

项目说明文档

数据结构课程设计

——考试报名系统

作者姓名：刘雪迪

学号：1752985

指导教师：张颖

学院/专业：软件学院/软件工程

一、分析

1.1 项目背景分析

1.2 项目功能要求

二、设计

2.1 数据结构设计

2.2 类结构设计

2.3 成员与操作设计

1. 节点类 (Node)

2. 链表类 (LinkList)

2.4 系统设计

三、实现

3.1 插入功能的实现

1. 插入功能流程图

2. 插入功能核心代码

3. 插入功能示意图

3.2 删除功能的实现

1. 删除功能流程图

2. 删除功能核心代码

3. 删除功能示意图

3.3 查找功能的实现

1. 查找功能流程图

2. 查找功能核心代码

3. 查找功能示意图

3.4 修改功能的实现

1. 修改功能流程图

2. 修改功能核心代码

3. 修改功能示意图

3.5 统计功能的实现

1. 统计功能流程图

2. 统计功能核心代码

3. 统计功能示意图

3.6 退出操作的实现

1. 退出操作核心代码

2. 退出操作功能示意图

3.7 总体系统的实现

1. 总体系统流程图

2. 总体系统的核心代码

3. 总体系统示意图

四、测试

4.1 功能测试

1. 插入功能测试

2. 删除功能测试

3. 查找功能测试

4. 修改功能测试

5. 统计功能测试

4.2 边界测试

1. 初始化无输入数据

2. 删除头节点

3. 删除后链表为空

4.3 出错测试

1. 考生人数错误
2. 操作码错误
3. 插入位置不存在
4. 删除考号不存在
5. 查找考号不存在
6. 修改考号不存在

一、分析

1.1 项目背景分析

考试报名系统是每个学校和教育机构都需要的系统，对于考试的管理者和报名考试的学生来说都十分重要。然而，由于考试人数众多，管理人员人手不够，考试报名工作往往会给教务管理部门增加很大的工作量。一个好用快捷的考试报名系统能简化操作，提高效率。本项目是对考试报名系统的简单模拟，用控制台选项的选择方式完成了下列功能：输入考生信息、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

作为一个最简单的考试报名系统，我们需要实现对考生信息的建立、查找、插入、修改、删除、退出等功能。其中，考生信息包括考号、姓名、性别、年龄和报考类别等信息。项目在设计时应该首先确定系统等数据结构，定义类等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。

二、设计

2.1 数据结构设计

由分析得，考试报名系统需要进行大量的增加、删除、修改操作，更新报名信息表的频率极高。而链表这种数据结构在进行增加、删除、修改等操作上，十分便捷，因此本系统的设计考虑采用链表这种数据结构。同时，为了使操作更简便、安全，本系统会在第一个储存信息的节点前附加一个头节点，在最后一个储存信息的节点后附加尾节点，并采用双向链表的模式，使得在处理头尾节点等边界情况等插入、删除操作时更便捷、安全，使程序更简洁，普适性更好。

链表是一种典型的动态存储结构。其中的数据，分散为一系列称作节点(node)的单位，节点之间通过指针相互索引和访问。为了引入新节点或删除原有节点，只需在局部，调整少量相关节点之间的指针。这就意味着，采用动态存储策略，可以大大降低动态操作的成本。

2.2 类结构设计

链表数据结构一般包含两个抽象数据类型——节点类（Node）和链表类（LinkedList）。两个类。节点类需不仅保存对应的数据项，还应记录其前驱和后继的位置。链表类则宏观地管理整个链表的更新操作。

2.3 成员与操作设计

1.节点类（Node）

私有成员：

```
int _number; //考号
std::string _name; //姓名
std::string _sex; //性别
int _age; //年龄
std::string _type; //报考科目
```

公有成员：

```
Node *next; //后继指针
Node *pre; //前驱指针
```

公有操作：

```

Node() = default;
Node(int number, std::string name, std::string sex, int age, std::string type); //初始化该节点
~Node();

Node *insertAsPred(Node *t); //紧接着该节点后面 插入新节点
Node *insertAsNext(Node *t); //紧接着这个节点前面 插入新的
void getInform(); //输出该节点（考生）的信息
void setInform(); //修改该节点（考生）的信息
int getNumber() { return this->_number; } //返回该节点（考生）的学号

```

2. 链表类（LinkedList）

私有成员：

```

int _size; //链表的规模
Node *header; //头哨兵
Node *trailer; //尾哨兵

```

公有操作：

```

LinkedList();
~LinkedList();

Node *insertAtPos(int pos); //在某个位置插入
Node *insertAsLast(Node *t); //在链表的最后插入
void remove(int num); //按考号删除考生位置
Node *find(int num); //按学号查找 返回对象
Node *change(int num); //按学号修改内容
void traverse(); //统计

```

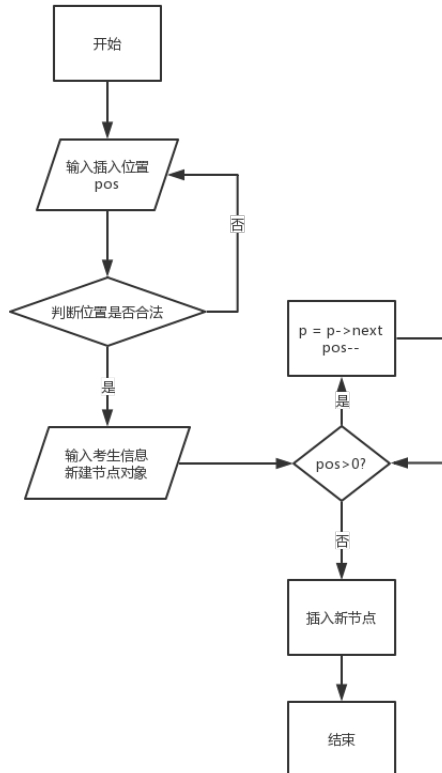
2.4 系统设计

系统在首次使用时，会提示使用者输入考生数量，并依次输入该数量考生的系列信息，并用此信息完成链表的初始化。然后根据用户输入的操作码实现对应操作，并给出操作码不规范时的提示。

三、实现

3.1 插入功能的实现

1. 插入功能流程图



2. 插入功能核心代码

```
int pos;
std::cout << "请输入你要插入的考生的位置: ";
std::cin >> pos;
if (nameList->insertAtPos(pos) != nullptr) {
    nameList->traverse();
}
break;
```

```
Node *LinkList::insertAtPos(int pos) {

    auto temp = this->header; //头哨兵
    while (--pos) {
        temp = temp->next; //现在要在他的后面插入
        if (temp == this->trailer || nullptr) {
            std::cout << "您输入的位置有误, 请重新输入\n";
        }
    }
    // 这里应该插入新节点
}
```

```

        return nullptr;
    }
}

std::cout << "请依次输入考生的考号、姓名、性别、年龄及报考类别\n";
int number;
std::string name;//姓名
std::string sex;//性别
int age;//年龄
std::string type;//报考科目
std::cin >> number >> name >> sex >> age >> type;
auto stu = new Node(number, name, sex, age, type);
_size++;
temp->insertAsNext(stu);
return stu;
}

```

3. 插入功能示意图

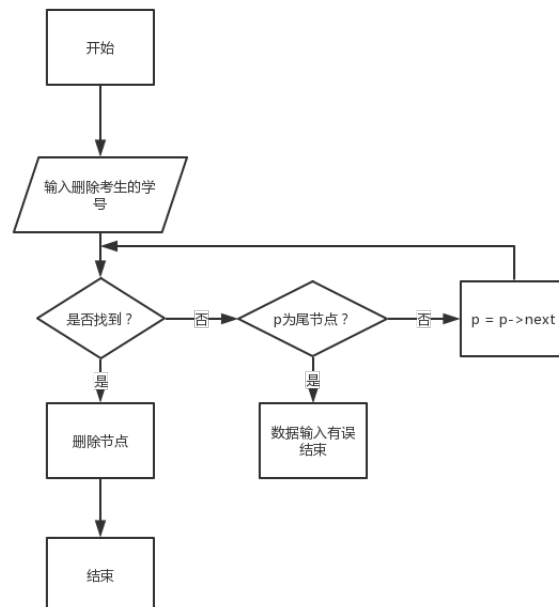
```

首先建立考生信息系统
首先输入考生人数： 3
请依次输入考生的考号、姓名、性别、年龄及报考类别
1 stu1 男 18 科目一
2 stu2 男 22 科目二
3 stu3 女 20 科目三
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，6为退出，0取消操作)：
请选择您要进行的操作： 1
请输入您要插入的考生的位置： 3
请依次输入考生的考号、姓名、性别、年龄及报考类别
4 stu4 女 70 挂科
1 stu1 男 18 科目一
2 stu2 男 22 科目二
4 stu4 女 70 挂科
3 stu3 女 20 科目三
请选择您要进行的操作： 1
请输入您要插入的考生的位置： 7
您输入的位置有误，请重新输入
请选择您要进行的操作：

```


3.2 删除功能的实现

1. 删除功能流程图



2. 删除功能核心代码

```
cout << "请输入要删除的考生的考号:";
cin >> number;
nameList->remove(number);
nameList->traverse();
```

```
void LinkedList::remove(int num) {
    auto p = this->header->next;
    while (p != this->trailer && p->getNumber() != num) {
        p = p->next;
    }
    if (p->getNumber() == num) {
        std::cout << "您要删除的考生信息是: \n";
        p->getInform();
        std::cout << '\n';
        p->pre->next = p->next;
        p->next->pre = p->pre;
        //delete p;
        _size--;
    } else {
        std::cout << "您的输入有误\n";
    }
}
```

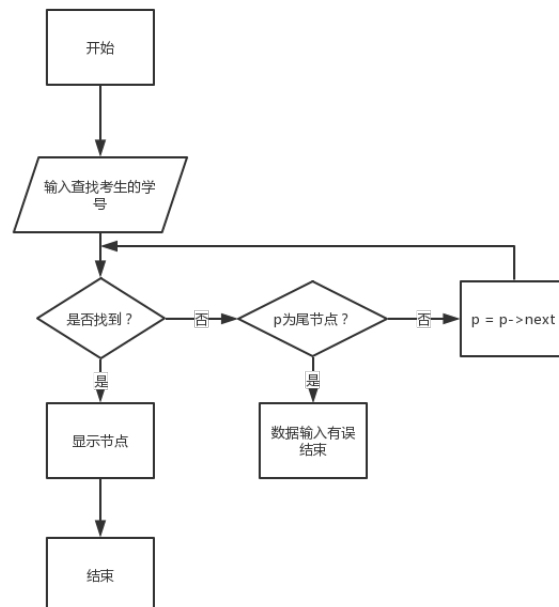
3. 删除功能示意图

```
首先建立考生信息系统
首先输入考生人数： 3
请依次输入考生的考号、姓名、性别、年龄及报考类别
1 stu1 男 18 科目一
2 stu2 男 22 科目二
3 stu3 女 20 科目三
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，6为退出，0取消操作)：
请选择您要进行的操作： 2
请输入要删除的考生的考号：2
您要删除的考生信息是：
2 stu2 男 22 科目二

1 stu1 男 18 科目一
3 stu3 女 20 科目三
请选择您要进行的操作： 2
请输入要删除的考生的考号：2
您的输入有误
1 stu1 男 18 科目一
3 stu3 女 20 科目三
请选择您要进行的操作：
```

3.3 查找功能的实现

1. 查找功能流程图



2. 查找功能核心代码

```
cout << "请输入您要查找的考生的考号：";
cin >> number;
nameList->find(number);
```

```
Node *LinkList::find(int num) {
    auto p = this->header->next;
    while (p != this->trailer && p->getNumber() != num) {
        p = p->next;
    }
    if (p->getNumber() == num) {
        p->getInform();
        return p;
    } else {
        std::cout << "您输入的信息有误\n";
        return nullptr;
    }
}
```

3. 查找功能示意图

首先建立考生信息系统

首先输入考生人数： 3

请依次输入考生的考号、姓名、性别、年龄及报考类别

1 stu1 男 18 科目一

2 stu2 男 22 科目二

3 stu3 女 20 科目三

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, 6为退出, 0取消操作) :

请选择您要进行的操作： 3

请输入您要查找的考生的考号： 3

3 stu3 女 20 科目三

请选择您要进行的操作： 3

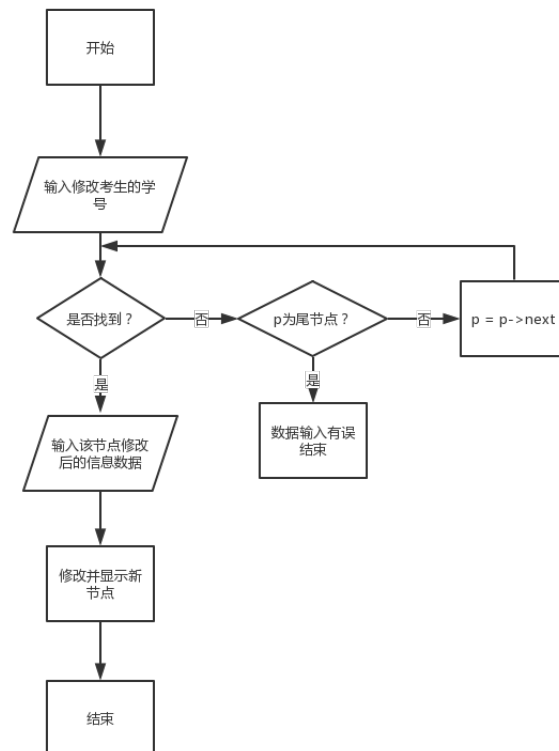
请输入您要查找的考生的考号： 8

您输入的信息有误

请选择您要进行的操作：

3.4 修改功能的实现

1. 修改功能流程图



2. 修改功能核心代码

```
cout << "请输入您要修改的考生的考号: ";
cin >> number;
if (nameList->change(number) != nullptr) {
    nameList->traverse();
}
```

```
Node *LinkedList::change(int num) {
    auto p = this->header->next;
    while (p != this->trailer && p->getNumber() != num) {
        p = p->next;
    }
    if (p->getNumber() == num) {
        p->getInform();
        std::cout << "请依次输入修改之后的学号 姓名 性别 年龄 报考类别: \n";
        p->setInform();
        return p;
    } else {
        std::cout << "您输入的信息有误\n";
        return nullptr;
    }
}
```

```
}  
}
```

3. 修改功能示意图

首先建立考生信息系统

首先输入考生人数： 3

请依次输入考生的考号、姓名、性别、年龄及报考类别

1 stu1 男 18 科目一

2 stu2 男 22 科目二

3 stu3 女 20 科目三

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, 6为退出, 0取消操作) :

请选择您要进行的操作： 4

请输入您要修改的考生的考号： 2

2 stu2 男 22 科目二

请依次输入修改之后的学号 姓名 性别 年龄 报考类别：

4 stu4 女 27 挂科

1 stu1 男 18 科目一

4 stu4 女 27 挂科

3 stu3 女 20 科目三

请选择您要进行的操作： 4

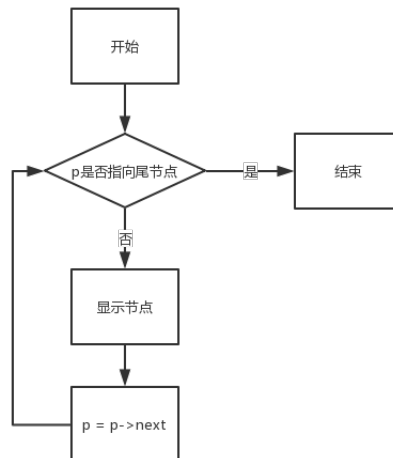
请输入您要修改的考生的考号： 2

您输入的信息有误

请选择您要进行的操作：

3.5 统计功能的实现

1. 统计功能流程图



2. 统计功能核心代码

```
nameList->traverse();
```

```
void LinkedList::traverse() {  
    auto p = this->header->next;  
    for (; p != this->trailer; p = p->next) {  
        p->getInform();  
    }  
}
```

3. 统计功能示意图

首先建立考生信息系统

首先输入考生人数： 3

请依次输入考生的考号、姓名、性别、年龄及报考类别

1 stu1 男 18 科目一

2 stu2 男 22 科目二

3 stu3 女 20 科目三

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, 6为退出, 0取消操作) :

请选择您要进行的操作： 5

1 stu1 男 18 科目一

2 stu2 男 22 科目二

3 stu3 女 20 科目三

请选择您要进行的操作： 1

请输入你要插入的考生的位置： 3

请依次输入考生的考号、姓名、性别、年龄及报考类别

4 stu4 女 23 科目四

1 stu1 男 18 科目一

2 stu2 男 22 科目二

4 stu4 女 23 科目四

3 stu3 女 20 科目三

请选择您要进行的操作： 5

1 stu1 男 18 科目一

2 stu2 男 22 科目二

4 stu4 女 23 科目四

3 stu3 女 20 科目三

请选择您要进行的操作：

3.6 退出操作的实现

1. 退出操作核心代码

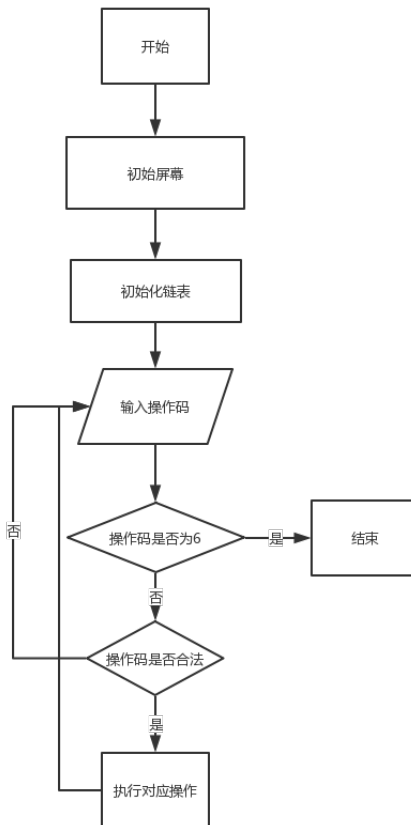
```
if (op == 6) {  
    cout << "您已退出系统。";  
    break;  
}
```

2. 退出操作功能示意图

```
首先建立考生信息系统  
首先输入考生人数： 3  
请依次输入考生的考号、姓名、性别、年龄及报考类别  
1 stu1 男 18 科目一  
2 stu2 男 22 科目二  
3 stu3 女 20 科目三  
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，6为退出，0取消操作)：  
请选择您要进行的操作： 6  
您已退出系统。
```

3.7 总体系统的实现

1. 总体系统流程图



2. 总体系统的核心代码

```
int op = 0;
while (1) {
    cout << "请选择您要进行的操作：";
    cin >> op;
    if (op == 6) {
        cout << "您已退出系统。";
        break;
    }
    switch (op) {
        case 0: {
            break;
        }
        case 1: {
            int pos;
            std::cout << "请输入你要插入的考生的位置： ";
            std::cin >> pos;
            if (nameList->insertAtPos(pos) != nullptr) {
```

```

        nameList->traverse();
    }
    break;
}
case 2: {
    cout << "请输入要删除的考生的考号:";
    cin >> number;
    nameList->remove(number);
    nameList->traverse();
    break;
}
case 3: {
    cout << "请输入您要查找的考生的考号: ";
    cin >> number;
    nameList->find(number);
    break;
}
case 4: {
    cout << "请输入您要修改的考生的考号: ";
    cin >> number;
    if (nameList->change(number) != nullptr) {
        nameList->traverse();
    }
    break;
}
case 5: {
    nameList->traverse();
    break;
}
default: {
    cout << "您输入的数字有误, 请重新输入: ";
    break;
}
}
}

```

3. 总体系统示意图

首先建立考生信息系统

首先输入考生人数： 3

请依次输入考生的考号、姓名、性别、年龄及报考类别

1 stu1 男 18 科目一

2 stu2 男 22 科目二

3 stu3 女 19 科目三

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, 6为退出, 0取消操作) :

请选择您要进行的操作: 1

请输入您要插入的考生的位置: 4

请依次输入考生的考号、姓名、性别、年龄及报考类别

4 stu4 女 25 科目四

1 stu1 男 18 科目一

2 stu2 男 22 科目二

3 stu3 女 19 科目三

4 stu4 女 25 科目四

请选择您要进行的操作: 3

请输入您要查找的考生的考号: 2

2 stu2 男 22 科目二

请选择您要进行的操作: 2

请输入要删除的考生的考号: 3

您要删除的考生信息是:

3 stu3 女 19 科目三

1 stu1 男 18 科目一

2 stu2 男 22 科目二

4 stu4 女 25 科目四

请选择您要进行的操作: 4

请输入您要修改的考生的考号: 2

2 stu2 男 22 科目二

请依次输入修改之后的学号 姓名 性别 年龄 报考类别:

3 stu3 女 19 科目三

1 stu1 男 18 科目一

3 stu3 女 19 科目三

4 stu4 女 25 科目四

请选择您要进行的操作: 6

您已退出系统。

四、测试

4.1 功能测试

1. 插入功能测试

测试用例：

3 stu3 男22 网络工程师

预期结果：

1 stu1 男20 软件开发师

2 stu2 女21 软件测试员

3 stu3 男22 网络工程师

实验结果：

```
首先建立考生信息系统
首先输入考生人数： 2
请依次输入考生的考号、姓名、性别、年龄及报考类别
1 stu1 男 20 软件开发师
2 stu2 女 21 软件测试员
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，6为退出，0取消操作）：
请选择您要进行的操作： 1
请输入你要插入的考生的位置： 3
请依次输入考生的考号、姓名、性别、年龄及报考类别
3 stu3 男 22 网络工程师
1 stu1 男 20 软件开发师
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
```

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 软件测试员

3 stu3 男22 网络工程师

实验结果：

```
首先建立考生信息系统
首先输入考生人数： 4
请依次输入考生的考号、姓名、性别、年龄及报考类别
1 stu1 男 20 软件开发师
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
4 stu4 男 25 软件架构师
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，6为退出，0取消操作)：
请选择您要进行的操作： 2
请输入要删除的考生的考号：4
您要删除的考生信息是：
4 stu4 男 25 软件架构师

1 stu1 男 20 软件开发师
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
```

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 软件测试员

实验结果：

```
请依次输入考生的考号、姓名、性别、年龄及报考类别
1 stu1 男 20 软件开发师
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
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，6为退出，0取消操作)：
请选择您要进行的操作： 3
请输入您要查找的考生的考号： 2
2 stu2 女 21 软件测试员
```

4. 修改功能测试

测试用例：

将考号1修改为性别女，年龄20，报考种类移动开发员。

预期结果：

1 stu1 女 20 移动开发员

实验结果：

```
请选择您要进行的操作：4
请输入您要修改的考生的考号：1
1 stu1 男 20 软件开发师
请依次输入修改之后的学号 姓名 性别 年龄 报考类别：
1 stu1 女 20 移动开发员
1 stu1 女 20 移动开发员
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
```

5. 统计功能测试

测试用例：

统计当前数据

预期结果：

1 stu1 女 20 移动开发员

2 stu2 女 21 软件测试员

3 stu3 男 22 网络工程师

实验结果：

```
请选择您要进行的操作：5
1 stu1 女 20 移动开发员
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
```

4.2 边界测试

1. 初始化无输入数据

测试用例：

初始化无输入数据

预期结果：

给出错误提示，程序运行正常不崩溃。

实验结果：

```
首先建立考生信息系统
首先输入考生人数： 0
请输入一个正整数： |
```

2. 删除头节点

测试用例：

删除头节点

预期结果：

程序正常运行，不崩溃。

实验结果：

```
请依次输入考生的考号、姓名、性别、年龄及报考类别
1 stu1 男 20 软件开发师
2 stu2 女 21 软件测试员
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，6为退出，0取消操作)：
请选择您要进行的操作： 2
请输入要删除的考生的考号：1
您要删除的考生信息是：
1 stu1 男 20 软件开发师

2 stu2 女 21 软件测试员
请选择您要进行的操作：
```

3. 删除后链表为空

测试用例：

删除前链表只有一个节点，删除后链表为空

预期结果：

程序正常运行，不崩溃。

实验结果：


```
2 stu2 女 21 软件测试员
请选择您要进行的操作：2
请输入要删除的考生的考号：2
您要删除的考生信息是：
2 stu2 女 21 软件测试员
请选择您要进行的操作：
```

4.3 出错测试

1. 考生人数错误

测试用例：

输入的 考生人数为负数

预期结果：

程序给出错误提示，正常运行不崩溃。

实验结果：

```
首先建立考生信息系统
首先输入考生人数： -2
请输入一个正整数：
```

2. 操作码错误

测试用例：

输入的操作码错误

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：

```
请依次输入考生的考号、姓名、性别、年龄及报考类别
1 stu1 女 20 移动开发员
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
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，6为退出，0取消操作)：
请选择您要进行的操作： 7
您输入的数字有误，请重新输入：请选择您要进行的操作：|
```

3. 插入位置不存在

测试用例：

在链表里仅有3个考生信息时，向第5个位置插入信息

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：

```
1 stu1 女 20 移动开发员
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
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, 6为退出, 0取消操作) :
请选择您要进行的操作: 1
请输入您要插入的考生的位置: 5
您输入的位置有误, 请重新输入
```

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:

程序给出提示信息, 程序正常运行不崩溃。

实验结果:

```
请选择您要进行的操作: 2
请输入要删除的考生的考号: 5
您的输入有误
1 stu1 女 20 移动开发员
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
请选择您要进行的操作:
```

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:

程序给出提示信息, 程序正常运行不崩溃。

实验结果:

```
1 stu1 女 20 移动开发员
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
请选择您要进行的操作: 3
请输入您要查找的考生的考号: 6
您输入的信息有误
请选择您要进行的操作:
```

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:

程序给出提示信息, 程序正常运行不崩溃。

实验结果：

```
1 stu1 女 20 移动开发员
2 stu2 女 21 软件测试员
3 stu3 男 22 网络工程师
请选择您要进行的操作： 4
请输入您要修改的考生的考号： 4
您输入的信息有误
请选择您要进行的操作：
```