**数据结构课程设计《考试报名系统》项目说明文档**

2253893 苗君文 软件工程

**1. 项目简介**

**1.1. 项目背景**

考试报名工作给各高校报名工作带来了新的挑战，给教务管理部门增加了很大的工作量。本项目是对考试报名管理的简单模拟，用控制台选项的选择方式完成下列功能：输入考生信息；输出考生信息；查询考生信息；添加考生信息；修改考生信息；删除考生信息；统计考生信息。

**1.2. 项目要求**

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，统计等功能。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. 输入格式**

本项目首先需输入现有考生人数及相关考生信息（包括准考证号，姓名，性别，年龄和报考类别）以建立初始考试系统。而后可以输入需要执行的操作以及根据相应操作输入对应的考生信息。

**1.4. 输出格式**

本项目针对输入有详细的输入错误处理，若用户输入错误，则程序会输出相关提示信息，并要求重新输入。而针对用户选择的操作，可以输出相应的操作结果并输出当前的考试系统内所有信息。

**2. 设计思路**

**2.1. 数据结构设计**

本项目是制作一个考试系统，需要存储大量的考生信息（包括准考证号，姓名，性别，年龄和报考类别），并且要能满足对大量数据的查找，插入，修改，删除，统计等功能，所以选择的数据结构要能相对快速地完成以上功能。因此，本项目采用单链表来存储考生的信息。

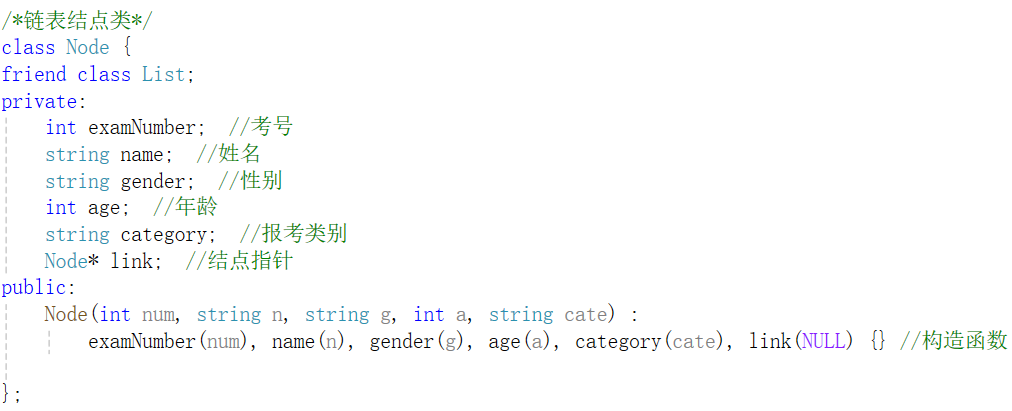
选择单链表的原因如下：单链表支持动态内存分配，可以根据实际需要在运行时动态申请和释放内存，这在考生人数不确定的情况下可以灵活地调整内存占用，只占用实际存储考生信息所需的内存，避免了对于预先分配数组可能导致的空间浪费；同时，单链表在插入或删除结点的操作效率高，仅需修改相邻结点的指针，而不需要移动其他的结点，非常便于插入、删除考生信息。

项目中实际采用的是带有单独的头结点的单链表，在程序中，使用first成员变量指向头结点，而头结点并不存储实际的考生信息，而是作为链表的起始位置。当链表为空时，first为NULL；而在链表非空时，first指向第一个存储考生信息的结点。这样，插入、删除第一个考生信息等操作就可以通过修改头结点的指针，而不必单独处理链表为空的情况，因此提高了代码的可维护性。

**2.2. 类设计**

为了实现考试报名系统的功能，选择使用链表结点Node和链表List 两个类。这两个类之间采用复合方式使用。Node类负责存储和表示当个考生的信息，而List类提供了对考生信息链表的各种管理操作，这种设计使得系统更加模块化，易于维护和扩展。

**2.2.1. Node类（链表结点类）**



Node结点是链表的基本单元，用于存储每个考生的信息。

Node结点类包含以下成员变量：、

* examNumber：考号。
* name：姓名，存储考生的姓名。
* gender：性别，表示考生的性别。
* age：年龄，记录考生的年龄。
* category：报考类别，表示考生报考的类别。
* link：指向下一个结点的指针，形成链表的连接。

包含以下成员函数：

* 构造函数：接受考号、姓名、性别、年龄、报考类别作为参数，初始化相应的成员变量，并将link设置为NULL。

**2.2.2. List类（链表类）**



List 类用于管理考生信息的链表，包含对链表的基本操作。其中，头结点的存在简化了链表操作的逻辑；Insert()、RemoveValue()、Modify() 等函数通过遍历链表找到目标结点，进行插入、删除、修改等操作；统计函数通过遍历链表计算各种统计信息，如总人数、性别分布、年龄分布等。

List链表类包含以下成员变量：、

* first: 指向头结点的指针。
* last: 指向链表最后一个结点的指针。

包含以下成员函数：

* 构造函数：创建一个头结点，将 first 和 last 都指向这个头结点。
* MakeEmpty(): 删除链表中除头结点外的所有结点，释放内存。
* 析构函数：调用 MakeEmpty() 来释放链表的内存。
* getFirst(): 返回第一个结点的指针。
* Insert(): 在指定位置插入新的考生信息。
* FindValue(): 在链表中搜索指定考号的结点的前一个结点。
* FindCurrentValue(): 查找指定考号的结点。
* Display(): 输出所有考生信息。
* RemoveValue(): 从链表中删除指定考号的结点。
* Modify(): 修改指定考号的考生信息。
* display1stu(): 展示一行考生信息。
* countTotalStudents(): 统计考生总人数。
* countGender(): 统计性别分布。
* countAge(): 统计年龄分布。

**2.3. 主程序设计**

首先，创建考生信息链表 List slist并输出欢迎信息和建立考生信息系统的提示。而后要求输入考生人数和考生信息，初始化考生信息链表，并输出初始化后的考生信息。之后，进入主循环，根据用户选择执行相应的操作，直到用户选择退出程序。在循环中调用相应的链表操作函数，如插入、删除、查询、修改、统计等，然后输出相应的结果。

**2.4. 输入错误处理设计**

**2.4.1. 单个参数的输入错误处理**

函数dealInputError使用 while 循环不断尝试获取用户输入，直到输入满足要求。使用 cin.fail() 及min、max来判断输入是否出错，检查输入是否在有效范围内。如果输入无效，则输出错误信息，清除输入缓冲区并忽略之后的字符，并重新输出输入的提示信息。通过一个字符来获取输入一个数之后的字符以检查输入字符的个数，若不正确，也输出错误信息，并重新输出输入的提示信息，执行相应清除操作。如果输入有效，则跳出循环。

此错误处理可以用在输入操作选项、输入要建立的学生的个数，并辅助用于输入查找、删除的学生考号、输入插入的位置等环节。

**3. 功能实现**

**3.1. 主体功能实现**

主体功能负责整个考试报名系统的运行，具体包括以下步骤：

（1）欢迎信息和初始化

* 输出欢迎信息。
* 创建一个空的考生信息链表 List slist。
* 输入考生人数 stuNums。

（2）初始化考生信息

* 循环输入每个考生的信息，包括考号、姓名、性别、年龄和报考类别。
* 调用链表的 Insert 函数，在链表尾部插入每个考生的信息。

（3）输出初始化后的考生信息

* 调用链表的 Display 函数，输出初始化后的考生信息。

（4）主循环

* 进入程序的主循环，等待用户选择操作。

（5）选择操作

* 调用 menu 函数输出菜单，等待用户选择。
* 根据用户选择调用相应的功能函数，或者退出程序。

（6）退出程序

* 输出退出信息。

**3.2. 初始化建立功能**

初始化建立功能主要用于创建考生信息链表，包括以下步骤：

（1）创建空的考生信息链表 List slist。

（2）输入考生人数 stuNums。

（3）循环输入每个考生的信息，包括考号、姓名、性别、年龄和报考类别，并使用Insert函数插入链表。

（4）调用链表的Display函数，输出初始化后的考生信息。

**3.3. 插入功能**

插入功能用于在链表的指定位置插入新的考生信息，包括以下步骤：

（1）输入要插入的位置 pos。

（2）输入新考生的信息，包括考号、姓名、性别、年龄和报考类别。

（3）调用链表的Insert函数，在指定位置插入新考生信息。Insert函数中，首先判断插入的位置是否有效，若无效则输出提示信息并返回；若为有效位置，则创建新节点，根据插入的位置为表头及其他位置来调整指针的位置完成插入。

（4）调用链表的Display函数，输出插入后的考生信息链表。

**3.4. 删除功能**

删除功能用于从链表中删除指定考生的信息，包括以下步骤：

（1）输入要删除的考生的考号 Rnum。

（2）调用链表的 RemoveValue 函数，删除指定考生信息。RemoveValue函数中首先根据考号找到该考生的结点，找不到则输出提示信息并退出。找到则可以移动指针以删除对应的考生信息。

（3）调用链表的Display函数，输出删除后的考生信息链表。

**3.5. 查找功能**

查找功能用于根据考号在链表中查找指定考生的信息，包括以下步骤：

（1）输入要查找的考生的考号 Fnum。

（2）调用链表的FindCurrentValue函数，查找指定考生信息。FindCurrentValue函数中，直接遍历链表，找到考号为 Fnum的结点并返回，未找到则输出相关提示信息。

（3）调用链表的display1stu函数，输出查找到的考生信息。

**3.6. 修改功能**

修改功能用于根据考号修改链表中指定考生的信息，包括以下步骤：

（1）输入要修改的考生的考号 Mnum。

（2）调用链表的 Modify 函数，修改指定考生信息。Modify 函数中，调用 FindValue 函数查找考号为 Mnum 的结点的前一个结点，若找到则修改该结点的信息；若未找到则输出提示信息。

（3）调用链表的Display函数，输出修改后的考生信息链表。

**3.6. 统计功能**

统计功能用于对考生信息进行统计，包括以下步骤：

（1）调用链表的Display函数，输出所有考生的信息。

（2）调用链表的countTotalStudents函数，输出考生总人数。

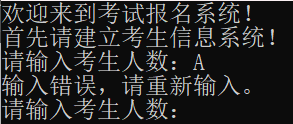
（3）调用链表的countGender函数，输出男女考生人数。

（4）调用链表的countAge函数，输出不同年龄段考生人数。

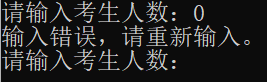
**4. 测试结果（包括边界测试）**

**4.1. 输入考生人数**

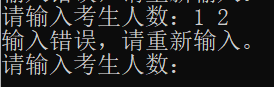
（1）输入的类型错误



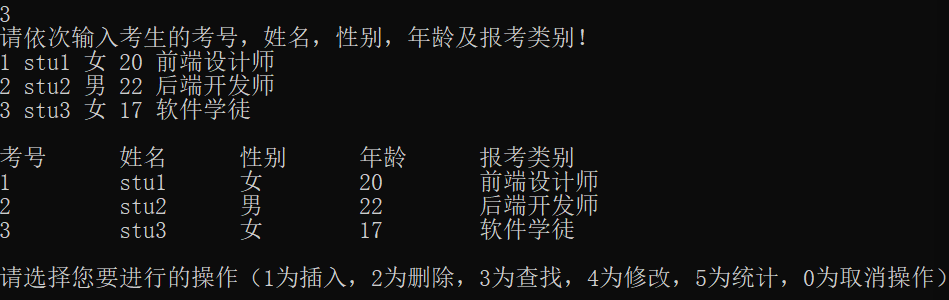
（2）输入的范围错误



（3）输入的个数超过一个



（4）输入正确即可依次输入考生的各种信息，并输出现有的考生信息

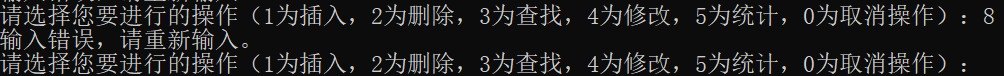


**4.2. 输入操作序号**

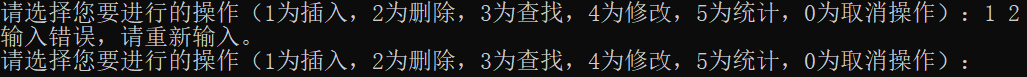
（1）输入的类型错误



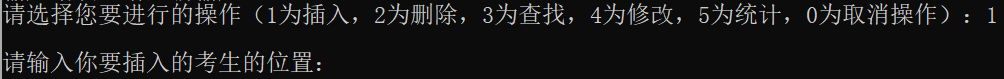
（2）输入的范围错误



（3）输入的个数超过一个



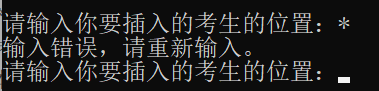
（4）输入正确即可进入对应的操作



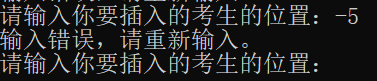
**4.3. 插入操作（选项1）**

**4.3.1. 输入要插入考生的位置**

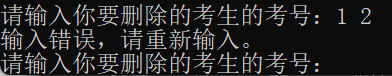
（1）输入的类型非法



（2）输入的范围非法

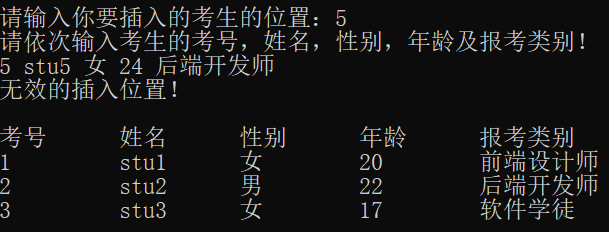


（3）输入的个数超过一个



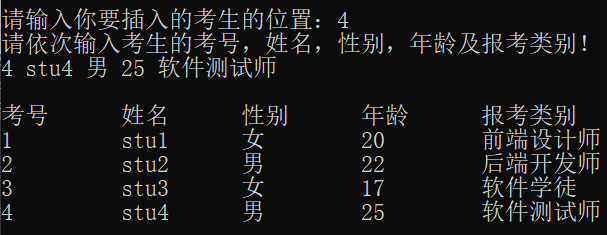
**4.3.2. 输入的位置合法的结果**

（1）输入的位置无效（在现有的位置之外）

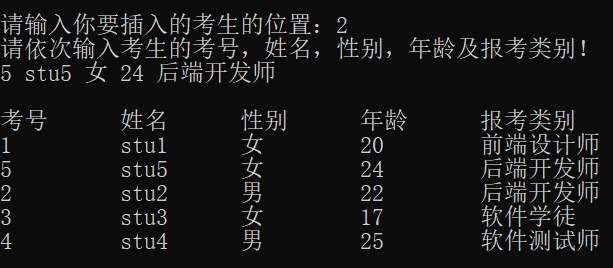


（2）输入的位置有效

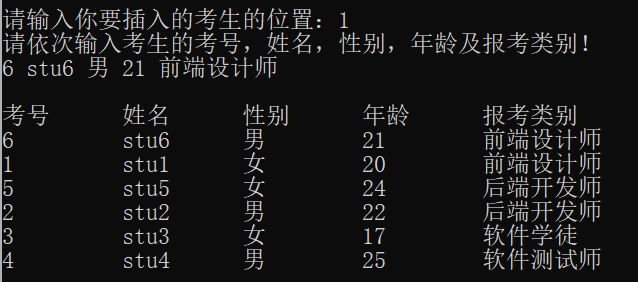
a. 插入在最后



b. 插入在中间



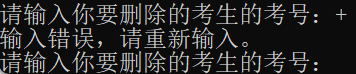
c. 插入在最前



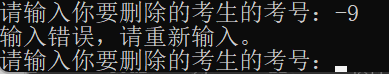
**4.4. 删除操作（选项2）**

**4.4.1. 输入要删除的考生的考号**

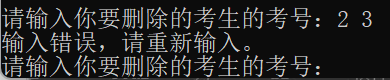
（1）输入的类型非法



（2）输入的范围非法

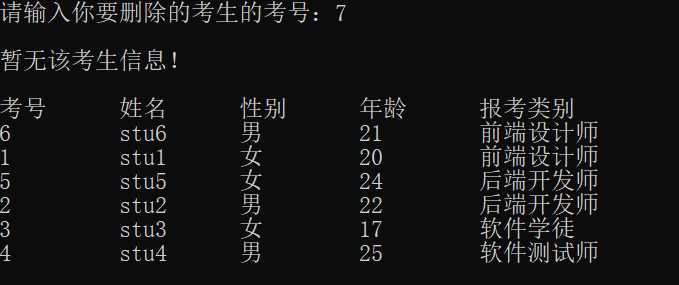


（3）输入的个数超过一个

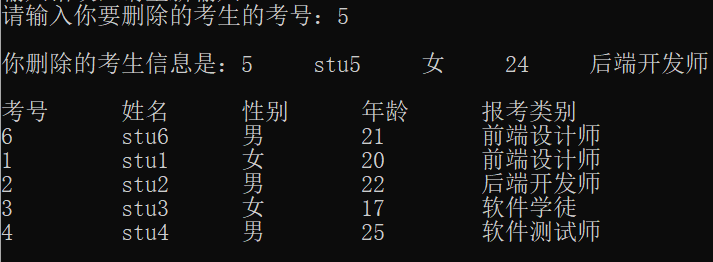


**4.4.2. 输入的考号合法的结果**

（1）输入的考号不在系统内，提示相关信息



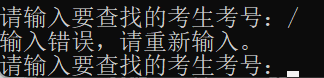
（2）输入的考号在系统中，输出删除的内容



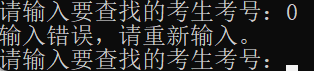
**4.5. 查找操作（选项3）**

**4.5.1. 输入要查找的考生考号**

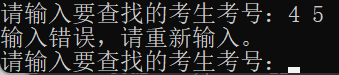
（1）输入的类型非法



（2）输入的范围非法

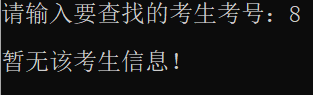


（3）输入的个数超过一个

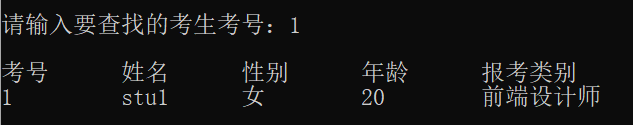


**4.5.2. 输入的考号合法的结果**

（1）查找失败，提示相关信息



（2）查找成功，输出该考生信息



**4.6. 修改操作（选项4）**

**4.6.1. 输入要修改的考生考号**

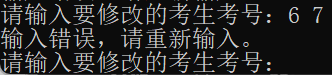
（1）输入的类型非法



（2）输入的范围非法

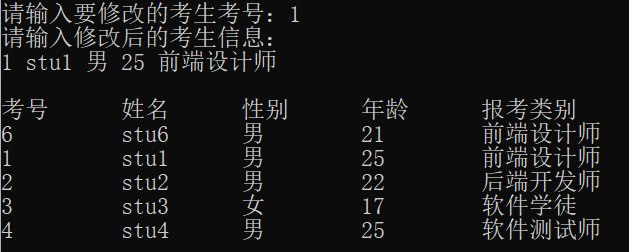


（3）输入的个数超过一个

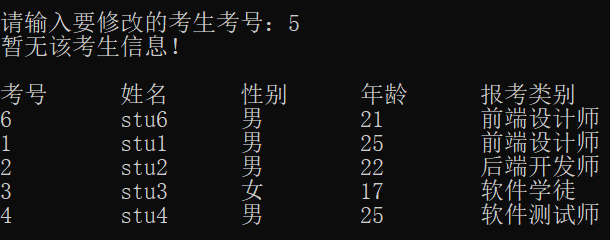


**4.6.2. 输入的考号合法的结果**

（1）考号存在于系统中，修改成功

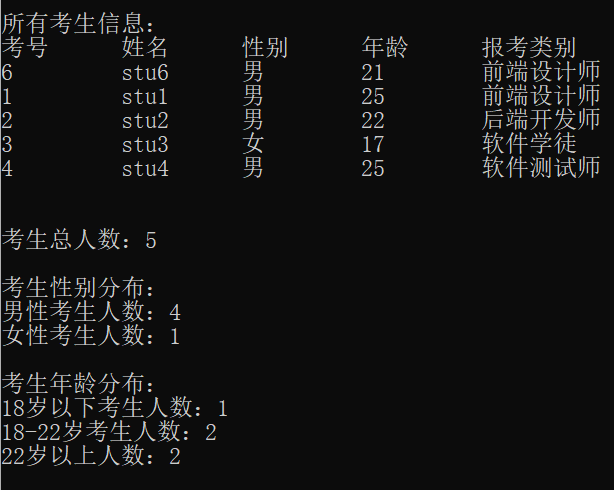


（2）考号不存在于系统中，修改失败

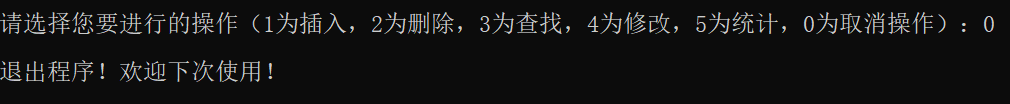


**4.7. 统计操作（选项5）**

输出所有考生信息及统计结果。统计内容包括考生总人数，考生性别分布以及考生的年龄分布。



**4.8. 取消操作（选项0）**



**4.9. 程序的测试结果总览（此处仅展示输入正确有效的情况）**

