考试报名系统

作 者 姓 名： 王凌

学 号： 1951504

指 导 教 师： 张颖

学院、 专业： 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc8729)

[1.1 项目简介 1](#_Toc18999)

[1.2 项目功能要求 1](#_Toc25966)

[2 设计 2](#_Toc10497)

[2.1 链式存储结构的设计 2](#_Toc28373)

[2.2 LinkList类结构的设计 2](#_Toc15138)

[2.3 Student类结构设计 3](#_Toc20111)

[2.4 主程序设计 3](#_Toc30263)

[3 实现 4](#_Toc14501)

[3.1 插入功能的实现 4](#_Toc26522)

[3.1.1 链表头部位置的插入 4](#_Toc20655)

[3.1.2 链表中间与尾部位置的插入 5](#_Toc27009)

[3.1.3 链表插入中不合理情况的判断 5](#_Toc16647)

[3.2 删除功能的实现 7](#_Toc14838)

[3.2.1 链表为空的情况 7](#_Toc24664)

[3.2.2 删除头部节点 8](#_Toc19092)

[3.2.3 删除中间结点与尾结点 8](#_Toc236)

[3.3 修改功能的实现 9](#_Toc5937)

[3.4 查找功能的实现 9](#_Toc7976)

[3.5 统计功能的实现 10](#_Toc737)

[3.6 总体系统的实现 10](#_Toc4366)

[3.6.1 总体系统描述 10](#_Toc30189)

[3.6.2 总体系统核心代码 10](#_Toc9929)

[4 测试 13](#_Toc3260)

[4.1 功能测试 13](#_Toc27780)

[4.1.1 建立功能测试 13](#_Toc26296)

[4.1.2 插入功能测试 13](#_Toc35)

[4.1.3 删除功能测试 14](#_Toc23997)

[4.1.4 查找功能测试 14](#_Toc16787)

[4.1.5 修改功能测试 15](#_Toc29071)

[4.1.6 统计功能测试 15](#_Toc6863)

[4.2 边界测试 17](#_Toc20582)

[4.2.1 初始化输入人数为0 17](#_Toc17608)

[4.2.2 删除头结点 17](#_Toc27808)

[4.2.3 删除后链表为空 18](#_Toc22735)

[4.2.4 输入考生人数为负数 18](#_Toc20743)

[4.2.5 输入的插入位置不正确 18](#_Toc14915)

[4.2.6 表中没有输入的id 20](#_Toc12372)

# 1 分析

## 1.1 项目简介

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。本项目是对考试报名管理的简单模拟，用控制台选项的选择方式完成下列功能：输入考生信息；输出考生信息；查询考生信息；添加考生信息；修改考生信息；删除考生信息。

## 1.2 项目功能要求

本项目的实质是完成对考生信息的建立，查找，插入，修改，删除等功能。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。（建议采用链表实现）

# 2 设计

## 2.1 链式存储结构的设计

因为要进行大量的删减和增加操作，如果用顺序存储结构的话，会有大量的移动操作，大大增加了时间复杂度，所以我们采取链式存储结构。由于主要操作只有增、删、改、展示，为了实现简易和程序简洁，所以只需要用到一个链表的头指针变量。

## 2.2 LinkList类结构的设计

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。所以我首先用类模板LinkList存储一条由Node结点类构成的链表，之后再用LinkList<Student>模板类来存储所有学生信息。LinkList中，通过insert()，pop()，change()，search()，display()等函数，来进行对应的操作，最终实现了对学生信息表的增删改查等功能。

**LinkList类的成员与操作设计如下：**

**私有成员：**

Node<T> \*first;//头指针

**公有操作：**

LinkList():first(NULL){}

//自定义构造函数

void insert(int pos,T t)

//在第pos位前插入

void pop(const string &id0)

//删除考号为id0的结点

void change(const string &id,int choice,string value)

//改变考号为id的第choice个信息，改成value的值

void search(const string &id)

//查找并展示相关id

void display()

//全部展示

## 2.3 Student类结构设计

Student类含有该描述学生的几个成员变量：准考证号，姓名，性别，年龄和报考类别。为了方便读写，我将所有的成员变量变为公有成员变量，并重载了输入输出。

**Student类的成员与操作设计如下：**

**公有成员：**

string id;//考号

string name;//姓名

string gender;//性别

int age;//年龄

string Application;//报考类别英文为Application\_Type

**公有操作：**

Student(){}

Student(string &id0,string &name0,string &gender0,int age0,string &Application0)

//自定义赋值构造函数

friend istream &operator>>(istream &input,Student &student)

//重载输入

friend ostream &operator<<(ostream &output, const Student &student)

//重载输出

## 2.4 主程序设计

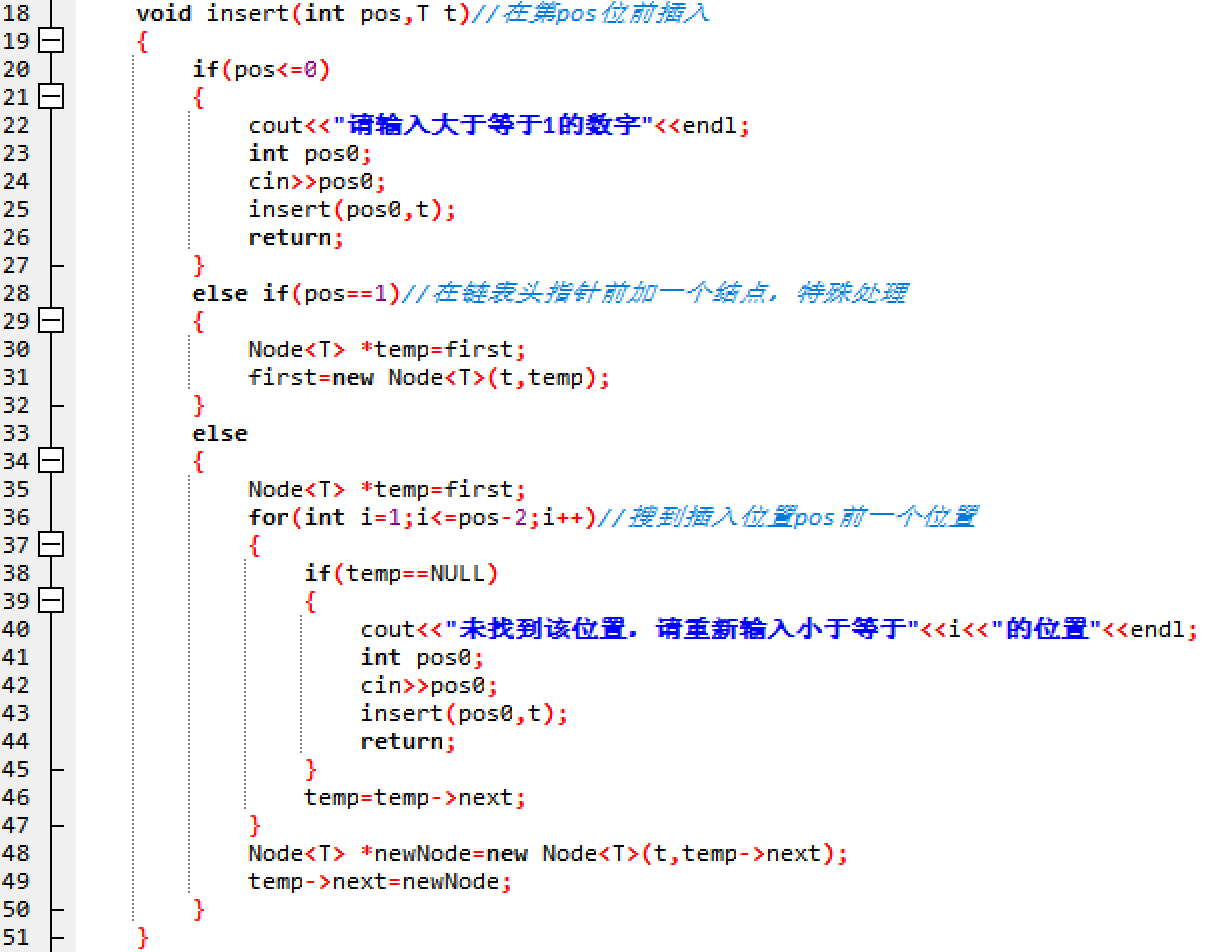
首先建立模板类LinkList<Student>的对象studentList，同时输出窗口将提示引导用户逐步输入数据来建立学生信息表，之后展示所建立的学生信息表。然后进入循环，直到用户按0退出程序。循环中，根据用户输入的数字判断进行何种操作，调用studentList对象相应的函数，完成相应的功能.

# 3 实现

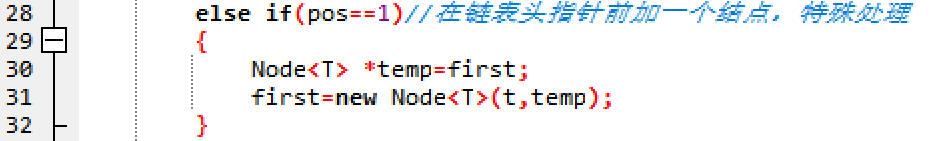
## 3.1 插入功能的实现

用insert()函数实现，在第pos个位置前面插入。但是插入时候要分情况进行讨论，我分了三种情况进行讨论，链表链表头部位置的插入、链表中间与尾部位置的插入、链表插入中不合理情况的判断。

insert()函数的完整截图如下：

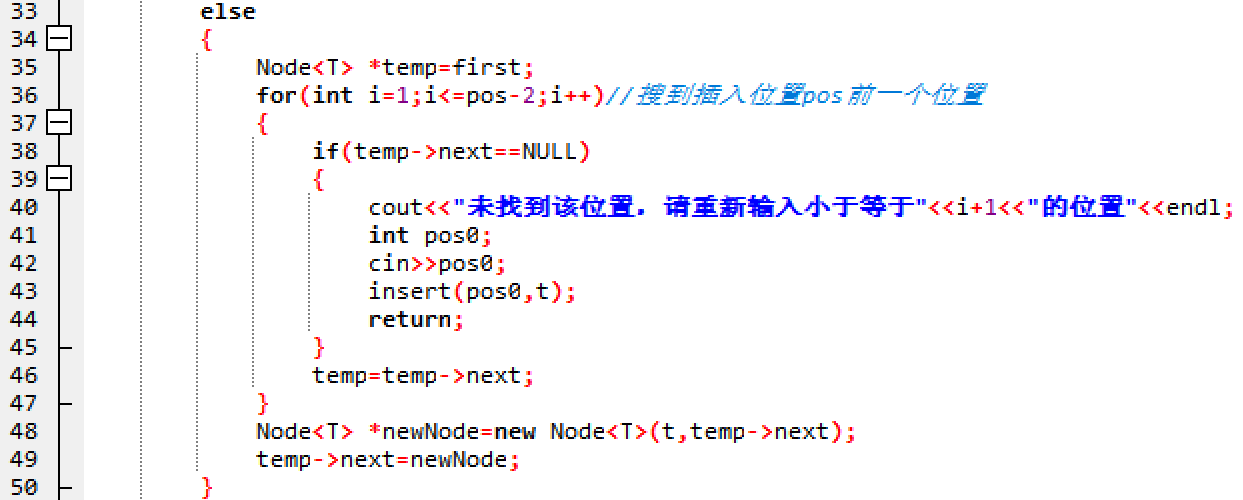


### 3.1.1 链表头部位置的插入



在链表头部位置插入的步骤是：先建立一个后继为当前链表头结点的结点，再将链表头结点改为这个结点。

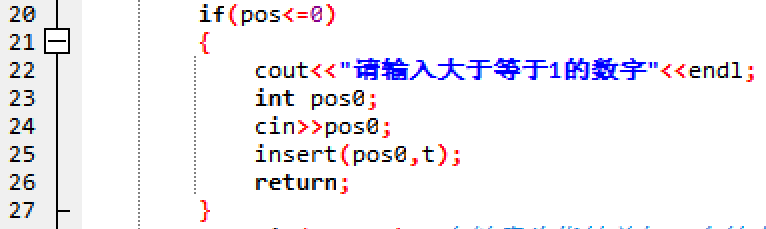
### 3.1.2 链表中间与尾部位置的插入



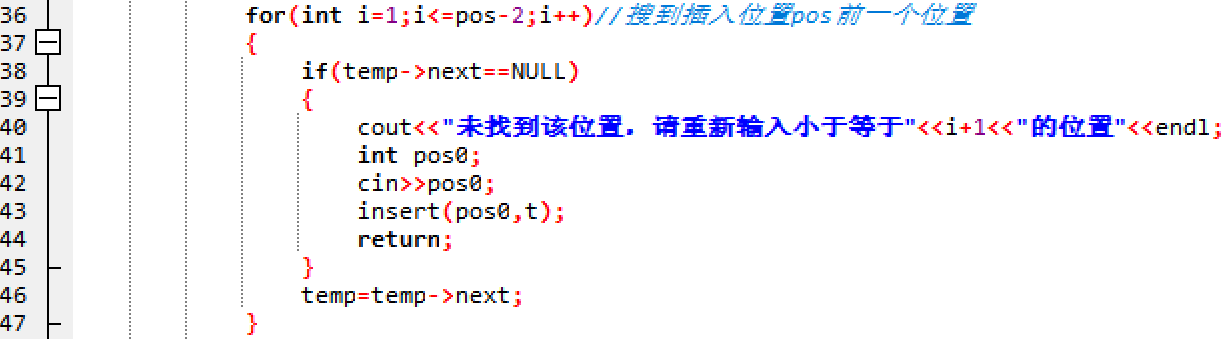
链表中间与尾部位置插入的步骤是：找到要插入的位置前一个位置temp，先建立一个后继为temp->next的结点，再将temp结点的后继改为这个结点。

### 3.1.3 链表插入中不合理情况的判断

第一处，如果位置小于等于0，输出“请输入大于等于1的数字”，输入数字后，递归调用insert()函数，重新再判断输入是否合理，一直到输入合理为止才继续做下去。代码如下：

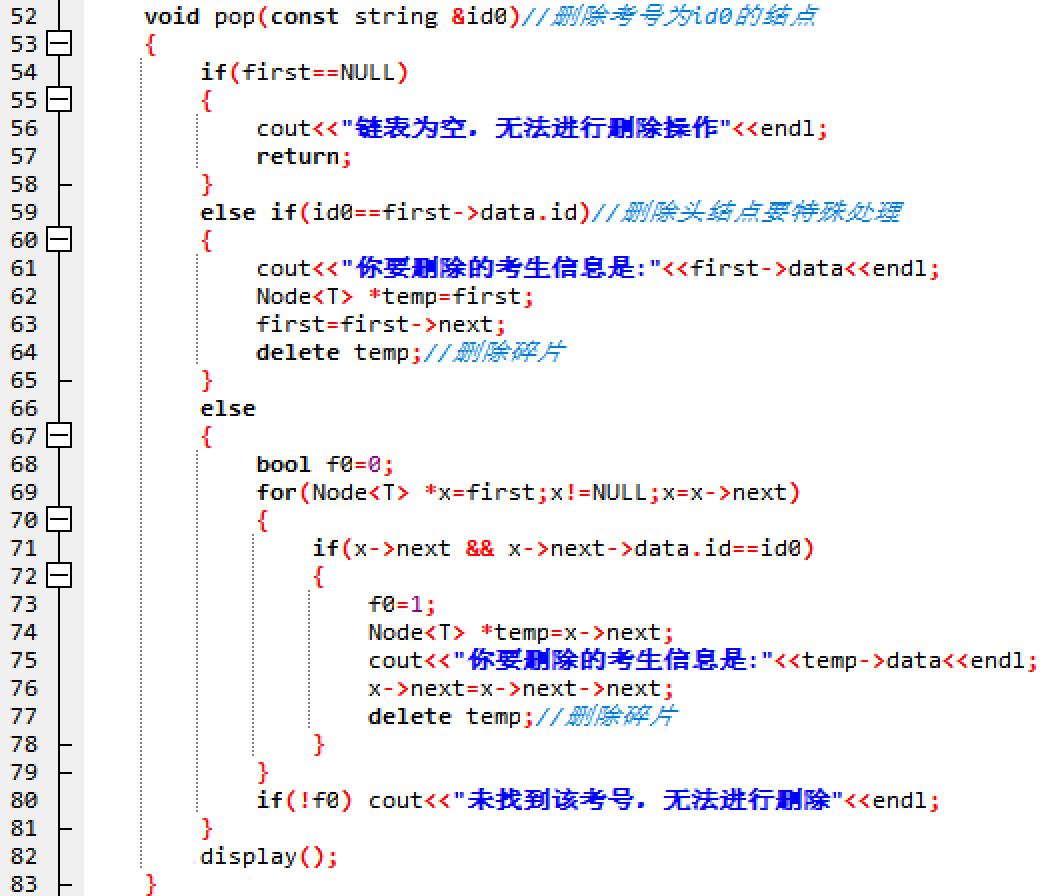


第二处，由于我没有设计链表长度的成员变量，所以当pos超过链表长度时，只能归结到循环到temp为空时，说明找到链表末尾都没有找到该位置。此时知道了链表的长度为i，（在第i+1位插入时，是插入链表最后一个元素后面；在第i位插入时，时插入链表最后一个元素前面），输出“未找到该位置，请重新输入小于等于i+1的位置”，输入数字后，递归调用insert()函数，重新再判断输入是否合理，一直到输入合理为止才继续做下去。代码如下：



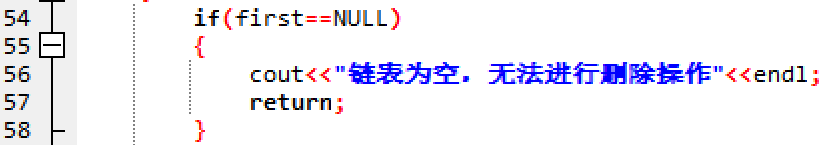
## 3.2 删除功能的实现

用pop()函数实现，删除考号为id0的结点。但是删除时要分两种情况进行讨论：删除头部结点和删除中间与尾部结点。pop()函数的完整截图如下：



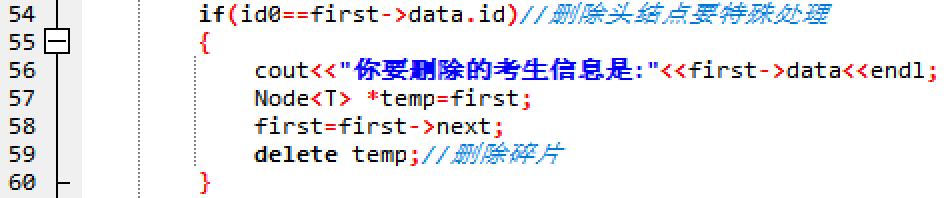
### 3.2.1 链表为空的情况

输出“链表为空，无法进行删除操作”，并返回，防止程序崩溃。



### 3.2.2 删除头部节点

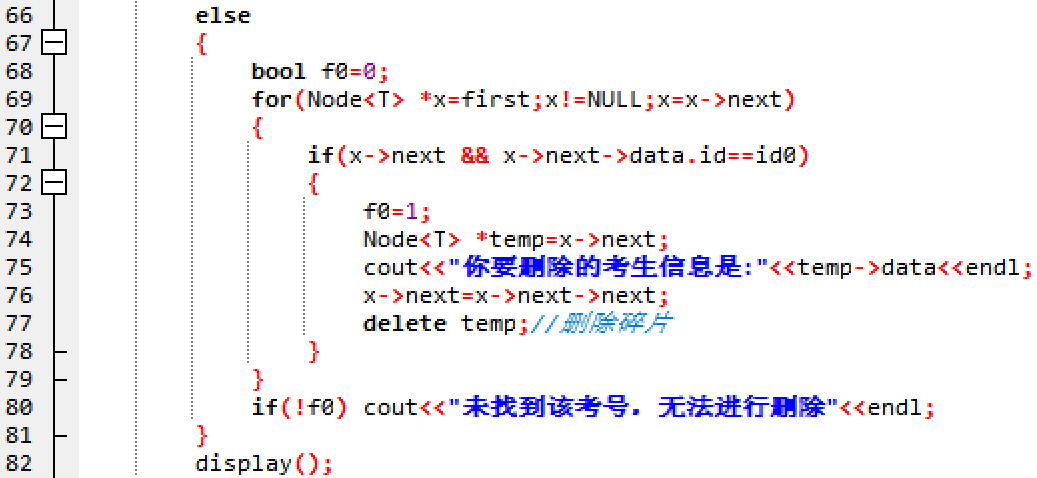
如果id0的结点是首元素，首先按要求输出要删除的考生信息，然后直接让first指向它的下一个元素，最后删除首元素的存储碎片。但我们要考虑一下如果只有一个元素，这样是否可行——事实证明是可行的，如果只有一个元素，first指向下个元素为NULL，头指针为空，链表为空。



### 3.2.3 删除中间结点与尾结点

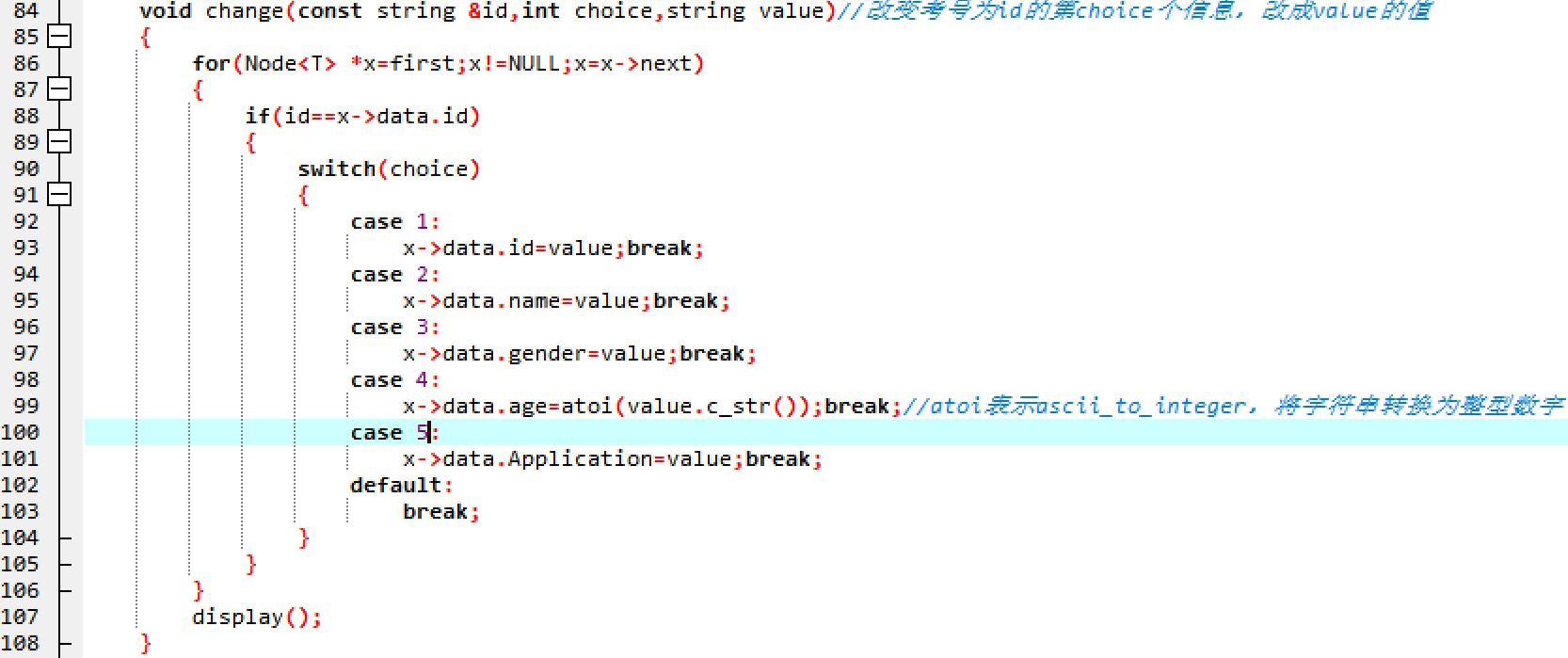
若非首元素，则通过检测下一个元素是否为空，找到当前元素的后继元素的id等于id0的元素，记下当前元素的后继元素位置temp，先将当前元素的后继改为它后继的后继，再删除temp并消除它的存储碎片。

如果链表循环完都没有找到考号为id0的考生，则输出“未找到该考号，无法进行删除”。



## 3.3 修改功能的实现

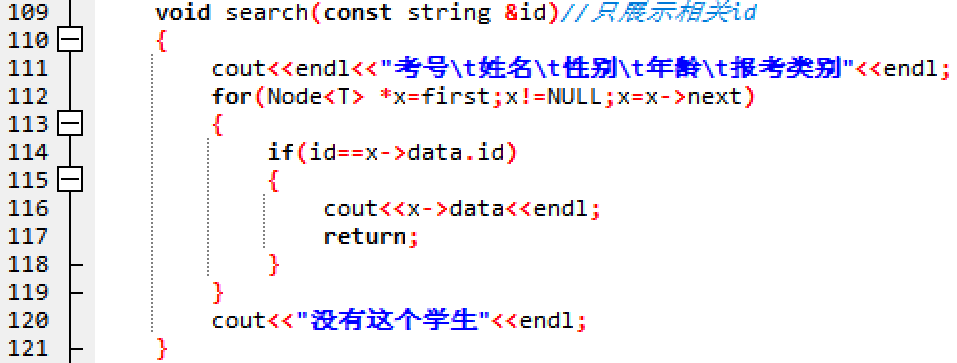
首先找到考号为id的学生，然后用switch(choice)找到想要修改的属性（第choice个信息），并将它改为value值。由于年龄记录的类型为int，但是传入的age的数据类型为string，经过查阅相关资料，可以用stdlib.h库中的atoi函数，即ascii\_to\_integer，将字符串转换为整型数字。



## 3.4 查找功能的实现

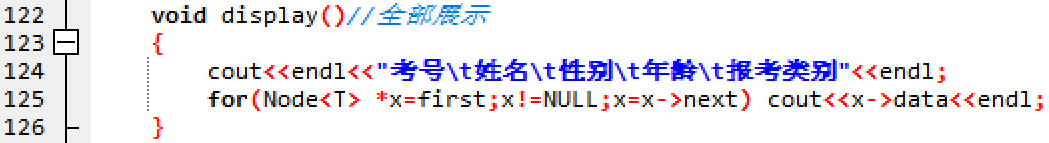
用for循环来遍历链表，找到链表中考号等于id的学生，通过重载的输出把打印该学生信息。

如果遍历到链表末尾都没有return，则输出“没有这个学生”。



## 3.5 统计功能的实现

统计功能即全部展示。



## 3.6 总体系统的实现

### 3.6.1 总体系统描述

输入总人数num，创建表studentList，同时输出窗口将提示引导用户逐步输入数据来建立学生信息表，之后展示所建立的学生信息表。然后进入循环，用switch根据根据用户输入的不同的operatorCode来调用链表里相应的函数，直到用户按0退出程序。

### 3.6.2 总体系统核心代码

int main(){

cout<<"首先请建立考试信息系统!"<<endl; cout<<"首先输入考试人数："; int num; cin>>num;//输入学生人数

while(num<0) { cout<<"请输入大于等于0的考生人数";

cin>>num;

} cout<<"请依次输入考生的考号，姓名，性别，年龄和报考类别!"<<endl; LinkList<Student> studentList; for(int i=1;i<=num;i++) { string id,name,gender,Application; int age; Student student; cin>>student; studentList.insert(i,student); } studentList.display(); cout<<"请选择你要进行的操作(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)"<<endl<<endl; bool is\_continue=true; while(is\_continue) { cout<<"请选择你要进行的操作："; int operatorCode; int pos; string id,name,gender,Application; int age; string value; Student student; cin>>operatorCode; switch(operatorCode)//1、2、3、4、5、0分别对应如上的操作 { case 1: cout<<"请输入你要插入的考生的位置："; cin>>pos; 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！"<<endl; cin>>student; studentList.insert(pos,student); studentList.display(); break; case 2: cout<<"请输入要删除的考生的考号："; cin>>id; studentList.pop(id); break; case 3: cout<<"请输入要查找的考生的考号："; cin>>id; studentList.search(id); break; case 4: cout<<"请输入要修改的考生的考号："; cin>>id; cout<<"请输入要修改的位置："; cin>>pos; cout<<"请输入要修改的值："; cin>>value; studentList.change(id,pos,value); break; case 5: studentList.display(); break; case 0: is\_continue=false; break; default:break; } } return 0; }

# 4 测试

## 4.1 功能测试

### 4.1.1 建立功能测试

**测试用例**：

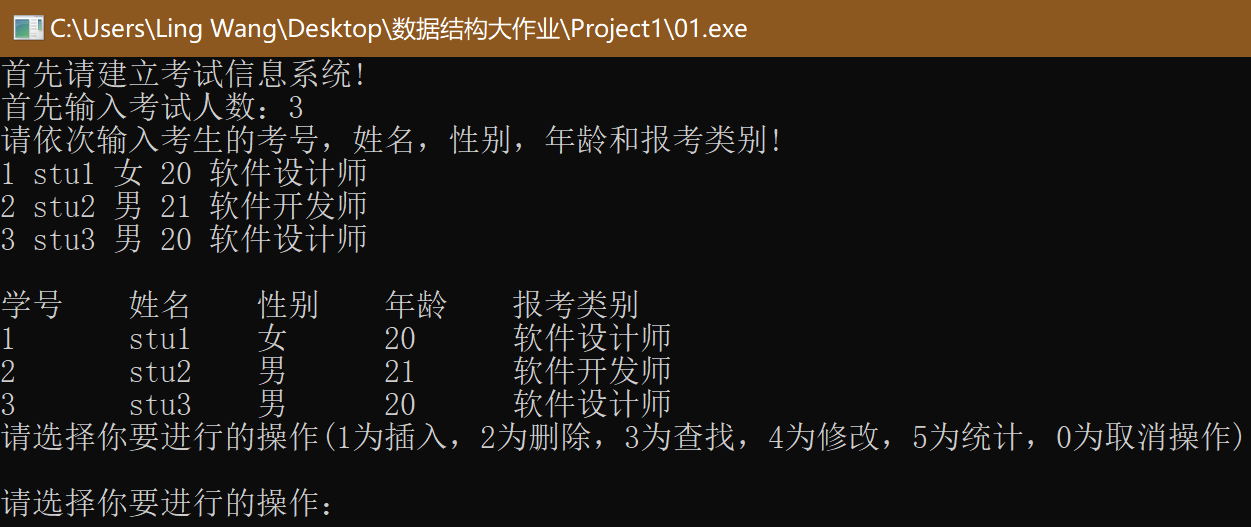
3

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

**实验结果：**



### 4.1.2 插入功能测试

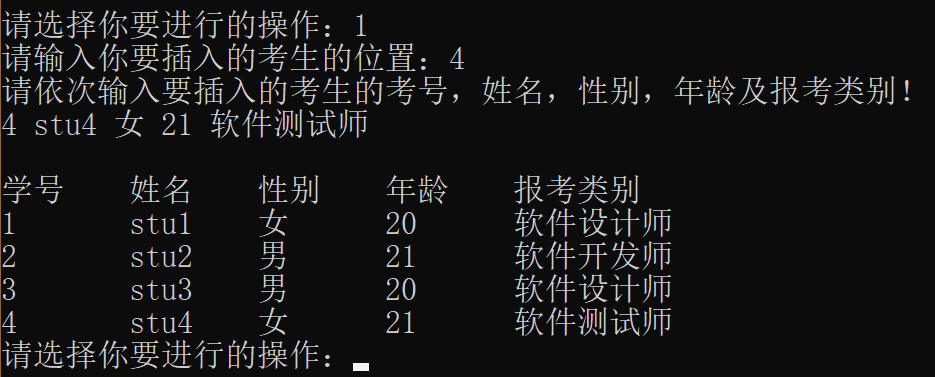
**测试用例：**

1

4

4 stu4 女 21 软件测试师

**实验结果：**



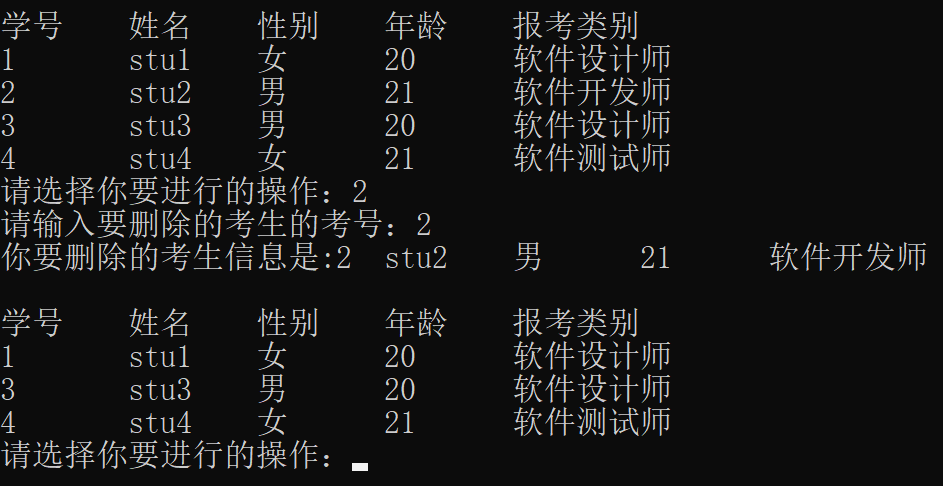
### 4.1.3 删除功能测试

**测试用例：**

2

2

**实验结果：**



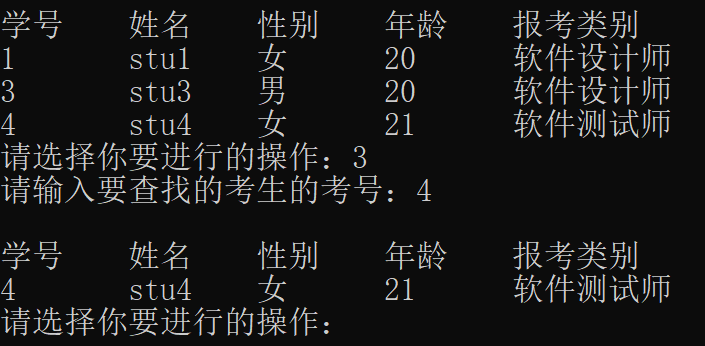
### 4.1.4 查找功能测试

**测试用例：**

3

4

**实验结果：**



### 4.1.5 修改功能测试

**测试用例：**

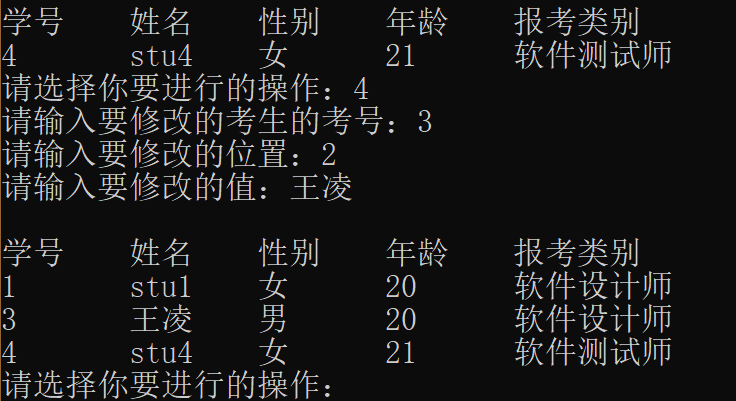
4

3

2

王凌

**实验结果：**

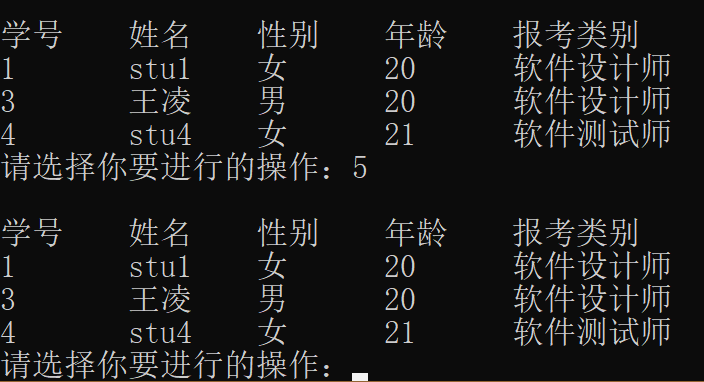


### 4.1.6 统计功能测试

**测试用例：**

5

**实验结果：**



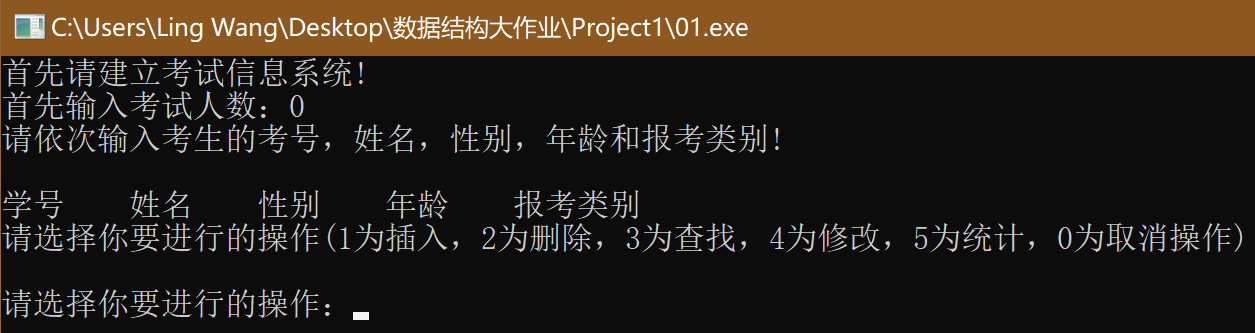
## 4.2 边界测试

### 4.2.1 初始化输入人数为0

**测试用例：**输入人数为0

**预期结果：**不报错，继续执行

**实验结果：**

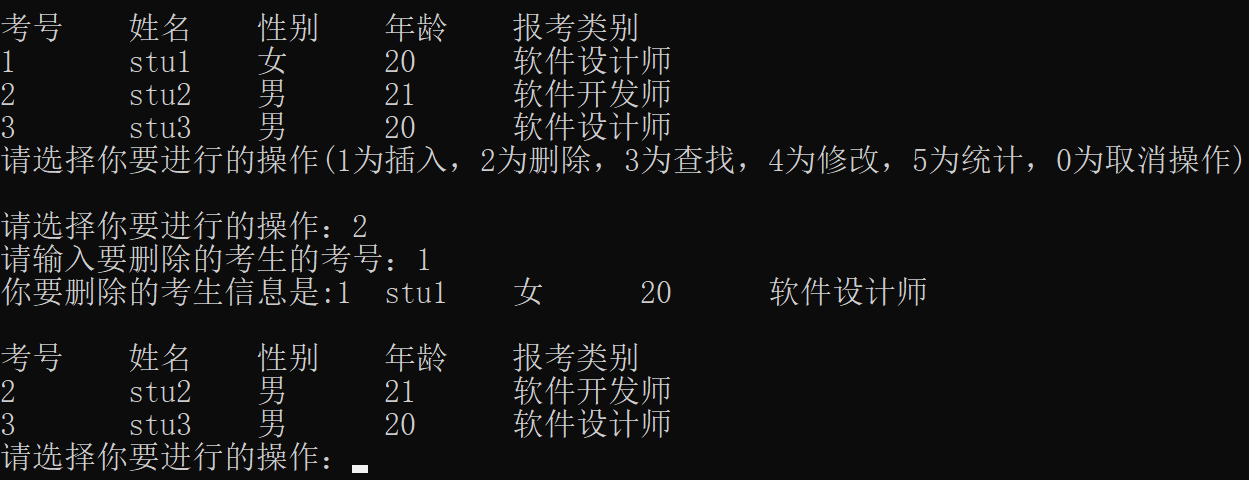


### 4.2.2 删除头结点

**测试用例：**删除头结点

**预期结果：**正常运行

**实验结果：**

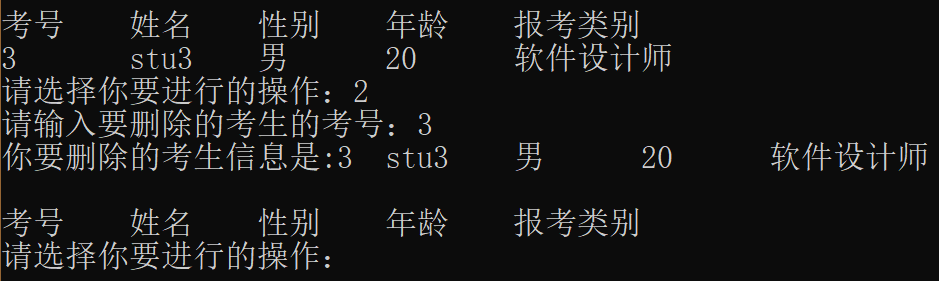


### 4.2.3 删除后链表为空

**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：**正常运行

**实验结果：**

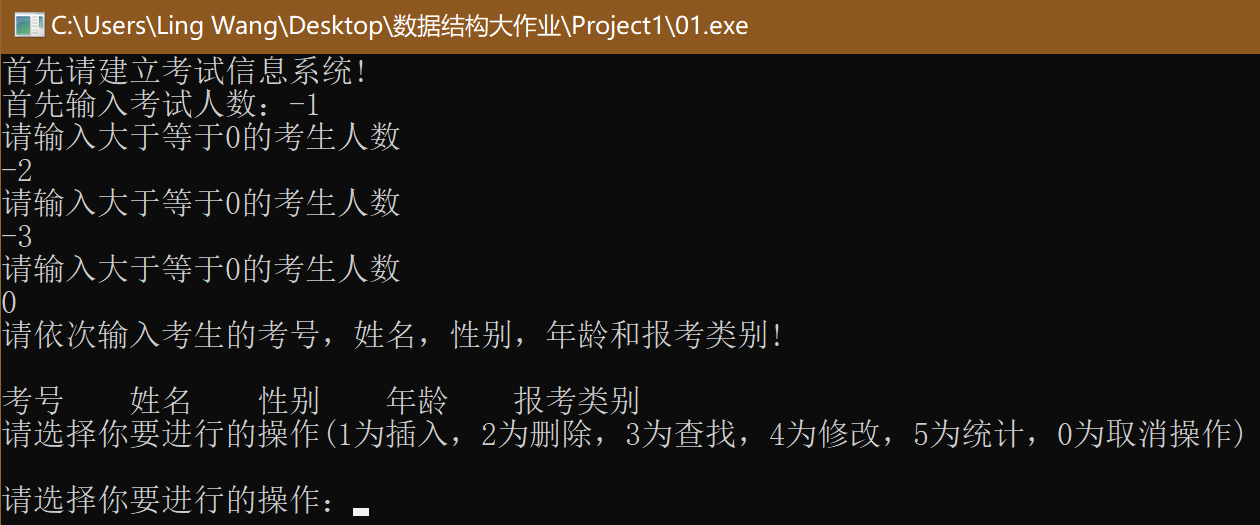


### 4.2.4 输入考生人数为负数

**测试用例：**输入考生人数为负数

**预期结果：**输出"请输入大于等于0的考生人数"，并一直循环到输入值不为负数为止。

**实验结果：**



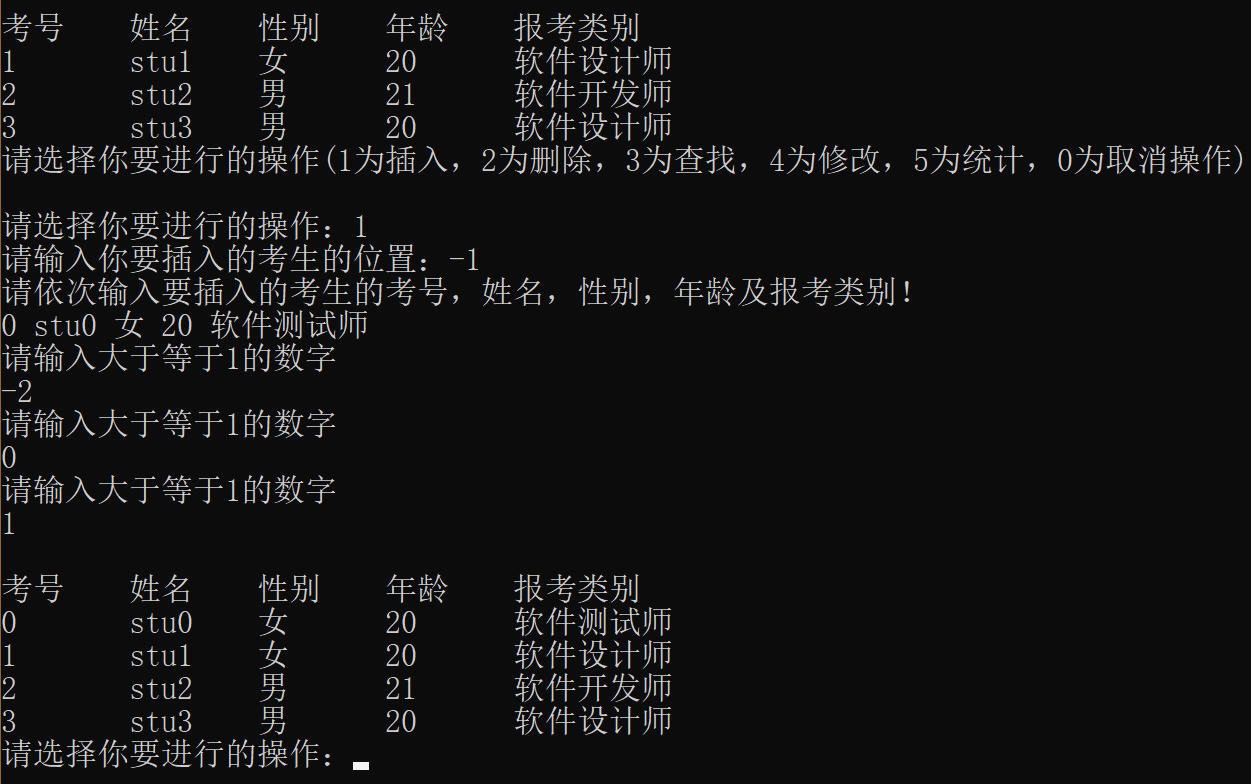
### 4.2.5 输入的插入位置不正确

**测试用例：**在不正确的位置进行插入

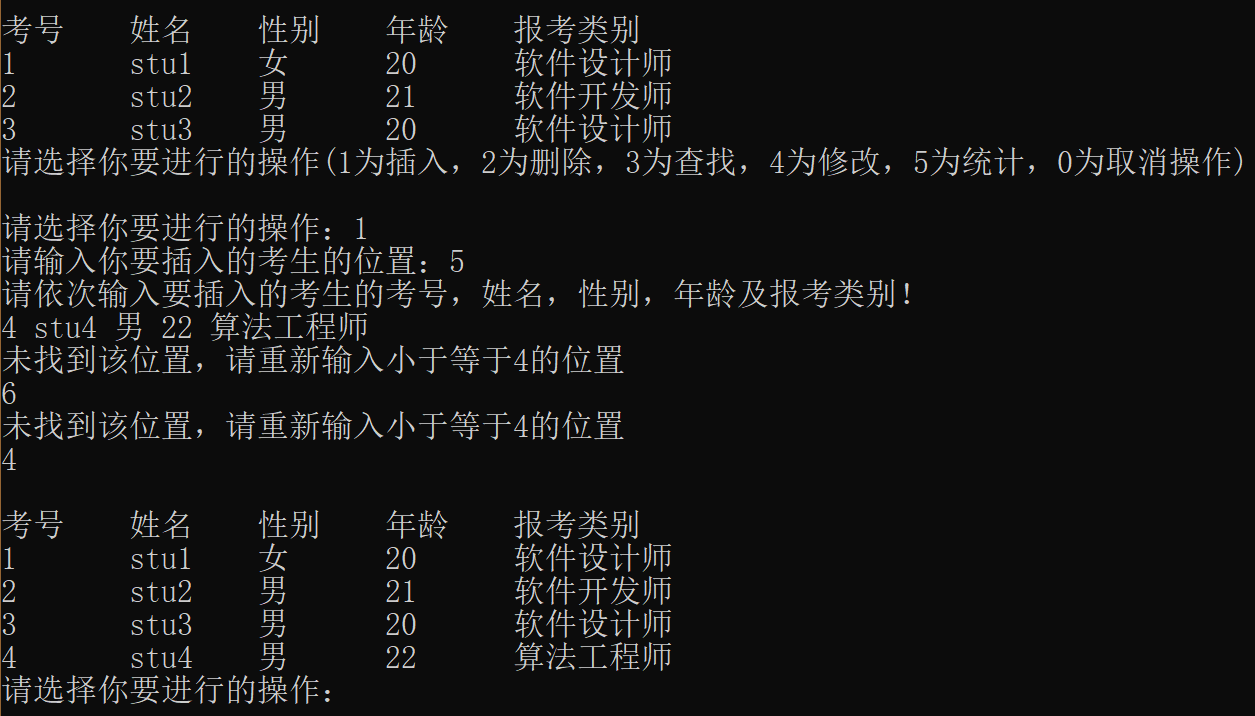
**预期结果：**如果位置小于等于0，输出“请输入大于等于1的数字”，输入数字后，递归调用insert()函数，重新再判断输入是否合理，一直到输入合理为止才继续做下去。如果位置超出链表长度，输出“未找到该位置，请重新输入小于等于i+1(数字，链表长度)的位置”，输入数字后，递归调用insert()函数，重新再判断输入是否合理，一直到输入合理为止才继续做下去。

**实验结果：**

1.



2.



### 4.2.6 表中没有输入的id

**测试用例：**删除不存在的id

**预期结果：**输出“未找到该考号，无法进行删除”

**实验结果：**

