项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名　林继申

学　　号　2250758

指 导 教 师　张　颖

学 院 专 业　软件学院　软件工程

同济大学
TONGJI UNIVERSITY

二〇二三 年 十二 月 十三 日

目录

1 项目分析

1.1 项目背景分析

考试报名是高校教务管理中一项至关重要的工作。随着教育系统的不断发展和进步，考试报名系统也需要不断升级和改进以适应现代社会的需求。

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。

1.2 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。

1.3 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。需要设计合适的数据结构、开发用户友好的控制台界面，并考虑系统的稳定性、安全性以及未来的扩展性。通过该项目的实施，可以提高考试报名管理的效率和准确性，为教务管理部门和考生提供更好的服务。下面对项目的功能进行详细分析。

1.3.1 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。

1.3.2 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。

1.3.3 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。

1.3.4 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。

1.3.5 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。

1.3.6 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。

1.3.7 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。

1.3.8 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。

2 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)不需要随机访问：如果系统不需要通过索引随机访问考生信息，而只是按顺序处理，链表可以满足需求；

(5)内存分配灵活性：链表的内存分配比较灵活，可以根据需求动态分配，而数组需要一次性分配固定大小的内存空间；

(6)避免数组扩展的开销：使用数组可能需要额外的扩展和拷贝操作，而链表避免了这种开销；

(7)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以及不需要随机访问的情况下使用，而数组更适合需要随机访问和固定大小的情况。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 StuInfo 结构体的设计

2.2.1.1 概述

StuInfo 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。

2.2.1.2 结构体定义

```
struct StuInfo {
    char no[MAX_SIZE] = { 0 };
    char name[MAX_SIZE] = { 0 };
    bool sex = true;
    int age = 0;
    char category[MAX_SIZE] = { 0 };
};
```

2.2.1.3 数据成员

char no[MAX_SIZE]：考号（第一位数字可以是 0，故使用 char 数组类型）

char name[MAX_SIZE]：姓名

bool sex：性别

int age：年龄

char category[MAX_SIZE]：报考类别

2.2.2 MyLinkNode 结构体的设计

2.2.2.1 概述

MyLinkNode 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。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）。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。

2.2.2.2 结构体定义

```
template <typename Type>
struct MyLinkNode {
    Type data;
    MyLinkNode<Type>* link;
    MyLinkNode(MyLinkNode<Type>* ptr = NULL) { link = ptr; }
    MyLinkNode(const Type& item, MyLinkNode<Type>* ptr = NULL)
{ data = item; link = ptr; }
};
```

2.2.2.3 数据成员

Type data：数据域，存储节点的数据

MyLinkNode<Type>* link：指针域，指向下一个节点的指针

2.2.2.4 构造函数

MyLinkNode(MyLinkNode<Type>* ptr = NULL);

构造函数，初始化指针域。

MyLinkNode(const Type& item, MyLinkNode<Type>* ptr = NULL);

构造函数，初始化数据域和指针域。

2.2.3 MyList 类的设计

2.2.3.1 概述

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，

简化了操作和提高了效率。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，满足了常见的链表操作需求。

2.2.3.2 类定义

```
template <typename Type>
class MyList {
private:
    MyLinkNode<Type>* first;
    MyLinkNode<Type>* last;
public:
    MyList();
    MyList(const Type& item);
    MyList(MyList<Type>& L);
    ~MyList();
    void makeEmpty(void);
    int getLength(void) const;
    MyLinkNode<Type>* getHead(void) const;
    MyLinkNode<Type>* getTail(void) const;
    MyLinkNode<Type>* search(Type item) const;
    MyLinkNode<Type>* locate(int i) const;
    bool getData(int i, Type& item) const;
    bool setData(int i, Type& item);
    bool insert(int i, Type& item);
    bool remove(int i, Type& item);
    void pushFront(Type& item);
    void popFront(void);
    bool isEmpty(void) const;
    void output(void) const;
    void splice(MyLinkNode<Type>* pos, MyList<Type>& other);
    void merge(MyList<Type>& other);
    void reverse(void);
    MyList<Type>& operator=(MyList<Type> L);
};
```

2.2.3.3 私有数据成员

`MyLinkNode<Type>* first`：指向链表的第一个节点（头节点）的指针

`MyLinkNode<Type>* last`：指向链表的最后一个节点的指针

2.2.3.4 构造函数

`MyList();`

默认构造函数，创建一个空链表。

`MyList(const Type& item);`

转换构造函数，创建一个只包含一个元素的链表。

`MyList(MyList<Type>& L);`

复制构造函数，通过复制另一个链表创建新链表。

2.2.3.5 析构函数

`~MyList();`

析构函数，释放链表的内存资源，包括所有节点的内存。

2.2.3.6 公有成员函数

`void makeEmpty(void);`

清空链表，释放所有节点的内存。

`int getLength(void) const;`

获取链表中节点的个数。

`MyLinkNode<Type>* getHead(void) const;`

获取链表头节点的指针。

`MyLinkNode<Type>* getTail(void) const;`

获取链表尾节点的指针。

`MyLinkNode<Type>* search(Type item) const;`

搜索链表中值为 `item` 的节点，返回该节点的指针，若不存在返回 `NULL`。

`MyLinkNode<Type>* 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`。

`bool getData(int i, Type& item) const;`

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。

`bool setData(int i, Type& 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, Type& 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, Type& 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。

```
void pushFront(Type& item);
```

在链表的开头插入一个新元素

```
void popFront(void);
```

删除链表的第一个元素。

```
bool isEmpty(void) const;
```

检查链表是否为空。

```
void output(void) const;
```

输出链表中所有节点的数据。

```
void splice(MyLinkNode<Type>* pos, MyList<Type>& other);
```

将另一个链表的元素合并到当前链表的指定位置处。

```
void merge(MyList<Type>& other);
```

将另一个已排序链表的元素合并到当前已排序链表中，保持有序性。

```
void reverse(void);
```

反转链表中的元素顺序。

2.2.3.7 运算符重载

```
MyList<Type>& operator=(MyList<Type> L);
```

重载赋值运算符，用于将一个链表赋值给另一个链表。

2.3 项目主体架构设计

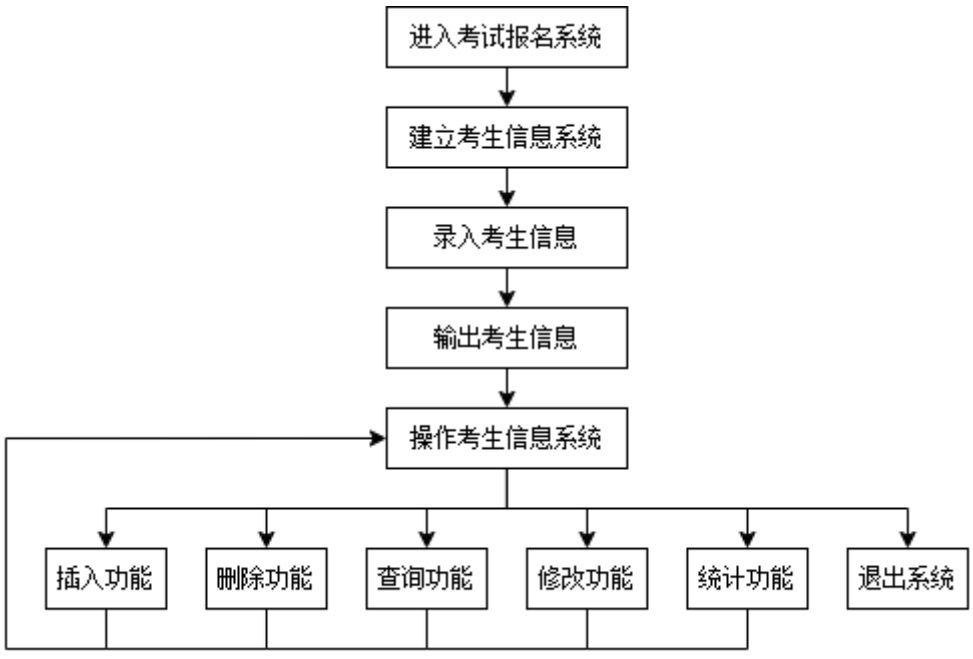

图 2.3.1 项目主体架构设计流程图

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。

3 项目功能实现

3.1 项目主体架构的实现

3.1.1 项目主体架构实现思路

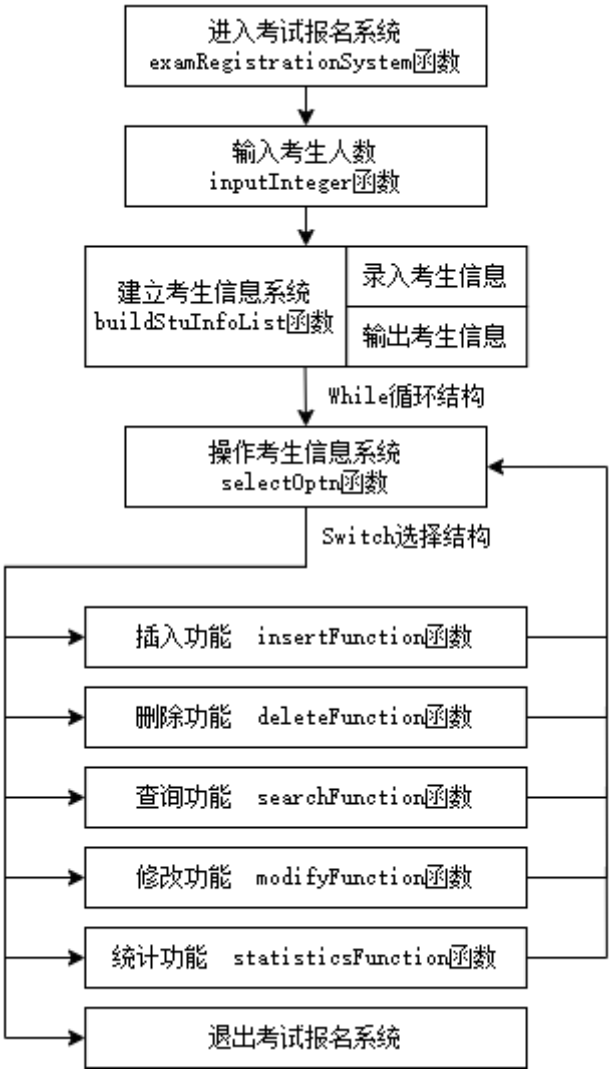

图 3.1.1.1 项目主体架构实现流程图

项目主体架构实现思路为：

⑴在 main 函数中调用 examRegistrationSystem 函数进入考试报名系统，并且建立考生信息系统；

⑵通过 inputInteger 函数输入考生人数并进入 buildStuInfoList 函数，在 buildStuInfoList 函数中通过 for 循环调用 inputStuInfo 函数依次录入考生信息；

⑶考生信息系统建立完成后调用 printStuInfo 函数输出所有考生信息；

⑷通过循环结构操作考生信息系统，根据每次 selectOptn 函数返回的操作选项调用包括插入（insertFunction 函数）、删除（deleteFunction 函数）、查询（searchFunction 函数）、修改（modifyFunction 函数）、统计（statisticsFunction 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；

⑸由用户选择退出考试报名系统。

3.1.2 项目主体架构核心代码

```
    void examRegistrationSystem(void)
    {
        /* System entry prompt */
        std::cout << "+----------------------------+" << std::endl;
        std::cout << "|        考试报名系统        |" << std::endl;
        std::cout << "|  Exam Registration System  |" << std::endl;
        std::cout << "+----------------------------+" << std::endl;

        /* Build a student information system */
        std::cout << std::endl << ">>> 请建立考生信息系统" << std::endl <<
std::endl;
        MyList<StuInfo> stuInfo;
        int stuNum = inputInteger(1, INT_MAX, "考生人数");
        buildStuInfoList(stuInfo, stuNum);

        /* Perform operations on the exam registration system */
        while (true) {
            switch (selectOptn()) {
                case 1:
                    insertFunction(stuInfo);
                    break;
                case 2:
                    deleteFunction(stuInfo);
                    break;
                case 3:
                    searchFunction(stuInfo);
```

```
                break;
            case 4:
                modifyFunction(stuInfo);
                break;
            case 5:
                statisticsFunction(stuInfo);
                break;
            default:
                std::cout << ">>> 考试报名系统已退出" << std::endl;
                return;
        }
    }
}
```

3.1.3 项目主体架构示例

```
+--------------------------+
|       考试报名系统       |
|  Exam Registration System  |
+--------------------------+

>>> 请建立考生信息系统

请输入考生人数 [整数范围: 1~2147483647]: 6

>>> 请依次录入考生信息:
    [输入格式] 考号 姓名 性别 年龄 报考类别 (用空格分隔数据)
    [考    号] 不超过 12 个数字字符组成的字符串, 超出部分将被截断
    [姓    名] 不超过 24 个英文字符或 12 个汉字字符组成的字符串, 超出部分将被截断
    [性    别] 男 / 女
    [年    龄] 在 1 至 99 范围内的整型数据
    [报考类别] 不超过 32 个英文字符或 16 个汉字字符组成的字符串, 超出部分将被截断

2250001 张三 男 18 软件设计师
2250002 李四 男 19 软件架构师
2250003 王五 女 20 软件开发师
2250004 Amy 女 20 软件设计师
2250005 Mike 男 18 软件开发师
2250006 Tom 男 19 软件测试师

>>> 考生信息系统建立完成 (考生人数: 6)

>>> 全体考生信息 (考生人数: 6)
+-------------+--------------------------+------+------+----------------------------------+
| 考号        | 姓名                     | 性别 | 年龄 | 报考类别                         |
+-------------+--------------------------+------+------+----------------------------------+
| 2250001     | 张三                     | 男   | 18   | 软件设计师                       |
| 2250002     | 李四                     | 男   | 19   | 软件架构师                       |
| 2250003     | 王五                     | 女   | 20   | 软件开发师                       |
| 2250004     | Amy                      | 女   | 20   | 软件设计师                       |
| 2250005     | Mike                     | 男   | 18   | 软件开发师                       |
| 2250006     | Tom                      | 男   | 19   | 软件测试师                       |
+-------------+--------------------------+------+------+----------------------------------+

>>> 菜单: [1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统

请选择操作类型: [0]

>>> 考试报名系统已退出
```

图 3.1.3.1 项目主体架构示例

3.2 录入考生信息功能的实现

3.2.1 录入考生信息功能实现思路

录入考生信息功能的函数名为 inputStuInfo，录入考生信息功能实现的思路为：

(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 StuInfo 对象 tempStu，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，如考号是数字，性别是“男”或“女”等；

(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。

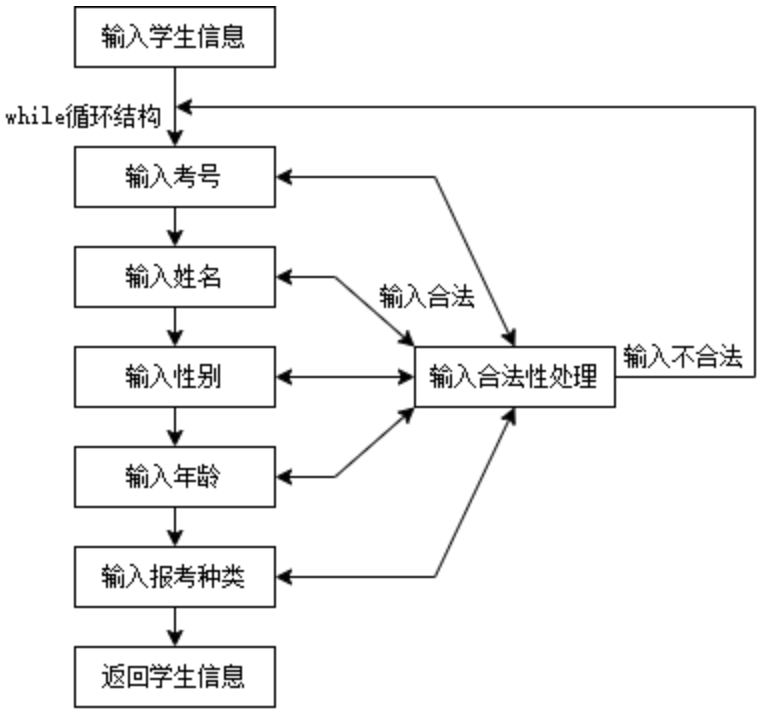

图 3.2.1.1 录入考生信息功能实现流程图

3.2.2 录入考生信息功能核心代码

```
StuInfo inputStuInfo(MyList<StuInfo>& stuInfo)
{
    while (true) {
        /* Create a temporary StuInfo object to store student
```

```
information */
            StuInfo tempStu;

            /* Input student number and validate its format */
            std::cin >> tempStu.no;
            truncateString(tempStu.no, 12);
            if (!isNumericString(tempStu.no)) {
                std::cerr << std::endl << ">>> 考号输入不合法，请重新输入当
前考生信息！" << std::endl << std::endl;
                std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
                continue;
            }
            else if (findPosByStuNo(stuInfo, tempStu.no)) {
                std::cerr << std::endl << ">>> 已存在考号相同的考生，请重新
输入当前考生信息！" << std::endl << std::endl;
                std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
                continue;
            }

            /* Input student name and validate its format */
            std::cin >> tempStu.name;
            truncateString(tempStu.name, 24);

            /* Input student sex and validate its format */
            char sex[MAX_SIZE] = { 0 };
            std::cin >> sex;
            if (!strcmp(sex, "男"))
                tempStu.sex = true;
            else if (!strcmp(sex, "女"))
                tempStu.sex = false;
            else {
                std::cerr << std::endl << ">>> 性别输入不合法，请重新输入当
前考生信息！" << std::endl << std::endl;
                std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
                continue;
            }

            /* Input student age and validate its format */
```

```
            double tempInput;
            std::cin >> tempInput;
            if (std::cin.good() && tempInput == static_cast<int>(tempInput)
&& tempInput > 0 && tempInput < 100)
                tempStu.age = static_cast<int>(tempInput);
            else {
                std::cerr << std::endl << ">>> 年龄输入不合法，请重新输入当
前考生信息！" << std::endl << std::endl;
                std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
                continue;
            }

            /* Input exam Category and validate its format */
            std::cin >> tempStu.category;
            truncateString(tempStu.category, 32);

            /* Return inputted student information */
            return tempStu;
        }
    }
```

3.2.3 录入考生信息功能示例

录入考生信息功能示例见项目主体架构示例（见图 3.1.3.1）。

3.3 输出考生信息功能的实现

3.3.1 输出考生信息功能实现思路

输出考生信息功能的函数名为 printStuInfo，输出考生信息功能实现的思路为：

(1)打印表格头部和考生信息的标题，同时显示考生总人数；

(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 StuInfo& stuInfo，使用 std::cout 输出参数 stuInfo 中的考生信息，该参数为一个 StuInfo 对象，包含考生的考号、姓名、性别、年龄、报考类别。若参数为 MyList<StuInfo>& stuInfo，则检查考生信息列表是否为空，若考生信息列表不为空，调用 stuInfo.output 函数输出全体考生信息，该函数会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 StuInfo 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。

重载左移运算符<<函数 std::ostream& operator<<(std::ostream& out, const StuInfo& stuInfo)的代码见 3.3.2 输出考生信息功能核心代码。

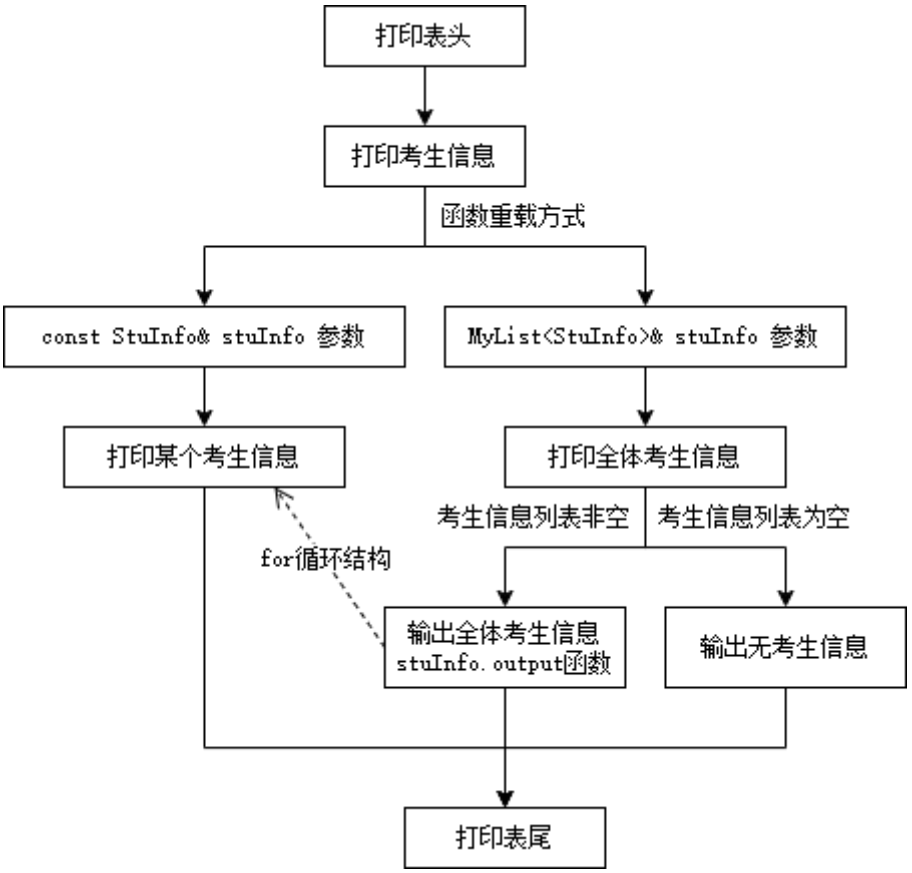

图 3.3.1.1 输出考生信息功能实现流程图

3.3.2 输出考生信息功能核心代码

```
std::ostream& operator<<(std::ostream& out, const StuInfo& stuInfo)
{
    out << std::setiosflags(std::ios::left)
        << "| " << std::setw(12) << stuInfo.no
        << " | " << std::setw(24) << stuInfo.name
        << " | " << (stuInfo.sex ? " 男 " : " 女 ")
        << " |  " << std::setw(3) << stuInfo.age
        << " | " << std::setw(32) << stuInfo.category
        << " |" << std::resetiosflags(std::ios::left);
    return out;
}
```

3.3.3 输出考生信息功能示例

输出考生信息功能示例见项目主体架构示例（见图 3.1.3.1）。

3.4 插入考生功能的实现

3.4.1 插入考生功能实现思路

插入考生功能的函数名为 insertFunction，插入考生功能实现的思路为：

(1)调用 inputInteger 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 inputPrompt 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 inputStuInfo 函数，获取用户输入的考生信息，并将其存储在一个 StuInfo 结构体中；

(4)调用 stuInfo.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 printStuInfo 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。

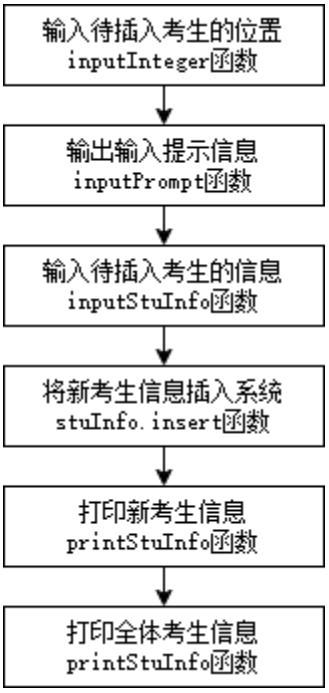

图 3.4.1.1 插入考生功能实现流程图

3.4.2 插入考生功能核心代码

```
void insertFunction(MyList<StuInfo>& stuInfo)
{
    int position = inputInteger(1, stuInfo.getLength() + 1, "待插入考生的位置");
    inputPrompt("请录入待插入考生信息:");
    StuInfo tempStu = inputStuInfo(stuInfo);
```

```
        stuInfo.insert(position - 1, tempStu);
        printStuInfo(tempStu);
        printStuInfo(stuInfo);
    }
```

3.4.3 插入考生功能示例

```
>>> 菜单: [1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统

请选择操作类型: [1]

请输入待插入考生的位置 [整数范围: 1~7]: 4

>>> 请录入待插入考生信息:
    [输入格式] 考号 姓名 性别 年龄 报考类别 (用空格分隔数据)
    [考    号] 不超过 12 个数字字符组成的字符串, 超出部分将被截断
    [姓    名] 不超过 24 个英文字符或 12 个汉字字符组成的字符串, 超出部分将被截断
    [性    别] 男 / 女
    [年    龄] 在 1 至 99 范围内的整型数据
    [报考类别] 不超过 32 个英文字符或 16 个汉字字符组成的字符串, 超出部分将被截断

2251001 李华 男 25 软件架构师

>>> 考生信息
+-------------+--------------------------+------+------+----------------------------------+
| 考号        | 姓名                     | 性别 | 年龄 | 报考类别                         |
+-------------+--------------------------+------+------+----------------------------------+
| 2251001     | 李华                     | 男   | 25   | 软件架构师                       |
+-------------+--------------------------+------+------+----------------------------------+

>>> 全体考生信息 (考生人数: 7)
+-------------+--------------------------+------+------+----------------------------------+
| 考号        | 姓名                     | 性别 | 年龄 | 报考类别                         |
+-------------+--------------------------+------+------+----------------------------------+
| 2250001     | 张三                     | 男   | 18   | 软件设计师                       |
| 2250002     | 李四                     | 男   | 19   | 软件架构师                       |
| 2250003     | 王五                     | 女   | 20   | 软件开发师                       |
| 2251001     | 李华                     | 男   | 25   | 软件架构师                       |
| 2250004     | Amy                      | 女   | 20   | 软件设计师                       |
| 2250005     | Mike                     | 男   | 18   | 软件开发师                       |
| 2250006     | Tom                      | 男   | 19   | 软件测试师                       |
+-------------+--------------------------+------+------+----------------------------------+
```

图 3.4.3.1 插入考生功能示例

3.5 删除考生功能的实现

3.5.1 删除考生功能实现思路

删除考生功能的函数名为 deleteFunction，删除考生功能实现的思路为：

(1)调用 getStuNoAndPos 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 getStuNoAndPos 函数中首先输入待删除考生的考号并进行输入合法性处理，然后调用 findPosByStuNo 函数遍历链表，根据考号查询考生位置；

(2)创建一个 StuInfo 结构体对象 delStu 用于存储待删除的考生信息；

(3)调用 stuInfo.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 delStu 中。如果删除成功，该函数会返回 true，否则返

回 false；

(4)如果删除成功（即返回 true），则通过调用 printStuInfo 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 printStuInfo 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；

(5)如果删除操作失败（即返回 false），则显示一条消息表明未查询到要删除的考生，因为指定的考生考号在系统中不存在。

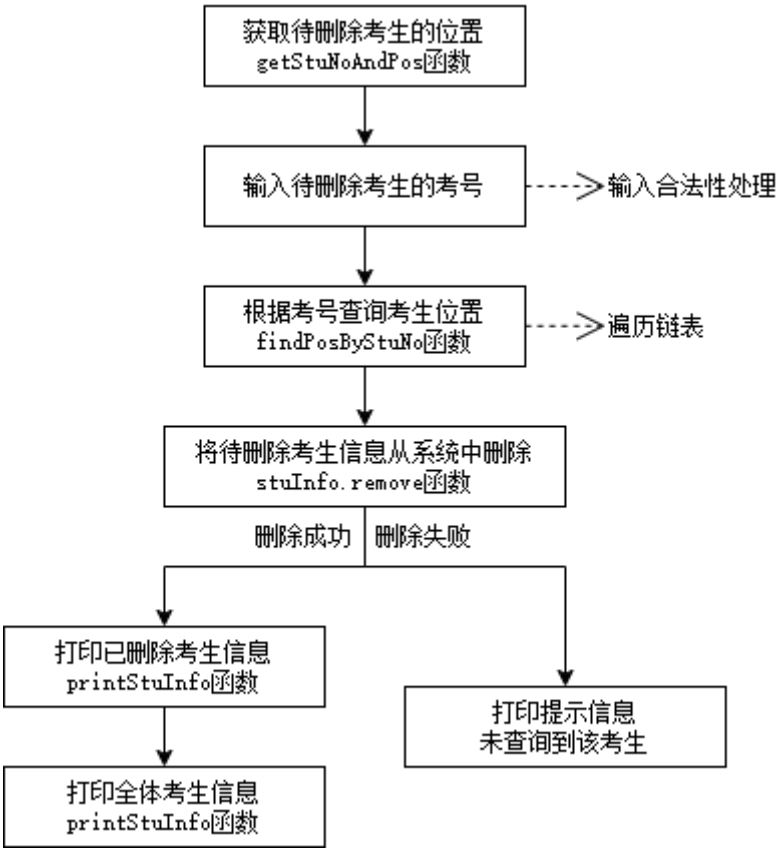

图 3.5.1.1 删除考生功能实现流程图

3.5.2 删除考生功能核心代码

```
void deleteFunction(MyList<StuInfo>& stuInfo)
{
    int position = getStuNoAndPos(stuInfo, "待删除考生的考号");
    StuInfo delStu;
    if (stuInfo.remove(position, delStu)) {
        printStuInfo(delStu);
        printStuInfo(stuInfo);
    }
    else
        std::cout << std::endl << ">>> 未查询到该考生" << std::endl;
}
```

3.5.3 删除考生功能示例

```
>>> 菜单：[1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统
请选择操作类型：[2]
请输入待删除考生的考号 [不超过 12 个数字字符组成的字符串，超出部分将被截断]：2250007
>>> 未查询到该考生
>>> 菜单：[1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统
请选择操作类型：[2]
请输入待删除考生的考号 [不超过 12 个数字字符组成的字符串，超出部分将被截断]：2250003
>>> 考生信息
+-------------+-------------------------+------+------+---------------------------------+
| 考号        | 姓名                    | 性别 | 年龄 | 报考类别                        |
+-------------+-------------------------+------+------+---------------------------------+
| 2250003     | 王五                    | 女   | 20   | 软件开发师                      |
+-------------+-------------------------+------+------+---------------------------------+

>>> 全体考生信息（考生人数：6）
+-------------+-------------------------+------+------+---------------------------------+
| 考号        | 姓名                    | 性别 | 年龄 | 报考类别                        |
+-------------+-------------------------+------+------+---------------------------------+
| 2250001     | 张三                    | 男   | 18   | 软件设计师                      |
| 2250002     | 李四                    | 男   | 19   | 软件架构师                      |
| 2251001     | 李华                    | 男   | 25   | 软件架构师                      |
| 2250004     | Amy                     | 女   | 20   | 软件设计师                      |
| 2250005     | Mike                    | 男   | 18   | 软件开发师                      |
| 2250006     | Tom                     | 男   | 19   | 软件测试师                      |
+-------------+-------------------------+------+------+---------------------------------+
```

图 3.5.3.1 删除考生功能示例

3.6 查询考生功能的实现

3.6.1 查询考生功能实现思路

查询考生功能的函数名为 searchFunction，查询考生功能实现的思路为：

(1)调用 getStuNoAndPos 函数获取用户待查询的考生的位置。这个位置是通过输入考生的考号来确定的，在 getStuNoAndPos 函数中首先输入待查询考生的考号并进行输入合法性处理，然后调用 findPosByStuNo 函数遍历链表，根据考号查询考生位置；

(2)创建一个 StuInfo 结构体对象 tempStu 用于存储待查询的考生信息；

(3)调用 stuInfo.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 tempStu 中。如果成功查询该考生记录，该函数返回 true，否则返回 false；

(4)如果查询成功（即返回 true），则通过调用 printStuInfo 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；

(5)如果查询失败（即返回 false），则显示一条消息表明未查询到要查询的考生，因为指定的考生考号在系统中不存在。

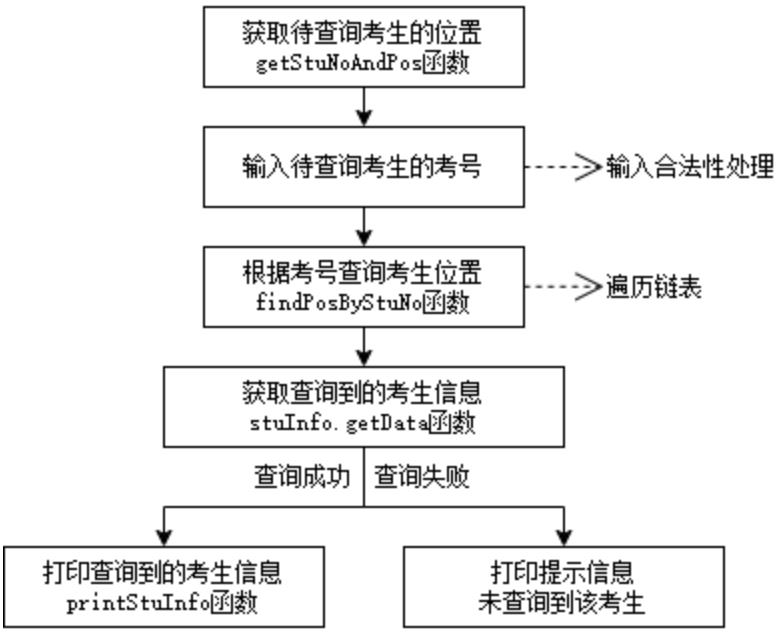

图 3.6.1.1 查询考生功能实现流程图

3.6.2 查询考生功能核心代码

```
void searchFunction(MyList<StuInfo>& stuInfo)
{
    int position = getStuNoAndPos(stuInfo, "待查询考生的考号");
    StuInfo tempStu;
    if (stuInfo.getData(position, tempStu))
        printStuInfo(tempStu);
    else
        std::cout << std::endl << ">>> 未查询到该考生" << std::endl;
}
```

3.6.3 查询考生功能示例

```
>>> 菜单: [1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统
请选择操作类型: [3]
请输入待查询考生的考号 [不超过 12 个数字字符组成的字符串，超出部分将被截断]: 2250007
>>> 未查询到该考生
>>> 菜单: [1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统
请选择操作类型: [3]
请输入待查询考生的考号 [不超过 12 个数字字符组成的字符串，超出部分将被截断]: 2250002
>>> 考生信息
+-------------+------------------------+------+------+--------------------------------+
| 考号        | 姓名                   | 性别 | 年龄 | 报考类别                       |
+-------------+------------------------+------+------+--------------------------------+
| 2250002     | 李四                   |  男  |  19  | 软件架构师                     |
+-------------+------------------------+------+------+--------------------------------+
```

图 3.6.3.1 查询考生功能示例

3.7 修改考生功能的实现

3.7.1 修改考生功能实现思路

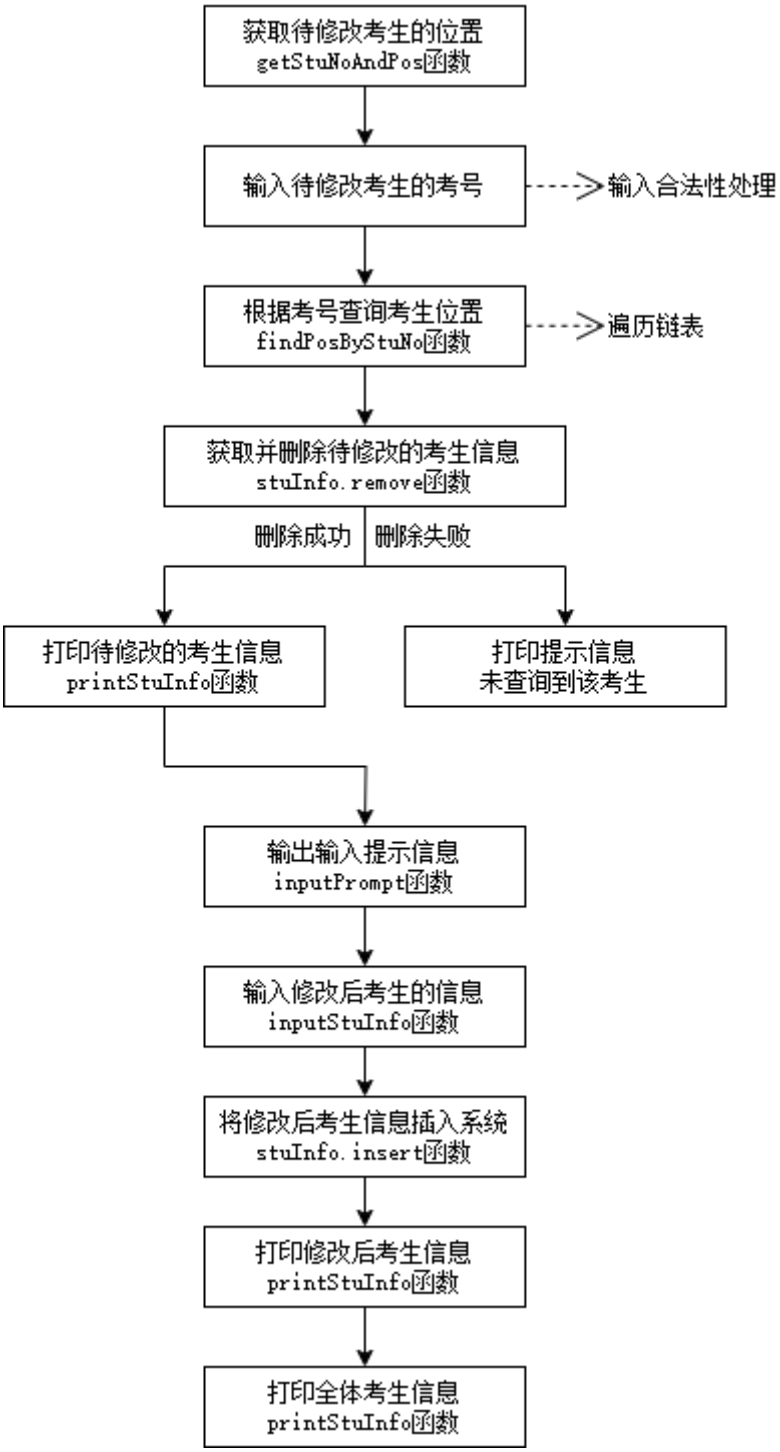

图 3.7.1.1 修改考生功能实现流程图

修改考生功能的函数名为 modifyFunction，修改考生功能实现的思路为：

(1)调用 getStuNoAndPos 函数获取用户待修改的考生的位置。这个位置是通过输入考生的考号来确定的，在 getStuNoAndPos 函数中首先输入待修改考

生的考号并进行输入合法性处理，然后调用 findPosByStuNo 函数遍历链表，根据考号查询考生位置；

⑵创建一个 StuInfo 结构体对象 modifyStu 用于存储待修改的考生信息；

⑶调用 stuInfo.remove 函数，将待修改的考生信息删除，并将其存储在 modifyStu 中。如果删除成功，该函数会返回 true，否则返回 false；

⑷如果删除成功（即返回 true），则调用 printStuInfo 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；

⑸如果删除失败（即返回 false），则显示一条消息表明未查询到要查询的考生，因为指定的考生考号在系统中不存在，并退出本函数；

⑹调用 inputPrompt 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；

⑺调用 inputStuInfo 函数，获取用户输入的考生信息，并将其存储在 modifyStu 中；

⑻调用 stuInfo.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；

⑼调用 printStuInfo 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。

3.7.2 修改考生功能核心代码

```
void modifyFunction(MyList<StuInfo>& stuInfo)
{
    /* Delete the information of the student before modification */
    int position = getStuNoAndPos(stuInfo, "待修改考生的考号");
    StuInfo modifyStu;
    if (stuInfo.remove(position, modifyStu))
        printStuInfo(modifyStu);
    else {
        std::cout << std::endl << ">>> 未查询到该考生" << std::endl;
        return;
    }

    /* Add the information of the student after modification */
    inputPrompt("请录入待修改考生信息:");
    modifyStu = inputStuInfo(stuInfo);
    stuInfo.insert(position - 1, modifyStu);
    printStuInfo(modifyStu);
    printStuInfo(stuInfo);
}
```

3.7.3 修改考生功能示例

```
>>> 菜单: [1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统
请选择操作类型: [4]
请输入待修改考生的考号 [不超过 12 个数字字符组成的字符串，超出部分将被截断]: 2250007
>>> 未查询到该考生
>>> 菜单: [1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统
请选择操作类型: [4]
请输入待修改考生的考号 [不超过 12 个数字字符组成的字符串，超出部分将被截断]: 2250004
>>> 考生信息
+--------------+--------------------------+------+------+--------------------------------+
| 考号         | 姓名                     | 性别 | 年龄 | 报考类别                       |
+--------------+--------------------------+------+------+--------------------------------+
| 2250004      | Amy                      | 女   | 20   | 软件设计师                     |
+--------------+--------------------------+------+------+--------------------------------+

>>> 请录入待修改考生信息:
    [输入格式] 考号 姓名 性别 年龄 报考类别 (用空格分隔数据)
    [考    号] 不超过 12 个数字字符组成的字符串，超出部分将被截断
    [姓    名] 不超过 24 个英文字符或 12 个汉字字符组成的字符串，超出部分将被截断
    [性    别] 男 / 女
    [年    龄] 在 1 至 99 范围内的整型数据
    [报考类别] 不超过 32 个英文字符或 16 个汉字字符组成的字符串，超出部分将被截断
2250004 Alice 女 25 软件开发师
>>> 考生信息
+--------------+--------------------------+------+------+--------------------------------+
| 考号         | 姓名                     | 性别 | 年龄 | 报考类别                       |
+--------------+--------------------------+------+------+--------------------------------+
| 2250004      | Alice                    | 女   | 25   | 软件开发师                     |
+--------------+--------------------------+------+------+--------------------------------+

>>> 全体考生信息 (考生人数: 6)
+--------------+--------------------------+------+------+--------------------------------+
| 考号         | 姓名                     | 性别 | 年龄 | 报考类别                       |
+--------------+--------------------------+------+------+--------------------------------+
| 2250001      | 张三                     | 男   | 18   | 软件设计师                     |
| 2250002      | 李四                     | 男   | 19   | 软件架构师                     |
| 2251001      | 李华                     | 男   | 25   | 软件架构师                     |
| 2250004      | Alice                    | 女   | 25   | 软件开发师                     |
| 2250005      | Mike                     | 男   | 18   | 软件开发师                     |
| 2250006      | Tom                      | 男   | 19   | 软件测试师                     |
+--------------+--------------------------+------+------+--------------------------------+
```

图 3.7.3.1 修改考生功能示例

3.8 统计考生功能的实现

3.8.1 统计考生功能实现思路

统计考生功能的函数名为 `statisticsFunction`，统计考生功能实现的思路为：

(1) 调用 `stuInfo.getLength` 函数获取考生信息系统中的总学生数量 `stuNum`；

(2) 创建变量 `maleCount` 用于统计男性学生的数量，以及一个数组 `ageCount` 用于统计各个年龄的学生数量；

(3)创建一个整型数组 traversedFlag，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；

(4)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；

(5)释放动态分配的内存，即释放 traversedFlag 数组的内存；

(6)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 statisticsGender 函数，打印性别统计信息；

(7)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 statisticsAge 函数，打印年龄统计信息。

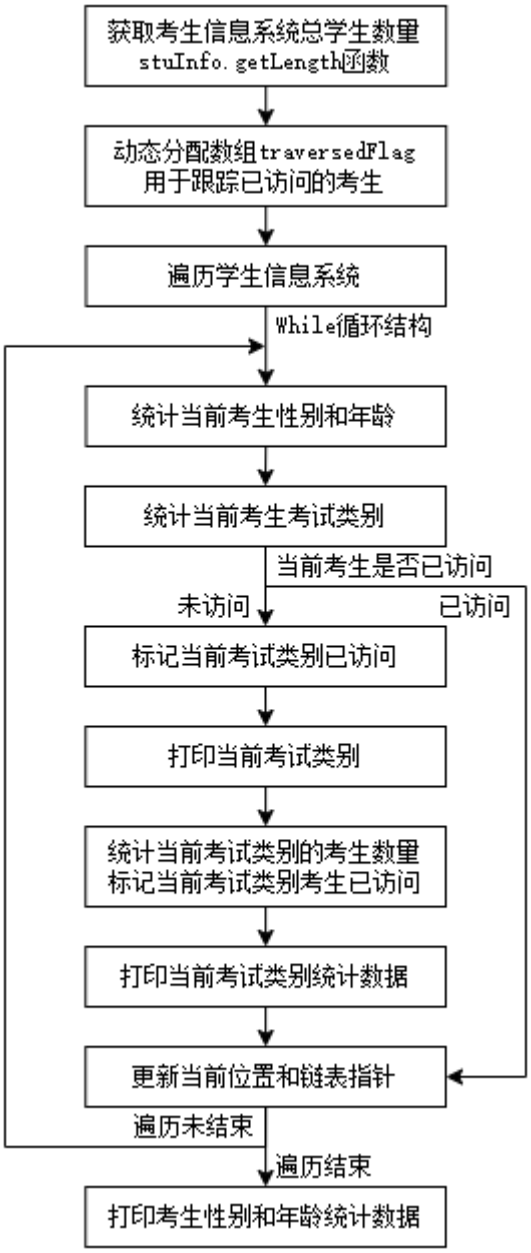

图 3.8.1.1 统计考生功能实现流程图

3.8.2 统计考生功能核心代码

```
    void statisticsFunction(MyList<StuInfo>& stuInfo)
    {
        /* Initialize the number of students and counter for student gender
and age*/
        int stuNum = stuInfo.getLength(), maleCount = 0, ageCount[100] =
{ 0 };

        /* Array to keep track of visited nodes */
        int* traversedFlag = new(std::nothrow) int[stuNum + 2];
        if (traversedFlag == NULL) {
            std::cerr << "Error: Memory allocation failed." << std::endl;
            exit(MEMORY_ALLOCATION_ERROR);
        }
        for (int count = 0; count < stuNum + 2; count++)
            traversedFlag[count] = UNTRAVELED;

        /* Traverse the student info */
        printCategoryChartHeader();
        MyLinkNode<StuInfo>* exterCurr = stuInfo.getHead()->link;
        int exterPos = 1;
        while (exterCurr != NULL) {
            if (exterCurr->data.sex)
                maleCount++;
            ageCount[exterCurr->data.age]++;
            if (traversedFlag[exterPos - 1] == UNTRAVELED) {
                traversedFlag[exterPos - 1] = TRAVELED;
                std::cout << std::setiosflags(std::ios::left) << "| " <<
std::setw(32) << exterCurr->data.category << " | ";
                int cateCount = 1, interPos = exterPos + 1;
                MyLinkNode<StuInfo>* interCurr = exterCurr->link;
                while (interCurr != NULL) {
                    if                  (!strcmp(exterCurr->data.category,
interCurr->data.category)) {
                        cateCount++;
                        traversedFlag[interPos - 1] = TRAVELED;
                    }
                    interCurr = interCurr->link;
                    interPos++;
                }
                std::cout  <<  std::setw(10)  <<  cateCount  <<  "  |  "  <<
std::setw(10) << (cateCount * 100.0 / stuNum) << " |" << std::endl <<
std::resetiosflags(std::ios::left);
            }
```

```
            exterCurr = exterCurr->link;
            exterPos++;
        }
        std::cout << "+---------------------------------+-----------+-
-----------+" << std::endl;

        /* Free dynamically allocated memory */
        delete[] traversedFlag;

        /* Count student gender and age */
        statisticsGender(stuNum, maleCount);
        statisticsAge(stuNum, ageCount);
    }
```

3.8.3 统计考生功能示例

```
>>> 全体考生信息 (考生人数: 6)
+-------------+-------------------------+------+------+---------------------------------+
| 考号        | 姓名                    | 性别 | 年龄 | 报考类别                        |
+-------------+-------------------------+------+------+---------------------------------+
| 2250001     | 张三                    | 男   | 18   | 软件设计师                      |
| 2250002     | 李四                    | 男   | 19   | 软件架构师                      |
| 2250003     | 王五                    | 女   | 20   | 软件开发师                      |
| 2250004     | Amy                     | 女   | 20   | 软件设计师                      |
| 2250005     | Mike                    | 男   | 18   | 软件开发师                      |
| 2250006     | Tom                     | 男   | 19   | 软件测试师                      |
+-------------+-------------------------+------+------+---------------------------------+

>>> 菜单: [1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统
请选择操作类型: [5]

>>> 考生报考类别统计
+---------------------------------+-----------+-----------+
| 报考类别                        | 人数      | 比例(%)   |
+---------------------------------+-----------+-----------+
| 软件设计师                      | 2         | 33.3333   |
| 软件架构师                      | 1         | 16.6667   |
| 软件开发师                      | 2         | 33.3333   |
| 软件测试师                      | 1         | 16.6667   |
+---------------------------------+-----------+-----------+

>>> 考生性别统计
+------+-----------+-----------+
| 性别 | 人数      | 比例(%)   |
+------+-----------+-----------+
| 男   | 4         | 66.6667   |
| 女   | 2         | 33.3333   |
+------+-----------+-----------+

>>> 考生年龄统计
+------+-----------+-----------+
| 年龄 | 人数      | 比例(%)   |
+------+-----------+-----------+
| 18   | 2         | 33.3333   |
| 19   | 2         | 33.3333   |
| 20   | 2         | 33.3333   |
+------+-----------+-----------+
```

图 3.8.3.1 统计考生功能示例

3.9 异常处理功能的实现

3.9.1 动态内存申请失败的异常处理

在进行 MyLinkNode 类等的动态内存申请时，程序使用 new(std::nothrow) 来尝试分配内存。new(std::nothrow)在分配内存失败时不会引发异常，而是返回一个空指针（NULL 或 nullptr），代码检查指针是否为空指针，如果为空指针，意味着内存分配失败，这时程序将执行以下操作：

(1) 向标准错误流 std::cerr 输出一条错误消息"Error: Memory allocation failed."，指出内存分配失败；

(2)调用 exit 函数，返回错误码 MEMORY_ALLOCATION_ERROR（通过宏定义方式定义为-1），用于指示内存分配错误，并导致程序退出。

下面是动态内存申请的异常处理的一个代码示例：

```
template <typename Type>
MyList<Type>::MyList()
{
    first = new(std::nothrow) MyLinkNode<Type>;
    if (first == NULL) {
        std::cerr << "Error: Memory allocation failed." << std::endl;
        exit(MEMORY_ALLOCATION_ERROR);
    }
    last = first;
}
```

3.9.2 MyList 类的异常处理

3.9.2.1 MyList 类索引越界的异常处理

在调用 MyList 类的 getData，setData，insert，remove 等成员函数时，程序会检查传入的索引参数 i 进行检查。若索引越界，函数返回 false，表示操作失败（索引无效）。

在调用 MyList 类的 locate 函数时，程序会检查传入的索引参数 i 进行检查。若索引越界，函数返回 NULL，表示操作失败（索引无效）。

3.9.2.2 MyList 类参数非法的异常处理

在调用 MyList 类的 splice 函数时，程序会对参数的合法性进行检查。如果要插入的位置为 NULL 或者 other 链表为空，则不会执行插入操作。

在调用 MyList 类的 merge 函数时，程序会对参数的合法性进行检查。如果当前链表与 other 链表是同一个链表(即自身合并自身)，则不会执行合并操作。

3.9.2.3 MyList 类自赋值的异常处理

MyList 类重载了赋值运算符，用于将一个链表赋值给另一个链表。自赋值是一种常见的错误操作，可能导致资源泄漏和其他问题，该运算符重载函数对自赋值（将链表赋值给自身）的情况进行了处理。

程序会检查 this 指针和传入的 L 链表对象是否相同，如果 this 指针与 L 相同，表示自赋值操作，为了防止自赋值，运算符重载函数不会执行赋值操作，而是直接返回当前对象的引用*this。这个错误处理机制确保了当尝试将链表赋值给自身时，不会导致资源泄漏或其他问题。

3.9.3 输入非法的异常处理

3.9.3.1 考生人数输入非法的异常处理

程序通过调用 inputInteger 函数输入考生人数。inputInteger 函数用于获取用户输入的整数，同时限制输入必须在指定的范围内，函数的代码如下：

```
int inputInteger(int lowerLimit, int upperLimit, const char* prompt)
{
    while (true) {
        std::cout << "请输入" << prompt << " [整数范围: " << lowerLimit
<< "~" << upperLimit << "]: ";
        double tempInput;
        std::cin >> tempInput;
        if (std::cin.good() && tempInput == static_cast<int>(tempInput)
&& tempInput >= lowerLimit && tempInput <= upperLimit) {
            std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
            return static_cast<int>(tempInput);
        }
        else {
            std::cerr << std::endl << ">>> " << prompt << "输入不合法，
请重新输入" << prompt << "! " << std::endl << std::endl;
            std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        }
    }
}
```

inputInteger 函数对输入非法的情况进行了处理，代码具体执行逻辑如下：

⑴进入一个无限循环，它会一直运行直到用户提供有效的输入；

⑵用户的输入被读取到 tempInput 变量中，这里采用 double 类型来接收输入以便后续检查；

(3)进行输入验证：std::cin.good()检查输入流的状态是否正常，确保没有发生数据类型输入错误，tempInput==static_cast<int>(tempInput)检查用户输入是否为整数，通过将其转换为整数再比较，tempInput>=lowerLimit 和 tempInput<=upperLimit 确保输入在指定的范围内；

(4)合法输入处理：如果用户提供了合法的输入，函数会清除输入流的错误状态，丢弃输入缓冲区中的任何剩余内容，然后返回转换后的整数值；

(5)非法输入处理：如果用户提供的输入不合法，函数会输出错误消息，清除输入流的错误状态，丢弃输入缓冲区中的内容，并继续循环以等待用户提供合法的输入。

3.9.3.2 考生信息输入非法的异常处理

为了更好地对考生信息系统的数据进行管理，并提高程序的适用性与扩展性，本程序对考生信息的输入格式和数据类型进行了规定：

(1)输入格式：考号 姓名 性别 年龄 报考类别(用空格分隔数据)；

(2)考号：不超过 12 个数字字符组成的字符串，超出部分将被截断；

(3)姓名：不超过 24 个英文字符或 12 个汉字字符组成的字符串，超出部分将被截断；

(4)性别：男/女；

(5)年龄：在 1 至 99 范围内的整型数据；

(6)报考类别：不超过 32 个英文字符或 16 个汉字字符组成的字符串，超出部分将被截断。

程序通过调用 inputStuInfo 函数输入考生信息。inputStuInfo 函数用于输入学生信息，并包含了多个步骤来验证和处理用户输入错误，代码具体执行逻辑如下：

(1)主循环：进入一个无限循环，用于不断尝试获取用户输入，直到用户提供有效的学生信息；

(2)创建临时学生信息对象：在每次循环迭代中，创建一个临时的 StuInfo 对象来存储用户输入的学生信息；

(3)输入学生考号并验证：用户被要求输入学生的考号，并调用 truncateString 函数将其截断为最多 12 个字符。如果输入不是纯数字（调用 isNumericString 函数用于验证），则输出错误消息并继续下一次迭代。并验证考号的唯一性：调用 findPosByStuNo 函数检查学生信息列表中是否已存在相同考号的学生，如果存在，输出错误消息并继续下一次迭代；

(4)输入学生姓名并验证：用户被要求输入学生的姓名，并调用 truncateString 函数将其截断为最多 24 个字符；

(5)输入学生性别并验证：用户被要求输入学生的性别，通过比较输入与“男”

和“女”来验证。如果输入不是这两者之一，则输出错误消息并继续下一次迭代；

(6)输入学生年龄并验证：用户被要求输入学生的年龄，验证输入是否是正整数，并在有效范围（1 到 99 之间）。如果输入不满足这些条件，则输出错误消息并继续下一次迭代；

(7)输入考试类别并验证：用户被要求输入学生的考试类别，并调用 truncateString 函数将其截断为最多 32 个字符；

(8)返回临时学生信息对象：如果用户成功提供了所有必要信息，并没有任何非法输入，函数将返回输入的学生信息对象，考生信息输入结束。

3.9.3.3 操作类型输入非法的异常处理

操作类型输入非法的异常处理通过如下代码实现：

```
int selectOptn(void)
{
    std::cout << std::endl << ">>> 菜单: [1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统" << std::endl;
    std::cout << std::endl << "请选择操作类型: ";
    char optn;
    while (true) {
        optn = _getch();
        if (optn == 0 || optn == -32)
            optn = _getch();
        else if (optn >= '0' && optn <= '5') {
            std::cout << "[" << optn << "]" << std::endl << std::endl;
            return optn - '0';
        }
    }
}
```

这段代码会一直等待用户输入，只有当用户输入有效的数字字符（'0'到'5'）时，才会返回该数字的整数值，表示用户选择的操作选项。如果用户输入无效字符，循环会继续等待用户提供有效的输入。这段代码的具体执行逻辑如下：

(1)显示提示信息，其中包含选项[1]到[5]，以及[0]退出系统的选项；

(2)进入一个无限循环，以等待用户输入；

(3)用户输入的字符会被_getch()函数获取。这个函数通常用于在控制台应用程序中获取单个字符而不显示在屏幕上。用户的输入存储在变量 optn 中；

(4)代码检查用户输入是否是数字字符（'0'到'5'）或特殊的控制字符。如果用户按下非数字字符或特殊控制字符，它将不执行下面的操作；

(5)如果用户输入是数字字符（'0'到'5'），它会在屏幕上显示用户输入的数字字符，并返回对应的整数值；

(6)如果用户输入的不是数字字符（'0'到'5'），代码将保持在循环中，继续

等待有效的输入，这确保了只有在用户输入正确的选项时才会退出循环。

4 项目测试

4.1 项目主体架构测试

4.1.1 输入考生人数功能测试

分别输入超过上下限的整数、浮点数、字符、字符串，可以验证程序对输入非法的情况进行了处理。

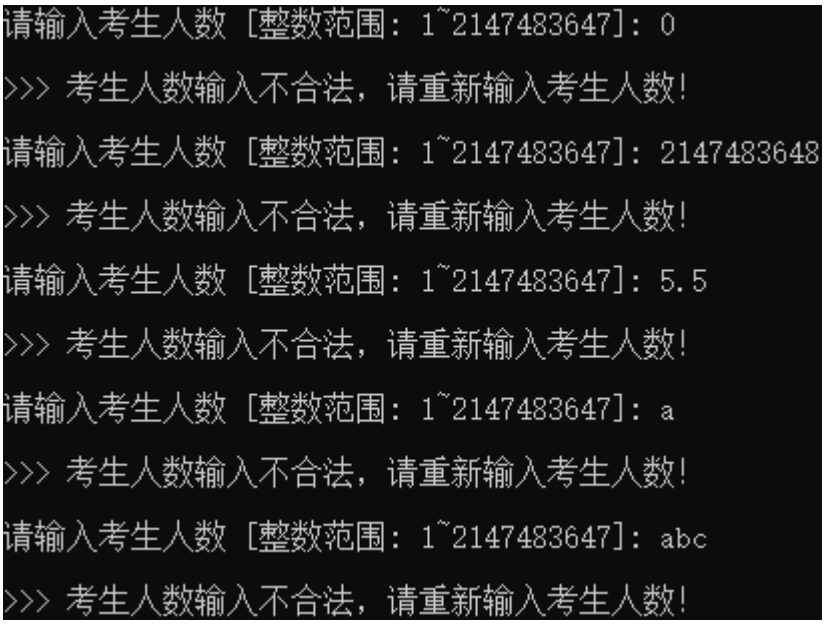

图 4.1.1.1 输入考生人数功能测试（输入非法）

当输入合法时，程序继续运行。

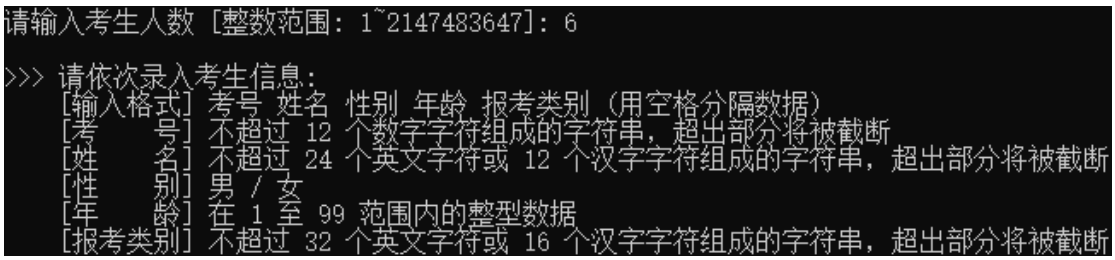

图 4.1.1.2 输入考生人数功能测试（输入合法）

4.1.2 录入考生信息功能测试

4.1.2.1 录入考生考号功能测试

程序对考生考号信息的输入格式和数据类型进行了规定：不超过 12 个数字字符组成的字符串，超出部分将被截断。若已存在考号相同的考生，则重新输入。

```
225000X 测试考生 男 20 软件开发师

>>> 考号输入不合法，请重新输入当前考生信息!

01234567890123456789 测试考生 男 20 软件开发师

>>> 考生信息系统建立完成（考生人数：1）

>>> 全体考生信息（考生人数：1）
+--------------+------------------------+------+------+--------------------------------+
| 考号         | 姓名                   | 性别 | 年龄 | 报考类别                       |
+--------------+------------------------+------+------+--------------------------------+
| 012345678901 | 测试考生               | 男   | 20   | 软件开发师                     |
+--------------+------------------------+------+------+--------------------------------+
```

图 4.1.2.1.1 录入考生考号功能测试（输入合法性）

```
2250001 测试考生1 男 20 软件开发师
2250001 测试考生2 女 18 软件开发师

>>> 已存在考号相同的考生，请重新输入当前考生信息!
```

图 4.1.2.1.2 录入考生考号功能测试（考号唯一性）

当输入合法时，程序继续运行（继续录入考生信息或操作考生信息系统）。

4.1.2.2 录入考生姓名功能测试

程序对考生姓名信息的输入格式和数据类型进行了规定：不超过 24 个英文字符或 12 个汉字字符组成的字符串，超出部分将被截断。

```
2250001 测试考生 男 20 软件开发师
2250002 TestCandidatesTestCandidates 女 18 软件开发师

>>> 考生信息系统建立完成（考生人数：2）

>>> 全体考生信息（考生人数：2）
+--------------+------------------------+------+------+--------------------------------+
| 考号         | 姓名                   | 性别 | 年龄 | 报考类别                       |
+--------------+------------------------+------+------+--------------------------------+
| 2250001      | 测试考生               | 男   | 20   | 软件开发师                     |
| 2250002      | TestCandidatesTestCandid | 女   | 18   | 软件开发师                     |
+--------------+------------------------+------+------+--------------------------------+
```

图 4.1.2.2.1 录入考生姓名功能测试

当输入合法时，程序继续运行（继续录入考生信息或操作考生信息系统）。

4.1.2.3 录入考生性别功能测试

程序对考生性别信息的输入格式和数据类型进行了规定：男/女。

```
2250001 测试考生 Female 20 软件开发师

>>> 性别输入不合法，请重新输入当前考生信息!

2250001 测试考生 女 20 软件开发师

>>> 考生信息系统建立完成（考生人数：1）

>>> 全体考生信息（考生人数：1）
+--------------+------------------------+------+------+--------------------------------+
| 考号         | 姓名                   | 性别 | 年龄 | 报考类别                       |
+--------------+------------------------+------+------+--------------------------------+
| 2250001      | 测试考生               | 女   | 20   | 软件开发师                     |
+--------------+------------------------+------+------+--------------------------------+
```

图 4.1.2.3.1 录入考生性别功能测试

当输入合法时，程序继续运行（继续录入考生信息或操作考生信息系统）。

4.1.2.4 录入考生年龄功能测试

程序对考生年龄信息的输入格式和数据类型进行了规定：在 1 至 99 范围内的整型数据。

```
2250001 测试考生 男 0 软件开发师
>>> 年龄输入不合法，请重新输入当前考生信息!
2250001 测试考生 男 100 软件开发师
>>> 年龄输入不合法，请重新输入当前考生信息!
2250001 测试考生 男 twenty 软件开发师
>>> 年龄输入不合法，请重新输入当前考生信息!
2250001 测试考生 男 20.5 软件开发师
>>> 年龄输入不合法，请重新输入当前考生信息!
2250001 测试考生 男 a 软件开发师
>>> 年龄输入不合法，请重新输入当前考生信息!
2250001 测试考生 男 20 软件开发师
>>> 考生信息系统建立完成（考生人数：1）
>>> 全体考生信息（考生人数：1）
+-------------+------------------------+------+------+--------------------------------+
| 考号        | 姓名                   | 性别 | 年龄 | 报考类别                       |
+-------------+------------------------+------+------+--------------------------------+
| 2250001     | 测试考生               | 男   | 20   | 软件开发师                     |
+-------------+------------------------+------+------+--------------------------------+
```

图 4.1.2.4.1 录入考生年龄功能测试

当输入合法时，程序继续运行（继续录入考生信息或操作考生信息系统）。

4.1.2.5 录入考生报考类别功能测试

程序对考生报考类别信息的输入格式和数据类型进行了规定：不超过 32 个英文字符或 16 个汉字字符组成的字符串，超出部分将被截断。

```
2250001 测试考生1 男 20 软件开发师
2250002 测试考生2 女 18 软件开发师、软件架构师、软件设计师、软件测试师
>>> 考生信息系统建立完成（考生人数：2）
>>> 全体考生信息（考生人数：2）
+-------------+------------------------+------+------+--------------------------------+
| 考号        | 姓名                   | 性别 | 年龄 | 报考类别                       |
+-------------+------------------------+------+------+--------------------------------+
| 2250001     | 测试考生1              | 男   | 20   | 软件开发师                     |
| 2250002     | 测试考生2              | 女   | 18   | 软件开发师、软件架构师、软件设计 |
+-------------+------------------------+------+------+--------------------------------+
```

图 4.1.2.5.1 录入考生报考类别功能测试

当输入合法时，程序继续运行（继续录入考生信息或操作考生信息系统）。

4.1.3 输出考生信息功能测试

输出考生信息功能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 StuInfo& stuInfo，使用 std::cout 输出参数 stuInfo 中的考生信息，若参数为 MyList<StuInfo>& stuInfo，则检查考生信息列表是否为空，若考生信息列表不为空，调用 stuInfo.output 函数输出全体考生信息否则输出一行信息表示无考生信息。

下面的测试对上述三种情况的输出考生信息功能进行了测试。

```
2250001 测试考生1 男 20 软件开发师
2250002 测试考生2 女 18 软件设计师

>>> 考生信息系统建立完成（考生人数：2）

>>> 全体考生信息（考生人数：2）
+--------------+--------------------------+------+------+--------------------------------+
| 考号         | 姓名                     | 性别 | 年龄 | 报考类别                       |
+--------------+--------------------------+------+------+--------------------------------+
| 2250001      | 测试考生1                | 男   | 20   | 软件开发师                     |
| 2250002      | 测试考生2                | 女   | 18   | 软件设计师                     |
+--------------+--------------------------+------+------+--------------------------------+

>>> 菜单：[1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统

请选择操作类型：[2]

请输入待删除考生的考号（不超过 12 个数字字符组成的字符串，超出部分将被截断）：2250001

>>> 考生信息
+--------------+--------------------------+------+------+--------------------------------+
| 考号         | 姓名                     | 性别 | 年龄 | 报考类别                       |
+--------------+--------------------------+------+------+--------------------------------+
| 2250001      | 测试考生1                | 男   | 20   | 软件开发师                     |
+--------------+--------------------------+------+------+--------------------------------+

>>> 全体考生信息（考生人数：1）
+--------------+--------------------------+------+------+--------------------------------+
| 考号         | 姓名                     | 性别 | 年龄 | 报考类别                       |
+--------------+--------------------------+------+------+--------------------------------+
| 2250002      | 测试考生2                | 女   | 18   | 软件设计师                     |
+--------------+--------------------------+------+------+--------------------------------+

>>> 菜单：[1]插入功能 [2]删除功能 [3]查询功能 [4]修改功能 [5]统计功能 [0]退出系统

请选择操作类型：[2]

请输入待删除考生的考号（不超过 12 个数字字符组成的字符串，超出部分将被截断）：2250002

>>> 考生信息
+--------------+--------------------------+------+------+--------------------------------+
| 考号         | 姓名                     | 性别 | 年龄 | 报考类别                       |
+--------------+--------------------------+------+------+--------------------------------+
| 2250002      | 测试考生2                | 女   | 18   | 软件设计师                     |
+--------------+--------------------------+------+------+--------------------------------+

>>> 全体考生信息（考生人数：0）
+--------------+--------------------------+------+------+--------------------------------+
| 考号         | 姓名                     | 性别 | 年龄 | 报考类别                       |
+--------------+--------------------------+------+------+--------------------------------+
| 无考生信息   |                          |      |      |                                |
+--------------+--------------------------+------+------+--------------------------------+
```

图 4.1.3.1 输出考生信息功能测试

4.2 插入考生功能测试

分别输入超过上下限的整数、浮点数、字符、字符串，可以验证程序对输入非法的情况进行了处理。

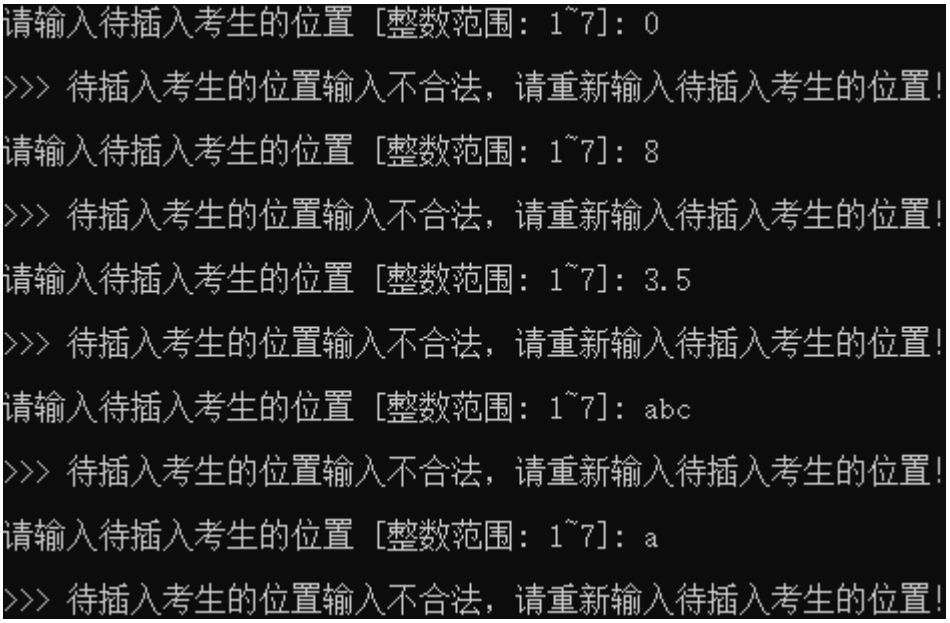

图 4.2.1 插入考生功能测试（输入合法性）

若已存在考号相同的考生，则重新输入。

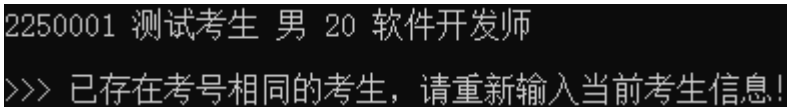

图 4.2.2 插入考生功能测试（考号唯一性）

当输入合法时，程序继续运行（见图 3.4.3.1），打印插入的考生信息和更新后的全体考生信息。

4.3 删除考生功能测试

程序对考生考号信息的输入格式和数据类型进行了规定：不超过 12 个数字字符组成的字符串，超出部分将被截断。

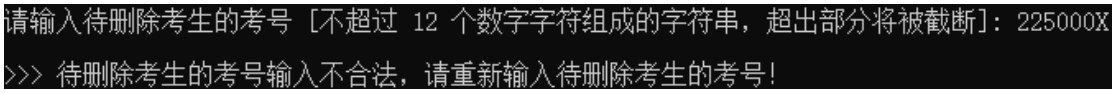

图 4.3.1 删除考生功能测试

当输入合法时，程序继续运行（见图 3.5.3.1），打印删除的考生信息和更新后的全体考生信息。

4.4 查询考生功能测试

程序对考生考号信息的输入格式和数据类型进行了规定：不超过 12 个数字字符组成的字符串，超出部分将被截断。

图 4.4.1 查询考生功能测试

当输入合法时，程序继续运行（见图 3.6.3.1），打印查询到的考生信息。

4.5 修改考生功能测试

程序对考生考号信息的输入格式和数据类型进行了规定：不超过 12 个数字字符组成的字符串，超出部分将被截断。

图 4.5.1 修改考生功能测试

当输入合法时，程序继续运行（见图 3.7.3.1），录入待修改考生信息，若输入非法则重新输入，若输入合法则打印修改后的考生信息和更新后的全体考生信息。

4.6 统计考生功能测试

统计考生功能函数统计考生报考类别、性别和年龄数据，并打印其统计信息（人数和所占比例）（见图 3.8.3.1）。

4.7 退出考试报名系统功能测试

在操作考生信息系统时，选择选项[0]退出考试报名系统。

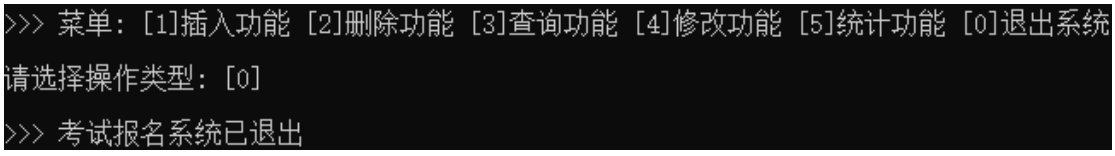

图 4.7.1 退出考试报名系统功能测试

5 集成开发环境与编译运行环境

Windows 系统：Windows 11 x64

Windows 集成开发环境：Microsoft Visual Studio 2022 (Release 模式)

Windows 编译运行环境：本项目适用于 x86 架构和 x64 架构

Linux 系统：CentOS 7 x64

Linux 编译命令：

g++ '/root/ 桌 面 /Share_Folder/exam_registration_system.cpp' -std=c++11 -o '/root/ 桌 面 /Share_Folder/exam_registration_system' -lncurses

Linux 运行命令：

'/root/桌面/Share_Folder/exam_registration_system'

图 5.1 Linux 环境程序运行示例

本项目使用条件编译解决 Windows 系统和 Linux 系统编译环境的差异，示例代码如下。

```
#ifdef _WIN32
#include <conio.h>
#elif __linux__
#include <ncurses.h>
#endif
```