项目说明文档

数据结构课程设计

——算术表达式求解

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

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

平时我们使用的运算表达式就是中缀表达式，例如**1+3*2**，中缀表达式的特点就是：**二元运算符总是置于与之相关的两个运算对象之间**。这种表达式人读起来比较好理解，但是计算机处理起来就很麻烦，运算顺序往往因表达式的内容而定，不具规律性。所以为了方便计算机求解中缀表达式的值，我们考虑将中缀表达式转换成计算机容易理解的格式，再进行表达式的求值运算。

1.2 项目功能要求

相比于中缀表达式，明显具有计算优势的一种表达式是后缀表达式（又名逆波兰表达式）。后缀表达式的特点就是：每一运算符都置于其运算对象之后，且表达式中不含有括号，运算符的顺序与计算顺序相关，以上面的中缀表达式**1+2*3**为例子，转为后缀表达式就是**123*+**。对于生成的后缀表达式，我们可以很容易的根据读取的运算符进行对运算符之前的数字的计算求值。

二、设计

2.1 数据结构设计

由分析得，我们主要需要进行的操作是

1. 将中缀表达式转化为后缀表达式，

2. 求后缀表达式的值

但因为不是所有的表达式都是由简单的个位数配合双目运算符组成，除了体重所给出的单目运算符存在，本题还需要处理一个步骤：

0. 将中缀表达式进行预处理

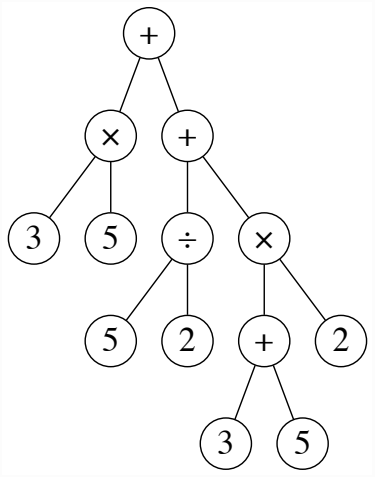
其中，需要特别处理的有以下部分：

1. 前置单目运算符“+”、“-”的处理

2. 多位数的处理

3. 小数的处理(包含对小数做取余运算时的取整处理)

对于预处理后的中缀表达式，将其转化成后缀表达式，有两种常见方法。我们在数据结构课程上有学到，对于一个二叉树，如果它的叶节点存放的是数字，其余节点存放操作符的话，那么这个二叉树的中序遍历序列会得到一个中缀表达式（需在某些地方手动添加括号使表达式逻辑正确），相应的，该二叉树的前序遍历序列对应该算术表达式的前缀表达式，后序遍历序列对应后缀表达式。以表达式**3×5+5÷2+(3+5)×2**为例，将其表示成二叉树形式，如下图所示：



将中缀表达式表示为二叉树形式的过程如下：

1. 找出表达式中最后进行运算的操作符，即优先级最低的操作符，作为根节点

2. 在步骤 1 找出的操作符的左侧部分找出最后进行运算的操作符，作为步骤 1 根节点的左子节点；右侧部分进行同样的操作
3. 若步骤 1 找出的操作符左侧或右侧部分只含有操作数，不含有操作符，则将该操作数作为对应的叶子节点。
4. 不断重复上述过程直到二叉树包含表达式的所有操作数和操作符

建立完毕表达式的二叉树，从叶子节点开始对二叉树进行后序遍历（左子树 -> 右子树 -> 根节点），即可得到中缀表达式对应的后缀表达式。

而二叉树的遍历分为递归法与非递归法，对于二叉树的中序遍历和后序遍历，我们都可以用**栈（Stack）**这种数据结构来实现二叉树的非递归遍历。同样的，为了方便表达式的转化，我们可以直接只用栈来实现中缀表达式转化为后缀表达式，从而避免二叉树的构建。该过程与计算中缀表达式的过程类似，使用一个栈保存操作符，一个队列结构用于保存转换得到的后缀表达式。其过程如下：

1. 从头到尾扫描表达式，若遇到操作符，则与操作符栈的栈顶元素比较优先级：如果当前操作符优先级高于栈顶元素的优先级，则压入操作符栈；否则不断弹出操作符栈顶元素压入结果队列，直到栈顶元素的优先级低于当前操作符的优先级，将当前操作符压入操作符栈
2. 若遇到操作数，则直接压入结果队列
3. 若遇到左括号，则压入操作符栈
4. 若遇到右括号，则依次弹出操作符栈顶元素压入结果队列，直到遇到左括号，将左括号弹出操作符栈

对于后缀表达式的计算，实现思路是：

1. 如果是操作数，那么将其压入"结果栈"中
2. 如果是操作符，从"结果栈"中取出两个元素，进行计算。（注意从栈中取元素的顺序和操作数的顺序是相反的）遍历后缀表达式结束后，"结果栈"中的元素就是最终的结果。

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（LinkList）。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（LinkList）

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：

```
LinkList();
~LinkList();

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
```

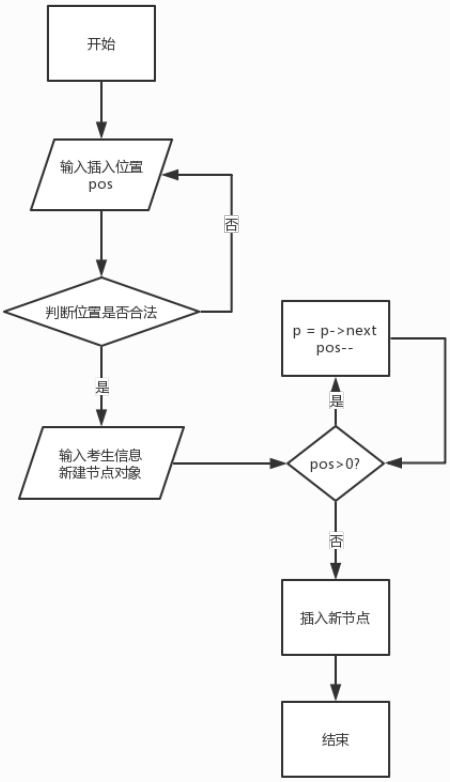
2.4 系统设计

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) {
```

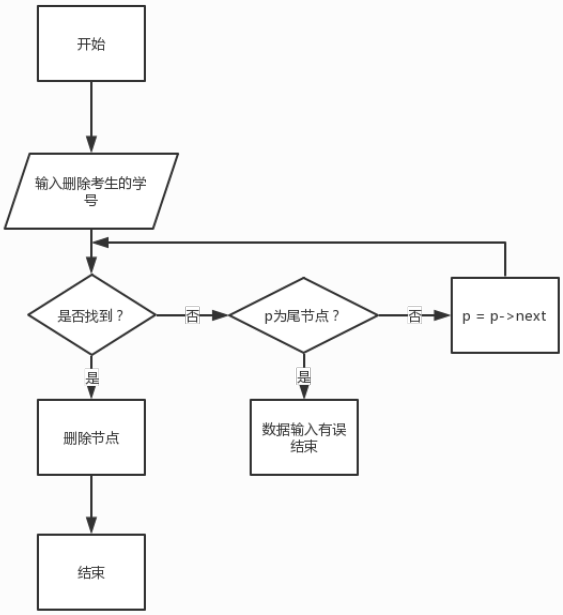
```
            std::cout << "您输入的位置有误，请重新输入\n";
            return nullptr;
        }
    }

    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 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
请输入你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();
```

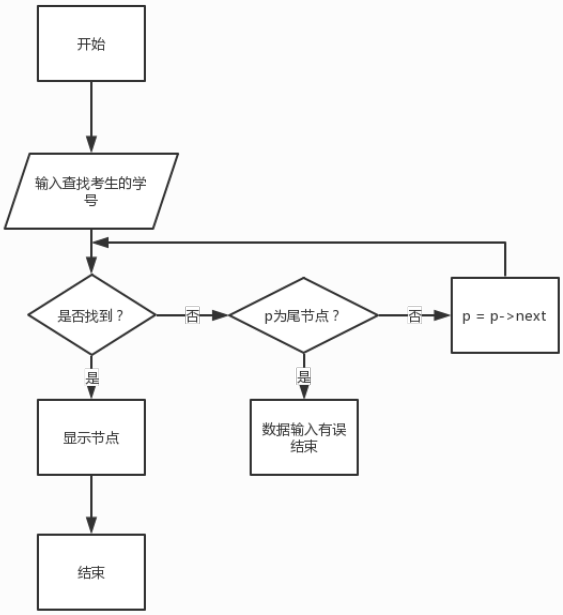
```
void LinkList::remove(int num) {
    auto p = this->header->next;
    while (p != this->trailer && p->getNumber() != num) {
        p = p->next;
    }
    if (p->getNumber() == num) {
        std::cout << "您要删除的考生信息是: \n";
        p->getInform();
        std::cout << '\n';
        p->pre->next = 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);
```

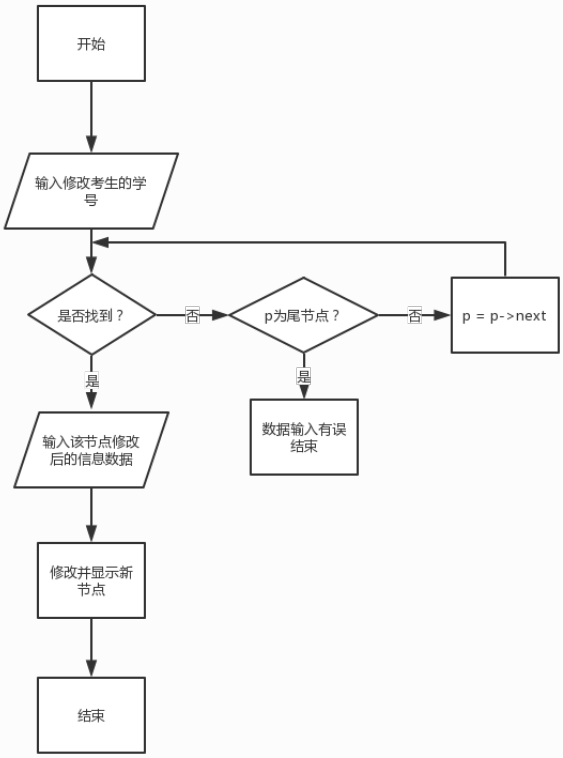
```
Node *LinkList::find(int num) {
    auto p = this->header->next;
    while (p != this->trailer && p->getNumber() != num) {
        p = p->next;
    }
    if (p->getNumber() == num) {
        p->getInform();
        return p;
    } else {
        std::cout << "您输入的信息有误\n";
        return nullptr;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

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 *LinkList::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;
```

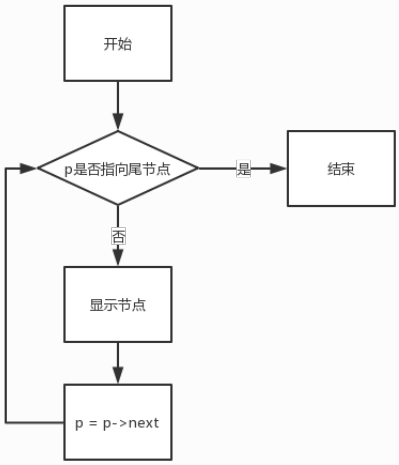
```
    }
}
```

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 LinkList::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

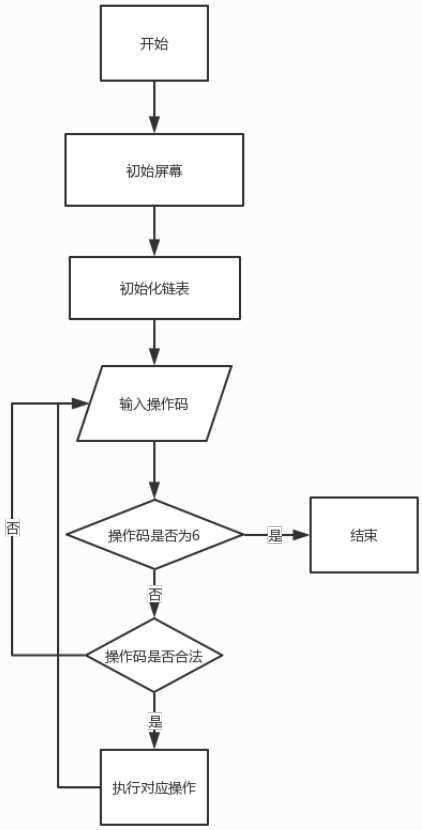
```
if (op == 6) {
    cout << "您已退出系统。";
    break;
}
```

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
请输入你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

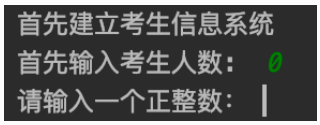
测试用例：

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：



2. 删除头节点

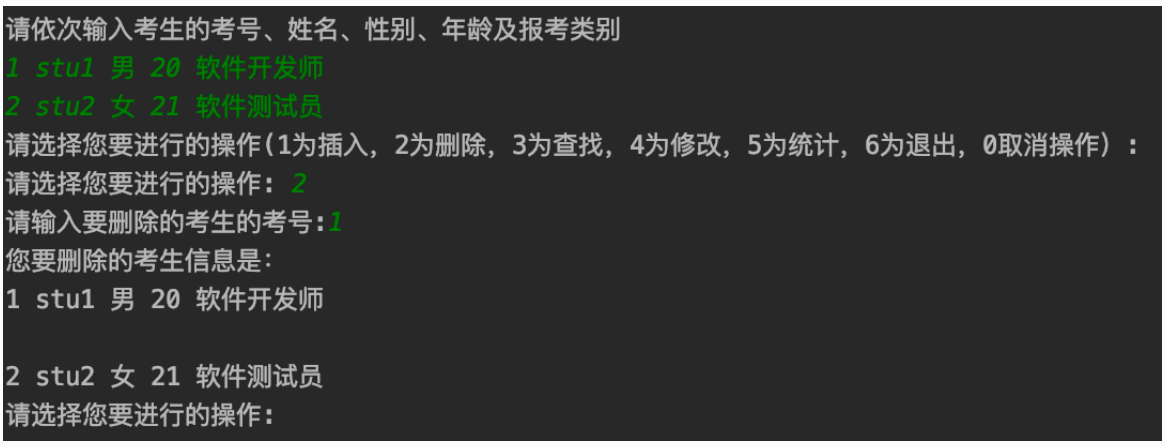
测试用例：

删除头节点

预期结果：

程序正常运行，不崩溃。

实验结果：



3. 删除后链表为空

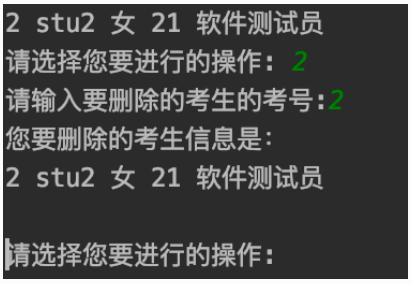
测试用例：

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：



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
请输入你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：