项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 杨宇琨

学 号： 2252843

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 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，使得程序简洁。

## 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。为方便处理，本系统采用struct描述链表结点类（student），这样使得链表结点类（student\_List）可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类（**student**）**

**公有成员：**

struct student

{

int num;//The number of the student

int age;//The age of the student

char name[max\_input\_name+1];//The name of the student

char catagory[11];//The catagory the student applies for

char gender[3];//The gender of the student

student\* next;//Pointer points to the next node

};

**链表类（**student\_List**）**

**私有成员：**

student\* head;//The head pointer

**公有操作：**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:student\_List

Function:To create a new space for the head node

Input Parameters:void

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

student\_List()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:List\_insert

Function:To insert a new element into our linked list

Input Parameters:void

Return Value:Bool whether the operation is successful,if it is return 1

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool List\_insert()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:List\_delete

Function:To delete a new element from our linked list

Input Parameters:void

Return Value:Bool whether the operation is successful,if it is return 1

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool List\_delete()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:List\_edit

Function:To edit a new element in our linked list

Input Parameters:void

Return Value:Bool whether the operation is successful,if it is return 1

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool List\_edit()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:List\_search

Function:To search a element in our linked list

Input Parameters:void

Return Value:Bool whether the operation is successful,if it is return 1

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool List\_search()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:calculateSum

Function:To calculate the amount of student in our linked list

Input Parameters:void

Return Value:Bool whether the operation is successful,if it is return 1

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool calculateSum()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:List\_display

Function:To display our linked list

Input Parameters:void

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void List\_display()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:establish

Function:To create our linked list

Input Parameters:void

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void establish()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:~student\_List

Function:To delete all the space we created in this class

Input Parameters:void

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

~student\_List()

## 2.4 系统设计

系统首先调用List.establish()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 插入功能核心代码

//Let the current points to the element we want,always is the one that is previous to the one we are operating on

student\* current = head;

for (int i = 0; i < num - 1; i++,current=current->next) {

if (current->next == NULL) {

//If the current is NULL ,it means that we have gone too far

std::cout << "当前位置无法插入元素" << std::endl;

std::cout << "本次操作结束，按回车键进行新一轮操作...";

pause();

return 0;

}

}

student\* new\_current = new student;

if (new\_current == NULL) {

std::cerr << "内存申请失败" << std::endl;

std::cout << "本次操作结束，按回车键进行新一轮操作...";

pause();

return 0;

}

//Input the new data

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

Input(new\_current);

//Insert

student \* temp;

if (current->next == NULL) {

current->next = new\_current;

new\_current->next = NULL;

}

else {

temp = current->next;

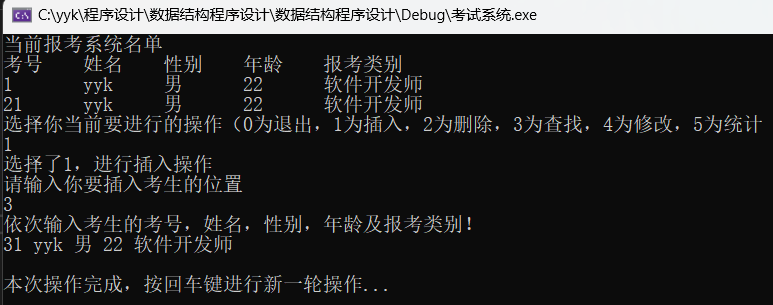
current->next = new\_current;

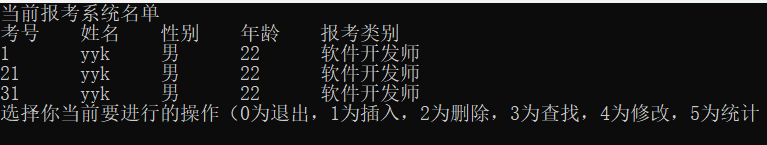
new\_current->next = temp;

}

### 

### 3.1.2 插入功能截屏示例





### 3.1.3 插入功能实现流程

用户输入插入位置——>输入位置合法（若不合法则回到上一步）——>指针移动到相应位置（到空指针强制停止移动并退出）——>输入相应的数据——>插入

## 3.2 删除功能的实现

### 3.2.1 删除功能核心代码

//Let the current points to the element we want,always is the one that is previous to the one we are operating on

for (student\* current = head; current->next!= NULL; current = current->next) {

//If found,display the information,and exit the function

if (current->next->num == num) {

//Display the information we are deleting

std::cout << "删除以下学生信息！" << std::endl;

std::cout << std::left << std::setw(8) << "考号" << std::setw(8) << "姓名" << std::setw(8) << "性别" << std::setw(8) << "年龄" << std::setw(8) << "报考类别" << std::endl;

std::cout << std::left << std::setw(8) << current->next->num;

std::cout << std::left << std::setw(8) << current->next->name;

std::cout << std::left << std::setw(8) << current->next->gender;

std::cout << std::left << std::setw(8) << current->next->age;

std::cout << std::left << std::setw(8) << current->next->catagory;

std::cout << std::endl;

//delete

student\* temp = current->next->next;

delete current->next;

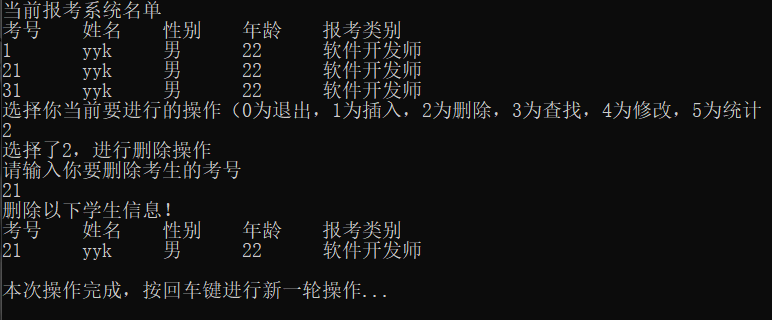
current->next = temp;

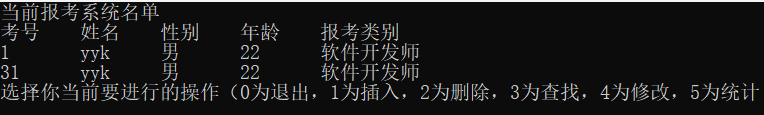
return 1;

}

}

### 3.2.2 删除功能截屏示例





### 3.2.3 删除功能实现流程

用户输入删除考号——>输入合法（若不合法则回到上一步）——>循环查找考号（找不到输出提示并退出）——>输出查找到考生的信息——>删除

## 3.3 查找功能的实现

### 3.3.1 查找功能核心代码

//Let the current points to the element we want,always is the one that is previous to the one we are operating on

for (student\* current = head; current!=NULL;current = current->next) {

//If found,display the information,and exit the function

if (current->num == num) {

std::cout << "查找到该学生信息！" << std::endl;

std::cout << std::left << std::setw(8) << "考号" << std::setw(8) << "姓名" << std::setw(8) << "性别" << std::setw(8) << "年龄" << std::setw(8) << "报考类别" << std::endl;

std::cout << std::left << std::setw(8) << current->num;

std::cout << std::left << std::setw(8) << current->name;

std::cout << std::left << std::setw(8) << current->gender;

std::cout << std::left << std::setw(8) << current->age;

std::cout << std::left << std::setw(8) << current->catagory;

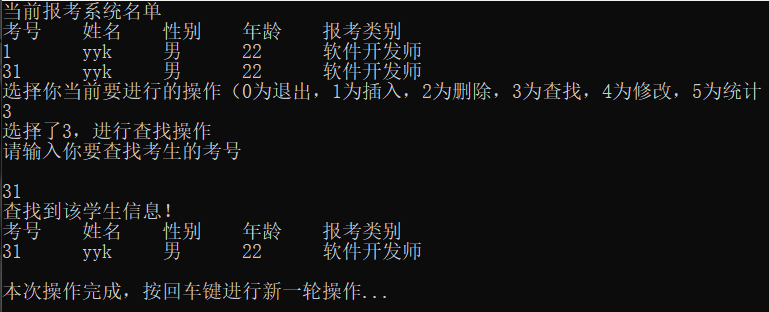
std::cout << std::endl;

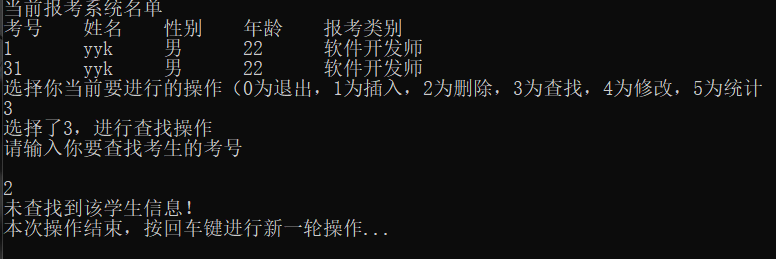
return 1;

}

}

### 3.3.2 查找功能截图示例





### 3.3.3 查找功能实现流程

用户输入查找考号——>输入合法（若不合法则回到上一步）——>循环查找考号（找不到输出提示并退出）——>输出查找到考生的信息

## 3.4 修改功能的实现

### 3.4.1 修改功能核心代码

//Let the current points to the element we want,always is the one that is previous to the one we are operating on

student\* current = head;

for (int i = 0; i < num - 1; i++, current = current->next) {

if (current->next == NULL) {

//If the current is NULL ,it means that we have gone too far

std::cout << "当前位置无元素" << std::endl;

std::cout << "本次操作结束，按回车键进行新一轮操作...";

pause();

return 0;

}

}

//Edit

if (current->next == NULL) {

//If the current-> is NULL ,it means that we reached the last node and there is no node after it

std::cout << "当前位置无元素" << std::endl;

std::cout << "本次操作结束，按回车键进行新一轮操作...";

pause();

return 0;

}

else {

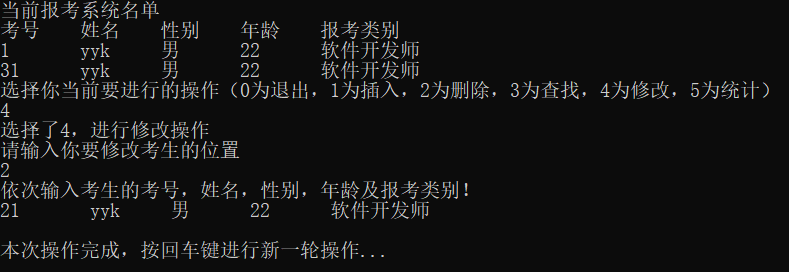
std::cout << "依次输入考生的考号，姓名，性别，年龄及报考类别！" << std::endl;

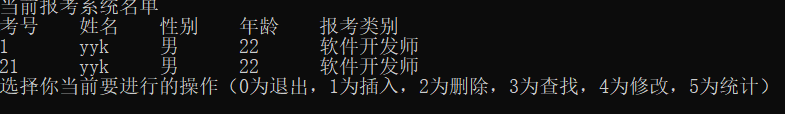
Input(current->next);

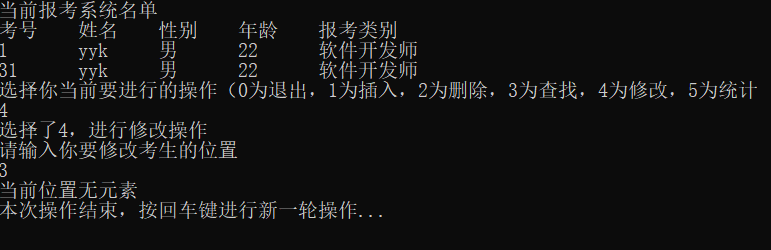
return 1;

}

### 3.4.2 修改功能截屏示例







### 3.4.3 修改功能实现流程

用户输入修改位置——>输入合法（若不合法则回到上一步）——>指针移动到相应位置（到空指针强制停止移动并退出）——>输入相应的数据——>插入

## 3.5 统计功能的实现

### 3.5.1 统计功能核心代码

if (head->next) {

int count\_male = 0, count\_female = 0, total = 0;//Create variables to store the amount

//Count

for (student\* current = head; current != NULL; current = current->next) {

if (std::strcmp(current->gender, "男") == 0) {

count\_male++;

}

else

count\_female++;

}

//Sum

total = count\_male + count\_female;

//Display the outcome

std::cout << "统计结果为：一共有" << total << "人，其中男生有" << count\_male << "人，女生有" << count\_female << "人" << std::endl;

return 1;

}

else {

std::cout<<"该表为空表无法统计！" << std::endl;

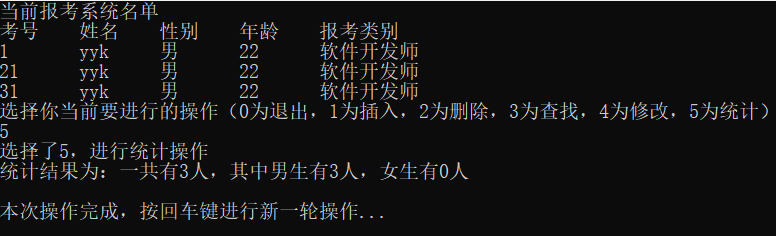
std::cout << "本次操作结束，按回车键进行新一轮操作...";

pause();

return 0;

}

### 3.5.2 统计功能截屏示例

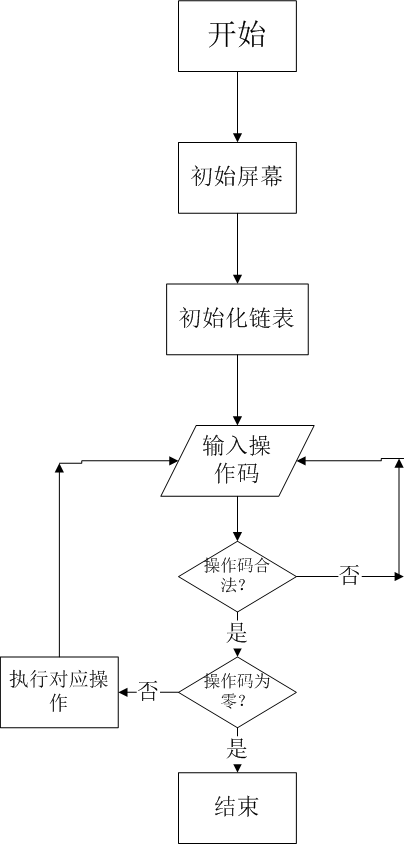


### 3.5.3 修改功能实现流程

循环遍历链表所有元素，遇到男性male++，遇到女性female++，最后total=male+female.输出结果

## 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

//Display the prompt

std::cout << "选择你当前要进行的操作（0为退出，1为插入，2为删除，3为查找，4为修改，5为统计）" << std::endl;

char controller;//Create a variable to hold the choice we input

while (true) {

controller = \_getch();//\_getch can avoid displaying undesired input on the screen

if (controller >= '0' && controller <= '5') {

std::cout << controller << std::endl;//Display the choice on the screen

switch (controller)

{

case '0':

std::cout << "程序结束！" << std::endl;

return 0;

case '1':

std::cout << "选择了1，进行插入操作" << std::endl;

if (List.List\_insert()) {

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

}

break;

case '2':

std::cout << "选择了2，进行删除操作" << std::endl;

if (List.List\_delete()) {

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

}

break;

case '3':

std::cout << "选择了3，进行查找操作" << std::endl;

if (List.List\_search()) {

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

}

break;

case '4':

std::cout << "选择了4，进行修改操作" << std::endl;

if (List.List\_edit()) {

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

}

break;

case '5':

std::cout << "选择了5，进行统计操作" << std::endl;

if (List.calculateSum()){

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

}

break;

default:

continue;

}

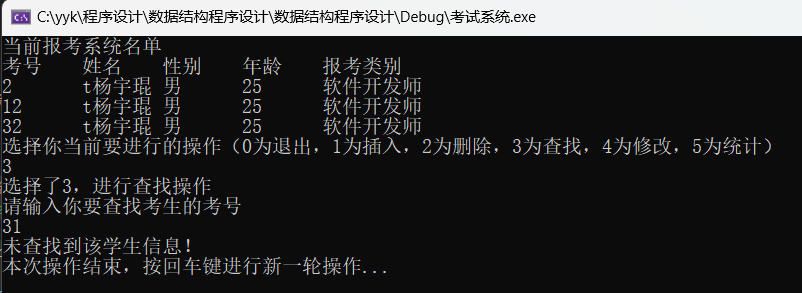
break;

}//end of if

}//end of while

}//end of while

### 3.6.3 总体系统截屏示例



# 当选择非0到5时，程序在用户界面无任何反应，直到输入正确的选择，程序会给出反显及提示

# 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 软件设计师

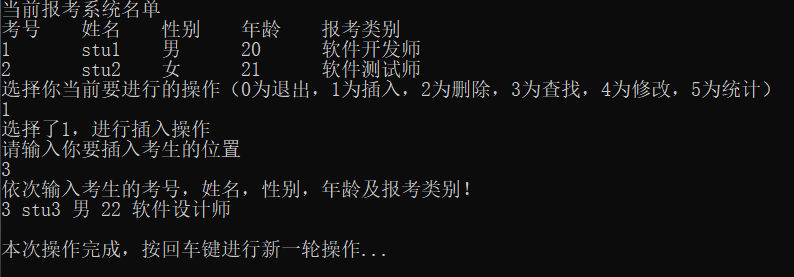
**预期结果**：

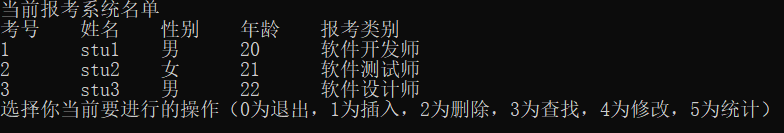
1 stu1 男 20 软件开发师

2 stu2 女 21 软件测试师

3 stu3 男 22 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

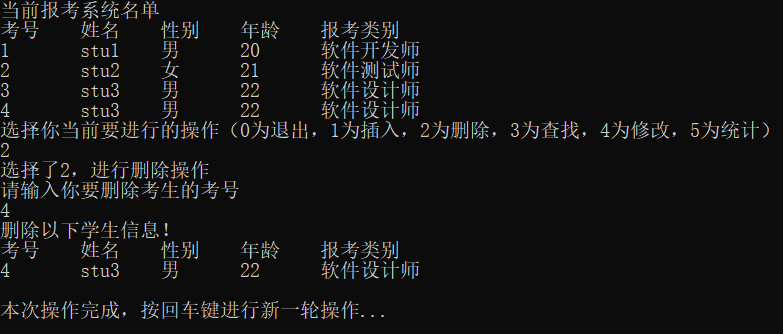
**预期结果：**

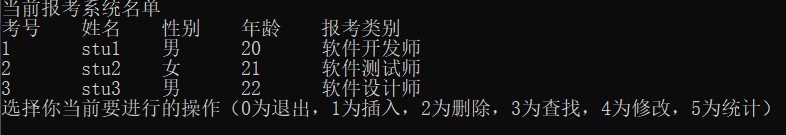
1 stu1 男 20 软件开发师

2 stu2 女 21 软件测试师

3 stu3 男 22 软件设计师

**实验结果：**

****



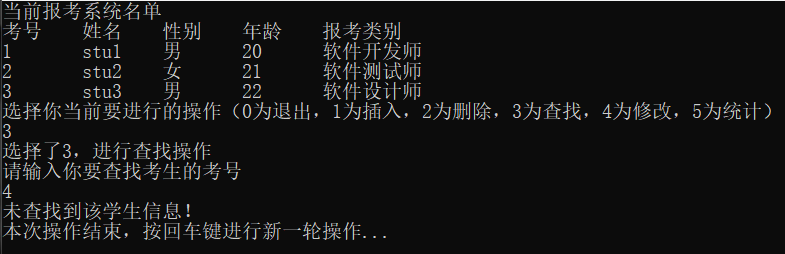
### 4.1.3 查找功能测试

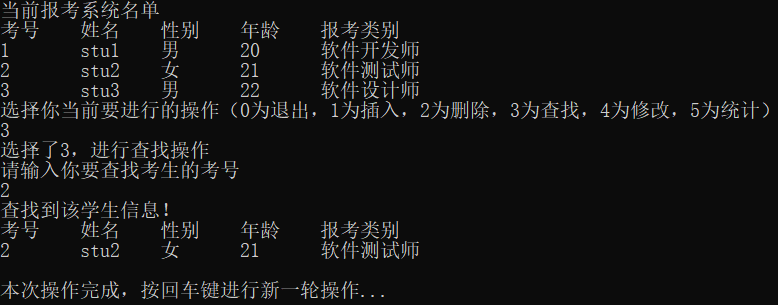
**测试用例：**查找考号为2的考生

**预期结果：**

2 stu2 女 21 软件测试师

**实验结果：**





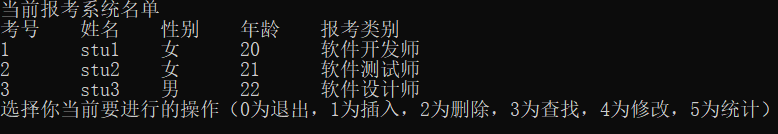
### 4.1.4 修改功能测试

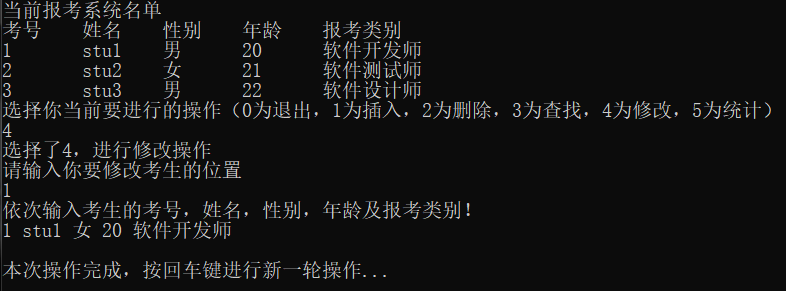
**测试用例：**将考号1修改为性别女，年龄20，报考种类软件开发师。

**预期结果：**

1 stu1 女 20 软件开发师

**实验结果：**

****

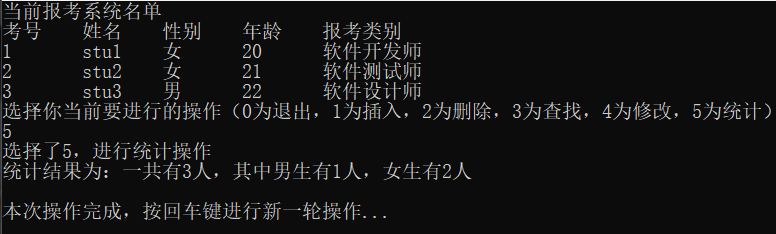
****

### 4.1.5 统计功能测试

**测试用例：**统计当前人数

**预期结果：**统计结果为：一共有3人，其中男生有1人，女生有2人

**实验结果：**



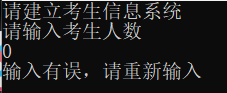
## 4.2 边界测试

### 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：**

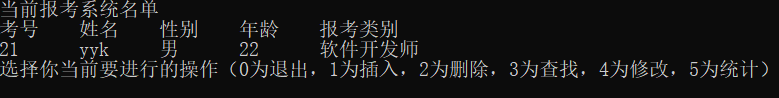


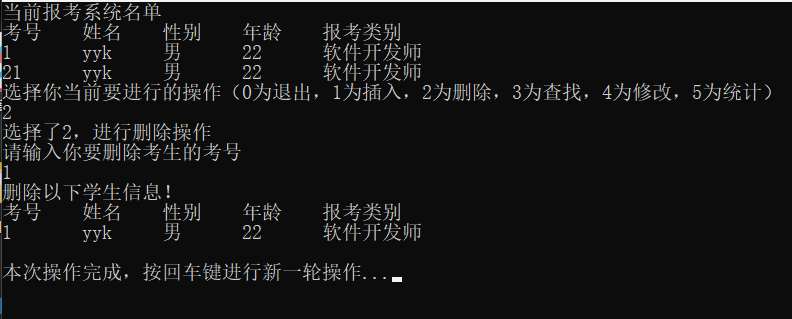
### 4.2.2 删除头结点

**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：**

****

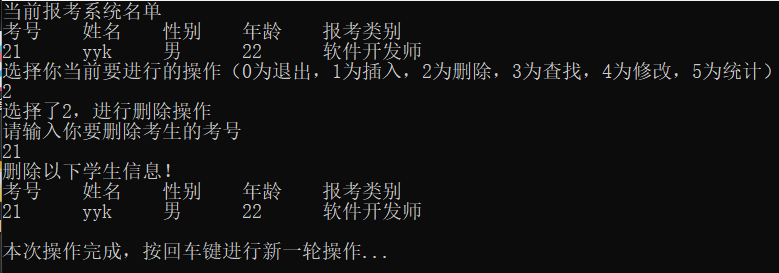


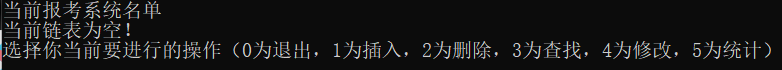
### 4.2.3 删除后链表为空

**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：**

****



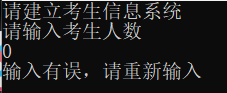
## 4.3 出错测试

### 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：**当选择非0到5时，程序在用户界面无任何反应，直到输入正确的选择，程序会给出反显及提示。

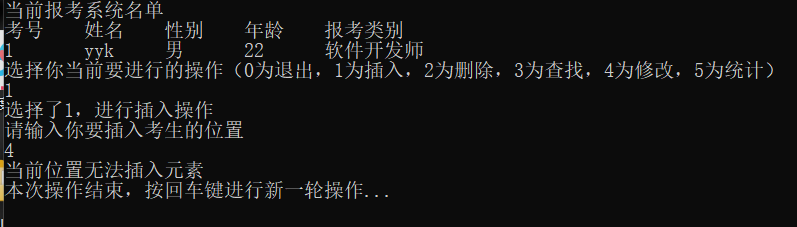
**实验结果：**无反应

### 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：**

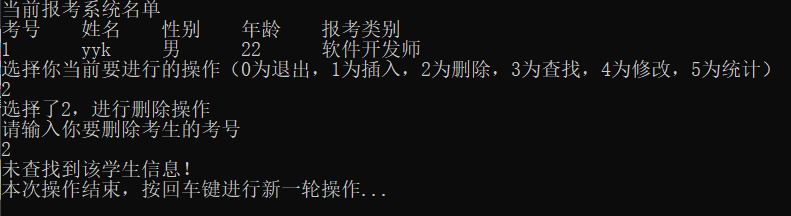


### 4.3.4 删除考号不存在

**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：**

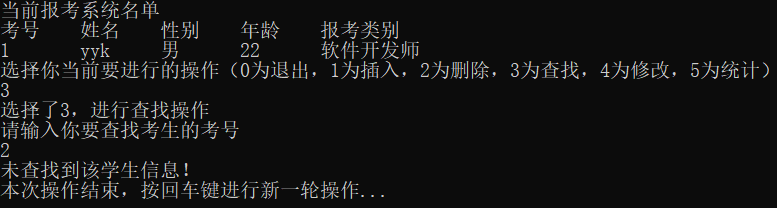
****

### 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：**

