项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 王星洲

学 号： 1652977

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](#_Toc534066875)

[1.1 背景分析 1](#_Toc534066876)

[1.2 功能分析 1](#_Toc534066877)

[2 设计 2](#_Toc534066878)

[2.1 数据结构设计 2](#_Toc534066879)

[2.2 类结构设计 2](#_Toc534066880)

[2.3 成员与操作设计 2](#_Toc534066881)

[2.4 系统设计 3](#_Toc534066882)

[3 实现 4](#_Toc534066883)

[3.1 创建链表 4](#_Toc534066884)

[3.1.1 创建链表核心代码 4](#_Toc534066885)

[3.1.2 创建说明 4](#_Toc534066886)

[3.1.3 初始化截屏示例 4](#_Toc534066887)

[3.2 插入功能的实现 5](#_Toc534066888)

[3.2.1 插入功能流程图 5](#_Toc534066889)

[3.2.2 插入功能核心代码 6](#_Toc534066890)

[3.2.3 插入功能截屏示例 7](#_Toc534066891)

[3.3 删除功能的实现 8](#_Toc534066892)

[3.3.1 删除功能流程图 8](#_Toc534066893)

[3.3.2 删除功能核心代码 9](#_Toc534066894)

[3.3.3 删除功能截屏示例 10](#_Toc534066895)

[3.4 查找功能的实现 11](#_Toc534066896)

[3.4.1 查找功能流程图 11](#_Toc534066897)

[3.4.2 查找功能核心代码 12](#_Toc534066898)

[3.4.3 查找功能截图示例 13](#_Toc534066899)

[3.5 修改功能的实现 14](#_Toc534066900)

[3.5.1 修改功能流程图 14](#_Toc534066901)

[3.5.2 修改功能核心代码 15](#_Toc534066902)

[3.5.3 修改功能截屏示例 16](#_Toc534066903)

[3.6 统计功能的实现 17](#_Toc534066904)

[3.6.1 统计功能流程图 17](#_Toc534066905)

[3.6.2 统计功能核心代码 18](#_Toc534066906)

[3.6.3 统计功能截屏示例 18](#_Toc534066907)

[3.7 总体系统的实现 19](#_Toc534066908)

[3.7.1 总体系统流程图 19](#_Toc534066909)

[3.7.2 总体系统核心代码 20](#_Toc534066910)

[4 测试 23](#_Toc534066911)

[4.1 功能测试 23](#_Toc534066912)

[4.1.1 插入功能测试 23](#_Toc534066913)

[4.1.2 删除功能测试 24](#_Toc534066914)

[4.1.3 查找功能测试 24](#_Toc534066915)

[4.1.4 修改功能测试 25](#_Toc534066916)

[4.1.5 统计功能测试 26](#_Toc534066917)

[4.2 边界测试 27](#_Toc534066918)

[4.2.1 初始化无输入数据 27](#_Toc534066919)

[4.2.2 删除后链表为空 27](#_Toc534066920)

[4.3 出错测试 28](#_Toc534066921)

[4.3.1 考生人数错误 28](#_Toc534066922)

[4.3.2 操作码错误 28](#_Toc534066923)

[4.3.3 插入位置不存在 28](#_Toc534066924)

[4.3.4 删除考号不存在 29](#_Toc534066925)

[4.3.5 查找考号不存在 29](#_Toc534066926)

[4.3.6 修改考号不存在 30](#_Toc534066927)

[5 亮点 31](#_Toc534066928)

[5.1 人机交互 31](#_Toc534066929)

[5.2 带头结点链表的使用 31](#_Toc534066930)

# 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。

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 功能分析

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。

# 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。本系统的数据结构设计采取友元的方式，设计一个有结点的链表。

## 2.3 成员与操作设计

class LNode { //结点类

friend MyList;

public:

string m\_studentID; //表示学生考号

string m\_studentName;//表示学生姓名

string m\_sex;//表示学生性别

int m\_age;//表示学生年龄

string m\_category;//表示学生报考种类

LNode \*m\_next;//链表结点的指针域

LNode() {}

LNode(string ID, string name, string sex, int age, string cate, LNode \*ptr = NULL);

void Update();//更改结点数据

void print();//打印结点数据

};

class MyList { //链表类

public:

void InitList(int); //头插法初始化带头结点的表,长度为n,不算头结点

void ClearList(); //清空表

bool ListEmpty(); //判空

LNode\* LocateElemByID(string ID); //找出学号为ID的元素并返回

bool ListInsert(int i, string ID, string name, string sex, int age, string cate); //第i个位置插入元素e

bool ListDelete(string ID); //删除学号为ID的元素

void PrintList(); //输出链表

private:

LNode \*head; //头指针

};

## 2.4 系统设计

系统首先调用接收用户的输入，创建一个初始链表，再通过循环和选择语句接收用户的指令，直到退出系统。

# 3 实现

## 3.1 创建链表

### 3.1.1 创建链表核心代码

void MyList::InitList(int num)

{

int age;

string ID, name, sex, cate;

head = new LNode;

LNode\* last = head;

head -> m\_next = NULL;

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

for (int i = 0; i<num; ++i)

{

cin >> ID >> name >> sex >> age >> cate;

LNode \*p = new LNode;

p->m\_age = age;

p->m\_category = cate;

p->m\_sex = sex;

p->m\_studentID = ID;

p->m\_studentName = name;

last->m\_next = p;

last = p;

m\_length++;

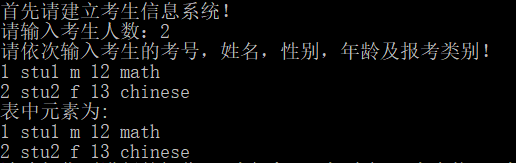
}

}

### 3.1.2 创建说明

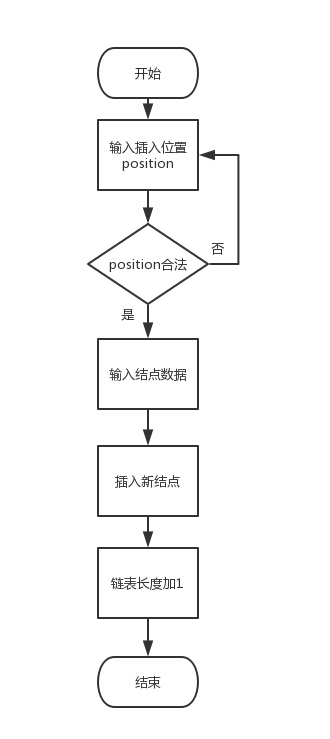
创建链表采用了后插式创建的方法，创建结束后的链表顺序与输入顺序一致。创建之前赋给链表一个头结点，接下来在创建一个尾指针，让尾指针指向链表最后一个元素，插入新元素时，将新元素插入尾指针的下一个位置，并将尾指针指向新元素。

### 3.1.3 初始化截屏示例



## 3.2 插入功能的实现

### 3.2.1 插入功能流程图



### 3.2.2 插入功能核心代码

case 1:

{

int position = 0;

string ID, name, sex, cate;

int age;

cout << "请输入你要插入的考生的位置：";

while (!(cin >> position))

{

cout << "输入有误，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

while(position <= 0 || position > Students.m\_length + 1)

{

cout << "您输入的位置超出范围，请重新输入" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

cin >> position;

}

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

cin >> ID >> name >> sex >> age >> cate;

if (Students.ListInsert(position, ID, name, sex, age, cate))

{

cout << "插入成功" << endl;

Students.PrintList();

}

else

{

cout << "插入失败，请检查输入并重试" << endl;

}

break;

}

bool MyList::ListInsert(int i, string ID, string name, string sex, int age, string cate)

{

LNode \*p, \*s;

int j = 1;

p = head;

while (p&&j<i) { p = p->m\_next; j++; }

if (p == NULL) return false;

if ((s = new LNode) == NULL) return false;

s->m\_age = age;

s->m\_category = cate;

s->m\_sex = sex;

s->m\_studentID = ID;

s->m\_studentName = name;

s->m\_next = p->m\_next;

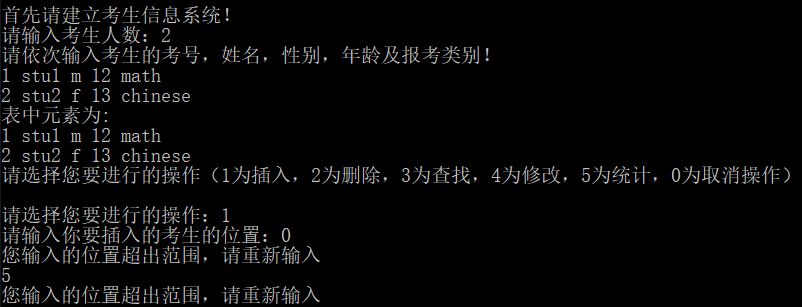
p->m\_next = s;

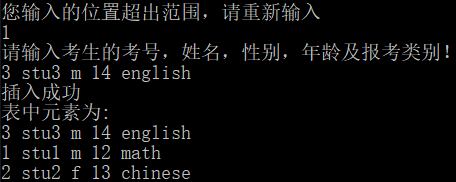
m\_length++;

return true;

}

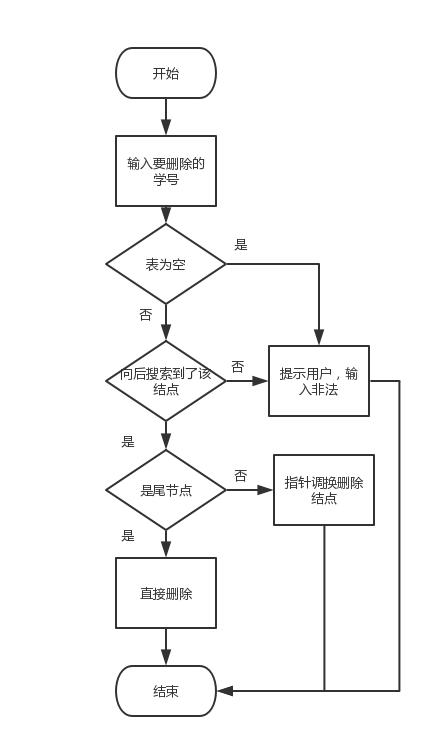
### 3.2.3 插入功能截屏示例





## 3.3 删除功能的实现

### 3.3.1 删除功能流程图



### 3.3.2 删除功能核心代码

case 2:

{

string ID;

cout << "请输入您要删除的考生的考号：";

cin >> ID;

if (Students.ListDelete(ID))

{

cout << "删除成功" << endl;

Students.PrintList();

}

else

{

cout << "删除失败，请检查输入并重试" << endl;

}

break;

}

bool MyList::ListDelete(string ID)

{

LNode \*p, \*q;

int i = 1;

if (ListEmpty())

{

cout << "表已经空了，不能删除。" << endl;

return false;

}

p = head->m\_next;

while (p && p->m\_studentID != ID) { p = p->m\_next; i++; }

if (i > m\_length)

{

return false;

}

else

{

if (i < m\_length)

{

q = p->m\_next;

p->m\_next = q->m\_next;

delete q;

q = NULL;

m\_length--;

return true;

}

else

{

delete p;

p = NULL;

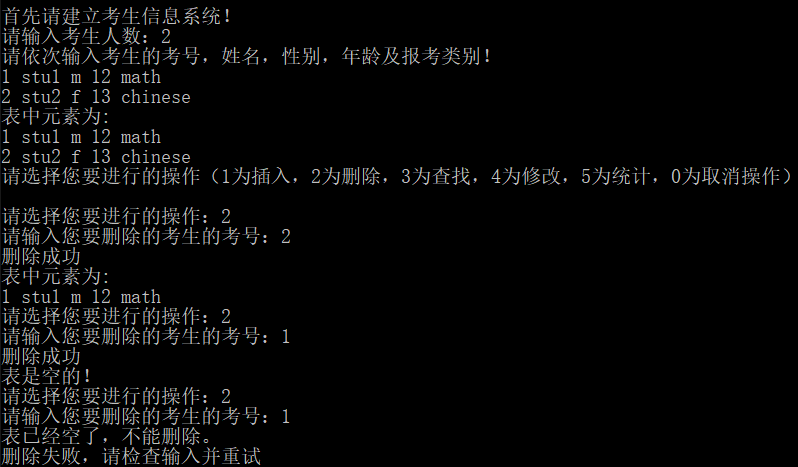
m\_length--;

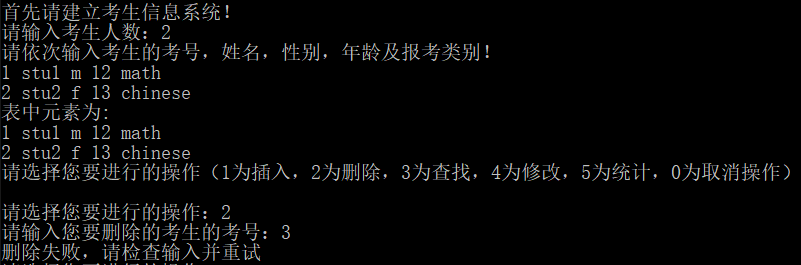
}

}

}

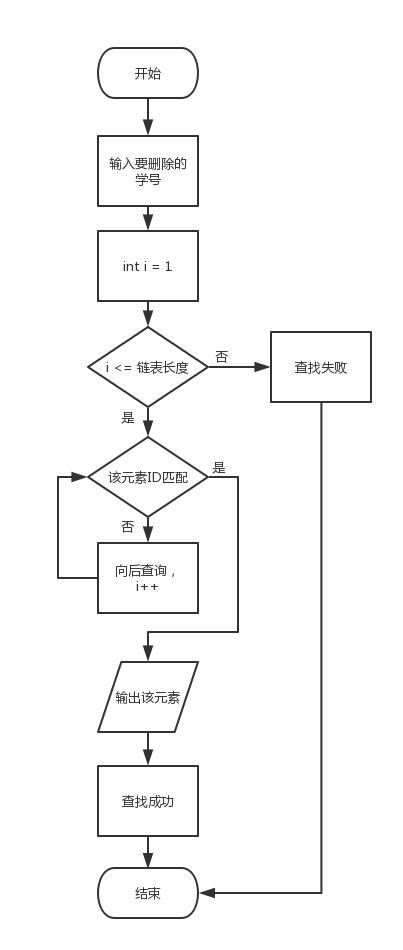
### 3.3.3 删除功能截屏示例





## 3.4 查找功能的实现

### 3.4.1 查找功能流程图



### 3.4.2 查找功能核心代码

case 3:

{

string ID;

cout << "请输入您要查找的考生的考号：";

cin >> ID;

LNode \*p = Students.LocateElemByID(ID);

if (p != Students.head)

{

p->print();

}

else

{

cout << "您查找的考生并不存在" << endl;

}

break;

}

LNode\* MyList::LocateElemByID(string ID)

{

int i = 1;

LNode \*p = head->m\_next;

while (i <= m\_length && p->m\_studentID != ID)

{

p = p->m\_next;

i++;

}

if (i > m\_length)

{

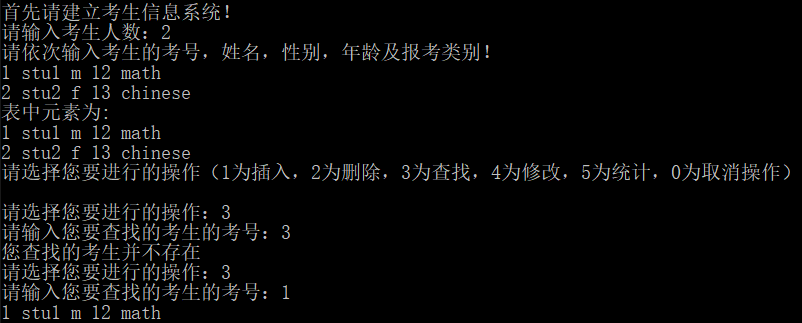
return head;

}

else return p;

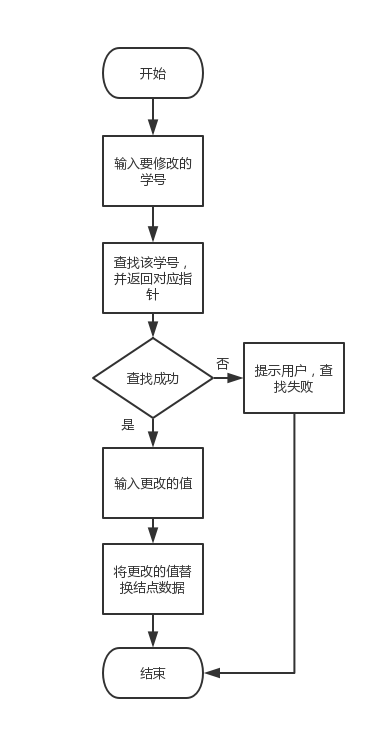
}

### 3.4.3 查找功能截图示例



## 3.5 修改功能的实现

### 3.5.1 修改功能流程图



### 3.5.2 修改功能核心代码

case 4:

{

string ID;

cout << "请输入您要更改的考生的考号：";

cin >> ID;

LNode \*p = Students.LocateElemByID(ID);

if (p != Students.head)

{

p->Update();

}

else

{

cout << "您查找的考生并不存在" << endl;

}

break;

}

void LNode::Update()

{

int age;

string ID, name, sex, cate;

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

cin >> ID >> name >> sex >> age >> cate;

m\_age = age;

m\_category = cate;

m\_sex = sex;

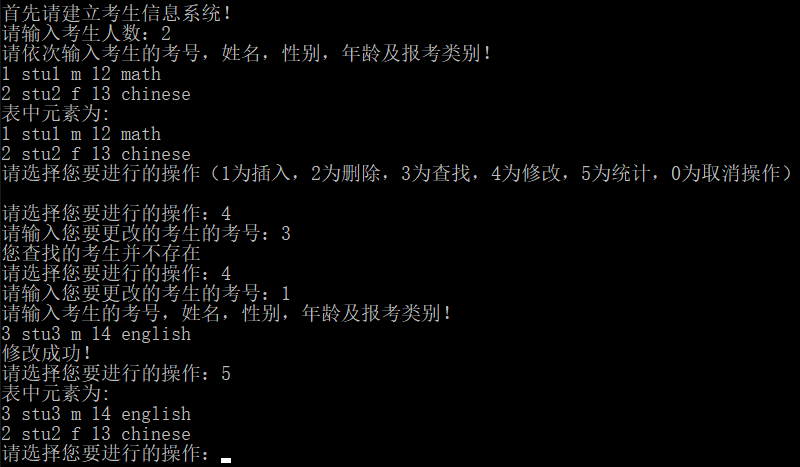
m\_studentID = ID;

m\_studentName = name;

cout << "修改成功！" << endl;

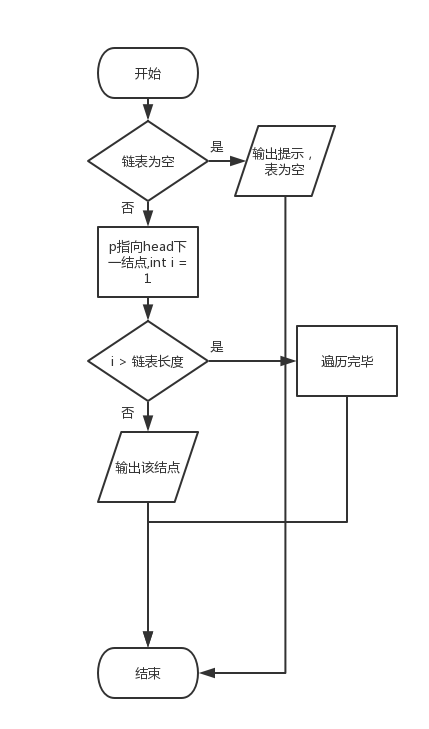
}

### 3.5.3 修改功能截屏示例



## 3.6 统计功能的实现

### 3.6.1 统计功能流程图



### 3.6.2 统计功能核心代码

void MyList::PrintList()

{

int i = 0;

if (ListEmpty()) { cout << "表是空的！" << endl; }

else

{

cout << "表中元素为:" << endl;

LNode \*p = head-> m\_next;

while (i < m\_length)

{

p->print();

p = p-> m\_next;

i++;

}

}

}

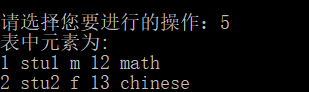
void LNode::print()

{

cout << m\_studentID << " " << m\_studentName << " " << m\_sex << " " << m\_age << " " << m\_category << endl;

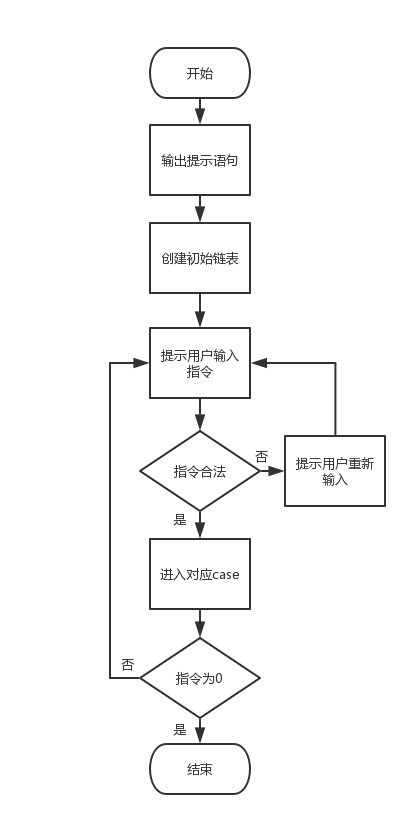
}

### 3.6.3 统计功能截屏示例



## 3.7 总体系统的实现

### 3.7.1 总体系统流程图



### 3.7.2 总体系统核心代码

cout << "首先请建立考生信息系统！" << endl;

cout << "请输入考生人数：";

cin >> number;

Students.InitList(number);

Students.PrintList();

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）" << endl;

cout << endl;

while (1)

{

cout << "请选择您要进行的操作：";

if (!(cin >> order))

{

cout << "输入非法，请重新输入" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

continue;

}

else

{

switch (order)

{

case 1:

{

int position = 0;

string ID, name, sex, cate;

int age;

cout << "请输入你要插入的考生的位置：";

while (!(cin >> position))

{

cout << "输入有误，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

while(position <= 0 || position > Students.m\_length + 1)

{

cout << "您输入的位置超出范围，请重新输入" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

cin >> position;

}

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

cin >> ID >> name >> sex >> age >> cate;

if (Students.ListInsert(position, ID, name, sex, age, cate))

{

cout << "插入成功" << endl;

Students.PrintList();

}

else

{

cout << "插入失败，请检查输入并重试" << endl;

}

break;

}

case 2:

{

string ID;

cout << "请输入您要删除的考生的考号：";

cin >> ID;

if (Students.ListDelete(ID))

{

cout << "删除成功" << endl;

Students.PrintList();

}

else

{

cout << "删除失败，请检查输入并重试" << endl;

}

break;

}

case 3:

{

string ID;

cout << "请输入您要查找的考生的考号：";

cin >> ID;

LNode \*p = Students.LocateElemByID(ID);

if (p != Students.head)

{

p->print();

}

else

{

cout << "您查找的考生并不存在" << endl;

}

break;

}

case 4:

{

string ID;

cout << "请输入您要更改的考生的考号：";

cin >> ID;

LNode \*p = Students.LocateElemByID(ID);

if (p != Students.head)

{

p->Update();

}

else

{

cout << "您查找的考生并不存在" << endl;

}

break;

}

case 5:

{

Students.PrintList();

break;

}

case 0:

{

cout << "退出系统成功！欢迎下次使用" << endl;

system("pause");

return 0;

}

default:

cout << "您输入的数值有误，请重新输入" << endl;

break;

}

}

}

system("pause");

return 0;

# 4 测试

## 4.1 功能测试

### 4.1.1 插入功能测试

**测试用例**：

初始链表

1 stu1 m 12 math

2 stu2 f 13 chinese

插入

位置3

3 stu3 m 14 english

**预期结果**：

1 stu1 m 12 math

2 stu2 f 13 chinese

3 stu3 m 14 english

**实验结果**

### 4.1.2 删除功能测试

**测试用例：**

初始链表

1 stu1 m 12 math

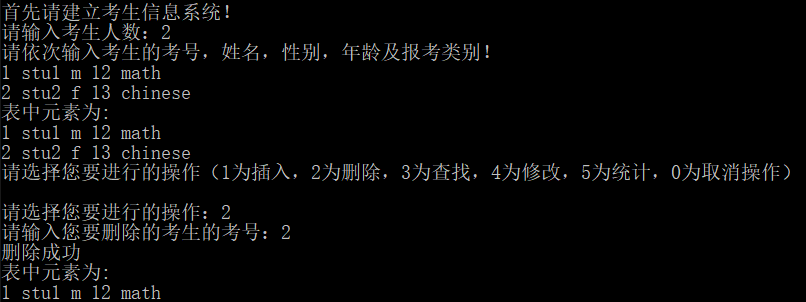
2 stu2 f 13 chinese

删除考号为2的考生

**预期结果：**

1 stu1 m 12 math

**实验结果：**



### 4.1.3 查找功能测试

**测试用例：**

初始链表

1 stu1 m 12 math

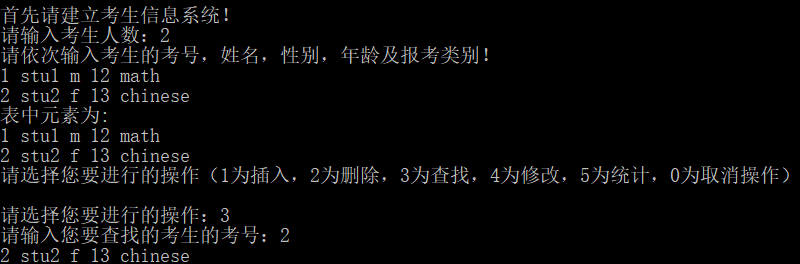
2 stu2 f 13 chinese

查找考号为2的考生

**预期结果：**

2 stu2 f 13 chinese

**实验结果：**



### 4.1.4 修改功能测试

**测试用例：**

初始链表

1 stu1 m 12 math

2 stu2 f 13 chinese

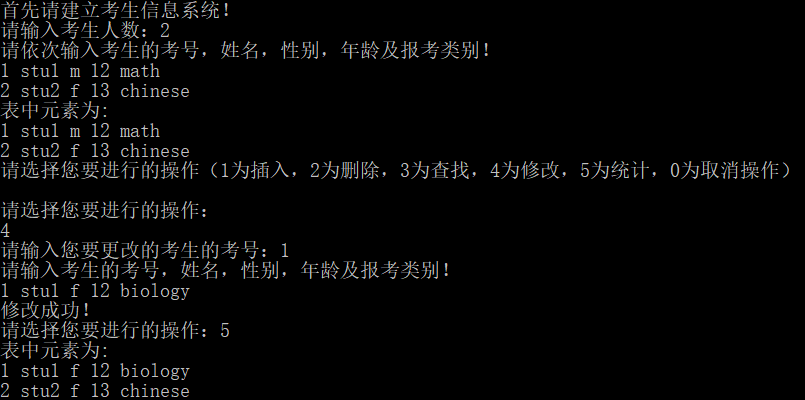
将考号1的人改为性别f 报考生物

**预期结果：**

1 stu1 f 12 biology

2 stu2 f 13 chinese

**实验结果：**



### 4.1.5 统计功能测试

**测试用例：**

初始链表

1 stu1 m 12 math

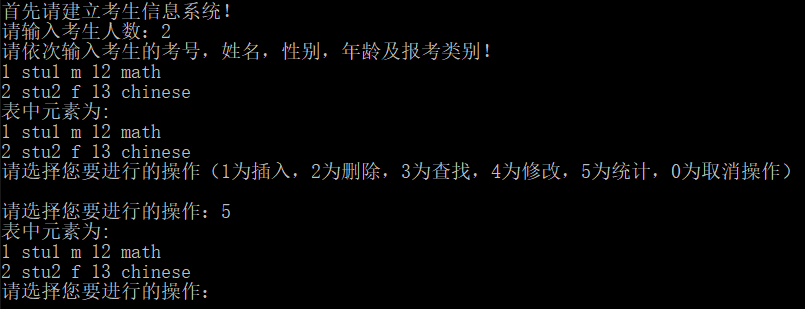
2 stu2 f 13 chinese

**预期结果：**

1 stu1 m 12 math

2 stu2 f 13 chinese

**实验结果：**



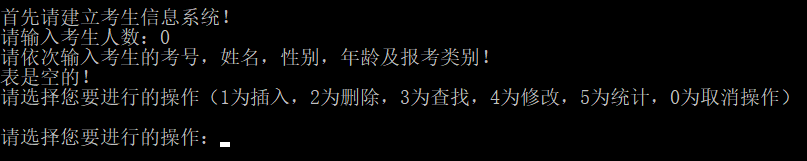
## 4.2 边界测试

### 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：**建立空表，程序正常运行

**实验结果：**

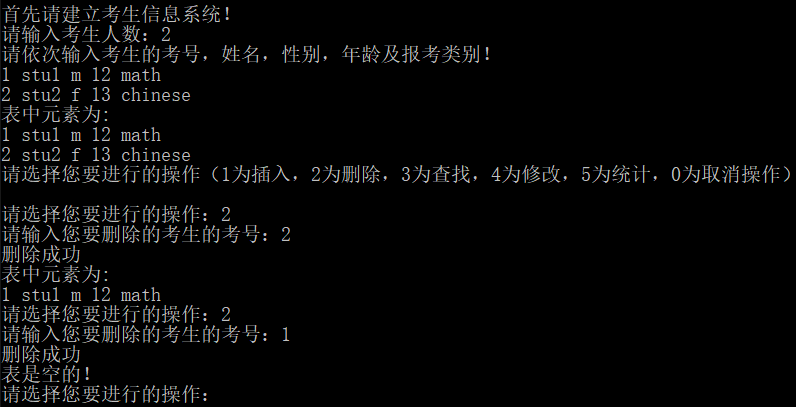


### 4.2.2 删除后链表为空

**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：**



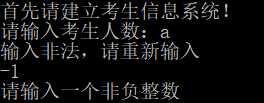
## 4.3 出错测试

### 4.3.1 考生人数错误

**测试用例：**输入考生人数为负数或字母

**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：**

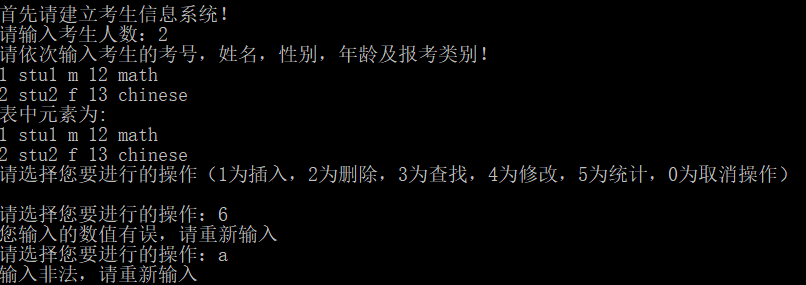


### 4.3.2 操作码错误

**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：**

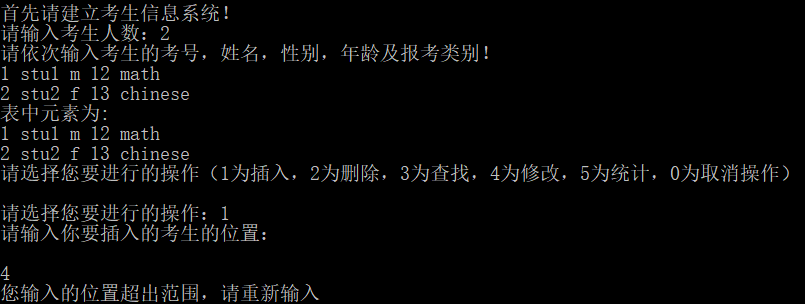


### 4.3.3 插入位置不存在

**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：**

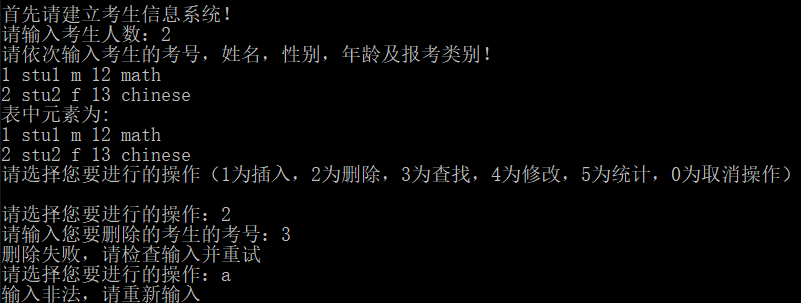


### 4.3.4 删除考号不存在

**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：**

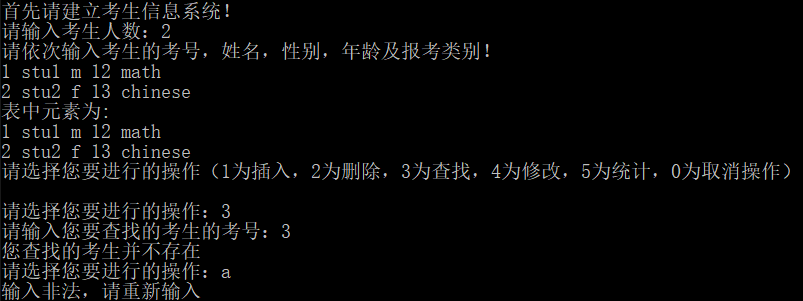


### 4.3.5 查找考号不存在

**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：**

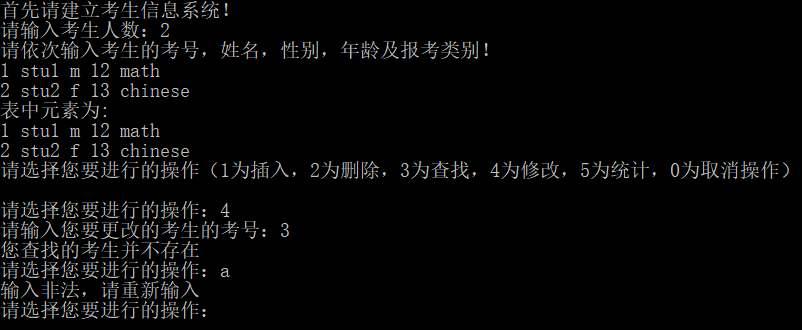


### 4.3.6 修改考号不存在

**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：**



# 5 亮点

## 5.1 人机交互

本题中我添加了大量人机代码满足人机交互需求，防止意外退出或者崩溃。详情见4.3

## 5.2 带头结点链表的使用

本题中，为了使链表的使用更加方便快捷，我使用了带头结点的链表，程序代码可读性，健壮性更强，详情见2.1， 3.1