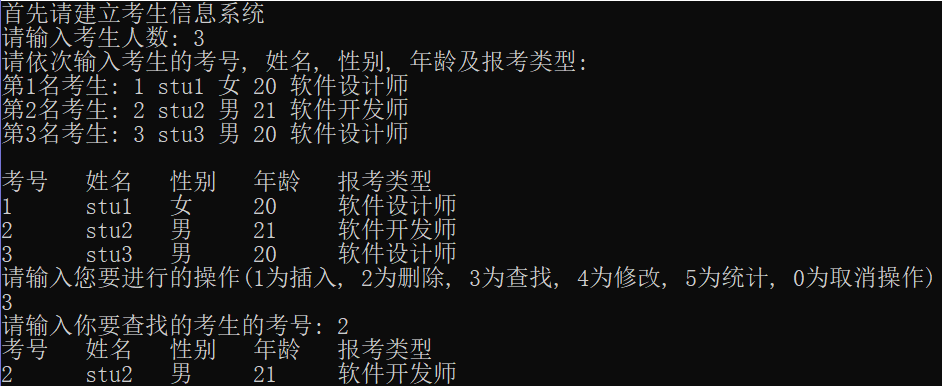
return true;

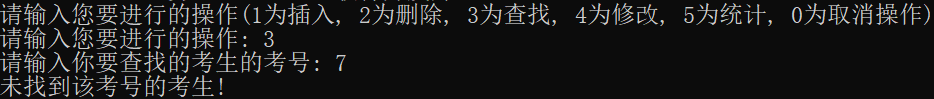
}

}

### return false;

### 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

Stu buf;

if (!FindLink(num, buf))

//如果不存在该结点, 则输出提示信息, 返回调用函数

{

printf("未找到该考生号!");

return;

}

NODE \*move = head->next;

for (; move != NULL; move = move->next)

{

if (move->data.\_ExamNum == num)

{

cout << "请依次输入修改后的考号, 姓名, 性别, 年龄及报考类型: " << endl;

cin >> move->data.\_ExamNum >> move->data.\_name >> move->data.\_sex

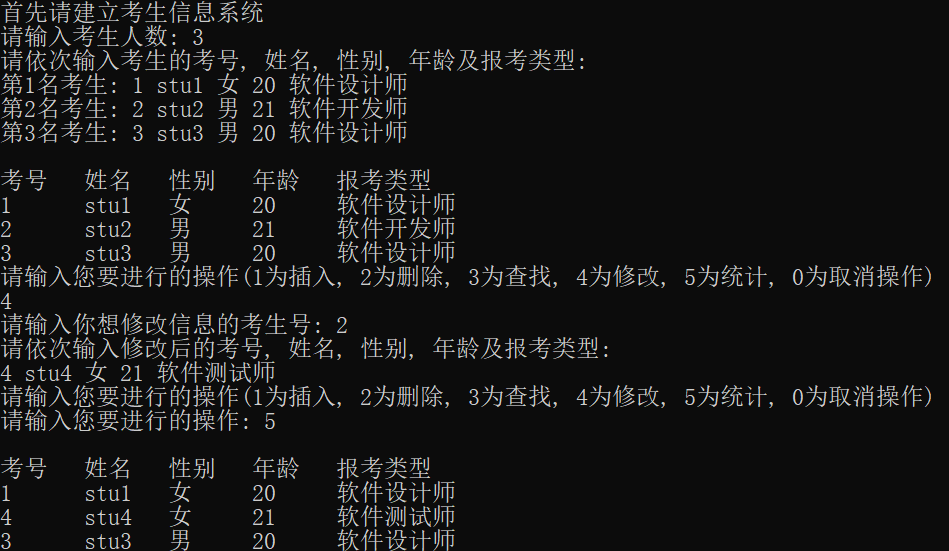
>> move->data.\_age >> move->data.\_AppCate;

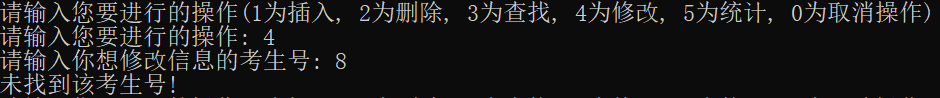
break;

}

### }

### 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

NODE \*move = head->next;

if (this->\_size == 0)

{

cout << "考生信息为空!\n";

return;

}

NODE \*move = head->next;

//用move指向每一个链表的结点, 初始化为首结点

printf("\n考号 姓名 性别 年龄 报考类型\n");

for (; move != NULL; move = move->next)

{

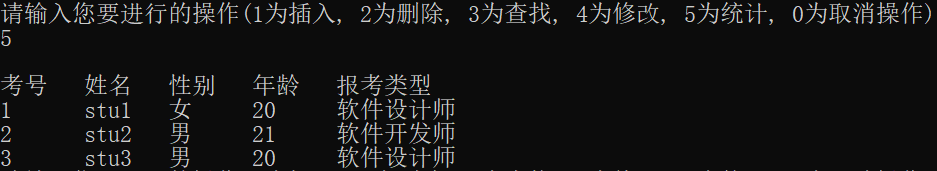
printf("%-7d%-7s%-7s%-7d%-7s\n", move->data.\_ExamNum, move->data.\_name.c\_str(),

move->data.\_sex.c\_str(), move->data.\_age, move->data.\_AppCate.c\_str());

//指定字段宽度打印信息

### }

### 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

//创建长度为size的链表

LinkList list(size);

//读入链表的结点信息

list.InitLink();

//打印初始的链表信息

list.OutputLink();

while (operatorCode)

{

if (1 == operatorCode) //插入

{

printf("你想前插还是后插?(后插0, 前插1): ");

cin >> flag;

printf("你想在第几个考生"); printf(flag == BACK ? "后插入: " : "前插入: ");

cin >> num;

list.InsertLink(num, flag);

}

else if (2 == operatorCode) //删除

{

printf("请输入要删除的考生的考号: ");

cin >> num;

if (list.DeleteLink(num, buf))

{

printf("你删除的考生的信息是: ");

printf("%-7d%-7s%-7s%-7d%-7s\n", buf.\_ExamNum, buf.\_name.c\_str(),

buf.\_sex.c\_str(), buf.\_age, buf.\_AppCate.c\_str());

}

else

{

printf("未找到该考生号!\n");

}

}

else if (3 == operatorCode) //查找

{

printf("请输入你要查找的考生的考号: ");

cin >> num;

if (list.FindLink(num, buf))

{

printf("考号 姓名 性别 年龄 报考类型\n");

printf("%-7d%-7s%-7s%-7d%-7s\n", buf.\_ExamNum, buf.\_name.c\_str(),

buf.\_sex.c\_str(), buf.\_age, buf.\_AppCate.c\_str());

}

else

{

printf("未找到该考号的考生!\n");

}

}

else if (4 == operatorCode) //修改

{

printf("请输入你想修改信息的考生号: ");

cin >> num;

list.ChangeLink(num);

}

else if (5 == operatorCode) //统计

{

list.OutputLink();

}

else //非法操作码处理

{

cout << "操作非法!\n";

cout << "请重新输入\n";

}

printf("请输入您要进行的操作(1为插入, 2为删除, 3为查找, 4为修改, 5为统计, 0为取消操作)\n");

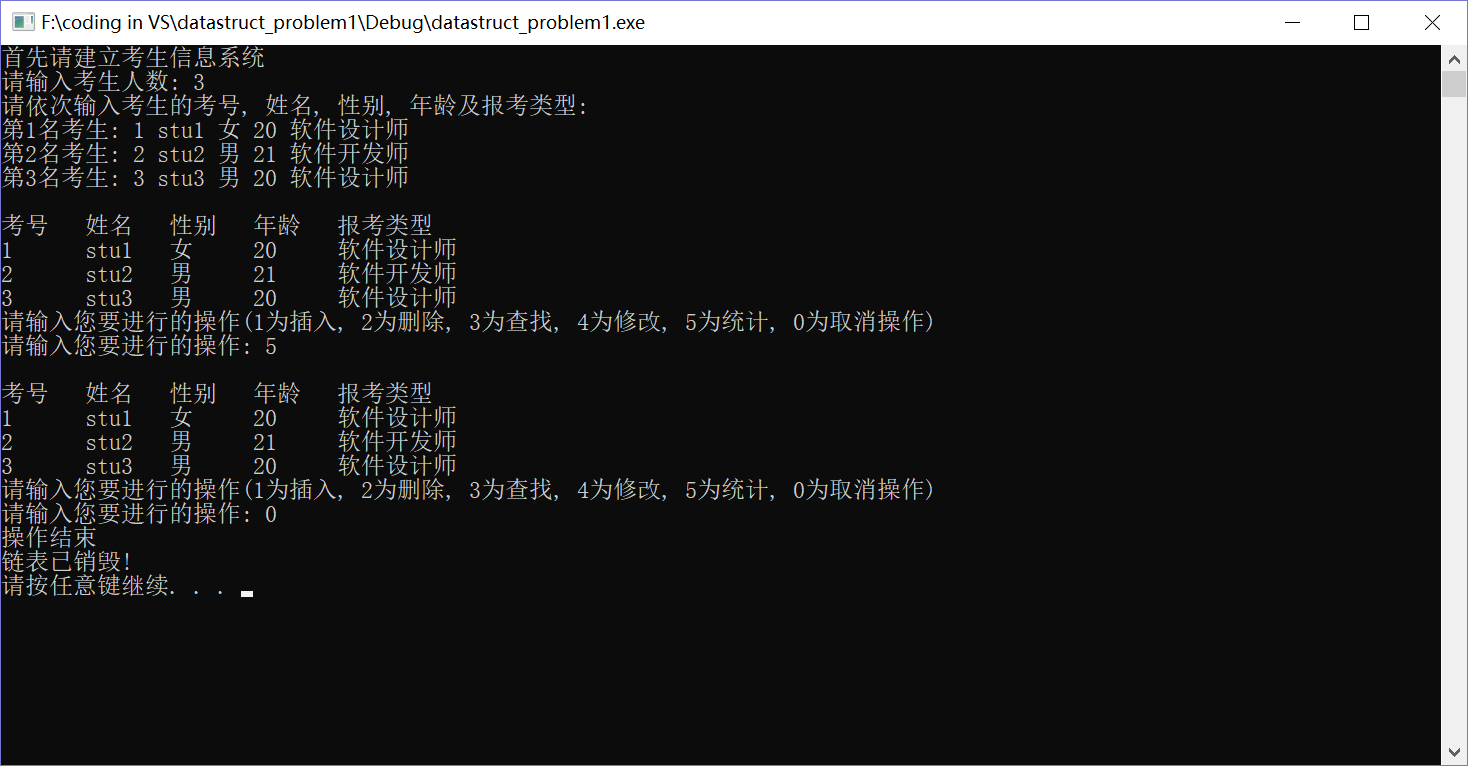
printf("请输入您要进行的操作: ");

cin >> operatorCode;

}

### cout << "操作结束" << endl;

### 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

**前插法测试用例**：4 stu4 女 21 软件测试师

**预期结果**：

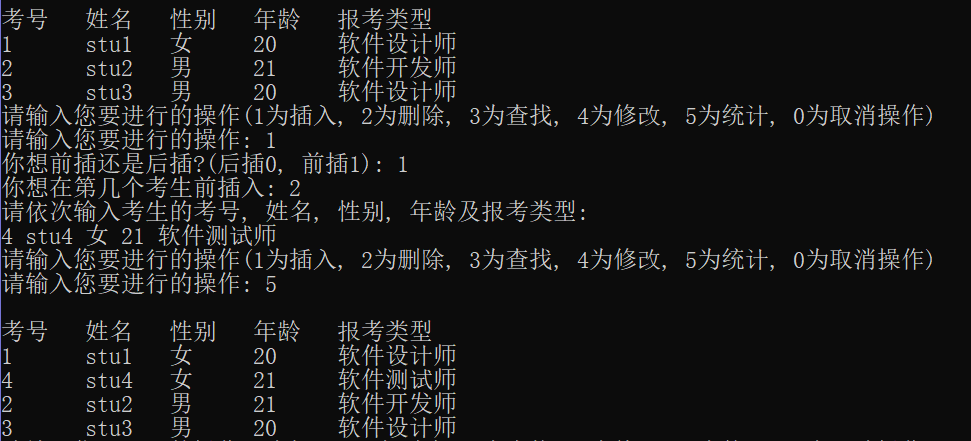
1 stu1 女 20 软件设计师

4 stu4 女 21 软件测试师

2 stu2 男 21 软件开发师

3 stu3 男 20 软件设计师

**实验结果**



**后插法测试用例**：5 stu5 女 19 软件开发师

**预期结果**：

1 stu1 女 20 软件设计师

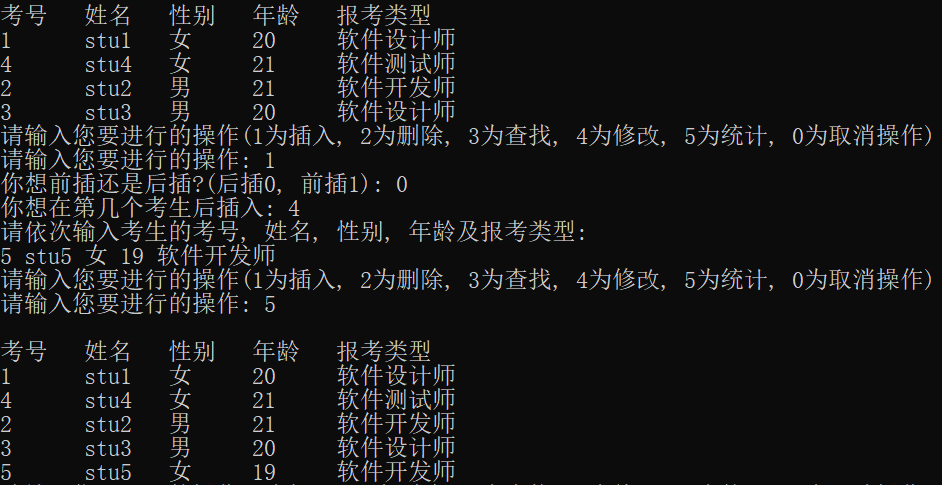
4 stu4 女 21 软件测试师

2 stu2 男 21 软件开发师

3 stu3 男 20 软件设计师

5 stu5 女 19 软件开发师

**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：**

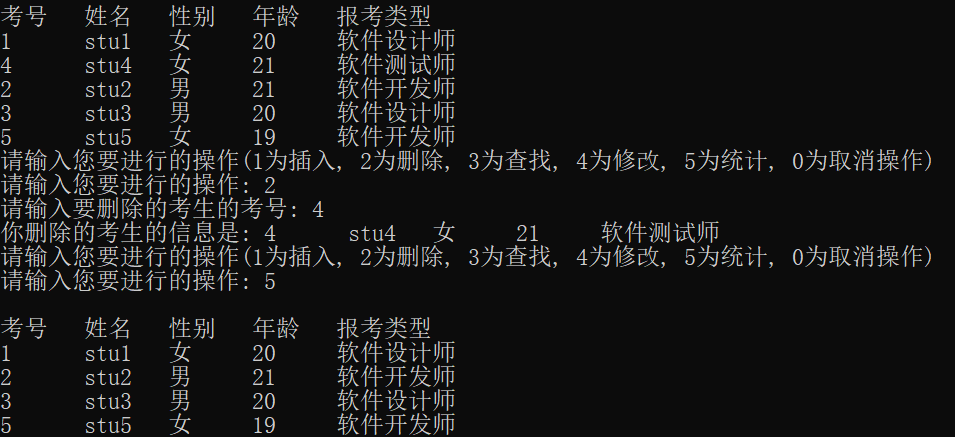
1 stu1 女 20 软件设计师

2 stu2 男 21 软件开发师

3 stu3 男 20 软件设计师

5 stu5 女 19 软件开发师

**实验结果：**



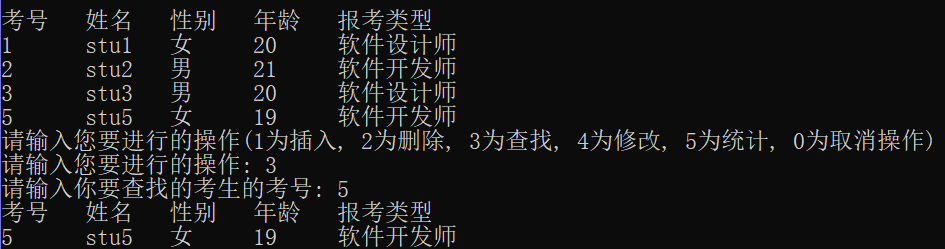
### 4.1.3 查找功能测试

**测试用例：**查找考号为5的考生

**预期结果：**

5 stu5 女 19 软件开发师

**实验结果：**



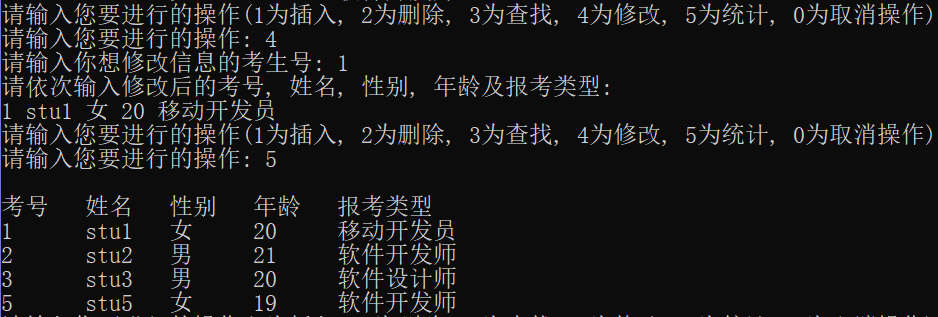
### 4.1.4 修改功能测试

**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：**

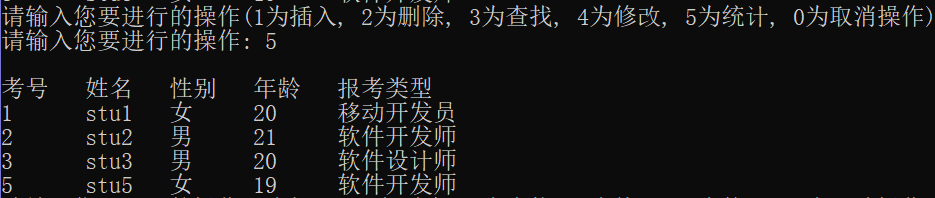
1 stu1 女 20 移动开发员

2 stu2 男 21 软件开发师

3 stu3 男 20 软件设计师

5 stu5 女 19 软件开发师

**实验结果：**



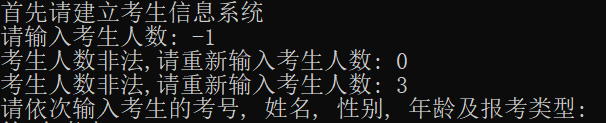
## 4.2 边界测试

### 4.2.1 初始化无输入数据

**测试用例：**考生人数分别输入-1, 0, 3

**预期结果：**给出错误提示，程序运行正常不崩溃，并提示用户重新输入。

**实验结果：**

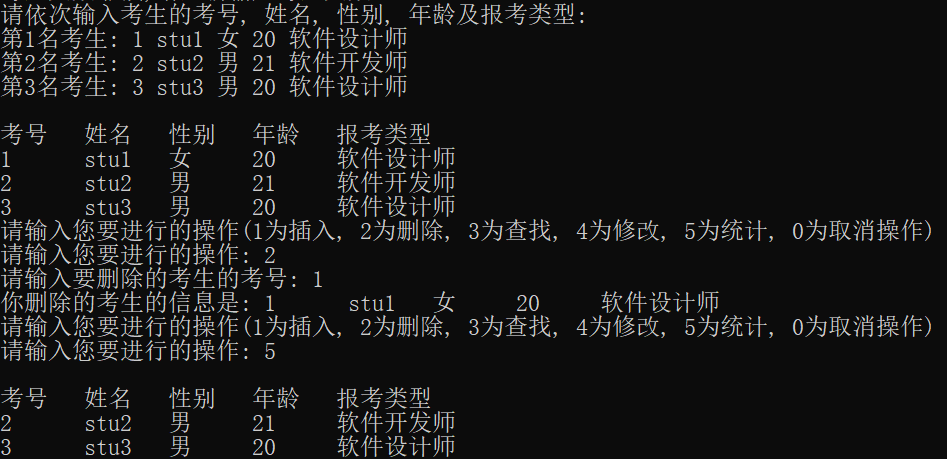


### 4.2.2 删除头结点

**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：**

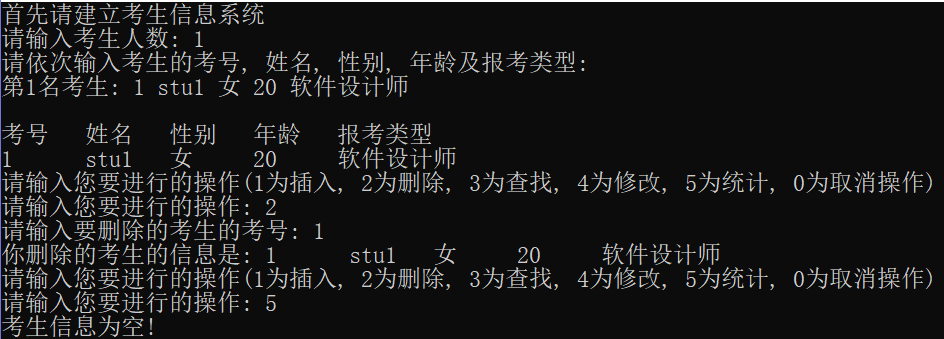


### 4.2.3 删除后链表为空

**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：**



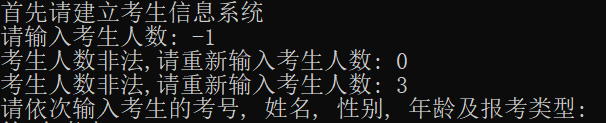
## 4.3 出错测试

### 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，并让用户继续输入。

**实验结果：**

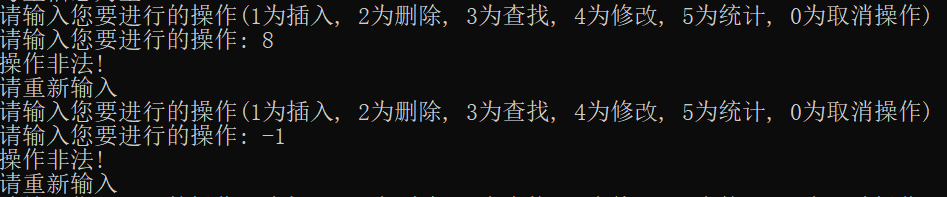


### 4.3.2 操作码错误

**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，并允许用户重新输入。

**实验结果：**

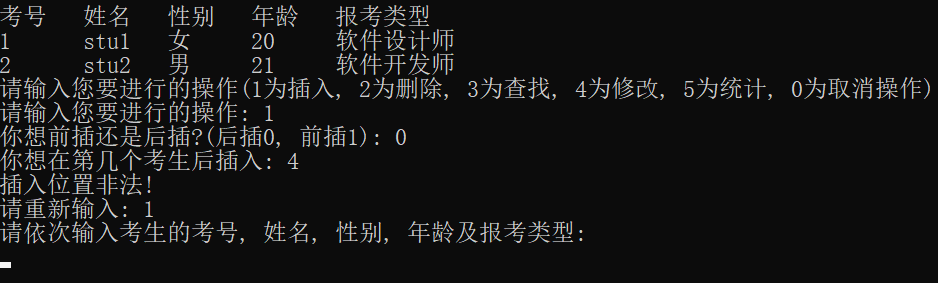


### 4.3.3 插入位置不存在

**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，并允许用户重新输入。

**实验结果：**

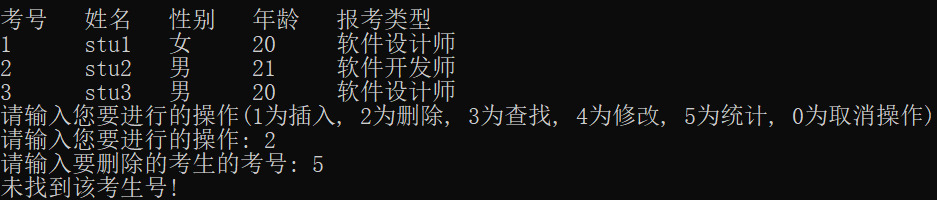


### 4.3.4 删除考号不存在

**测试用例：**要删除的考号不存在

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

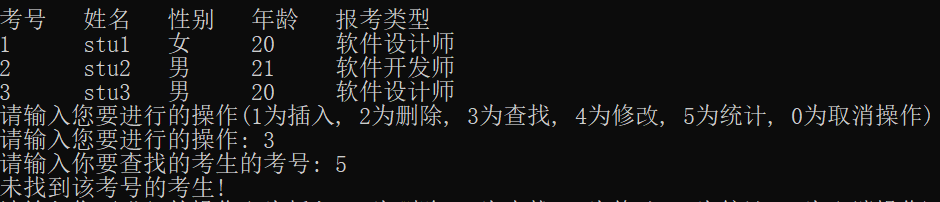


### 4.3.5 查找考号不存在

**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：**

