**项目说明文档**

**数据结构课程设计**

**——考试报名系统**

作 者 姓 名 刘淑仪

学 号 2251730

指 导 教 师 张 颖

学 院 专 业 软件学院 软件工程



二〇二三 年 十 月 二十 日

目录

1 项目分析 1

1.1 项目背景分析 1

1.2 项目功能分析 1

2 项目设计 1

2.1 数据结构设计 1

2.2 类结构设计 2

2.3 成员与操作设计 2

2.4 系统设计 4

3 功能设计与项目实现 4

3.1 总体设计功能的实现 4

3.1.1 总体系统实现思路 4

3.1.2 总体系统核心代码 5

3.1.3 总体功能实现示例 7

3.2 插入功能的实现 7

3.2.1 插入功能实现思路 7

3.2.2 插入功能核心代码 8

3.2.3 插入功能实现示例 9

3.3 删除功能的实现 10

3.3.1 删除功能实现思路 10

3.3.2 删除功能核心代码 11

3.3.3 删除功能实现示例 11

3.4 查找功能的实现 12

3.4.1 查找功能实现思路 12

3.4.2 查找功能核心代码 13

3.4.3 查找功能实现示例 14

3.5 修改功能的实现 14

3.5.1 修改功能实现思路 14

3.5.2 修改功能核心代码 15

3.5.3 修改功能实现示例 16

3.6 统计功能的实现 16

3.6.1 统计考生总人数功能 16

3.6.1.1 统计考生总人数实现思路 16

3.6.1.2 统计计考生总人数实现核心代码 17

3.6.1.3 统计考生总人数实现示例 18

3.6.2 统计性别人数分布功能 19

3.6.2.1 统计性别人数分布实现思路 19

3.6.2.2 统计性别人数分布实现核心代码 20

3.6.2.3 统计性别人数分布实现示例 21

3.6.3 统计年龄人数分布功能 21

3.6.3.1 统计年龄人数分布实现思路 21

3.6.3.2 统计年龄人数分布实现核心代码 22

3.6.3.3 统计年龄人数分布实现示例 23

3.6.4 统计报考类别人数分布功能 24

3.6.4.1 统计报考类别人数实现思路 24

3.6.4.2 统计报考类别人数实现核心代码 24

3.6.4.3 统计报考类别人数实现示例 26

4 项目测试 26

4.1 功能测试 26

4.1.1 插入功能测试 26

4.1.1.1 在开头、中间、结尾位置插入 26

4.1.1.2 插入相同的考生信息 28

4.1.2 删除功能测试 28

4.1.3 查找功能测试 29

4.1.4 修改功能测试 30

4.1.4.1 在开头、中间、结尾位置修改 30

4.1.4.4 修改时输入已有的考生信息 31

4.1.4.4 修改时输入已经被修改过的考生的考号 31

4.1.5 统计功能测试 32

4.1.5.1 经过插入后统计是否有误差 32

4.1.5.2 经过删除后统计是否有误差 33

4.1.5.3 经过修改后统计是否有误差 34

4.2 错误测试 35

4.2.1 考生人数错误 35

4.2.1.1 考生人数输入判断思路 35

4.2.1.2 考生人数输入核心代码 36

4.2.1.3 考生人数输入示例 36

4.2.2 考生信息错误 38

4.2.2.1 考生信息输入判断思路 38

4.2.2.2 考生信息输入核心代码 38

4.2.2.3 考生信息输入示例 39

4.2.3 插入位置不存在 42

4.2.4 删除考号不存在 43

4.2.5 查找考号不存在 43

4.2.6 修改考号不存在 44

4.3 Linux环境测试 44

5 集成开发环境与编译运行环境 45

# 项目分析

## 项目背景分析

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。考试报名工作给各高校报名工作带来了新的挑战，给教务管理部门增加了很大的工作量。

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。

## 项目功能分析

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，本项目的实质是完成对考生信息的建立，查找，插入，修改，删除，统计等功能，且需要保证项目能够正常退出。其中考生信息包括准考证号，姓名，性别，年龄和报考类别等信息。项目在设计时应首先确定系统的数据结构，定义类的成员变量和成员函数；然后实现各成员函数以完成对数据操作的相应功能；最后完成主函数以验证各个成员函数的功能并得到运行结果。

# 项目设计

## 数据结构设计

本系统需要大量的插入、删除、修改操作，且考生容量本身巨大且报考人数不确定，因此考虑使用链表数据结构而非数组结构。链表作为主要数据结构在"考生报名系统"中可以提供更好的动态性、高效的插入和删除操作，以及更好的灵活性，适应系统中考生信息的动态变化，避免了整体拷贝、移动，大大降低了平均时间复杂度，同时避免了空间浪费或不足的问题。

同时，在建立链表时，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。

为了方便程序维护，首先定义了一个函数别名来方便获取函数状态，又定义了多个常量来处理各个数据的最大值最小值以及长度。

本系统需要存储大量的考生信息，故使用结构体studentInformation(SI)来存储每个考生的信息，同时建立链表ApplicationCategoryCount（ACC）来存储统计的考生信息，方便功能5的报考类别统计。

## 类结构设计

本系统用solution来存储整个解决方案。在解决方案里，私有成员(private)分别为存储考生信息的链表、学生信息、职业统计信息以及其他考号、序号、人数等信息防止泄露；公有成员(public)为实现本系统功能的各函数。

## 成员与操作设计

未注释的函数请见下文实现对应功能时代码的函数标签。

·存储考生具体信息 studentInformation结构体

typedef struct studentInformation {

int examNum;

char name[maxNameLength];

char sex[maxSexLength];

int age;

char applicationCategory[maxCategoryLength];

}SI;

·存储统计的报考类别信息 ApplicationCategoryCount结构体

typedef struct ApplicationCategoryCount {

char category[maxCategoryLength];

int count;

ApplicationCategoryCount\* next;

}ACC;

·存储考生信息 LNode链表结构体

typedef struct LNode {

SI data;//结点的数据域

struct LNode\* next;//结点的指针域

} \*LinkList;

·存储系统具体方案 solution类

class solution {

private:

LNode\* studentList;

SI getStudentData;

ACC\* categoryCounts = nullptr;

int personNum;//总人数

int opt;

int operationNum;//输入的考号

int locationNum;//输入的位置序号

int operationLocation;//输入的考号所对应的位置序号

int femaleNum = 0;//女性人数

int maleNum = 0;//男性人数

int searchExamNum = 0;//需要查找的学生的考号

bool NumorInformation;//查找元素时需要返回序号还是值

public:

// 初始化链表

Status InitList() {

studentList = new LNode;

studentList->next = NULL;

return OK;

}

~solution() {

DestroyList();

}

// 销毁单链表L

Status DestroyList();

// 判断链表是否为空

bool listEmpty();

// 输出考生信息

void showData(SI studentData, bool opt);

Status listInsert();

Status listDelete();

int getElem(bool InformationorNum);

Status listRevise();

void deletRepetitiveList();

int listLength();

void statsSexDistribution();

void statsAge();

void countApplicationCategory();

void statsInformation();

void createList();

void inputPersonNum();

bool inputPersonInformation(SI& studentData);

void inputOperationLocation(int maxNum);

void inputOperationNum();

bool option();

void output(bool opt);

};

## 系统设计

系统首先创建类解决方案ERS，初始化考生报名系统界面，再调用InitList来初始化链表，输入考生的总人数与考生的具体信息，并使用尾插法建立链表，再根据所输入的内容输出学生信息，根据提示进行进一步操作。

同时，由于本系统需要输入的数据较多、数据类型也较为复杂，故在需要输入操作选项和总人数等int型变量时，采用cin相关函数获取输入并进行错误处理；在输入一位考生的所有信息(SI)时，采用iss相关函数来获取整行输入并进行错误处理。当然，由于课设不允许使用STL容器的要求，sstring库里原本用string处理整行输入的方法此时并不太适用，遂即自我实现MyGetline函数，并用字符数组char[]来代替string存储并处理输入。

为了美观，本系统打表了一个表格来输出，且使用#include <iomanip>来进行对齐。整体实现效果见总体功能示例。

# 功能设计与项目实现

## 总体设计功能的实现

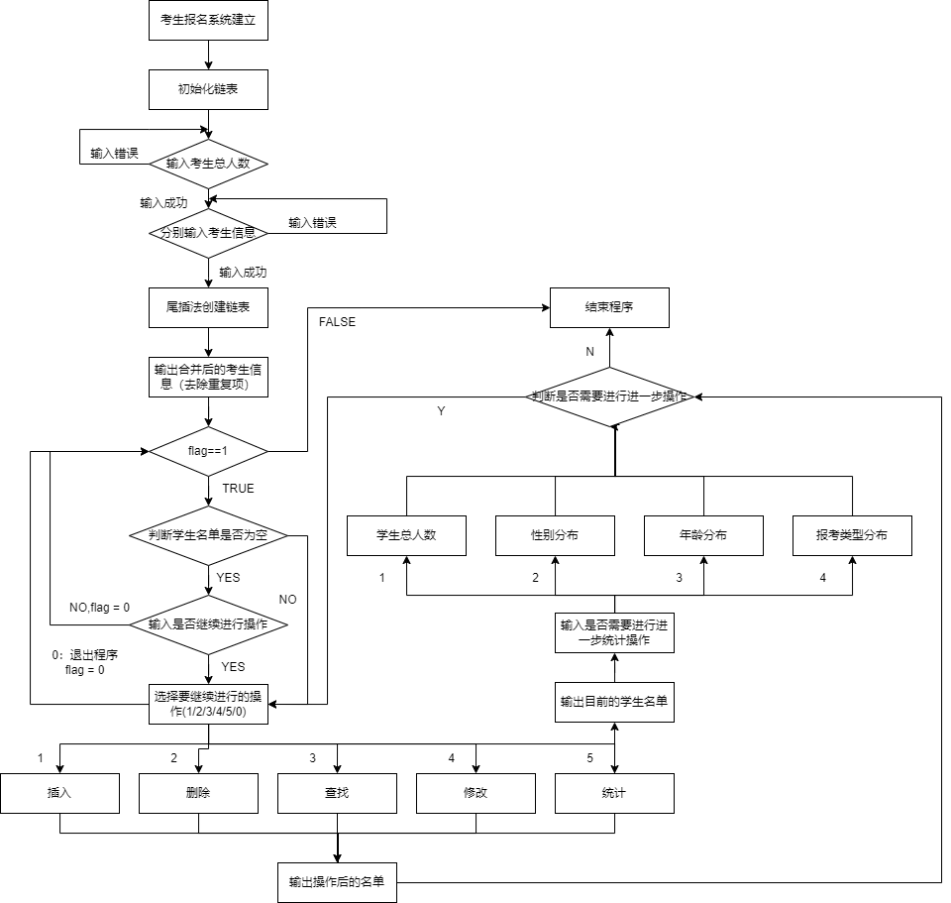
### 总体系统实现思路

建立考生报名系统，首先自我生成一个结构体链表存储考生信息。链表的优点已在上文中阐释。链表的结点的数据域为考生信息，考生信息由结构体SI存储。首先初始化链表，根据提示信息输入考生总人数，输入小数、字母、负数以及输入多个数字均为错误输入。读取到总人数(personNum)后，开始分别输入考生信息。由于考号、姓名、性别、年龄、报考信息分别为不同类型的数据，故采用getline获取整行输入，再通过sstream里的iss输入到结构体的各数据中。同时，也设置了错误处理：输入非法、姓名输入过长、年龄不在18~35岁之间、性别为男/女之外的输入都会被判为非法输入。

输入的信息本身将会被存储在p -> data中，由此用尾插法建立考生信息链表。考生信息输入完毕后，调用deletRepetitiveList()函数删除重复的元素。操作完毕后，调用output(bool opt)函数输出当前所有考生信息，进入while循环开始进行一系列操作。

在进行一系列操作之前，先调用listEmpty()函数判断链表是否为空（由于“删除”的操作可能会使链表被清空），如果为空是否进行下一步操作。不进行则退出程序，如果要继续进行操作，根据提示信息0~5选择对应的操作。错误的输入同样会被处理（详见总人数的输入）。

在进行一步操作之后，会继续进入while循环继续操作，直到输入0退出程序。



### 总体系统核心代码

int main() {

solution ERS;

bool flag = 1;

std::cout << "·······················" << std::endl;

std::cout << "· 考生报名系统 ·" << std::endl;

std::cout << "· Examination Registration System ·" << std::endl;

std::cout << "·······················" << std::endl;

/\* 初始化链表 \*/

ERS.InitList();

/\* 输入考生总人数 \*/

ERS.inputPersonNum();

/\* 尾插法创建链表 \*/

ERS.createList();

/\* 输出学生信息 \*/

ERS.output(1);

while (flag) {

/\* 判断链表是否为空 \*/

if (ERS.listEmpty())…;

if (flag) {

/\* 选择操作 \*/

std::cout << "请选择您要进行的操作(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)";

flag = ERS.option();

}

}

return 0;

}

/\* 输出考生具体信息 \*/

void showData(SI studentData, bool opt);

/\* 插入考生具体信息 \*/

Status solution::listInsert();

/\* 删除考生信息 \*/

Status solution::listDelete();

/\* 获取（查找）指定考生信息，根据要求选择返回具体值或是顺序序号 \*/

int solution::getElem(bool InformationorNum);

/\* 修改考生信息 \*/

Status solution::listRevise();

/\* 实现各项统计功能 \*/

void solution::statsInformation();

}

### 总体功能实现示例



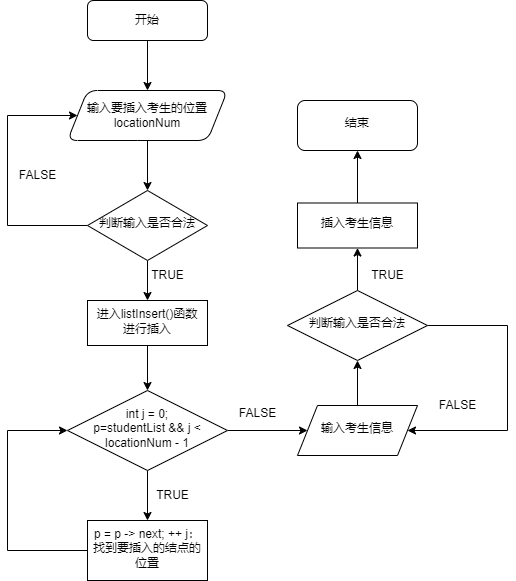
## 插入功能的实现

### 插入功能实现思路

首先输入需要插入的考生的位置(locationNum)，进行输入错误判断后，调用listInsert()进行元素的插入。

首先新建一个LNode\* p遍历链表寻找第i-1个结点，再新建一个LNode\* s，将插入的考生信息存储到s->data中，再将结点s插入原链表中。输入的考生信息同样要进行判断是否合法。

插入后，需要对原链表再进行一次除重合并处理，再输出最后的总体考生信息。



### 插入功能核心代码

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

函数名称： inputOperationLocation函数

输入参数： 无

功 能： 输入你要进行的选项

返 回 值： 无

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

void solution::inputOperationLocation()

{

while (1) {

std::cin >> locationNum;

if (std::cin.fail() || locationNum <= 0 || locationNum > personNum + 1) {

std::cin.clear();

std::cin.ignore(65536, '\n');

std::cout << "输入数据有误，请重新输入\n";

}

else {

if (std::cin.get() == '\n')

break;

}

}

}

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

函数名称： listInsert函数

输入参数： 无

功 能： 在第i个元素前插入考生信息

返 回 值： 无

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

Status solution::listInsert()

{

LNode\* p = studentList;

LNode\* s;

int j = 0;

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

while (p && j < locationNum - 1) { p = p->next; ++j; }//寻找第i-1个结点，p指向i-1结点

if (!p || j > locationNum - 1) return ERROR;//i大于表长+1或者小于1，插入位置非法

s = new LNode;

while (!inputPersonInformation(s->data)) continue;

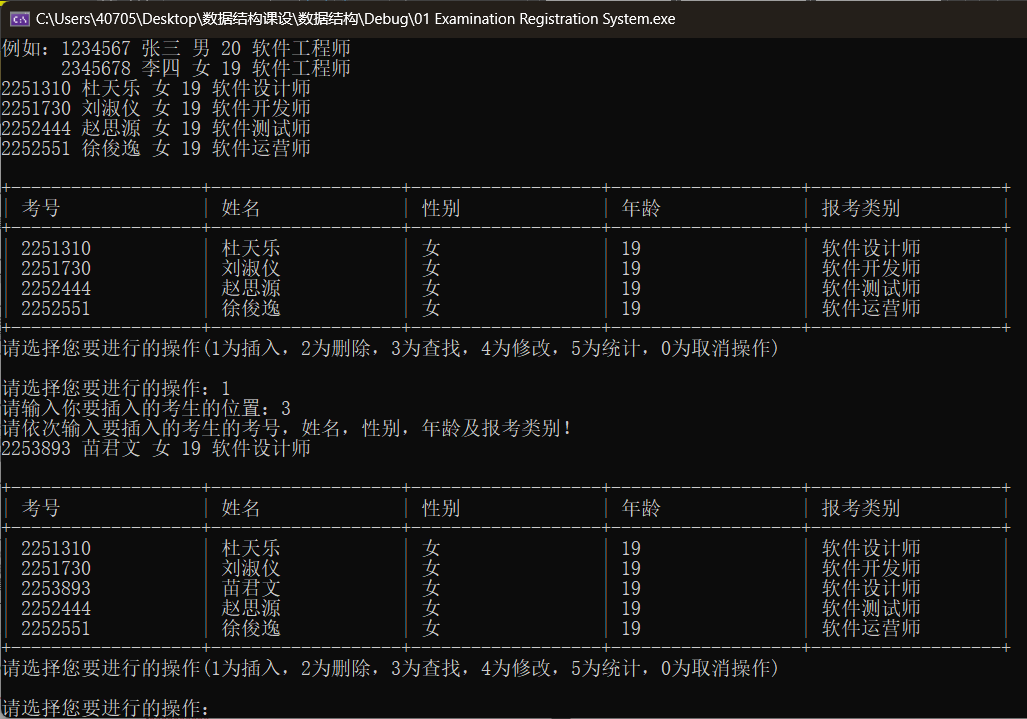
s->next = p->next;//将结点s插入L中

p->next = s;

return OK;

}

### 插入功能实现示例

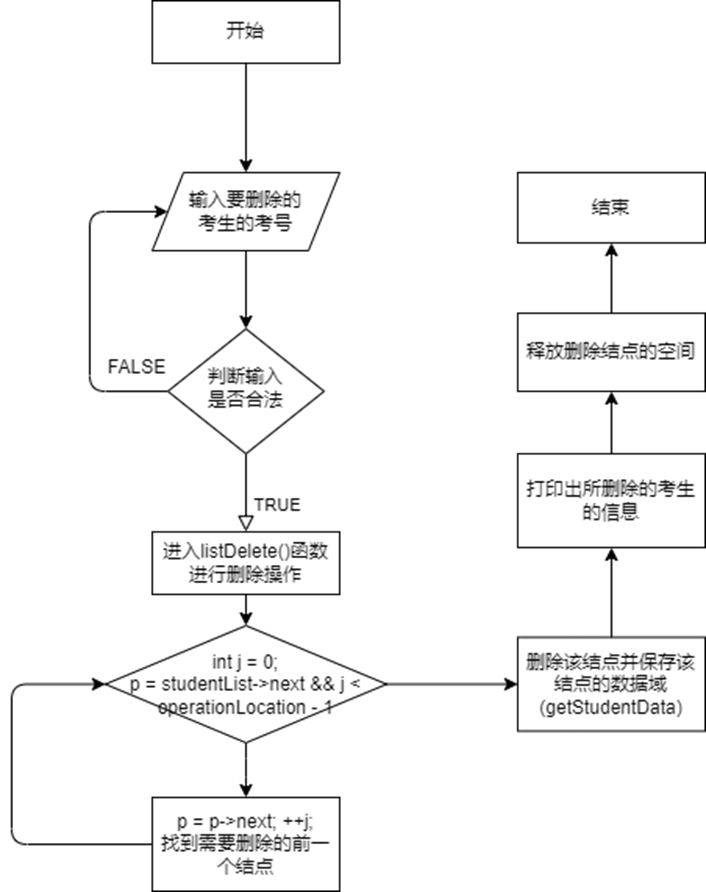


## 删除功能的实现

### 3.3.1 删除功能实现思路

首先输入需要删除的考生的考号(operationNum)。在输入考号时，需要调用int getElem(bool InformationorNum)函数获取输入的考号对应的位置序号。在调用该函数的同时还能判断输入正确与否：如果未查找到该考号，输入非法；如果查找到该考号，返回该考号的位置(operationLocation)。找到位置后，调用Status listDelete()函数进行删除。

首先新建一个LNode\* p遍历链表寻找第i-1个结点，再新建一个LNode\* q，将删除的结点的地址存储到q中以备释放，再改变删除结点的前驱结点数据域、保存删除结点的数据域到getStudentData中以备后续提示操作，最后释放删除结点的空间。



### 3.3.2 删除功能核心代码

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

函数名称： listDelete函数

输入参数： 无

功 能： 在第i个元素删除考生信息

返 回 值： 无

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

Status solution::listDelete() {

LNode\* p = studentList;

int j = 0;

while (p->next && j < operationLocation - 1) { p = p->next; ++j; }//寻找第i个结点，并令p指向其前驱

if (!(p->next) || j > operationLocation - 1)return ERROR;

LNode\* q;

q = p->next;//临时保存被删结点的地址以备释放

p->next = q->next;//改变删除结点前驱结点的指针域

getStudentData = q->data;//保存删除结点的数据域

std::cout << "你删除的考生信息是：";

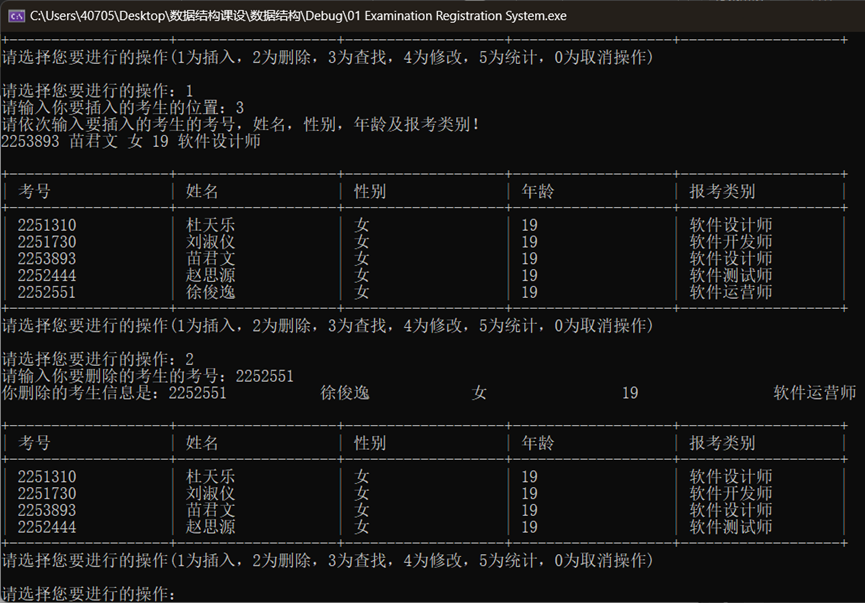
showData(getStudentData, 0);

delete q;//释放删除结点的空间

return OK;

}

### 3.3.3 删除功能实现示例



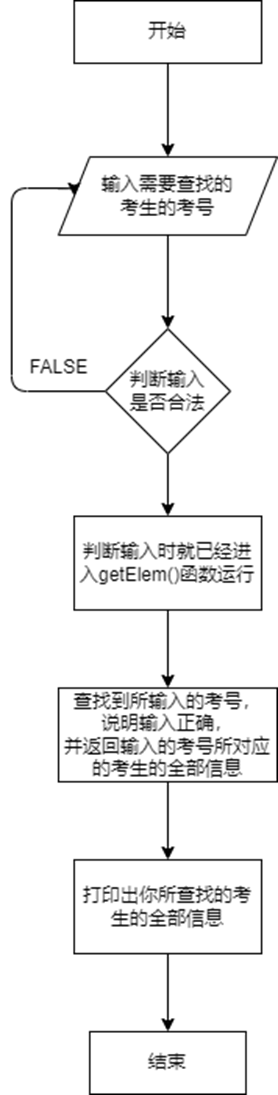
## 查找功能的实现

### 3.4.1 查找功能实现思路

首先输入需要查找的考生的考号(operationNum)。输入及错误处理见3.3.1。

在进行错误处理时，将int getElem(bool InformationorNum)函数的输入置为1，即可将需要查找的考生信息存储到getStudentData中。

对于getElem函数，同样首先新建一个LNode\* p以及int j = 1.遍历链表寻找p->data.examNum == operationNum的位置。若遍历完毕均未找到，就返回错误信息。若找到，就返回考生的序号j或者考生信息getStudentData.二者通过InformationorNum的输入来进行不同的返回。1返回考生信息，0返回序号。



### 3.4.2 查找功能核心代码

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

函数名称： getElem函数

输入参数： InformationorNum -- 1 获得当前元素的值

-- 0 获得当前元素的序号

功 能： 按值查找元素

返 回 值： ERROR -- 查找失败

OK -- 查找成功，返回所查找的考生的全部信息

j -- 查找成功，返回所要查找的排序序号

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

int solution::getElem(bool InformationorNum) {

LNode\* p;

p = studentList->next;

int j = 1;

while (p && p->data.examNum != operationNum) {

p = p->next;

j++;

}

if (!p || j > personNum) {

std::cout << "无您所要查找的考生的信息！" << std::endl;

return ERROR;

}

if (InformationorNum) {

getStudentData = p->data;

return OK;

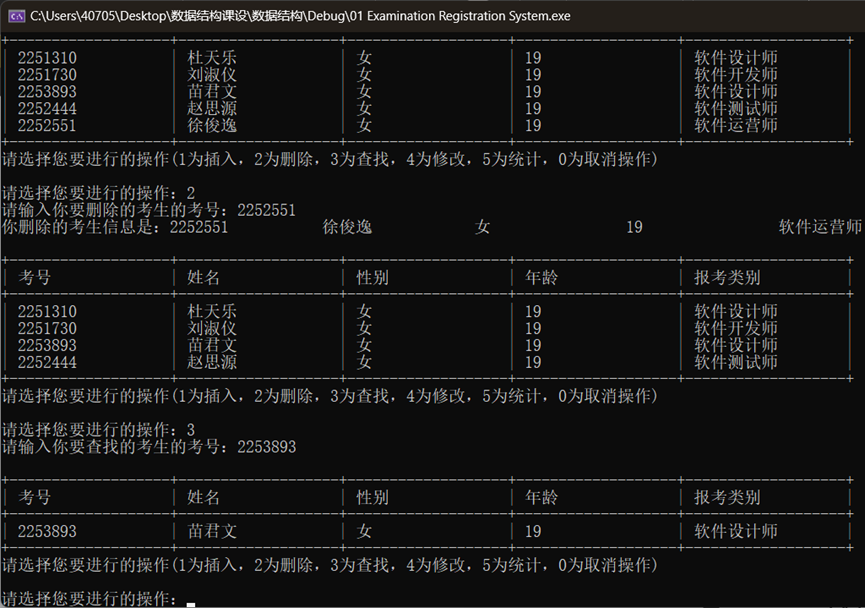
}

else

return j;

}

### 3.4.3 查找功能实现示例

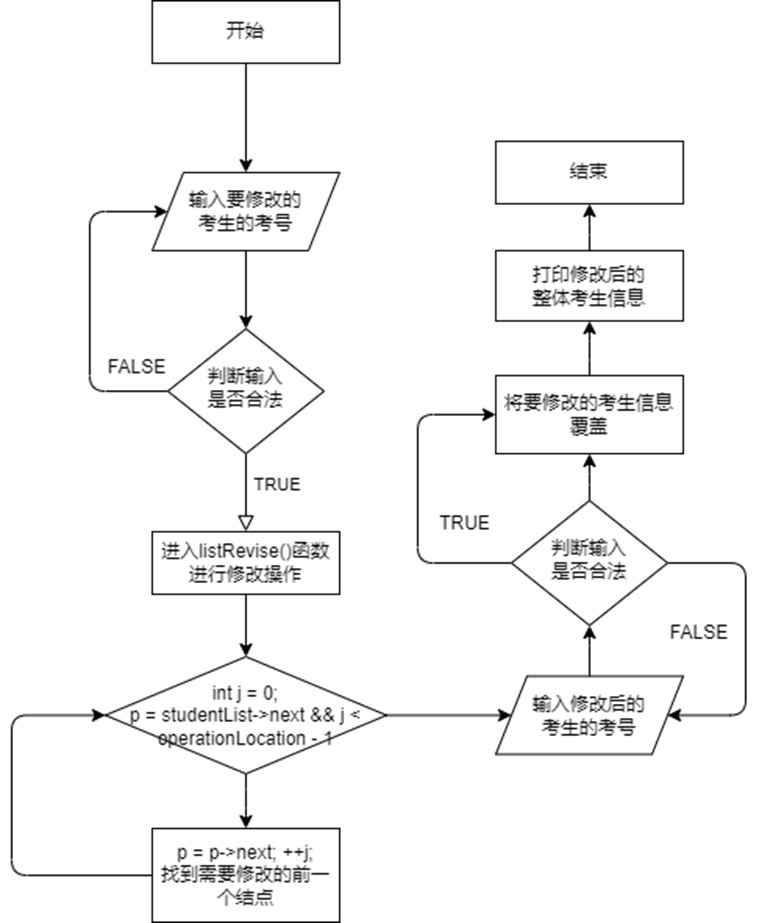


## 修改功能的实现

### 3.5.1 修改功能实现思路

首先输入需要修改的考生的考号(operationNum)。在输入考号时，需要调用int getElem(bool InformationorNum)函数获取输入的考号对应的位置序号。在调用该函数的同时还能判断输入正确与否：如果未查找到该考号，输入非法；如果查找到该考号，返回该考号的位置(operationLocation)。找到位置后，调用Status listRevise()函数进行删除。

首先新建一个LNode\* p遍历链表寻找第i-1个结点，找到后，再输入修改后的考生的信息，输入错误处理同前文。得到getStudentData后，将此时p结点的指针指向第i个结点p->next，再将其数据域赋值考生信息。



### 3.5.2 修改功能核心代码

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

函数名称： listRevise函数

输入参数： 无

功 能： 修改考生信息

返 回 值： 无

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

Status solution::listRevise() {

LNode\* p = studentList;

int j = 0;

while (p->next && j < operationLocation - 1) { p = p->next; ++j; }//寻找第i个结点，并令p指向其前驱

if (!(p->next) || j > operationLocation - 1)return ERROR;

std::cout << "请依次输入修改后的的考生的考号，姓名，性别，年龄及报考类别！\n";

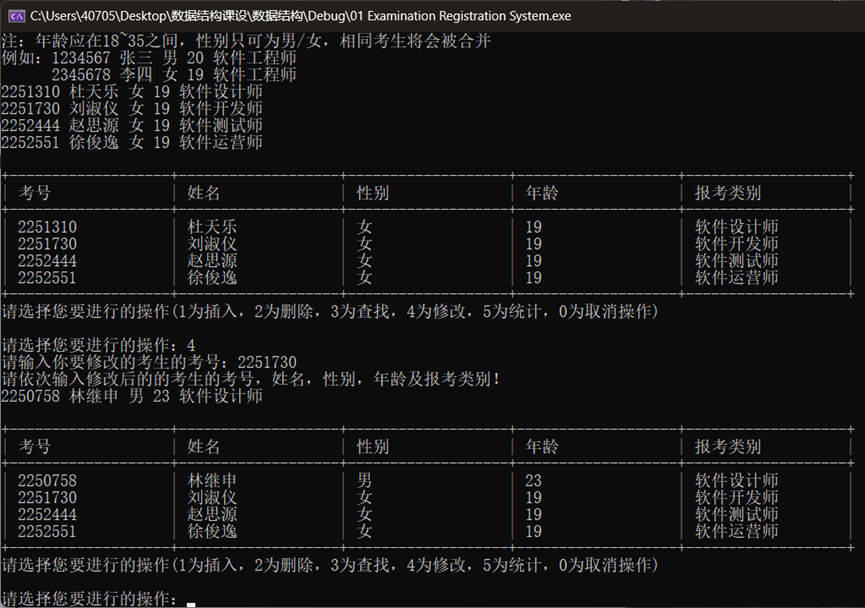
while (!inputPersonInformation(getStudentData)) continue;

p->next->data = getStudentData;

return OK;

}

### 3.5.3 修改功能实现示例



## 统计功能的实现

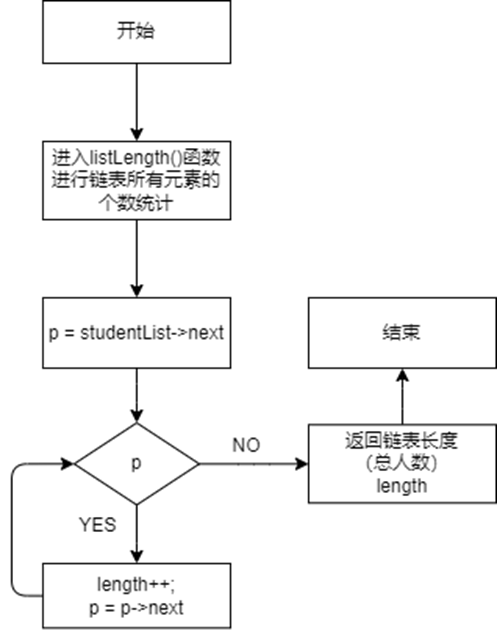
首先进入deletRepetitiveList()函数，将现有的考生名单降重，再输出现有的所有考生信息然后再根据输入提示输入Y/N（大小写不敏感）来进行进一步操作：统计考生总人数、统计男女性别人数比例、统计每个年龄所占的人数、统计所报考的种类的人数及比例。

### 统计考生总人数功能

#### 3.6.1.1 统计考生总人数实现思路

进入listLength()函数统计链表所有元素的个数，这个个数就是所有考生的总人数，返回总长度length即为总人数。

在该函数中，建立LNode\* p指向studentList->next，若其存在，length++，直到p!，退出while循环，返回length.



#### 3.6.1.2 统计计考生总人数实现核心代码

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

函数名称： listLength函数

输入参数： 无

功 能： 统计链表长度

返 回 值： 链表长度length

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

int solution::listLength() {

LNode\* p = studentList->next;

int length = 0;

while (p) {

length++;

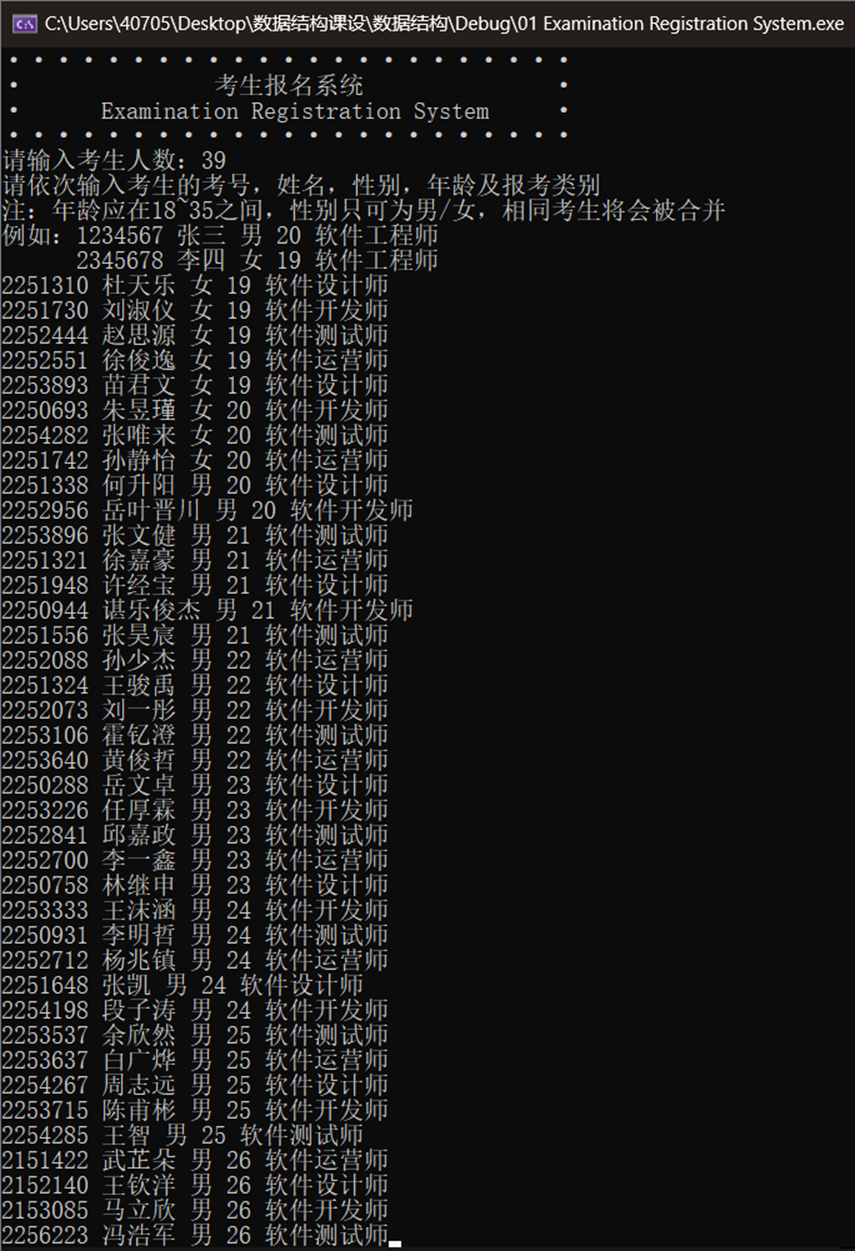
p = p->next;

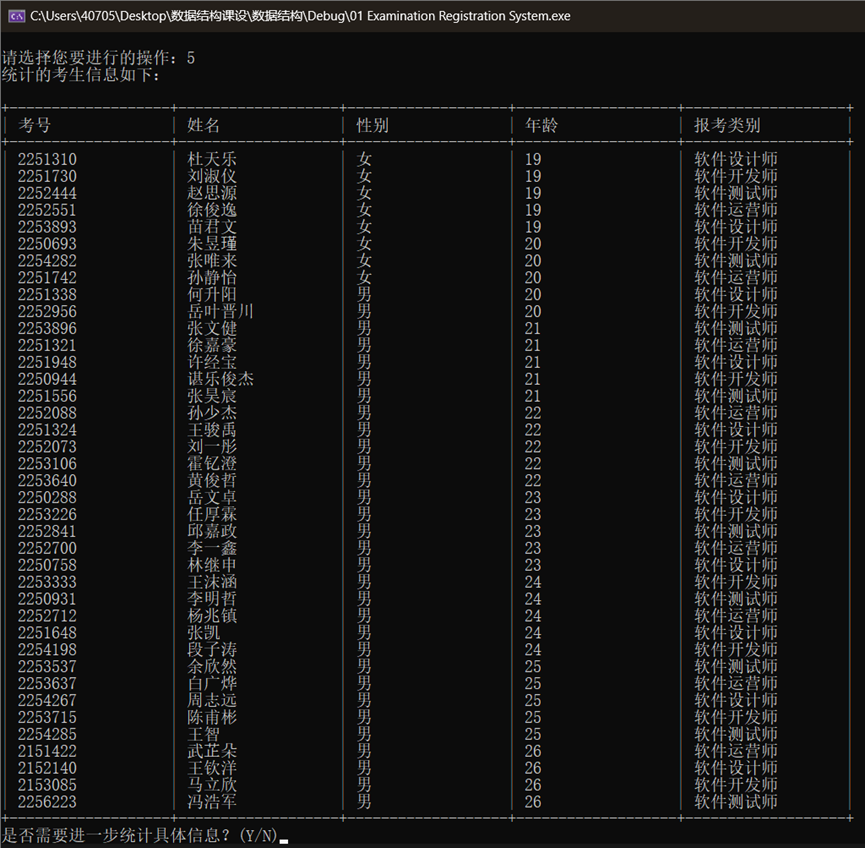
}

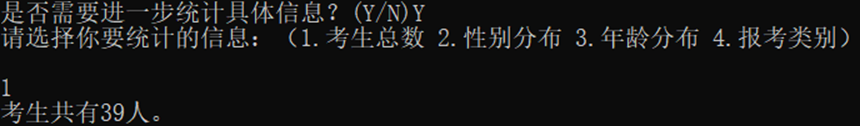
return length;

}

#### 3.6.1.3 统计考生总人数实现示例





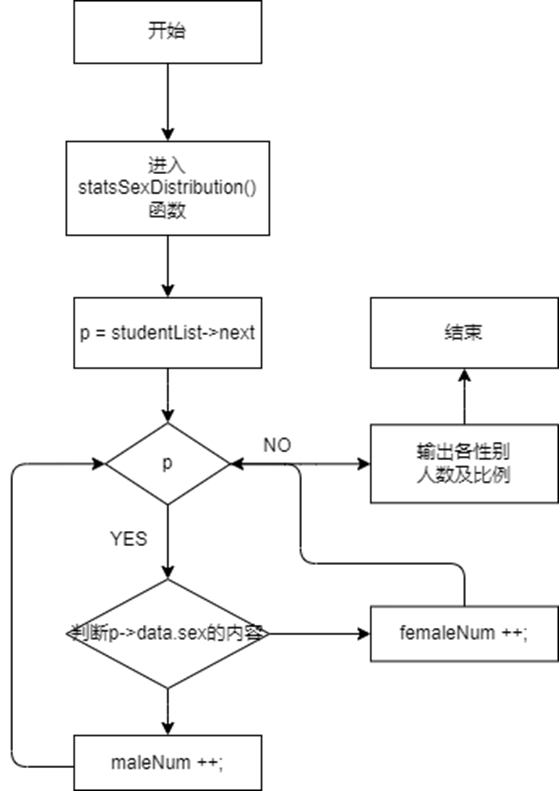


### 3.6.2 统计性别人数分布功能

#### 3.6.2.1 统计性别人数分布实现思路

进入statsSexDistribution()函数，新建LNode\*p对链表进行遍历，若数据域p->data.sex的值为“女”，则femaleNum++，若为“男”，则maleNum++其他输入则在最初的考生信息系统建立的输入便处理过，算作错误输入。

遍历完后，输出性别种类及其对应的人数和比例。



#### 3.6.2.2 统计性别人数分布实现核心代码

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

函数名称： statsSexDistribution函数

输入参数： 无

功 能： 统计男女人数分布

返 回 值： 无

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

void solution::statsSexDistribution()

{

LNode\* p = studentList->next;

while (p) {

if (p->data.sex[0] == -59 && p->data.sex[1] == -82)//女

femaleNum++;

else if (p->data.sex[0] == -60 && p->data.sex[1] == -48)//男

maleNum++;

p = p->next;

}

std::cout << std::endl;

std::cout << "考生总人数：" << personNum << std::endl;

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

std::cout << std::setw(20) <<std::left << "| 性别种类";

std::cout << std::setw(20) <<std::left << "| 性别人数";

std::cout << std::setw(20) <<std::left << "| 性别比例 |\n";

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

std::cout << "| " << std::setw(18) <<std::left << "男";

std::cout << "| " << std::setw(18) <<std::left << maleNum;

std::cout << "| " << std::setw(17) <<std::left << maleNum \* 100.0 / personNum << '%';

std::cout << "|\n";

std::cout << "| " << std::setw(18) <<std::left << "女";

std::cout << "| " << std::setw(18) <<std::left << femaleNum;

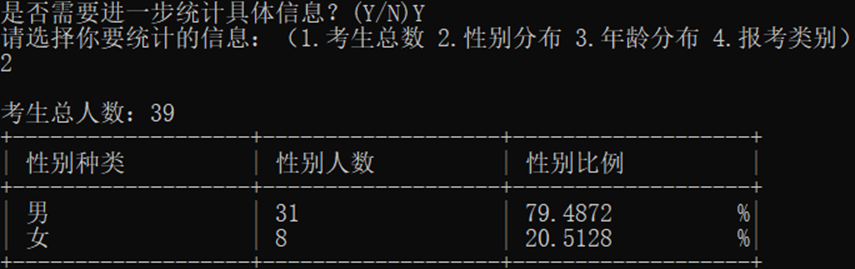
std::cout << "| " << std::setw(17) <<std::left << femaleNum \* 100.0 / personNum << '%';

std::cout << "|\n";

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

}

#### 3.6.2.3 统计性别人数分布实现示例

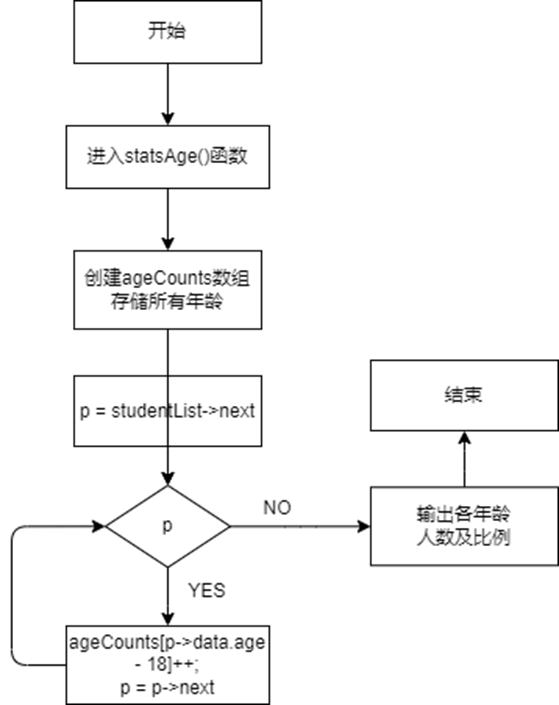


### 3.6.3 统计年龄人数分布功能

#### 3.6.3.1 统计年龄人数分布实现思路

进入statsSexDistribution()函数，新建一个ageCounts数组来统计年龄，数组大小为指定的最大最小年龄差+1.新建LNode\*p对链表进行遍历，统计每个年龄所对应的学生数量。数组下标即为p->data.age – 18。

遍历完后，输出年龄及其对应的人数和比例。使用flag变量来存储已经遍历过的考生人数，防止考生人数过于集中导致数组值为0的数较多，循环无效重复进行。



#### 3.6.3.2 统计年龄人数分布实现核心代码

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

函数名称： statsAge函数

输入参数： 无

功 能： 统计年龄分布

返 回 值： 无

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

void solution::statsAge()

{

int ageCounts[maxAge - minAge + 1];

int flag = 0;

for (int j = 0; j < maxAge - minAge + 1; j++)

ageCounts[j] = 0;

LNode\* p = studentList->next;

//遍历链表并统计每个年龄对应的学生数量

while (p) {

ageCounts[p->data.age - 18]++;

p = p->next;

}

//输出结果

std::cout << std::endl;

std::cout << "考生总人数：" << personNum << std::endl;

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

std::cout << std::setw(20) <<std::left << "| 年龄大小";

std::cout << std::setw(20) <<std::left << "| 年龄人数";

std::cout << std::setw(20) <<std::left << "| 年龄比例 |\n";

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

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

if (flag == personNum)

break;

if (ageCounts[i] > 0) {

std::cout << "| " << std::setw(18) <<std::left << i + 18;

std::cout << "| " << std::setw(18) <<std::left << ageCounts[i];

std::cout << "| " << std::setw(17) <<std::left << ageCounts[i] \* 100.0 / personNum << '%';

std::cout << "|\n";

flag+= ageCounts[i];

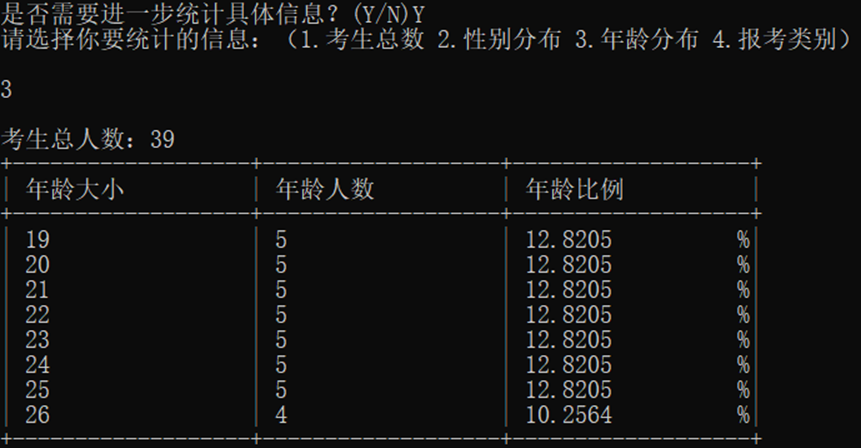
}

}

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

}

#### 3.6.3.3 统计年龄人数分布实现示例



### 3.6.4 统计报考类别人数分布功能

#### 3.6.4.1 统计报考类别人数实现思路

进入countApplicationCategory()函数，新建一个LNode\* p指向考生信息的链表。char型指针变量指向p数据域中考生信息的类别。再新建一个ACC\*指针变量指向类里原有的ACC\*变量，开始进行遍历统计寻找是否已经存在该报考类型。如果该报考类型没有被统计过，添加新的统计项。

将所有报考信息类别统计完毕后，每个报考类别里所含的人数也在ACC中的count里被统计了。

遍历完毕后，输出各报考类型的人数及比例。



#### 3.6.4.2 统计报考类别人数实现核心代码

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

函数名称： countApplicationCategory函数

输入参数： 无

功 能： 统计报考信息分布

返 回 值： 无

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

void solution::countApplicationCategory()

{

LNode\* p = studentList->next;

while (p) {

char\* category = p->data.applicationCategory;

// 寻找是否已存在该报考类型的统计

ACC\* current = categoryCounts;

while (current != nullptr) {

if (strcmp(category, current->category) == 0) {

current->count++;

break;

}

current = current->next;

}

// 该报考类型未统计过，添加新的统计项

if (current == nullptr) {

ACC\* newCount = new ACC;

strcpy(newCount->category, category);

newCount->count = 1;

newCount->next = categoryCounts;

categoryCounts = newCount;

}

p = p->next;

}

// 输出统计结果

std::cout << "报考类型统计:\n";

ACC\* current = categoryCounts;

std::cout << std::endl;

std::cout << "考生总人数：" << personNum << std::endl;

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

std::cout << std::setw(20) <<std::left << "| 报考种类";

std::cout << std::setw(20) <<std::left << "| 报考人数";

std::cout << std::setw(20) <<std::left << "| 报考比例 |\n";

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

while (current != nullptr) {

std::cout << "| " << std::setw(18) <<std::left << current->category;

std::cout << "| " << std::setw(18) <<std::left << current->count;

std::cout << "| " << std::setw(17) <<std::left << current->count \* 100.0 / personNum << '%';

std::cout << "|\n";

ACC\* temp = current;

current = current->next;

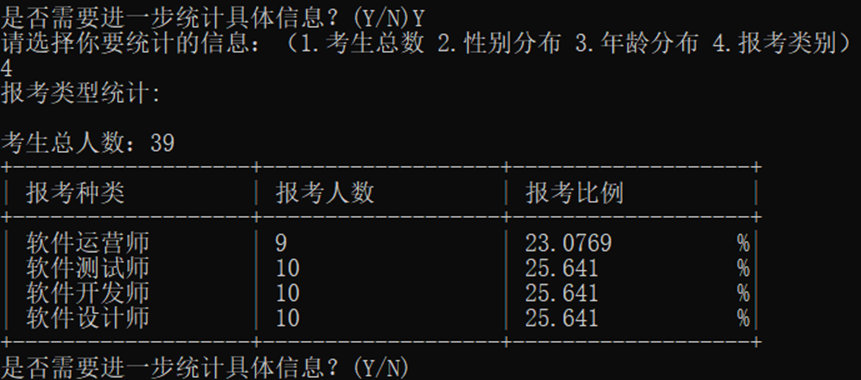
delete temp;

}

std::cout << "+-------------------+-------------------+-------------------+" << std::endl;

}

#### 3.6.4.3 统计报考类别人数实现示例



# 项目测试

## 功能测试

### 插入功能测试

#### 在开头、中间、结尾位置插入

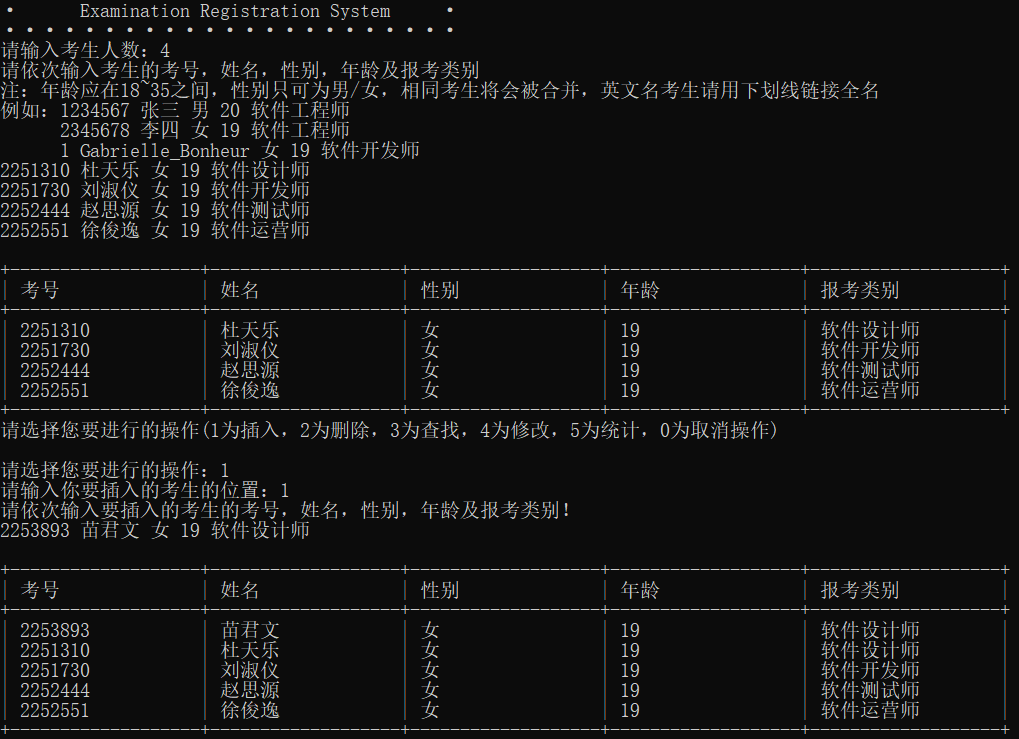
**测试用例：**1 ; 2253893 苗君文 女 19 软件设计师

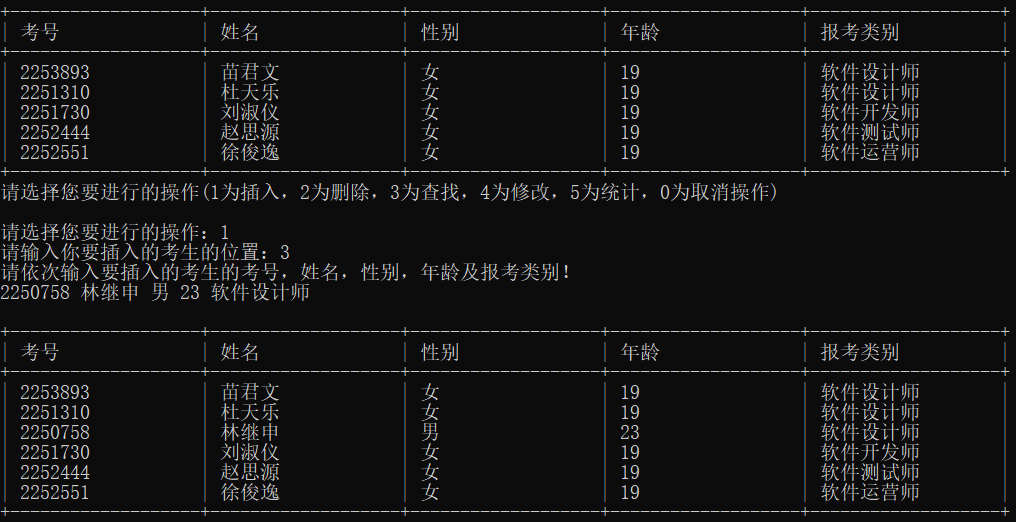
3 ; 2250758 林继申 男 23 软件设计师

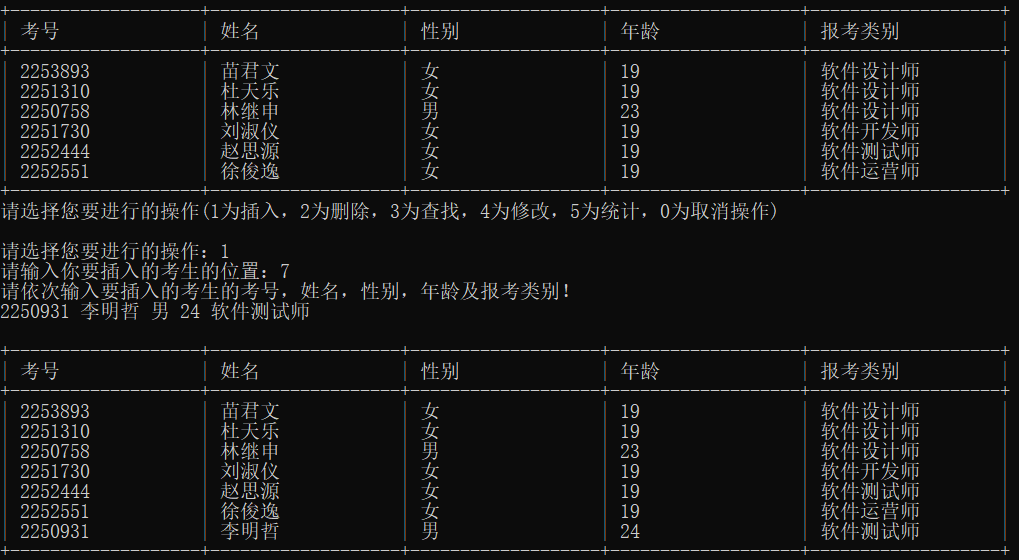
7 ；2250931 李明哲 男 24 软件测试师

**预期结果：**正常运行并打印出现有的考生信息名单

**实验结果：**





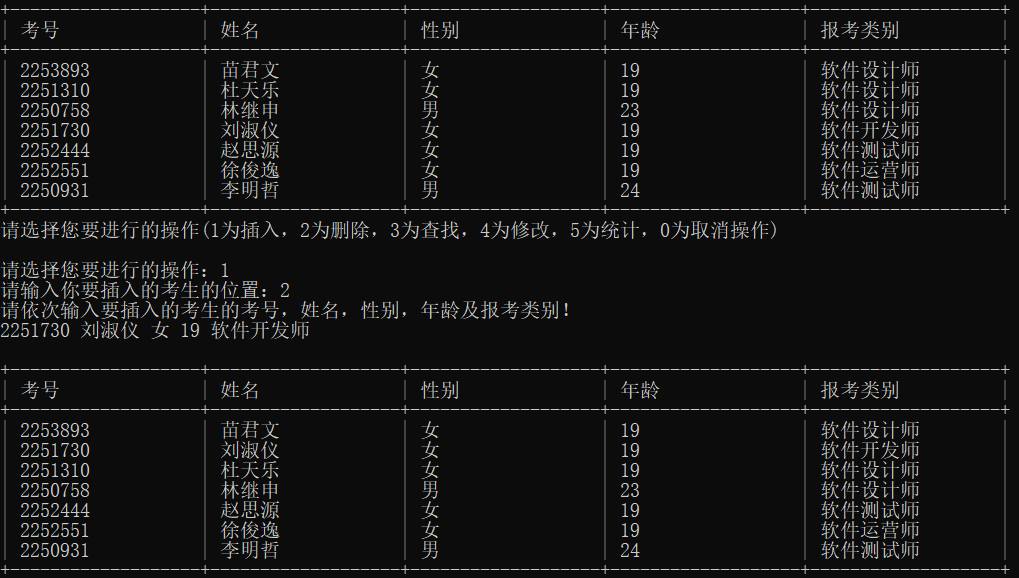


#### 插入相同的考生信息

**测试用例：**2251730 刘淑仪 女 19 软件开发师

**预期结果：**程序正常运行，但会降重，可以起到改变该生顺序的作用

**实验结果：**



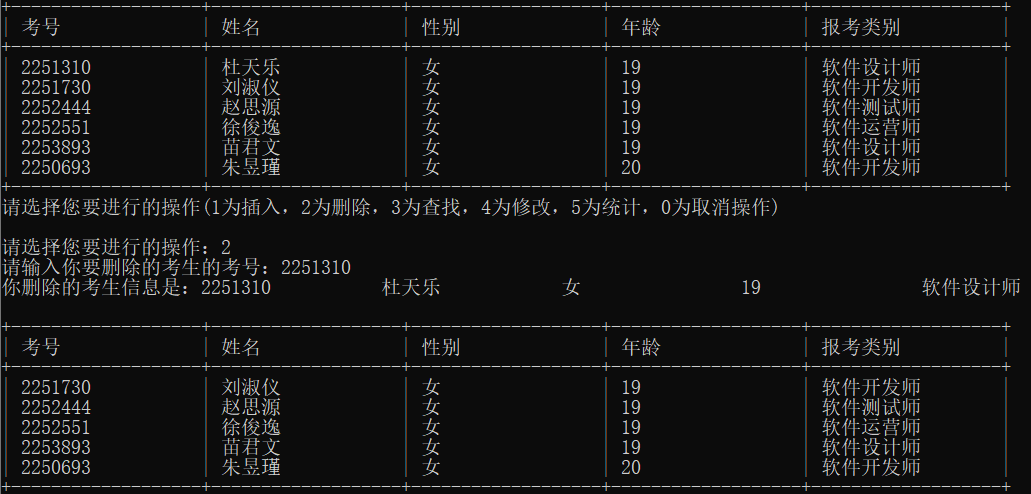
### 4.1.2 删除功能测试

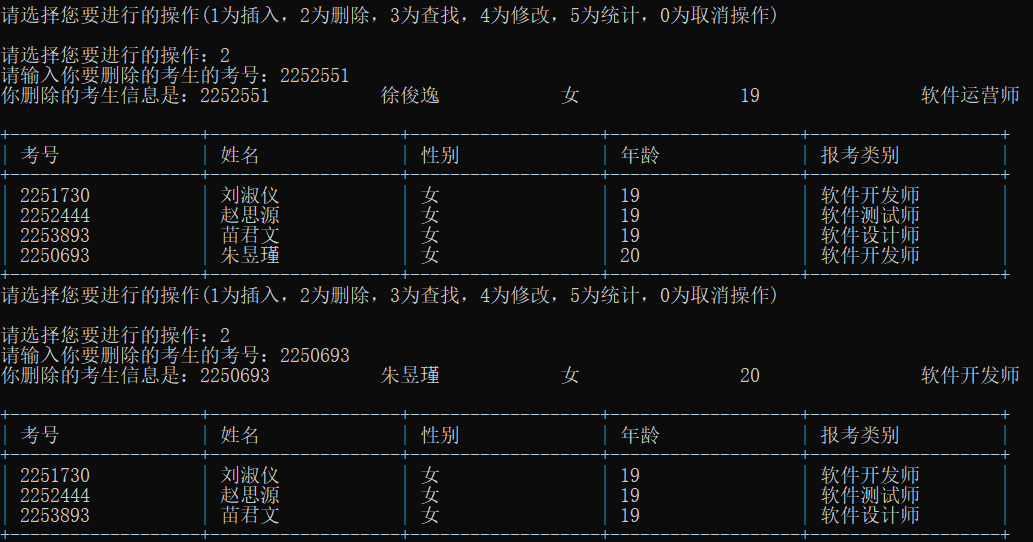
**测试用例：**2251310 / 2252551 / 2250693

**用例说明：**在开头、中间、结尾位置删除

**预期结果：**正常运行并打印出现有的考生信息名单

**实验结果：**





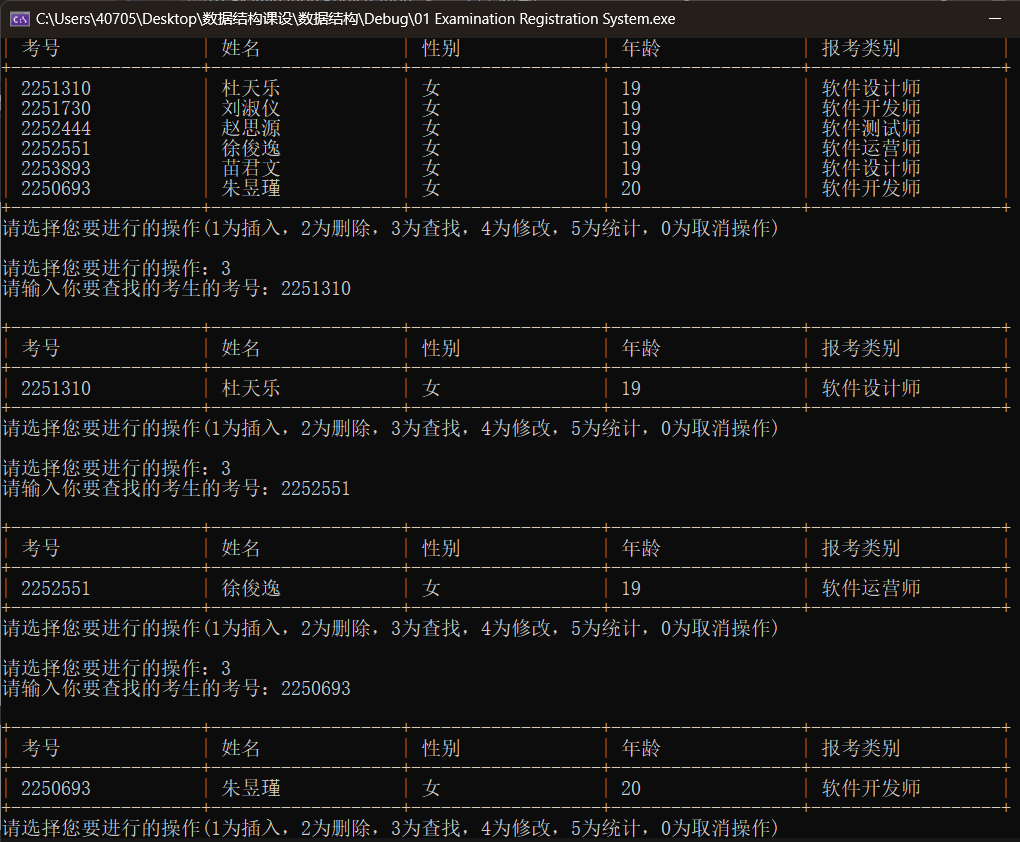
### 4.1.3 查找功能测试

**测试用例：**2251310 / 2252551 / 2250693

**用例说明：**在开头、中间、结尾位置查找

**预期结果：**正常运行并打印出现有的考生信息名单

**实验结果：**



### 4.1.4 修改功能测试

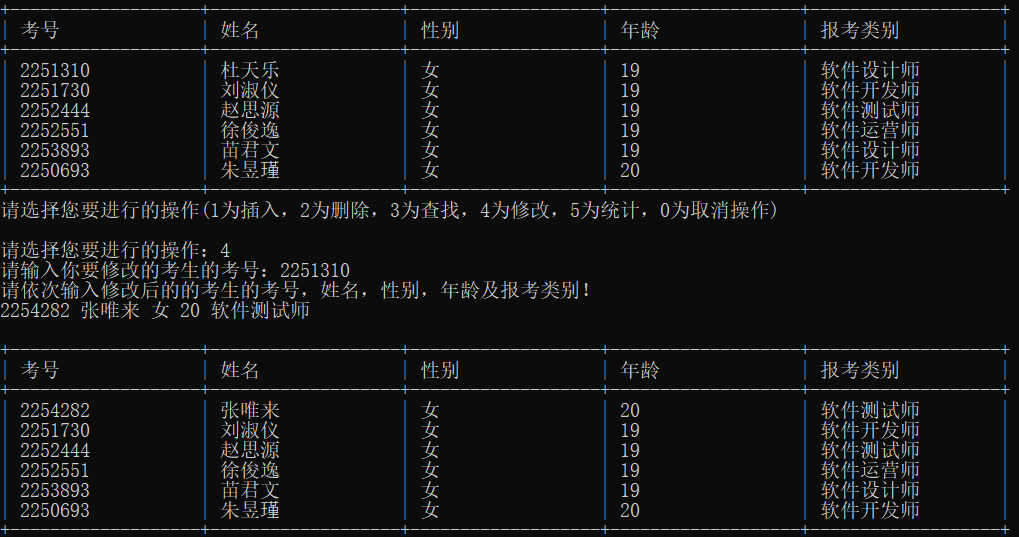
#### 4.1.4.1 在开头、中间、结尾位置修改

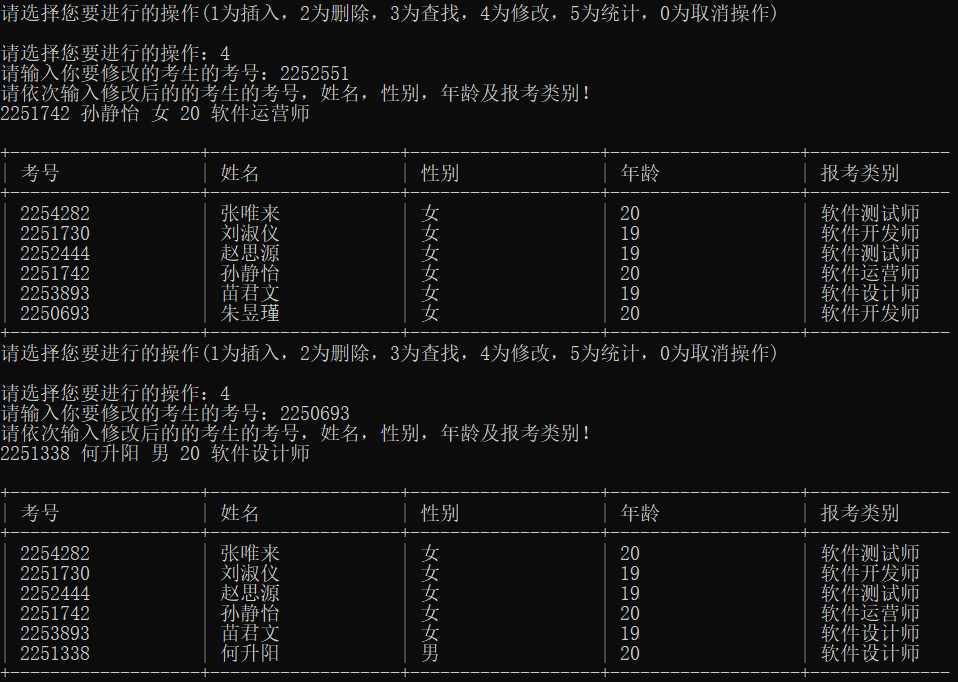
**测试用例：**2251310 / 2252551 / 2250693

**用例说明：**在开头、中间、结尾位置修改

**预期结果：**正常运行并打印出现有的考生信息名单

**实验结果：**





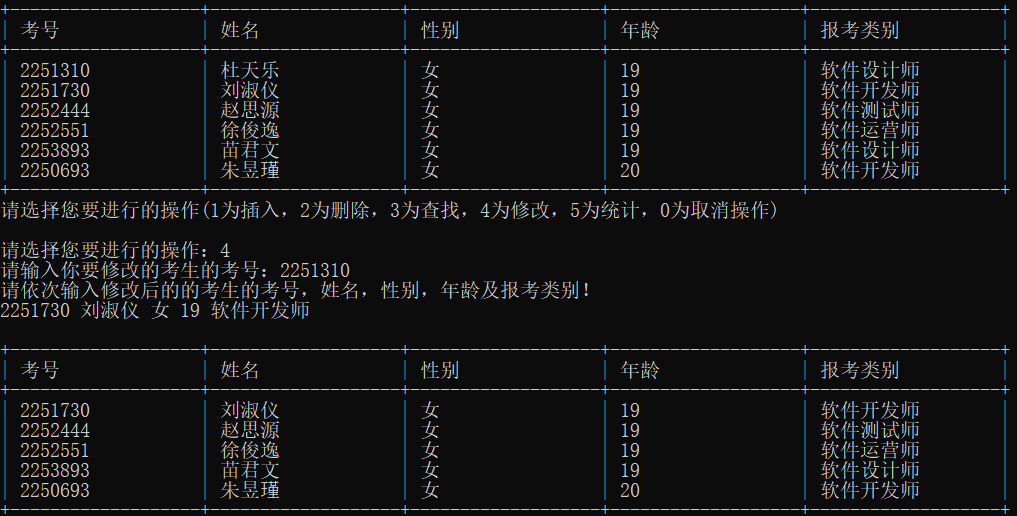
#### 4.1.4.4 修改时输入已有的考生信息

**测试用例：**2251310 / 2251730 刘淑仪 女 19 软件开发师

**用例说明：**修改输入已有的考生信息看是否覆盖

**预期结果：**正常运行并打印出现有的考生信息名单

**实验结果：**



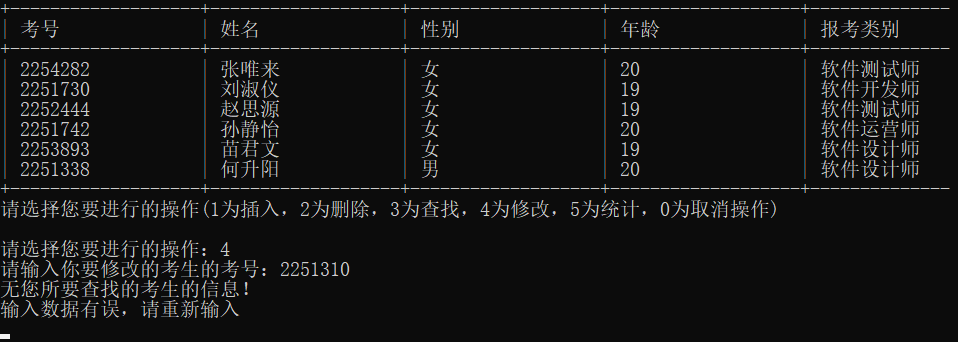
#### 4.1.4.4 修改时输入已经被修改过的考生的考号

**测试用例：**2251310

**用例说明：**修改时输入已经被修改过的考生的考号

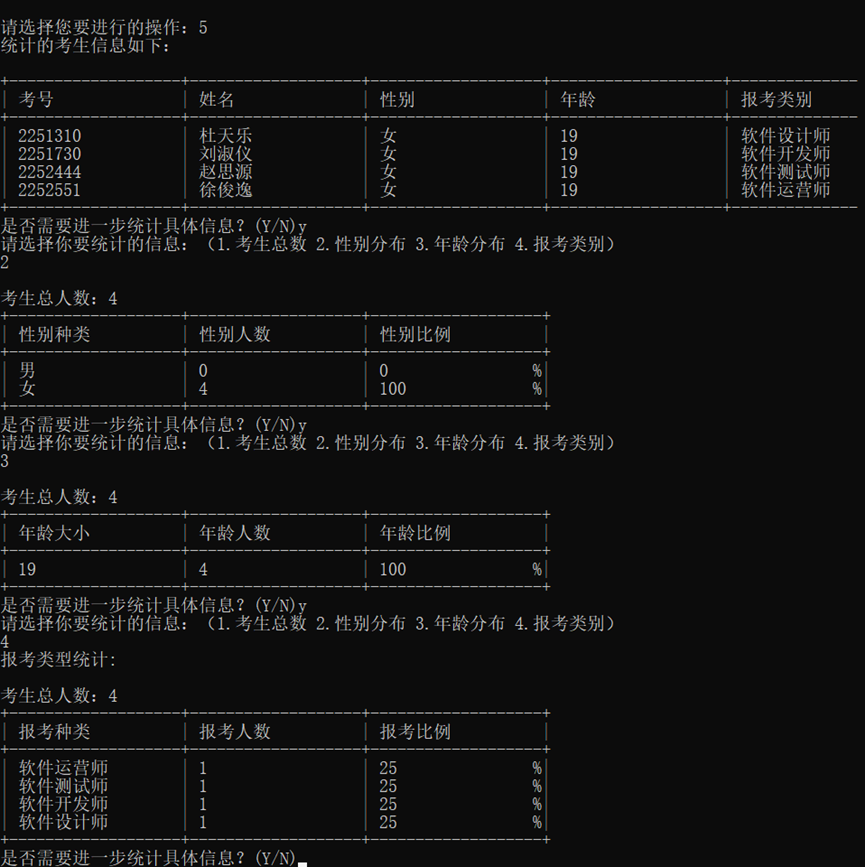
**预期结果：**提示错误信息，程序不崩溃

**实验结果：**



### 4.1.5 统计功能测试

正常情况：

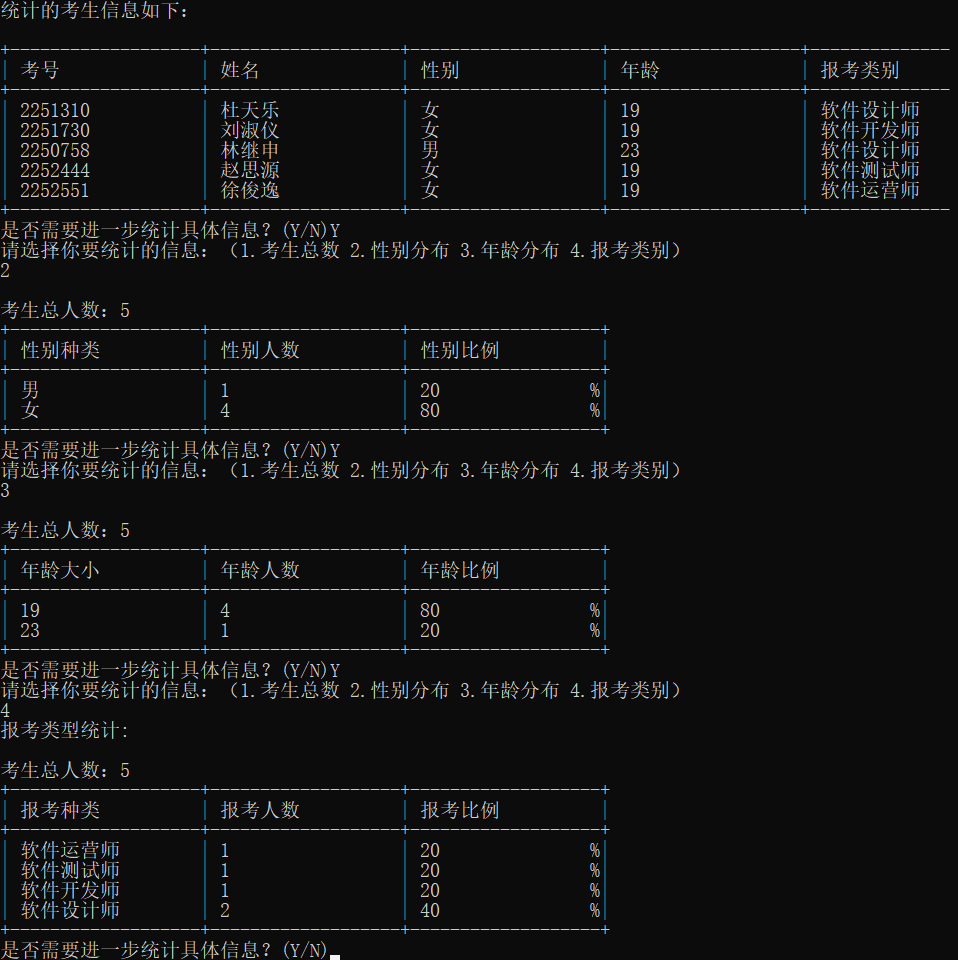


#### 4.1.5.1 经过插入后统计是否有误差

**测试用例：**3 ; 2250758 林继申 男 23 软件设计师

**预期结果：**正常运行，没有误差

**实验结果：**

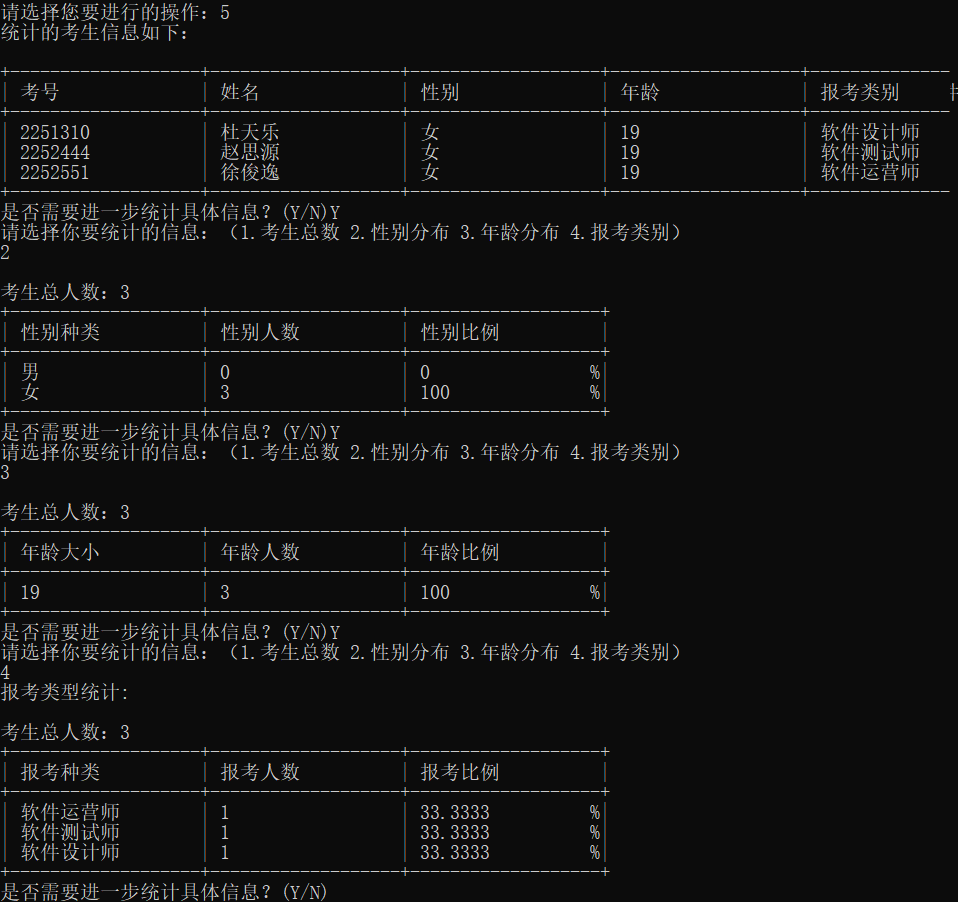


#### 4.1.5.2 经过删除后统计是否有误差

**测试用例：**2251730

**预期结果：**正常运行，没有误差

**实验结果：**

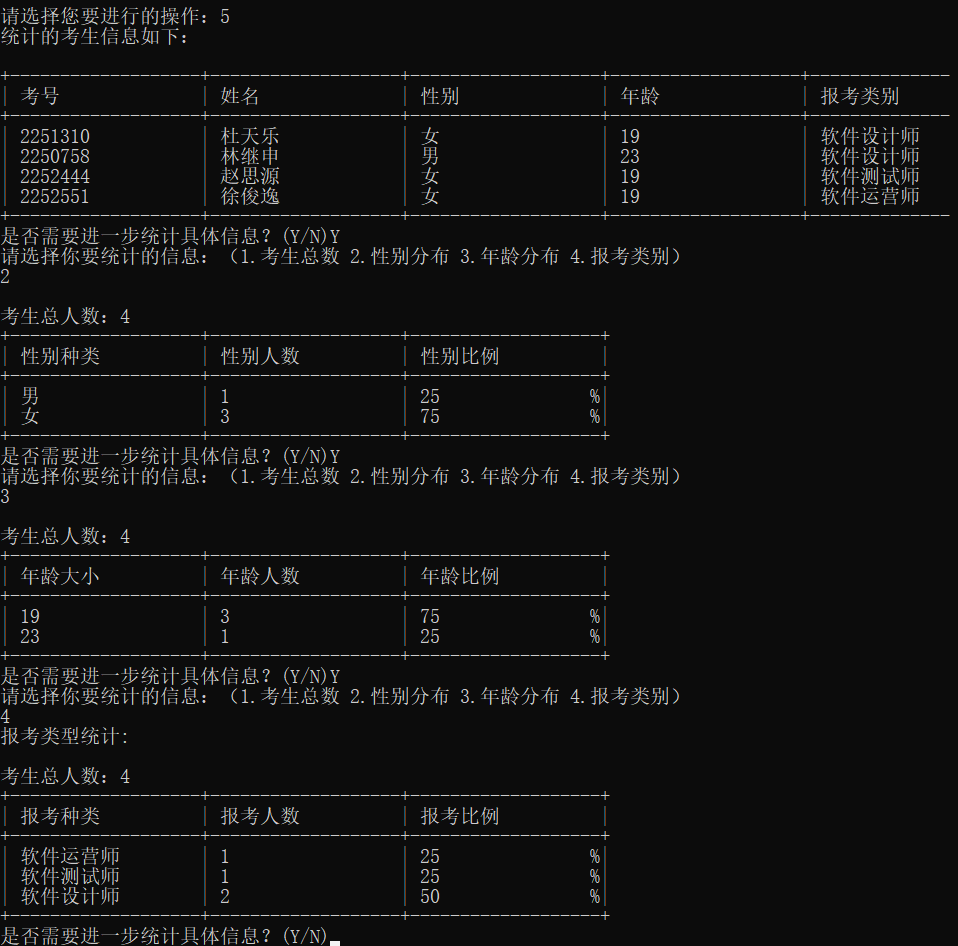


#### 4.1.5.3 经过修改后统计是否有误差

**测试用例：**2251730 / 2250758 林继申 男 23 软件设计师

**预期结果：**正常运行，没有误差

**实验结果：**



## 错误测试

### 考生人数错误

#### 4.2.1.1 考生人数输入判断思路

使用cin读取考生人数(personNum)，考生人数为int型变量，故非法输入会使得读取错误从而进入第一个if分支，提示错误信息；同时，如果考生人数过小或过大同样会被判错误输入。

当读取到错误输入时，调用cin.clear()和cin.ignore()来清空缓存区，while循环继续读取输入，直到输入正确，break推出循环。

#### 4.2.1.2 考生人数输入核心代码

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

函数名称： inputPersonNum函数

输入参数： 无

功 能： 处理学生人数

返 回 值： 无

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

void solution::inputPersonNum()

{

std::cout << "请输入考生人数：";

while (1) {

std::cin >> personNum;

if (std::cin.fail() || personNum < minLength || personNum > maxLength) {

std::cin.clear();

std::cin.ignore(65536, '\n');

std::cout << "输入数据有误，请重新输入\n";

}

else {

if (std::cin.get() == '\n')

break;

else {

std::cin.clear();

std::cin.ignore(65536, '\n');

std::cout << "输入数据有误，请重新输入\n";

}

}

}

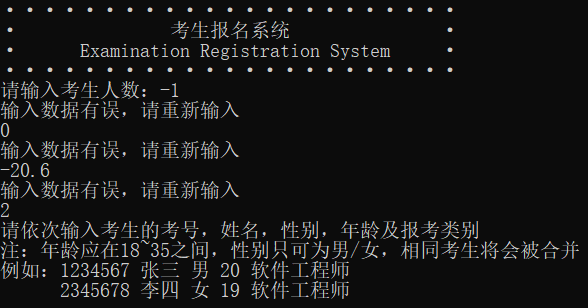
}

#### 4.2.1.3 考生人数输入示例

**测试用例：**-1 / 0 / -20.6

**用例说明：**测试负数、0

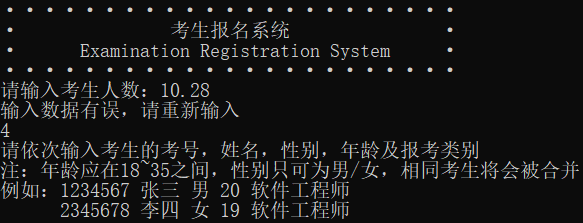
**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

**测试用例：**10.28

**用例说明：**测试小数

**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：**

**测试用例：**10 22

**用例说明：**测试多个输入

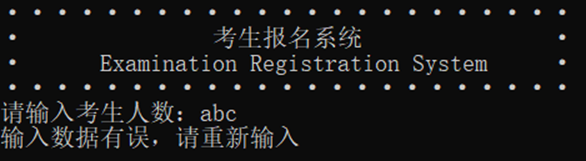
**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

**测试用例：**abc

**用例说明：**测试非法输入（如字母）

**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：**

### 考生信息错误

#### 4.2.2.1 考生信息输入判断思路

首先进入while循环，由于需要判断考生姓名是否超出字符数组最大限制，先初始化studentData.name[maxNameLength - 1] = '\0';便于后续错误判断。再创建一个字符串存储输入。用于string不可用，故创建一个char型字符数组，再自我实现了一个myGetline函数来获取输入。

通过sstream中的iss来提取输入内容至结构体studentData内的各项。当考号数字大小不合理、年龄不在范围内或输入的是小数、姓名超过上限、性别不是男或女时为错误输入。此时屏幕上输出错误提示，清空iss缓存区并返回false，重新输入。

当以上条件均正常满足，且考生报考类别不为空（防止有信息漏输）时，退出循环，返回true，输入成功。

#### 4.2.2.2 考生信息输入核心代码

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

函数名称： inputPersonInformation函数

输入参数： SI studentData

功 能： 输入考生信息

返 回 值： true --输入成功

false --输入失败

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

bool solution::inputPersonInformation(SI& studentData)

{

while (true) {

studentData.name[maxNameLength - 1] = '\0';

/\* 创建一个字符串存储输入 \*/

char input\_line[65536] = { 0 };

/\* 获取输入 \*/

myGetline(input\_line, 65536);

std::istringstream iss(input\_line);

iss >> studentData.examNum >> studentData.name >> studentData.sex >> studentData.age >> studentData.applicationCategory;

/\* 非法输入、年龄过大或过小、性别不是男/女\*/

if (iss.fail() || studentData.examNum <=0 ||studentData.age < minAge || studentData.age > maxAge || !((studentData.sex[0] == -59 && studentData.sex[1] == -82) ||

(studentData.sex[0] == -60 && studentData.sex[1] == -48))||studentData.name[maxNameLength - 1]!='\0' || studentData.applicationCategory[0]=='.') {

std::cout << "输入数据非法，请重新输入（从头输入）\n";

iss.str("");

iss.clear();

iss.ignore(65536, '\n');

return false;

}

if (studentData.applicationCategory) break; // 退出循环

}

return true;

}

/\* getline函数自我实现 \*/

int myGetline(char s[], int lim)

{

int c, i;// getchar()返回值为int型，故c用int型定义

for (i = 0; i < lim - 1 && (c = getchar()) != EOF && c != '\n'; ++i)

s[i] = c;

if (c == '\n') {

s[i] = c;

++i;

}

s[i] = '\0';

return i;

}

#### 4.2.2.3 考生信息输入示例

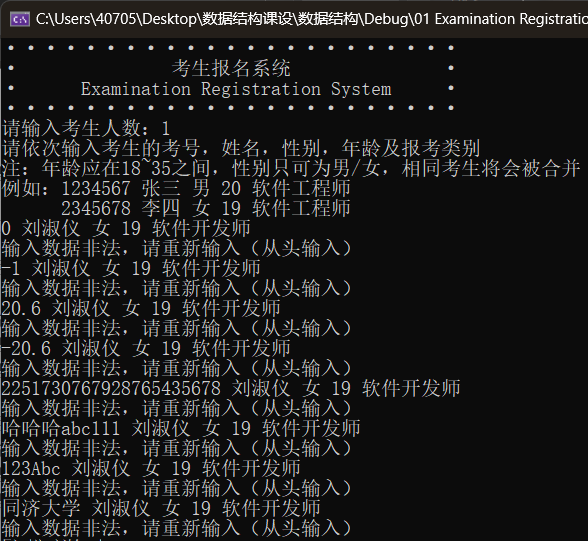
**测试用例：**考号分别为：0 / -1 / 20.6 / -20.6 / 2251730767928765435678 / 哈哈哈abc111 / 123Abc / 同济大学

后续信息统一为 刘淑仪 女 19 软件开发师

**用例说明：**考号输入非法：非正整数、数据过大超上限、汉字数字混合输入、汉字输入

**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：**



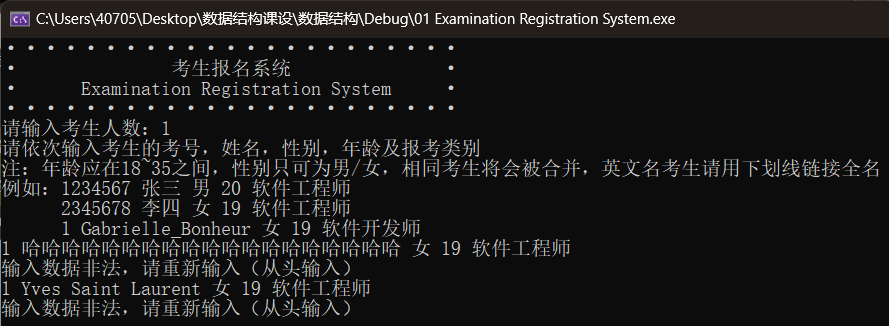
**测试用例：**姓名分别为：哈哈哈哈哈哈哈哈哈哈 / Yves Saint Laurent

其余信息统一为：1 女 19 软件工程师

**用例说明：**姓名输入非法：非法字符、过长的名字、英文名输入不带下划线

**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：**



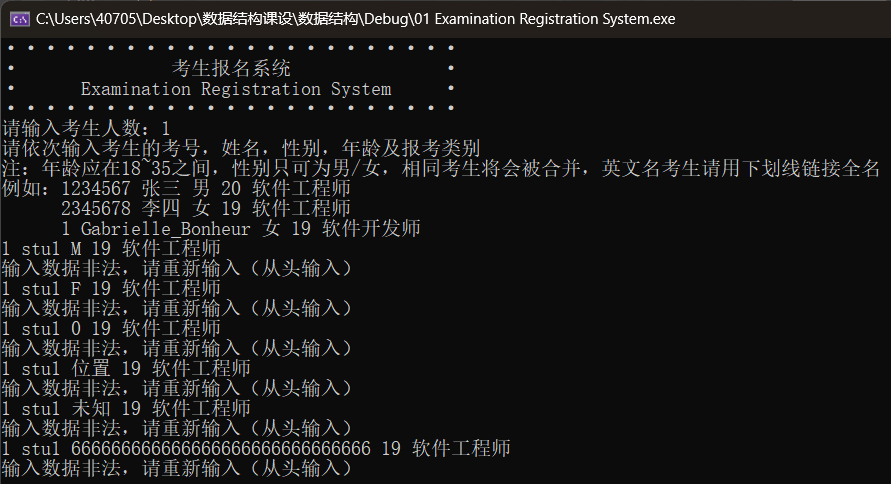
**测试用例：**性别分别为：0 / M / F / 未知 / 666666666666666666666666666666

其余信息统一为：1 stu1 19 软件工程师

**用例说明：**性别输入非法：除男/女以外的输入均为非法

**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：**



**测试用例：**年龄分别为：0 / -1 / 20.6 / -20.6 / 2251730767928765435678 / 哈哈哈abc111 / 123Abc / 同济大学

其余信息统一为 1 stu1 女 软件开发师

**用例说明：**年龄输入非法：非正整数、数据过大超上限、汉字数字混合输入、汉字输入

**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：**

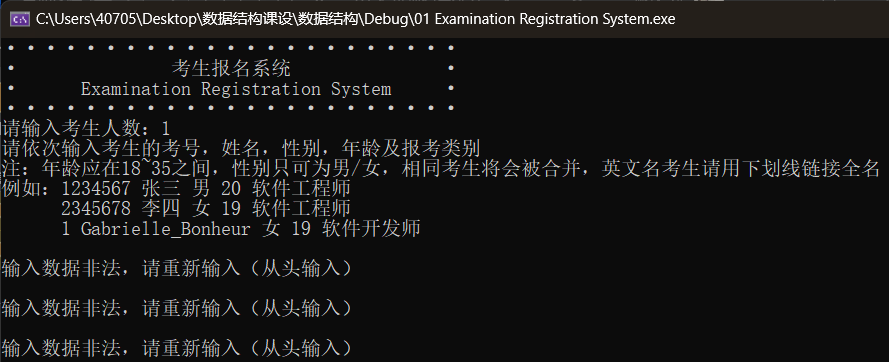


**测试用例：** ‘\n’

**用例说明：**无输入

**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：**



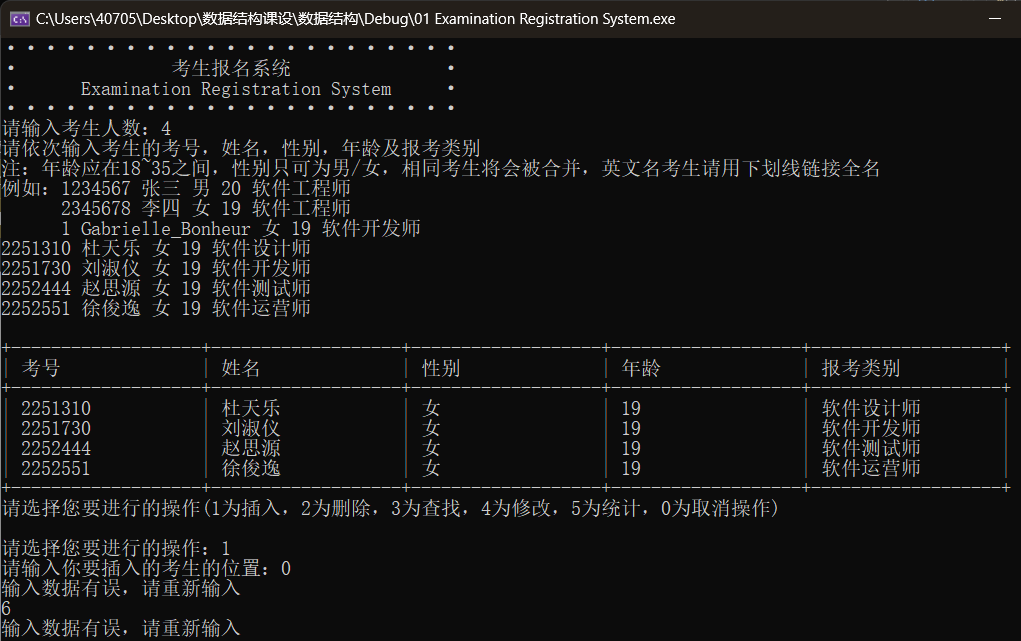
### 插入位置不存在

**测试用例：** 0 / 大于目前的人数总数

**用例说明：**测试插入位置不存在的反应

**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：**



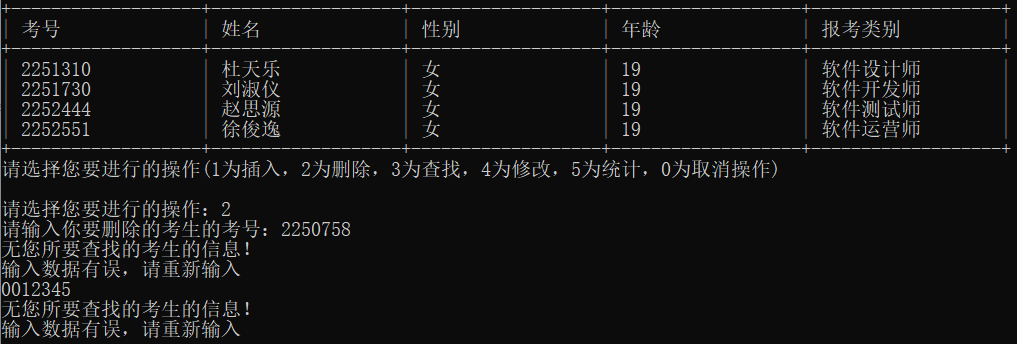
### 删除考号不存在

**测试用例：**2250758 / 0012345

**用例说明：**删除考号不存在

**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：**



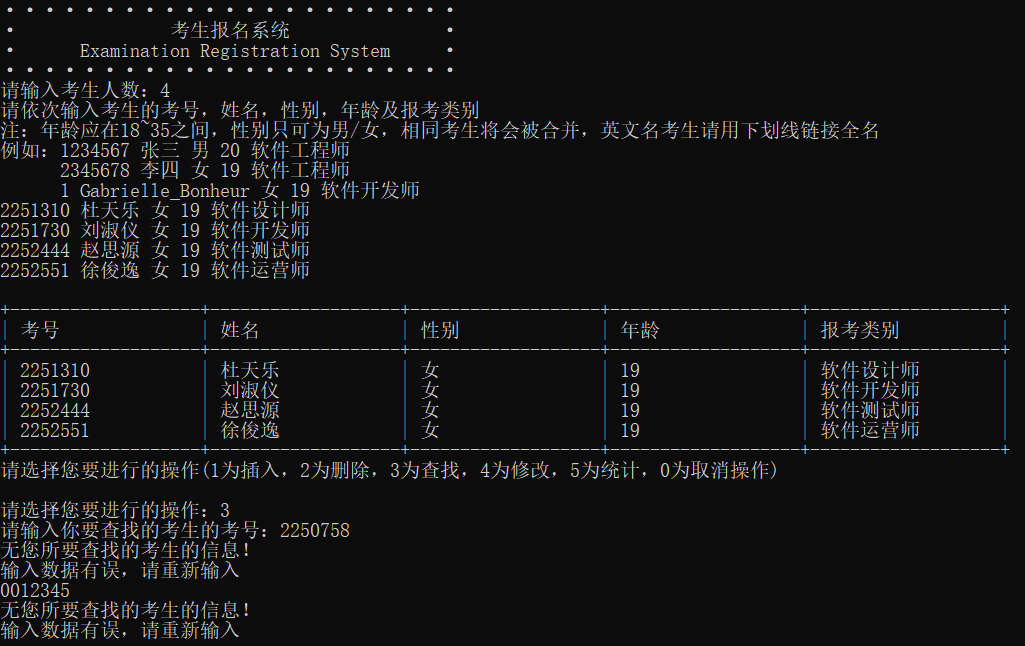
### 查找考号不存在

**测试用例：**2250758 / 0012345

**用例说明：**删除考号不存在

**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：**



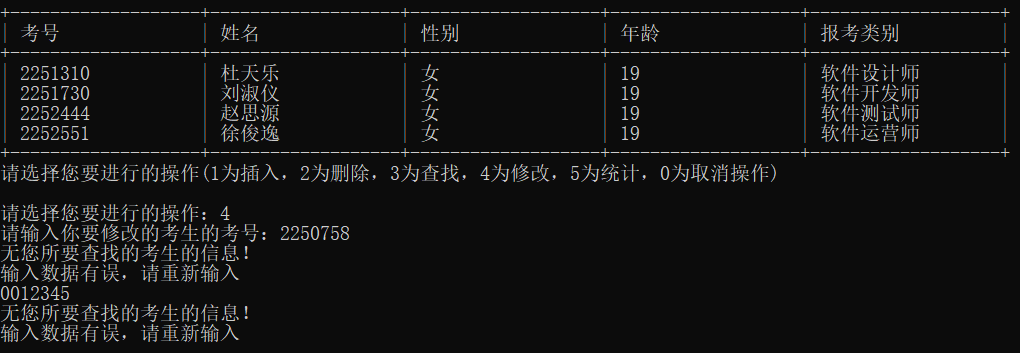
### 修改考号不存在

**测试用例：**2250758 / 0012345

**用例说明：**删除考号不存在

**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：**



## Linux环境测试

本项目使用条件编译解决Windows系统和Linux系统编译环境的差异，示例代码如下

// 区分系统

#ifdef \_WIN32

#include <conio.h>

#elif \_\_linux\_\_

#include <termios.h>

#include <unistd.h>

// 用于Linux的\_getch()的实现

int \_getch() {

struct termios oldt, newt;

int ch;

tcgetattr(STDIN\_FILENO, &oldt);

newt = oldt;

newt.c\_lflag &= ~(ICANON | ECHO);

tcsetattr(STDIN\_FILENO, TCSANOW, &newt);

ch = getchar();

tcsetattr(STDIN\_FILENO, TCSANOW, &oldt);

return ch;

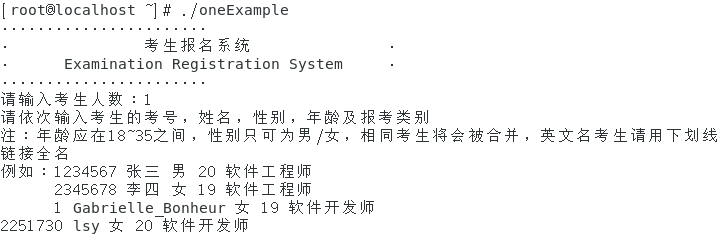
}

#endif

**编译命令：** g++’/mnt/hgfs/ShareFolder/01\_examination\_registration\_system.cpp' -o oneExample

**运行命令：** ./oneExample

**实验结果：**



1. **集成开发环境与编译运行环境**

Windows系统：Windows 11 x64

Windows集成开发环境：Microsoft Visual Studio 2022 (Release模式)

Windows编译运行环境：本项目适用于x86架构和x64架构

Linux系统：CentOS 7 x64