项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名：翟晨昊

学　　号：1952216

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

1.1 背景分析

在学校里，考试是必不可少的一环，而考试报名作为举行考试流程的第一项任务，自然也就十分重要，如果报名过程中出现错误或者混乱，很有可能会影响到考试的正常举行，进而影响到学生利益与学校计划。因此一个良好的考试报名系统对学校和学生来说都是至关重要的，现如今随着学生数量的增多与考试所需数据的日益繁杂，传统的手工管理方法不仅工作量大，还比较容易出错，与学校与学生的高要求出现了矛盾。如何能够快速准确的进行考试信息管理，是目前的考试报名系统需要解决的关键问题。

随着计算机科学技术的高速发展，使用计算机来对考试报名系统进行管理，相较于传统手工管理方法，具有信息容量大，出错率低，管理快速，维护便捷等多种优势。这些优势都可以使得考试报名更有效率，帮助考试能够正常进行，能够保障学生利益与学校计划的顺利进行。因此，开发一个基于计算机操作的考试报名系统是非常有必要的。

1.2 功能分析

作为一个比较简单的考试报名系统，其基础的功能是能够接收输入的考生信息并且储存下来，同时也能够显示已经保存的考生信息。在此基础上，考试报名系统还具有插入，删除，修改功能，能够在系统里实时更改或添加考生信息，以便考生可以随时更改自己的报名情况；系统还拥有查询功能，可以让管理人员和考生能够迅速查到某位考生的信息；最后，考试报名系统还要能够被正常关闭。删除和查询功能最好用考号和所在位置两种方式均可以实现，减少因为忘记信息而导致操作无法进行的情况发生。

综上所述，一个考试报名系统需要有输入、输出、插入、按考号查询、按位置查询、按考号删除、按位置删除、修改、退出的功能。

2 设计

2.1 数据结构设计

如上功能分析所述，该考试报名系统要求大量增加、删除、修改操作，使用数组，链表等数据结构都可以完成这些操作。但数组在进行这些操作的时候可能会移动整个数组，太过浪费时间，相比较之下，链表在进行增加，删除等操作时非常简便，因此最后选择使用双向链表来维护信息，同时在链表前附加了一个头结点，这样使得增加和删除头结点时与处理其它结点方法相同，简化了代码。

2.2 类结构设计

链表一般包括两个抽象数据类型——链表结点类（Node 类）与链表类（LinkList 类），Node 类来储存每一位考生信息，LinkList 类来将考生信息整合在一起，并提供查询，插入，删除，修改等操作。这两个类通过友元来建立起联系，这样使得 LinkList 类可以访问 Node 类。除此之外，本系统还设计了 Student 类与 InformationSystem 类，分别用来储存考号姓名等信息，以及给用户提供与系统交互的接口。同时为了减少接口，直接将 InformationSystem 类声明为 Node 类的友元，使得其可以直接访问 Node 类的数据信息。为了使链表更加具有泛用性，本系统将 Node 类与 LinkList 类都设计为了模板类。

2.3 成员与操作设计

链表结点类（Node）：

```
template <typename ElementType>
class Node {
public:
    Node() = default;
    Node(ElementType& inputStu) :data(inputStu) {}
    friend class LinkList<ElementType>;
    friend class InformationSystem<ElementType>;
private:
    ElementType data;
    Node<ElementType>* next = nullptr;
    Node<ElementType>* prev = nullptr;
};
```

私有成员：

ElementType data;//链表结点中的学生信息
Node<ElementType>* next;//指针域，表示该结点的下一个结点
Node<ElementType>* prev;//指针域，表示该结点的上一个结点

公有操作：

Node () = default;

//默认构造函数

Node(ElementType& inputStu);
//含输入信息的构造函数

friend class LinkList<ElementType>;
//将 LinkList 声明为友元

friend class InformationSystem<ElementType>;
//将 InformationSystem 类声明为友元

链表类（LinkList）：

```
template <typename ElementType>
class LinkList {
public:
    LinkList()
    {
        head = new Node<ElementType>;
        size = 0;
    }
    ~LinkList();
    int getSize();
    void append(ElementType& inputStu, int position);
    Node<ElementType>* findByID(string studentID);
    Node<ElementType>* findByLocation(int position);
    void pop(Node<ElementType>* deleteStu);
    void change(ElementType& changeStu, int position);
    void display();
private:
    Node<ElementType>* head;
    int size;
};
```

私有成员：

int size;//链表中的结点个数
Node<ElementType>* head;//指针域，指向链表的头结点

公有操作：

LinkList();
//构造函数，开辟一个附加头结点并将链表结点个数设为 0

~LinkList();
//析构函数，将链表中的结点删除，实现对内存的回收

```
int getSize();
//返回当前链表结点个数

void append(ElementType& inputStu, int position);
//将输入考生信息插入到相应的位置上

Node<ElementType>* findByID(string studentID);
//通过考号来寻找考生

Node<ElementType>* findByLocation(int position);
//通过位置来寻找考生

void pop(Node<ElementType>* deleteStu);
//删除目标考生信息所在的结点

void change(ElementType& changeStu, int position);
//改变某位置中的考生信息

void display();
//展示报名系统中的所有考生信息
```

学生类（Student）：

```
class Student {
public:
    Student() = default;
    Student(string tempID, string tempN, string tempS, string tempA,
    string tempC):
    ID(tempID),name(tempN),sex(tempS),age(tempA),applicationCategory
    (tempC){}
    string getID();
    Student& operator =(const Student& temp);
    friend ostream& operator<<(ostream& os, const Student& temp);
    friend istream& operator>>(istream& is, Student& temp);
private:
    string ID;
    string name;
    string sex;
    string age;
    string applicationCategory;
};
```

私有成员：

```
string ID;//考生考号
string name;//考生姓名
string sex;//考生性别
string age;//考生年龄
string applicationCategory;//考生报考类别
```

公有操作：

```
Student() = default;
//默认构造函数

Student(string tempID, string tempN, string tempS, string tempA,
string tempC);
//含参构造函数

string getID();
//获取该考生的考号

Student& operator =(const Student& temp);
//重载=运算符，使该类支持赋值操作

friend ostream& operator<<(ostream& os, const Student& temp);
//重载<<运算符并声明为友元，使该类可以进行<<运算

friend istream& operator>>(istream& is, Student& temp);
//重载>>运算符并声明为友元，使该类可以进行>>运算
```

考试信息系统类（InformationSystem）：

```
template <typename ElementType>
class InformationSystem {
public:
    void init();
    void append();
    void popByID();
    void popByLocation();
    void findByID();
    void findByLocation();
    void change();
    void display();
private:
    ElementType headline;
    LinkList<ElementType> studentInformation;
};
```

私有成员：

```
ElementType headline;//考生信息系统中的表头
LinkList<ElementType> studentInformation;//存储信息的链表
```

公有操作：

```
void init();
//初始化考试报名系统

void append();
//向系统中添加一名考生

void popByID();
//通过考号查询删除对应的考生

void popByLocation();
//通过输入考生所在系统的位置删除对应的考生

void findByID();
//通过考号查询对应的考生

void findByLocation();
//通过输入考生所在系统的位置查询对应的考生

void change();
//通过输入系统位置来修改该位置的考生信息

void display();
//统计并展示目前系统中的考生信息
```

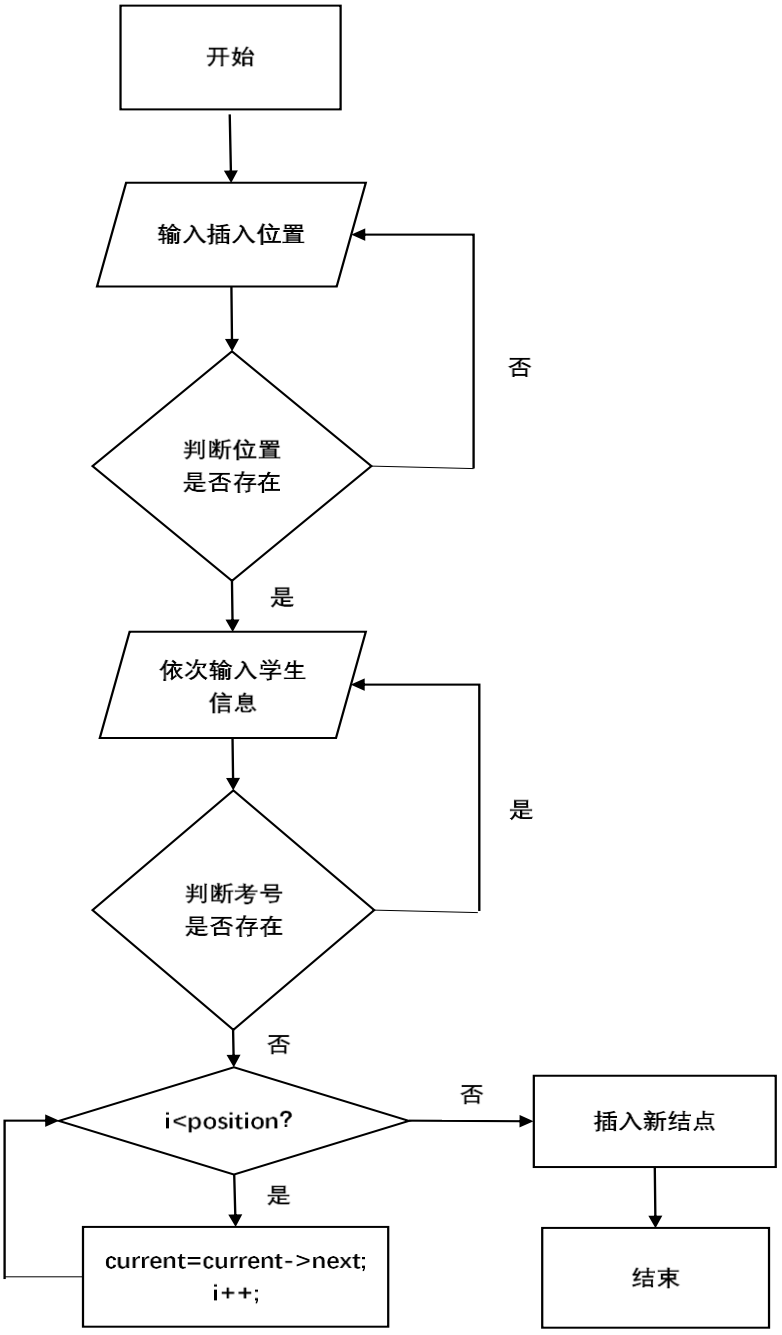
2.4 系统设计

系统在首次使用时，会先让用户输入初始考生录入数量，通过 init（）函数来完成系统的初始化，然后再根据用户后续所输入的操作码完成相应的操作。

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

InformationSystem 类中：

```
template <typename ElementType>
void InformationSystem<ElementType>::append()
{
    cout << "请输入你要插入的考生位置：";
    int position = 0;
    cin >> position;
    while (position <= 0 || position > studentInformation.getSize() + 1
 || cin.fail())
    {
        cout << "插入位置不正确！" << endl;
        cout << "请重新输入插入位置：";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cin >> position;
    }
    cout << "请依次输入要插入考生的考号，姓名，性别，年龄及报考类别！
" << endl;
    ElementType inputStu;
    cin >> inputStu;
    while (studentInformation.findByID(inputStu.getID()) != nullptr)
    {
        cout << "该考号已被其他考生使用，请检查是否录入错误并重新输入！
" << endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cin >> inputStu;
    }
    studentInformation.append(inputStu, position);
    cout << "插入完成！" << endl;
}
```

Linklist 类中：

```
template <typename ElementType>
void LinkList<ElementType>::append(ElementType& inputStu, int position)
{
    Node<ElementType>* insert = new Node<ElementType>(inputStu);
    Node<ElementType>* current = head;
    for (int i = 1; i < position; i++)
    {
        current = current->next;
    }
```

```
    if (position != size + 1)
    {
        current->next->prev = insert;
        insert->next = current->next;
    }
    insert->prev = current;
    current->next = insert;
    size++;
}
```

3.1.3 插入功能截屏示例

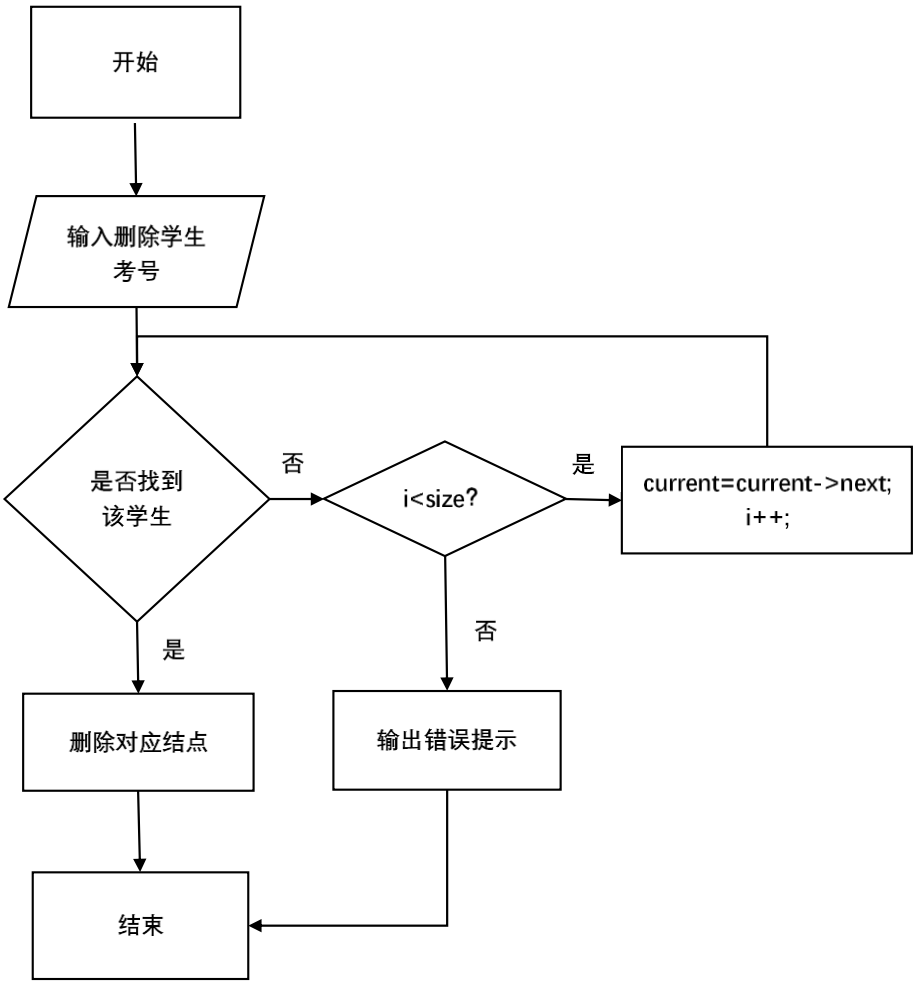
```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 女 20 软件设计师
2 stu2 男 21 软件开发师
3 stu3 男 20 软件设计师
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）:
1
请输入你要插入的考生的位置：4
请依次输入要插入考生的考号，姓名，性别，年龄及报考类别!
4 stu4 女 21 软件测试师
插入完成!

目前考生数量：4
考号    姓名    性别    年龄    报考类别
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
4       stu4    女      21      软件测试师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）:
1
请输入你要插入的考生的位置：6
插入位置不正确!
请重新输入插入位置：5
请依次输入要插入考生的考号，姓名，性别，年龄及报考类别!
4 stu5 男 22 软件开发师
该考号已被其他考生使用，请检查是否录入错误并重新输入!
```

3.2 按考号删除功能的实现

3.2.1 按考号删除功能流程图



3.2.2 按考号删除功能核心代码

InformationSystem 类中：

```
template <typename ElementType>
void InformationSystem<ElementType>::popByID()
{
    cout << "请输入要删除考生的考号：" << endl;
    string inputID = "";
    cin >> inputID;
    Node<ElementType>* deleteStu = studentInformation.findByID(inputID)
;
    if (deleteStu == nullptr)
    {
        cout << "该考号对应考生不存在，请检查是否输入错误！" << endl;
        return;
    }
    else
    {
        cout << "你删除的考生的信息为：" << endl;
        cout << deleteStu->data << endl;
        studentInformation.pop(deleteStu);
        cout << "删除成功！" << endl;
    }
}
```

Linklist 类中：

```
template <typename ElementType>
void LinkList<ElementType>::pop(Node<ElementType>* deleteStu)
{
    if (deleteStu->next == nullptr)
    {
        deleteStu->prev->next = nullptr;
    }
    else
    {
        deleteStu->prev->next = deleteStu->next;
        deleteStu->next->prev = deleteStu->prev;
    }
    delete deleteStu;
    size--;
}
```

3.2.3 按考号删除功能截屏示例

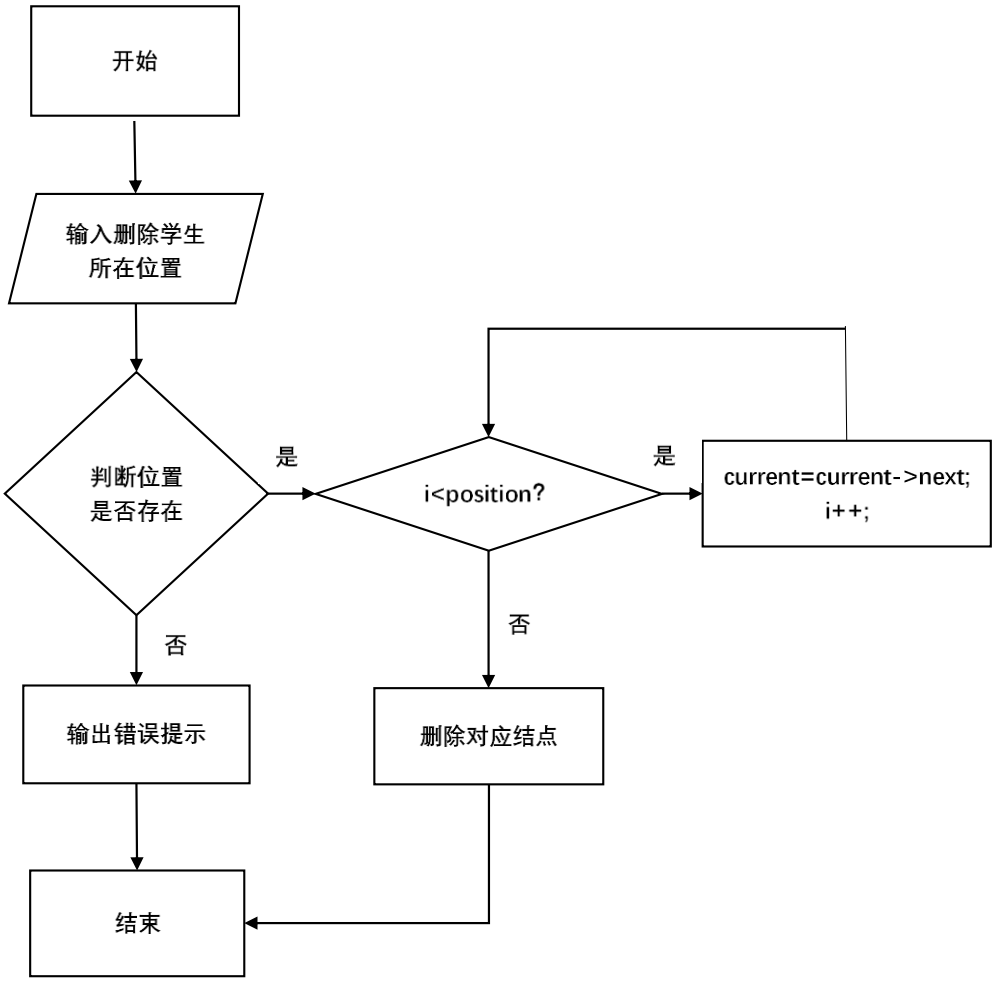
```
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 女 20 软件设计师
2 stu2 男 21 软件开发师
3 stu3 男 20 软件设计师
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）:
2
请输入要删除考生的考号:
2
你删除的考生的信息为:
2       stu2    男      21      软件开发师
删除成功!

目前考生数量：2
考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
3       stu3    男      20      软件设计师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）:
2
请输入要删除考生的考号:
4
该学号对应考生不存在，请检查是否输入错误
```

3.3 按位置删除功能的实现

3.3.1 按位置删除功能流程图



3.3.2 按位置删除功能核心代码

InformationSystem 类中：

```
template <typename ElementType>
void InformationSystem<ElementType>::popByLocation()
{
    cout << "请输入要删除考生的位置：" << endl;
    int inputLocation = 0;
    cin >> inputLocation;
    if (inputLocation > studentInformation.getSize() || inputLocation <
= 0 || cin.fail())
    {
        cout << "该位置对应考生不存在，请检查是否输入错误！" << endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return;
    }
    else
    {
        Node<ElementType>* deleteStu = studentInformation.findByLocatio
n(inputLocation);
        cout << "你删除的考生的信息为：" << endl;
        cout << deleteStu->data << endl;
        studentInformation.pop(deleteStu);
        cout << "删除成功！" << endl;
    }
}
```

Linklist 类中：

```
template <typename ElementType>
void LinkList<ElementType>::pop(Node<ElementType>* deleteStu)
{
    if (deleteStu->next == nullptr)
    {
        deleteStu->prev->next = nullptr;
    }
    else
    {
        deleteStu->prev->next = deleteStu->next;
        deleteStu->next->prev = deleteStu->prev;
    }
    delete deleteStu;
    size--;
}
```

3.3.3 按位置删除功能截屏示例

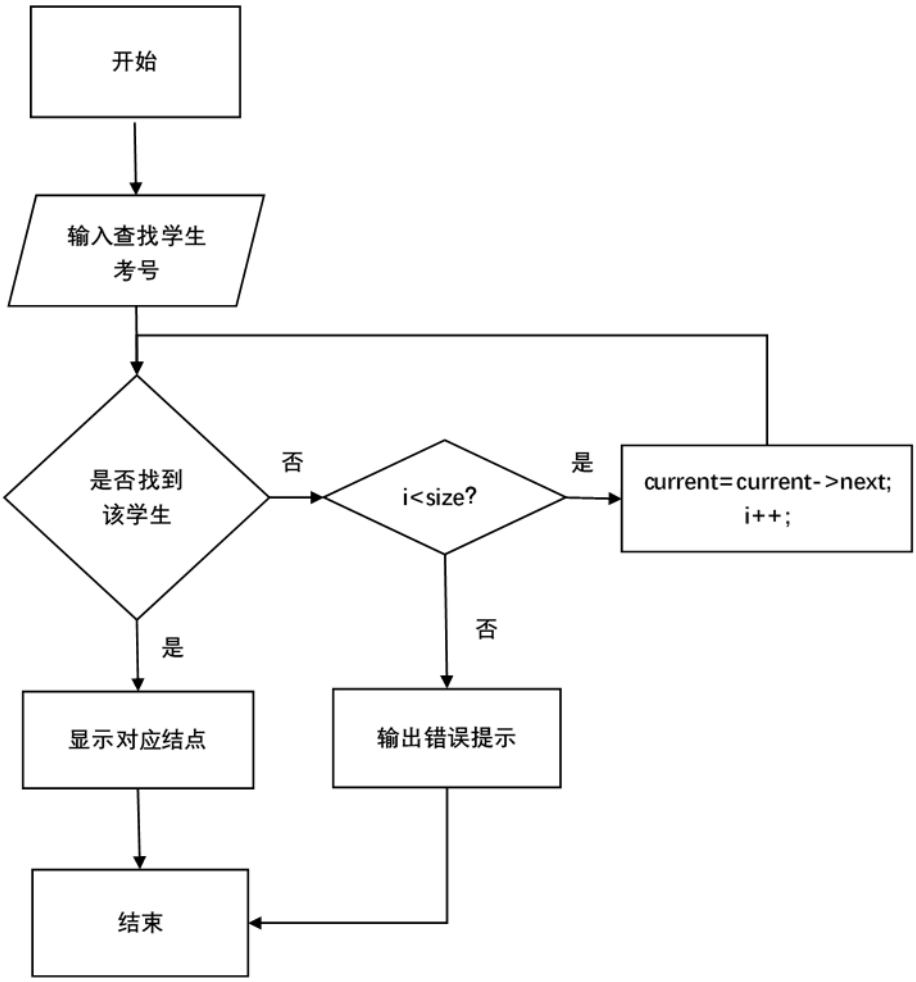
```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 女 20 软件设计师
2 stu2 男 21 软件开发师
3 stu3 男 20 软件设计师
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
3
请输入要删除考生的位置：
1
你删除的考生的信息为：
1       stu1    女      20      软件设计师
删除成功!

目前考生数量：2
考号    姓名    性别    年龄    报考类别
2       stu2    男      21      软件开发师
3       stu3    男      20      软件设计师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
3
请输入要删除考生的位置：
3
该位置对应考生不存在，请检查是否输入错误
```

3.4 按考号查找功能的实现

3.4.1 按考号查找功能流程图



3.4.2 按考号查找功能核心代码

InformationSystem 类中：

```
template <typename ElementType>
void InformationSystem<ElementType>::findByID()
{
    cout << "请输入要查找的考生的考号：" << endl;
    string findID = "";
    cin >> findID;
    Node<ElementType>* findStu = studentInformation.findByID(findID);
    if (findStu == nullptr)
    {
        cout << "该考号对应考生不存在，请检查是否输入错误！" << endl;
        return;
    }
    else
    {
        cout << headline << endl;
        cout << findStu->data << endl;
        cout << "查找成功！" << endl;
    }
}
```

Linklist 类中：

```
template <typename ElementType>
Node<ElementType>* LinkList<ElementType>::findByID(string studentID)
{
    Node<ElementType>* current = head;
    for (int i = 0; i < size; i++)
    {
        current = current->next;
        if (studentID == current->data.getID())
        {
            return current;
        }
    }
    return nullptr;
}
```

3.4.3 按考号查找功能截屏示例

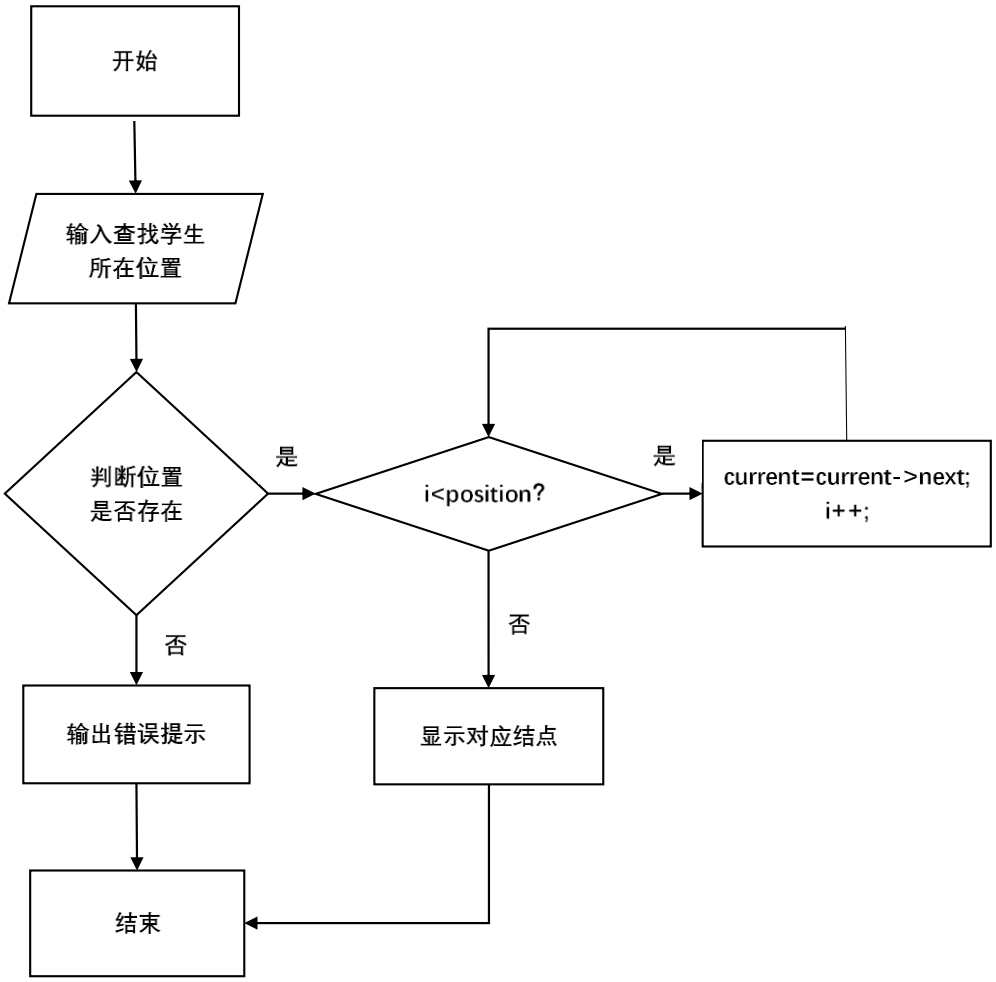
```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 女 20 软件设计师
2 stu2 男 21 软件开发师
3 stu3 男 20 软件设计师
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
4
请输入要查找的考生的考号：
2
考号    姓名    性别    年龄    报考类别
2       stu2    男      21      软件开发师
查找成功!

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
4
请输入要查找的考生的考号：
5
该学号对应考生不存在，请检查是否输入错误
```

3.5 按位置查找功能的实现

3.5.1 按位置查找功能流程图



3.5.2 按位置查找功能核心代码

InformationSystem 类中：

```
template <typename ElementType>
void InformationSystem<ElementType>::findByLocation()
{
    cout << "请输入要查找的考生的位置：" << endl;
    int findLocation = 0;
    cin >> findLocation;
    if (findLocation > studentInformation.getSize() || findLocation <=
0 || cin.fail())
    {
        cout << "该位置对应考生不存在，请检查是否输入错误！" << endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return;
    }
    else
    {
        Node<ElementType>* findStu = studentInformation.findByLocation(
findLocation);
        cout << headline << endl;
        cout << findStu->data << endl;
        cout << "查找成功！" << endl;
    }
}
```

Linklist 类中：

```
template <typename ElementType>
Node<ElementType>* LinkList<ElementType>::findByLocation(int position)
{
    Node<ElementType>* current = head;
    for (int i = 0; i < position; i++)
    {
        current = current->next;
    }
    return current;
}
```

3.5.3 按位置查找功能截屏示例

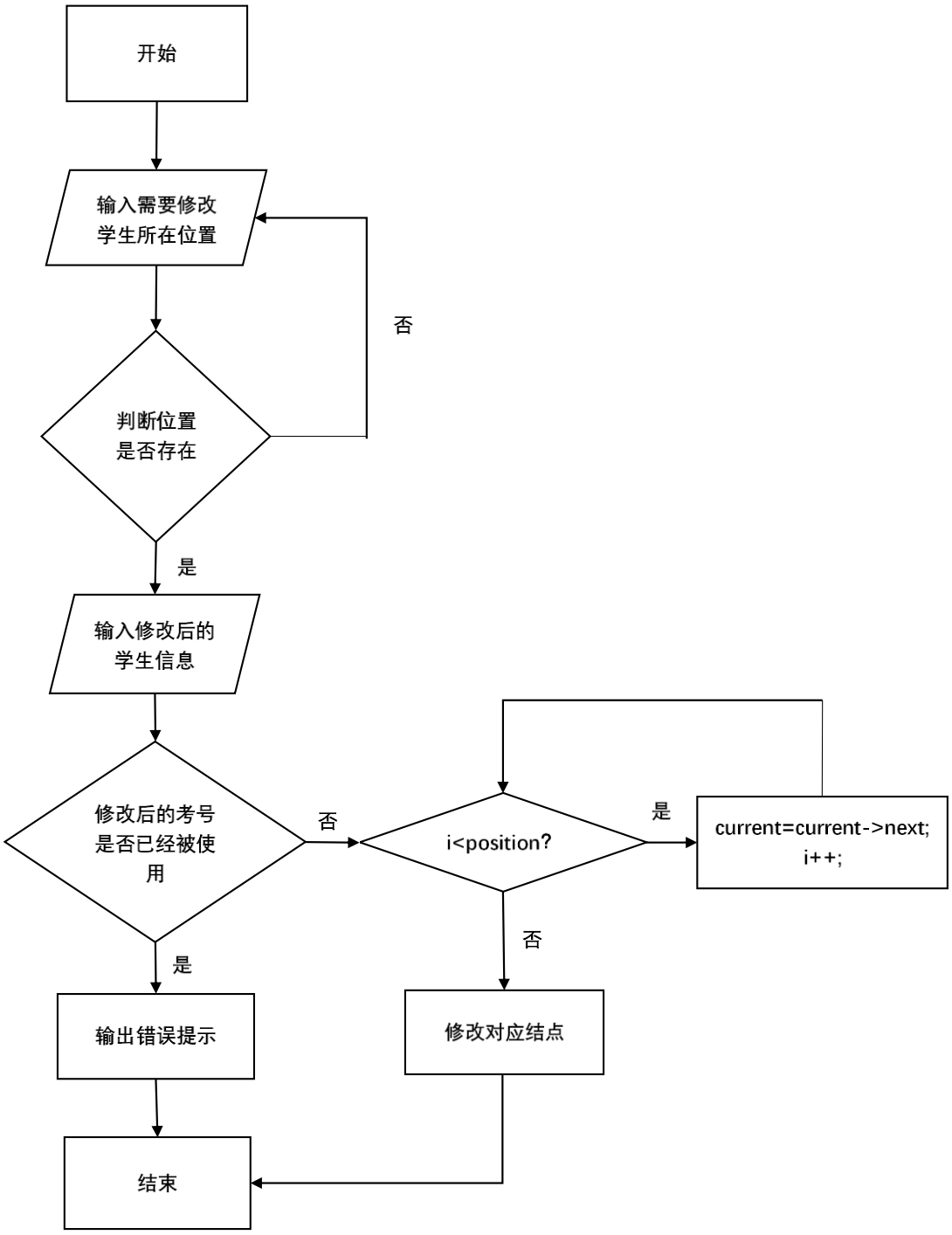
```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 女 20 软件设计师
2 stu2 男 21 软件开发师
3 stu3 男 20 软件设计师
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）:
5
请输入要查找考生的位置:
3
考号    姓名    性别    年龄    报考类别
3       stu3    男      20      软件设计师
查找成功!

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）:
5
请输入要查找考生的位置:
4
该位置对应考生不存在，请检查是否输入错误
```

3.6 修改功能的实现

3.6.1 修改功能流程图



3.6.2 修改功能核心代码

InformationSystem 类中：

```
template <typename ElementType>
void InformationSystem<ElementType>::change()
{
    cout << "请输入你要修改的考生的位置：";
    int position = 0;
    cin >> position;
    while (position <= 0 || position > studentInformation.getSize() ||
cin.fail())
    {
        cout << "位置输入不正确！" << endl;
        cout << "请重新输入位置：";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cin >> position;
    }
    cout << "请依次输入修改后的考号，姓名，性别，年龄及报考类别：" << endl;
    ElementType changeStu;
    cin >> changeStu;
    if (studentInformation.findByID(changeStu.getID()) != nullptr)
    {
        cout << "该考号已被其他考生使用，请检查是否录入错误！" << endl;
        return;
    }
    else
    {
        studentInformation.change(changeStu, position);
        cout << "修改完成" << endl;
    }
}
```

Linklist 类中：

```
template <typename ElementType>
void LinkList<ElementType>::change(ElementType& changeStu, int position
)
{
    Node<ElementType>* current = head;
    for (int i = 0; i < position; i++)
    {
        current = current->next;
    }
    current->data = changeStu;
}
```

3.6.3 修改功能截屏示例

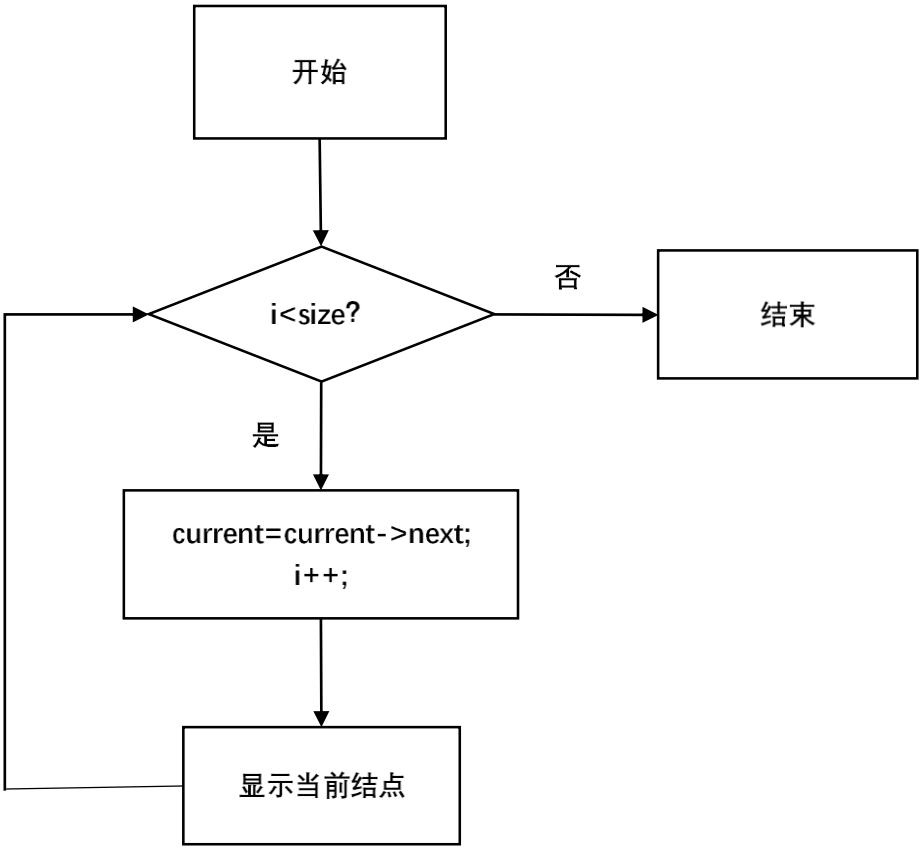
```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 女 20 软件设计师
2 stu2 男 21 软件开发师
3 stu3 男 20 软件设计师
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
6
请输入你要修改的考生的位置：2
请依次输入修改后的考号，姓名，性别，年龄及报考类别：
4 stu4 女 22 软件设计师
修改完成!

目前考生数量：3
考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
4       stu4    女      22      软件设计师
3       stu3    男      20      软件设计师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
6
请输入你要修改的考生的位置：4
位置输入不正确!
请重新输入位置：3
请依次输入修改后的考号，姓名，性别，年龄及报考类别：
1 stu5 男 22 软件开发师
该考号已被其他考生使用，请检查是否录入错误!
```

3.7 统计功能的实现

3.7.1 统计功能流程图



3.7.2 统计功能核心代码

InformationSystem 类中：

```
template <typename ElementType>
void InformationSystem<ElementType>::display()
{
    cout << "目前考生数量：" << studentInformation.getSize() << endl;
    cout << headline << endl;
    studentInformation.display();
}
```

Linklist 类中：

```
template <typename ElementType>
void LinkList<ElementType>::display()
{
    Node<ElementType>* current = head;
    for (int i = 0; i < size; i++)
    {
        current = current->next;
        cout << current->data << endl;
    }
}
```

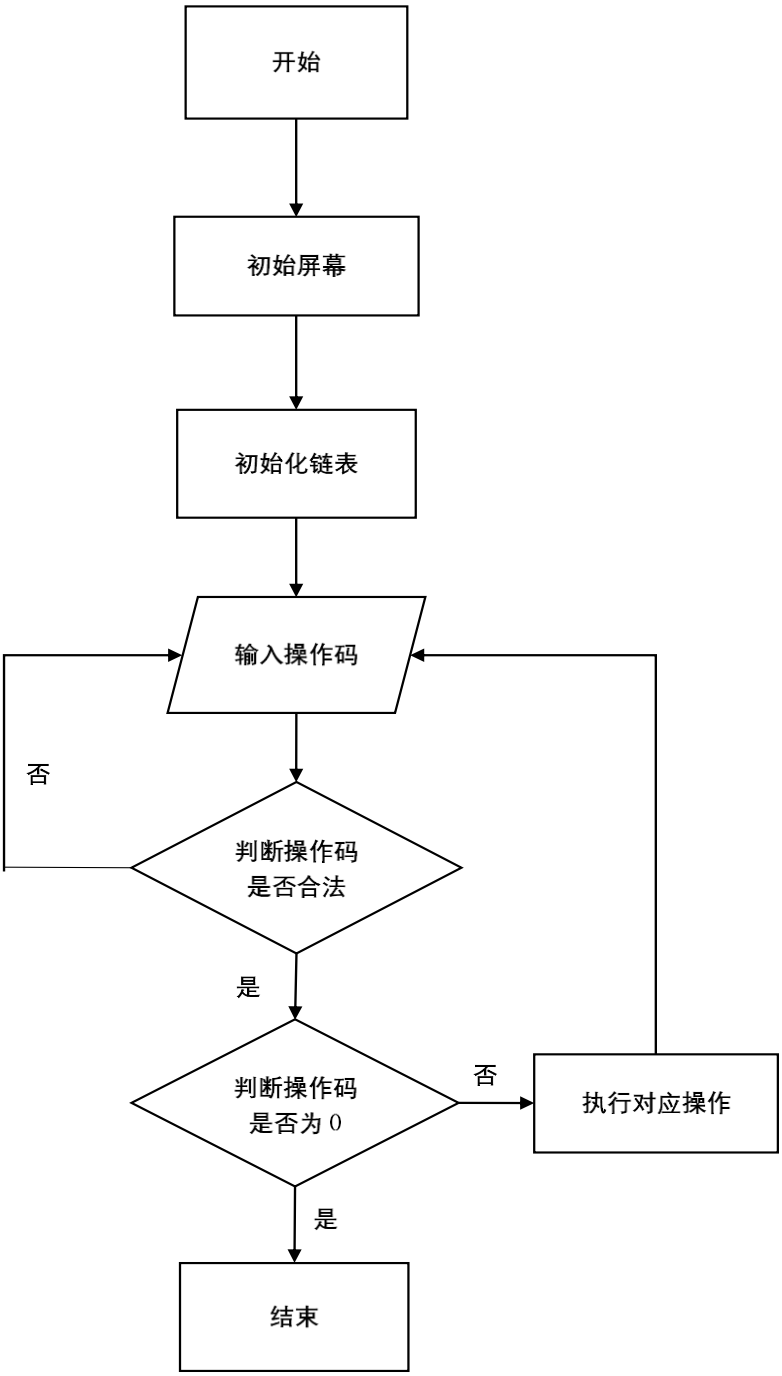
3.7.3 统计功能截屏示例

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
7

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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
```

3.8 总体功能的实现

3.8.1 总体功能流程图



3.8.2 总体功能核心代码

```
int main()
{
    InformationSystem<Student> testSystem;
    testSystem.init();
    while (true)
    {
        cout << "请选择您要进行的操作（1 为插入，2 为按考号删除，3 为按位置删除，"
             << "4 为按考号查找，5 为按位置查找，6 为修改，7 为统计，0 为取消操作）:"
             << endl;
        int operater;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cin >> operater;
        switch (operater)
        {
        case 1:
            testSystem.append();
            break;
        case 2:
            testSystem.popByID();
            break;
        case 3:
            testSystem.popByLocation();
            break;
        case 4:
            testSystem.findByID();
            break;
        case 5:
            testSystem.findByLocation();
            break;
        case 6:
            testSystem.change();
            break;
        case 7:
            break;
        case 0:
            cout << "成功退出系统！" << endl;
            return 0;
        default:
            cout << "操作数输入不正确，请重新输入！" << endl;
```

```
            break;
        }
        cout << '\n';
        testSystem.display();
        cout << '\n';
    }
    return 0;
}
```

3.8.3 总体功能截屏示例


```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）:
8
操作数输入不正确，请重新输入!

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）:
0
成功退出系统!
```

4 测试

4.1 功能测试

4.1.1 插入功能测试

测试用例：stu4 男 22 软件测试师
预期结果：
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
4 stu4 男 22 软件测试师
实验结果：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
1
请输入你要插入的考生的位置：4
请依次输入要插入考生的考号，姓名，性别，年龄及报考类别!
4 stu4 男 22 软件测试师
插入完成!

目前考生数量：4
考号    姓名    性别    年龄    报考类别
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
4       stu4    男      22      软件测试师
```

4.1.2 按考号删除功能测试

测试用例：删除考号为 4 的考生
预期结果：
1 stu1 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：

```
目前考生数量：4
考号    姓名    性别    年龄    报考类别
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
4       stu4    男      22      软件测试师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
2
请输入要删除考生的考号：
4
你删除的考生的信息为：
4       stu4    男      22      软件测试师
删除成功!

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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
```

4.1.3 按位置删除功能测试

测试用例：删除位置在 3 的考生
预期结果：
1 stu1 女 20 软件设计师
2 stu2 男 21 软件开发师
实验结果：

```
目前考生数量：3
考号    姓名    性别    年龄    报考类别
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

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
3
请输入要删除考生的位置：
3
你删除的考生的信息为：
3       stu3    男      20      软件设计师
删除成功!

目前考生数量：2
考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
2       stu2    男      21      软件开发师
```

4.1.4 按考号查找功能测试

测试用例：查找考号为 2 的考生
预期结果：
2 stu2 男 21 软件开发师

实验结果：

```
目前考生数量：2
考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
2       stu2    男      21      软件开发师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
4
请输入要查找的考生的考号：
2
考号    姓名    性别    年龄    报考类别
2       stu2    男      21      软件开发师
查找成功!
```

4.1.5 按位置查找功能测试

测试用例：查找位置为 1 的考生
预期结果：
1 stu1 女 20 软件设计师
实验结果：

```
目前考生数量：2
考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
2       stu2    男      21      软件开发师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
5
请输入要查找考生的位置：
1
考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
查找成功!
```

4.1.6 修改功能测试

测试用例：将位置 1 修改为考号 5，姓名 stu5，性别男，年龄 22，报考种类网络工程师。
预期结果：
5 stu5 男 22 网络工程师

实验结果：

```
目前考生数量：2
考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
2       stu2    男      21      软件开发师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
6
请输入你要修改的考生的位置：1
请依次输入修改后的考号，姓名，性别，年龄及报考类别：
5 stu5 男 22 网络工程师
修改完成!

目前考生数量：2
考号    姓名    性别    年龄    报考类别
5       stu5    男      22      网络工程师
2       stu2    男      21      软件开发师
```

4.1.7 统计功能测试

测试用例：统计当前数据
预期结果：
5 stu5 男 22 网络工程师
2 stu2 男 21 软件开发师
实验结果：

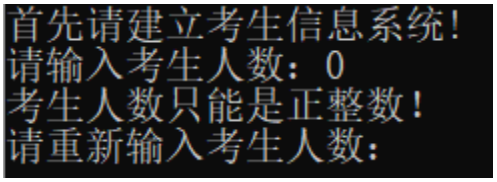
```
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
7

目前考生数量：2
考号    姓名    性别    年龄    报考类别
5       stu5    男      22      网络工程师
2       stu2    男      21      软件开发师
```

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：给出错误提示，程序运行正常不崩溃。
实验结果：



4.2.2 删除根结点

测试用例：删除根结点
预期结果：程序正常运行，不崩溃。
实验结果：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
2
请输入要删除考生的考号：
1
你删除的考生的信息为：
1       stu1    女      20      软件设计师
删除成功!

目前考生数量：2
考号    姓名    性别    年龄    报考类别
2       stu2    男      21      软件开发师
3       stu3    男      20      软件设计师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
3
请输入要删除考生的位置：
1
你删除的考生的信息为：
2       stu2    男      21      软件开发师
删除成功!

目前考生数量：1
考号    姓名    性别    年龄    报考类别
3       stu3    男      20      软件设计师
```

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：程序正常运行，不崩溃。
实验结果：

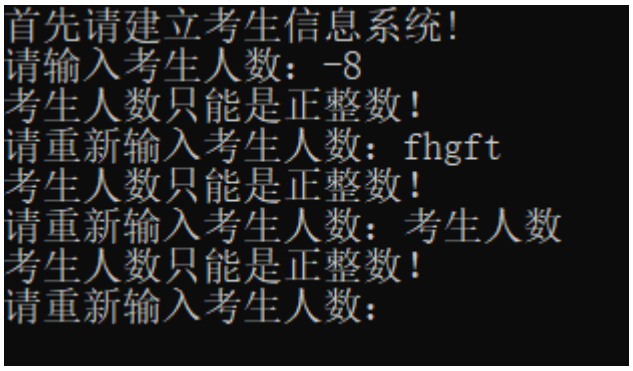
```
首先请建立考生信息系统!
请输入考生人数：1
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 女 20 软件设计师
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
3
请输入要删除考生的位置：
1
你删除的考生的信息为：
1       stu1    女      20      软件设计师
删除成功!

目前考生数量：0
考号    姓名    性别    年龄    报考类别
```

4.3 出错测试

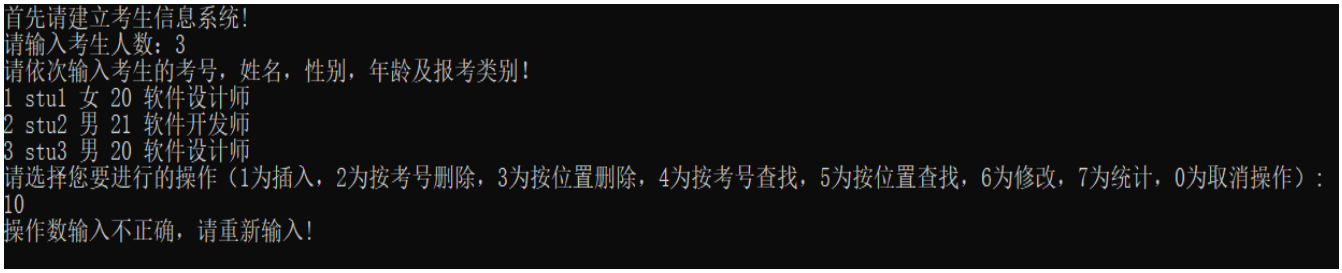
4.3.1 考生人数错误

测试用例：输入考生人数为负数或者不是数字
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 插入时位置输入错误

测试用例：插入时输入位置不存在或不为数字
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：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
1
请输入你要插入的考生的位置：6
插入位置不正确!
请重新输入插入位置：fhsdfsd
插入位置不正确!
请重新输入插入位置：插入考生
插入位置不正确!
请重新输入插入位置：
```

4.3.4 插入考生的考号已经存在

测试用例：系统中有三条信息，向其中插入考号已经存在于系统中的考生
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：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
1
请输入你要插入的考生的位置：4
请依次输入要插入考生的考号，姓名，性别，年龄及报考类别!
2 stu4 女 22 网络工程师
该考号已被其他考生使用，请检查是否录入错误并重新输入!
```

4.3.5 删除考号不存在

测试用例：在按考号删除时，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：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
2
请输入要删除考生的考号：
4
该学号对应考生不存在，请检查是否输入错误
```

4.3.6 删除位置不存在

测试用例：在按位置删除时，要删除的位置不存在或不是数字
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：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
3
请输入要删除考生的位置：
4
该位置对应考生不存在，请检查是否输入错误

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
3
请输入要删除考生的位置：
dsadsafsdf
该位置对应考生不存在，请检查是否输入错误

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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
```

4.3.7 查找考号不存在

测试用例：在按考号查找时，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：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）
4
请输入要查找的考生的考号：
5
该学号对应考生不存在，请检查是否输入错误
```

4.3.8 查找位置不存在

测试用例：在按位置查找时，要查找的位置不存在或不是数字
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：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
5
请输入要查找考生的位置：
5
该位置对应考生不存在，请检查是否输入错误

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
5
请输入要查找考生的位置：
fareegre
该位置对应考生不存在，请检查是否输入错误

目前考生数量：3
考号    姓名    性别    年龄    报考类别
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
```

4.3.9 修改位置不存在

测试用例：要修改的位置不存在或不是数字
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：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
6
请输入你要修改的考生的位置：4
位置输入不正确!
请重新输入位置：fdsafsdg
位置输入不正确!
请重新输入位置：修改位置
位置输入不正确!
请重新输入位置：
```

4.3.10 修改考生的考号已存在

测试用例：系统中有三条信息，修改考生的考号为系统中存在的考号
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：

```
首先请建立考生信息系统!
请输入考生人数：3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 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
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
6
请输入你要修改的考生的位置：1
请依次输入修改后的考号，姓名，性别，年龄及报考类别：
3 stu4 女 22 网络工程师
该考号已被其他考生使用，请检查是否录入错误!
```

4.3.11 插入或初始时输入考生信息溢出

测试用例：
1 stu1 女 20 软件设计师 网络工程师
2 stu2 男 21 软件设计师 网络工程师
预期结果：忽略多余的消息，程序正常运行不崩溃。
实验结果：

```
首先请建立考生信息系统!
请输入考生人数：1
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 软件设计师 网络工程师
请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
1
请输入你要插入的考生位置：2
请依次输入要插入考生的考号，姓名，性别，年龄及报考类别!
2 stu2 男 21 软件设计师 网络工程师
插入完成!

目前考生数量：2
考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
2       stu2    男      21      软件设计师

请选择您要进行的操作（1为插入，2为按考号删除，3为按位置删除，4为按考号查找，5为按位置查找，6为修改，7为统计，0为取消操作）：
```