

数据结构课程设计

——考试报名系统

项目说明文档

作 者 姓 名： 韦世贸

学 号： 2351131

指 导 老 师： 软件学院

专 业： 软件工程

同济大学

Tongji University

1. 介绍
   1. 背景介绍

考试报名系统是一个学校不可缺少的部分，它对于学校的管理者和学生来说都至关重要，随着学生数量和考试数量的日益庞大，如何管理如此庞大的数据显得极为复杂，传统的手工管理工作量大且容易出错。

使用计算机对考试报名系统进行管理，具有手工管理所无法比拟的优势。这些优点能够极大地提高学校和学生的效率，也是学校走向信息化、科学化、国际化的重要条件。因此，开发一套考试报名系统具有十分重要的意义。

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了：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、删除考生信息等操作。

* 1. 功能介绍

作为一个最简易的考试报名系统，首先应该有的功能就是输入同学们的考试报名情况。其次，考试报名系统还应该具有插入、删除、修改功能，以保证同学可以随时更改自己的考试报名情况。最后，考试报名系统软件还应该确保软件可以正常关闭。

该报名系统也具有以下输入、输出、插入、删除、修改、退出的功能。

2 设计

2.1 数据结构设计

该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。因为涉及到大量的删除和插入操作运用数组存储显然功能性较差，并且考生的人数也是不定的，于是考虑采用链表进行存储，建立有头结点的链表，并在表尾也加入一个空结点，方便进行插入操作。

2.2 类结构设计

在该项目中创建一个节点类和一个链表类，将链表类定义为节点类的友元类，从而建立起一个链表，因为该系统数据类型确定，无使用抽象数据类型的必要故直接定义了类型。

2.3成员与操作设计

使用了自己写的string 类来替代c++库中的string

class MyString

{

public:

MyString(const char\* cstr = 0); //默认构造函数

MyString(const MyString& str); //拷贝构造函数

~MyString() {}

MyString& operator=(const MyString& str); //拷贝赋值运算符重载

MyString(MyString&& str); //移动构造函数

MyString& operator=(MyString&& str); //移动赋值

char\* get\_c\_str()const

{

return m\_data;

}

size\_t get\_length()const

{

return strlen(get\_c\_str());

}

friend ostream& operator<<(ostream& os, const MyString& str); //重载流输出运算符

friend istream& operator>>(istream& is, MyString& str); //重载流输入运算符

private:

char\* m\_data;

};

链表结点类（LinkNode）和信息类（Information）

信息类包括如下私有成员：

int No; //考号

MyString Name; //考生姓名

MyString Sex; //考生性别

int Age; //考生年龄

MyString ApplyCategory; //考生报考项目

void input() //用于输入信息

{

cin >> No;

cin >> Name;

cin >> Sex;

cin >> Age;

cin >> ApplyCategory;

return;

}

链表结点类包含信息类和link指针

链表类（Informationlist）

私有数据成员：

链表头结点first

公有数据操作：

Informationlist() //构造函数

{

first = new LinkNode(); //附加头结点

}

~Informationlist() //析构函数

{

;

}

void CreateNewList(int n) //建立新链表（后插法）

bool Insert(int location) //结点插入函数,插入后的元素成为链表中的第location个元素

bool Remove(int NO, int& location) //结点删除函数，删除考号为no的元素

void output() //按逻辑输出链表的元素

void Search(int NO) //查找元素

3 实现

3.1 插入功能的实现

3.1.1 插入功能流程简介

插入功能先判定输入考生的考号是否输入，如果输入不符合要求则进行驳回重新输入，符合要求后再判断该考号是否与已有考号重复如果重复依然驳回使其重新输入

3.1.2 插入功能核心代码

LinkNode\* current = first->link; //从第一个结点开始检测

for (int i = 1; i < location - 1; i++) { //找到目标位置的前一个结点

if (current == NULL)

break;

else

current = current->link;

}

if (current == NULL) {

cout << "无效的插入位置" << endl; //非空表但链太短

return false;

}

else {

LinkNode\* newNode = new LinkNode();

newNode->stuInformation.input();

if (newNode == NULL) {

cout << "存储分配错误！" << endl;

return false;

}

newNode->link = current->link;

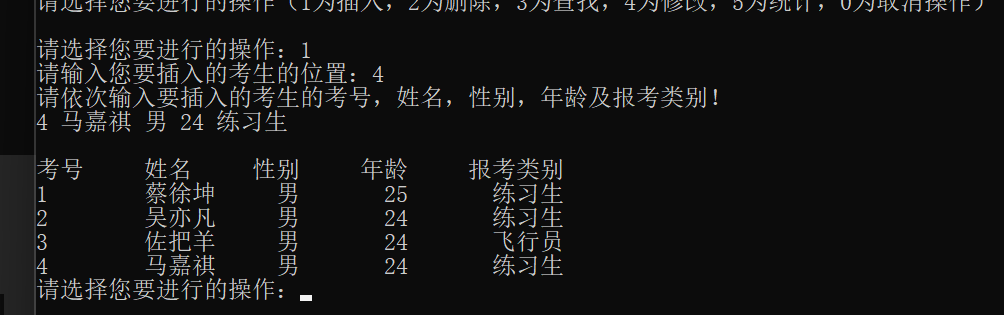
current->link = newNode;

return true;

}

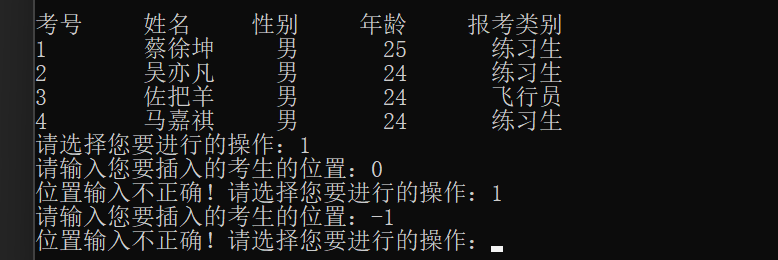
3.1.3插入功能截屏示例

正常的插入功能展示，未涉及边界情况。



3.1.4 插入功能测试以及边界测试

插入学号为负或者为0的同学时报错提示。



3.2 删除功能的实现

3.2.1 删除功能流程简介

删除功能首先进行判定输入的考号是否有效，如果有效，再进行是否有该考生的考号若无则请求重新查证后输入，如果有则进行删除操作。

3.2.2 删除功能核心代码

LinkNode\* del, \* current = first;

while (current->link != NULL) {

if (current->link->stuInformation.No == NO) {

cout << "你删除的考生信息是："; //打印提示

cout << current->link->stuInformation.No << " " << current->link->stuInformation.Name << " "

<< current->link->stuInformation.Sex << " " << current->link->stuInformation.Age << " "

<< current->link->stuInformation.ApplyCategory << endl;

break;

}

else

{

current = current->link;

location++; //同时能够找到结点在链表的逻辑位置

}

}

if (current == NULL || current->link == NULL) { //删除不成功

cout << "无效的删除位置！" << endl;

return false;

}

del = current->link;

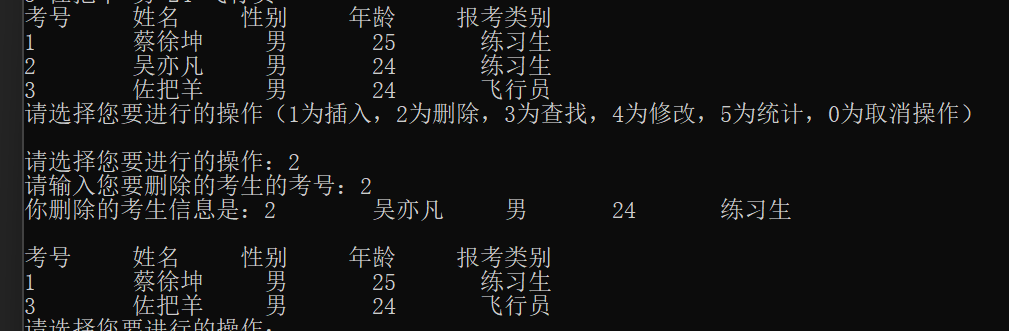
current->link = del->link;

delete del;

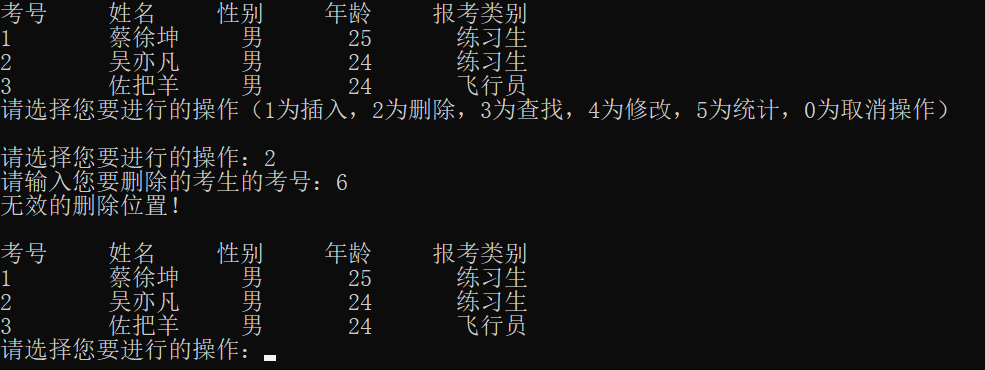
return true;

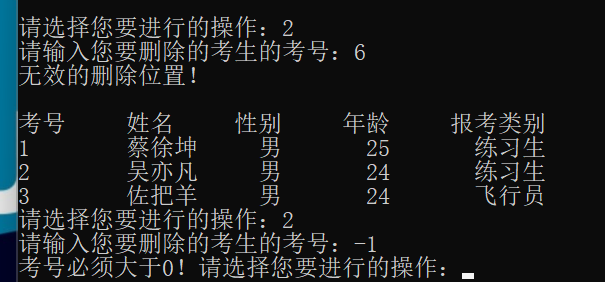
3.2.3 删除功能截屏示例

正常插入时代码展示，未涉及边界情况，涉及边界情况将在后一小节给出运行情况。



3.2.4 删除功能测试以及边界测试





3.3 查找功能的实现

3.3.1 查找功能流程简介

查找功能首先进行判定输入的考号是否有效，如果有效，再判定是否有该考生的考号若无则请求重新查证后输入，如果有则进行输出该考生信息操作。

3.3.2 查找功能核心代码

LinkNode\* current = first->link;

while (current != NULL) {

if (current->stuInformation.No == NO) {

cout << Tips << endl; //打印信息类别

cout << current->stuInformation.No << " " << current->stuInformation.Name << " "

<< current->stuInformation.Sex << " " << current->stuInformation.Age << " "

<< current->stuInformation.ApplyCategory << endl;

return;

}

else

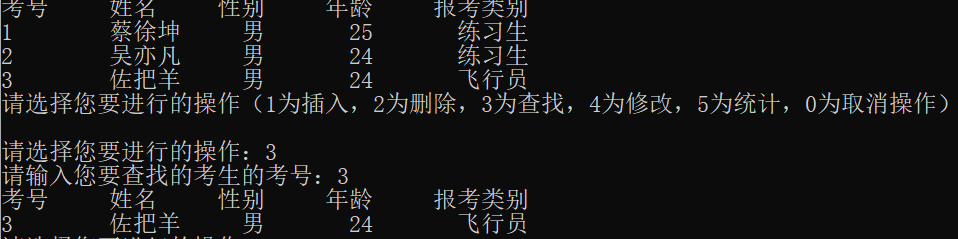
current = current->link;

}

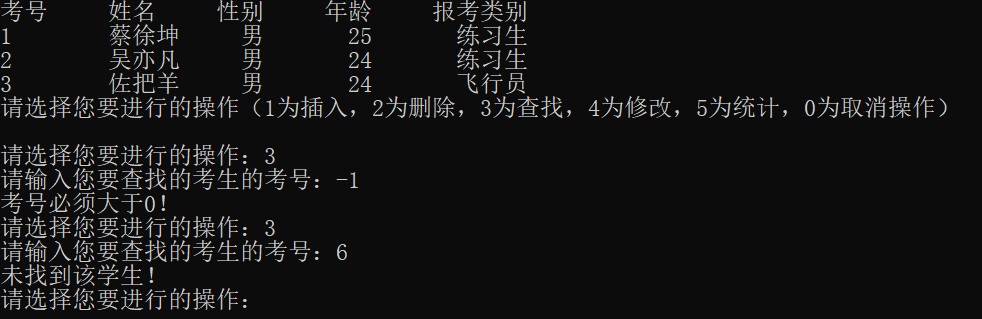
cout << "未找到该学生！" << endl;

return;

3.3.3 查找功能截图示例



3.3.4 查找功能测试以及边界测试



3.4 修改功能的实现

3.4.1 修改功能流程介绍

修改功能首先进行判定输入的考号是否有效，如果有效，再进行是否有该考生的考号若无则请求重新查证后输入，如果有则请求输入修改后的考生信息进行修改后输出。

3.4.2 修改功能核心代码

LinkNode\* current = first;

while (current != NULL) {

if (current->stuInformation.No == NO) { //找到了目标考生

cout << "请输入要修改的考生信息："; //输入提示

cin >> current->stuInformation.No >> current->stuInformation.Name >>

current->stuInformation.Sex >> current->stuInformation.Age >>

current->stuInformation.ApplyCategory ;

return true;

}

else

{

current = current->link;

}

}

if (current == NULL || current->link == NULL) { //修改不成功

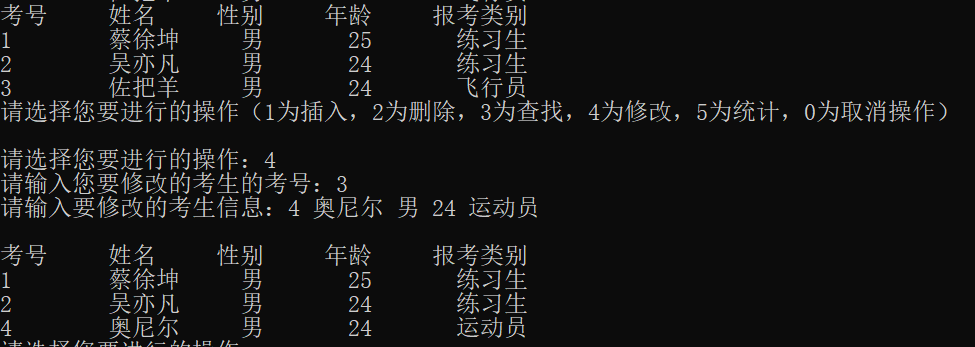
cout << "无效的修改位置！" << endl;

return false;

}

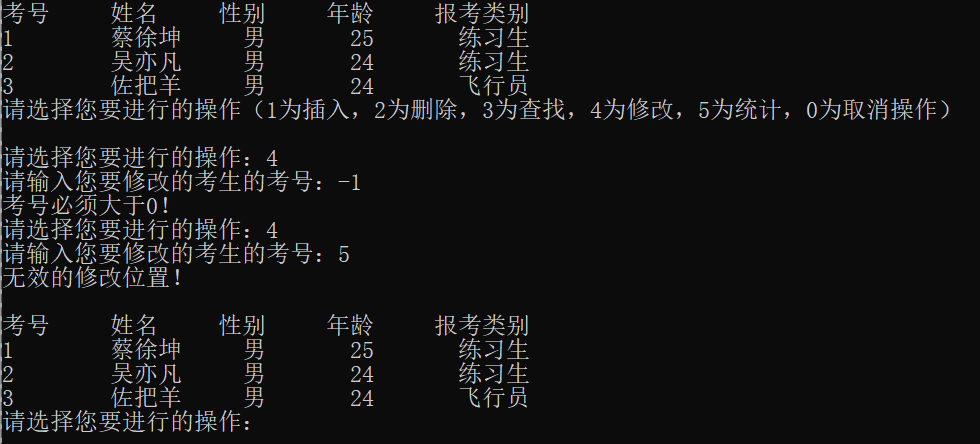
return true;

3.4.3 修改功能截屏示例



3.4.4 修改功能测试以及边界测试

若考生考号为负，则输出提示，若考号不存在，则输出提示并重新打印一遍考生信息，让用户重新查找要修改的考生的考号



3.5 统计功能的实现

3.5.1 关于统计系统功能的实现说明

因为在每次运行完操作后均会自动输出出所有考生信息，所以这一块并未进行特殊设计。

3.5.2 统计功能核心代码

LinkNode\* current = first->link;

while (current != NULL) {

cout << current->stuInformation.No << " " << current->stuInformation.Name << " "

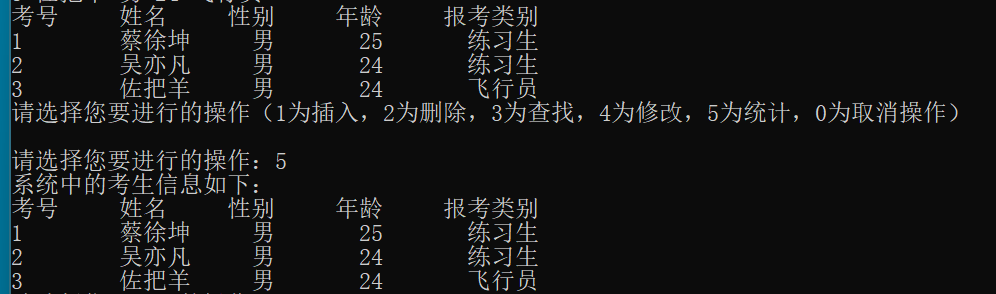
<< current->stuInformation.Sex << " " << current->stuInformation.Age << " "

<< current->stuInformation.ApplyCategory << endl;

current = current->link;

}

3.5.3 统计功能截屏示例



3.6 建立初始链表功能的实现

3.6.1 建立初始链表功能的简介

建立初始链表时先判定输入的考生人数是否合法，如不合法则让其重新输入，判定合法后进行输入考生信息，链表记录信息。

3.6.2 建立初始链表功能核心代码

LinkNode\* newNode, \* last;

last = first;

for (int i = 1; i <= n; i++) { //按顺序输入元素

newNode = new LinkNode();

newNode->stuInformation.input();

if (newNode == NULL) {

cout << "存储分配错误！" << endl;

return;

}

last->link = newNode; //尾指针指向新结点，连接起新结点

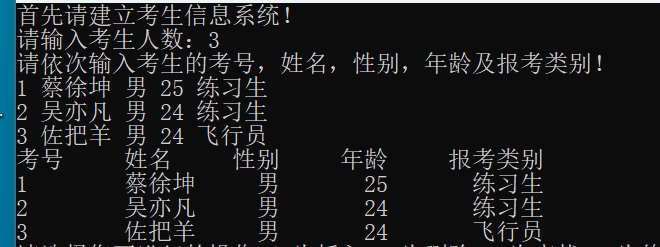
last = newNode; //更新尾指针的位置

}

last->link = NULL; //链表结尾

return;

3.6.3 建立初始链表截屏示例



3.7总体系统核心代码

Informationlist setSystem() //建立考生信息系统

{

int num;

cout << "首先请建立考生信息系统！" << endl;

cout << "请输入考生人数：";

while(1) //输入错误处理

{

cin >> num;

if (num <= 0 || cin.fail())

{

cin.clear();

cin.ignore(65472, '\n'); //清空缓冲区

cout << "考生人数必须大于0！" << endl;

}

else

break;else

break; }

cout << "请依次输入考生的考号，姓名，性别，年龄及报考类别！" << endl;

Informationlist inforlist;

inforlist.CreateNewList(num); //建立新链表

printInformation(inforlist);

cout << "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）" << endl;

cout << endl;

return inforlist;

}

int location = 0, no = 0;

switch (op)

{

case 0:

cout << "操作结束！" << endl;

cout << "已退出系统！" << endl;

return;

case 1:

cout << "请输入您要插入的考生的位置：";

cin >> location;

if (location <= 0)

cout << "位置输入不正确！";

else {

cout << "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！"<< endl;

inforlist.Insert(location);

cout << endl;

printInformation(inforlist);

}

break;

case 2:

cout << "请输入您要删除的考生的考号：";

cin >> no;

if (no <= 0)

cout << "考号必须大于0！" << endl;

else {

inforlist.Remove(no,location);

cout << endl;

printInformation(inforlist);

}

break;

case 3:

cout << "请输入您要查找的考生的考号：";

cin >> no;

if (no <= 0)

cout << "考号必须大于0！" << endl;

else {

inforlist.Search(no);

}

break;

case 4:

cout << "请输入您要修改的考生的考号：";

cin >> no;

if (no <= 0)

cout << "考号必须大于0！" << endl;

else {

inforlist.Change(no); //修改考生信息

cout << endl;

printInformation(inforlist);

}

break;

case 5: //统计

cout << "系统中的考生信息如下：" << endl;

printInformation(inforlist);

break;

default:

cout << "输入的选项不存在！" << endl;

break;

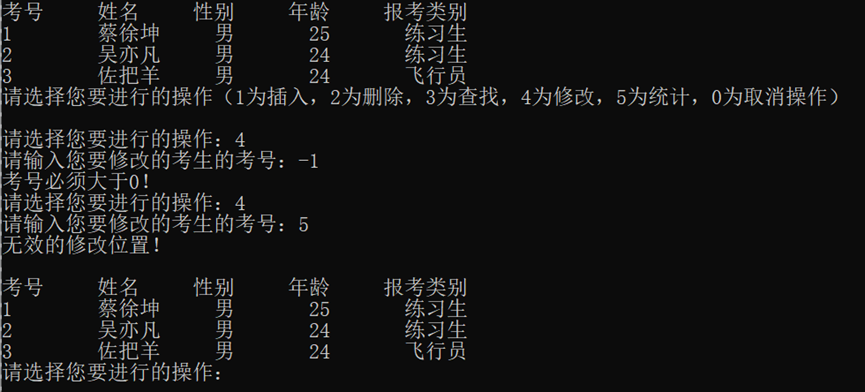
}

4 .设计小结

在这次考试系统的实现中我掌握了用一个结构体封装结点，用另一个类建立链表，实现数据的完全封装，比以前的使用结构体来说有明显的优势，主要收获是能够自己建立链表类，并熟悉了链表的各种操作。

4.1主要改进

我对每一项操作都进行了输入错误处理，并在修改考生报错时再次打印一遍考生信息，有助于用户寻找目标考生



在这次设计中，比较遗憾的地方是对于统计功能，我没能做到分类统计，即按照性别，年龄等信息进行统计。