同濟大學
TONGJI UNIVERSITY

项目说明文档

考试报名系统

指导老师：张颖

1851009 沈益立

目录

1.分析

1.1 背景分析

考试注册给高校注册带来了新的挑战，给教育行政部门带来了很大的压力。因此，一个好的注册系统应该能够为用户提供足够的信息和功能。考试注册系统对于学校加强考试管理至关重要，随着学生人数和考试数量的增长，管理如此大量的数据变得非常复杂。传统的人工管理非常繁琐且容易出错。

随着计算机科学和技术的不断成熟，使用计算机管理的考试注册系统的好处无与伦比。这些优势可以大大提高学校和学生的工作效率，也是学校实现信息化，科学化和国际化的重要条件。因此，开发考试注册系统非常重要。

1.2 功能分析

作为最简单的考试注册系统之一，需要的第一个功能是输入和查看学生的考试注册状态。其次，考试注册系统需要插入，删除和修改功能，以便学生可以随时更改其考试报名情况。最后，考试注册系统软件还必须确保可以正常关闭该软件。

简而言之，考试报名系统至少需要输入，输出，插入，删除，修改和退出功能。

2.设计

2.1 数据结构设计

如上所述，该系统要求大量的增删改查操作，而对于链表——在内存上离散分布的一种顺序存储结构来说，增加、删除等操作十分简便。因此，考虑使用单链表作为存储学生信息的数据结构。

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（Node）与链表类（LinkedList），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。本系统采用嵌套的方法，用class Node描述链表结点类（Node），这样使得链表结点类（LinkedList）可以访问链表结点，使用Node的成员变量，也使得Node有其独特的方法，开放接口给main函数使用。

2.3 成员和操作设计

链表节点类（Node）

公有操作：

```
Node(string stuNum, string name, string sex,     //含参构造函数
     int age, string type, Node* next) :
m_stuNum(stuNum), m_name(name), m_sex(sex),
m_age(age), m_type(type), m_next(next) {};
Node() {};                                       //不含参的默认构造函数
void show();                                     //格式化地输出该节点的考生信息
```

公有成员：

```
string m_stuNum = "";                                    //考号
string m_name = "";                                      //姓名
string m_sex = "";                                       //性别
int m_age = 0;                                           //年龄
string m_type = "";                                      //类别
Node* m_next = NULL;                                     //后继节点
地址
```

链表类LinkedList

公有操作：

```
LinkedList() {};                                         //构造函数
~LinkedList();                                           //析构函
数，会释放所有节点的内存，以防内存泄漏
Node* search(const string stuNum) const;                 //查找函
数，以考号为键值，从链表的头结点向后依次查找该考生，返回其地址
bool insert(int index, const string& stuNum,             //插入函
数，传入插入的目标位置和考生的参数，将新考生插入到链表指定位置
            const string& name, const string& sex,
            int& age, const string& type);
Node* erase(string stuNum);                              //删除函
数，以考号为键值，从链表的头结点向后
//依次查找该考生，若找到则将其从链表中移除，并返回该节点地址。
bool showCurrentList() const;                            //格式化地
输出当前表单情况
int getLength() const { return m_length; }               //返回链表
的长度，该接口被主函数调用
```

私有成员：

```
int m_length = 0;                                        //链表长度
Node* m_head = nullptr;                                  //头结点，
默认为空指针
```

2.4 系统设计

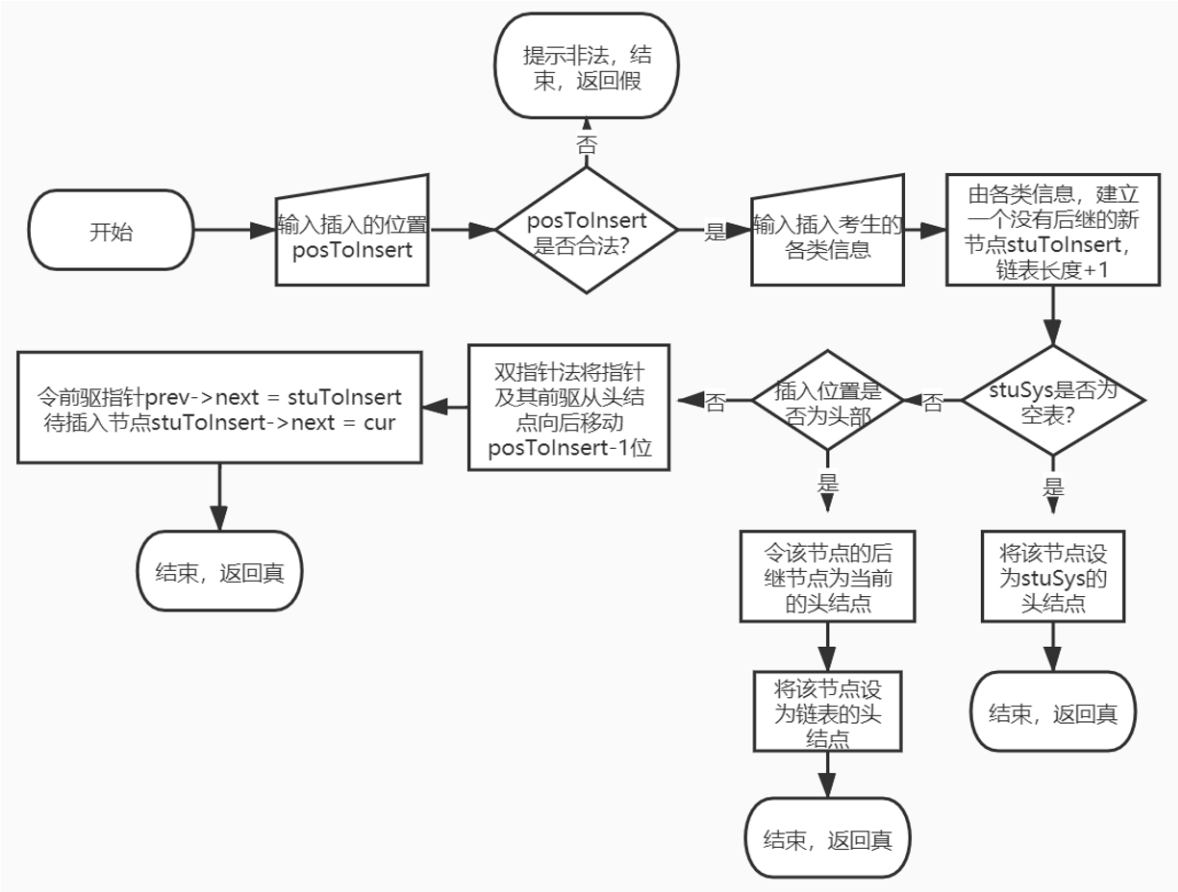
系统调用基本的IO、构造函数和链表的插入成员函数，使用尾插法完成对学生系统链表stuSys的创建和输入数据工作，然后根据用户所输入的操作码(optCode)执行链表stuSys对应的成员函数。

本系统将IO与成员函数分离，并且有较好的低耦合性。

3.实现

3.1 插入功能的实现

3.1.1 插入功能流程图



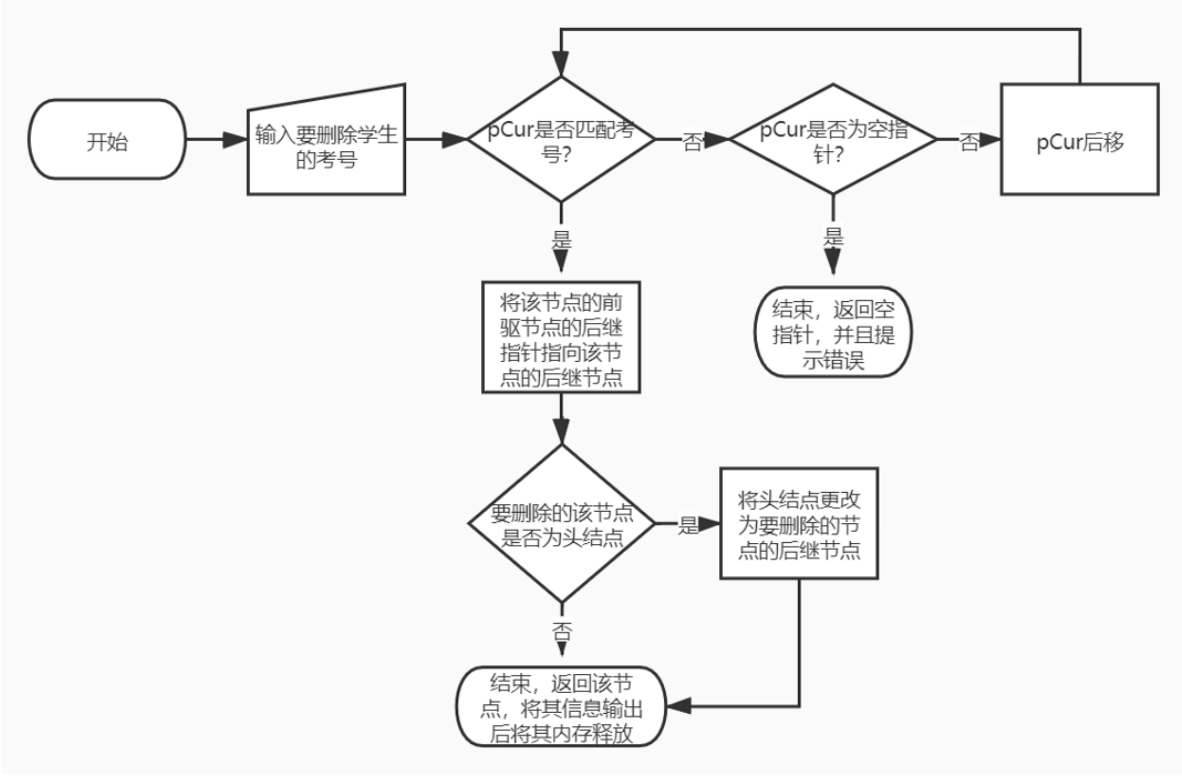
3.1.2 插入功能核心代码实现

```
bool LinkedList::insert(int index, const string& stuNum,
                        const string& name, const
string& sex,
                        int& age, const string& type)
{
    //双指针法插入链表，若插入失败，会返回false
    if (index > m_length + 1 || index <= 0)
    {
```

```
            return false;
        }
        //new Node object
        Node* stuToInsert = new Node(stuNum, name, sex,
 age, type, nullptr);
        m_length++;
        if (!m_head)
        {
            m_head = stuToInsert;
            return true;
        }
        else if (index == 1)
        {
            stuToInsert->m_next = m_head;
            m_head = stuToInsert;
            return true;
        }
        //insertion
        Node* fakeHead = new Node;
        fakeHead->m_next = m_head;
        //double pointer
        Node* prev = fakeHead;
        Node* cur = m_head;
        for (int i = 0; i < index - 1; i++)
        {
            prev = cur;
            cur = cur->m_next;
        }
        prev->m_next = stuToInsert;
        stuToInsert->m_next = cur;
        return true;
}
```

3.2 删除功能的实现

3.2.1 删除功能流程图



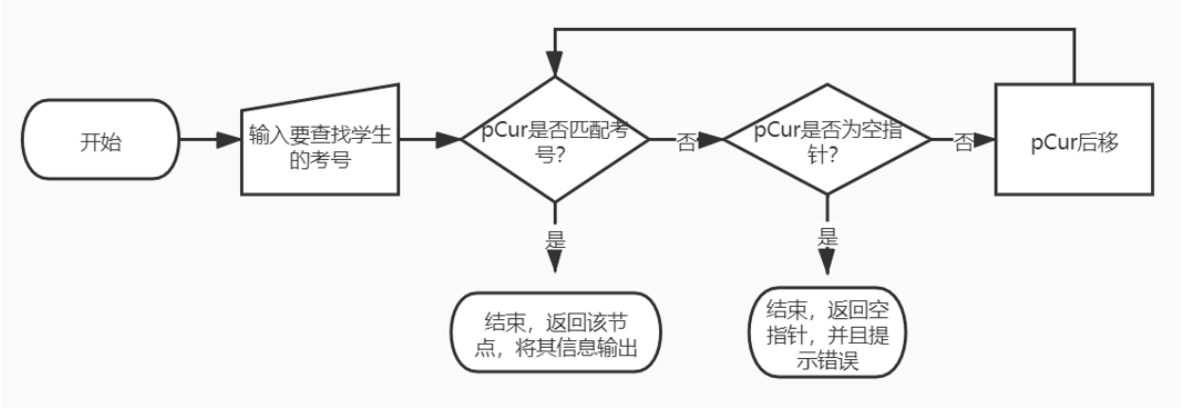
3.2.2 删除功能核心代码实现

```
Node* LinkedList::erase(const string& stuNum)
{
    Node* fakeHead = new Node();
    fakeHead->m_next = m_head;
    Node* pCur = m_head, * prev = fakeHead;
    while (pCur)
    {
        if (pCur->m_stuNum == stuNum)
        {
            prev->m_next = pCur->m_next;
            if (m_head == pCur)
            {
                m_head = pCur->m_next;
            }
            return pCur;
        }
        prev = pCur;
        pCur = pCur->m_next;
    }
```

```
    return nullptr;
}
```

3.3 查找数据功能的实现

3.3.1 查找数据功能流程图

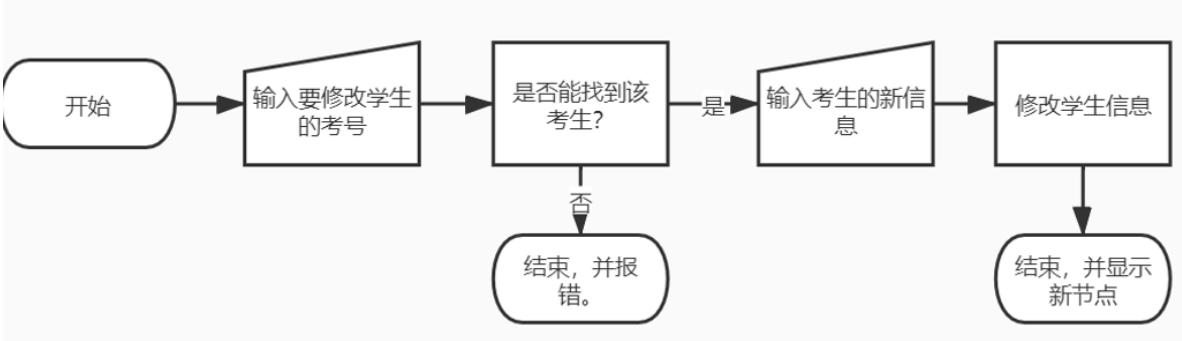


3.3.2 查找数据核心代码实现

```
Node* LinkedList::search(const string& stuNum) const
{
    //按学号检索学生，若找到则返回该节点的地址，否则返回一个空指针
    Node *pCur = m_head;
    while (pCur)
    {
        if (pCur->m_stuNum == stuNum)
        {
            return pCur;
        }
        pCur = pCur->m_next;
    }
    //若已查到该学生，则之前的while循环中将返回。
    //未查到该学生，返回空指针
    return nullptr;
}
```

3.4 修改数据功能的实现

3.4.1 修改数据功能流程图



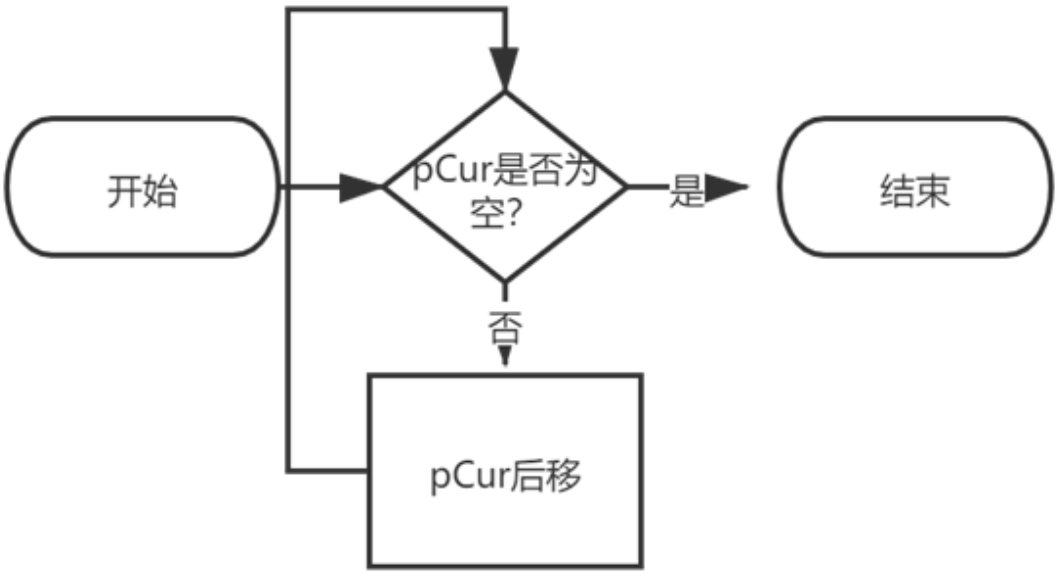
3.4.2 修改数据核心代码实现

```
cout << "请输入要修改的考生的考号：";
string stuNum;
cin >> stuNum;
Node* stuToChange = stuSys->search(stuNum);
if (!stuToChange)
{
    cout << "未找到该学生信息。" << endl;
}
else
{
    //修改
    cout << "请依次输入该考生修改后的的考号、"
         << "姓名、性别、年龄和报考类别" << endl;
    string stuNum, name, sex, type;
    int age;
    cin >> stuNum >> name >>
        sex >> age >> type;
    stuToChange->m_stuNum = stuNum;
    stuToChange->m_name = name;
    stuToChange->m_sex = sex;
    stuToChange->m_age = age;
    stuToChange->m_type = type;
    stuSys->showCurrentList();
```

```
}
```

3.5 统计人数功能的实现

3.5.1 统计人数功能流程图



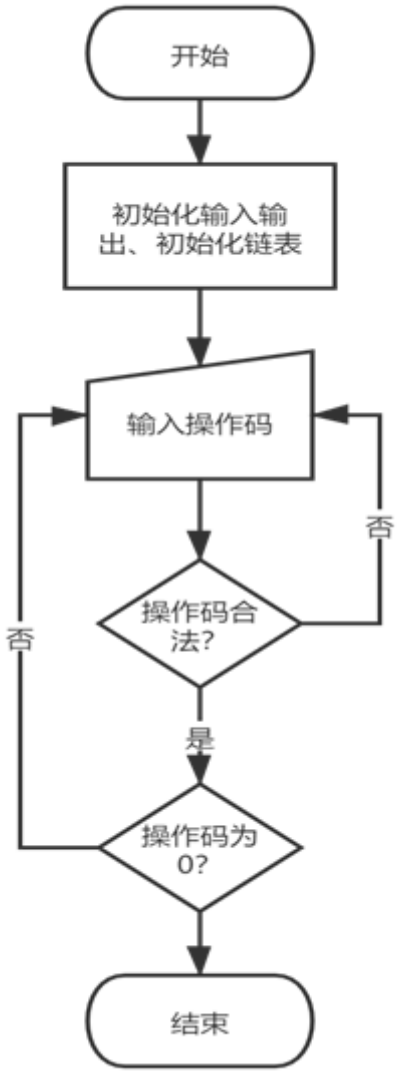
3.5.2 修改数据核心代码实现

```
bool LinkedList::showCurrentList() const
{
    printf("考号\t姓名\t性别\t年龄\t报考类别\n");
    Node* pCur = m_head;
    while (pCur)
    {
        cout << pCur->m_stuNum  << "\t"
             << pCur->m_name    << "\t"
             << pCur->m_sex          << "\t"
             << pCur->m_age          << "\t"
             << pCur->m_type    << endl;
        pCur = pCur->m_next;
    }
    return true;
```

```
}
```

3.6 整体系统功能的实现

3.6.1 整体系统功能流程图



3.6.2 整体系统核心代码实现

```
int optCode;
    while (1)
    {
        cout << "请选择要进行的操作：";
```

```
        cin >> optCode;
        if (optCode == 1)
        {
            //插入考生
            cout << "请输入要插入考生的位置：";
            int posToInsert;
            cin >> posToInsert;
            if (posToInsert > stuSys->getLength() + 1 ||
posToInsert <= 0)
            {
                cout << "考生位置不合法。" << endl;
                continue;
            }
            cout << "请依次输入要插入的考生的考号、"
                << "姓名、性别、年龄和报考类别" << endl;
            string stuNum, name, sex, type;
            int age;
            cin >> stuNum >> name >> sex >> age >> type;
            stuSys->insert(posToInsert, stuNum, name, sex,
age, type);
            stuSys->showCurrentList();
        }

        else if (optCode == 2)
        {
            //删除考生
            cout << "请输入要删除考生的考号：";
            string stuNum;
            cin >> stuNum;
            Node* stuToDel = stuSys->erase(stuNum);
            //若未找到
            if (stuToDel == nullptr)
            {
                cout << "不存在该考生！" << endl;
            }
            else
            {
                cout << "你删除的考生信息是：";
                stuToDel->show();
                cout << endl;
                delete stuToDel;
```

```
                stuSys->showCurrentList();
            }
        }

        else if (optCode == 3)
        {
            //查找考生
            cout << "请输入要查找的考生的考号：";
            string stuNumToSearch;
            cin >> stuNumToSearch;
            Node* stuSearched = stuSys-
>search(stuNumToSearch);
            //若未找到
            if (!stuSearched)
            {
                cout << "未找到该学生信息。" << endl;
            }
            else
            {
                printf("考号\t姓名\t性别\t年龄\t报考类别\n");
                stuSearched->show();
            }
        }

        else if (optCode == 4)
        {
            //修改考生信息
            cout << "请输入要修改的考生的考号：";
            string stuNum;
            cin >> stuNum;
            Node* stuToChange = stuSys->search(stuNum);
            if (!stuToChange)
            {
                cout << "未找到该学生信息。" << endl;
            }
            else
            {
                //修改
                cout << "请依次输入该考生修改后的的考号、"
                    << "姓名、性别、年龄和报考类别" << endl;
                string stuNum, name, sex, type;
```

```
                int age;
                cin >> stuNum >> name >>
                    sex >> age >> type;
                stuToChange->m_stuNum = stuNum;
                stuToChange->m_name = name;
                stuToChange->m_sex = sex;
                stuToChange->m_age = age;
                stuToChange->m_type = type;
                stuSys->showCurrentList();
            }
        }

        else if (optCode == 5)
        {
            //统计表格
            cout << "目前的表格情况：" << endl;
            stuSys->showCurrentList();
        }

        else if (optCode == 0)
        {
            cout << "取消操作" << endl;
            break;
        }
        else
        {
            //default
            cout << "请输入正确的操作号。" << endl;
        }
        cout << endl;
    }
```

4.测试

4.1 常规测试

4.1.1 初始化测试

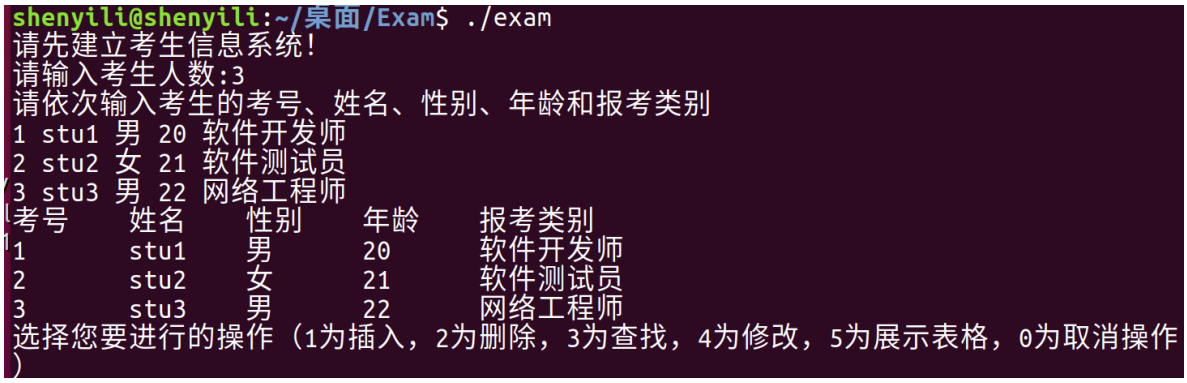
测试用例：

```
3
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
```

预期结果：

```
考号   姓名   性别   年龄   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
```

实验结果：



4.1.2 插入功能测试

测试用例：

```
1
4
4 stu4 男 19 初级码农
```

预期结果：

```
考号   姓名   性别   年龄   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    19   初级码农
```

实验结果：

```
请依次输入要插入的考生的考号、姓名、性别、年龄和报考类别
4 stu4 男 19 初级码农
考号     姓名     性别     年龄     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       19       初级码农
```

4.1.3 删除功能测试

测试用例：

```
2
2
```

预期结果：

```
删除成员：2   stu2     女    21   软件测试员
更新后的注册系统：
考号   姓名   性别   年龄   报考类别
1    stu1     男    20   软件开发师
3    stu3     男    22   网络工程师
4    stu4     男    19   初级码农
```

实验结果：

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

考号    姓名    性别    年龄    报考类别
1       stu1    男      20      软件开发师
3       stu3    男      22      网络工程师
4       stu4    男      19      初级码农
```

4.1.4 查找功能测试

测试用例：

```
3
3
```

预期结果：

```
3    stu3    男    22    网络工程师
```

实验结果：

```
请选择要进行的操作：3
请输入要查找的考生的考号：3
考号    姓名    性别    年龄    报考类别
3       stu3    男      22      网络工程师
```

4.1.5 修改功能测试

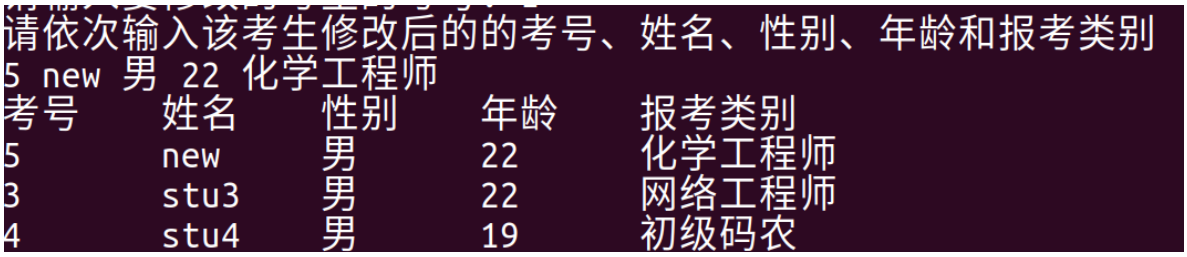
测试用例：

```
4
1
5 new 男 22 化学工程师
```

预期结果：

```
修改后：
考号    姓名    性别    年龄    报考类别
5       new     男      22      化学工程师
3       stu3    男      22      网络工程师
4       stu4    男      19      初级码农
```

实验结果：



4.1.6 统计功能测试

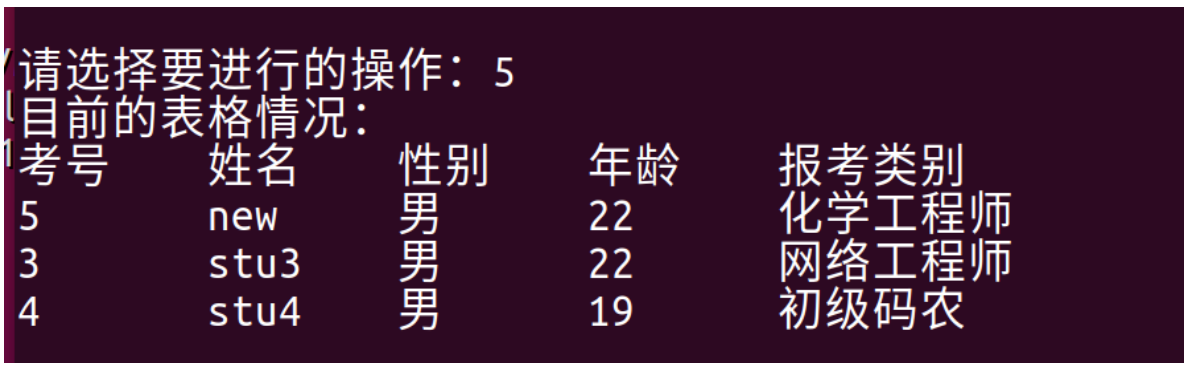
测试用例：

```
5
```

预期结果：

```
考号    姓名    性别    年龄    报考类别
5       new     男      22      化学工程师
3       stu3    男      22      网络工程师
4       stu4    男      19      初级码农
```

实验结果：



4.2 错误测试

4.2.1 考生人数错误

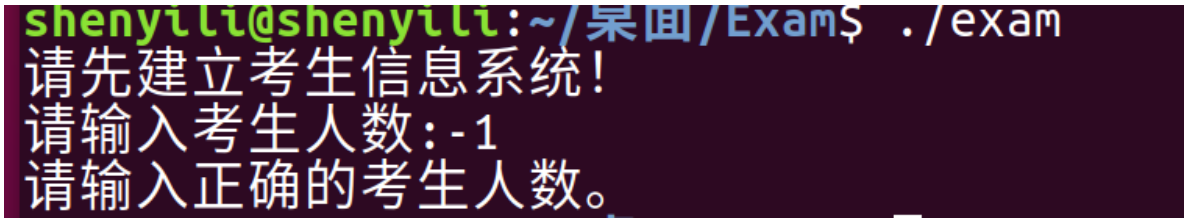
测试用例：

```
-1
```

预期结果：

```
报错考生人数错误
```

实验结果：



4.2.2 操作码输入错误

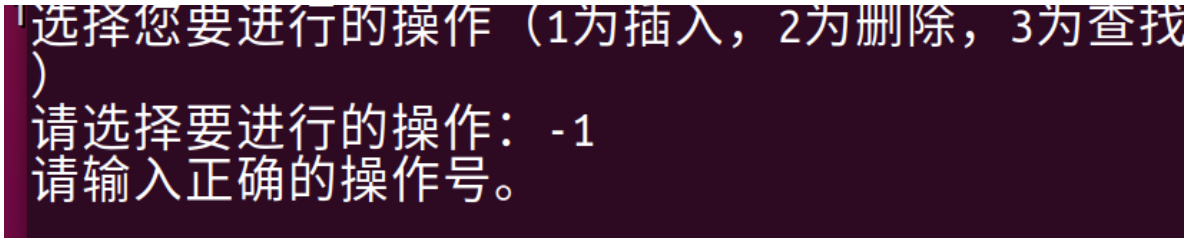
测试用例：

```
-1
```

预期结果：

```
报错操作码错误
```

实验结果：



4.2.3 插入位置不存在

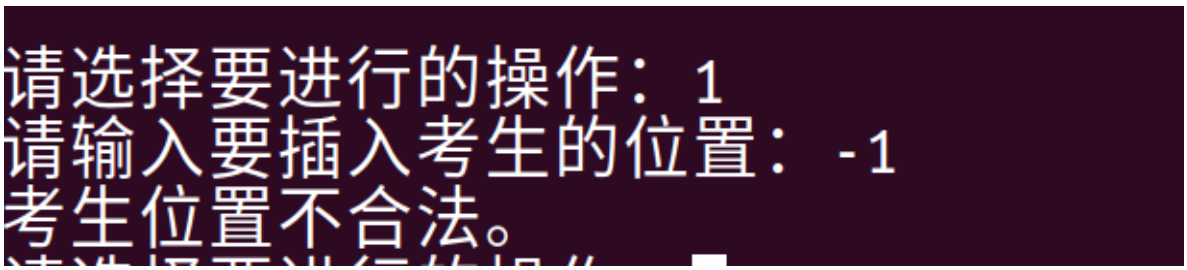
测试用例：

```
1
-1
```

预期结果：

```
报错插入位置错误
```

实验结果：



4.2.4 删除考生不存在

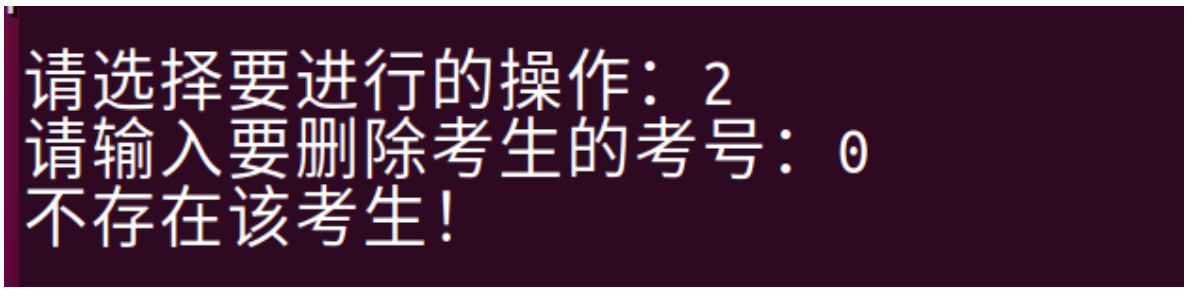
测试用例：

```
2
0
```

预期结果：

```
报错考生不存在
```

实验结果：



4.2.5 查找考生不存在

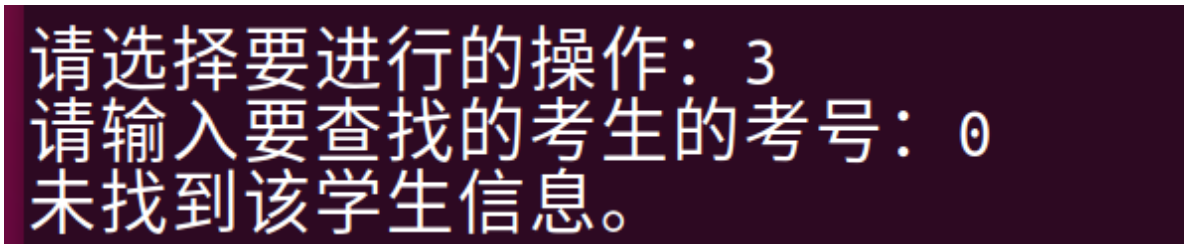
测试用例：

```
3
0
```

预期结果：

```
报错考生不存在
```

实验结果：



4.2.6 修改考生不存在

测试用例：

```
4
0
```

预期结果：

```
报错考生不存在
```

实验结果：