项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 何慧琳

学 号： 2152343

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。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，简化操作。

## 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（listnode）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，本系统采用继承方式，将linklist作为listnode的友元，使链表类可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类（listnode）**

**私有成员：**

char \*num; //表示学生考号

char \*name;//表示学生姓名

char \*sex;//表示学生性别

char \*age;//表示学生年龄

char \*work//表示学生报考种类

listnode \*link;//链表结点的指针域

**公有操作：**

listnode() :link(NULL) {};//默认构造函数

listnode(char\* nu, char\* na, char\* se, char \*ag, char\* wo, listnode\* li = NULL) :num(nu), name(na), sex(se), age(ag), work(wo), link(li) {};//带一个默认参数link=NULL的构造函数

listnode(const listnode& ln);//复制构造构造函数

**链表类（linklist）**

**私有成员：**

listnode\* first, \* last;//表头指针，表尾指针

**公有操作：**

linklist() { last = first = new listnode(); };//链表构造函数（创建链表对象时自动带表头结点）

~linklist();//析构函数，通过调用MakeEmpty()实现对内存的回收

void add(listnode l, int i = NULL);//将l项插入位置i，若未向i传递参数则在末尾添加l结点

int del(const char\* n);//将考号为n的考生信息删除

listnode\* find(const char\* n);//查找考号为n的考生信息，找到该结点则返回该结点前一位结点的地址，未找到则返回NULL

int modify(const char\* n, listnode newnode);//将考号为n的考生信息改为newnode

int sum();//统计链表中考生的总人数并打印链表

void MakeEmpty();//删链表置空

int print(char \*nu = NULL);//打印链表内容，，若向\*nu传递了参数则只打印学号为\*nu的考生信息

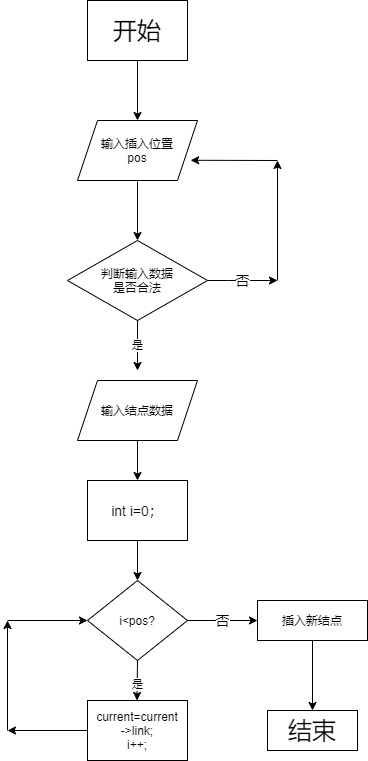
## 2.4 系统设计

系统首先调用main()函数实现对屏幕的初始化，完成对链表ll的创建和输入数据工作，然后根据用户所输入的操作码（program）执行链表ll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

listnode\* current = first;//检测指针current指示表头结点

int i = 0;

while (i < pos-1 && current ->link != NULL) {//将p指针移至插入位置前一位或链表末尾

i++;

current = current ->link;

}

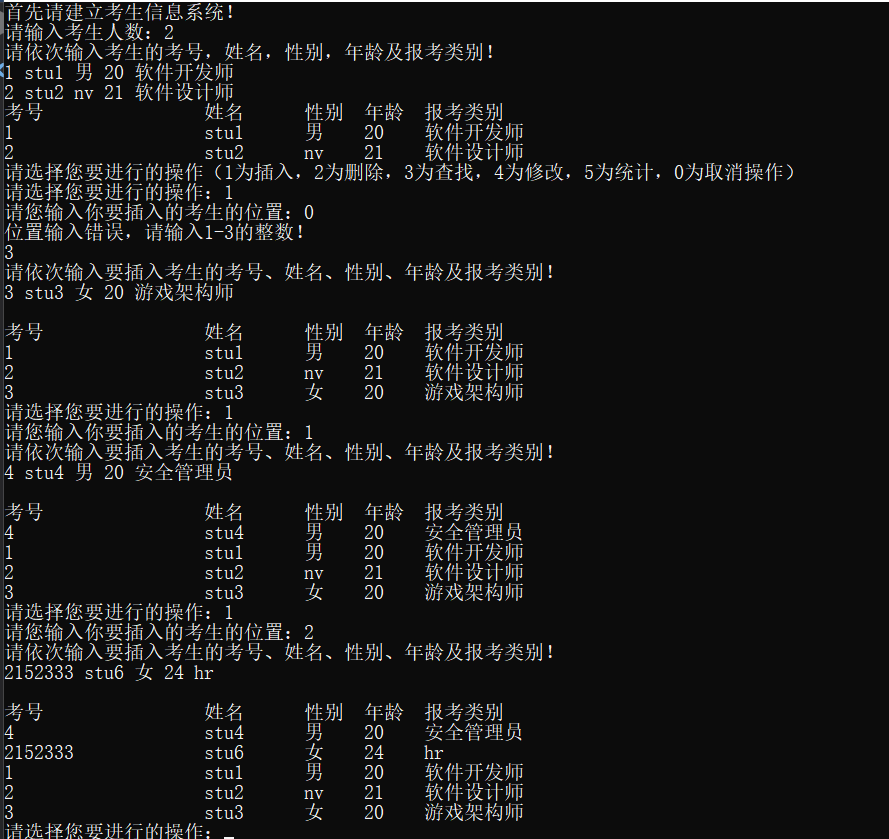
listnode \*n=new listnode(l);//创建新结点

n->link = current ->link;

current ->link = n;//实现插入新结点

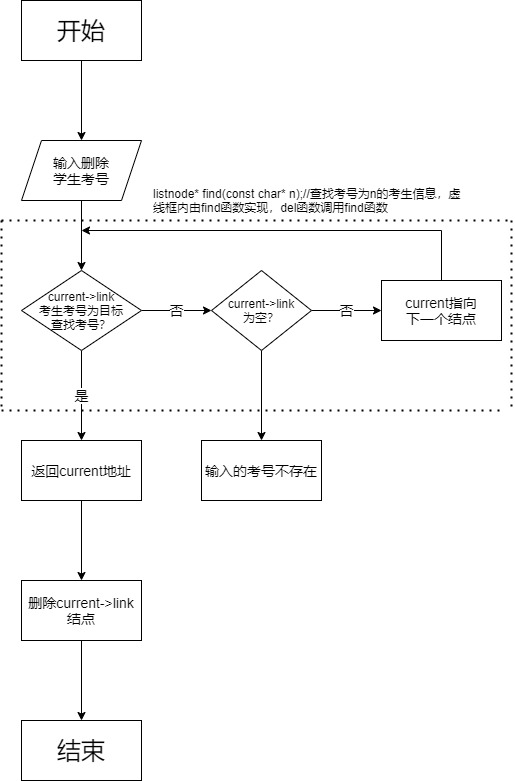
}

### 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

int linklist::del(const char\* n)

{//删除考号为n的考生信息

listnode\* current = find(n);//current指向目标结点前一个结点

if (current == NULL) {//p指针为NULL，说明查找对象不存在

cout << "该考生不存在！" << endl;

return 0;

}

listnode\* q = current->link;//q指针指向待删除结点

current->link = q->link;

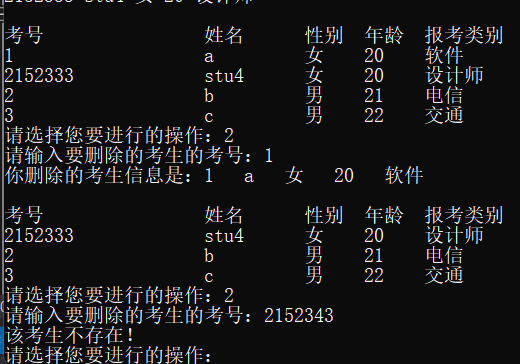
cout << "你删除的考生信息是：" << q->num << " " << q->name << " " << q->sex << " " << q->age << " " << q->work << endl;

delete q;//删除目标节点

return 1;

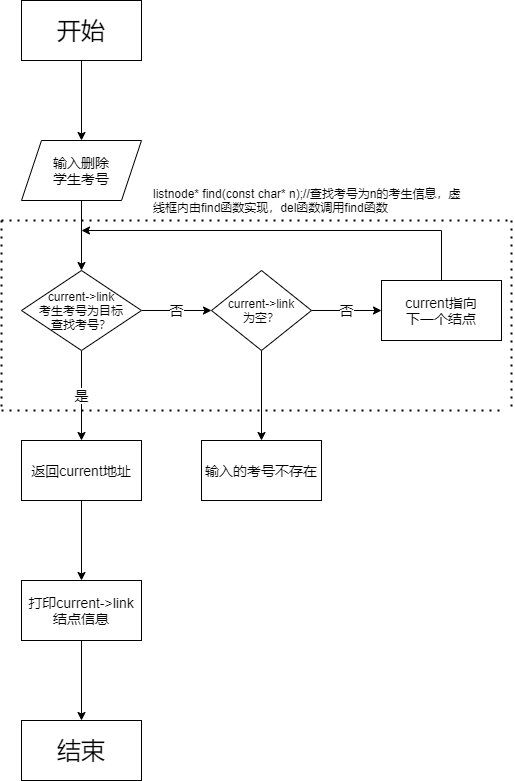
}

### 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

listnode\* linklist::find(const char\* n)

{//在链表中从头搜索学号num为n的结点。若存在，返回该结点前一个结点地址；若不存在，返回NULL

listnode \* current = first;//检测指针current指项表头结点

while (current != NULL) {

if (current->link!=NULL&&strcmp(current->link->num, n) == 0)

return current;//current的下一个结点为目标结点时返回p；

current = current->link;

}

return current;

}

int linklist::print(char \*nu)//无参传递则打印整个数组，有参则打印考号为nu的考生信息

{

……

listnode\* current;

if (nu != NULL) {//打印考号为nu的考生信息

current = find(nu);

if(current ==NULL)//该考号不存在

return 0;

else {

current = current->link;

cout << "考号 姓名 性别 年龄 报考类别" << endl;

printf("%-20s%-10s%-6s%-6s%s\n", current->num, current->name, current->sex, current->age, current->work);

return 1;

}

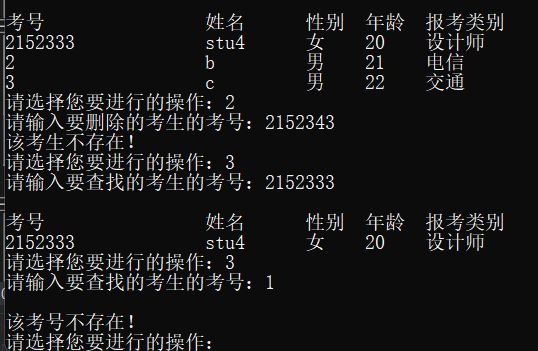
}

……

return 1;

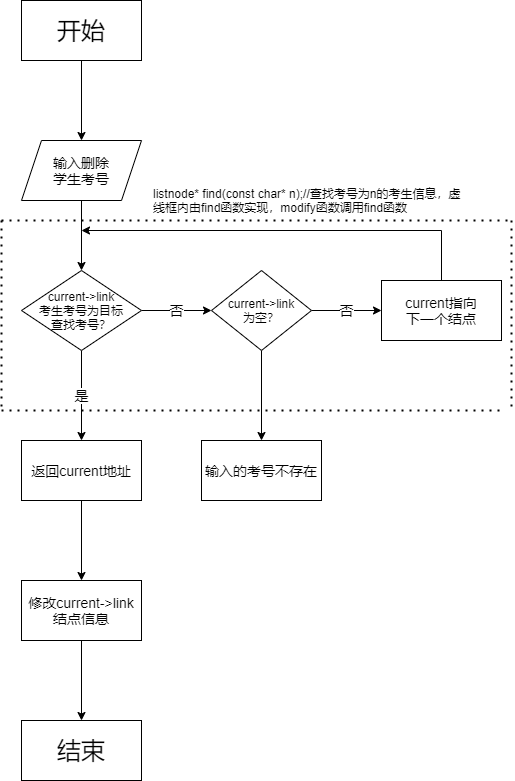
}

### 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

int linklist::modify(const char\* n, listnode newnode)

{//将考号为n的考生信息改为newnode中的信息

listnode\* current = find(n);//current指向目标结点前一个结点

if (current == NULL) {//current指针为NULL，说明查找对象不存在

cout << "该考生不存在！" << endl;

return 0;

}

current = current->link;//current指向待修改结点

strcpy(current->num,newnode.num);//修改current指向结点的内容

strcpy(current->name, newnode.name);

strcpy(current->sex, newnode.sex);

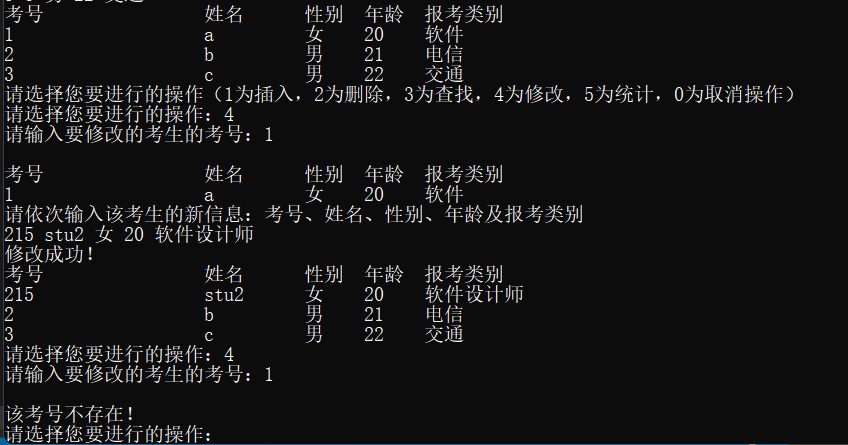
strcpy(current->age, newnode.age);

strcpy(current->work, newnode.work);

return 1;

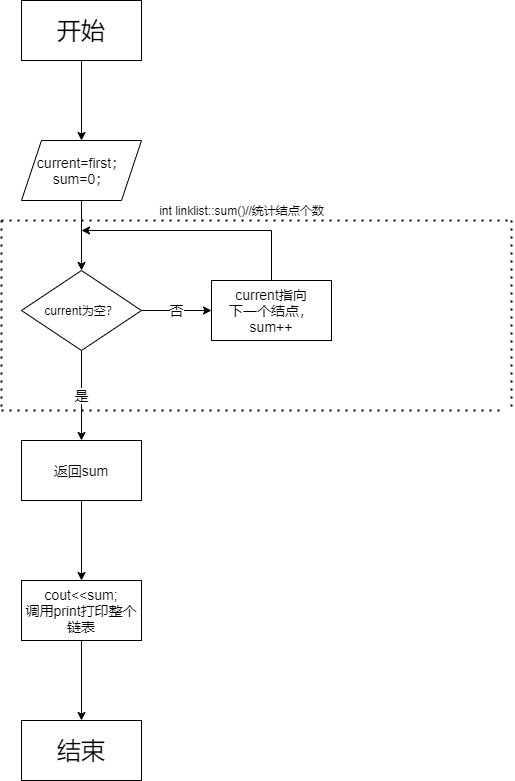
}

### 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

int linklist::sum()

{

int sum = 0;

listnode\* current = first;

while (current->link != NULL) {

current = current->link;

sum++;

}

return sum;

}

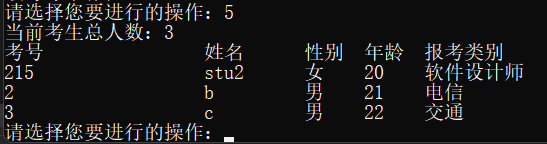
case 5:

cout <<"当前考生总人数："<< ll.sum() << endl;//输出统计的考生人数

ll.print();//打印当前全部考生信息

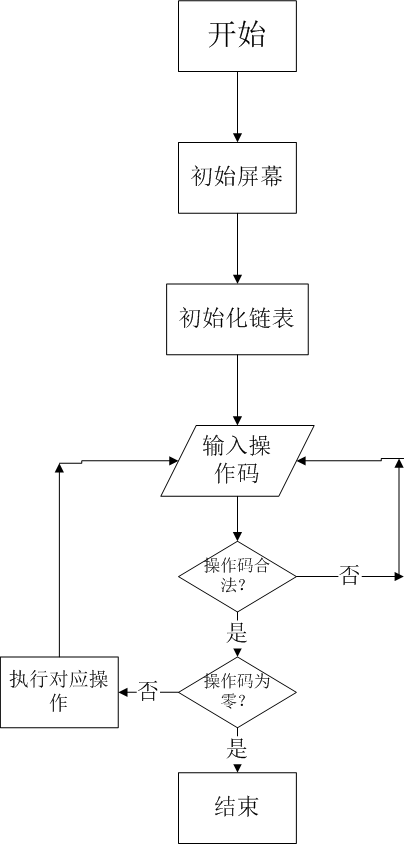
break;

### 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 (1) {

cout << "请选择您要进行的操作：";

cin >> program;

switch (program) {

case 1:

int pos;//要插入的位置

cout << "请您输入你要插入的考生的位置：";

cin >> pos;

{int length = ll.sum();

while (pos<1 || pos>length + 1) {

cout << "位置输入错误，请输入1-" << length + 1 << "的整数！\n";

cin >> pos;

}

}

cout << "请依次输入要插入考生的考号、姓名、性别、年龄及报考类别！\n";

cin >> num >> name >> sex >> age >> work;

{listnode newnode(num, name, sex, age, work);//创建一个以刚输入的信息为内容的新结点

ll.add(newnode, pos); }//插入新结点

cout << endl;

ll.print();//打印链表

break;

case 2:

cout << "请输入要删除的考生的考号：";

cin >> num;

if(ll.del(num)) {//del函数返回不为0，说明删除成功，打印链表

cout << endl;

ll.print();//打印链表

}

break;

case 3:

cout << "请输入要查找的考生的考号：";

cin >> num;

cout << endl;

if (!ll.print(num))

cout << "该考号不存在！" << endl;

break;

case 4:

cout << "请输入要修改的考生的考号：";

cin >> num;

cout << endl;

if (!ll.print(num))

cout << "该考号不存在！" << endl;

else {

char\* newnum=new char[20];

cout << "请依次输入该考生的新信息：考号、姓名、性别、年龄及报考类别\n";

cin >> newnum >> name >> sex >> age >> work;

listnode newnode(newnum, name, sex, age, work);//创建以新输入的信息为内容的结点

ll.modify(num, newnode);//在链表中修改原考号为num的结点信息为newnode

cout << "修改成功！\n";

ll.print();

}

break;

case 5:

cout <<"当前考生总人数："<< ll.sum() << endl;//输出统计的考生人数

ll.print();//打印当前全部考生信息

break;

case 0:

return 0;

break;

default:

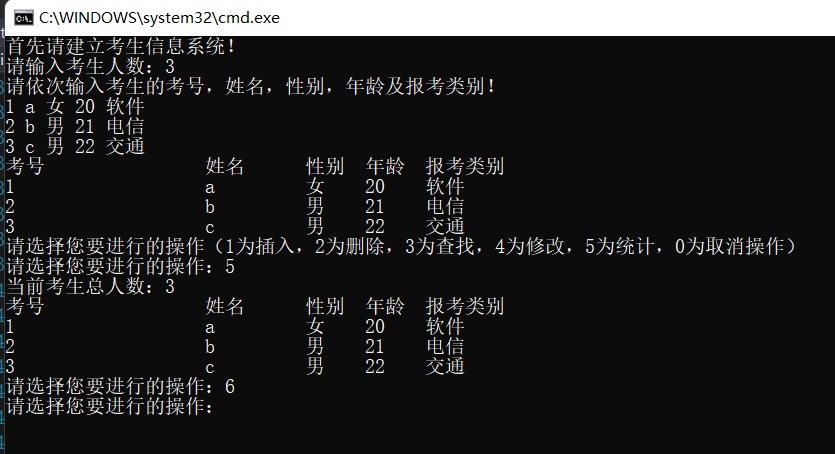
cin.clear();

cin.ignore(100, '\n');

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**：插入位置：1；插入信息：4 stu4 男 20 安全管理员

**预期结果**：

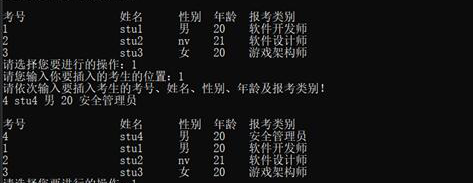
4 stu4 男 20 安全管理员

1 stu1 男 20 软件开发师

2 stu2 女 21 软件设计师

3 stu3 女 20 游戏架构师

**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

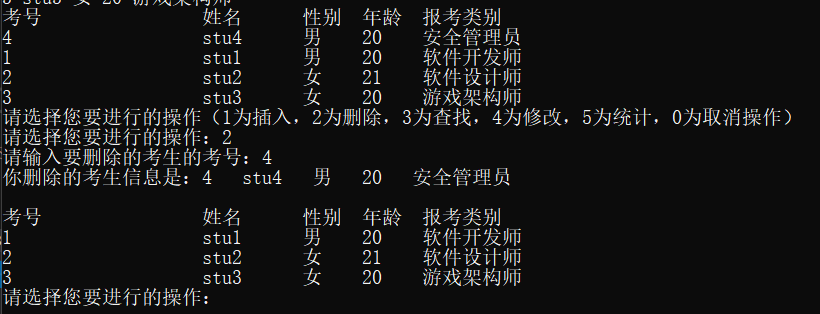
**预期结果：**

1 stu1 男 20 软件开发师

2 stu2 女 21 软件设计师

3 stu3 女 20 游戏架构师

**实验结果：**



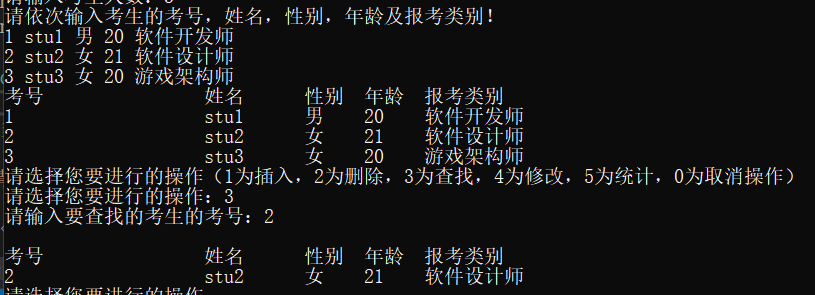
### 4.1.3 查找功能测试

**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：**



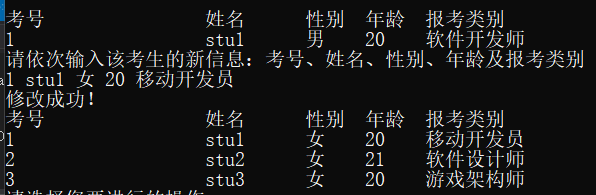
### 4.1.4 修改功能测试

**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：**

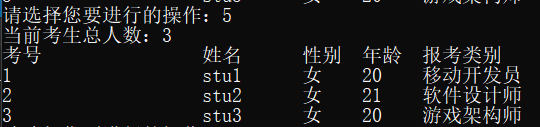
当前考生总人数：3

1 stu1 女 20 移动开发员

2 stu2 女 21 软件设计师

3 stu3 女 20 游戏架构师

**实验结果：**



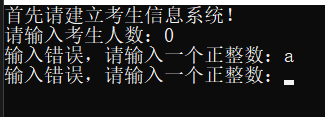
## 4.2 边界测试

### 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：**

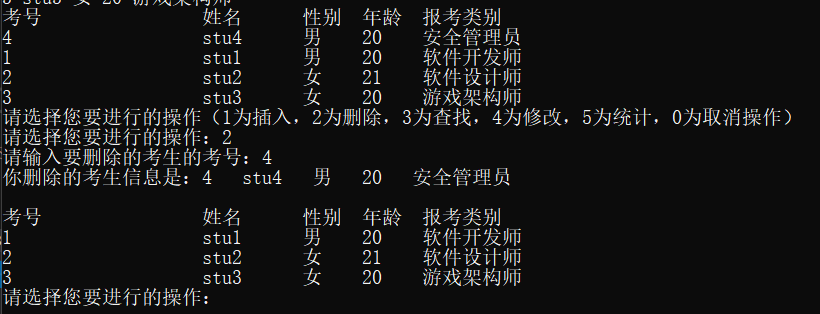


### 4.2.2 删除头结点

**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：**

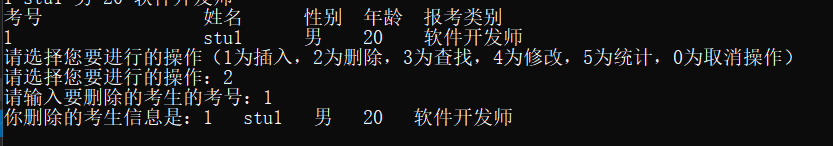


### 4.2.3 删除后链表为空

**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：**



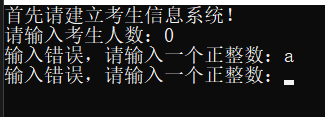
## 4.3 出错测试

### 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：**

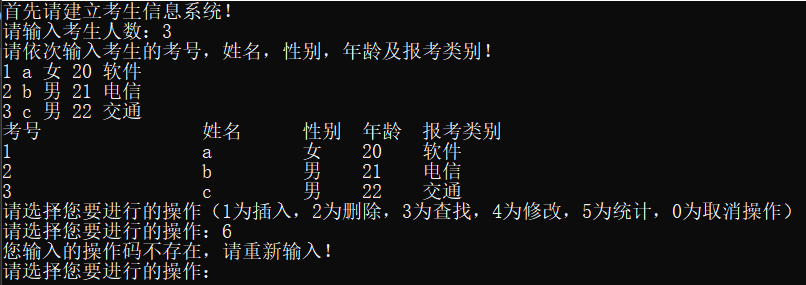


### 4.3.2 操作码错误

**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：**

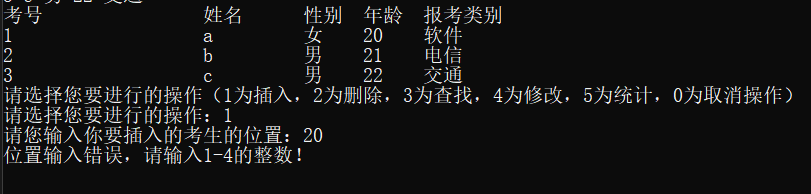
****

### 4.3.3 插入位置不存在

**测试用例：**链表里有3条记录，向链表的第20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：**

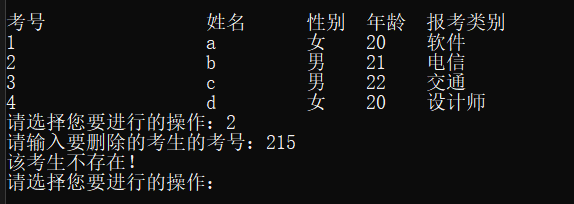


### 4.3.4 删除考号不存在

**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。

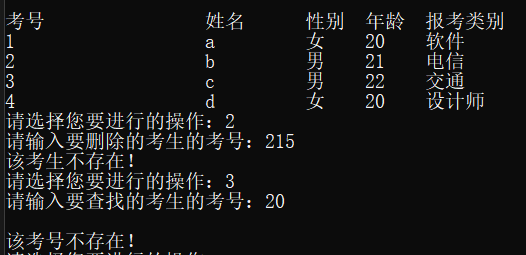
**实验结果：**

****

### 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。

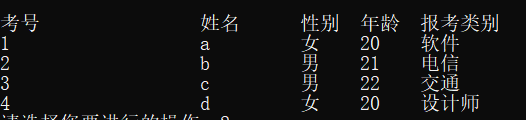
**实验结果：**

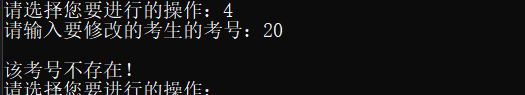
### 4.3.6 修改考号不存在

**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：**





# 