****

**Tongji University**

**数据结构课程设计实验报告**

**001-考试报名系统**

专 业： 软 件 工 程

指导教师： 张 颖

学 号： 2 1 5 3 0 6 1

姓 名： 谢 嘉 麒

1. **项目及基本要求概述**
   1. **项目概览**
      1. **项目综述**

本次项目要求对考试报名管理的进行模拟，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。

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。

* 1. **项目各模块分析**

1. 根据项目要求，因其有较多的增删查改工作，故确定使用链表作为本项目的数据结构，并用类方式对类成员及成员函数进行封装。
2. 程序首先需要提示客户输入初始考生人数，其后根据这一数据顺次建立单链表，每个链表结点包括考号、姓名、性别、年龄和报考类别五个子数。客户完成键入后，系统打印链表内容。
3. 系统提示客户选择将要执行的操作“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”，并在得到回复后跳转到相应执行函数中（此处使用switch-case语句）
4. 为进一步操作设定Insert、Remove、Search、Modify、Summary（分别对应插入，删除，查找，修改，统计）五个函数，并实现对应操作的执行。当客户输入“0”时，取消操作，跳转回（2），重新询问客户要求。
5. 在每次执行操作后打印链表的当前内容，并继续询问客户进行操作，循环执行以上内容。
6. **整体设计思路**

本项目重点采用链表方式的增删查改以实现各种功能，并在此基础上使用类进行封装，附加main函数程序进行功能验证，形成一套完整的考试报名系统模型。

* 1. **数据结构的设计与使用**

本次项目主要可以选择顺序表的数据结构方法，而考虑到客户对于数据的增删有较多要求，故而选择单链表的方式对数据进行存储，保证了数据增删的效率，同时也可以优化内存环境，确保内存有足够的空间存入考生信息。

* 1. **链表类及链表结点类的定义**

1. 定义链表结点类（examRegSys\_linkNode）中包含私有的（private）考号、姓名、性别、年龄和报考类别几个数据，并添加结点指针；在结点类的公共部分（public）定义结点类的构造/析构函数。
2. 为了方便访问，将链表类声明为友元类；同时为了优化输入输出，在结点类的公共部分将输入输出运算符的重载函数定义为友元函数。
3. 定义链表类（examRegSys\_list）中包含私有的（private）链表表头指针和总报考人数sum；在公共部分（public）定义构造/析构函数以及各种功能函数。
4. 链表类成员函数详细内容见下：

examRegSys\_list(); //链表类构造函数

~examRegSys\_list(); //链表类析构函数

void setSum(); //设定总人数

void makeEmpty(); //置空链表

examRegSys\_linkNode\* locate(int i); //定位链表中某一结点

void outputAll(); //打印链表所有结点

void outputSingle(examRegSys\_linkNode\* x);//根据地址输出某单个结点

void inputRear(); //后插法建立单链表

void choose(); //确定用户输入的操作要求

bool Insert(int i); //将新元素插入到链表中第i个结点后

bool Remove(int i); //将链表中第i个元素删去

examRegSys\_linkNode\* Search(int i); //在链表中查找含有数据i的结点examRegSys\_linkNode\* Modify(int i); //在链表中查找含有数据i的结点并对其重新输入进行修改

void Summary(char statistic); //根据用户需求统计性别，年龄或报考类别

* 1. **输入输出运算符的重载**

本次项目为了优化代码，简化项目内容，采用了重载输入输出运算符的方式，并将其声明为链表结点类的友元函数，其声明如下：

1. friend std::istream& operator>>(std::istream& istr, examRegSys\_linkNode& x);
2. friend std::ostream& operator<<(std::ostream& ostr, const examRegSys\_linkNode& x);

使用这两个函数完成类内数据的一次性输入与输出。

* 1. **对于客户输入内容正误的判断**

出于对代码健壮性的考虑，本项目对用户输入的内容进行了有效判定，主要分为以下两个部分：

1. 非逻辑错误的输入判断：对于用户输入的非逻辑错误，如输入总人数为“a”“#”等，输入操作不为0-5等，采用while循环结合cin.clear/cin.ignore函数进行处理，清空输入缓冲区后提示客户重新输入。
2. 逻辑错误的输入判断：对于用户输入的逻辑错误，如插入的位置被孤立，删除的位置没有数据或修改的数据不存在等错误，采用分别处理的方法，在各自的功能函数中进行分开处理，错误则返回不正确的提示值，保证代码健壮性。
   1. **动态内存分配的处理**

考虑到顺序表直接分配内存时存在两个问题：一是难以确定报名考生的最大数量以及后续增删时添加的单位数；二是直接进行分配容易导致内存的占用和浪费。

因此本次项目决定采用动态内存申请来进行，保证内存的利用效率，提高开辟内存的成功率，以适应更多的工作环境。

1. **主要功能的实现**
   1. **Main函数中类成员函数的调用流程图**

.setSum函数

建立单链表

后插法

.inputRear函数

建立类

main函数

.choose函数

.outputAll函数

后续操作函数在.choose函数中进行

* 1. .**choose函数中各操作成员函数的调用流程图**

1、插入信息

.Insert函数

2、删除信息

.Remove函数

3、查找信息

.Search函数

.choose函数

4、修改信息

.Modify函数

a.统计性别

.Modify函数

b.统计年龄

5、统计信息

c.统计报考类别

操作函数依靠.choose函数进行循环调用，完成增加信息，删除信息，查找信息（按照考号），修改信息和统计信息（分别可以选择统计性别、年龄和报考类别三种信息）

* 1. **后插法建立单链表的实现**

为了简化单链表增删等函数的代码，本项目建立链表中包含有附加头结点。为了和客户书写习惯相匹配，故采用后插法建立单链表。具体操作如下：

1. 声明结点类指针newNode,last和结点类数据node；建立前调用makeEmpty函数清空当前链表。
2. 让last指针指向链表头结点，进入循环，当循环计数器小于报考总人数时，输入node对象，并利用动态内存申请为其分配空间。
3. 根据当前建立过程不断更新last指针，使其last->link部分串接起新数据，而后last指向新数据位置。
4. 循环结束后last->link指向空指针。自此，后插法建立单链表完成。
   1. **确定客户输入内容的choose函数**

利用链表类成员函数choose确认客户输入的内容，在choose函数中调用switch-case语句进行跳转，输入错误处理后进入各个操作函数。

* 1. **利用定位函数locate定位到链表中某一结点**

考虑到功能函数大多需要对于某个特定结点进行操作，为了实现代码复用，故建立locate函数对结点进行定位。返回某个结点的内存地址。

* 1. **统计函数的功能设计**

本次项目的统计操作并未明确给出客户需求，考虑到可统计的数据共有性别、年龄和报考类别三种（考号不允许相同，统计相同姓名并无太大需要），故为统计函数设定三种功能区划，使得其能按需统计不同子数据。

1. **调试过程碰到的问题**
   1. **输入输出流的重载**

通过本次项目，笔者发现对于输入输出流的重载，如果按照传统重载方式而不考虑返回值的特殊性的话，会导致无法连续输入/输出（比如cin>>a>>b，cout<<a<<b）。

而将重载函数的返回值设定为istream&/ostream&后，由于返回的是参数&istr/&ostr（即cin/cout），则可以满足连续输入/输出的效果。

同时，笔者发现输入输出流的重载不能作为类成员函数进行声明，必须要声明为类的友元函数，并进行定义。

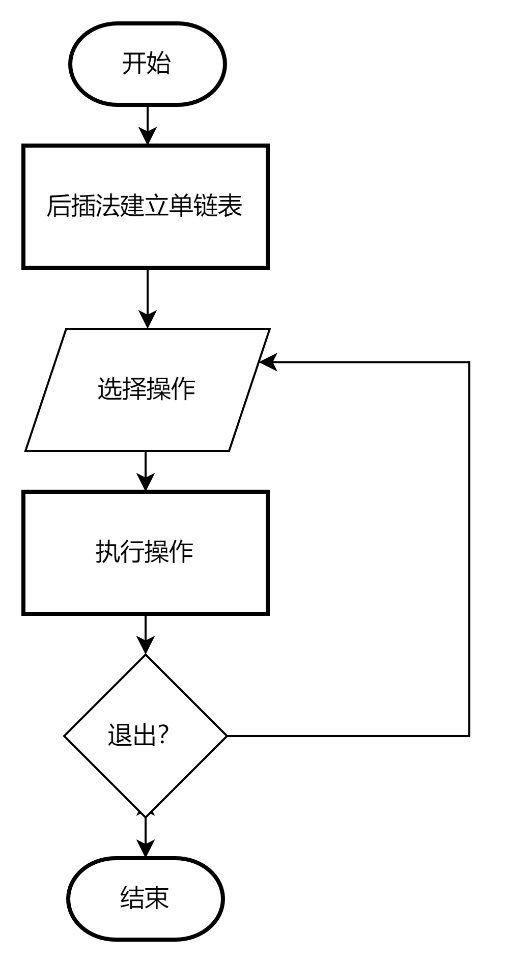
* 1. **类成员数据之间的比较**

通过本次项目，笔者发现相同类定义下的不同数据之间不能进行直接比较，比如结点类定义A,B后，不能直接通过A==B来进行比较。在本次项目中，为了进行数据的拷贝及比较，笔者自主编写了My\_strlen、My\_strcpy、My\_strcmp三个函数，代替string类函数实现功能，保证程序的正确性。

* 1. **对于代码健壮性的探索和优化**

本次项目笔者重点落实到代码健壮性问题上，并探索优化各种问题：

1. 对于客户输入如123a等错误，单纯判误并重复输入会导致后续输入内容出现问题，因为a将会存储在环境的输入缓冲区中。为了解决这个问题，笔者通过搜寻资料并自主探索，发现利用cin.clear和cin.ignore函数进行输入流的截断和缓冲区的清空，结合while循环进行错误处理，有效优化代码健壮性。
2. 对于增加信息位置不对，或者删除信息位置不存在等问题，需要在各个操作函数中进行分别判断，错误则返回FALSE或者空指针，保证代码健壮性。
3. **程序整体流程图**

****

是

否

1. **心得体会**
   1. **对于顺序表/链表数据结构的理解**
2. 通过本次项目，笔者认识到顺序表中数据结构的优缺点和选择的优先级，并根据判断选择链表进行项目操作。
3. 本次项目使笔者更加熟练掌握链表的实现和代码操作，加深了对附加头结点，链表类和结点类等概念的运用。
4. 在输入输出处理方面，笔者通过不断的试错和改正，结合多方面查询得到的知识，使得笔者能力得到很大提高。
   1. **对于程序项目设计的经验教训**
5. 笔者认识到函数划分不仅需要考虑到功能的区块和细节的划分，还不能过分地细节化函数，不然反而会弄巧成拙；
6. 变量的命名需要简洁明确，而变量的使用尽可能不要重复太多次，否则容易出现混淆等问题（在此处强调注释的重要性）
7. 在设计程序的时候要整体考虑，分块操作，这样在做每个模块的时候能保证思路的清晰与完整；同时需要考虑到模块之间的联系，以免出现之前的函数到后面兼容性 下降甚至不再适用的情况。
8. 尽量在完成每个功能的时候就做好相关程序的设计，或者为后续的修复与补充预留出足够的空间；在后续的功能添加中可能会引发连锁的bug，大大增加复盘工作量。
9. 在程序设计的过程中，设计好相应功能后最好能设置合理断点或者测试程序进行检查验证，否则功能累加太多后再查验 bug 会增加查验难度。
10. **程序测试**
    1. **后插法建立单链表测试**

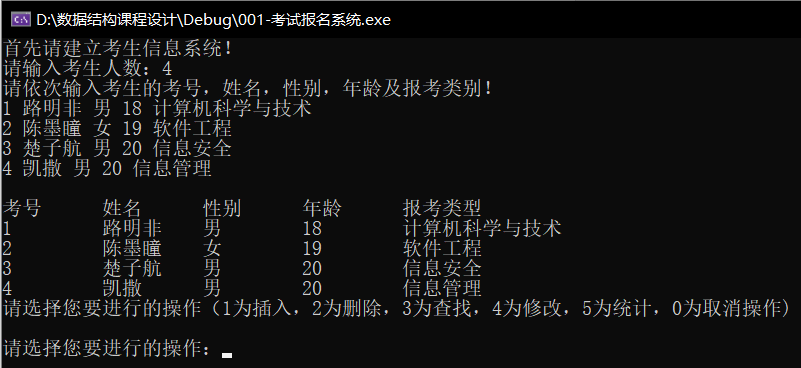
测试数据：

1 路明非 男 18 计算机科学与技术

2 陈墨瞳 女 19 软件工程

3 楚子航 男 20 信息安全

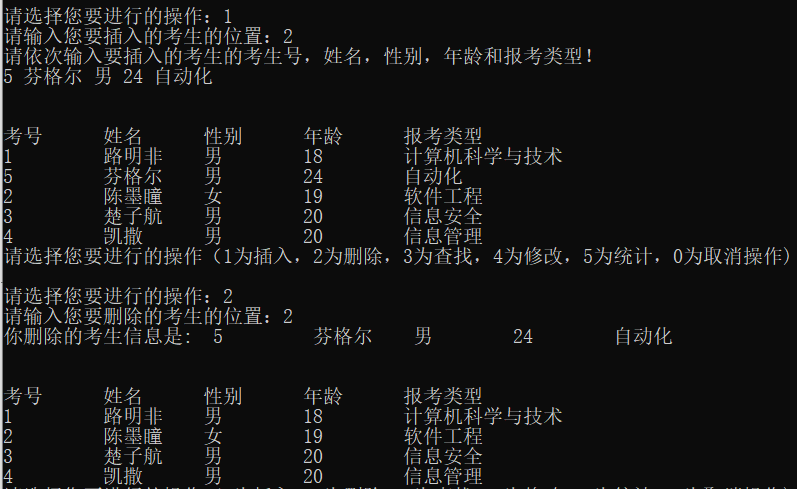
4 凯撒 男 20 信息管理



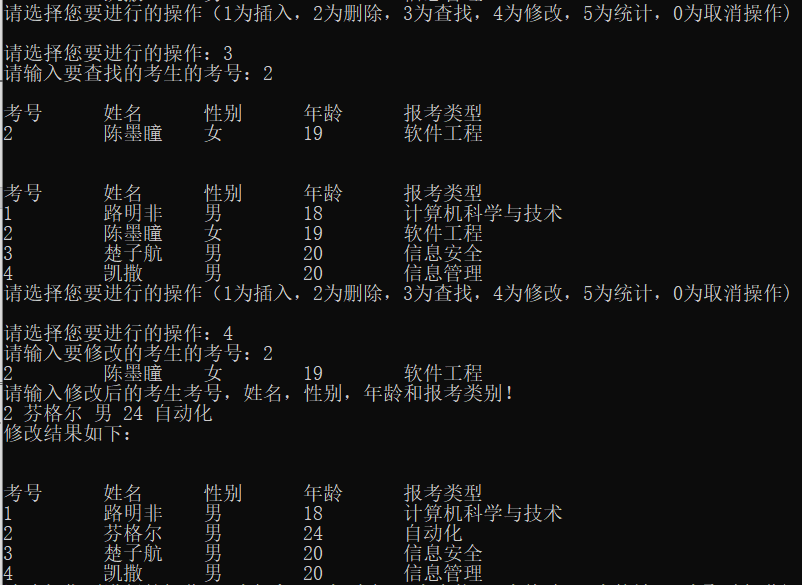
* 1. **插入与删除成员测试**

插入成员：5 芬格尔 男 24 自动化

删除成员考号：2

****

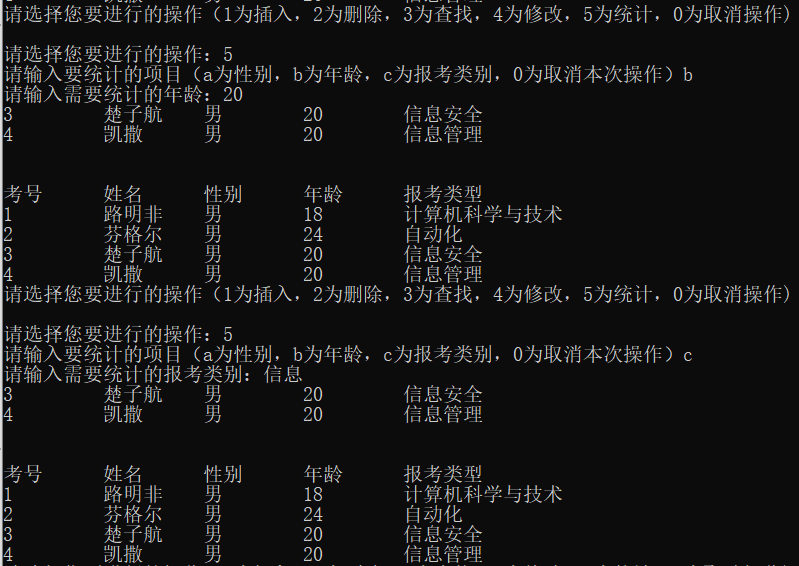
* 1. **查找与修改成员测试**

****

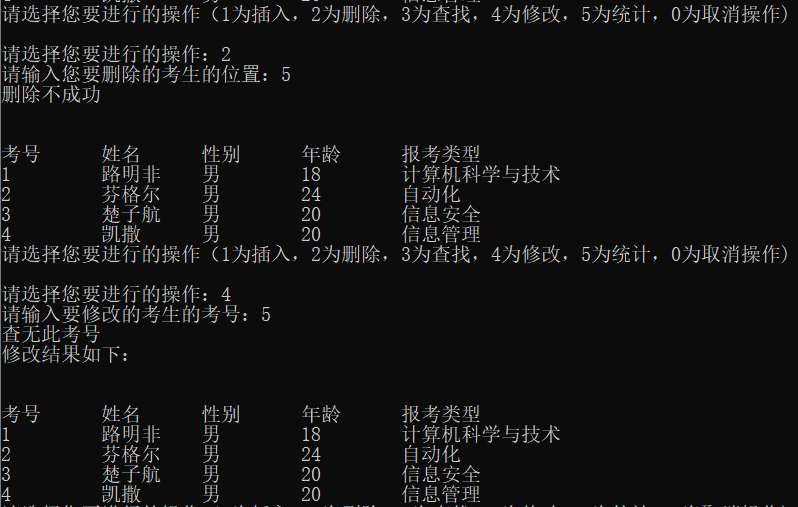
* 1. **统计成员测试**

测试1：统计相同年龄20；

测试2：统计报考类型中含关键字“信息”

****

* 1. **当删除和修改的考生不存在时提示“删除不成功”和考号不存在**

****