项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 曹晓慈

学 号： 2052844

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](#_Toc495668153)

[1 功能分析 1](#_Toc495668154)

[2 设计 1](#_Toc495668156)

[2.1 数据结构设计 1](#_Toc495668157)

[2.2 类结构设计 2](#_Toc495668158)

[2.3 成员与操作设计 2](#_Toc495668159)

[3 实现 3](#_Toc495668161)

[3.1 插入功能的实现 3](#_Toc495668162)

[3.1.1 插入功能流程图 3](#_Toc495668163)

[3.1.2 插入功能核心代码 3](#_Toc495668164)

[3.1.3 插入功能截屏示例 3](#_Toc495668165)

[3.2 删除功能的实现 4](#_Toc495668166)

[3.2.1 删除功能流程图 4](#_Toc495668167)

[3.2.2 删除功能核心代码 4](#_Toc495668168)

[3.2.3 删除功能截屏示例 4](#_Toc495668169)

[3.3 查找功能的实现 5](#_Toc495668170)

[3.3.1 查找功能流程图 5](#_Toc495668171)

[3.3.2 查找功能核心代码 5](#_Toc495668172)

[3.3.3 查找功能截图示例 5](#_Toc495668173)

[3.4 修改功能的实现 6](#_Toc495668174)

[3.4.1 修改功能流程图 6](#_Toc495668175)

[3.4.2 修改功能核心代码 6](#_Toc495668176)

[3.4.3 修改功能截屏示例 6](#_Toc495668177)

[3.5 统计功能的实现 7](#_Toc495668178)

[3.5.1 统计功能流程图 7](#_Toc495668179)

[3.5.2 统计功能核心代码 7](#_Toc495668180)

[3.5.3 统计功能截屏示例 7](#_Toc495668181)

[3.6 总体系统的实现 8](#_Toc495668182)

[3.6.1 总体系统流程图 8](#_Toc495668183)

[3.6.2 总体系统核心代码 9](#_Toc495668184)

[3.6.3 总体系统截屏示例 9](#_Toc495668185)

[4 测试 9](#_Toc495668186)

[4.1 边界测试 9](#_Toc495668193)

[4.1.1 初始化无输入数据 9](#_Toc495668194)

[4.1.2 删除头结点 10](#_Toc495668195)

[4.1.3 删除后链表为空 10](#_Toc495668196)

[4.2 出错测试 11](#_Toc495668197)

[4.2.1 考生人数错误 11](#_Toc495668198)

[4.2.2 操作码错误 12](#_Toc495668199)

[4.2.3 插入位置不存在 12](#_Toc495668200)

[4.2.4 删除考号不存在 13](#_Toc495668201)

[4.2.5 查找考号不存在 13](#_Toc495668202)

[4.2.6 修改考号不存在 13](#_Toc495668203)

# 1 功能分析

作为一个最简易的考试报名系统，首先应该有的功能就是输入同学们的考试报名情况并且可以予以显示。其次，考试报名系统还应该具有插入、删除、修改功能，以保证同学可以随时更改自己的考试报名情况。最后，考试报名系统软件还应该确保软件可以正常关闭。

综上所述，一个考试报名系统至少应该具有输入、输出、插入、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。为方便处理，本系统采用struct描述链表结点类（LNode），这样使得链表结点类（List）可以访问链表结点。

## 2.3 成员与操作设计

结构体（student）

成员：

int number; //学号

string name; //姓名

string gender; //性别

int age; //年龄

string type; //工作职位

friend istream& operator>>(istream&, student& student)；//输入重载

friend ostream& operator<<(ostream&, student& student)；//输出重载

**链表结点类（LNode）**

**公有成员：**

**T data;**

**LinkNode<T>\* link;**

**链表类（LinkList）**

**私有成员：**

LinkNode<T>\* first;//链表头指针

**公有操作：**

List() { first = new LinkNode<T>; } //析构函数

List(const T & x) { first = new LinkNode<T>(x); } //构造函数

List(List<T> & L); //复制构造函数

~List() { makeEmpty(); } //析构函数

void makeEmpty(); //将链表置为空表

int Length()const; //计算链表的长度

LinkNode<T>\* getHead()const{return first;} //返回附加头结点的地址

LinkNode<T>\* Search(T x); //搜索含数据x的元素

LinkNode<T>\* Locate(int i); //搜索第i个元素的地址

bool getData(int i, T& x); //取出第i个元素的值

void setData(int i, T& x); //用x修改第i个元素的值

bool Insert(int i, T& x); //在第i个元素后插入x

bool Remove(int i, T& x); //删除第i个元素，x返回该元素的值 bool IsEmpty()const //判表空否？空则返回true

# 绘图1.gif3 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

LinkNode<T>\* current = Locate(i-1);

if (current == NULL)return false;

LinkNode<T>\* newNode = new LinkNode<T>(x);

if (newNode == NULL)

{

cerr << "存储分配错误!" << endl;

exit(1);

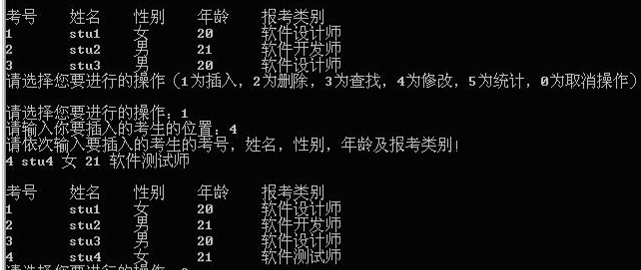
}

newNode->link = current->link;

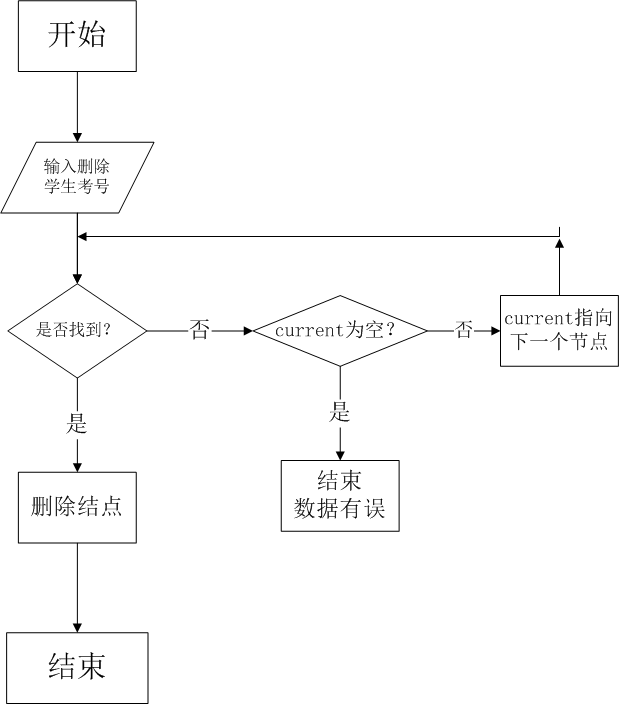
current->link = newNode;

return true;

3.1.3 插入功能截屏示例

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

bool List<T>:: Remove(int i, T& x)

{

LinkNode<T>\* current = Locate(i - 1);

if (current == NULL || current->link == NULL)return false;

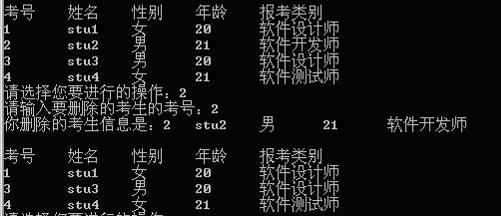
LinkNode<T>\* del = current->link;

current->link = del->link;

x = del->data; delete del;

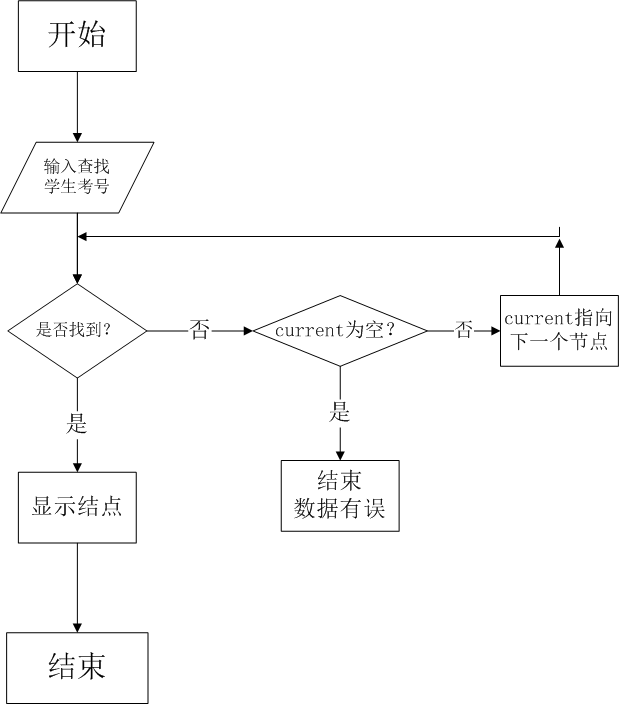
return true;

### }3.2.3 删除功能截屏示例



## 3.3 查找功能的实现

### 3.3.1 查找功能流程图



### 3.3.2 查找功能核心代码

bool List<T>:: getData(int i, T& x)

{

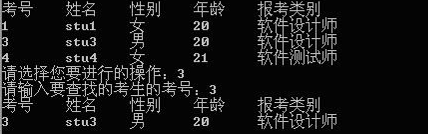
if (i <= 0)return NULL;

LinkNode<T>\* current = Locate(i);

if (current == NULL)return false;

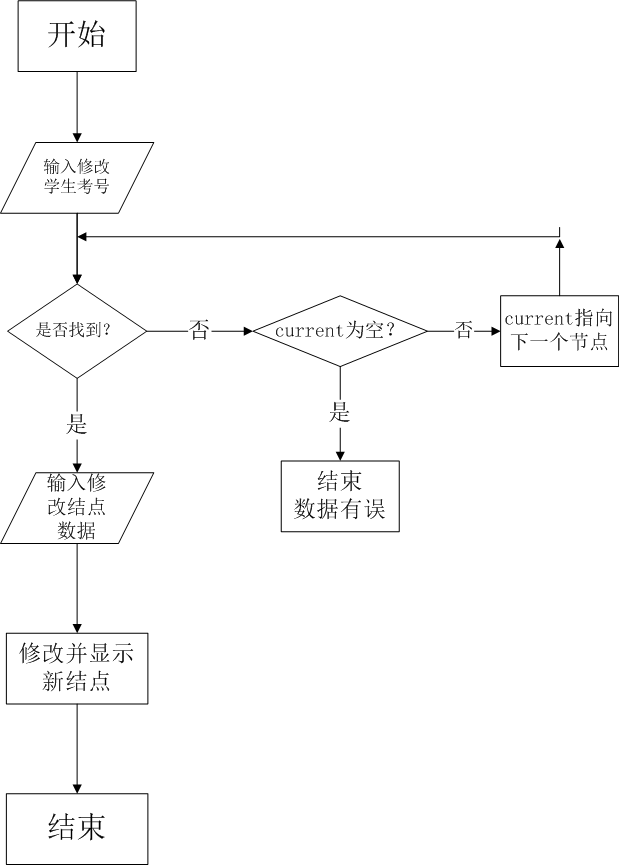
else { x = current->data; return true; }

### };3.3.3 查找功能截图示例



## 3.4 修改功能的实现

### 3.4.1 修改功能流程图



### 3.4.2 修改功能核心代码

template<class T>

void List<T>:: setData(int i, T& x) //给列表第i个元素赋值x

{

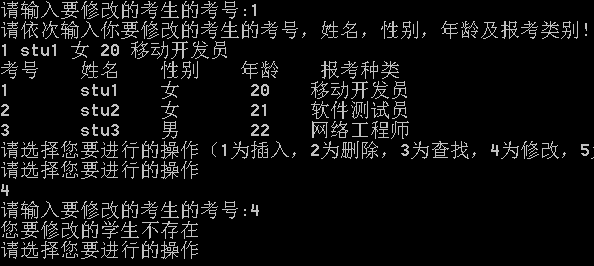
if (i <= 0)return;

LinkNode<T>\* current = Locate(i);

if (current == NULL)return;

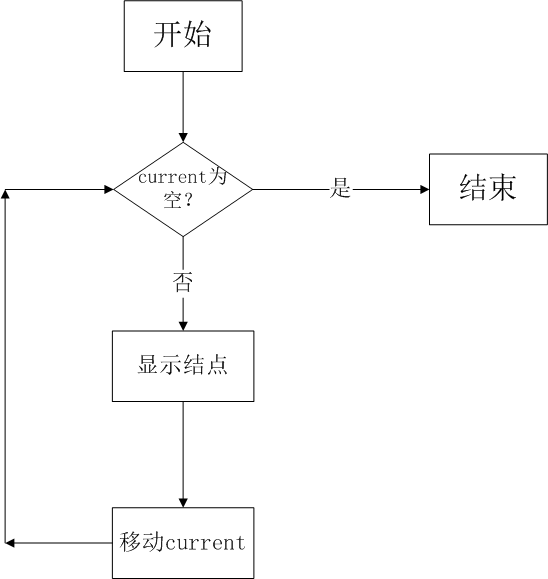
else current->data = x;

### };3.4.3 修改功能截屏示例



## 3.5 统计功能的实现

### 3.5.1 统计功能流程图



### 3.5.2 统计功能核心代码

void List<T>::output()

{

LinkNode<T>\* current = first->link;

while (current != NULL)

{

cout << current->data << endl;

current = current->link;

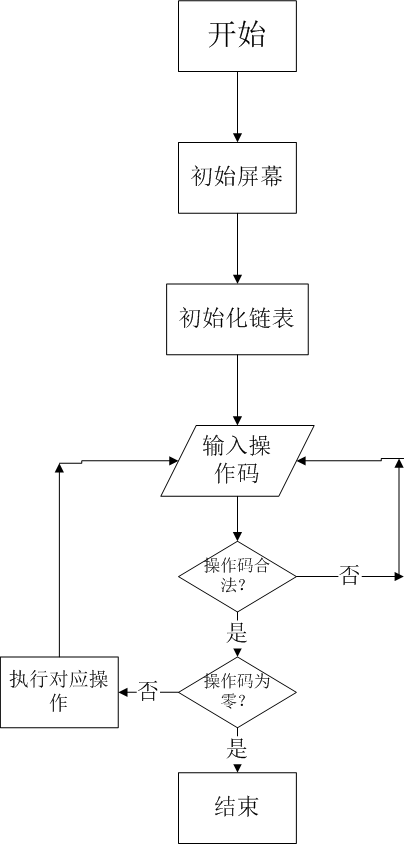
}

### }3.5.3 统计功能截屏示例



## 3.6 总体系统的实现

### 3.6.1 总体系统流程图



### 3.6.2 总体系统核心代码

while (choice != '0')

{

switch (chioce)

{

case 1: (插入功能); break;

case 2: （删除功能）; break;

case 3: （查找功能）; break;

case 4: （修改功能）; break;

case 5: （统计功能）; break;

//如果用户输入操作要求错误，提示用户重新输入

default: cout << "抱歉，您想要进行的操作不存在，请重新输入！" << endl; break;

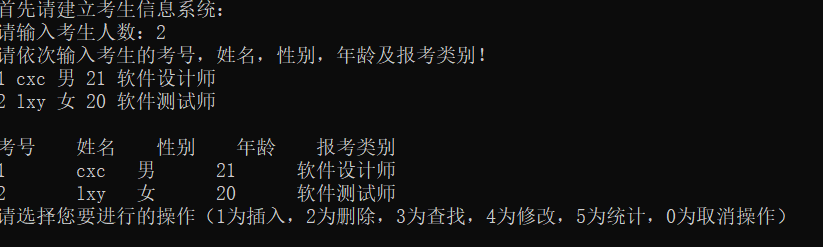
}

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;

cin >> choice;

}

### 3.6.3 总体系统截屏示例



# 4 测试

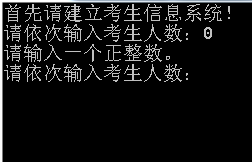
## 4.1 边界测试

### 4.1.1 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：**

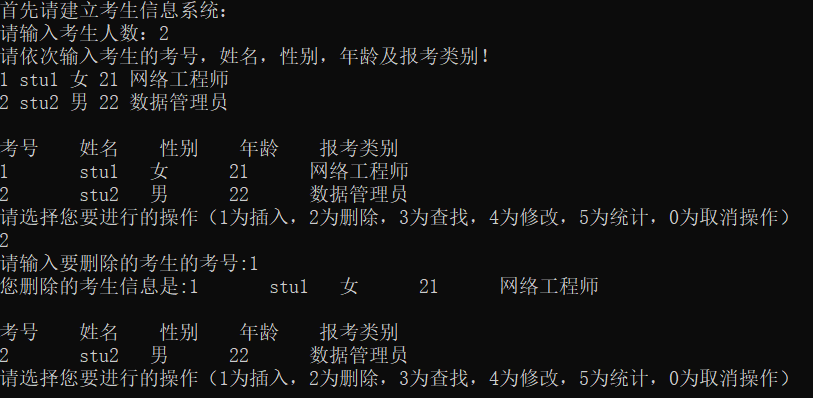


### 4.1.2 删除头结点

**测试用例：**删除头结点

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

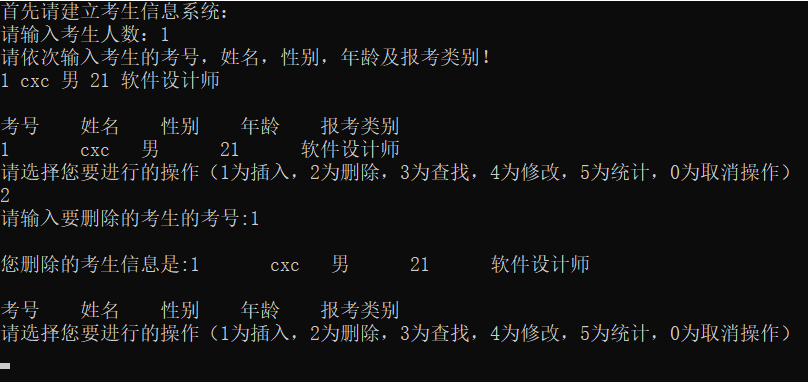


### 4.1.3 删除后链表为空

**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：**



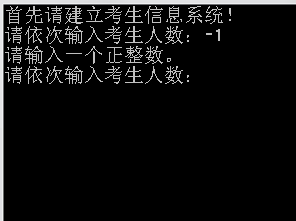
## 4.2 出错测试

### 4.2.1 考生人数错误

**测试用例：**输入考生人数为负数

**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：**

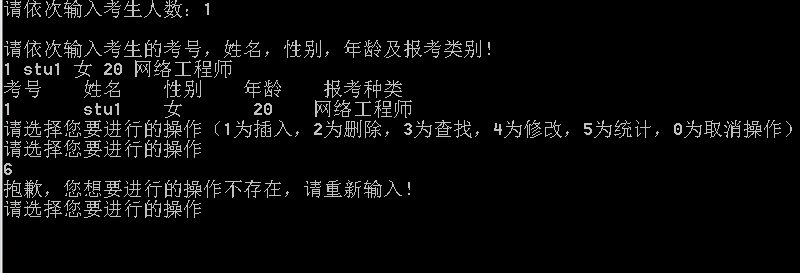
****

### 4.2.2 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：**

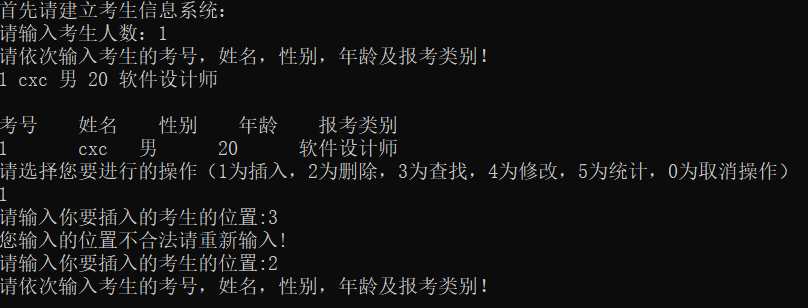
****

### 4.2.3 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：**

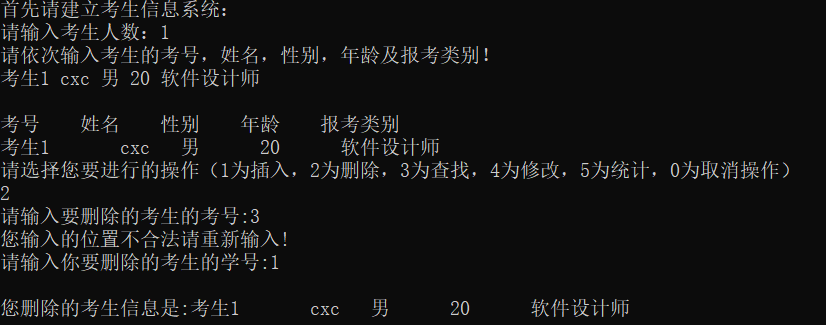


### 4.2.4 删除考号不存在

**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：**

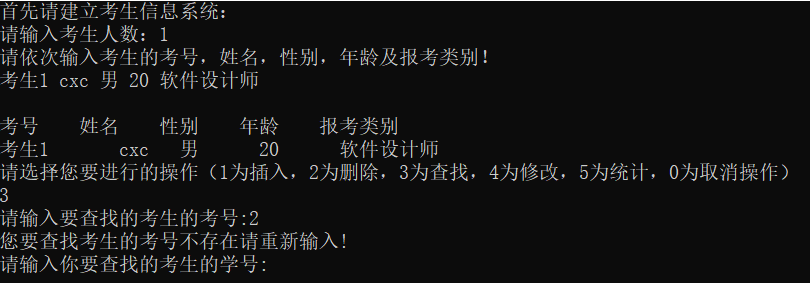
****

### 4.2.5 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：**

****

### 4.2.6 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：**

