同济大学
TONGJI UNIVERSITY

数据结构课程设计

项目说明文档

考试报名系统

姓　　名：　　马小龙

学　　号：　　2353814

学　　院：计算机科学与技术学院（软件学院）

专　　业：　　软件工程

指导教师：　　张颖

二〇二三年十月二十七日

目录

第 1 章 项目分析

1.1 项目背景分析

考试报名系统在高校管理中扮演着关键角色，对学校的管理者和学生均具有重要意义。一个高效的考试报名系统应能提供全面的信息和功能，以便更好地满足用户需求。随着学生数量和考试次数的迅速增长，传统的手工管理方式面临诸多挑战，既繁琐又容易出错，这给教务管理部门带来了巨大的压力。借助计算机技术，开发考试报名系统能够显著改善这一状况。计算机化管理不仅提高了数据处理的速度和准确性，还使得信息存储和查询更加便捷。这种转变将极大地减轻教务人员的工作负担，同时为学生提供更流畅的报名体验。

在信息化时代，建立一个全面的考试报名系统不仅是提高管理效率的必要手段，更是学校实现现代化的重要步骤。通过系统的应用，高校能够有效整合资源、优化流程，进而提升整体的教学管理水平。因此，开发此类系统具有深远的意义，为学校的可持续发展奠定了基础。

1.2 项目需求分析

功能需求方面系统允许教务人员输入考生的基本信息进行录入，输入考生信息应为常见格式，查询考生信息，修改已录入的信息，以及删除不再有效的考生信息。此外，为了更好的完成统计工作，本项目应当还要支持考生相关信息的人数统计，例如性别、年龄等信息的统计。

技术需求包括支持 Windows 和 Linux 等操作系统，采用 C++等主流编程语言，采用合适的数据结构，同时，由于考生人数不定，不能对人数做出相关限制。界面方面，初期为控制台界面，后期可考虑开发图形用户界面（GUI）。

1.3 项目功能分析

本项目旨在开发一套高效的考试报名系统，以解决高校在考试管理中面临的各种挑战。系统支持考生信息的录入、查询、修改、删除、统计等基本功能，从而实现对考生信息的高效管理，提高报名工作的效率和准确性。以下是项目主要功能的详细信息。

1.3.1 考生信息录入功能

允许教务人员根据输入的考生人数输入一定数目的考生基本信息，包括考号、姓名、性别、年龄、报考类别等，已建立考生信息系统。输入时，程序会验证考生的基本信息是否符合规范，如不规范，需要重新输入。具体规范要求如下：

1.考生考号应为正整数，且不能与已经录入系统的考号重复；

2.考生性别性别只能为“男”或“女”，不能输入其他多余字符；

3.考生年龄应该为非负整数，在初始程序中，通过全局 constexpr 设定最大年龄为 150，最小为 0；可以根据实际情况进行更改；

4.未对考生姓名、报考类别做出相关限制。

1.3.2 考生信息输出功能

能够按照固定格式对已录入的考生的基本信息进行输出，包括考号、姓名、性别、年龄、报考类别等。

1.3.3 考生插入功能

允许教务人员在已有考生信息系统上在需要的位置添加新的考生信息。

1.3.4 考生查询功能

允许教务人员在已有考生信息系统上根据考生的考号来查询考生，若查询到则输出该考生信息，否则说明考生不存在。

1.3.5 考生删除功能

允许教务人员在已有考生信息系统上根据考生的考号来查询是否存在该考生，若存在对考生进行删除工作。

1.3.6 考生修改功能

允许教务人员在已有考生信息系统上根据考生的考号来查询是否存在该考生，若存在需要输入新的考生信息，替换旧的考生信息。

1.3.7 考生统计功能

允许教务人员统计考生信息系统上的考生信息，如总人数、考生报考类别比例、考生年龄比例、考生性别比例。

1.3.8 异常处理功能

程序对各种异常进行了基本处理，以提升系统稳定性。

第 2 章 项目设计

2.1 数据结构设计

本项目设计了两种数据结构，分别是链表和 String（string 类型自己实现）。

（一）使用链表作为主要数据结构，用来储存整个考生信息系统的数据，主要是基于以下方面：

1.链表的动态大小需求使其在处理考生信息时更加灵活。与数组不同，链表能够动态分配内存，以适应不断变化的考生数量。这避免了数组在预先设定大小时可能造成的内存浪费或不足的问题。当考生人数增加或减少时，链表可以迅速调整，确保内存资源得到有效利用，从而提高了系统的灵活性。

2.链表在插入和删除操作上的高效性使其更具优势。由于链表只需调整结点指针，而不需要移动整个元素，插入和删除操作的时间复杂度较低，这对于频繁更新考生报名信息的场景尤为重要。在实际操作中，考生添加报名或撤回报名时，链表能够快速执行这些操作，显著提高系统的响应速度和用户体验。

3.考生信息的频繁修改也是选择链表的重要原因。对于需要经常修改或取消报名的情况，链表的设计使得结点的指针调整比数组元素的修改更加高效。这种特性不仅提升了系统的响应速度，也简化了考生信息的管理流程，使教务人员能够更加高效地处理日常事务。

4.链表提供了良好的内存利用率，避免了使用固定大小数组时可能出现的空间浪费或不足。对于动态变化的考生信息，链表的灵活性可以更好地适应实际需求，确保在不同情况下都能高效存储数据。此外，链表的结构使得在内存使用上更加合理，有助于提升整体性能。

5.从系统维护和扩展的复杂度考虑，链表提供了更好的可扩展性和易于修改的特性。在未来需要进行功能扩展时，链表结构的灵活性将大大简化设计和实现过程。例如，未来如果需要添加额外的考生信息字段，链表的结构允许快速调整而无需重构整个数据结构，从而减少开发时间和维护成本。

（二）使用 String 来储存考生基本信息中的姓名、性别以及报考类别，主要是基于一下方面：

1.string 类型具有良好的可读性和易用性，使得处理和展示考生的姓名、性别等信息变得直观。在代码中，string 的使用能够增强可维护性，方便后续的调试和修改。

2. string 类型的动态特性使得其在处理长度不确定的文本时非常灵活。考生的姓名和其他信息的长度可能各异，string 能够自动管理内存，适应不同长度的内容，避免了固定大小数组带来的限制。

3. string 提供了丰富的内置方法和操作，如字符串连接、查找、替换和分割等。这些功能使得在处理考生信息时可以轻松进行数据验证和格式化，提高了系统的效率。例如，在进行报名信息的录入和查询时，可以方便地对输入进行格式检查，确保数据的有效性。

4. string 也能够方便地与其他数据结构进行结合。与链表的结合使用，可以高效地管理考生信息链表中的每个结点，确保信息的动态性和易操作性。

2.2 结构体与类设计

为了使链表更加通用，本链表设计为链表模板设计，为链表添加了许多实用性功能。

2.2.1 链表结点（Node）设计

2.2.1.1 概述

Node 用于储存信息，包括结点储存的具体信息内容和下一结点的地址。

2.2.1.2 结构体定义

```
struct Node {
    T data;
    Node<T>* next;
    operator T();
    operator T& ();
    operator const T& () const;
    Node<T>& operator=(const Node<T>& node);
    bool operator==(const Node<T>& node) const;
    Node(Node<T>* ptr = nullptr);
    Node(const T& item, Node<T>* ptr = nullptr);
};
```

2.2.1.3 数据成员

T data: 数据域，存储结点的数据

Node<T>* next: 指针域，指向下一个结点的指针

2.2.2.4 构造函数

Node(Node<T>* ptr = nullptr);

构造函数，初始化指针域。

Node(const T& item, Node<T>* ptr = nullptr);

构造函数，初始化数据域和指针域。

2.2.1.5 成员函数

operator T();

允许 Node 对象被视为其数据类型的一个副本。

operator T& ();

允许 Node 对象被视为其数据类型的一个引用。

operator const T& () const;

提供了对数据的常量引用访问，确保在只读上下文中使用。

Node<T>& operator=(const Node<T>& node);

=运算符重载,允许一个 Node 对象通过赋值操作符=从另一个 Node 对象复制数据。

bool operator==(const Node<T>& node) const;

==运算符重载，用于比较两个 Node 对象是否相等;

2.2.2 链表（LinkList）设计

2.2.2.1 概述

该通用模板类 LinkList 用于表示单链表。此链表头结点只做定位用途，不储存数据，头结点的下一结点为数据储存的起点。链表结点由 Node 结构体表示，其中包含数据 和指向下一个结点的指针。该链表提供了一系列基本操作函数，包括结点的插入、 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。

2.2.2.2 Linklist 类定义

```
template <typename T>
class LinkList {
public:
    class iterator;
     iterator begin();
     iterator end();
     LinkList();
```

```
        LinkList(const T& x);
        LinkList(const LinkList<T>& L);
        ~LinkList();
        void makeEmpty();
        int Length();
        Node<T>* getHead();
        Node<T>* Search(T x);
        Node<T>* Locate(int i);
        bool getData(int i, T& x);
        bool setData(int i, const T& x);
        bool Insert(int i,const T& x);
        bool Remove(int i, T& x);
        bool IsEmpty() ;
        static bool IsFull() ;
        void inputFront(const T& x);
        void inputRear(const T& x);
        void output();
        LinkList<T>& operator=(const LinkList<T>& other);
        T& operator[](int index);
    private:
        Node<T>* head;
    };
```

2.2.2.3 私有数据成员

Node<T>* head: 指向链表头结点的指针

2.2.2.4 构造函数与析构函数

LinkList();

默认构造函数，创建一个空链表。

LinkList(const T& x);

转换构造函数，创建一个包含头结点和一个数据结点的链表。

LinkList(const LinkList<T>& L);

复制构造函数，通过复制另一个链表创建新链表。

~LinkList();

析构函数，释放链表的内存资源，包括所有结点的内存。

2.2.2.5 公有成员函数及数据类型

class iterator；

迭代器类定义。

iterator begin();

迭代器初始结点，指向链表头结点后一个结点

iterator end();

迭代器末结点，链表尾后一个结点，及 nullptr。

void makeEmpty();

清空链表，释放除头结点所有结点的内存。

int Length();

获取链表中结点的个数。

Node<T>* getHead();

获取链表头结点的指针。

Node<T>* Search(T x);

搜索链表中值为 x 的结点，返回该结点的指针，若不存在返回 nullptr。

Node<T>* Locate(int i);

返回链表中第 i 个结点的指针，若 i 超出链表长度或小于 0，则返回 nullptr

bool getData(int i, T& x);

获取链表中第 i 个结点的数据，并通过引用返回。返回值为操作是否成功

bool setData(int i, const T& x);

设置链表中第 i 个结点的数据。返回值为操作是否成功

bool Insert(int i,const T& x);

在链表中第 i-1 个结点后插入新结点，成为第 i 的结点（头结点记为第零个结点）。返回值为操作是否成功。

bool Remove(int i, T& x);

删除链表中第 i 个结点，并通过引用返回其数据。返回值为操作是否成功。

bool IsEmpty() ;

判断链表是否为空（只有头结点即为空）

void inputFront(const T& x);

在链表的开头（head 之后）插入一个新元素

void inputRear(const T& x);

在链表的尾部插入一个新元素。

void output();

输出链表中所有结点的数据。

LinkList<T>& operator=(const LinkList<T>& other);

赋值运算符重载，将一个链表的所有值赋给另外一个，形成两个相同的链表

T& operator[](int index);

下标运算符重载，返回第 index 结点的数据

2.2.3 迭代器设计

2.2.3.1 概述

由于需要验证学号是否重复，需要访问链表中的所有结点，为了实现这一过程，有三种方法：1.采用结点地址→data→number 的方式访问，通过结点地址→next 转到下一结点，这样太过于麻烦，且模块化不够好；2.采用下标 []运算，但每一次都要从头访问，时间复杂度较高。最好的方式是采用迭代器。

2.2.3.2 迭代器类定义

```
class iterator {
private:
    Node<T>* current;
public:
    iterator(Node<T>* ptr = nullptr);
    iterator& operator++();
    T& operator*();
    bool operator==(const iterator& other) const;
    bool operator!=(const iterator& other) const ;
};
```

2.2.3.3 私有数据成员

Node<T>* head: 指向当前结点的指针

2.2.3.4 构造函数

iterator(Node<T>* ptr = nullptr)；

构造函数，设定当前结点地址。

2.2.3.5 公有成员函数

iterator& operator++();

重载前缀递增运算符。

T& operator*();

重载解引用运算符，返回结点对象。

bool operator==(const iterator& other) const;

重载等于运算符，判断结点对象是否相等。

bool operator!=(const iterator& other) const;

重载不等于运算符，判断结点对象是否不相等。

2.2.4 String 类设计

2.2.4.1 概述

String 为自定义数据类型，可以用来储存及处理字符串，例如姓名、性别和报考类别。

2.2.4.2 String 类定义

```
class String {
private:
    char* _str;
    size_t _str_len;
    size_t _str_cap;
public:
    static constexpr size_t npos = -1;
    typedef char* iterator;
    typedef const char* const_iterator;
    iterator begin();
    iterator end();
    const_iterator begin() const;
    const_iterator end() const;
    String(const char* str = "");
    String(const String& s);
   ~String();
    const char* C_str ()const;
    size_t Size()const;
    size_t Capacity()const;
    void Swap(String& s);
    void Reserve(size_t n);
    void Resize(size_t n, char ch = '\0');
```

```
    void PushBack(char ch);
    void Append(const char* str);
    void Insert(size_t pos, char ch, size_t n=1);
    void Insert(size_t pos, const char* str);
    void Erase(size_t pos, size_t len = npos);
    size_t Find(char ch, size_t pos = 0) const;
    size_t Find(const char* str, size_t pos = 0) const;
    String Substr(size_t pos = 0, size_t len = npos) const;
    void Clear();
    static int Memcmp(const void *dst,const void *src,size_t n) ;
    static int Strlen(const char* src);
    static char* Strstr(const char* str1, const char* str2);
    String& operator+=(char ch);
    String& operator+=(const char* str);
    bool operator<(const String& s) const;
    bool operator==(const String& s) const;
    bool operator<=(const String& s) const;
    bool operator>(const String& s) const;
    bool operator>=(const String& s) const;
    bool operator!=(const String& s) const;
    char& operator[](size_t pos);
    const char& operator[](size_t pos) const;
};
```

2.2.4.3 私有数据成员

char* _str: 存储字符串的字符数组

size_t _str_len: 当前长度

size_t _str_cap: 当前容量

2.2.4.4 公有数据成员

static constexpr size_t npos = -1: npos 常量表示无效位

2.2.4.5 构造函数与析构函数

String(const char* str = "");

构造函数，默认空字符串

String(const String& s);

拷贝构造函数

~String();

析构函数，释放字符串所占内存

2.2.4.6 公有成员函数

iterator begin();

获取迭代器开始位置。

iterator end();

获取迭代器结束位置。

const_iterator begin() const;

获取常量迭代器开始位置。

const_iterator end() const;

获取常量迭代器结束位置。

const char* C_str ()const;

获取当前字符串。

size_t Size()const;

获取字符串长度。

size_t Capacity()const;

获取当前容量。

void Swap(String& s);

交换字符串内容。

void Reserve(size_t n);

预留容量，扩展字符串容量。

void Resize(size_t n, char ch = '\0');

调整当前字符串大小。

void PushBack(char ch);

在字符串末尾添加字符。

void Append(const char* str);

在字符串末尾添加字符串。

void Insert(size_t pos, char ch, size_t n=1);

在指定位置插入 n 个字符。

void Insert(size_t pos, const char* str);

在指定位置插入字符串。

void Erase(size_t pos, size_t len = npos);

删除指定位置指定长度的字符串。

size_t Find(char ch, size_t pos = 0) const;

在字符串中查找指定字符位置。

size_t Find(const char* str, size_t pos = 0) const;

在字符串查找指定字符串位置。

String Substr(size_t pos = 0, size_t len = npos) const;

提取子字符串。

void Clear();

清空字符串。

static int Memcmp(const void *dst,const void *src,size_t n) ;

检查字符串长度是否相等，根据返回结果进行判断 1 为 dst 大于 src，-1 为 dst 小于 srcdst ，0 为等于 src。

static int Strlen(const char* src);

获取字符串长度。

static char* Strstr(const char* str1, const char* str2);

查找子字符串。

String& operator+=(char ch);

+=运算符重载，将新的字符放在字符串末尾。

String& operator+=(const char* str);

+=运算符重载，将新的字符串放在字符串末尾。

bool operator<(const String& s) const;

小于运算符重载，用于字符串大小比较。

bool operator==(const String& s) const;

等于运算符重载，用于字符串大小比较。

bool operator<=(const String& s) const;

小于等于运算符重载，用于字符串大小比较。

bool operator>(const String& s) const;

大于运算符重载，用于字符串大小比较。

bool operator>=(const String& s) const;

大于等于运算符重载，用于字符串大小比较。

bool operator!=(const String& s) const;

不等于运算符重载，用于字符串大小比较。

char& operator[](size_t pos);

下标运算符重载，用于查找指定位置字符。

const char& operator[](size_t pos) const;

下标运算符重载，用于查找指定位置字符。

String& operator=(const String& s);

赋值运算符重载，用于 String 类的赋值。

2.2.4.7 输入重载

inline std::istream& operator>>(std::istream& input, String& s)；

输入流操作符重载，规范 String 数据类型的输入

2.2.5 Examinee 结构体设计

2.2.5.1 概述

该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。

2.2.5.2 结构体定义

```
struct Examinee {
    int examination_number;
    mine::String name;
    mine::String sex;
    int age;
    mine::String entry_category;
};
```

2.2.5.3 数据成员

int examination_number: 考号

mine::String name: 姓名

mine::String sex: 性别

int age: 年龄

mine::String entry_category: 报考类别

2.2.6 Category 结构体设计

2.2.6.1 概述

该结构体用来储存报考类别及该报考类别的人数，以便利统计。

2.2.6.2 结构体定义

```
struct Category {
mine::String entry_category;
int number;
};
```

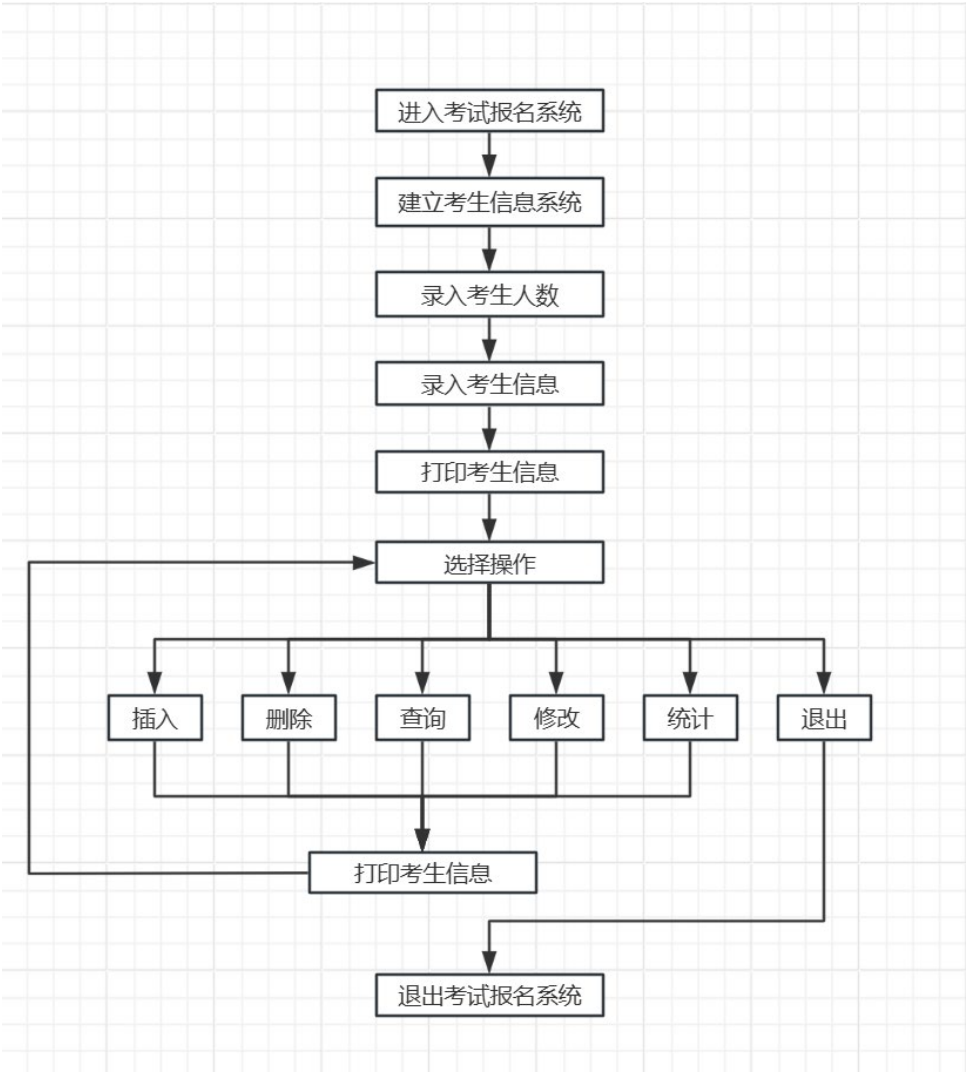
2.2.6.3 数据成员

mine::String entry_category: 储存报考类别

int number: 该报考类别的人数

2.3 项目框架设计

2.3.1 项目框架流程图



2.3.2 项目框架流程

1.进入考试报名系统。

2.输入考生人数和依次录入考生信息，以建立考生信息系统。

3. 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，输出所有考生信息，以实现所需的功能。

5.用户选择退出后，退出循环及考试报名系统。

第 3 章 项目功能实现

3.1 项目主体架构

3.1.1 实现思路

1.进入 ExaminationRegistrationSystem 函数以进入考试报名系统。

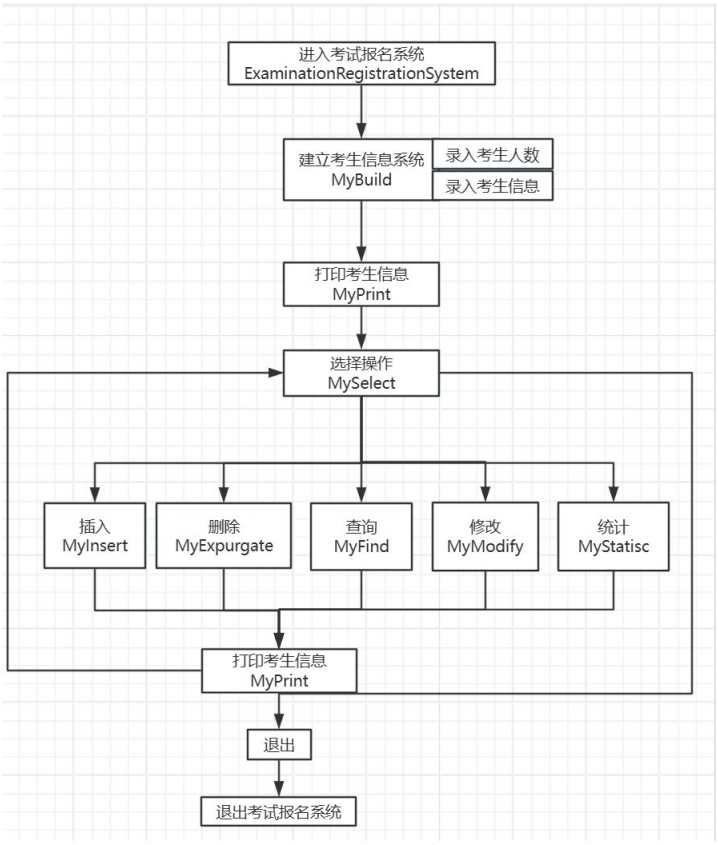
2.调用 MyBuild 函数开始建立考生信息系统，输入初始考生人数、考生基本信息，完成考生信息系统建立。

3.调用 MyPrint 函数打印全部考生信息。

4.进入循环，调用 MySelect 函数选择要进行地操作后，回到循环，根据选择的操作开始执行不同的函数，包括 MyInsert 插入、MyExpurgate 删除、MyFind 查找、MyModify 修改，MyStatisc 统计，之后再调用 MyPrint 函数打印全部考生信息，然后循环执行；直到选择退出则退出循环。

5.退出考试报名系统。

3.1.2 流程图



3.1.3 核心代码

```
    void ExaminationRegistrationSystem()
{
    std::cout << "+-----------------------------------------+\n";
    std::cout << "|                考生报名系统             |\n";
    std::cout << "|   Examination Registration System   |\n";
    std::cout << "+-----------------------------------------+\n";
    std::cout << "首先请建立考生信息系统！\n";
    LinkList<Examinee>
    MyBuild(examinee);
    MyPrint(examinee);
    while (true) {
        int ret = MySelect(examinee);
        if (ret == 1)
            MyInsert(examinee);
        else if (ret == 2)
            MyExpurgate(examinee);
        else if (ret == 3)
            MyFind(examinee);
        else if (ret == 4)
            MyModify(examinee);
        else if (ret == 5)
            MyStatisc(examinee);
        else {
            break;
        }
         if(ret!=0)
              MyPrint(examinee);
    }
    std::cout << "成功退出考试报名系统！\n";
}
```

3.1.4 示例

```
D:\Programme\DataStruct\examination_registration_system\cmake-build-debug\examination_registration_system.cpp
+---------------------------------------------+
|                 考生报名系统                |
|       Examination Registration System       |
+---------------------------------------------+
首先请建立考生信息系统！
请输入考生人数： 10
请依次输入考生的考号，姓名，性别，年龄以及报考类别！
2014199 张伟 男 21 软件开发师
2223540 李娜 女 22 软件测试师
2033638 王磊 男 20 软件运营师
2044268 刘洋 男 23 软件设计师
2213716 赵敏 女 22 软件开发师
2024286 陈佳 男 20 软件测试师
2131423 刘莉 女 21 软件运营师
2042141 孙强 男 23 软件设计师
2053086 朱莉 女 21 软件开发师
2236224 何俊 男 20 软件测试师


+-----------------------------------------------------------------------------------------+
|                                        考生信息系统                                     |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 考号            | 姓名            | 性别            | 年龄            | 报考类别        |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2014199         | 张伟            | 男              | 21              | 软件开发师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2223540         | 李娜            | 女              | 22              | 软件测试师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2033638         | 王磊            | 男              | 20              | 软件运营师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2044268         | 刘洋            | 男              | 23              | 软件设计师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2213716         | 赵敏            | 女              | 22              | 软件开发师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2024286         | 陈佳            | 男              | 20              | 软件测试师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2131423         | 刘莉            | 女              | 21              | 软件运营师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2042141         | 孙强            | 男              | 23              | 软件设计师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2053086         | 朱莉            | 女              | 21              | 软件开发师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 2236224         | 何俊            | 男              | 20              | 软件测试师      |
+-----------------+-----------------+-----------------+-----------------+-----------------+
考生总人数为：10

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）：0
成功退出考试报名系统！

进程已结束，退出代码为 0
```

3.2 考生信息录入功能

3.2.1 实现思路

考生信息录入功能的函数名为 MyBuild，考生信息录入功能实现的思路为：

1. 首先输入考生人数，输入不符合规范则需重新输入(Check 验证是否为非负整数，Switch 验证是否超 int 类型上限并通过引用返回 int 类型数据)；

2. 利用 for 循环，循环次数为考生人数；

3. 循环开始时，首先定义存储当前考生信息的结构体，然后调用 Input 函数进行信息输入；

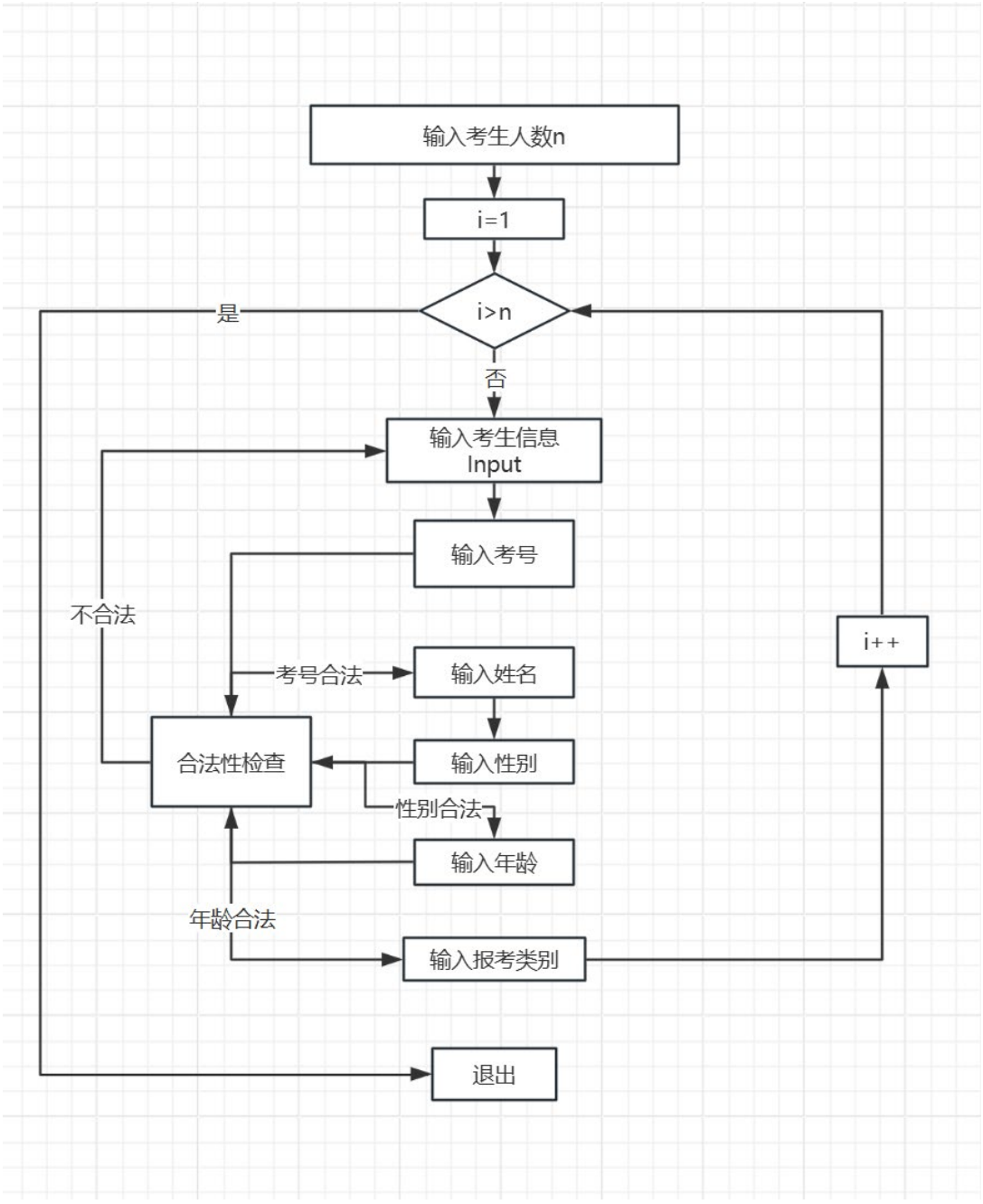
4.考生学生信息：输入学生的考号、姓名、性别、年龄、考试类别等信息，分别进行格式验证，若不符合规范则清除缓冲区，重新输入当前考生信息；

5. 当前考生信息输入成功后，将其链接到链表尾部，开始下一次循环；

6.所有考生信息输入完成后，退出循环，并退出 MyBuild 函数，成功建立考生信息系统。

注：考号不能重复，使用 IsSameExaminationNumber 检查；

3.2.2 流程图



3.2.3 核心代码

```
void MyBuild(LinkList<Examinee>& examinee)
    {
        int num_examinee;
        mine::String number_examinee;
        while (true) {
            std::cout << "请输入考生人数：  ";
            std::cin >> number_examinee;
            if (!Check(number_examinee)
|| !Switch(number_examinee,num_examinee)) {
                std::cout << "输入人数不合法，请重新输入！\n";
                ClearBuffer();
                continue;
            }
            break;
        }
        std::cout << "请依次输入考生的考号，姓名，性别，年龄以及报考类
别！\n";
        for (int i = 1; i <=num_examinee; i++) {
            Examinee student;
            student = Input(examinee);
            examinee.inputRear(student);
        }
    }
    Examinee Input(LinkList<Examinee>& examinee)
    {
        Examinee temp;
        while (true) {
            mine::String num_temp;
            std::cin >> num_temp;
            if (!Check(num_temp)
    || !Switch(num_temp,temp.examination_number)||temp.examination_number=
    =0) {
                std::cout << "考号输入不合法，请重新输入考生信息\n";
```

```
                ClearBuffer();
                continue;
            }
            if (IsSameExaminationNumber(temp.examination_number,
examinee)) {
                std::cout << "已存在考号相同的考生，请重新输入考生信息
\n";
                ClearBuffer();
                continue;
            }
            std::cin >> temp.name;
            std::cin >> temp.sex;
            if (temp.sex!="男" && temp.sex!= "女"){
                std::cout << "性别输入不合法，请重新输入考生信息\n";
                ClearBuffer();
                continue;
            }
            std::cin >> num_temp;
            if (!Check(num_temp) ||!Switch(num_temp,temp.age)|| temp.age <
kMinAge|| temp.age>kMaxAge
             ) {
                std::cout << "年龄输入不合法，请重新输入考生信息\n";
                ClearBuffer();
                continue;
            }
            std::cin >> temp.entry_category;
            break;
        }
        return temp;
}
int IsSameExaminationNumber(const int number, LinkList<Examinee>&
examinee)
{
        for (auto exa : examinee) {
```

```
            if (number == exa.examination_number)
                return 1;
        }
        return 0;
    }
```

3.2.4 示例

示例同 3.1.4

3.3 考生信息输出功能

3.3.1 实现思路

输出考生信息功能的函数名为 MyPrint，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；

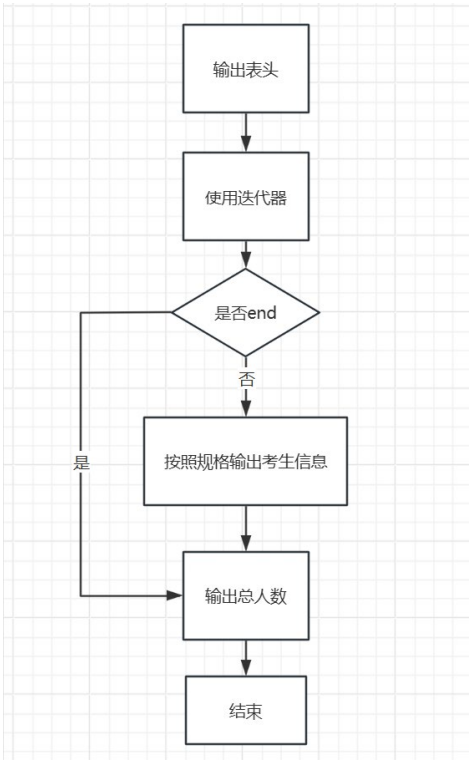
2.通过迭代器的方式调用 ExamineePrintTemplate 函数按照一定格式输出每个考生的基本信息；

3.输出考生总人数；

4. ExamineePrintTemplate 对每个考生的所有信息按照一定格式输出，使输出结果更加可视化；同时，函数应当应对不同编码的情况，尤其是编码为 UTF-8 的情况，汉字的显示长度与所占字节数不等；为了应对这一情况，调用 EncodingDeviation 函数补齐差距，使程序泛用性更高。

注：本程序使用的所有中文字符在 UTF-8 编码下为 3 字节，在 gbk 编码下为 2 字节

3.3.2 流程图



3.3.3 核心代码

```
void MyPrint(LinkList<Examinee>& examinee)
{
    std::cout << "\n\n";
    std::cout << "+------------------------------------------------------------------------
-------------------------+\n";
    std::cout << "|                                                                考生信
息系统                                                  |\n";
    std::cout << "+------------------+------------------+------------------+----------
--------+------------------+\n";
    std::cout << "| 考号               | 姓名               | 性别
| 年龄               | 报考类别           |\n";
    int n=0;
    for (auto exam_people: examinee){
    ExamineePrintTemplate(exam_people);
n++;
    }
    std::cout << "+------------------+------------------+------------------+----------
```

```
--------+------------------+\n";
            std::cout<<"考生总人数为："<<n<<"\n";
        }
        void ExamineePrintTemplate(const Examinee& student)
        {
            std::cout << "|------------------+------------------+------------------+------------
-------+------------------|\n";
            std::cout << std::setiosflags(std::ios::left)
           << "| " << std::setw(18) << student.examination_number
           << "| " << std::setw(18+EncodingDeviation(student.name))<<student.name
           << "| " << std::setw(18+EncodingDeviation(student.sex)) << student.sex
           << "| " << std::setw(18) << student.age
           << "| " << std::setw(18+EncodingDeviation(student.entry_category))<<
student.entry_category
           << std::resetiosflags(std::ios::left);
            std::cout << "|\n";
        }

        typedef unsigned long long size_t;
        constexpr int gbk_chinese_char_len=2;
        const mine::String demo="马";
        size_t EncodingDeviation(const mine::String& str) {
            size_t normal_chinese_char_len=demo.Size();
            if(normal_chinese_char_len==gbk_chinese_char_len)
                return 0;
            size_t deviation_value=str.Size()/normal_chinese_char_len;
            return deviation_value;
        }
```

3.3.4 示例

示例同 3.1.4

3.4 考生信息插入功能

3.4.1 实现思路

考生信息插入功能的函数名为 MyInsert，考生信息插入功能实现的思路为：

1.输入需要插入考生的位置，合法位置为 1 到当前考生数量+1 之间；

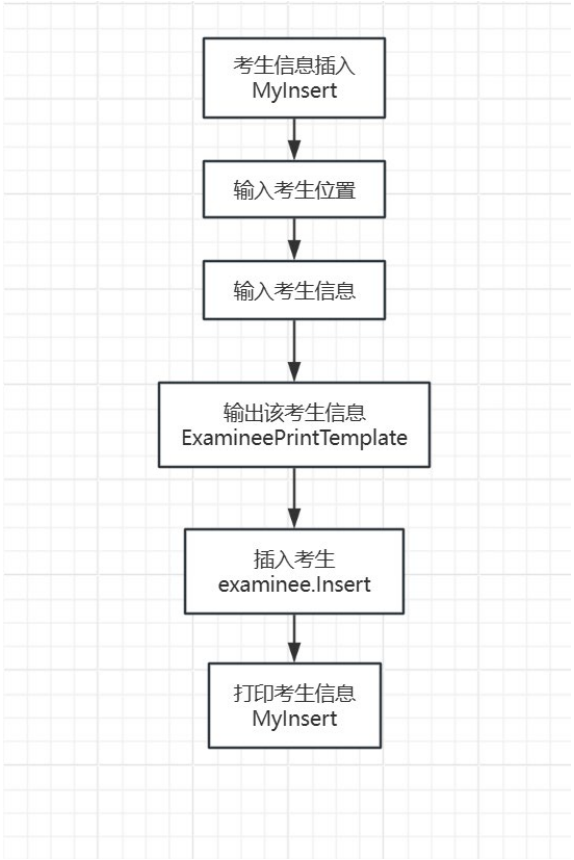
2.调用 Input 函数，输入需要插入的考生信息，包括考生的考号、姓名、性别、年龄和报考类别；

3.输出相关表头后，调用 ExamineePrintTemplate 函数输出插入的考生的信息；

4.调用 examinee.Insert 函数，将新考生信息插入到考生信息系统中的指定位置。

5. 退出 MyInsert 函数，执行 MyPrint 函数，输出全体考生信息

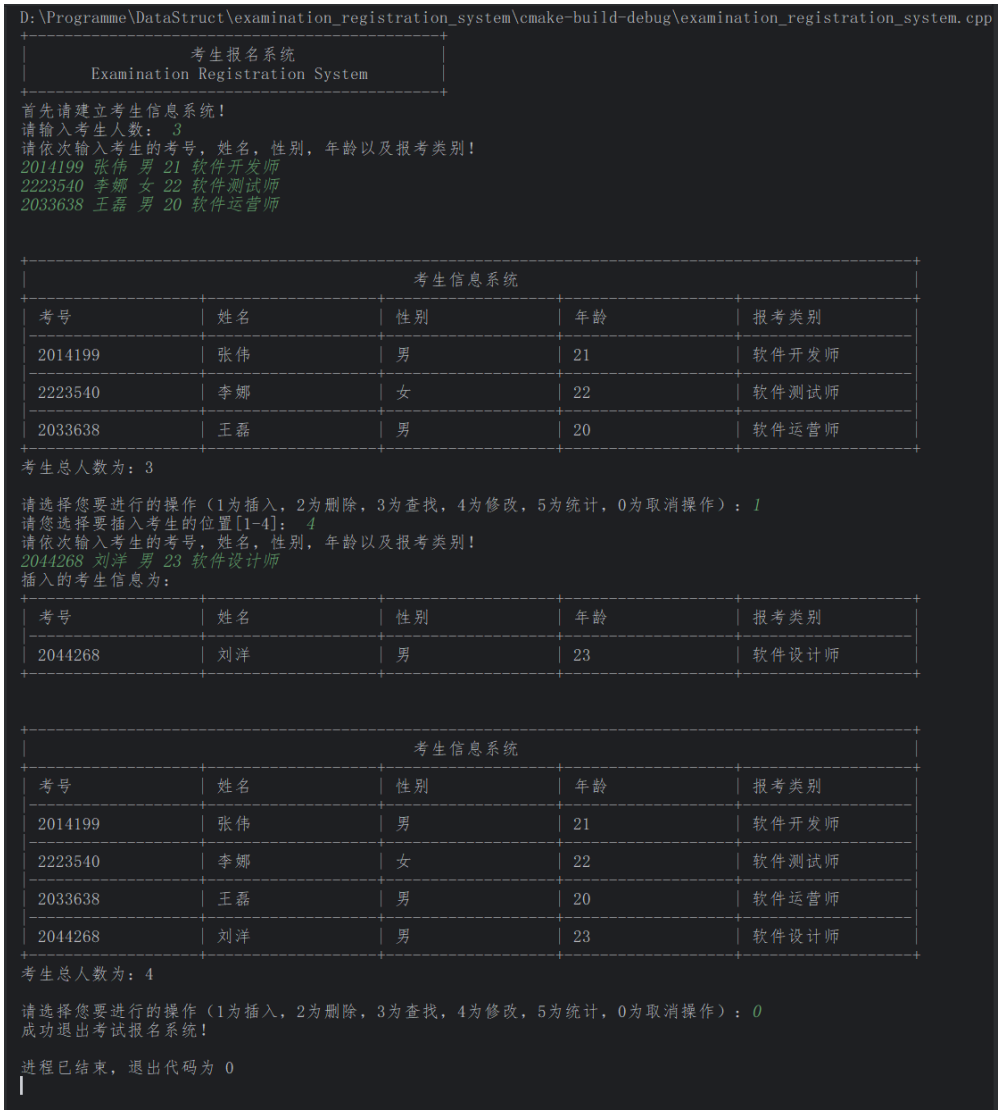
3.4.2 流程图



3.4.3 核心代码

```
void MyInsert(LinkList<Examinee>& examinee)
{
    mine::String num_temp;
    int position;
    while (true) {
        std::cout << "请您选择要插入考生的位置[1-" <<
examinee.Length() + 1 << "]：  ";
        std::cin >> num_temp;
        if !Check(num_temp)||!Switch(num_temp,position) || position<1 ||
position>examinee.Length() + 1) {
            std::cout << "输入非法！请重新输入\n";
            ClearBuffer();
            continue;
        }
        break;
    }
    std::cout << "请依次输入考生的考号，姓名，性别，年龄以及报考
类别！\n";
    Examinee student = Input(examinee);
    std::cout << "插入的考生信息为：  \n";
    std::cout << "+------------------+------------------+------------------+------
------------+------------------+\n";
    std::cout << "| 考号                    | 姓名                    | 性别
| 年龄                    | 报考类别              |\n";
    ExamineePrintTemplate(student);
    std::cout << "+------------------+------------------+------------------+------
------------+------------------+\n";
    examinee.Insert(position,student);
}
```

3.4.4 示例



3.5 考生信息删除功能

3.5.1 实现思路

考生信息删除功能的函数名为 MyExpurgate，考生信息删除功能实现的思路为：

1. 调用 PositionFind 函数获取待删除的考生的位置。位置通过输入考生的考号来唯一确定的；通过迭代器查找考生是否存在，存在返回在链表中的位置，否则返回-1；

2. 创建一个 Examinee 结构体对象 student 用于存储待删除的考生信息；

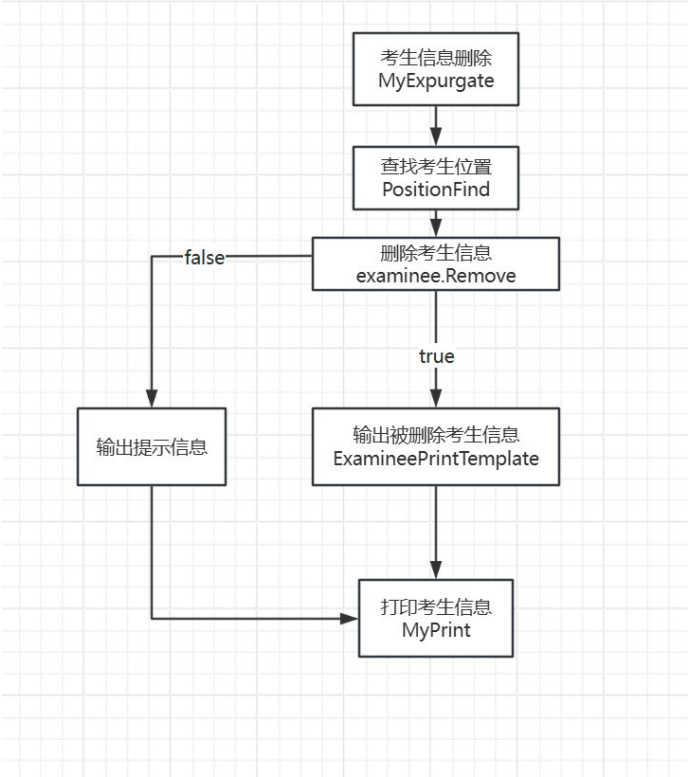
3. 调用 examinee.Remove 函数，删除指定位置考生，并通过引用返回考生信息；

4.输出相关表头后，调用 ExamineePrintTemplate 函数输出删除的考生的信息

5.若 PositionFind 返回值为-1，则 examinee.Remove 返回 false，则提示未查找到考生信息。

6. 退出 MyExpurgate 函数，执行 MyPrint 函数，输出全体考生信息。

3.5.2 流程图



3.5.3 核心代码

```
void MyExpurgate(LinkList<Examinee>& examinee)
{
    int position= PositionFind(examinee,"删除");
    Examinee student;
    if (examinee.Remove(position, student)) {
        std::cout << "删除的考生信息为：  \n";
        std::cout << "+------------------+------------------+------------------+--
----------------+------------------+\n";
        std::cout << "| 考号                  | 姓名                  | 性
```

```
别                    | 年龄                  | 报考类别                  |\n";
            ExamineePrintTemplate(student);
            std::cout << "+------------------+------------------+------------------+--
----------------+------------------+\n";
        }
        else
            std::cout << "未查找到考生信息！\n";
    }
```

3.5.4 示例

```
D:\Programme\DataStruct\examination_registration_system\cmake-build-debug\examination_registration_system.cpp
+-----------------------------------------+
|                考生报名系统               |
|       Examination Registration System    |
+-----------------------------------------+
首先请建立考生信息系统！
请输入考生人数： 4
请依次输入考生的考号，姓名，性别，年龄以及报考类别！
2014199 张伟 男 21 软件开发师
2223540 李娜 女 22 软件测试师
2033638 王磊 男 20 软件运营师
2044268 刘洋 男 23 软件设计师


+-----------------------------------------------------------------------------------------+
|                                        考生信息系统                                     |
+------------------+------------------+------------------+------------------+------------------+
| 考号             | 姓名             | 性别             | 年龄             | 报考类别         |
+------------------+------------------+------------------+------------------+------------------+
| 2014199          | 张伟             | 男               | 21               | 软件开发师       |
+------------------+------------------+------------------+------------------+------------------+
| 2223540          | 李娜             | 女               | 22               | 软件测试师       |
+------------------+------------------+------------------+------------------+------------------+
| 2033638          | 王磊             | 男               | 20               | 软件运营师       |
+------------------+------------------+------------------+------------------+------------------+
| 2044268          | 刘洋             | 男               | 23               | 软件设计师       |
+------------------+------------------+------------------+------------------+------------------+
考生总人数为：4

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）：2
请输入要删除的考生的考号：2033638
删除的考生信息为：
+------------------+------------------+------------------+------------------+------------------+
| 考号             | 姓名             | 性别             | 年龄             | 报考类别         |
+------------------+------------------+------------------+------------------+------------------+
| 2033638          | 王磊             | 男               | 20               | 软件运营师       |
+------------------+------------------+------------------+------------------+------------------+


+-----------------------------------------------------------------------------------------+
|                                        考生信息系统                                     |
+------------------+------------------+------------------+------------------+------------------+
| 考号             | 姓名             | 性别             | 年龄             | 报考类别         |
+------------------+------------------+------------------+------------------+------------------+
| 2014199          | 张伟             | 男               | 21               | 软件开发师       |
+------------------+------------------+------------------+------------------+------------------+
| 2223540          | 李娜             | 女               | 22               | 软件测试师       |
+------------------+------------------+------------------+------------------+------------------+
| 2044268          | 刘洋             | 男               | 23               | 软件设计师       |
+------------------+------------------+------------------+------------------+------------------+
考生总人数为：3

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）：0
成功退出考试报名系统！

进程已结束，退出代码为 0
```

3.6 考生信息查找功能

3.6.1 实现思路

考生信息查询功能的函数名为 MyFind，考生信息查询功能实现的思路为：

1.调用 PositionFind 函数获取待查询的考生的位置。位置通过输入考生的考号来唯一确定的；通过迭代器查找考生是否存在，存在返回在链表中的位置，否则返回-1；

2.创建一个 Examinee 结构体对象 student 用于存储待查询；

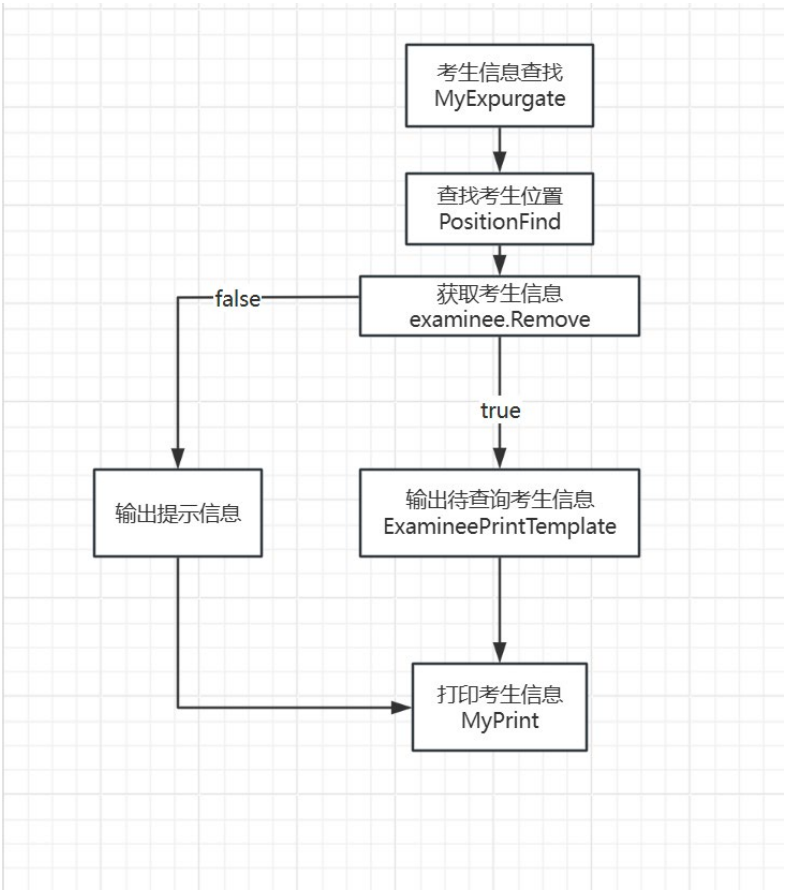
3.调用 examinee.getData 函数，返回需查询考生信息；

4.输出相关表头后，调用 ExamineePrintTemplate 函数输出需查询的考生的信息

5.若 PositionFind 返回值为-1，则 examinee.getData 返回 false，则提示未查找到考生信息；

6.退出 MyFind 函数，执行 MyPrint 函数，输出全体考生信息。

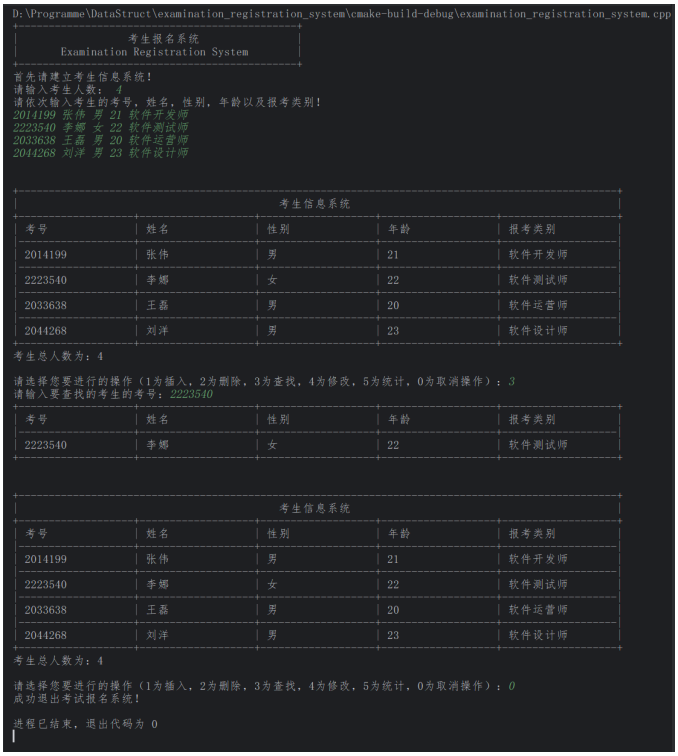
3.6.2 流程图



3.6.3 核心代码

```
void MyFind(LinkList<Examinee>& examinee)
{
    int position = PositionFind(examinee, "查找");查找考生位置
    Examinee student;
    if (examinee.getData(position, student)) {查找到考生并获取数据
        std::cout << "+------------------+------------------+------------------+----
----------------+------------------+\n";
        std::cout << "| 考号             | 姓名             | 性别
| 年龄             | 报考类别         |\n";
        ExamineePrintTemplate(student);
        std::cout << "+------------------+------------------+------------------+----
----------------+------------------+\n";
    }
    else
        std::cout << "未查找到考生信息！\n";
}
```

3.6.4 示例



3.7 考生信息修改功能

3.7.1 实现思路

考生信息查询功能的函数名为 MyModify，考生信息查询功能实现的思路为：

1.调用 PositionFind 函数获取待查询的考生的位置。位置通过输入考生的考号来唯一确定的；通过迭代器查找考生是否存在，存在返回在链表中的位置，否则返回-1；

2.调用 examinee.Remove 删除当前考生信息；

3. 调用 examinee.Remove 删除当前考生信息，若成功则输出相关表头，调用 ExamineePrintTemplate 函数输出需查询的考生的信息；反之，则提示未查找到考生信息，并退出 MyFind 函数；

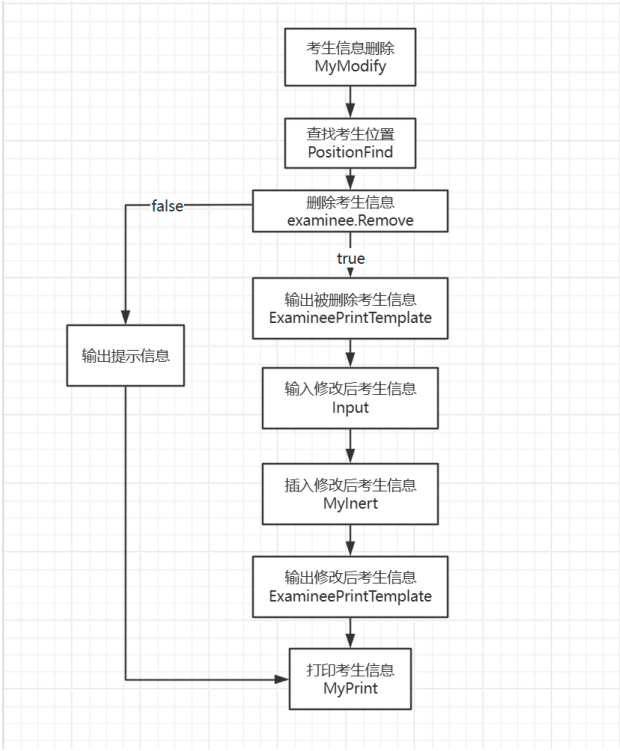
4.调用 Input 函数输入修改后的考生信息；

5. 输出相关表头后，调用 ExamineePrintTemplate 函数输出修改后的考生信息；

5.调用 examinee.Insert 将修改后的考生信息掺入到原有位置；

6.退出 MyModify 函数，执行 MyPrint 函数，输出全体考生信息。

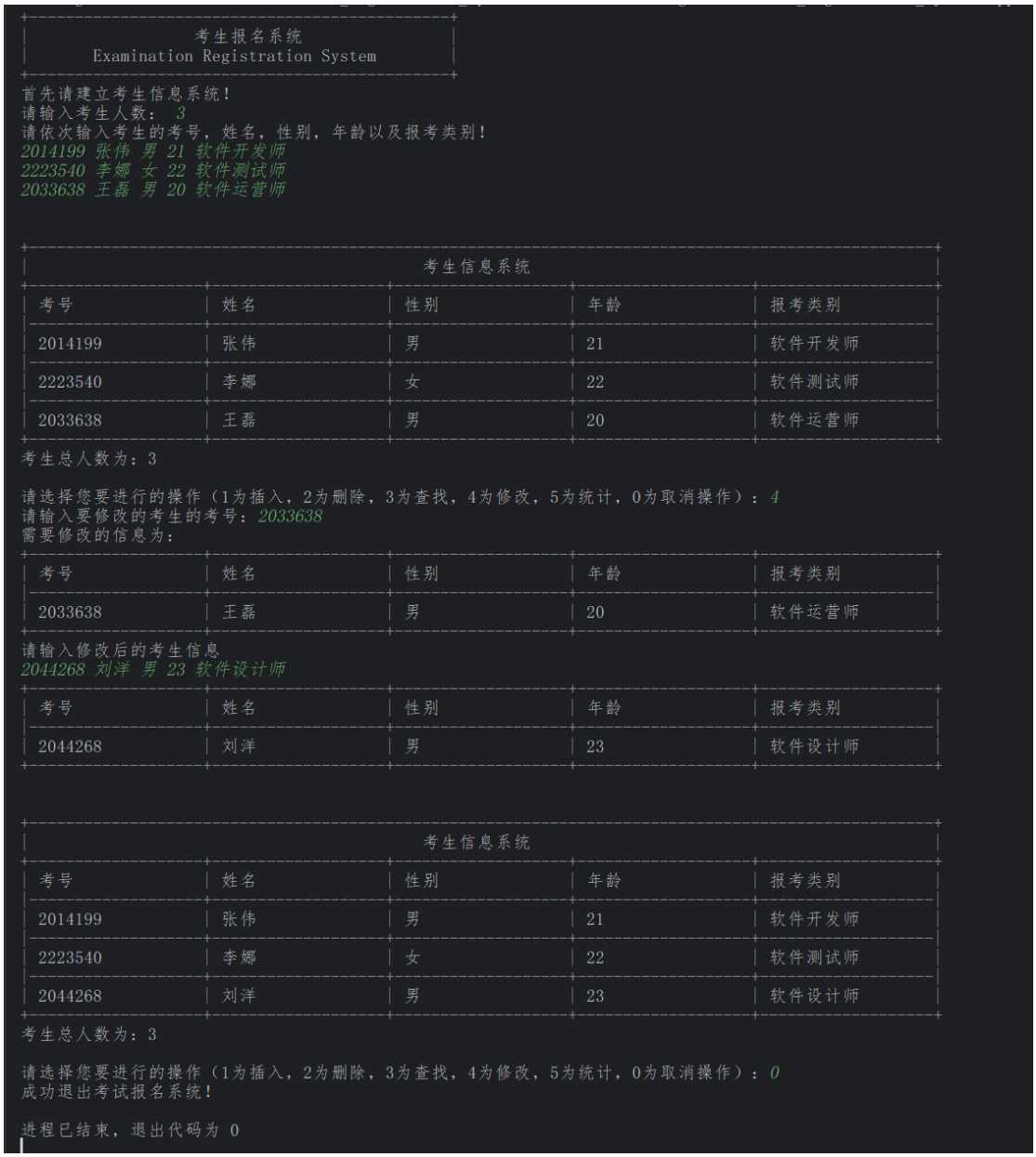
3.7.2 流程图



3.7.3 核心代码

```
void MyModify(LinkList<Examinee>& examinee)
{
    int position = PositionFind(examinee, "修改");
    Examinee student;
    if (examinee.Remove(position, student)) {
        std::cout << "需要修改的信息为：\n";
        std::cout << "+------------------+------------------+------------------+----
--------------+------------------+\n";
        std::cout << "| 考号                    | 姓名                    | 性别
| 年龄                    | 报考类别              |\n";
        ExamineePrintTemplate(student);
        std::cout << "+------------------+------------------+------------------+----
--------------+------------------+\n";
    }
    else {
        std::cout << "未查找到考生信息！\n";
        return;
    }
    std::cout << "请输入修改后的考生信息\n";
    student = Input(examinee);
    std::cout << "+------------------+------------------+------------------+--------
----------+------------------+\n";
    std::cout << "| 考号                    | 姓名                    | 性别
| 年龄                    | 报考类别              |\n";
    ExamineePrintTemplate(student);
    std::cout << "+------------------+------------------+------------------+--------
----------+------------------+\n";
    examinee.Insert(position, student);
}
```

3.7.4 示例



3.8 考生信息统计功能

3.8.1 实现思路

统计考生功能的函数名为 MyStatisc，统计考生功能实现的思路为：

1. 调用 examinee.Length()函数获取考生信息系统中的学生总数 total；

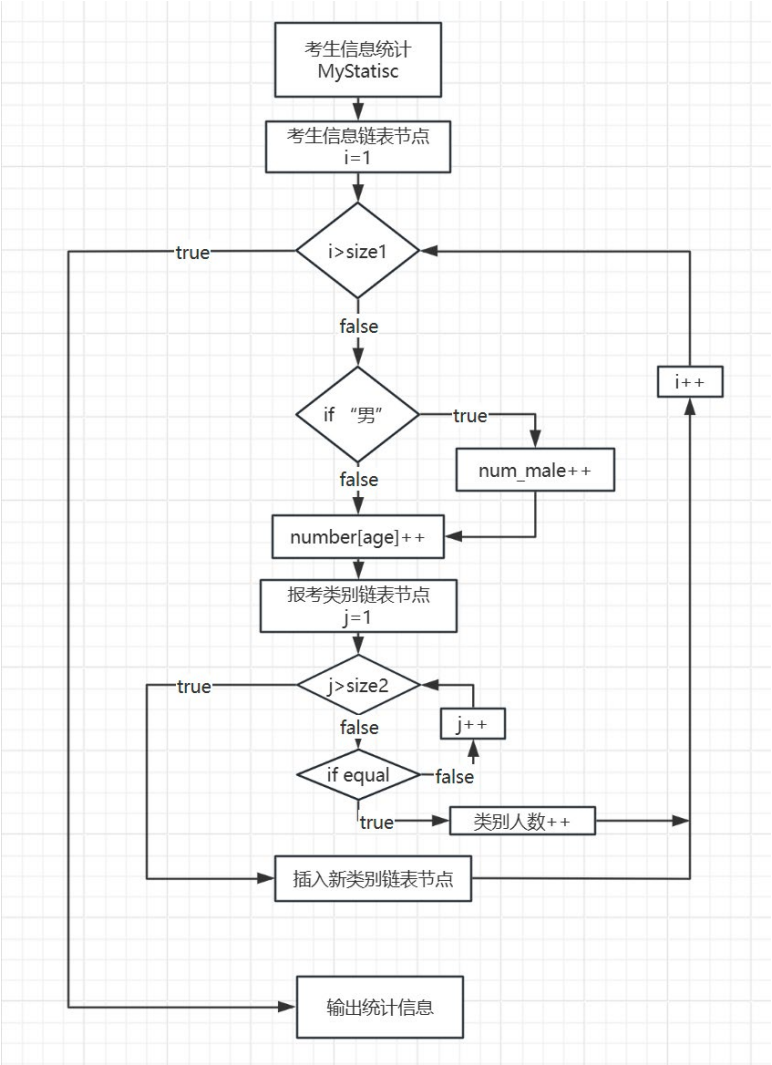
2. 创建变量 male_num 用于统计男性学生的数量，以及一个数组 age_number 用于统计年龄为 i 的考生数目 age_number[i]；

3. 创建一个链表 category_statisc，结点储存报考类别及该报考类别的人数

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。通过迭代器遍历考生信息系统，逐一处理每个考生信息：若为男性则 male_num 加 1，年龄为 i age_number[i]加 1，迭代器遍历 category_statisc 链表，若该报考类别存在，则对应人数加 1；若不存在则建立新的结点储存当前报考类别，且人数置为 1；

5. 根据统计情况输出结果。

3.8.2 流程图



3.8.3 核心代码

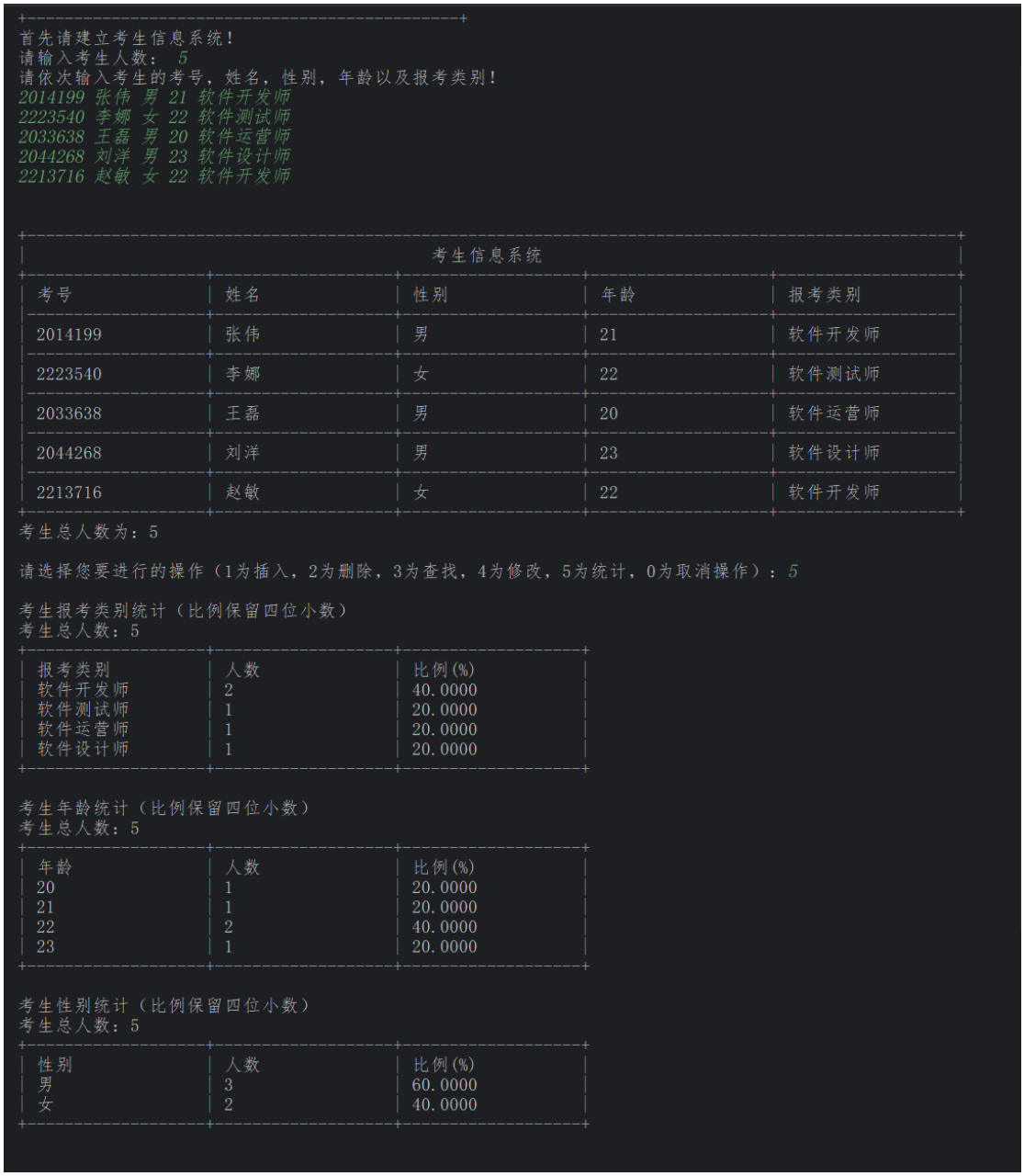
```
void MyStatisc(LinkList<Examinee>& examinee)
{
```

```
    int total = examinee.Length(), male_num = 0, age_number[kMaxAge+1]
= { 0 };
    LinkList<Category> category_statisc;
    for (auto exam: examinee) {
        if (exam.sex=="男")
            male_num++;
        age_number[exam.age]++;
        bool is_stored = false;
        int n=0;
        for (auto category: category_statisc) {
            n++;
            if (category.entry_category== exam.entry_category) {
                category_statisc.Remove(n,category);
                is_stored = true;
                category.number++;
                category_statisc.Insert(n,category);
                break;
            }
        }
        if (!is_stored) {
            Category temp;
            temp.entry_category= exam.entry_category;
            temp.number = 1;
            category_statisc.inputRear(temp);
        }
    }
    std::cout << "\n 考生报考类别统计（比例保留四位小数）\n"
        << "考生总人数：" << total << "\n";
    std::cout << "+------------------+------------------+------------------+\n";
    std::cout << "| 报考类别          | 人数             | 比例(%)
|\n";
    for (auto category : category_statisc)
        std::cout << std::setiosflags(std::ios::left)
        << "| " << std::setw(18+EncodingDeviation(category.entry_category))
```

```
<< category.entry_category
            << "| " << std::setw(18) << category.number
            << "| " << std::setw(18) << std::fixed << std::setprecision(4) <<
static_cast<double>(category.number) / static_cast<double>(total) * 100 << "|"
            << "\n";
        std::cout << "+------------------+------------------+------------------+\n";
        std::cout << "\n 考生年龄统计（比例保留四位小数）\n"
            << "考生总人数：" << total << "\n";
        std::cout << "+------------------+------------------+------------------+\n";
        std::cout << "| 年龄                    | 人数                    | 比例(%)
|\n";
        for (int i = 0; i <= 150; i++)
            if(age_number[i]!=0)
            std::cout<< "| " << std::setw(18) << i
                << "| " << std::setw(18) << age_number[i]
                << "| " << std::setw(18) << std::fixed << std::setprecision(4) <<
static_cast<double>(age_number[i]) / static_cast<double>(total)*100 << "|"
            << "\n";
        std::cout << "+------------------+------------------+------------------+\n";
        std::cout << "\n 考生性别统计（比例保留四位小数）\n"
            << "考生总人数：" << total << "\n";
        std::cout << "+------------------+------------------+------------------+\n";
        std::cout << "| 性别                    | 人数                    | 比例(%)
|\n";
        std::cout << "| "<<std::setw(18+EncodingDeviation("男")) << "男" << "|
"<<std::setw(18) << male_num <<"| "<<std::setw(18)<< std::fixed <<
std::setprecision(4) << static_cast<double>(male_num) /
static_cast<double>(total) * 100 << "|\n"
                    << "| "<<std::setw(18+EncodingDeviation("女")) << "女" <<
"| "<<std::setw(18) << total - male_num <<"| "<<std::setw(18)<< std::fixed <<
std::setprecision(4) << static_cast<double>(total - male_num) /
static_cast<double>(total) * 100 << "|\n";
        std::cout << "+------------------+------------------+------------------+\n";
```

```
        std::cout << std::resetiosflags(std::ios::left);
    }
```

3.8.4 示例



3.9 异常处理功能

3.9.1 输入非法的异常处理

3.9.1.1 考生信息系统建立时输入错误的异常处理

在建立考生信息系统时，首先要求输入考生人数，通过一个 while 无限循环来确保输入的考生人数无误；当输入错误时，会清除缓冲区，继续执行循环函数，当输入正确时，则会通过 break 退出循环，继续下一步输入操作。

在输入每个考生的基本信息时，会调用 Input 函数。为了规范输入，对输入的内容做出以下限制：

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 int 类型上限，且不能与已有考号重复；

3.姓名：无限制；

4.性别：只能输入男或女；

5.年龄：在 0 至 150 范围内的整型数据；如需调整，只需调整全局变量；

6.报考类别：无限制。

Input 函数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.创建临时学生信息对象，用以储存考生信息；

1.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.输入考号并验证：用户以字符串的方式输入考生的考号，通过 Check 函数和 Switch 函数验证输入字符串是否符合规范，规范则返回 int 类型考号，并进行范围判断；若不规范或不符合范围则清除当前输入状态和缓冲区，开始重新从头开始执行循环；若输入符合规范和范围，还应当对考号是否重复进行验证，调用 IsSameExaminationNumber 与已有考生考号对比，若重复则提示考号重复，清除当前输入状态和缓冲区，重新从头开始执行循环，反之，则开始输入考生姓名。

3.输入考生姓名；

4.输入并验证性别：用户输入考生的性别，并通过与固定字符串“男”或“女”对比判断是否相等，若都不等则提示输入不合法，清除当前输入状态和缓冲区，开始重新从头开始执行循环；反之则开始开始输入考生年龄；

5.输入并验证年龄：用户以字符串的方式输入考生的年龄，通过 Check 函数和 Switch 函数验证输入字符串是否符合规范，规范则返回 int 类型年龄，并进行范围判断；若不规范或不符合范围则清除当前输入状态和缓冲区，开始重新从头开始执行循环；反之，则开始输入考生报考类别；

6. 输入考生报考类别；

7. 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，且考号不重复，函数将返回输入的学生信息对象，考生信息输入结束。

3.9.1.2 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 MySelect(LinkList<Examinee>& examinee)
{
    mine::String num_temp;
    int select;
    while (true) {
        std::cout << "\n 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）：";
        std::cin >> num_temp;
        if (!Check(num_temp)||!Switch(num_temp,select)|| select>5) {
            std::cout << "输入非法！请重新输入\n";
            ClearBuffer();
            continue;
        }
        break;
    }
    return select;
}
```

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. 用户以字符串的方式输入选项，通过 Check 函数和 Switch 函数验证输入字符串是否符合规范，规范则返回 int 类型选项，并进行范围判断；若不规范或不符合范围则清除当前输入状态和缓冲区，开始重新从头开始执行循环；反之，则退出循环。

3.9.1.3 插入位置输入非法的异常处理

在插入考生地时候，需要输入插入的位置，输入非法的异常处理通过如下代码实现：

```
mine::String num_temp;
```

```
int position;
while (true) {
    std::cout << "请您选择要插入考生的位置[1-" << examinee.Length() + 1
<< "]:  ";
    std::cin >> num_temp;
    if (!Check(num_temp)||!Switch(num_temp,position) || position<1 ||
position>examinee.Length() + 1
    ) {
        std::cout << "输入非法！请重新输入\n";
        ClearBuffer();
        continue;
    }
    break;
}
```

用户以字符串的方式输入插入的位置，通过 Check 函数和 Switch 函数验证输入字符串是否符合规范，规范则返回 int 类型位置，并进行范围判断；若不规范或不符合范围则清除当前输入状态和缓冲区，开始重新从头开始执行循环；反之，则退出循环。

3.9.2 动态内存申请失败的异常处理

在进行 Linklist 类与 String 类的动态内存申请时，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，对于内存分配失败，可以采用两种方式处理：

1. 判断 new(std::nothrow)是否返回 nullptr，是则使用 exit 函数退出程序，并限定返回值，示例：

```
template <typename T>
LinkList<T>::LinkList(){
    head = new(std::nothrow) Node<T>;
    if (head == nullptr) {
        std::cout << "内存分配错误！\n";
        exit(MEMORY_ALLOCATION_ERROR);
    }
```

```
}
```

2. 通过 assert 语句判断 new(std::nothrow)是否返回 nullptr，示例：

```
inline String::String(const char* str)
{
    _str_len = Strlen(str);
    _str_cap = _str_len;
    _str = new(std::nothrow) char[_str_cap + 1];//多开一位用来存放'\0'
    assert(_str != nullptr);
    memcpy(_str, str, _str_len + 1);
}
```

3.9.3 索引越界的异常处理

在 LinkList 类中，getData、setData、Remove 等函数通过索引来查找结点时，实际是通过 Locate 通过索引定位，对与索引越界的情况，Locate 返回的地址为 nullptr，各个函数根据返回值是否为 nullptr 来确定返回值是 true 还是 false，false 则表明越界访问；对于 Insert 函数，其索引通过 3.9.1.3 来限定，不会出现越界访问的情况。

在 String 类中，许多操作，例如 Insert、Erase、Find、Substr 等函数，都需要通过索引访问，通过 assert 函数来判断索引是否正确，一般为 assert(pos < _str_len)，也可能为 assert(pos <= _str_len)。

第 4 章 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、负数、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。

```
D:\Programme\DataStruct\examination_registration_system\cmake-build-debug\examination_registration_system.cp
+-----------------------------------------+
|                考生报名系统               |
|       Examination Registration System    |
+-----------------------------------------+
首先请建立考生信息系统！
请输入考生人数： 99999999999999999999
输入人数不合法，请重新输入！
请输入考生人数： -99999999999999999999
输入人数不合法，请重新输入！
请输入考生人数： -5
输入人数不合法，请重新输入！
请输入考生人数： 5.5
输入人数不合法，请重新输入！
请输入考生人数： f
输入人数不合法，请重新输入！
请输入考生人数： ewave
输入人数不合法，请重新输入！
请输入考生人数：
```

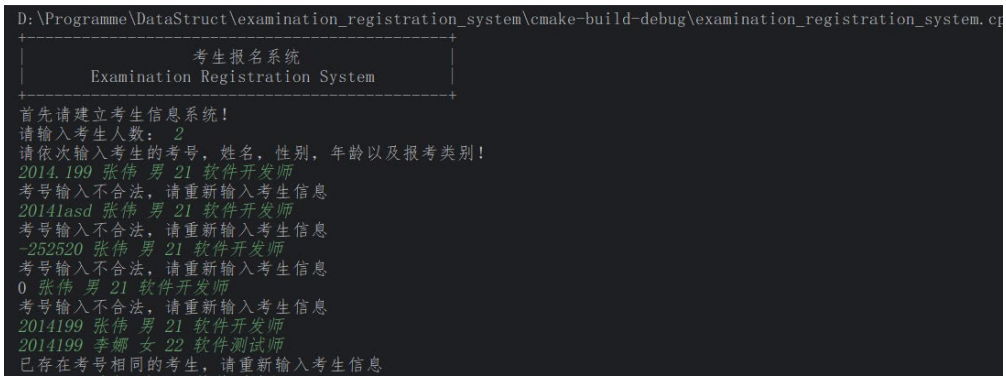
当输入合法时，程序继续运行

```
请输入考生人数： 5
请依次输入考生的考号，姓名，性别，年龄以及报考类别！
2014199 张伟 男 21 软件开发师
2223540 李娜 女 22 软件测试师
2033638 王磊 男 20 软件运营师
2044268 刘洋 男 23 软件设计师
2213716 赵敏 女 22 软件开发师


+-------------------------------------------------------------------------------------+
|                                      考生信息系统                                   |
+-----------+-----------+-----------+-----------+-----------+
| 考号      | 姓名      | 性别      | 年龄      | 报考类别  |
+-----------+-----------+-----------+-----------+-----------+
| 2014199   | 张伟      | 男        | 21        | 软件开发师|
+-----------+-----------+-----------+-----------+-----------+
| 2223540   | 李娜      | 女        | 22        | 软件测试师|
+-----------+-----------+-----------+-----------+-----------+
| 2033638   | 王磊      | 男        | 20        | 软件运营师|
+-----------+-----------+-----------+-----------+-----------+
| 2044268   | 刘洋      | 男        | 23        | 软件设计师|
+-----------+-----------+-----------+-----------+-----------+
| 2213716   | 赵敏      | 女        | 22        | 软件开发师|
+-----------+-----------+-----------+-----------+-----------+
考生总人数为：5

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）：
```

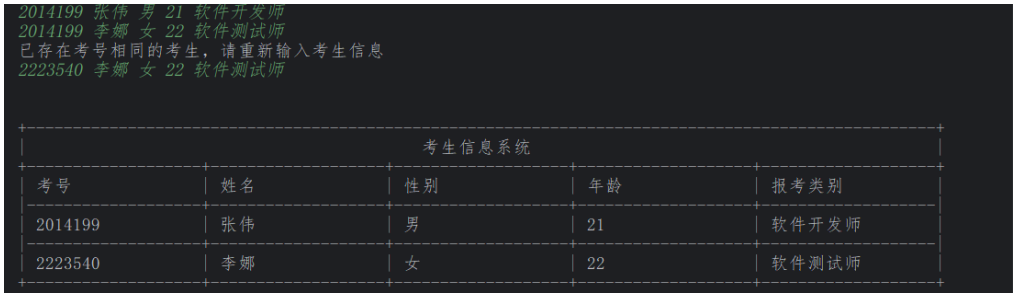
4.1.2 考生信息录入功能测试

4.1.2.1 录入考生考号功能测试

考生考号为不超过 2147483647 的正整数，且若已存在考号相同的考生，则重新输入。

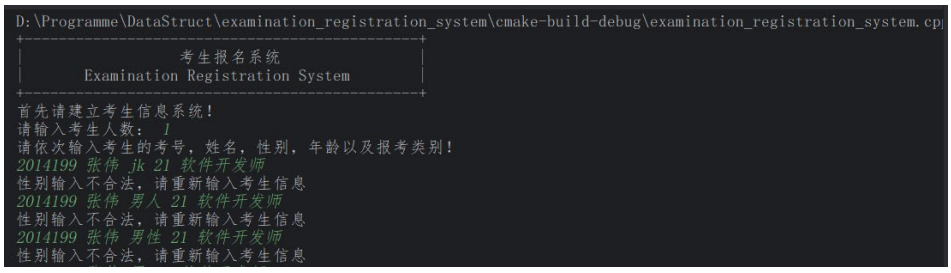


当输入合法时，程序继续运行。

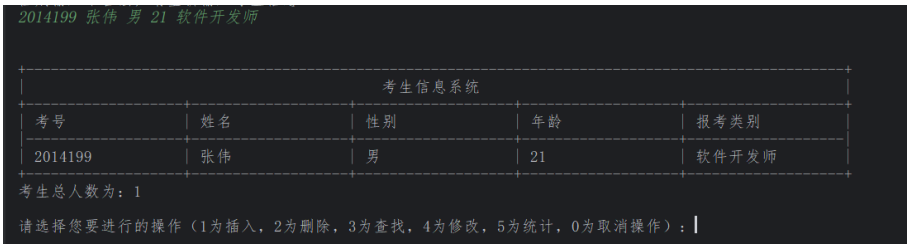


4.1.2.2 录入考生性别功能测试

程序对考生性别信息的只能输入“男”或“女”



当输入合法时，程序继续运行



4.1.2.3 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 150 范围内的整型数据。

```
D:\Programme\DataStruct\examination_registration_system\cmake-build-debug\examination_registration_system.cp
+-----------------------------------------+
|                 考生报名系统                |
|       Examination Registration System       |
+-----------------------------------------+
首先请建立考生信息系统！
请输入考生人数： 1
请依次输入考生的考号，姓名，性别，年龄以及报考类别！
2014199 张伟 男 -1 软件开发师
年龄输入不合法，请重新输入考生信息
2014199 张伟 男 1.1 软件开发师
年龄输入不合法，请重新输入考生信息
2014199 张伟 男 s 软件开发师
年龄输入不合法，请重新输入考生信息
2014199 张伟 男 151 软件开发师
年龄输入不合法，请重新输入考生信息
```

当输入合法时，程序继续运行

```
2014199 张伟 男 21 软件开发师


+-------------------------------------------------------------------------------------+
|                                      考生信息系统                                     |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 考号            | 姓名            | 性别            | 年龄            | 报考类别    |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 2014199         | 张伟            | 男              | 21              | 软件开发师  |
+-----------------+-----------------+-----------------+-----------------+-------------+
考生总人数为：1

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）：
```

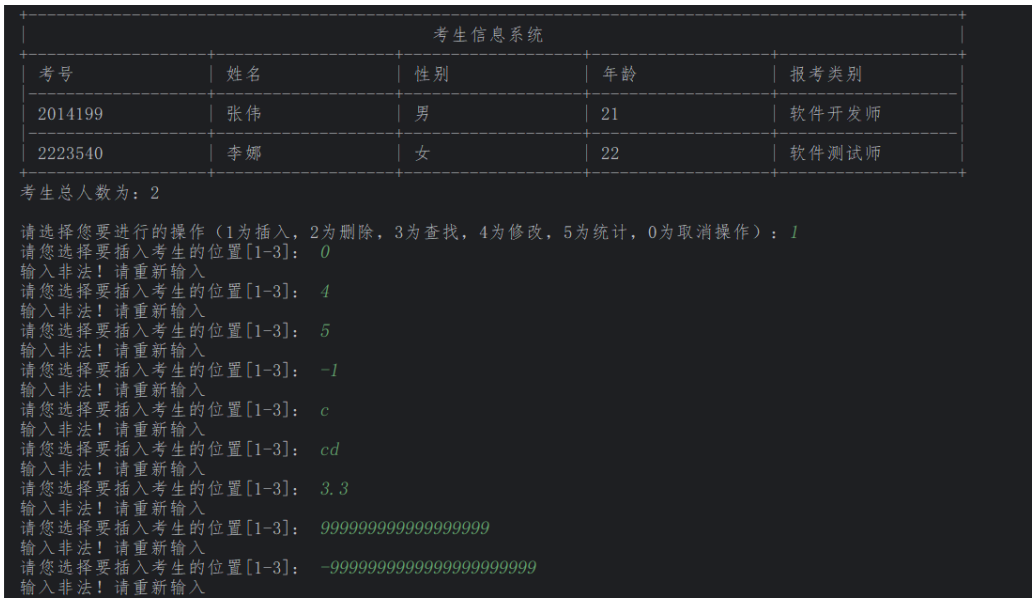
4.1.3 考生信息输出功能测试

输出考生信息分为总体输入和单个输出的情况，总体输出通过调用 MyPrint 函数，来循环调用 ExamineePrintTemplate 函数实现输出，单个输出只需一次调用 ExamineePrintTemplate 来输出。

```
+-----------------------------------------+
|                 考生报名系统                |
|       Examination Registration System       |
+-----------------------------------------+
首先请建立考生信息系统！
请输入考生人数： 3
请依次输入考生的考号，姓名，性别，年龄以及报考类别！
2014199 张伟 男 21 软件开发师
2223540 李娜 女 22 软件测试师
2033638 王磊 男 20 软件运营师


+-------------------------------------------------------------------------------------+
|                                      考生信息系统                                     |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 考号            | 姓名            | 性别            | 年龄            | 报考类别    |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 2014199         | 张伟            | 男              | 21              | 软件开发师  |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 2223540         | 李娜            | 女              | 22              | 软件测试师  |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 2033638         | 王磊            | 男              | 20              | 软件运营师  |
+-----------------+-----------------+-----------------+-----------------+-------------+
考生总人数为：3

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）：2
请输入要删除的考生的考号：2014199
删除的考生信息为：
+-------------------------------------------------------------------------------------+
| 考号            | 姓名            | 性别            | 年龄            | 报考类别    |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 2014199         | 张伟            | 男              | 21              | 软件开发师  |
+-----------------+-----------------+-----------------+-----------------+-------------+


+-------------------------------------------------------------------------------------+
|                                      考生信息系统                                     |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 考号            | 姓名            | 性别            | 年龄            | 报考类别    |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 2223540         | 李娜            | 女              | 22              | 软件测试师  |
+-----------------+-----------------+-----------------+-----------------+-------------+
| 2033638         | 王磊            | 男              | 20              | 软件运营师  |
+-----------------+-----------------+-----------------+-----------------+-------------+
考生总人数为：2
```

4.2 考生信息插入功能测试

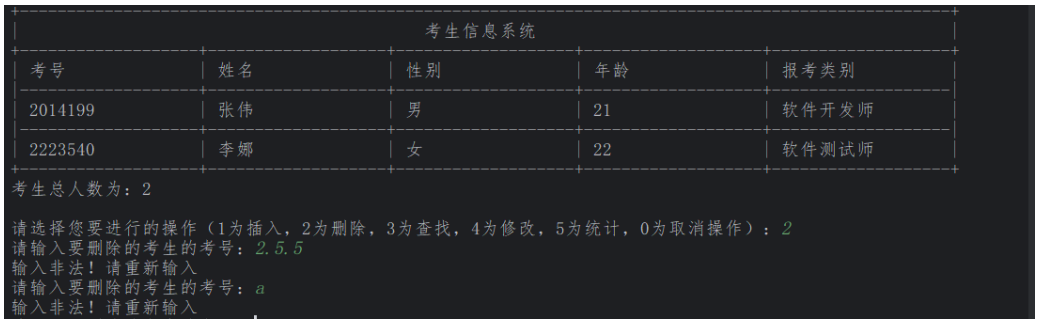
分别输入超过上下限的整数、负数、浮点数、字符、字符串，可以验证程序对输入非法的情况进行了处理。



输入合法数据后，开始输入考生基本信息，见 3.4.4。

4.3 考生信息删除功能测试

考生考号为不超过 2147483647 的正整数，且若已存在考号相同的考生，则重新输入。



当输入合法时，程序继续运行，见 3.5.4。

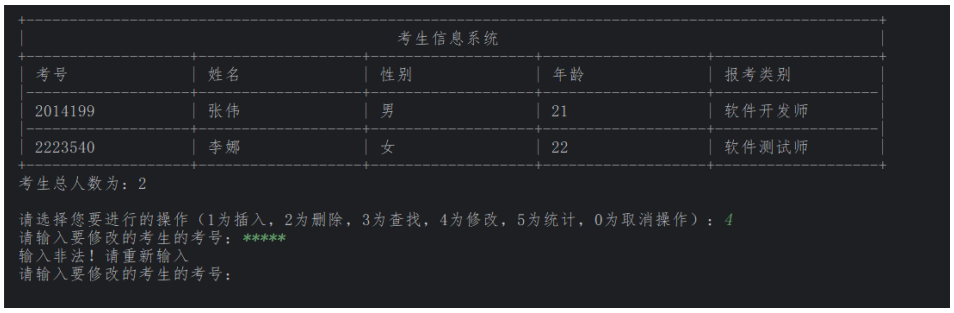
4.4 考生信息查找功能测试

考生考号为不超过 2147483647 的正整数，且若已存在考号相同的考生，则重新输入。

当输入合法时，程序继续运行，见 3.6.4。

4.5 考生信息修改功能测试

考生考号为不超过 2147483647 的正整数，且若已存在考号相同的考生，则重新输入。



当输入合法时，程序继续运行，见 3.7.4。

4.6 考生信息统计功能测试

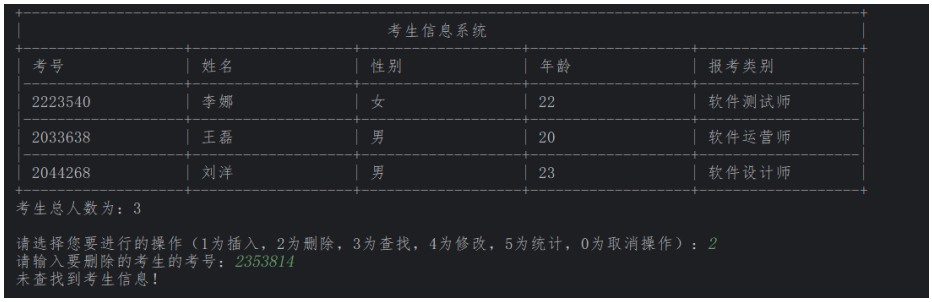
统计考生功能函数统计考生报考类别、性别和年龄数据，并打印其统计信息 （人数和所占比例），见 3.8.4。

4.7 退出考试报名系统功能测试

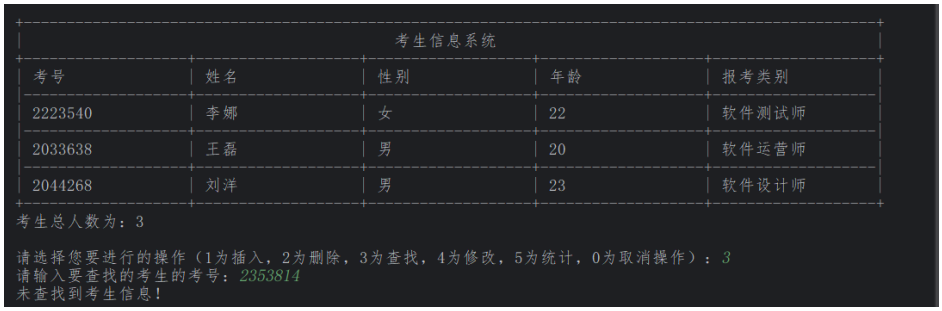
在操作考生信息系统时，选择选项 0 退出考试报名系统。

4.8 其他

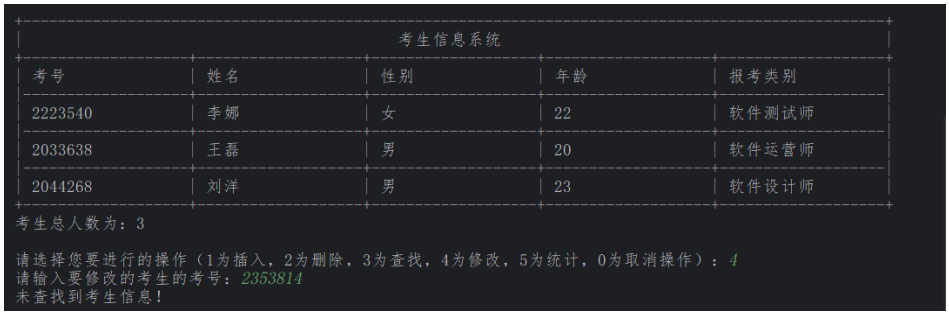
对于删除、查找、修改等功能，都会出现未查找到考生的情况



```
                                    考生信息系统
+----------------+----------------+----------------+----------------+----------------+
| 考号           | 姓名           | 性别           | 年龄           | 报考类别       |
+----------------+----------------+----------------+----------------+----------------+
| 2223540        | 李娜           | 女             | 22             | 软件测试师     |
+----------------+----------------+----------------+----------------+----------------+
| 2033638        | 王磊           | 男             | 20             | 软件运营师     |
+----------------+----------------+----------------+----------------+----------------+
| 2044268        | 刘洋           | 男             | 23             | 软件设计师     |
+----------------+----------------+----------------+----------------+----------------+
考生总人数为：3

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）：2
请输入要删除的考生的考号：2353814
未查找到考生信息！
```



```
                                    考生信息系统
+----------------+----------------+----------------+----------------+----------------+
| 考号           | 姓名           | 性别           | 年龄           | 报考类别       |
+----------------+----------------+----------------+----------------+----------------+
| 2223540        | 李娜           | 女             | 22             | 软件测试师     |
+----------------+----------------+----------------+----------------+----------------+
| 2033638        | 王磊           | 男             | 20             | 软件运营师     |
+----------------+----------------+----------------+----------------+----------------+
| 2044268        | 刘洋           | 男             | 23             | 软件设计师     |
+----------------+----------------+----------------+----------------+----------------+
考生总人数为：3

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）：3
请输入要查找的考生的考号：2353814
未查找到考生信息！
```



```
                                    考生信息系统
+----------------+----------------+----------------+----------------+----------------+
| 考号           | 姓名           | 性别           | 年龄           | 报考类别       |
+----------------+----------------+----------------+----------------+----------------+
| 2223540        | 李娜           | 女             | 22             | 软件测试师     |
+----------------+----------------+----------------+----------------+----------------+
| 2033638        | 王磊           | 男             | 20             | 软件运营师     |
+----------------+----------------+----------------+----------------+----------------+
| 2044268        | 刘洋           | 男             | 23             | 软件设计师     |
+----------------+----------------+----------------+----------------+----------------+
考生总人数为：3

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）：4
请输入要修改的考生的考号：2353814
未查找到考生信息！
```

第 5 章 相关说明

5.1 编程语言

本项目全部.cpp 文件以及.h 文件均使用 C++编译完成。

5.2 Windows 环境

Windows 系统：Windows 11 x64

Windows 集成开发环境：CLion 2024

工具集：MinGW 11.0 w64

5.3 Linux 环境

基于 Linux 内核的操作系统发行版：Ubuntu 24.04.1

Linux 命令编译过程为：

1. 定位项目所在文件夹，包括.pp 与.h 文件；具体命令为: cd /home/bruce/programe/examination_registration_system

2.编译项目，生成可执行文件；具体命令为: g++ -static -o examination_registration_system_linux examination_registration_system.cpp my_singly_link_list.h my_string.h;

其中指令含义分别为：

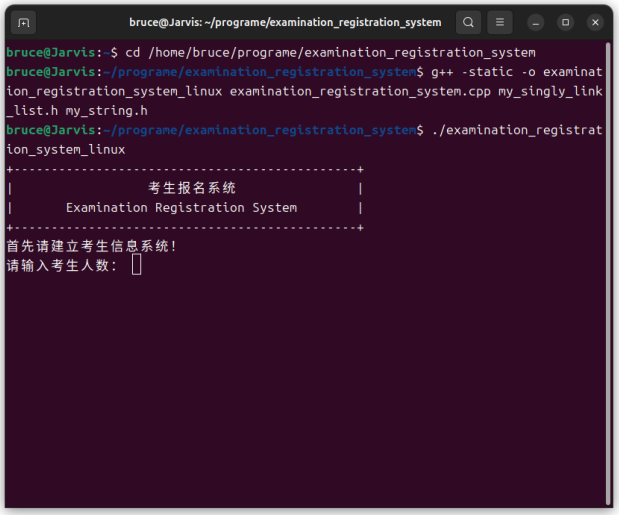
g++: 调用 GNU 的 C++编译器

-static: 使用静态链接而非动态链接，将所有依赖库直接嵌入到可执行文件，文件存储空间变大，但可以单独运行

-o examination_registration_system_linux: -o 表示输出文件选项，examination_registration_system_linux 为可执行文件名

examination_registration_system.cpp my_singly_link_list.h my_string.h:编译所需要的文件。

3.运行可执行文件；具体命令为 ./ examination_registration_system_linux


bruce@Jarvis: ~/programe/examination_registration_system
bruce@Jarvis:~$ cd /home/bruce/programe/examination_registration_system
bruce@Jarvis:~/programe/examination_registration_system$ g++ -static -o examination_registration_system_linux examination_registration_system.cpp my_singly_link_list.h my_string.h
bruce@Jarvis:~/programe/examination_registration_system$ ./examination_registration_system_linux
+---+
| 考生报名系统 |
| Examination Registration System |
+---+
首先请建立考生信息系统!
请输入考生人数: