项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 赵子昱

学 号： 1951459

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](#_Toc59527341)

[1.1 背景分析 1](#_Toc59527342)

[1.2 功能分析 1](#_Toc59527343)

[2 设计 2](#_Toc59527344)

[2.1 数据结构设计 2](#_Toc59527345)

[2.2 类结构设计 2](#_Toc59527346)

[2.3 成员与操作设计 2](#_Toc59527347)

[2.4 系统设计 4](#_Toc59527348)

[3 实现 4](#_Toc59527349)

[3.1 初始化功能**Input()**的实现 4](#_Toc59527350)

[3.1.1 初始化功能代码 4](#_Toc59527351)

[3.1.2 初始化功能截屏示例 5](#_Toc59527352)

[3.2 插入功能**Insert(…)**的实现 5](#_Toc59527353)

[3.2.1 插入功能流程图 6](#_Toc59527354)

[3.2.2 插入功能代码 7](#_Toc59527355)

[3.2.3 插入功能截屏示例 7](#_Toc59527356)

[3.3 统计功能**Statistics()**的实现 8](#_Toc59527357)

[3.3.1 统计功能流程图 8](#_Toc59527358)

[3.3.2 统计功能代码 8](#_Toc59527359)

[3.3.3 统计功能截屏示例 9](#_Toc59527360)

[3.4 查找功能**Find(…)**的实现 9](#_Toc59527361)

[3.4.1 查找功能代码 10](#_Toc59527362)

[3.4.2 查找功能截图示例 10](#_Toc59527363)

[3.5 修改功能**Revise(…)**的实现 11](#_Toc59527364)

[3.5.1 修改功能代码 11](#_Toc59527365)

[3.5.2 修改功能截屏示例 11](#_Toc59527366)

[3.6 删除功能**Remove(…)**的实现 12](#_Toc59527367)

[3.6.1 删除功能代码 12](#_Toc59527368)

[3.6.2 删除功能截屏示例 13](#_Toc59527369)

[3.7 创建一个考生对象**GetStudent()**的实现 13](#_Toc59527370)

[3.7.1 创建一个考生对象功能代码 13](#_Toc59527371)

[3.8 输出考生信息**Show()**的实现 14](#_Toc59527372)

[3.8.1 输出考生信息功能代码 14](#_Toc59527373)

[3.9 主函数的实现 14](#_Toc59527374)

[3.9.1 主函数流程图 15](#_Toc59527375)

[3.9.2 主函数代码 15](#_Toc59527376)

[3.9.3 主函数截屏示例 17](#_Toc59527377)

[4 测试 18](#_Toc59527378)

[4.1 功能测试 18](#_Toc59527379)

[4.1.1 插入功能测试 18](#_Toc59527380)

[4.1.2 删除功能测试 18](#_Toc59527381)

[4.1.3 查找功能测试 19](#_Toc59527382)

[4.1.4 修改功能测试 19](#_Toc59527383)

[4.1.5 统计功能测试 20](#_Toc59527384)

[4.2 边界测试 21](#_Toc59527385)

[4.2.1 初始化无输入数据 21](#_Toc59527386)

[4.2.2 删除第一个考生 21](#_Toc59527387)

[4.2.3 删除后链表为空 22](#_Toc59527388)

[4.3 出错测试 22](#_Toc59527389)

[4.3.1 考生人数错误 22](#_Toc59527390)

[4.3.2 操作码错误 23](#_Toc59527391)

[4.3.3 插入位置不存在 24](#_Toc59527392)

[4.3.4 删除考号不存在 24](#_Toc59527393)

[4.3.5 查找考号不存在 25](#_Toc59527394)

[4.3.6 修改考号不存在 26](#_Toc59527395)

# 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。因为报名系统的一些操作涉及到链表的遍历，因此还用到了链表的迭代器。

## 2.2 类结构设计

在项目中，定义了考试报名系统类，封装了报名系统的各个操作与存储的信息；还定义了学生类，封装了每个考生的信息；还定义了科目类，封装了科目名称与报名人数，用于统计各科目的人数。

## 2.3 成员与操作设计

**考试报名系统类（ExamRegistrationSystem** **）**

1. **class** ExamRegistrationSystem {
2. **private**:
3. List<Student> Students;
4. //学生信息
6. **public**:
7. ExamRegistrationSystem();
9. **int** Input();
10. //输入考生信息，初始化报名系统
12. **int** Insert();
13. //插入
15. Student \* Find(string num);
16. //查找
18. **int** Revise(string num);
19. //修改
21. Student \* Remove(string num);
22. //删除
24. **void** Show();
25. //输出当前所有考生信息
27. List<Subject> \*Statistics();
28. //统计
30. Student \*GetStudent();
31. //输入一个考生的信息并返回指向该考生的指针
32. };

**考生类（Student）**

1. **class** Student {
2. **friend** **class** ExamRegistrationSystem;
3. string m\_number;
4. //考号
6. string m\_name;
7. //姓名
9. string m\_sex;
10. //性别
12. **int** m\_age;
13. //年龄
15. string m\_type;
16. //报考类型
18. **public**:
19. Student() {};
20. Student(string number, string name, string sex, **int** age, string type) :m\_number(number), m\_name(name), m\_sex(sex), m\_age(age), m\_type(type) {}
21. };

**报考类型类（Subject）**

1. **class** Subject {
2. **friend** **class** ExamRegistrationSystem;
3. **public**:
4. Subject() {};
5. Subject(string sub) :m\_subject(sub), m\_count(1) {}
6. **int** operator==(Subject &sub) { **return** sub.m\_subject == m\_subject; }
7. **void** Show() { cout << left << setw(15) << m\_subject  << m\_count << endl;  }
8. **private**:
9. string m\_subject;
10. //报考类型名称
12. **int** m\_count;
13. //报考人数
14. };

## 2.4 系统设计

系统首先创建一个报名系统对象，自动调用其构造函数来输入考生信息完成对链表的初始化，然后根据用户所输入的操作码调用相应的函数实现相应的操作。

# 3 实现

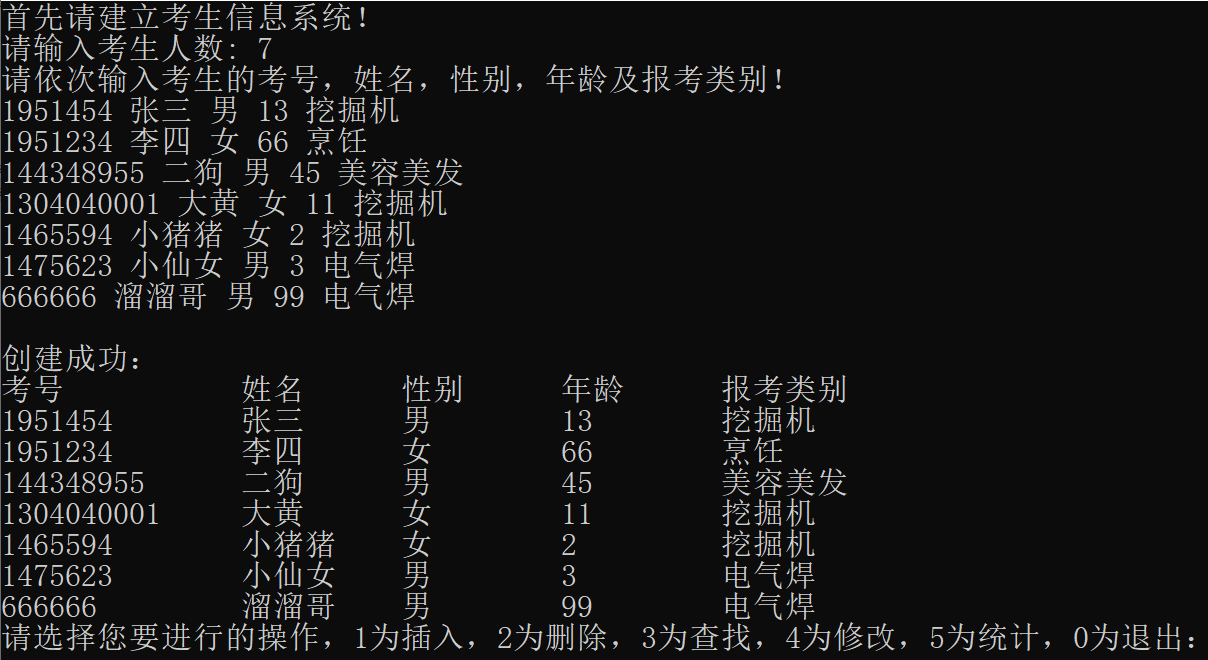
## 3.1 初始化功能**Input()**的实现

* 初始化功能实现考生人数的输入，并逐条输入考生信息
* 被**ExamRegistrationSystem()**函数调用

### 3.1.1 初始化功能代码

1. **int** ExamRegistrationSystem::Input() {
2. **int** count;
3. cout << "请输入考生人数: ";
4. cin >> count;
5. **while** (!cin.good() || count <= 0) {
6. cin.clear();
7. cin.ignore(1024, '\n');
8. cout << "非法输入，请重新输入 ：";
9. cin >> count;
10. }
11. **if** (count <= 0) {
12. cout << "人数应大于0！" << endl;
13. **return** 0;
14. }
15. Student stu;
16. cout << "请依次输入考生的考号，姓名，性别，年龄及报考类别！" << endl;
17. **while** (count--) {
18. cin >> stu.m\_number >> stu.m\_name >> stu.m\_sex >> stu.m\_age >> stu.m\_type;
19. **while** (!cin.good()) {
20. cin.clear();
21. cin.ignore(1024, '\n');
22. cout << "非法输入，请重新输入!"<<endl;
23. cout << "请依次输入考生的考号，姓名，性别，年龄及报考类别！" << endl;
24. cin >> stu.m\_number >> stu.m\_name >> stu.m\_sex >> stu.m\_age >> stu.m\_type;
25. }
26. Students.Insert(stu, Students.Length());
27. }
28. cout << endl;
29. cout << "创建成功：" << endl;
30. **return** 1;
31. }

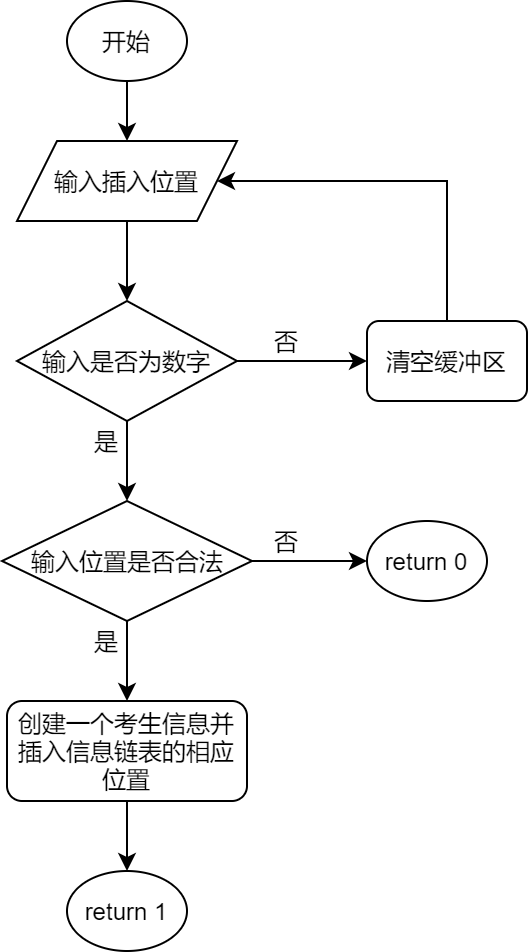
### 3.1.2 初始化功能截屏示例



## 3.2 插入功能**Insert(…)**的实现

* 实现向考试报名系统中指定位置插入一个考生信息
* 被主函数调用

### 3.2.1 插入功能流程图



### 

### 3.2.2 插入功能代码

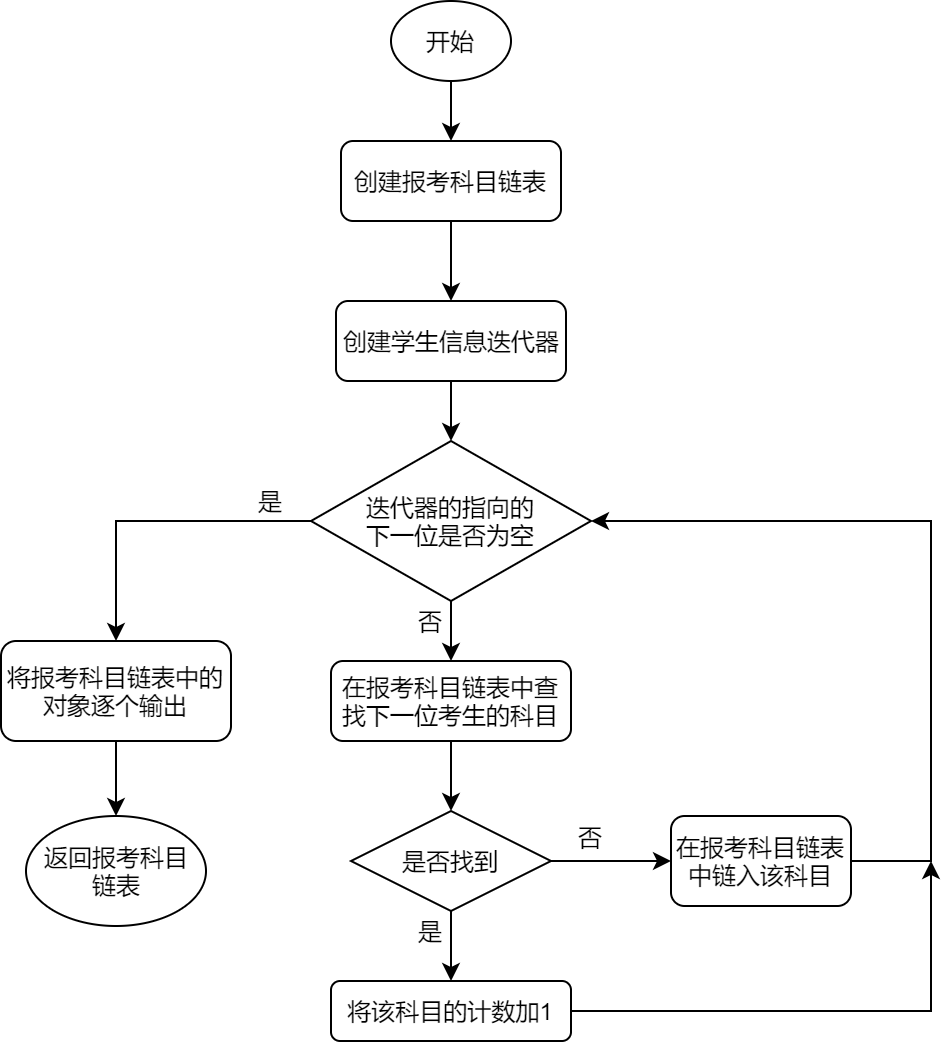
1. **int** ExamRegistrationSystem::Insert() {
2. **int** i;
3. cout << "请输入您要插入的位置：";
4. cin >> i;
5. **while** (!cin.good()) {
6. cin.clear();
7. cin.ignore(1024, '\n');
8. cout << "非法输入，请重新输入 ：";
9. cin >> i;
10. }
11. i -= 1;
12. **if** (i < 0 || i>Students.Length()) {
13. cin.ignore(1024, '\n');
14. cout << "非法的位置！" << endl;
15. **return** 0;
16. }
17. Student \*stu = GetStudent();
18. Students.Insert(\*stu, i);
19. cout << "插入成功" << endl;
20. **return** 1;
21. }

### 3.2.3 插入功能截屏示例

## 3.3 统计功能**Statistics()**的实现

* 统计功能实现统计各个报考科目的报考人数并输出
* 构建了一个报考科目对象链表，保存不同科目与对应报考人数
* 被主函数调用

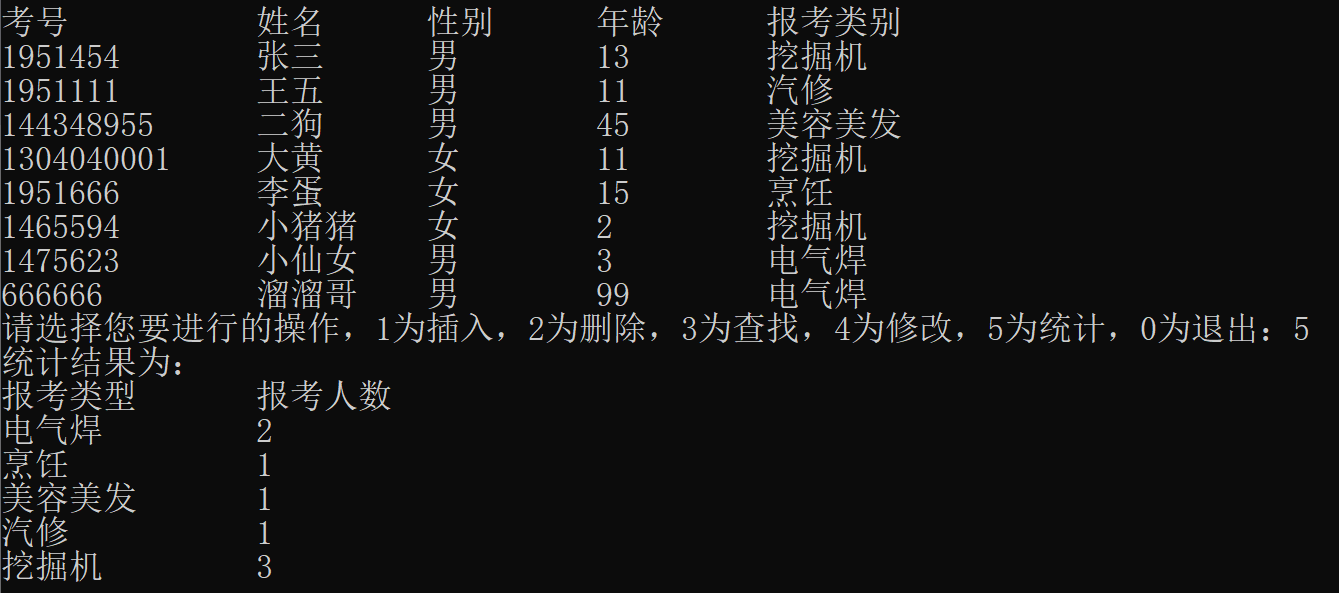
### 3.3.1 统计功能流程图



### 3.3.2 统计功能代码

1. List<Subject> \*ExamRegistrationSystem::Statistics() {
2. List<Subject> \*p = **new** List<Subject>;   //报考科目链表
3. ListIterator<Student> iter(Students); //学生信息迭代器
4. **while** (iter.NextNotNull()) {
5. //遍历考生信息链表，找出所有不同科目，链入报考科目链表并计数
6. string type = iter.Next()->m\_type;
7. Subject sub(type);
8. Subject \*pp = p->Find(sub);
9. **if** (pp) {
10. (pp->m\_count)++;
11. }
12. **else** {
13. p->Insert(sub, 0);
14. }
15. }
16. **if** (p->NotEmpty()) {
17. //输出结果
18. cout << "统计结果为：" << endl;
19. cout << left << setw(15) << "报考类型" << "报考人数" << endl;
20. ListIterator<Subject> iterSub(\*p);
21. **while** (iterSub.NextNotNull()) {
22. iterSub.Next()->Show();
23. }
24. **return** p;
25. }
26. **return** NULL;
27. }

### 3.3.3 统计功能截屏示例



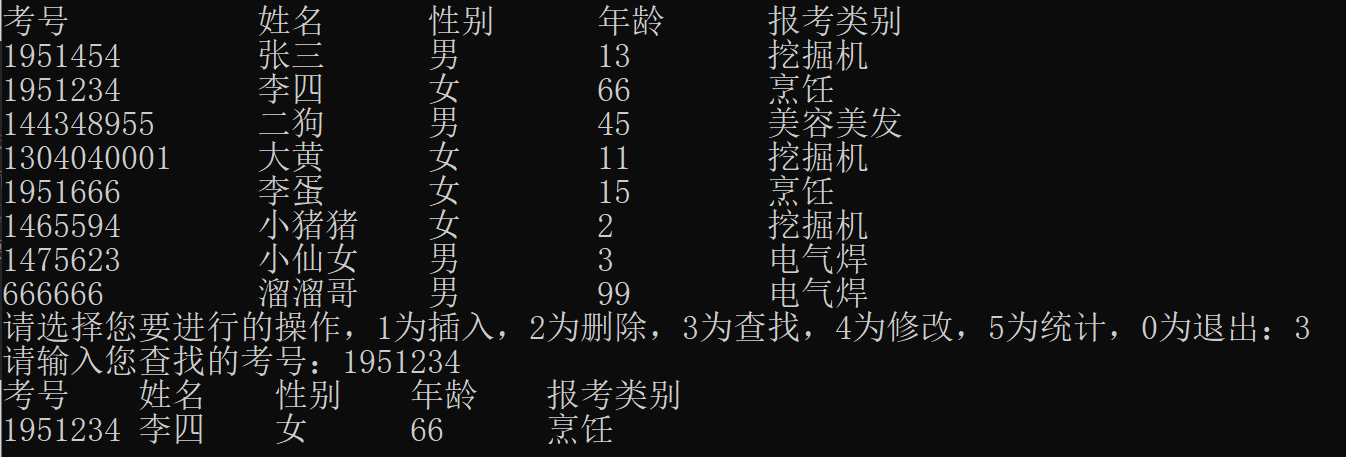
## 3.4 查找功能**Find(…)**的实现

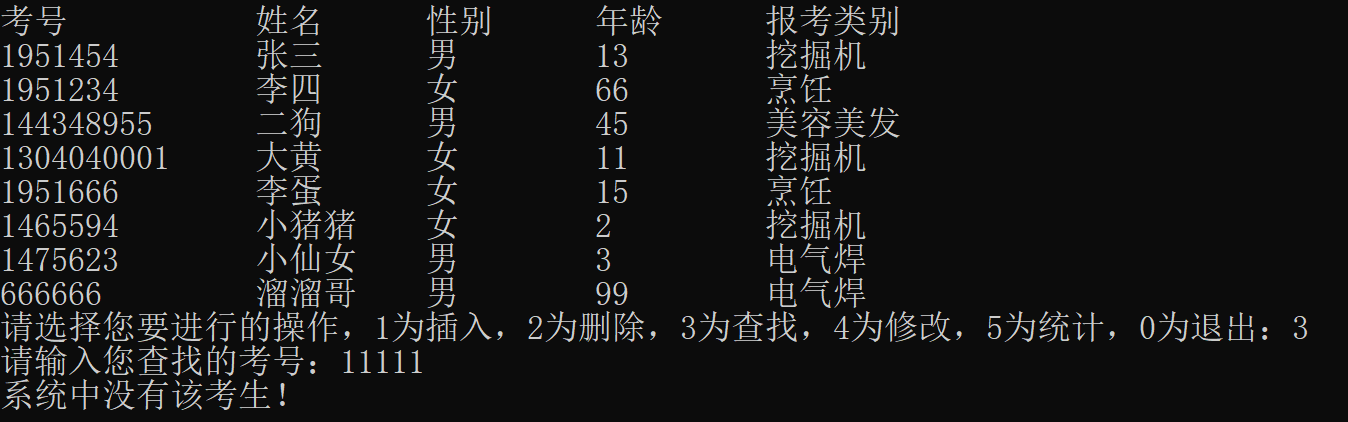
* num为考号
* 查找成功是输出该考生信息，并返回该考生信息
* 查找失败给出提示
* 被**Revise(…)**函数、主函数调用

### 3.4.1 查找功能代码

1. Student \*ExamRegistrationSystem::Find(string num) {
2. ListIterator<Student> iter(Students);
3. Student \*p;
4. **while** (p = iter.Next()) {
5. **if** (num == p->m\_number) {
6. cout << "考号" << '\t' << "姓名" << '\t' << "性别" << '\t' << "年龄" << '\t' << "报考类别" << endl;
7. cout << p->m\_number << '\t' << p->m\_name << '\t' << p->m\_sex << '\t' << p->m\_age << '\t' << p->m\_type << endl;
8. **return** p;
9. }
10. }
11. cout << "系统中没有该考生！" << endl;
12. **return** NULL;
13. }

### 3.4.2 查找功能截图示例





## 3.5 修改功能**Revise(…)**的实现

* num为考号
* 首先调用Find()函数进行查找，查找成功则修改，不成功给出提示
* 修改成功返回1，不成功返回0
* 被主函数调用

### 3.5.1 修改功能代码

1. **int** ExamRegistrationSystem::Revise(string num) {
2. Student \*p = Find(num);
3. **if** (p) {
4. cout << "请依次输入修改后考生的考号，姓名，性别，年龄及报考类别！" << endl;
5. cin >> p->m\_number >> p->m\_name >> p->m\_sex >> p->m\_age >> p->m\_type;
6. cout << "修改成功" << endl;
7. **return** 1;
8. }
9. **return** NULL;
10. }

### 3.5.2 修改功能截屏示例

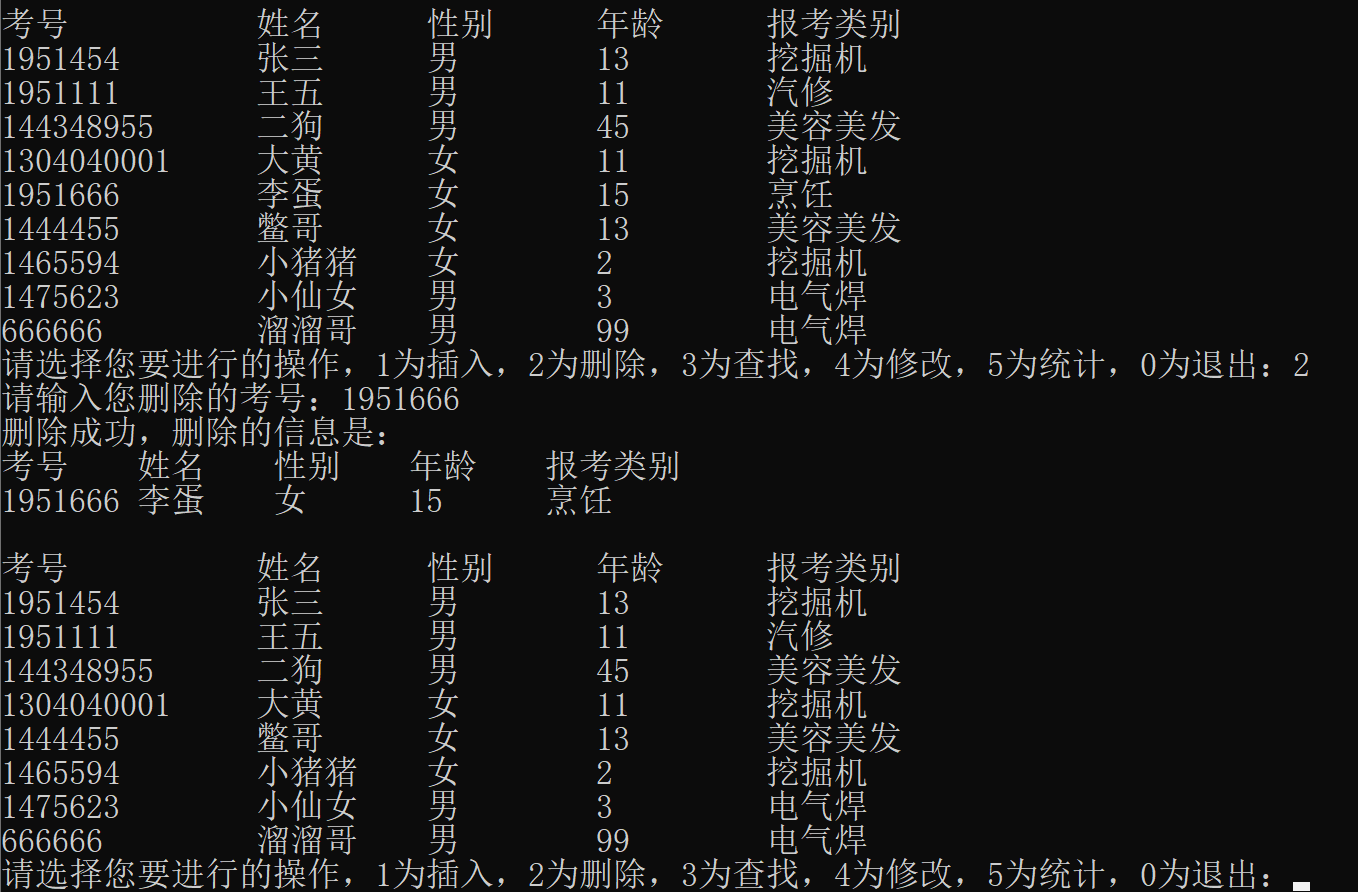
## 3.6 删除功能**Remove(…)**的实现

* 删除指定考号的考生
* num为考号
* 被主函数调用

### 3.6.1 删除功能代码

1. Student \*ExamRegistrationSystem::Remove(string num) {
2. ListIterator<Student> iter(Students);
3. Student \*p;
4. **int** i = 0;
5. **while** (p = iter.Next()) {
6. **if** (num == p->m\_number) {
7. cout << "删除成功，删除的信息是：" << endl;
8. cout << "考号" << '\t' << "姓名" << '\t' << "性别" << '\t' << "年龄" << '\t' << "报考类别" << endl;
9. cout << p->m\_number << '\t' << p->m\_name << '\t' << p->m\_sex << '\t' << p->m\_age << '\t' << p->m\_type << endl;
10. **return** Students.Remove(i);
11. }
12. i++;
13. }
14. cout << "系统中没有该考生！" << endl;
15. **return** NULL;
16. }

### 3.6.2 删除功能截屏示例



## 3.7 创建一个考生对象**GetStudent()**的实现

* 返回一个考生对象指针
* 被Insert()函数调用

### 3.7.1 创建一个考生对象功能代码

1. Student \*ExamRegistrationSystem::GetStudent() {
2. string number;
3. string name;
4. string sex;
5. **int** age;
6. string type;
7. cout << "请依次输入考生的考号，姓名，性别，年龄及报考类别：" << endl;
8. cin >> number >> name >> sex >> age >> type;
9. Student \*stu = **new** Student(number, name, sex, age, type);
10. **return** stu;
11. }

## 3.8 输出考生信息**Show()**的实现

* 利用一个迭代器，遍历考生信息链表，并将其输出
* 被**Statistics()、**主函数调用

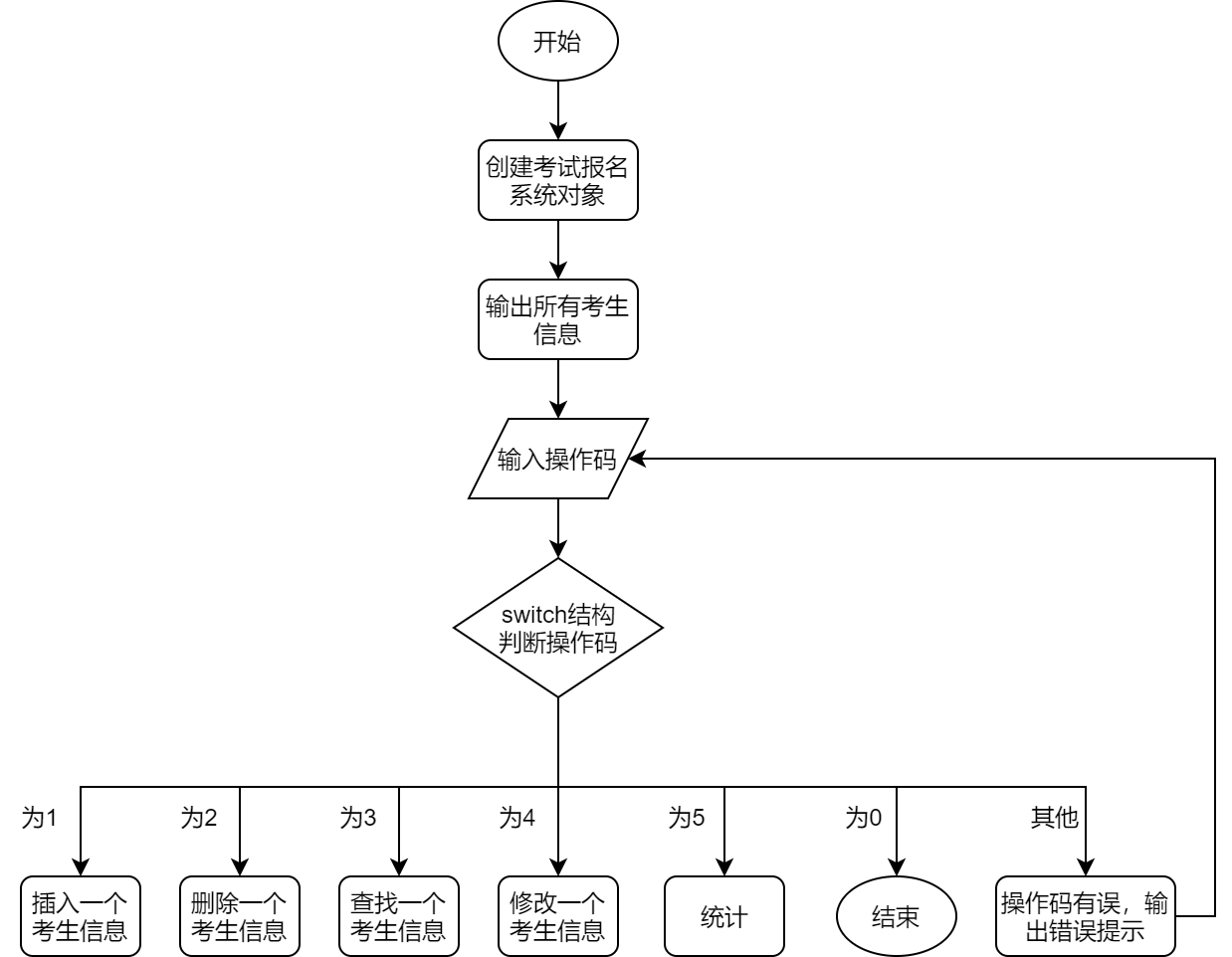
### 3.8.1 输出考生信息功能代码

1. **void** ExamRegistrationSystem::Show() {
2. ListIterator<Student> iter(Students);
3. Student \*p;
4. cout << left<<setw(15) << "考号" << setw(10) << "姓名" << setw(10) << "性别" << setw(10) << "年龄" << setw(10) << "报考类别" << endl;
5. **while** (p = iter.Next()) {
6. cout << left<<setw(15) << p->m\_number << setw(10) << p->m\_name << setw(10) << p->m\_sex << setw(10) << p->m\_age << setw(10) << p->m\_type << endl;
7. }
8. }

## 3.9 主函数的实现

主函数中，首先创建一个报名系统对象，自动调用其构造函数来输入考生信息完成对链表的初始化，然后根据用户所输入的操作码调用相应的函数实现相应的操作。

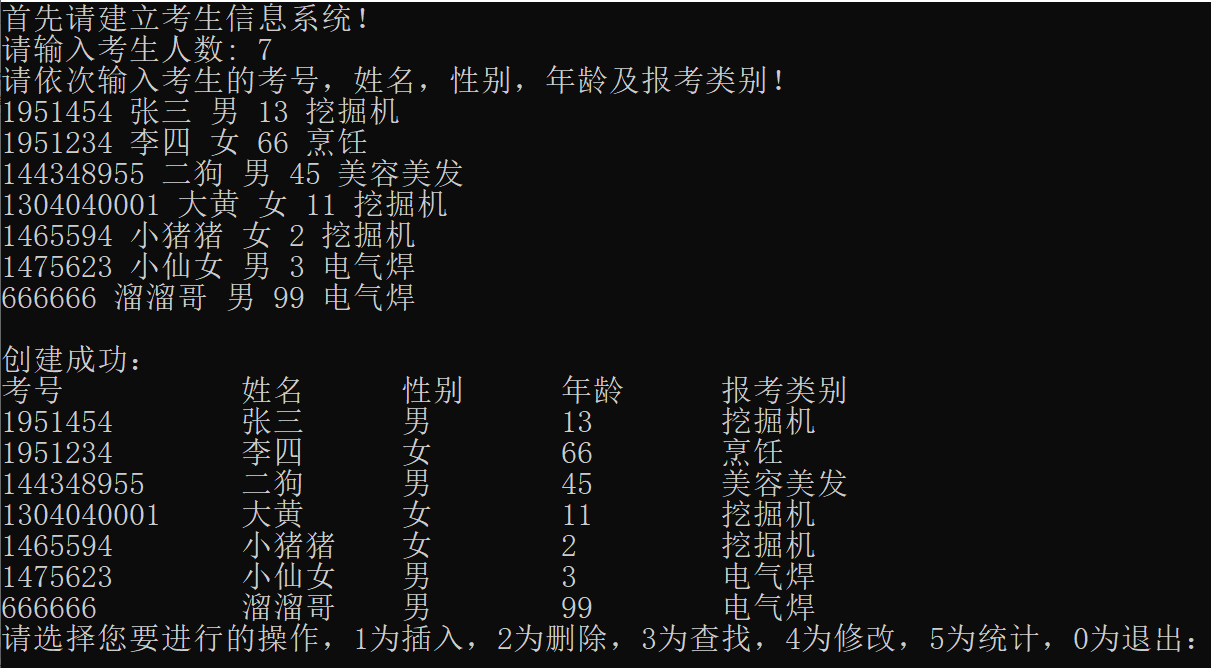
### 3.9.1 主函数流程图



### 3.9.2 主函数代码

1. **int** main() {
2. ExamRegistrationSystem sys;
3. **int** operation;
4. **while** (1) {
5. sys.Show();
6. 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为退出：";
7. cin >> operation;
8. **while** (!cin.good()) {
9. cin.clear();
10. cin.ignore(1024, '\n');
11. cout << "非法输入，请重新输入 ：";
12. cin >> operation;
13. }
14. **switch** (operation) {
15. **case** 1: {
16. sys.Insert(); }
17. **break**;
18. **case** 2: {
19. string a;
20. cout << "请输入您删除的考号：";
21. cin >> a;
22. sys.Remove(a); }
23. **break**;
24. **case** 3: {
25. string a;
26. cout << "请输入您查找的考号：";
27. cin >> a;
28. sys.Find(a); }
29. **break**;
30. **case** 4: {
31. string a;
32. cout << "请输入您修改的考号：";
33. cin >> a;
34. sys.Revise(a); }
35. **break**;
36. **case** 5: {
37. sys.Statistics(); }
38. **break**;
39. **case** 0:
40. **break**;
41. **default**:
42. cout << "输入有误！" << endl;
43. }
44. **if** (operation == 0)**break**;
45. cout << endl;
46. }
47. **return** 0;
48. }

### 3.9.3 主函数截屏示例



# 4 测试

## 4.1 功能测试

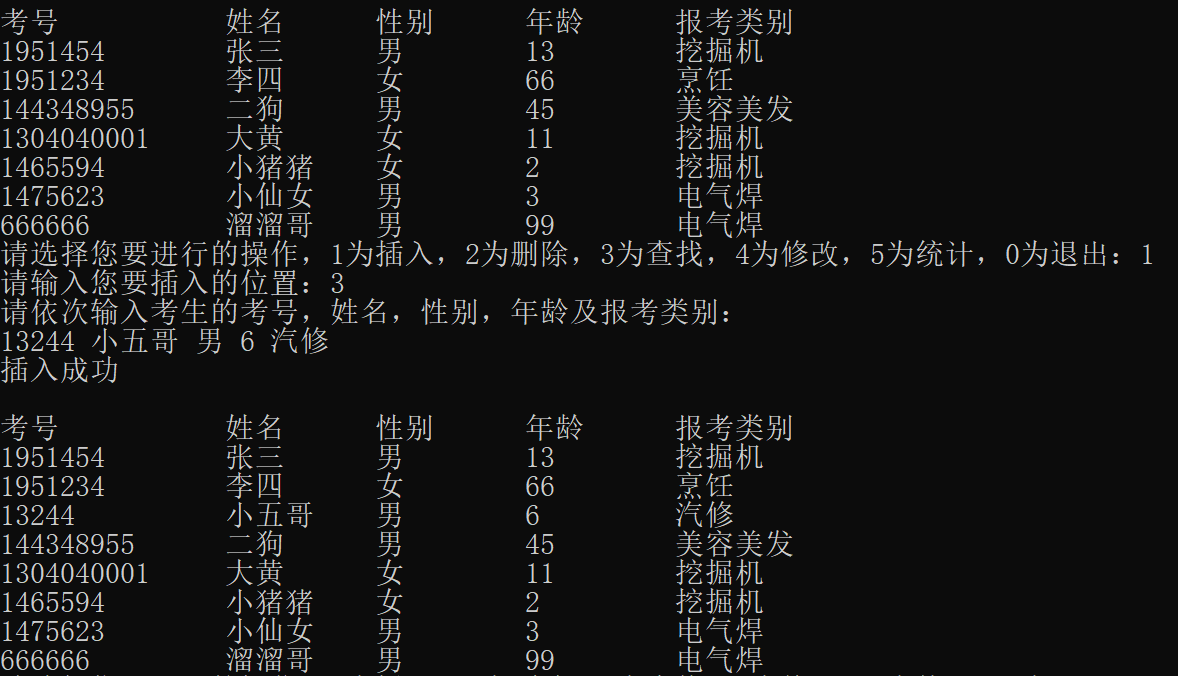
### 4.1.1 插入功能测试

**测试用例**：

3

13244 小五哥 男 6 汽修

**实验结果**

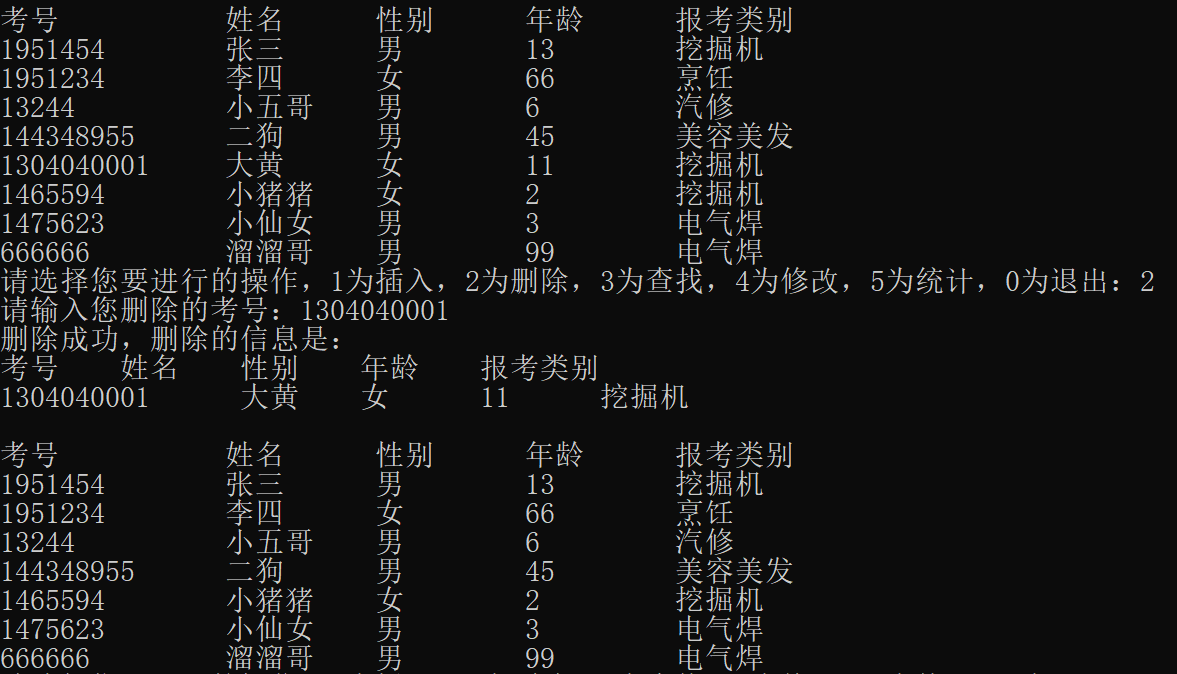


### 4.1.2 删除功能测试

**测试用例：**

删除考号为1304040001的考生

**实验结果：**

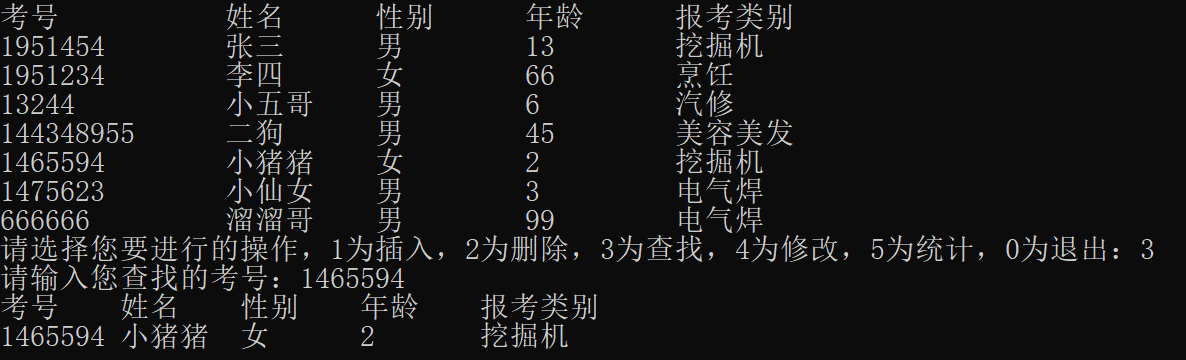


### 4.1.3 查找功能测试

**测试用例：**

查找考号为1465594的考生

**实验结果：**

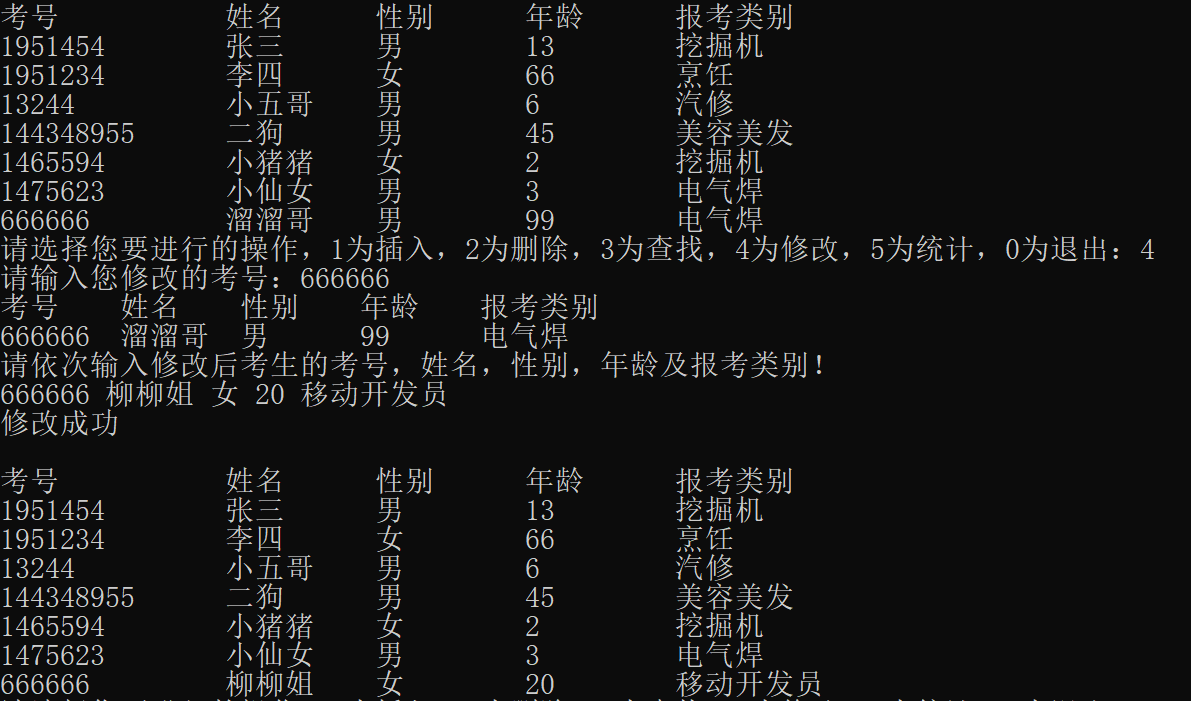


### 4.1.4 修改功能测试

**测试用例：**

将考号666666修改为性别女，年龄20，报考种类移动开发员。

**实验结果：**

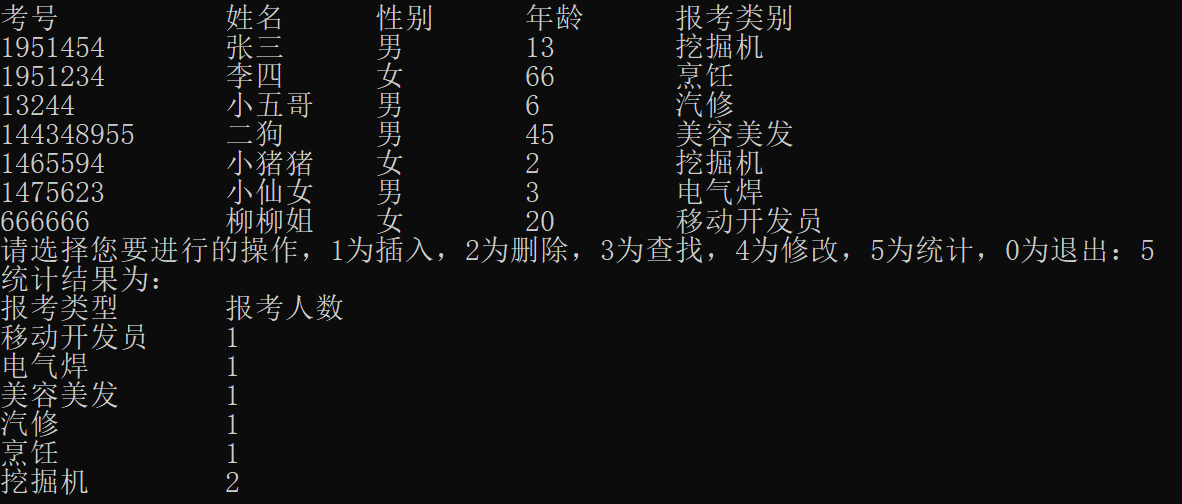


### 4.1.5 统计功能测试

**测试用例：**

统计当前数据

**实验结果：**



## 4.2 边界测试

### 4.2.1 初始化无输入数据

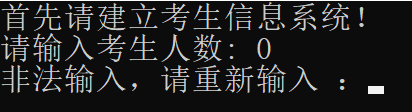
**测试用例：**

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 删除第一个考生

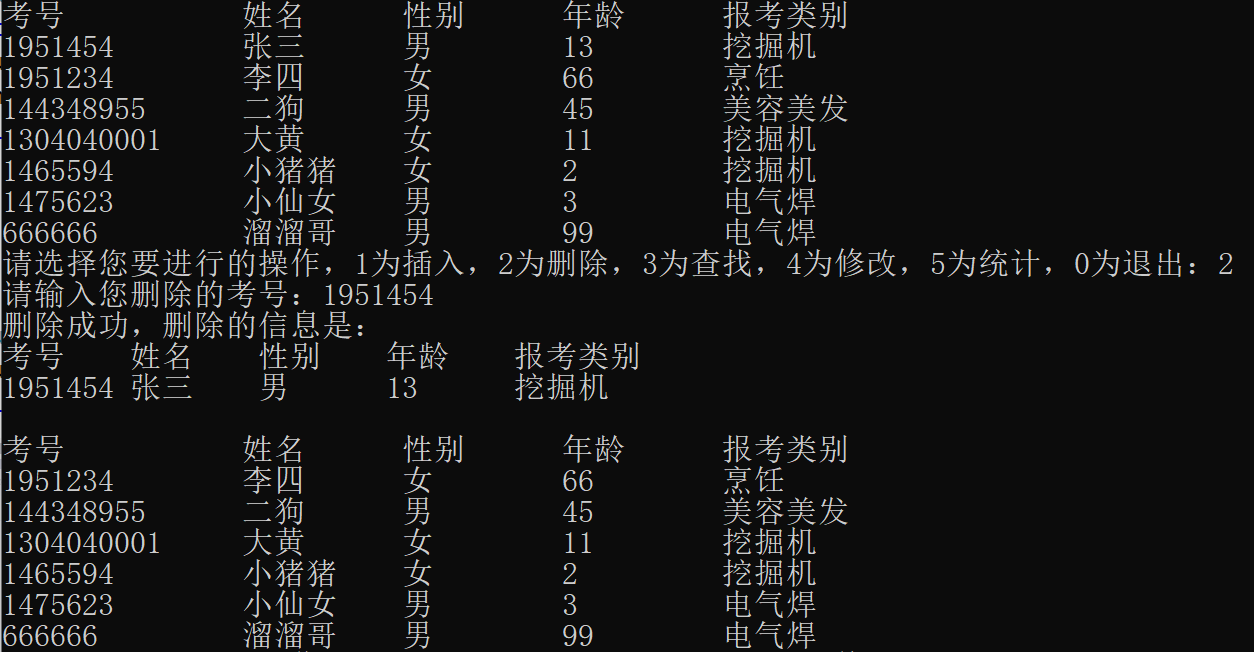
**测试用例：**

删除1951459

**预期结果：**

程序正常运行，不崩溃。

**实验结果：**

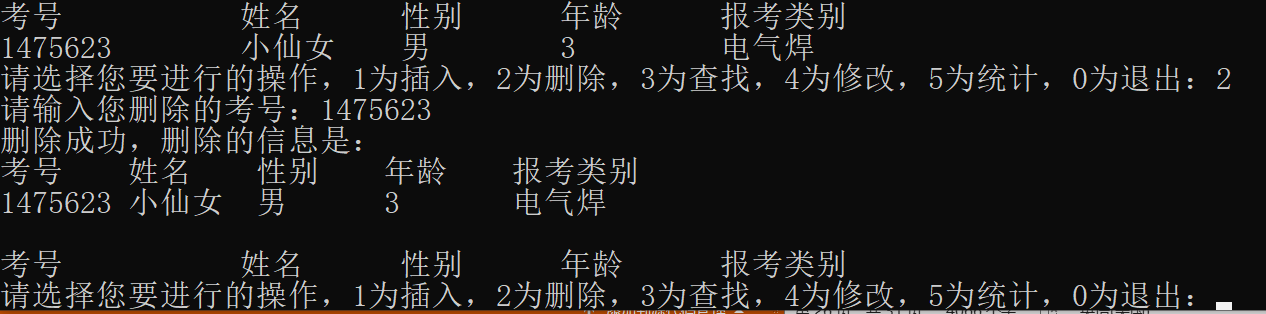


### 4.2.3 删除后链表为空

**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

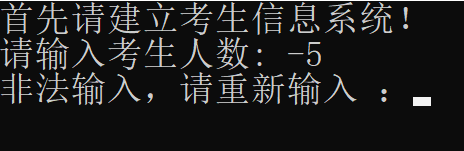
**测试用例：**

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

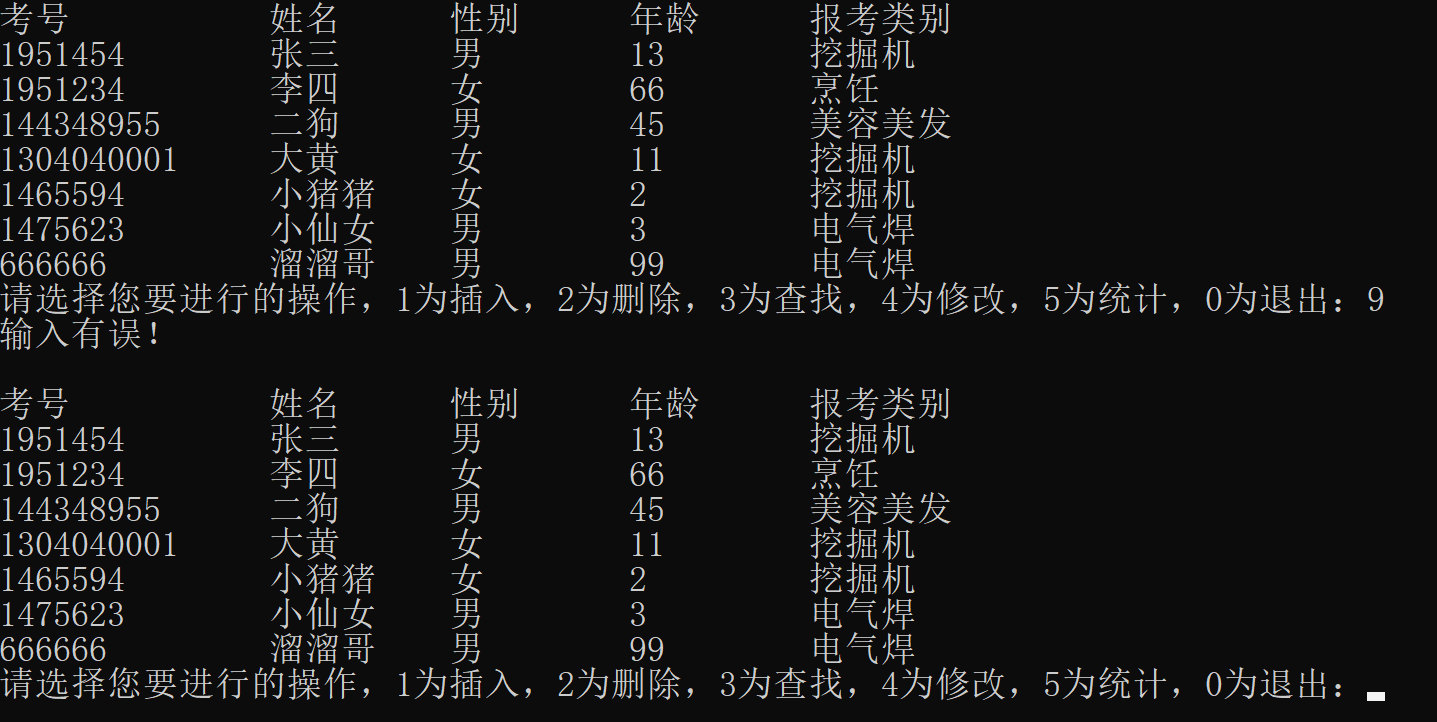
**测试用例：**

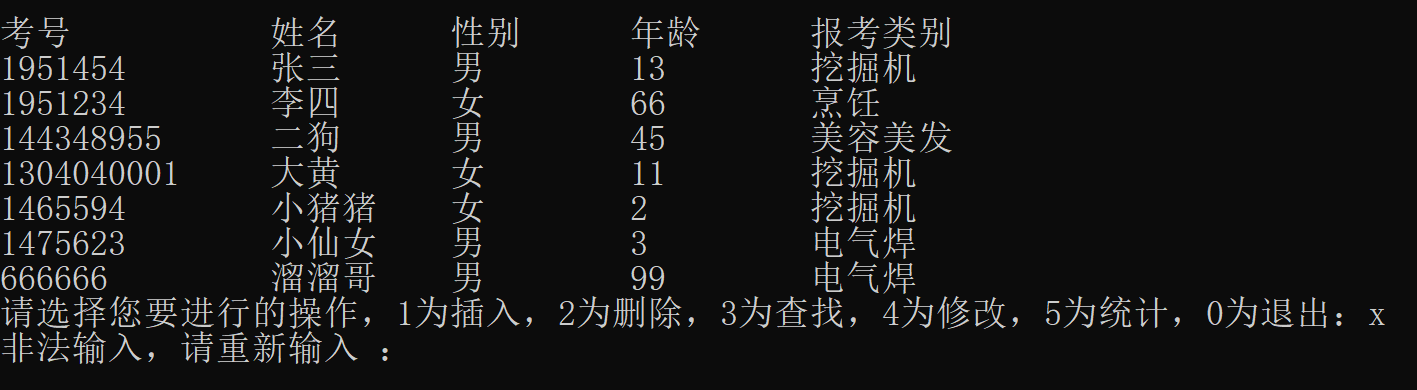
输入操作码错误

**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

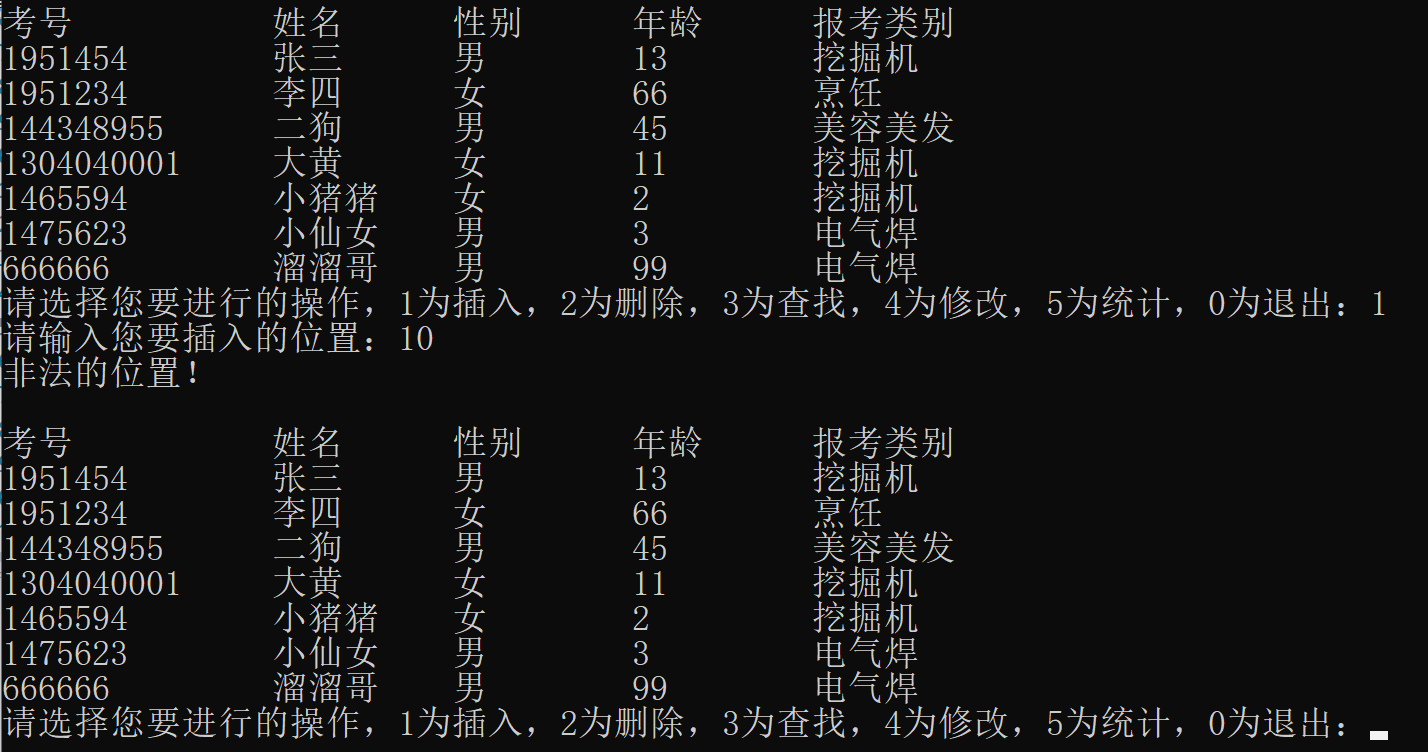
**测试用例：**

链表里有7条记录，向链表的第10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

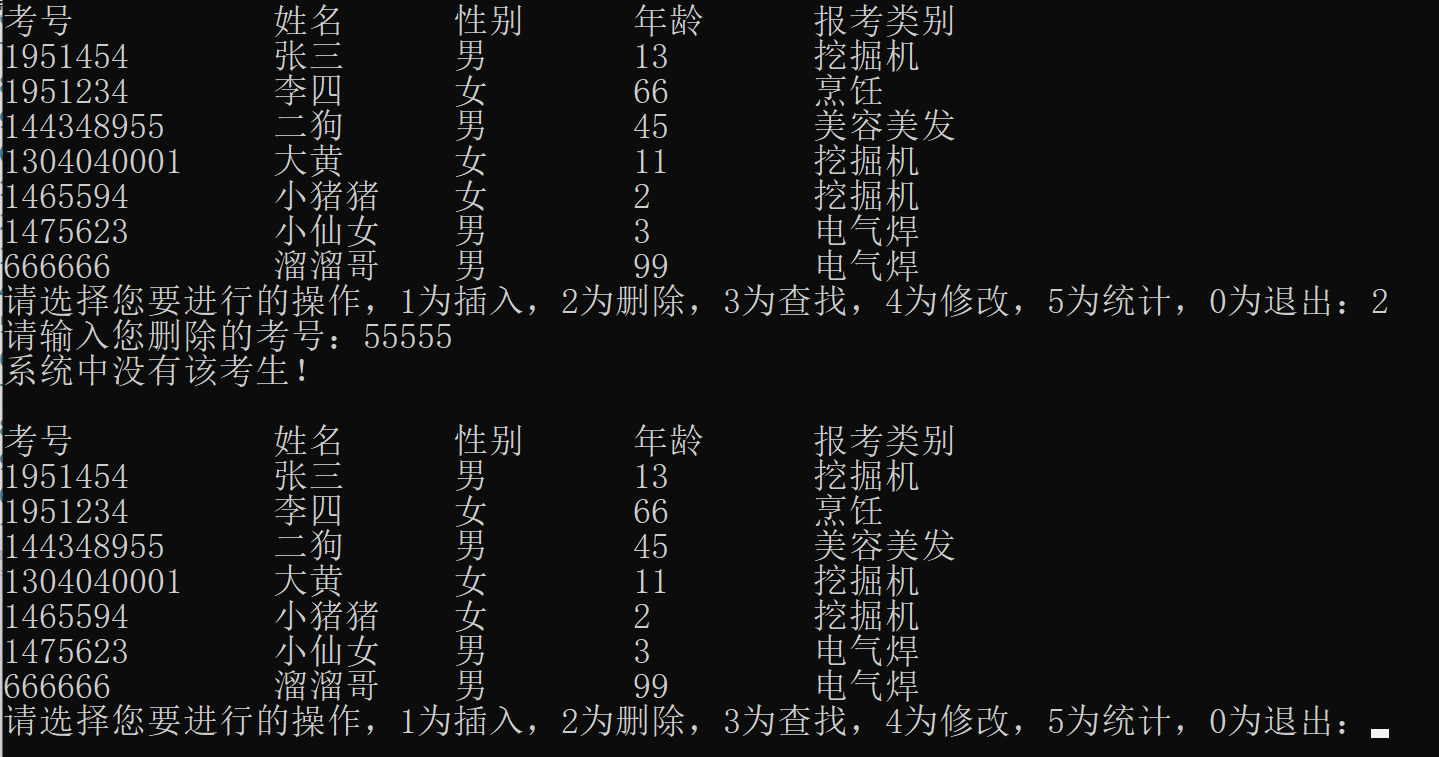
**测试用例：**

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

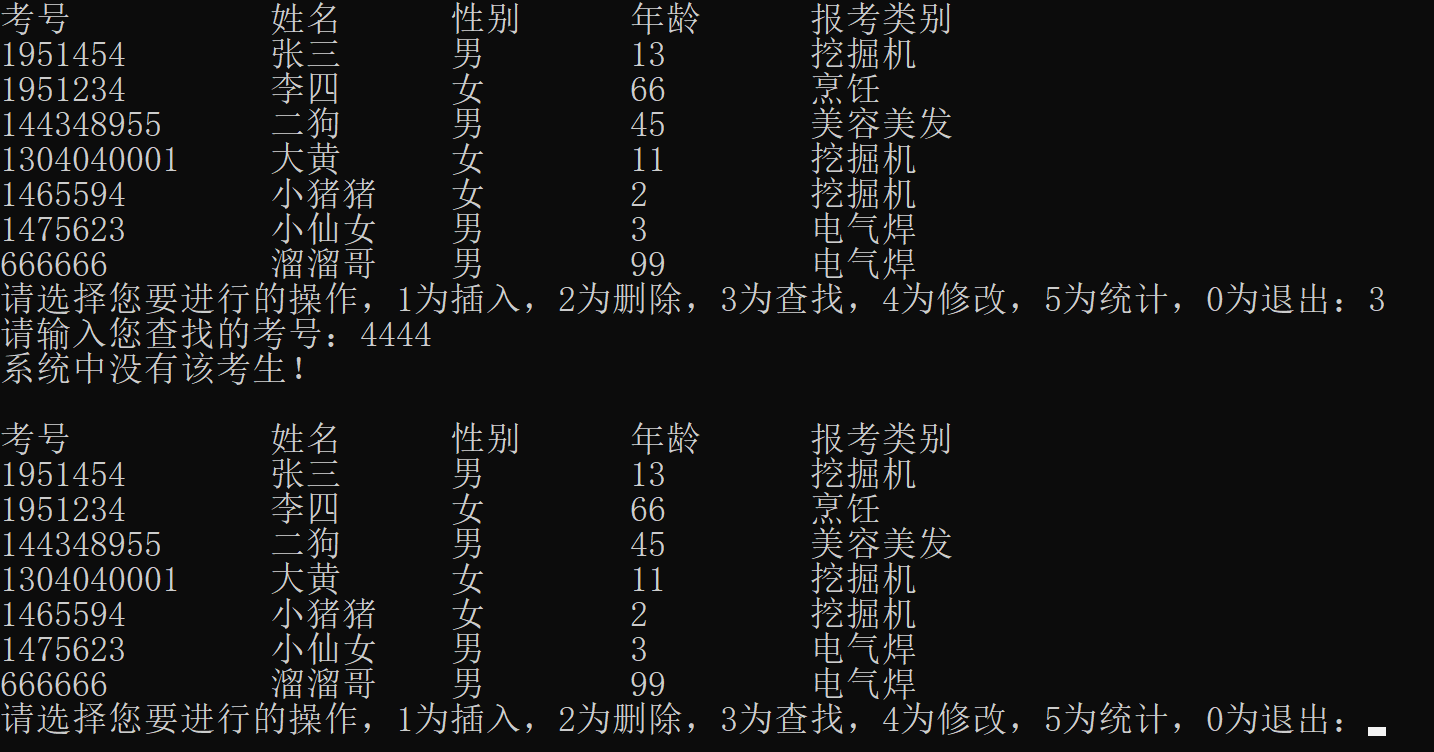
**测试用例：**

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：**

