数据结构课程设计

项目说明文档

考试报名系统

|  |  |
| --- | --- |
| 作者姓名： | 罗斌江 |
| 学 号： | 2053285 |
| 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项目分析 4](#_Toc32476)

[1.1项目背景 4](#_Toc12802)

[1.2 项目需求分析 5](#_Toc3007)

[1.3 项目要求 6](#_Toc21803)

[1.3.1 功能要求 6](#_Toc9321)

[1.3.2 输入格式 6](#_Toc26078)

[1.3.3 输出格式 6](#_Toc16101)

[1.3.4 项目示例 6](#_Toc6863)

[2 项目设计 7](#_Toc21639)

[2.1 数据结构设计 7](#_Toc30636)

[2.2 类设计 7](#_Toc13126)

[2.2.1 结点类（ListNode） 7](#_Toc3978)

[2.2.2 单向链表类（SinglyLinkedList） 7](#_Toc4853)

[2.2.3 数据库类（DataBase） 9](#_Toc12220)

[3 项目实现 11](#_Toc21824)

[3.1 项目主体功能流程图 11](#_Toc24152)

[3.2 项目初始化流程图 11](#_Toc20119)

[3.3 插入数据流程图 12](#_Toc12791)

[3.4 删除数据流程图 12](#_Toc10094)

[3.5 修改数据流程图 13](#_Toc29817)

[3.6 查询数据流程图 13](#_Toc8311)

[3.7 统计数据 13](#_Toc20685)

[4 项目测试 14](#_Toc6602)

[4.1 项目初始化测试 14](#_Toc3084)

[4.1.1 读取文件测试 14](#_Toc31755)

[4.1.2 手动输入测试 14](#_Toc3182)

[4.2 插入测试 15](#_Toc23572)

[4.3 删除测试 15](#_Toc17631)

[4.4 查找测试 15](#_Toc13494)

[4.5 修改测试 16](#_Toc4669)

[4.6 统计测试 16](#_Toc12152)

[4.7 边界测试 17](#_Toc16262)

[4.7.1 插入已存在考生 17](#_Toc31888)

[4.7.2 删除不存在考生 17](#_Toc20609)

[4.7.3 查找不存在考生 18](#_Toc31595)

[4.7.4 修改不存在考生 18](#_Toc28330)

# 1项目分析

## 1.1项目背景

对于每一所学校而言，都有着举行考试的需求。因此，考试报名系统也就成为了一个学校所不可或缺的一部分。同时，也正因为它对于学校的教务处的重要作用，一个良好的考试报名系统应该能为用户提供充分的功能。

而事实上，由于数量逐渐增加的考试人数，考试报名工作给当前的高校报名工作带来了新的挑战，也给教务管理部门增加了很大的工作量。例如在我们日常生活中，比较常接触到的考试就有四、六级考试、普通话考试、计算机水平考试等等，这些考试报名者都需要能够合理的管理，从而实现分配考场等功能。随着学生数量和考试数量的日益庞大，如何管理如此庞大的数据显得极为复杂，传统的手工管理工作量大且容易出错。

随着计算机科学技术的不断成熟，使用计算机对考试报名系统进行管理，具有手工管理所无法比拟的优势。这些优点能够极大地提高学校和学生的效率，也是学校走向信息化、科学化、国际化的重要条件。因此，开发一套考试报名系统具有十分重要的意义。

## 1.2 项目需求分析

一个完善的考试报名系统应当满足以下需求：

* 功能完善

作为一个最基本的数据库，增删改查是最基本的功能。除了这些基本功能外，还应该有人数统计，按类别和年龄筛选，信息有效性检查的功能。

* 可复用

为了避免下次重新导入信息，本系统应当支持存储信息的功能。

同时，也应该为使用者提供保存信息文件，以便下次重复使用的功能。

* 执行效率高

能够存储的数据条数应该足够多，至少能够存储一个学校（万条以上）学生信息。

即使在存入大量数据后进行查询也应该有较快的响应速度，其内部使用的数据结构和算法应该具有较低的时间和空间复杂度。

* 健壮性

对于错误的输入如输入了已存在的考号等，程序应该做到不崩溃并且给予一定的提示。

* 美观的界面

在第一次运行时，程序会提示用户初始化考生信息系统。

在打印考生信息时，应该注意页面篇幅，若信息条数极多，则应该只打印前50条。

* 数据可视化

在进行数据统计的时候，应当尽可能将数据信息通过可以直观感知的信息进行体现。

# 1.3 项目要求

### 1.3.1 功能要求

本项目的实质是完成对考生信息的建立，查找，插入，修改，删除等功能。其中考生信息包括准考证号，姓名，性别，年龄和报考类别等信息。项目在设计时应首先确定系统的数据结构，定义类的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。

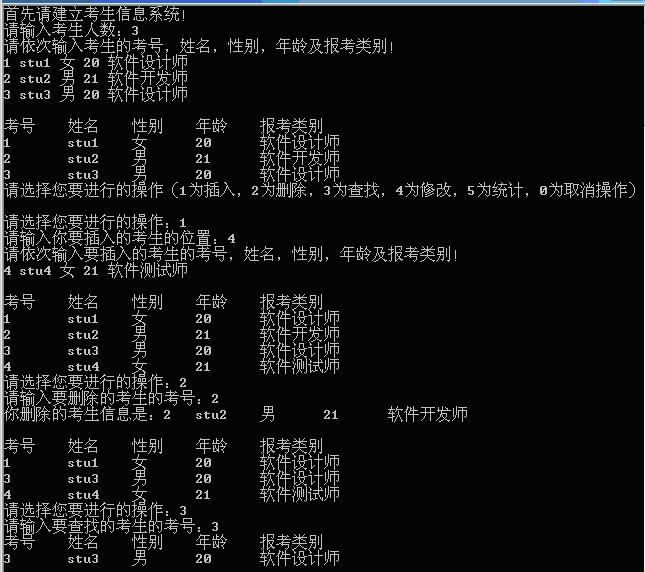
### 1.3.2 输入格式

输入需要执行的操作，以及考生的信息（包括准考证号、姓名、年龄和报考类别）

### 1.3.3 输出格式

输出查找学生的信息

### 1.3.4 项目示例



# 2 项目设计

## 2.1 数据结构设计

本系统是一个存储了学生类的存储系统，关键操作是大量对于数据的增加、删除、修改、查找等。因此，选择的数据结构要应该能够快速的对数据执行以上操作。本系统采用了链表作为底层数据。

## 2.2 类设计

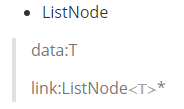
经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（ListNode）与链表类（List），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。

为了实现代码的复用性，本系统实现了一个**链表**。采用struct描述链表结点类（ListNode），这样使得链表结点类（List）可以直接访问链表结点而不需要定义友元关系。本系统实现的链表结构各种操作的时间复杂度如下：

* 插入操作：O(n)
* 删除操作：O(n)
* 查询操作：O(n)
* 遍历操作：O(n)

### 2.2.1 结点类（ListNode）

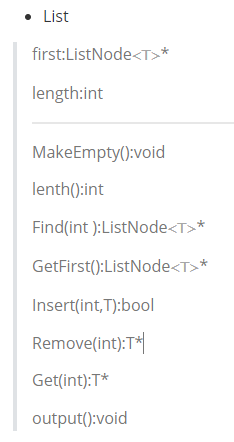
链表的结点中存储的数据有链表数据、后继结点和前驱结点。其UML图如下所示：



### 2.2.2 单向链表类（SinglyLinkedList）

链表的实现原理大同小异，不同之处在于：是否带头结点、是否带尾结点、每一个结点是否带前驱结点等。为了使得链表中各种操作的时间复杂度都尽可能低，因此这里选择了带头结点和尾结点的双向链表来实现链表中的各种操作。也即：选择了牺牲空间来达到降低时间复杂度的效果。

链表的UML图如下所示：



其中，List类中的主要函数如下所示：

* int **lenth**()const

返回链表中结点的个数，不包括头结点；

* bool Make**Empty**()

清除链表内的数据，不包括头节点；

* void **output**()

输出表中所有数据；

* bool Insert(**T** value, int i)

在i位置插入数据T；

* **ListNode<T>\* find**(int i)const

返回第i个数据；

* **T\*** **remove**(**int i**)

移除i所处位置的元素，返回移除位置元素的值；

* **T\*** **Get**(**int i**)；

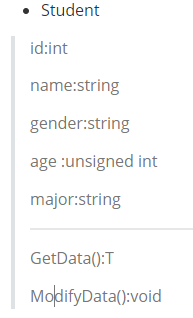
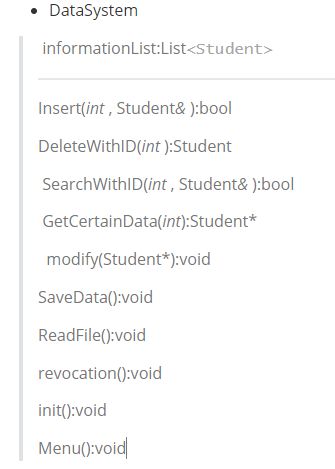
返回下标为i-1处数据的地址。

### 2.2.3 数据库类（DataBase）

针对于考试报名系统，其内部存储的数据应当是每一个考生。每一个考生都有着相应的姓名、性别、年龄、报考类别，因此首先应当定义一个学生类（Student）来存储学生信息。

此外，为了便于管理学生的信息，定义一个数据库类（DataBase）。内部使用Map存储学生的数据信息，同时提供各种操作方法，如初始化、添加考生、删除考生、修改考生信息、统计考生信息等。

基于上述原理，这两个类的UML图及关系如下图所示：

数据库类的关键操作函数及其介绍如下所示：

* void **init**()

初始化数据库，本系统中可以通过手动输入和读取文件两种方法进行初始化内部的学生信息。

* void **output**()

打印学生信息。

* bool **insert**(int i,Student& x)

向数据库中插入一条新的学生数据。

* Student **DeleteWithID**(int ID)

删除数据库中指定的学生数据。

* bool SearchWithID (int ID,Student& ptr)

在数据库中根据学号查询指定的学生信息。

* int **Size**()

返回数据库中的数据个数。

* void **modify**()

修改考生信息。

* void **ReadFile**()

从文件data.txt中读取数据。

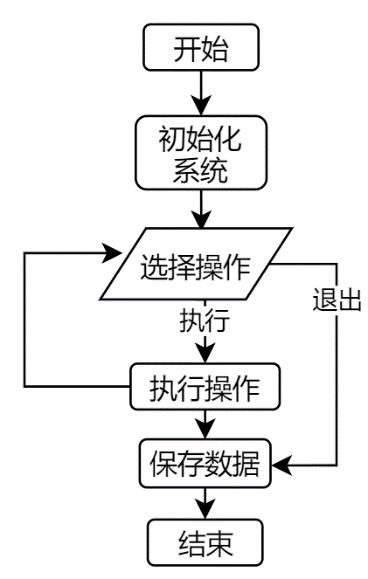
* void **SaveData**()

将数据库保存到本地文件data.txt中，以便下次打开系统操作数据。

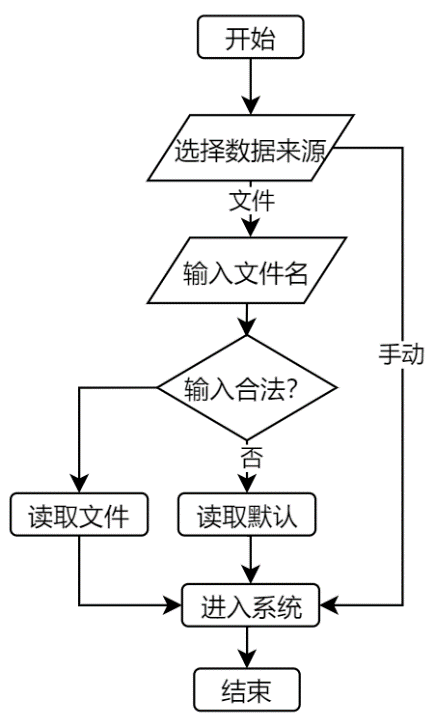
通过以上的类以及数据库类，并且加以耦合，即完成了考试报名系统。

# 3 项目实现

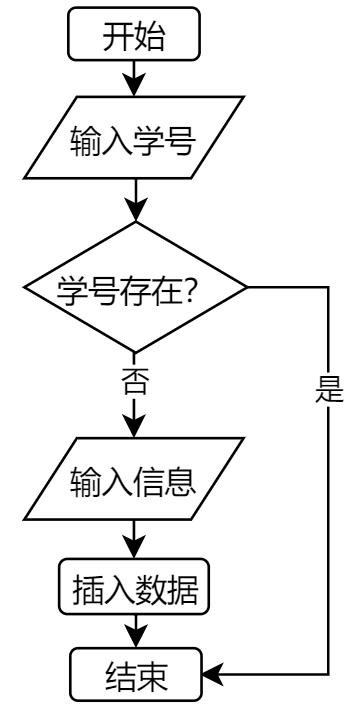
## 3.1 项目主体功能流程图



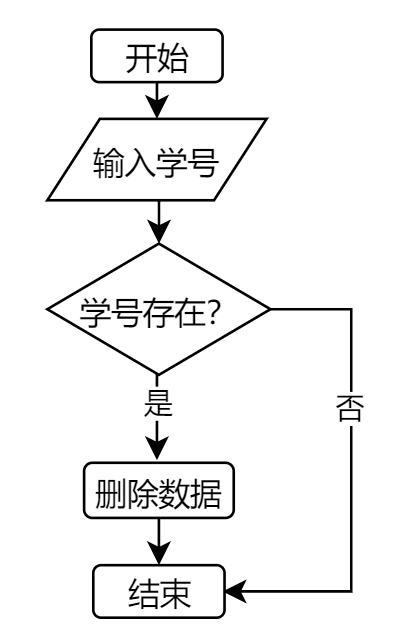
## 3.2 项目初始化流程图



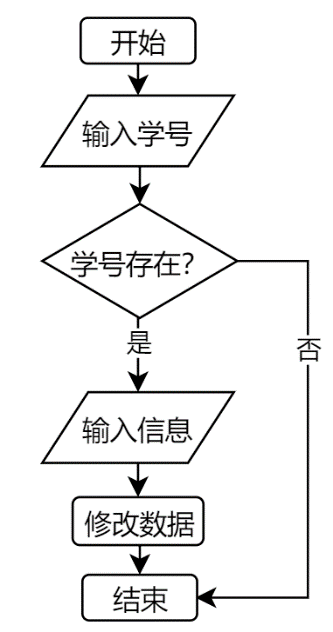
# 3.3 插入数据流程图



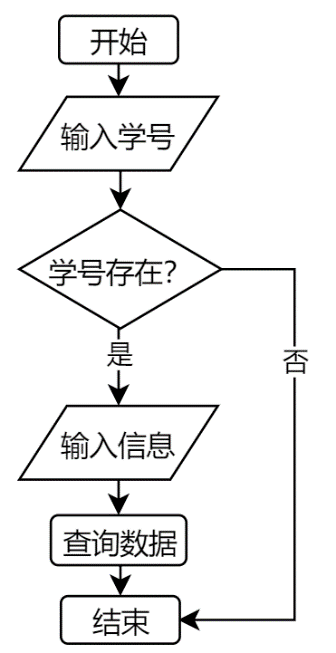
## 3.4 删除数据流程图



## 3.5 修改数据流程图



## 3.6 查询数据流程图



## 3.7 统计数据

项目提供了函数用于统计系统内部的数据，包括考生总数、男女比例、各年龄人数等信息。代码如下所示：

# 4 项目测试

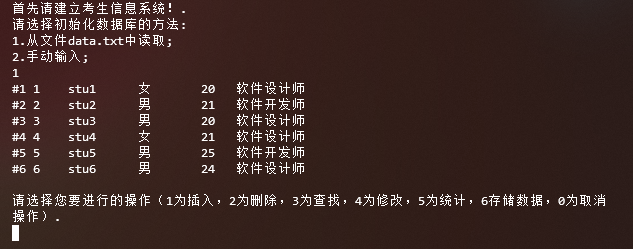
## 4.1 项目初始化测试

### 4.1.1 读取文件测试

测试用例：1

预期结果：从本地文件中读取初始化的数据库

实验结果：



### 4.1.2 手动输入测试

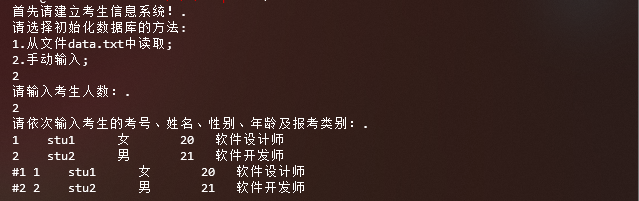
测试用例：2

1 stu1 女 20 软件设计师

2 stu2 男 21 软件开发师

预期结果：空数据库

实验结果：



## 4.2 插入测试

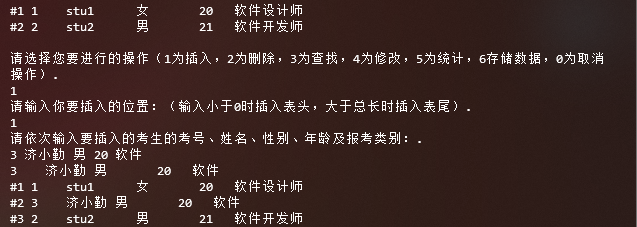
测试用例：

1

3 济小勤 男 20 软件

预期结果：成功插入新数据后的数据库

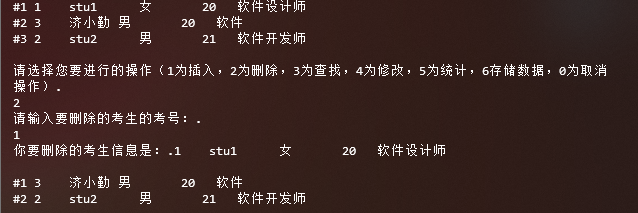
实验结果：



## 4.3 删除测试

测试用例：2 1

预期结果：删除学号为1的考生后的数据库

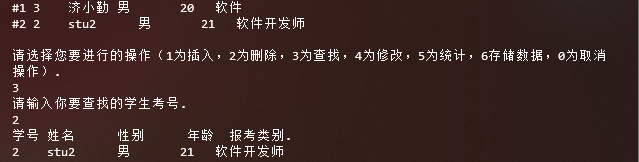
实验结果：  


## 4.4 查找测试

测试用例：3 2

预期结果：学号为2的考生的信息

实验结果：



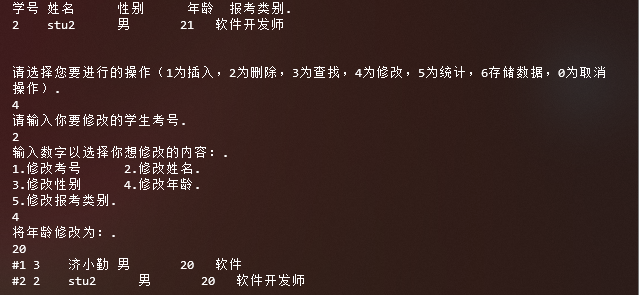
## 4.5 修改测试

测试用例：4 2

2 stu2 男 20 软件工程师

预期结果：考号为2的考生信息被修改后的数据库

实验结果：

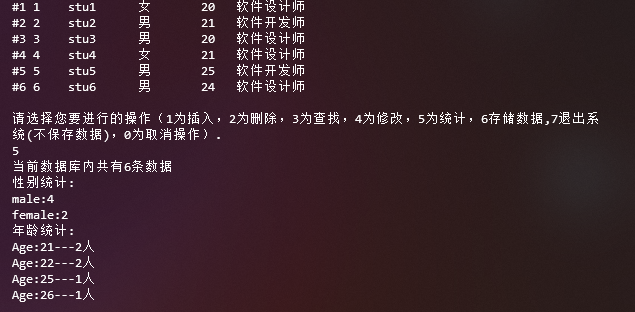


## 4.6 统计测试

测试用例：5

预期结果：统计当前数据库的信息

实验结果：



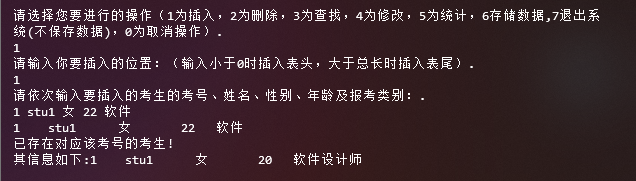
## 4.7 边界测试

### 4.7.1 插入已存在考生

测试用例：1 2

预期结果：提示无法插入

实验结果：

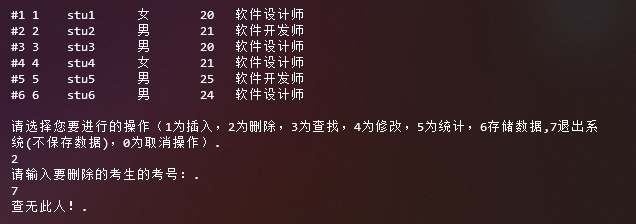


### 4.7.2 删除不存在考生

测试用例：2 1

预期结果：提示删除失败

实验结果：

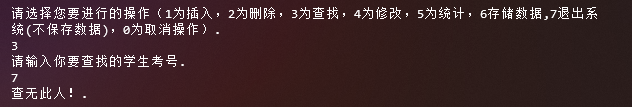


### 4.7.3 查找不存在考生

测试用例：3 1

预期结果：提示查找失败

实验结果：



### 4.7.4 修改不存在考生

测试用例：4 1

预期结果：提示修改失败

实验结果：

