数据结构项目———考试报名系统

同济大学软件学院 15级2班 1552651 王依睿

C++ 11 编程工具 VS 技术实现 链表

目录

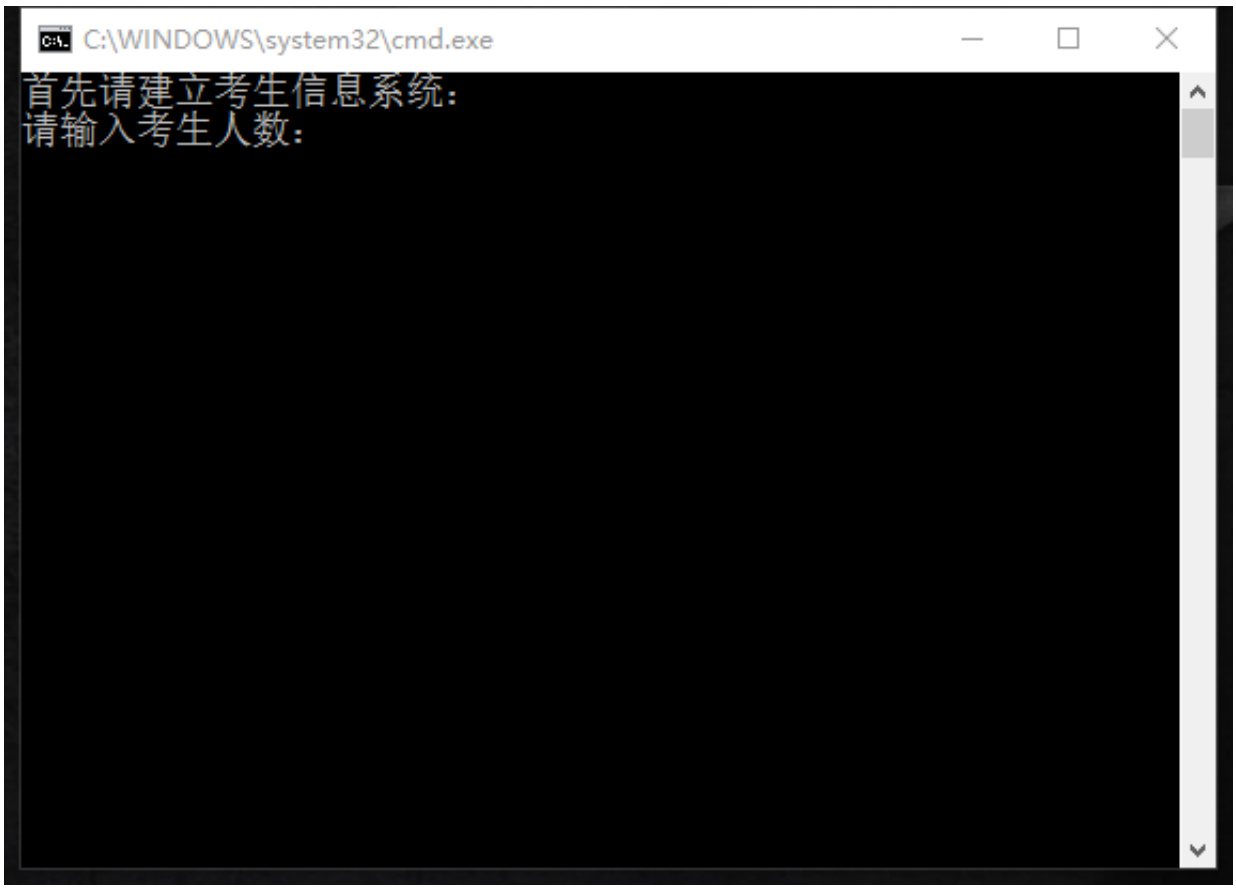
使用说明

操作手册

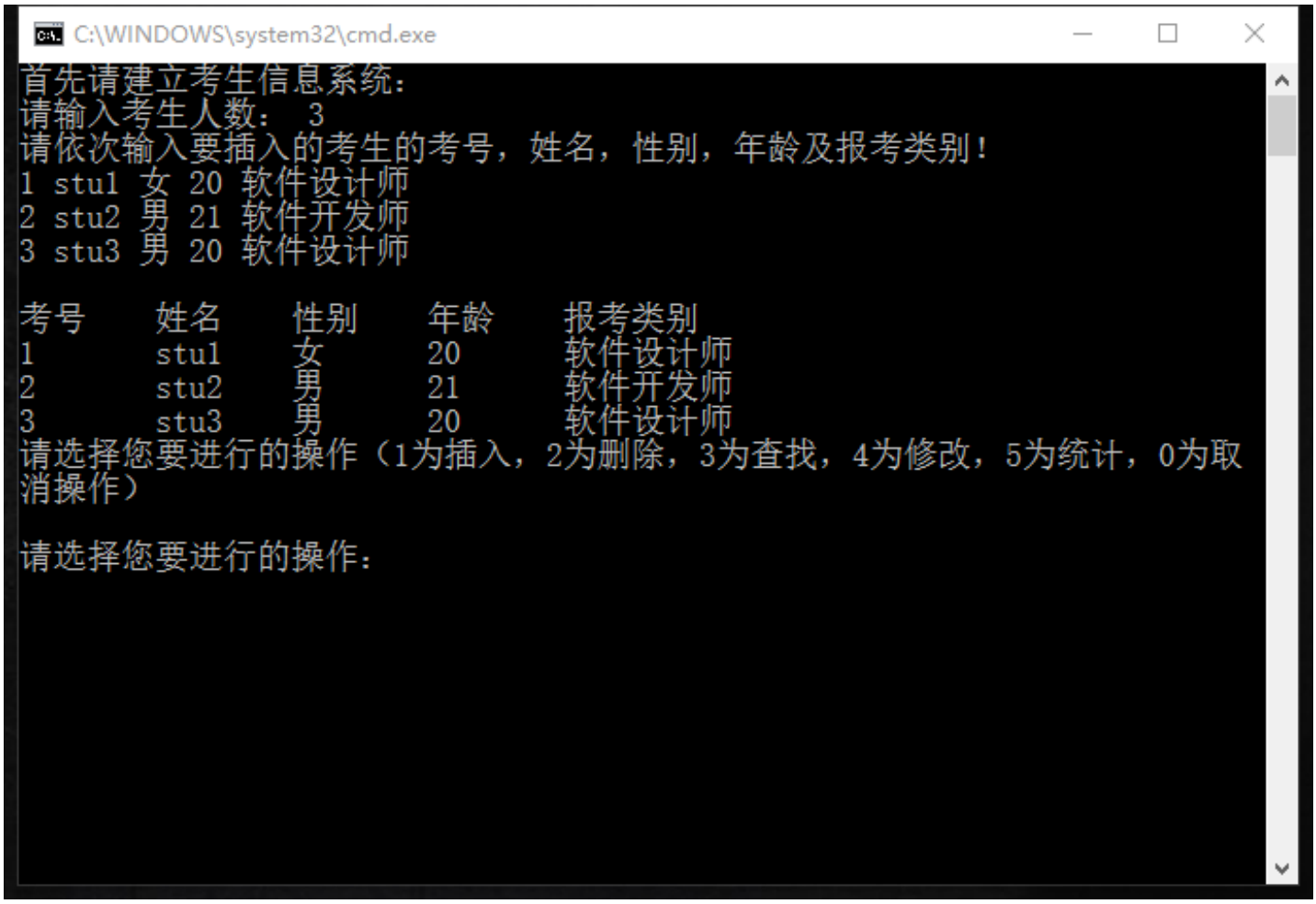
开始

运行程序后，进入欢迎界面，首先要建立考生信息系统。

第一步，输入考生人数。

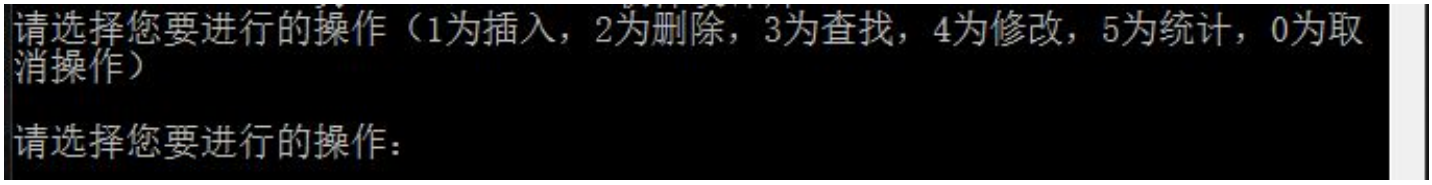


第二步，输入考生信息。



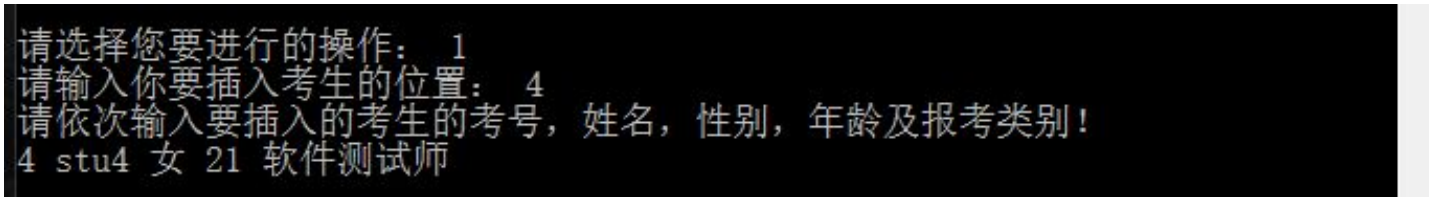
输入结束后，系统会显示输入的数据。

第三步，进行具体操作。



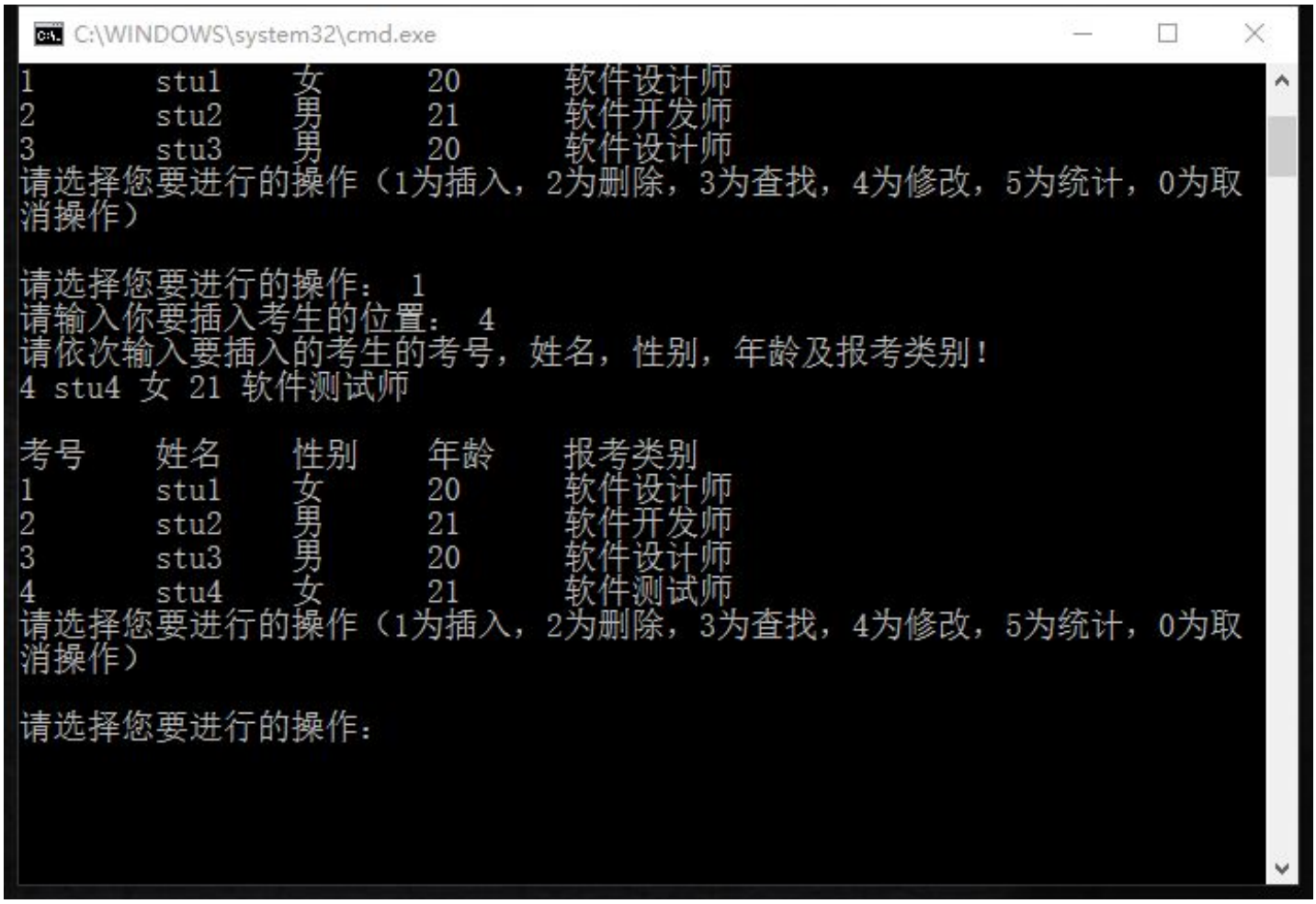
插入

在指定位置插入一个考生信息，例如：



在位置4插入stu4的信息。

成功后，系统会显示插入完成后的所有信息。



删除

删除指定考号的考生信息，例如：

```
请选择您要进行的操作： 2
请输入要删除考生的学号： 2
```

删除考号为2的学生。删除后系统会显示删除的考生信息和删除操作后的所有信息。

```
请选择您要进行的操作： 2
请输入要删除考生的学号： 2

考号 姓名 性别 年龄 报考类别
1 stu1 女 20 软件设计师
3 stu3 男 20 软件设计师
4 stu4 女 21 软件测试师
```

查找

查找指定考号的考生信息，例如：

```
请选择您要进行的操作： 3
请输入要查找的考生的考号： 3
```

查找考号为3的考试信息。查找完成后系统会显示查找到的考生信息。

```
请选择您要进行的操作： 3
请输入要查找的考生的考号： 3

3       stu3    男      20      软件设计师
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）
```

修改

修改指定考号的考生信息，例如：

```
请选择您要进行的操作： 4
请输入要修改的考生的考号： 3
请依次输入要插入的考生的考号，姓名，性别，年龄及报考类别！
3 stu2 男 21 软件工程师
```

修改考号为3的考生的信息。修改完成后系统会显示所有考生信息。

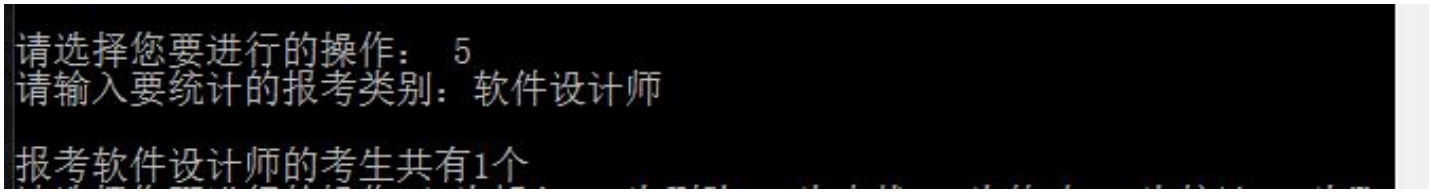
```
请选择您要进行的操作： 4
请输入要修改的考生的考号： 3
请依次输入要插入的考生的考号，姓名，性别，年龄及报考类别！
3 stu2 男 21 软件工程师

3       stu3    男      20      软件设计师

考号    姓名    性别    年龄    报考类别
1       stu1    女      20      软件设计师
3       stu2    男      21      软件工程师
4       stu4    女      21      软件测试师
```

统计

统计报考某一类别的考生数量，例如：

```
请选择您要进行的操作： 5
请输入要统计的报考类别：软件设计师
```

系统会显示报考该类别的人数。



退出

退出系统。

注意事项

系统的考号请勿超过9位（16位操作系统请勿超过5位）。

概述

本项目的实质是完成对考生信息的建立，查找，插入，修改，删除等功能。其中考生信息包括准考证号，姓名，性别，年龄和报考类别等信息。项目在设计时应首先确定系统的数据结构，定义类的成员变量和成员函数；然后实现各成员函数以完成对数据操作的相应功能；最后完成主函数以验证各个成员函数的功能并得到运行结果。（建议采用链表实现）

项目简介

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

程序设计目的

实现一个基于链表的管理统计考生信息的简单系统，练习链表的编写实现，熟悉面向对象程序设计。

算法思路

以顺序遍历为基本算法，实现节点位置的确定，而后通过改变节点的指针实现插入与删除；通过特定的比较算法判断是不是搜索目标；进行一次删除和插入实现数据的修改；逐一检查报考类别进行统计。

数据结构

采用双向无环链表，链表与节点分别封装，以C++模板类实现。学生数据封装为一个类，套用链表模板进行

操作。采用STL中的map记录报考类别及其报考人数。

文件目录

1_1552651_wangyirui.cpp（主文件）

examinee_class.h（节点类封装）

examinee_class.cpp（节点类函数实现）

examineelist_class.h（链表类封装）

examineelist_class.cpp（链表类函数实现）

1_1552651_wangyirui.exe（程序可执行文件）

1_1552651_wangyirui.pdf（项目文档）

函数接口

节点（Examinee）接口

成员函数名	公有/私有性	功能	参数	返回值类型
Examinee(int num, string name, string gender, int age, string category)	public	构造函数	学号int，姓名string，性别string，年龄int，报考类别string	/
int num() const	public	返回类内的num_	/	/
void Output()	public	打印节点内内容	/	/

链表节点类

```
class Examinee {

public:

    Examinee(int num, string name, string gender, int age, string category) :
        num_(num), name_(name), gender_(gender), age_(age), category_(category), p
rior_(NULL), next_(NULL) {};
    Examinee() = default;
    ~Examinee() = default;

    int num() const { return num_; }
    string category() { return category_; }

    //print this node's information
    void Output();

public:
    Examinee *prior_;
    Examinee *next_;

private:
    int num_;
    string name_;
    string gender_;
    int age_;
    string category_;
};
```

链表（Examineelist）接口

成员函数名	公有/私有性	功能	参数	返回值类型
ExamineeList()	private	私有构造函数，使Examineelist作为单例类使用	/	/
~ExamineeList()	private	私有析构函数，通过迭代删除链表的所有节点	/	/
static ExamineeList& Instance()	public	声明并初始化单例类Examineelist	/	ExamineeList&
int total()	public	返回类私有成员total_	/	int
void set_total(int total)	public	设置类私有成员total_	int total	/
Examinee * IndexPosition(int index)	public	通过传入的位置参数index检索第index个节点，返回检索到的节点的指针	int index	/
Examinee * IndexNum(int index)	public	通过传入的学号index检索对应的节点，返回检索到的节点的指针	int index	Examinee *
void Insert(int index, Examinee * node)	public	通过传入的位置参数index和节点指针node在位置index插入节点node	int index, Examinee * node	/
void Delete(int index)	public	通过传入的位置参数index删除链表中位置index的节点	int index	/
void Modify(int index, Examinee * node)	public	通过传入的位置参数index和节点指针node将位置index上的节点替换成节点node	int index, Examinee *node	/
void Statistics(string category)	public	通过传入的报考类别category作为关键字检索类中categorylist_（map类型）的值，并输出这个值	string category	/
void SearchOutput(int index)	public	通过传入的学号index检索对应的节点，输出此节点的信息	int index	/
void Output()	public	输出链表中所有节点的信息	/	/

链表类

```
//using Singleton Pattern
class ExamineeList {

//Singleton Pattern
public:
    static ExamineeList& Instance(){
        static ExamineeList instance;
        return instance;
    }

private:
    //initialize the head and the tail
    ExamineeList() {
        head_->prior_ = NULL;
        head_->next_ = tail_;

        tail_->prior_ = head_;
        tail_->next_ = NULL;

    }
    //delete the linked list
    ~ExamineeList() {

        auto current = head_;
        Examinee *following = current;

        while (following != tail_) {
            following = current->next_;
            delete current;
            current = following;
        }
        delete tail_;
    }
    //Singleton Pattern

public:
    // return and set total_
    int total() const { return total_; }
    void set_total(int total) { total_ = total; }

    //index with position
    Examinee *IndexPosition(int index);
    //index with number
    Examinee *IndexNum(int index);

    //required operation
```

```
    void Insert(int index, Examinee *node);
    void Delete(int index);
    void Modify(int index, Examinee *node);
    void Statistics(string category);
    //print the searched student's information
    void SearchOutput(int index);
    //print the list
    void Output();

private:
    static Examinee *head_;
    static Examinee *tail_;
    static int total_;//the total number
    map<string, int> categorylist_;
};
```