**项目说明文档**

**数据结构课程设计**

**——考试报名系统**

### 作 者 姓 名 尹诚成

学 号 2351279

指 导 教 师 张 颖

学 院 专 业 计算机科学与技术学院 软件工程



二〇二三 年 十二 月 十三 日

**目录**

[1.项目分析 1](#_bookmark0)

* 1. .[项目背景分析 1](#_bookmark1)
  2. .[项目需求分析 1](#_bookmark2)
  3. .[项目功能分析 1](#_bookmark3)

1.3.1.录入考生信息功能 1

1.3.2.输出考生信息功能 1

1.3.3.插入考生功能 1

1.3.4.删除考生功能 1

1.3.5.查询考生功能 1

1.3.6.修改考生功能 2

1.3.7.统计考生功能 2

1.3.8.异常处理功能 2

[2.项目设计 2](#_bookmark12)

* 1. .[数据结构设计 2](#_bookmark13)
  2. .[结构体与类设计 2](#_bookmark14)

2.2.1.ExamerInfo 结构体的设计 2

2.2.2 LinkNode 结构体的设计 3

2.2.3 List 类的设计 3

* 1. .[项目主体架构设计 4](#_bookmark32)

[3.项目功能实现 4](#_bookmark33)

3.1.项目主体架构的实现 4

3.2.录入考生信息功能的实现 5

3.3.输出考生信息功能的实现 5

3.4.插入考生信息功能的实现 5

3.5.删除考生功能的实现 6

3.6.查询考生功能的实现 6

3.7.修改考生功能的实现 6

3.8.统计考生功能的实现 7

4.项目测试 7

4.1.录入考生信息功能测试

4.2.输出考生信息功能测试

4.3.插入考生信息功能测试

4.4.删除考生功能测试

4.5.查询考试功能测试

4.6.修改考生功能测试

4.7.统计考生功能测试

[5.集成开发环境与编译运行环境 35](#_bookmark92)

1. **项目分析**
   1. 项目背景分析

考试报名是一项庞大的工作，涉及多个环节和大量考生信息的管理。传统的手工管理已经无法满足快速、高效的需求。考生的信息包括考号、姓名、性别、年龄、报考类别等多种属性。这些信息需要被准确、高效地录入、修改、查询和删除。

* 1. 项目需求分析

基于以上背景分析，本项目需要实现需求如下：

(1)实现对考生信息的录入、输出、查询、添加、修改和删除等功能，确保 数据的准确、高效管理；

(2)设计简单直观的控制台界面，使操作便捷、容易上手，适应不同用户的操作习惯；

(3)选择合适的数据结构，以支持对考生信息的高效操作，同时考虑信息的关联性和复杂度；

(4)实现异常处理机制，确保系统稳定性和安全性，避免因用户输入错误导致系统崩溃或信息丢失；

(5)设计系统以支持未来的扩展和功能增加，满足不同用户、不同应用场景下的需求。

* 1. 项目功能分析

本项目旨在通过模拟考试报名管理过程，实现对考生信息的录入、输出、查 询、添加、修改和删除等功能，从而实现对考生信息的高效管理。下面对项目的功能进行详细分析。

* + 1. 录入考生信息功能

允许用户输入考生的基本信息，包括考号、姓名、性别、年龄、报考类别等，并建立考生信息系统。程序需要验证输入的信息是否符合规范，例如考号是否为由若干数字字符组成的字符串、年龄是否为正整数等。

* + 1. 输出考生信息功能

能够输出已录入的考生信息，包括考号、姓名、性别、年龄、报考类别等。

* + 1. 插入考生功能

允许用户在已有考生信息的基础上继续添加新的考生信息，包括考号、姓名、性别、年龄、报考类别等。

* + 1. 删除考生功能

允许用户根据考号等关键信息选择要删除的考生信息，进行考生信息的删除操作。

* + 1. 查询考生功能

允许用户通过考号等关键信息进行查询，程序能够返回符合条件的考生信息。

* + 1. 修改考生功能

允许用户根据考号等关键信息选择要修改的考生信息，可以修改考生的姓名、性别、年龄、报考类别等。

* + 1. 统计考生功能

允许用户在考生信息系统中统计考生信息，如性别比例情况、年龄比例情况、报考类别比例情况等，以对所有考生的信息有更全面的掌握和了解。

* + 1. 异常处理功能

实现异常处理机制，处理用户可能输入的非法信息，确保系统的稳定性和安全性。

1. **项目设计**

2.1.数据结构设计

基于项目分析，考试报名系统的设计中选择使用链表作为数据结构而不是数组，主要基于以下几个考虑：

(1)动态大小需求：链表可以动态地分配内存，适应不同数量的考生信息， 而数组需要预先确定大小，可能会导致内存浪费或不足；

(2)插入和删除操作效率高：链表对于插入和删除操作效率较高，因为只需 要调整节点的指针即可，而数组需要移动元素，时间复杂度效率较高；

(3)频繁的数据修改：如果考生信息需要频繁修改，例如修改报名信息、取 消报名等，链表更适合，因为修改节点的指针比修改数组元素更高效；

(4)考虑系统维护的复杂度：考虑系统的维护和扩展性，链表可以更方便地 进行扩展和修改，而数组可能需要重新设计和实现。

链表适合在需要动态调整大小、频繁插入和删除操作的情况下使用，而数组更适合固定大小的情况。基于上述分析，在设计考试报名系统时，选择链表作为数据结构更合适。

2.2.结构体与类设计

2.2.1 ExamerInfo结构体的设计

ExamerInfo 结构体用于表示考生的基本信息，其中包括考号、姓名、性别、年龄和报考类别，其数据成员定义及含义如下：

char no[MAX\_SIZE]：考号

char name[MAX\_SIZE]：姓名

bool sex：性别

int age：年龄

char category[MAX\_SIZE]：报考种类

* + 1. LinkNode结构体的设计

LinkNode 结构体是一个用于构建链表节点的模板结构体。该结构体用于表示链表中的每个节点，其中包括节点存储的数据以及指向下一个节点的指针。本项目希望链表结点类可以直接访问链表结点，所以使用 struct 而不是 class 描述链表结点类。其数据成员和构造函数定义及含义如下：

T data：数据域，存储节点的数据。

LinkNode<T>\* link：指针域，指向下一个节点的指针。

LinkNode(LinkNode\* ptr = NULL)：构造函数，初始化指针域。

LinkNode(const T& item, LinkNode\* ptr = NULL)：构造函数，初始化数据域和指针域。

* + 1. List类的设计

该通用模板类 MyList 用于表示单链表。此链表以附加的头节点作为起点。链表节点由 MyLinkNode 结构体表示，其中包含数据和指向下一个节点的指针。该链表提供了一系列基本操作函数，包括节点的插入、删除、查找、访问等，以及链表的构造和析构，满足了链表的常见操作需求。其数据成员、构造函数、析构函数、公有成员函数，运算符重载定义及含义如下：

LinkNode\* first：指向链表的第一个节点（头节点）的指针。

LinkNode\* last：指向链表的最后一个节点的指针。

List()：默认构造函数，创建一个空链表。

List(const T& item)：转换构造函数，创建一个只包含一个元素的链表。 List(List& L)：复制构造函数，通过复制另一个链表创建新链表。

~List()：析构函数，释放链表的内存资源，包括所有节点的内存。

void makeEmpty()：清空链表，释放所有结点的内存。

int getLength()const：获取链表中节点的个数。

LinkNode<T>\* getHead()const：获取链表头节点的指针。

LinkNode<T>\* getTail()const：获取链表尾节点的指针。

LinkNode<T>\* search(T item)const：搜索链表中值为 item 的节点，返回指向该节点的指针，若不存在则返回 NULL。

LinkNode<T>\* locate(int i)const：返回链表中第 i 个节点的指针，若 i 超出链表长度或小于 0，则返回 NULL。

int findData(T& item)const：获取链表中值为 item 的节点所在的位置，并通过返回值返回。

bool getData(int i, T& item)：获取链表中第i个节点的数据，并通过引用返回。返回值为操作是否成功。

void setData(int i, T& item)：设置链表中第i个节点的数据。返回值为操作是否成功。

bool insert(int i, T& item)：在链表中第i个节点后插入新节点。返回值为操作是否成功。

bool remove(int i, T& item)：删除链表中第i个节点，并通过引用返回其数据。返回值为操作是否成功。

bool isEmpty()const：检查链表是否为空。

void output()const：输出链表中所有节点的数据。

void addFront(T& item)：在链表的开头插入一个新元素，并通过引用返回其数据。

void removeFront(T& item)：删除链表的第一个元素，并通过引用返回其数据。

List<T>& operator =(List<T>& L)：重载赋值运算符，用于将一个链表赋值给另一个链表。

* 1. 项目主体架构设计

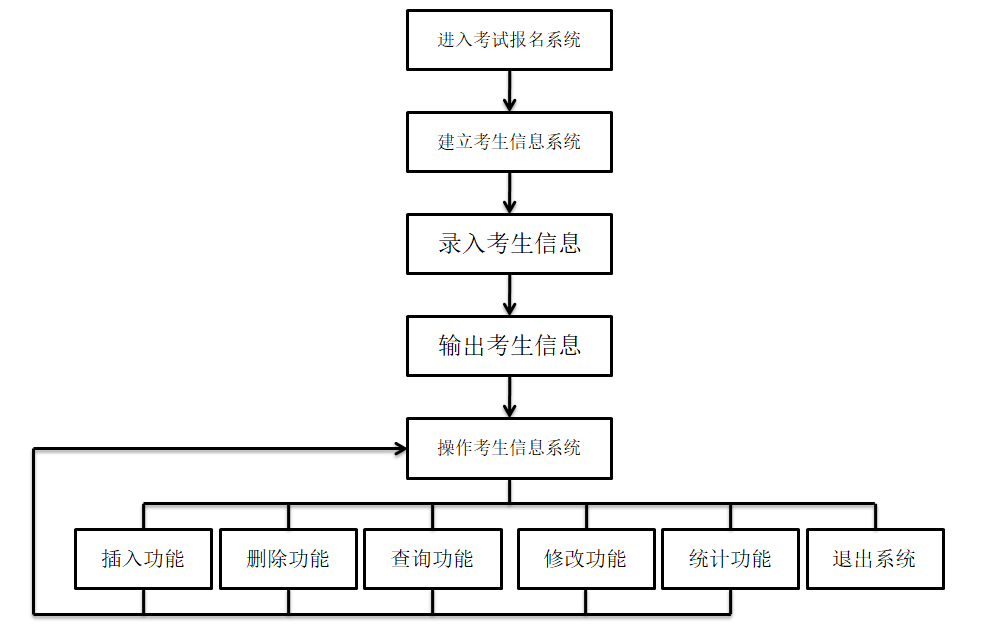


图 2.3 项目主体架构设计流程图

项目主体架构设计为：

(1)进入考试报名系统，并且建立考生信息系统；

(2)输入考生人数和依次录入考生信息；

(3)考生信息系统建立完成后输出所有考生信息；

(4)通过循环结构操作考生信息系统，调用包括插入、删除、查询、修改、 统计操作函数，这些函数实现了对学生信息的各种管理和操作；

(5)由用户选择退出考试报名系统。

1. **项目功能实现**

3.1.项目主体架构的实现

项目主体架构实现思路为：

(1)在main 函数中调用examinationSystem函数进入考试报名系统，并且建立考生信息系统；

(2)通过 inputPeopleNum 函数输入考生人数并进入buildExamerList函 数，在buildExamerList函数中通过for循环调用inputExamerInfo函数依次 录入考生信息；

(3)考生信息系统建立完成后调用outputMeterHeader和outputAllExamerInfo函数输出所有考生信息；

(4)通过循环结构操作考生信息系统，根据每次makeChoice函数返回的操作选项调用包括插入（insertExamer函数）、删除（removeExamer函数）、 查询（findExamer函数）、修改（modifyExamer函数）、统计 （statisticExamer函数）操作函数，这些函数实现了对学生信息的各种管理和操作；

(5)由用户选择退出考试报名系统。

3.2.录入考生信息功能的实现

录入考生信息功能的函数名为inputExamerInfo，其实现思路为：

(1)循环结构：使用一个无限循环，确保用户能够一直输入学生信息，直到信息输入合法并返回；

(2)创建临时学生信息对象：在每次循环开始时，创建一个临时的 ExamerInfo对象new\_examer，用于存储当前输入的学生信息；

(3)输入学生信息：输入学生的考号、姓名、性别、年龄、考试类别等信息，分别进行相应的格式验证；

(4)格式验证：对输入的各个字段进行格式验证，确保输入的格式合法。

(5)错误处理：如果输入的格式不合法，输出错误信息并清空输入缓冲区，继续下一个循环，等待用户重新输入；

(6)返回学生信息：如果输入的信息合法，将临时学生信息对象返回。

3.3.输出考生信息功能的实现

输出考生信息功能的函数名为outputMeterHeader和outputExamerInfo函数，其实现思路为：

(1)outputMeterHeader函数打印表格头部和考生信息

(2)通过函数重载的方式决定打印某个考生信息或全体考生信息。若参数为const ExamerIinfo examer，使用std::cout输出参数ExamerIinfo中的考生信息，该参数为一个ExamerIinfo对象，包含考生的考号、姓名、性别、年龄、报考类别。若参数为const List<ExamerIinfo>& examer\_list，则检查考生信息列表是否为空，若考生信息列表不为空，则输出全体考生信息，遍历链表并依次输出每个考生的信息，否则输出一行信息表示无考生信息；

(3)打印表格底部，用于结束表格的显示。在输出考生信息的过程中，重载了左移运算符<<,用于将ExamerIinfo类型的对象输出到指定的输出流std::ostream 中，并使用了std::setw设置字段宽度和std::setiosflags设置输出格式，确保输出的信息按指定的格式对齐。

3.4.插入考生功能的实现

插入考生功能的函数名为 insertExamer，其实现思路为：

(1)调用inputInsertLocation函数获取用户要插入考生的位置。这个位置是用户指定的，并且必须在合法的范围内（1到当前考生数量+1之间）；

(2)调用outputInputRequirement函数，向用户展示输入提示信息，要求用户输入待插入考生的信息。这个信息包括考生的考号、姓名、性别、年龄和报考类别；

(3)调用inputExamerInfo 函数，获取用户输入的考生信息，并将其存储在一个ExamerIinfo结构体中； (4)调用examer\_list.insert 函数，将新考生信息插入到考生信息系统中的指定位置。需要注意插入的位置需要减去1（在前一个位置之后1个位置插入）。

(5)调用outputMeterHeader和outputExamerInfo函数，将插入的考生信息和更新后的全体考生信息打印出来。

3.5.删除考生功能的实现

删除考生功能的函数名为removeExamer，其实现思路为：

(1)检查链表是否为空，若为空，则打印提示信息并退出；

(2)调用getExmaerNoAndPos函数获取用户待删除的考生的位置。这个位置是通过输入考生的考号来确定的，在getExmaerNoAndPos函数中首先输入待删除考生的考号并查找是否存在，然后调用findNoLocation函数遍历链表，根据考号查询考生位置；

(3)创建一个ExamerIinfo结构体对象temp\_examer用于存储待删除的考生信息；

(4)调用examer\_list.remove 函数，将位于指定位置的考生记录删除，并将删 除的考生信息存储在temp\_examer中。

(5)调用outputExamerInfo函数，将被删除的考生信息打印出来，以显示已删除的考生记录,通过再次调用outputExamerInfo函数，将更新后的全体考生信息打印出来，以显示已删除的考生记录已从系统中移除；

3.6.查询考生功能的实现

查询考生功能的函数名为findExamer，查询考生功能实现的思路为：

(1)检查链表是否为空，若为空，则打印提示信息并退出；

(2)调用getExmaerNoAndPos函数获取用户待删除的考生的位置。这个位置是通过输入考生的考号来确定的，在getExmaerNoAndPos函数中首先输入待删除考生的考号并查找是否存在，然后调用findNoLocation函数遍历链表，根据考号查询考生位置；

(3)创建一个ExamerIinfo结构体对象temp\_examer用于存储待查询的考生信息；

(4)调用examer\_list.getData函数，从系统中获取位于指定位置的考生记录，并将该考生信息存储在temp\_examer中。

(5)通过调用outputExamerInfo函数，将查找到的考生信息打印出来，以显示该考生的详细信息；

3.7.修改考生功能的实现

修改考生功能的函数名为modifyExamer，其实现思路为：

(1)检查链表是否为空，若为空，则打印提示信息并退出；

(2)调用getExmaerNoAndPos函数获取用户待删除的考生的位置。这个位置是通过输入考生的考号来确定的，在getExmaerNoAndPos函数中首先输入待删除考生的考号并查找是否存在，然后调用findNoLocation函数遍历链表，根据考号查询考生位置；

(3)创建一个ExamerIinfo结构体对象temp\_examer用于存储待修改的考生信息；

(4)调用examer\_list.remove函数，将待修改的考生信息删除，并将其存储在temp\_examer中。

(5)调用outputExamerInfo函数，将待修改的考生信息打印出来，以显示原始的考生信息；

(6)调用outputInputRequirement函数，向用户展示输入提示信息，要求用户输入修改 后考生的信息。这个信息包括考生的考号、姓名、性别、年龄和报考类别；

(7)调用inputExamerInfo函数，获取用户输入的考生信息，并将其存储在temp\_examer中； (8)调用examer\_list.insert函数，将修改后考生信息插入到考生信息系统中原来的位置； (9)调用outputExamerInfo函数，将修改后的考生信息和更新后的全体考生信息打印出来。

3.8.统计考生功能的实现

统计考生功能的函数名为statisticExamer，其实现思路为：

(1)检查链表是否为空，若为空，则打印提示信息并退出；

(2)调用examer\_list.getLength函数获取考生信息系统中的总学生数量examer\_nums；

(3)创建变量male\_num用于统计男性学生的数量，以及一个数组age\_num用于统计各个年龄的学生数量；

(4)创建一个整型数组outer，用于跟踪已访问的节点。初始化数组并将所有节点标记为未访问；

(5)统计各个考试类别的学生数量，并打印报考类别统计信息。通过遍历考生信息系统，逐一处理每个考生记录，并根据报考类别进行计数。同时，标记已经处理过的考试类别为已访问，以便只统计一次相同类别的考生；

(6)释放动态分配的内存，即释放outer数组的内存；

(7)统计男女学生的数量，通过遍历学生信息系统，统计男性和女性学生的数量，并计算比例。通过调用statisticSex函数，打印性别统计信息；

(8)统计不同年龄段的学生数量，通过遍历学生信息系统，统计各个年龄段的学生数量，并计算比例。通过调用statisticAge函数，打印年龄统计信息。

1. **项目测试**

4.1.录入考生信息功能测试

程序对考生信息的输入格式和数据类型进行了规定，具体如下图：

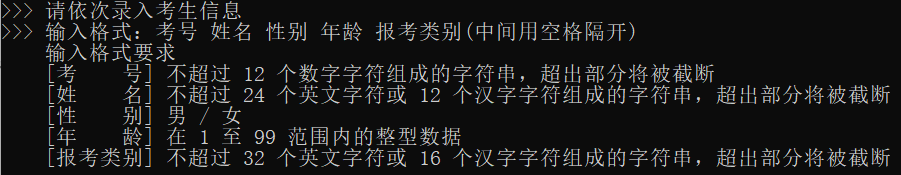


图 4.1.1 录入考生信息输入格式要求

考生的考号只能由数字组成，超出部分将被截断；若存在考号相同的考生，则重新输入。

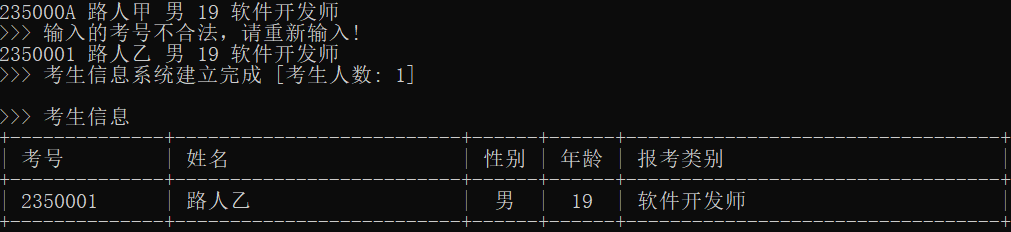


图4.1.1.2 录入考生考号功能测试（输入合法性）

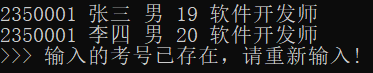


图 4.1.1.3 录入考生考号功能测试（考号唯一性）

当考生的姓名超出规定长度时，超出部分将被截断。

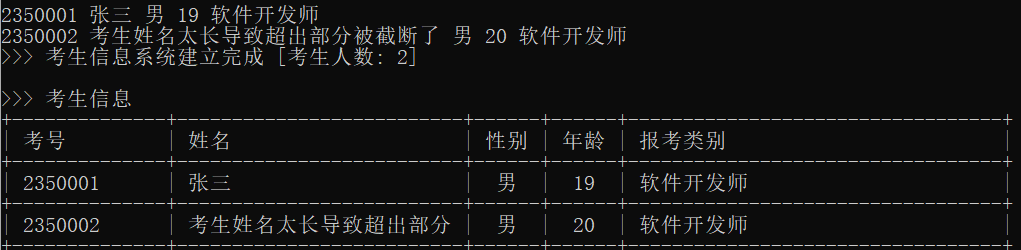


图 4.1.1.4 录入考生姓名功能测试

考生的姓名只能输入“男/女”。

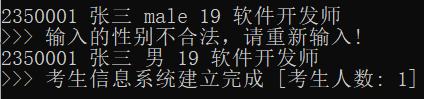


图 4.1.1.5 录入考生性别功能测试

考生的年龄只能为1到99的整形型。

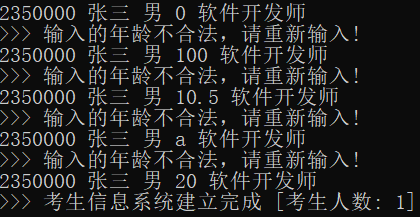


图 4.1.1.6 录入考生年龄功能测试

当考生的报考类别超出规定长度时，超出部分将被截断。

4.2.输出考生信息功能测试

下面的测试就三种情况的输出考生信息功能进行了测试

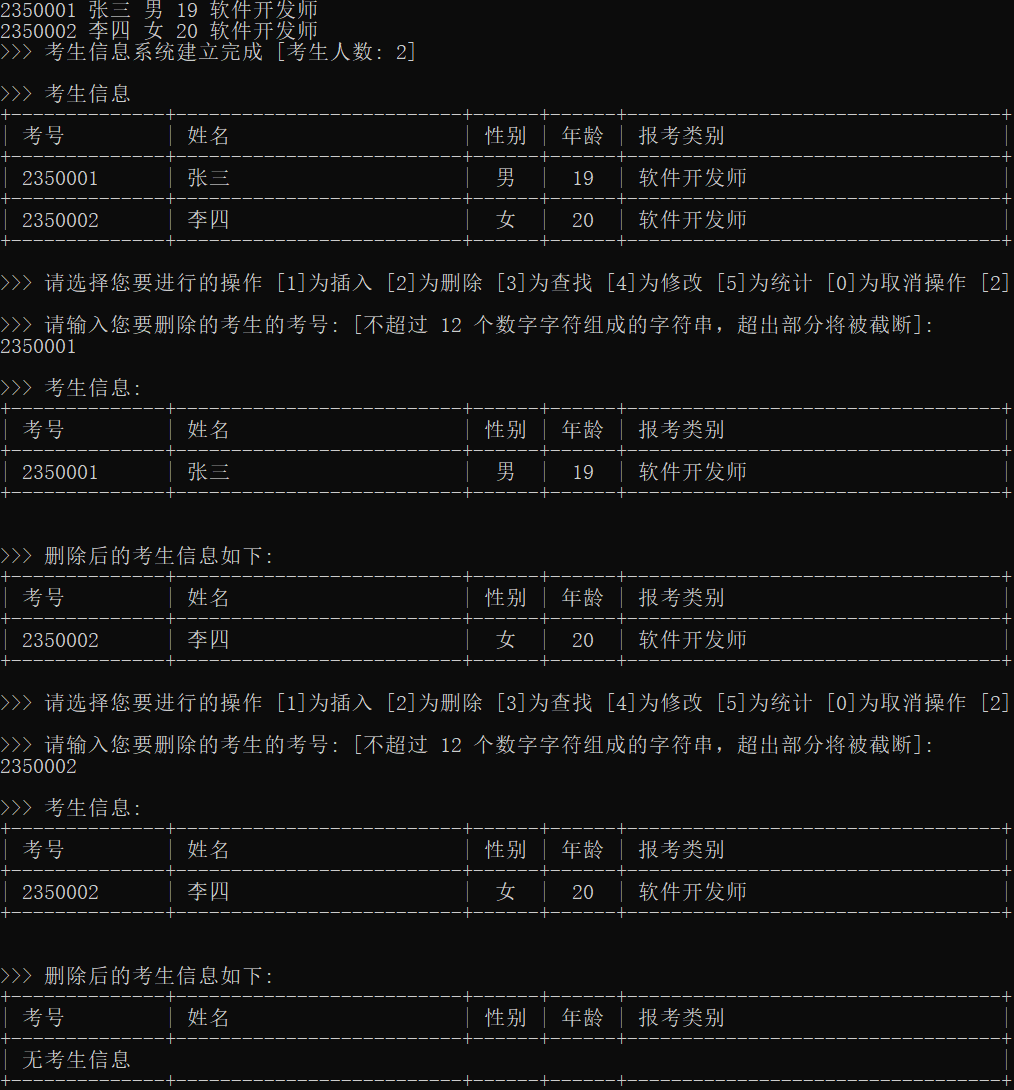


图 4.2 输出考生信息功能测试

4.3.插入考生功能测试

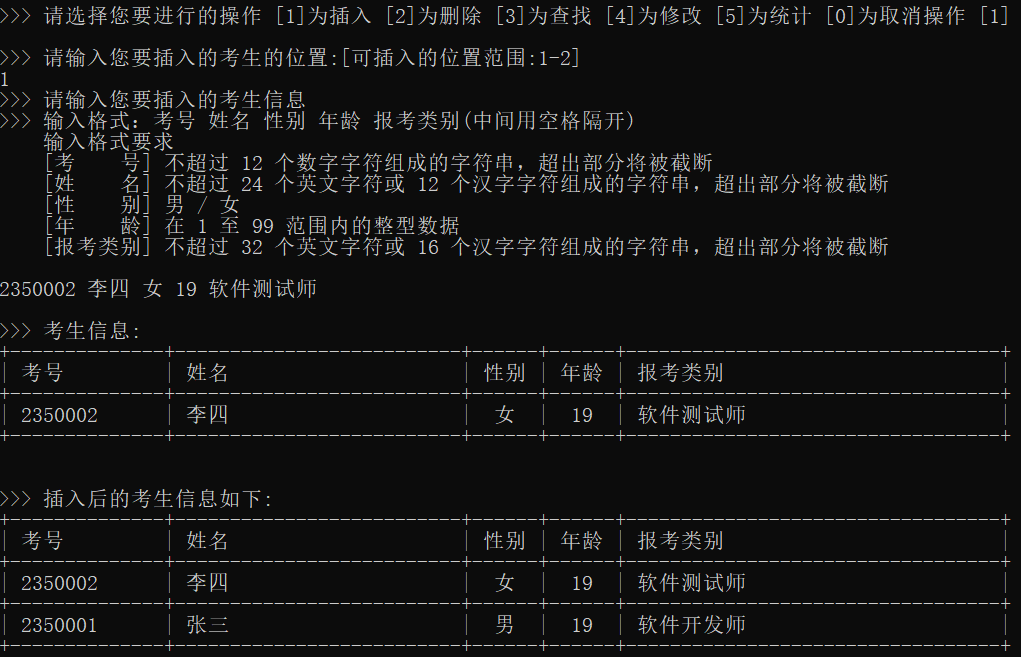


图 4.3 插入考生功能测试

4.4.删除考生功能测试

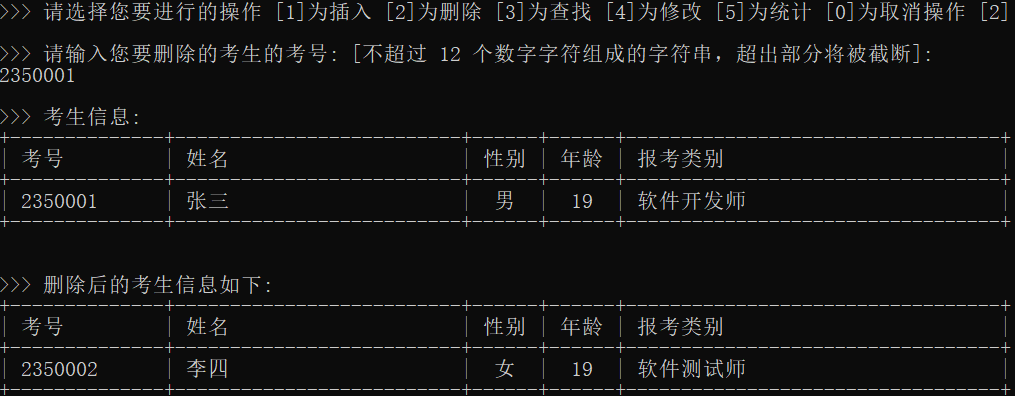


图 4.4 删除考生功能测试

4.5.查询考生功能测试

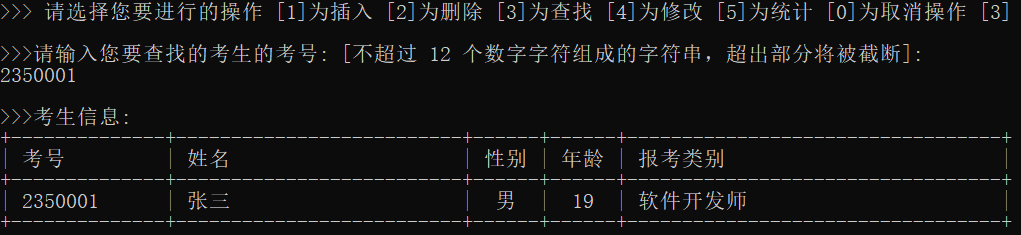


图 4.5 查询考生功能测试

4.6.修改考生功能测试

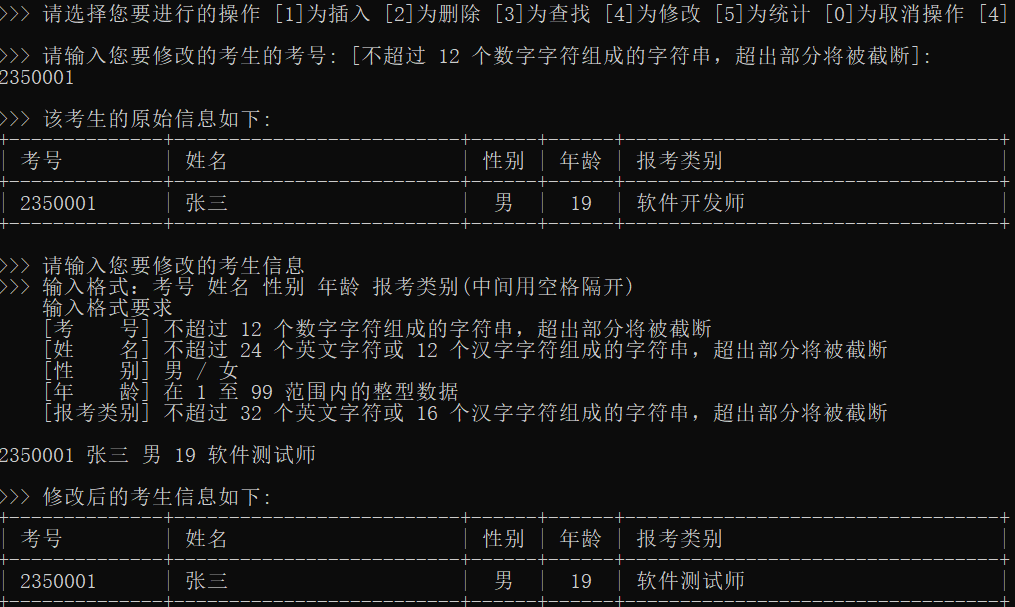


图 4.6 修改考生功能测试

4.7.统计考生功能测试

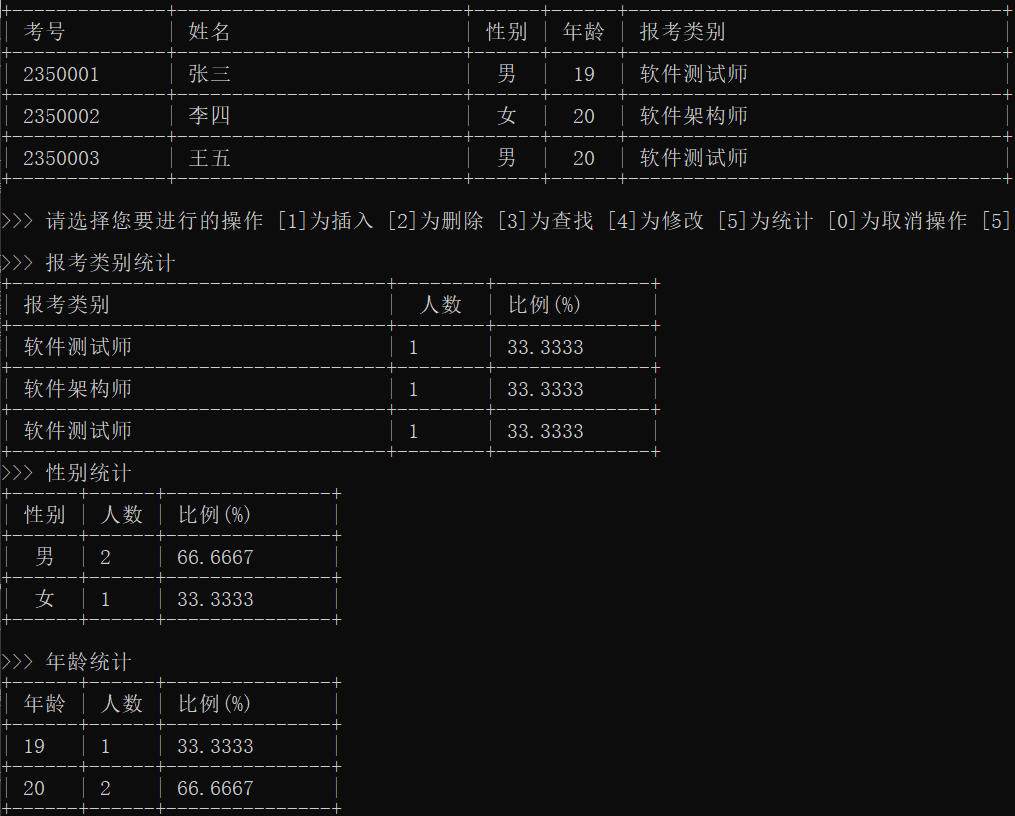


图 4.7 统计考生功能测试