数据结构课程设计

项目说明文档

**考试报名系统**



同济大学

Tongji University

姓名： 林觉凯

学号： 2253744

指导老师： 张颖

学院专业： 软件学院 软件工程

**目录**

**1.项目分析-------------------------------------------------------------------------4**

**1.1 项目背景分析-------------------------------------------------------------------------4**

**1.2项目功能分析--------------------------------------------------------------------------4**

**1.2.1项目功能要求**-------------------------------------------------------------------------------**4**

**1.2.2项目输入要求-------------------------------------------------------------------------------4**

**1.2.3项目输出要求-------------------------------------------------------------------------------4**

**1.2.4项目示例-------------------------------------------------------------------------------------5**

**2.项目设计-------------------------------------------------------------------------6**

**2.1 数据结构设计-------------------------------------------------------------------------6**

**2.2类设计----------------------------------------------------------------------------------6**

**2.2.1链表结点类(ListNode)--------------------------------------------------------------------6**

**2.2.2链表类(LinkList)---------------------------------------------------------------------------7**

**2.3成员与操作设计----------------------------------------------------------------------7**

**2.4系统流程设计-------------------------------------------------------------------------8**

**3.项目实现-------------------------------------------------------------------------9**

**3.1 四个基本函数的实现----------------------------------------------------------------9**

**3.1.1 输出链表结点函数-------------------------------------------------------------------------9**

**3.1.2 输出链表内容函数-------------------------------------------------------------------------9**

**3.1.3 析构函数-------------------------------------------------------------------------------------9**

**3.1.4 Located函数——将考生所在链表位置转换为所在链表位置的地址----------10**

**3.2 建立考生信息库的实现------------------------------------------------------------11**

**3.2.1 建立考生信息库的流程图---------------------------------------------------------------11**

**3.2.3 建立考生信息库的核心代码------------------------------------------------------------12**

**3.2.3 建立考生信息库的功能截图------------------------------------------------------------14**

**3.3 插入考生信息的实现---------------------------------------------------------------15**

**3.3.1 插入考生信息的流程图-----------------------------------------------------------------15**

**3.3.2 插入考生信息的核心代码--------------------------------------------------------------16**

**3.3.3 插入考生信息的功能截图--------------------------------------------------------------16**

**3.4 删除考生信息的实现---------------------------------------------------------------17**

**3.4.1 删除考生信息的流程图------------------------------------------------------------------17**

**3.4.2 删除考生信息的核心代码---------------------------------------------------------------18**

**3.4.3 删除考生信息的功能截图---------------------------------------------------------------18**

**3.5 修改考生信息的实现---------------------------------------------------------------19**

**3.5.1 修改考生信息的流程图-----------------------------------------------------------------19**

**3.5.2 修改考生信息的核心代码--------------------------------------------------------------20**

**3.5.3 修改考生信息的功能截图--------------------------------------------------------------20**

**3.6 查找考生信息的实现---------------------------------------------------------------21**

**3.6.1 查找考生信息的流程图-----------------------------------------------------------------21**

**3.6.2 查找考生信息的核心代码--------------------------------------------------------------22**

**3.6.3 查找考生信息的功能截图--------------------------------------------------------------22**

**3.7 统计考生信息的实现---------------------------------------------------------------23**

**3.7.1 统计考生信息的流程图-----------------------------------------------------------------23**

**3.7.2 统计考生信息的核心代码--------------------------------------------------------------24**

**3.7.3统计考生信息的功能截图---------------------------------------------------------------24**

**3.8 主函数的实现------------------------------------------------------------------------25**

**4.项目代码功能测试------------------------------------------------------------28**

**4.1 建立考生信息库的测试------------------------------------------------------------28**

**4.1.1 建立考生信息库的功能测试------------------------------------------------------------28**

**4.1.2 建立考生信息库的代码健壮性测试----------------------------------------------28**

**4.2 插入考生信息的测试---------------------------------------------------------------29**

**4.2.1 插入考生信息的功能测试---------------------------------------------------------------29**

**4.2.2 插入考生信息的代码健壮性测试-------------------------------------------------29**

**4.3 删除考生信息的测试---------------------------------------------------------------30**

**4.3.1 删除考生信息的功能测试---------------------------------------------------------------31**

**4.3.2 删除考生信息的代码健壮性测试-------------------------------------------------31**

**4.4 修改考生信息的测试---------------------------------------------------------------31**

**4.4.1 修改考生信息的功能测试---------------------------------------------------------------31**

**4.4.2 修改考生信息的代码健壮性测试-------------------------------------------------32**

**4.5 查找考生信息的测试---------------------------------------------------------------33**

**4.5.1 查找考生信息的功能测试---------------------------------------------------------------33**

**4.5.2 查找考生信息的代码健壮性测试-------------------------------------------------33**

**4.6 统计考生信息的测试---------------------------------------------------------------34**

**4.7 其他基本输入的健壮性测试------------------------------------------------------34**

**4.7.1 建立考试报名系统输入人数数字的健壮性-----------------------------------------34**

**4.7.2 插入操作时输入位置、删除、修改、查找操作时输入考号数字的健壮性---35**

**4.7.3 在main函数中选择操作数的健壮性-------------------------------------------------35**

**5.项目心得与体会---------------------------------------------------------------35**

**1.项目分析**

**1.1 项目背景分析**

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，所以一个良好的考试报名系统应该能够为用户提供充足的信息和功能。考试报名系统对于学校加强考试管理有极其重要的作用。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。

随着计算机科学技术的不断成熟，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。

**1.2项目功能分析**

**1.2.1项目功能要求**

本项目是对考试报名系统的简单模拟，用控制台选项的选择方式完成以下功能：建立考试信息系统；输入考生信息；输出考生信息；插入考生信息；删除考生信息；修改考生信息；查询考生信息；统计考生人数等等。

同时，该项目也对于代码的健壮性上下了很大的功夫，比如在输入考生的一系列信息的同时，都有对基本的五个考生信息做一定的判断；在修改考生的考号的同时，也对已存在的考号进行的比对，保证修改考号不重复等等。

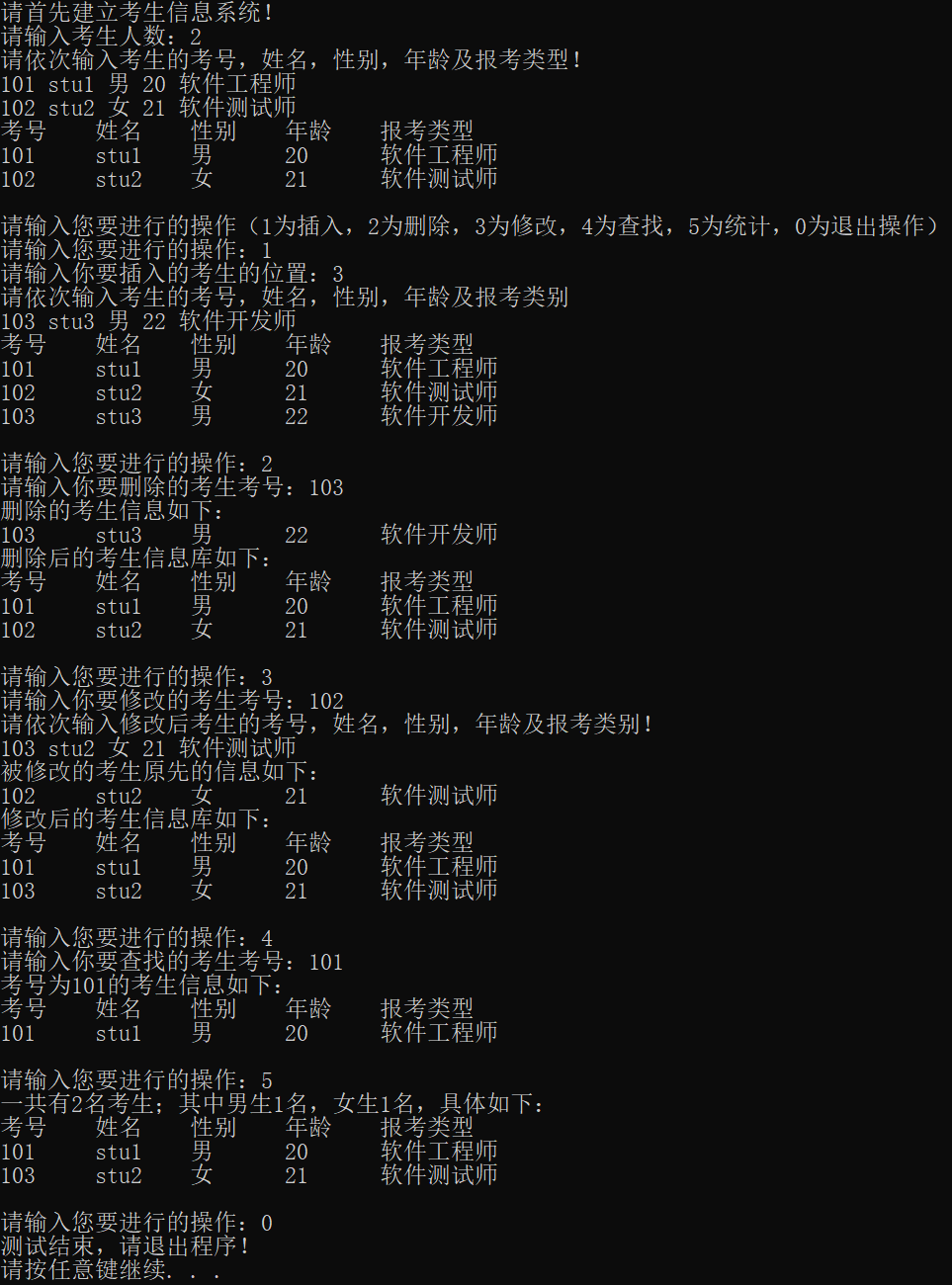
**1.2.2项目输入要求**

每次操作都输入相对应的操作数字，同时输入正确的考生信息。（包括考号、姓名、性别、年龄和报考类型这五个基本的考生信息）

**1.2.3项目输出要求**

输出建立后的考试信息库中的考生信息，进行一系列操作后（插入、删除、修改）的考试信息库中的考生信息，查询某一学号所对应的考生的基本信息，统计考生人数和男生女生分别人数的考生信息等等。

**1.2.4项目示例**

****

**2.项目设计**

**2.1 数据结构设计**

该考试报名系统要求快速地完成考生信息的插入、删除、修改等等操作，因此要选对合适的数据结构来进行编写程序。本程序采用了带头结点的单链表作为最基本的数据结构。带头结点的单链表具有以下一系列优点：

\*链表是一个动态的数据结构，可以在运行时根据用户的需要来分配内存从而实现链表长度的增加或者缩短，无需初始链表长度；

\*链表结点的插入、删除、修改等等操作比较方便。在进行这一系列操作的时候，我们只需通过相应函数找到要操作链表节点的地址即可，然后对其进行适当的操作，无需移动其他元素，这样也使操作的时间复杂度减小为O(n)；

\*链表可以在运行的时候可以根据用户的需要自动地增加和缩短长度，利用计算机里零散的内存进行分配，这样可以减少大量内存的浪费。

我们在单链表的开始处再加一个头结点，这样也使操作定位时更加简便。

**2.2类设计**

经典的链表一般包括两个抽象数据类型(ADT)——链表结点类(ListNode)与链表类(LinkList)，而两个类之间的耦合关系可以采用嵌套、继承等多种关系。本程序也构造了这两个基本的链表类来进行操作。

**2.2.1链表结点类**(ListNode)

\*每一个链表结点储存着一名考生的信息和下一名考生信息的地址

class ListNode//定义链表结点类(ListNode)，每一个结点储存着一名考生信息和下一名考生信息的地址

{

public:

int student\_id; //考号

int student\_age; //年龄

string student\_name; //姓名

string student\_sex; //性别

string student\_category; //报考类型

ListNode\* next; //指向下一个结点的指针

ListNode(int number1 = 0, int number2 = 0, string str1 = "", string str2 = "", string str3 = "", ListNode\* point = NULL)

{

student\_id = number1;

student\_age = number2;

student\_name = str1;

student\_sex = str2;

student\_category = str3;

next = NULL;

}/ /构造函数，初始化考生的信息

void ShowNode(); //公有操作：输出结点考生的基本信息

};

**2.2.2链表类**(LinkList)

class LinkList

{

public:

LinkList(); //构造函数，建立考试报名信息库

~LinkList(); //析构函数，实现释放内存

//计算链表长度（即考试报名系统总人数，同时分别统计男生、女生人数）

void Length();

//依次输出链表内的每个结点的信息

void ShowList();

//寻找函数，输入一个考生的考号，返回这位考生在链表中的位置（第几个）

int Search(int id);

//定位函数，输入这位考生再链表中的位置，返回这位考生所在位置的地址

ListNode\* Located(int Location);

//插入函数，输入要插入考生的位置(通过Search函数获得)和基本信息，返回插入成功或失败

bool Insert(int Location,ListNode& node);

//修改函数，输入要修改考生的位置(通过Search函数获得)，返回修改成功或失败，node用于记录修改前考生的基本信息

bool Change(int Location,ListNode& anode, ListNode& bnode);

//删除函数，输入要删除考生的位置(通过Search函数获得)，返回插入成功或失败，anode用于记录修改后的考生的基本信息，bnode用于记录删除的考生的基本信息

bool Delete(int Location,ListNode& node);

private:

ListNode\* head; //私有成员：链表的头结点

};

**2.3成员与操作设计**

链表结点类的公有成员为考生的考号、姓名、性别、年龄和报考类型，公有操作为输出该结点的基本信息；链表类的公有操作为计算链表长度、输出链表内每一个结点的信息、寻找函数、定位函数、插入函数、修改函数和删除函数，私有成员为链表的头结点。

用户可以通过控制台上显示的操作数字，进行选择操作，然后通过输入相应的信息和指令，完成相应的操作。

**2.4系统流程设计**

系统操作的整体流程大致如下：

**程序开始运行**

**建立考生信息库**

**（输入人数和信息）**

**选择操作**

**（插入、删除、修改查找、统计和退出）**

**退出程序**

**执行操作**

**显示操作后数据**

**3.项目实现**

**3.1 四个基本函数的实现**

**3.1.1 输出链表结点函数**

//输出某个结点储存的考生基本信息函数

void ListNode::ShowNode()

{

cout << student\_id << "\t";

cout << student\_name << "\t";

cout << student\_sex << "\t";

cout << student\_age << "\t";

cout << student\_category << endl;

}

**3.1.2 输出链表内容函数**

//依次输出链表内每个结点的信息

void LinkList::ShowList()

{

cout << "考号\t" << "姓名\t" << "性别\t" << "年龄\t" << "报考类型" << endl;

ListNode\* current = head->next;

while (current != NULL)

{

cout << current->student\_id << "\t" << current->student\_name << "\t" << current->student\_sex << "\t" << current->student\_age << "\t" << current->student\_category << endl;

current = current->next;

}

}

**3.1.3 析构函数**

//析构函数，实现释放内存

LinkList::~LinkList()

{

ListNode\* current = head->next;

while (head->next != NULL)

{

current = head->next;

head->next = current->next;

delete current;

}

delete head;

析构函数，从头结点开始遍历链表，在遍历过程中释放经过节点空间。最后再释放头结点的空间，完成链表的销毁。

**3.1.4 Located函数——将考生所在链表位置转换为所在链表位置的地址**

//定位函数，输入这位考生在链表中的位置，返回这位考生所在位置的地址

ListNode\* LinkList::Located(int Location) //上一个Search函数可以找到考生是在链表第几位，Located函数返回这一位的地址

{

if (Location < 0)

{

return NULL; //代码的健壮性，如果Location为负数说明该位置不存在，return NULL

}

else

{

ListNode\* current = head;

int counter = 0;

while (current != NULL && counter < Location)

{

current = current->next;

counter++;

}

return current;

}

}

Located函数是本程序中一个很重要的函数，它在各个操作中几乎都有被运用到。因为我们插入信息需要知道插入点在链表中的位置，查找、删除、和修改相应考号的考生需要通过Search函数把学号转化为该学号考生所在链表中的第几位，然后才能定位对其进行相应的操作。Located函数主要是将考生所在链表中的“第几位”转化为这一位的链表结点所对应的地址。有了指向这一地址的指针，我们后续才好进行相对应的一系列操作。

**学生学号(id)**

**Search函数**

**所在链表位置**

**(Location)**

**Located函数**

**该位置对应的地址**

**指针**

**3.2 建立考生信息库的实现**

**3.2.1 建立考生信息库的流程图**

**开始**

**输入考生人数**

**不合法**

**判断输入人数合法性**

**合法**

**输入考生基本信息**

**不合法**

**判断输入数据合法性**

**显示考生信息库**

**3.2.2 建立考生信息库的核心代码**

//构造函数，建立考试报名信息库

LinkList::LinkList()

{

head = NULL;

head = new ListNode; //给头结点分配空间

if (head == NULL) //代码的健壮性，如果分配失败则会给出提示

{

cout << "链表构建失败！" << endl;

system("pause");

return;

}

cout << "请首先建立考生信息系统！" << endl;

cout << "请输入考生人数：";

int number;

while (1)

{

cin >> number;

if (cin.good() == 0 || number < 1)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入的考生人数错误，请重新输入："; //代码健壮性，保证输入的考生人数是正整数

continue;

}

break;

}

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

for (int i = 0; i < number; i++)

{

ListNode current;

while (1)

{

cin >> current.student\_id;

if (cin.good() == 0 || current.student\_id < 1)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入的第" << i + 1 << "位考生考号错误，请从此项开始重新输入，考号，姓名，性别，年龄及报考类型：" << endl;

continue;

} //代码的健壮性，可以精确的定位到是哪一位学生的输入考号错误，并且从 此项 开始继续输入

break;

}

if (Search(current.student\_id) != -1)//代码的健壮性，可以防止输入两位考生的考号相同

{

cout << "第" << i + 1 << "位的考生考号已经存在，请此项开始重新输入，考号，姓名，性别，年龄及报考类型：" << endl;

i--;

cin.clear();

cin.ignore(1024, '\n');

continue;

}

while (1)

{

cin >> current.student\_name;

if (cin.good() == 0 || current.student\_name.size() < 2 || current.student\_name.size() > 10)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入的第" << i + 1 << "位考生姓名有误，请此项开始重新输入，姓名，性别，年龄及报考类型：" << endl;

continue; //代码的健壮性，可以精确的定位到是哪一位学生的输入姓名错误，并且从 此项 开始继续输入

}

break;

}

while (1)

{

cin >> current.student\_sex;

if (cin.good() == 0 || current.student\_sex != "男" && current.student\_sex != "女")

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入的第" << i + 1 << "位次考生性别有误，请此项开始重新输入，性别，年龄及报考类型：" << endl;

continue; //代码的健壮性，可以精确的定位到是哪一位学生的输入性别错误，（这里的性别只能是“男”或者“女”）并且从 此项 开始继续输入

}

break;

}

while (1)

{

cin >> current.student\_age;

if (cin.good() == 0 || current.student\_age < 1 || current.student\_age> 100)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入的第" << i + 1 << "位考生年龄有误，请此项开始重新输入，年龄、报考类型：" << endl;

continue; //代码的健壮性，可以精确的定位到是哪一位学生的输入年龄错误，并且从 此项 开始继续输入

}

break;

}

while (1)

{

cin >> current.student\_category;

if (cin.good() == 0 || current.student\_category.size() < 2 || current.student\_category.size() > 20)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入的第" << i + 1 << "位考生报考类型有误，请此项开始重新输入，报考类型：" << endl;

continue; //代码的健壮性，可以精确的定位到是哪一位学生的输入报考类型错误，并且从 此项 开始继续输入

}

break;

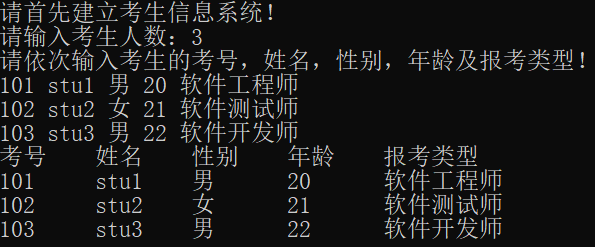
}

Insert(i, current);

}

}

在创建考生信息库的时候有着健全的健壮性检验，这在第四部分的项目测试会具体讲到；同时也用到了Insert(int Location,ListNode&node)函数，如果输入的新的考生信息正确，则会将其插入到链表的末尾，从而完成考试报名系统的构建。

**3.2.3 建立考生信息库的功能截图**

**3.3 插入考生信息功能的实现**

**3.31 插入考生信息功能的流程图**

**开始**

**输入插入位置**

**不合法**

**判断插入位置合法性**

**合法**

**通过Located函数找到地址**

**地址不存在**

**地址存在**

**输入考生基本信息**

**插入位置不合法**

**不合法**

**判断输入数据合法性**

**合法**

**显示考生信息库**

**3.32 插入考生信息的核心代码**

//插入函数，输入要插入考生的位置和基本信息，返回插入成功或失败

bool LinkList::Insert(int Location, ListNode& node)

{

ListNode\* current = Located(Location);

if (current == NULL)

{

return false; //代码的健壮性，如果该地址不存在返回false

}

else

{

ListNode\* Newnode = new ListNode(node);

if (Newnode == NULL)

{

cout << "新链表结点构建失败！" << endl; //代码的健壮性，如果申请失败则会提示

return false;

}

Newnode->next = current->next; //插入新的结点

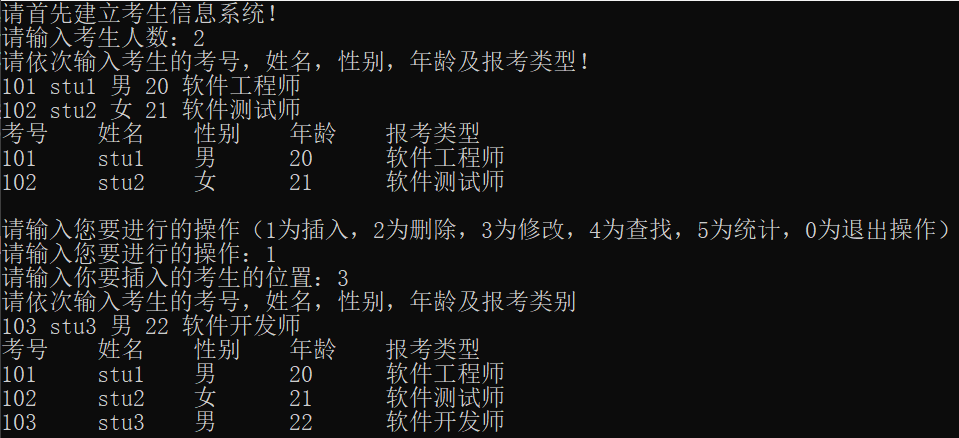
current->next = Newnode;

return true;

}

}

插入函数要求输入一个整型参数(插入位置)和一个新的结点(储存新的考生信息)，其中新的结点的考生信息在主函数里可以输入，插入位置利用Located函数可以返回一个插入点位置的地址，如果插入点地址不存在则返回false，在主函数里报插入位置不合法；如果插入点地址存在则返回true，进行插入操作，然后再通过ShowList()函数将新的考试报名系统展示出来。

**3.33 插入考生信息的功能截图**

**3.4 删除考生信息的实现**

**3.4.1 删除考生信息功能的流程图**

**开始**

**输入删除位置**

**不合法**

**判断删除位置合法性**

**合法**

**前一个地址、删除地址不存在**

**通过Located函数找到前一个地址**

**前一个地址、删除地址存在**

**进行删除操作**

**删除考生不存在**

**显示删除的考生信息和删除后的考生信息库**

**3.4.2 删除考生信息的核心代码**

//删除函数，输入要删除考生的位置（在main函数里面输入学号，通过Search函数返回一个位置，所以这个函数的一个输入为待删除考生的位置即可），返回删除成功或失败，node用于记录删除的考生的基本信息

bool LinkList::Delete(int Location, ListNode& node)

{

ListNode\* current, \* temp = NULL;

current = Located(Location - 1); //得到待删除结点的前一个地址

if (current == NULL || current->next == NULL)

{

return false; //代码的健壮性，如果当前结点或下一个结点不存在，无法进行删除操作

}

else

{

temp = current->next;

current->next = temp->next;

node.student\_id = temp->student\_id;

node.student\_age = temp->student\_age;

node.student\_name = temp->student\_name;

node.student\_sex = temp->student\_sex;

node.student\_category = temp->student\_category; //node用来储存删除结点信息

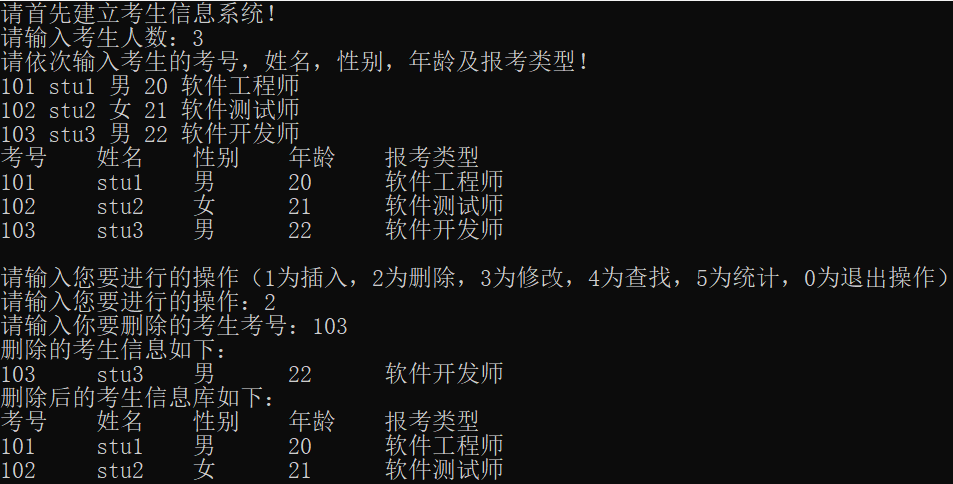
delete temp; //删除结点

return true;

}

}

删除函数输入一个整型参数(删除位置)和一个新结点(储存被删除考生信息)，利用Located函数得到插入点前一个位置的地址，如果插入点前一个地址(current)不存在或插入点地址(current->next)不存在则返回false，在主函数里报删除考生不存在；如果插入点地址存在则返回true，进行删除操作，然后再通过ShowNode()和ShowList()函数将被删除考生信息和删除后的考试报名系统展示出来。

**3.4.3 删除考生信息的功能截屏**

**3.5 修改考生信息的实现**

**3.5.1 修改考生信息功能的流程图**

**开始**

**输入修改位置**

**不存在**

**修改考生不存在**

**判断修改位置存在性**

**存在**

**通过Delete函数将当前结点删除**

**输入修改信息**

**不合法**

**判断修改信息合法性**

**合法**

**通过Insert函数将信息插入**

**显示修改前考生信息和修改后考生信息库**

**3.52 修改考生信息的核心代码**

//修改函数，输入要修改考生的位置，（在main函数里面输入学号，通过Search函数返回一个位置，所以这个函数的一个输入为待修改考生的位置即可），返回修改成功或失败，anode为修改后考生的基本信息，bnode用于记录修改前考生的基本信息

bool LinkList::Change(int Location, ListNode& anode, ListNode& bnode)

{

ListNode\* current = Located(Location);

if (current == NULL)

{

return false;

}

else

{

Delete(Location, bnode); //调用Delete函数删除这个结点

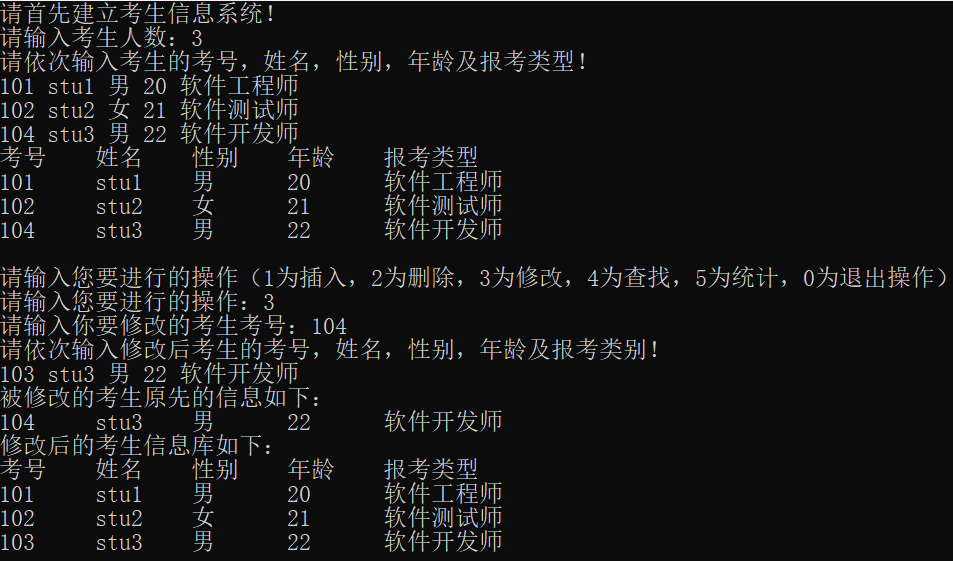
Insert(Location - 1, anode); //调用Insert函数在删除的这个结点插入新的结点，完成修改

}

return true;

}

修改函数，在main函数里输入要修改考生的学号，通过Search函数将学号转化为位置，在通过Located函数获得一个指针current指向要修改的考生的信息地址，再结合Delete函数删除这个考生当前的信息，最后通过插入函数，在这个考生被删除的位置插入一个新的考生信息，从而完成考生信息的修改。其中anode记录修改后的考生信息，在main函数里输入；bnode记录修改前考生信息，通过ShowNode函数可以把它输出出来。

**3.5.3 修改考生信息的功能截屏**

**3.6 查找考生信息的实现**

**3.6.1 修改考生信息功能的流程图**

**开始**

**输入查找考号**

**不存在**

**查找考生不存在**

**判断查找考号存在性**

**存在**

**通过Located函数将查找考生位置找到**

**通过ShowNode函数显示查找考生信息**

**3.6.2 查找考生信息的核心代码**

//寻找函数，输入一个考生的考号，返回这位考生在链表中的位置（第几个）

int LinkList::Search(int id) //通过考号定位这位考生在链表第几位

{

int Location = 1;

ListNode\* current = head->next;

while (current != NULL)

{

if (current->student\_id == id)

{

return Location;

}

else

{

current = current->next;

Location++;

}

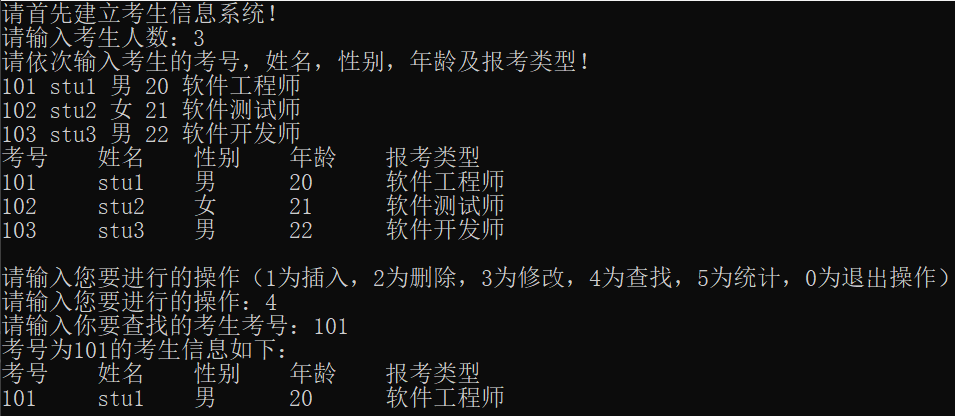
}

return -1; //如果没有找到这位考生（即考生不存在），return -1

}

Search函数通过输入一个考生的考号，遍历链表寻找考生在链表的位置，如果考生存在则返回该考生在链表中的第几位，然后再通过Located函数找到这位考生信息所在的地址，通过ShowNode函数将它输出出来如果不存在，则返回-1，提示查找的考生不存在。

**3.6.3 查找考生信息的功能截屏**

****

**3.7 统计考生信息的实现**

**3.7.1 统计考生信息功能的流程图**

**开始**

**是**

**结束程序**

**输出总人数**

**输出男生女生人数**

**当前结点是否为空**

**否**

**当前结点储存的性别**

**男**

**女**

**女生人数+1**

**男生人数+1**

**总人数+1**

**向下一个结点移动**

**3.7.2 统计考生信息的核心代码**

//计算链表长度（即考试报名系统总人数，同时分别统计男生、女生人数）

void LinkList::Length()

{

int length = 0; //length为链表长度，即考生的总人数

int number\_man = 0;

int number\_woman = 0;

ListNode\* current = head->next; //从头开始扫描链表，计算长度

while (current != NULL)

{

if (current->student\_sex == "男")

{

number\_man++;

}

else

{

number\_woman++;

}

current = current->next;

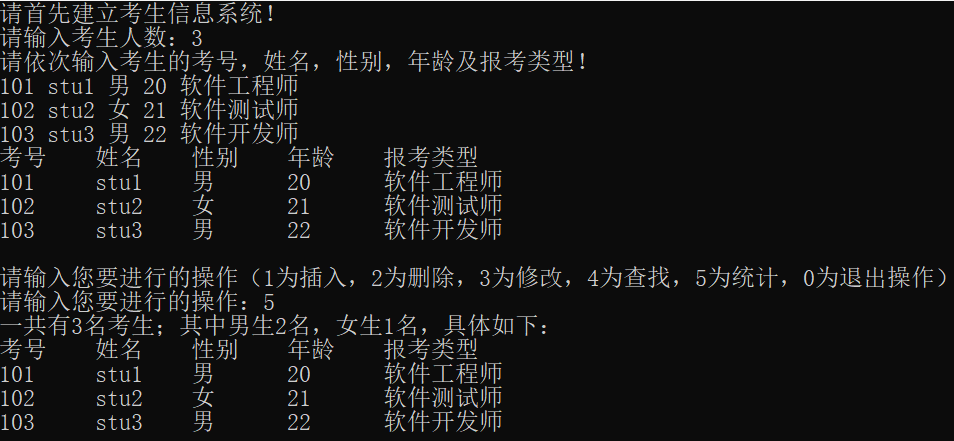
length++;

}

cout << "一共有" << length << "名考生；" << "其中男生" << number\_man << "名，女生" << number\_woman << "名，具体如下：" << endl;

}

Length函数可以统计链表的长度(即考生报名系统的总人数)，同时分别统计男生、女生的人数，最后进行输出。main函数里再利用ShowList的函数，将具体的男生女生考生基本信息输出出来。

**3.7.3 统计考生信息的功能截屏**

**3.8 主函数的实现**

**(为了节省篇幅，以下为主函数的部分源代码，并不是主函数所有代码，删去了一些健壮性检验的代码，这里只展示主函数的主体逻辑部分，cpp中有完整的代码)**

int main()

{

LinkList StudentSystem; //LinkList类的实例化

StudentSystem.ShowList();

int KeyDown;

cout << "请输入您要进行的操作（1为插入，2为删除，3为修改，4为查找，5为统计，0为退出操作）" << endl;

//代码的健壮性，保证输入的操作数字为0，1，2，3，4，5

while (1)

{

cout << "请输入您要进行的操作：";

while (1)

{

cin >> KeyDown;

if (cin.good() == 0 || KeyDown < 0 || KeyDown > 5)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入错误！请重新请输入您要进行的操作：";

continue;

}

break;

}

//0-退出程序

if (KeyDown == 0)

{

cout << "测试结束，请退出程序！" << endl;

break;

}

//1-插入操作

else if (KeyDown == 1)

{

cout << "请输入你要插入的考生的位置：";

int location;

cin >> location;

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

cin >> NewNode.student\_id;

cin >> NewNode.student\_name;

cin >> NewNode.student\_sex;

cin >> NewNode.student\_age;

cin >> NewNode.student\_category;

if (StudentSystem.Insert(location - 1, NewNode) == 0)

{

cout << "插入位置不合法，操作失败！" << endl;

}

StudentSystem.ShowList();

}

//2-删除操作

else if (KeyDown == 2)

{

cout << "请输入你要删除的考生考号：";

int id;

cin >> id;

int Location = StudentSystem.Search(id); //将学号变为该学生所在链表的位置

ListNode DeleteNode;

if (StudentSystem.Delete(Location, DeleteNode) == 0)

{

cout << "删除学生不存在，请重新操作！" << endl; //代码的健壮性，对没有该学生的提示

}

else

{

cout << "删除的考生信息如下：" << endl;

DeleteNode.ShowNode();

}

cout << "删除后的考生信息库如下：" << endl;

StudentSystem.ShowList();

}

//3-修改操作

else if (KeyDown == 3)

{

cout << "请输入你要修改的考生考号：";

int id;

cin >> id;

ListNode AfterChangeNode;

ListNode BeforeChangeNode;

cin >> AfterChangeNode.student\_id

cin >> AfterChangeNode.student\_name;

cin >> AfterChangeNode.student\_sex;

cin >> AfterChangeNode.student\_age;

cin >> AfterChangeNode.student\_category;

int Location = StudentSystem.Search(id);

if (StudentSystem.Change(Location, AfterChangeNode, BeforeChangeNode) == 0)

{

cout << "修改的考生不存在，操作失败！" << endl;//代码的健壮性，如果考生不存在则会提示

}

else

{

cout << "被修改的考生原先的信息如下：" << endl;

BeforeChangeNode.ShowNode();

cout << "修改后的考生信息库如下：" << endl;

StudentSystem.ShowList();

}

}

//4-查找操作

else if (KeyDown == 4)

{

cout << "请输入你要查找的考生考号：";

int id;

cin >> id;

if (StudentSystem.Search(id) == -1)

{

cout << "查找的考生不存在，操作失败！" << endl;

cout << endl;

}//代码的健壮性，如果查找的考生不存在则会提示

else

{

int Location = StudentSystem.Search(id);

cout << "考号为" << id << "的考生信息如下：" << endl;

cout << "考号\t" << "姓名\t" << "性别\t" << "年龄\t" << "报考类型\t" << endl;

(\*StudentSystem.Located(Location)).ShowNode();

}

}

//5-统计操作

else if (KeyDown == 5)

{

StudentSystem.Length();

StudentSystem.ShowList();

}

}

system("pause");

return 0;

}

主函数首先实例化了一个StudentSystem的链表类的对象，采用if-else加循环语句来确定用户所需要的操作，在每一个操作中都调用链表类对象的函数，完成相应的操作，并最后输出操作过后的结果。

**4.项目代码功能测试**

**4.1 建立考生信息库的测试**

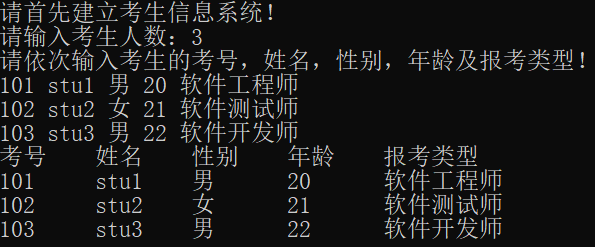
**4.1.1 建立考生信息库的功能测试**

**测试用例：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

**预期结果：**依次显示输入的考生的基本信息

**实验结果**：

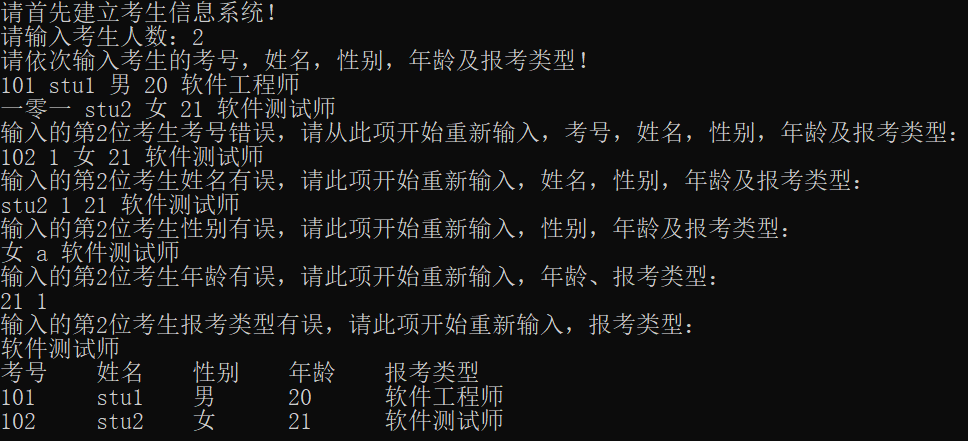
**4.1.2 建立考生信息库的代码健壮性测试**

**测试用例：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

1.但是在输入第二位考生的基本信息数据的时候考号、姓名、性别、年龄和报考类型分别输入不合法数据。

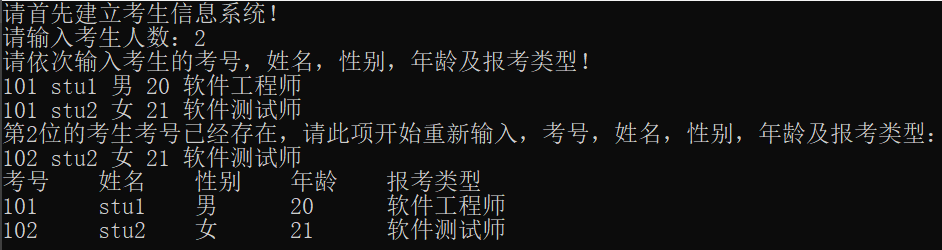
**预期结果：**给出提示(第几位考生的哪一个信息出错)，从输入错误的那一项开始重新输入考生基本信息

**实验结果**：

2**.**但是在输入第二位考生考号的时候考号输入了与前面考生相同的考号。

**预期结果：**给出提示(第几位考生的考号已经存在)，从输入错误的那一项开始重新输入考生基本信息

**实验结果**：

**4.2 插入考生信息的测试**

**4.2.1 插入考生信息的功能测试**

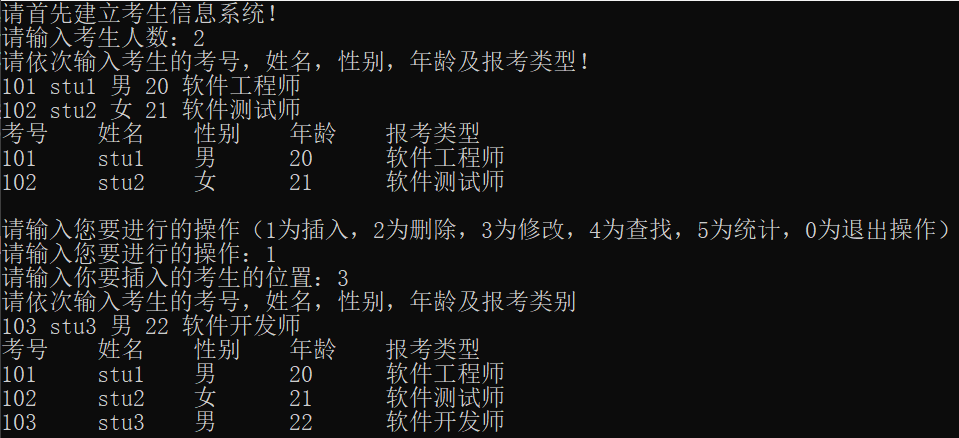
**测试用例：** 103 stu3 男 22 软件开发师

**预期结果：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

**实验结果**：

****

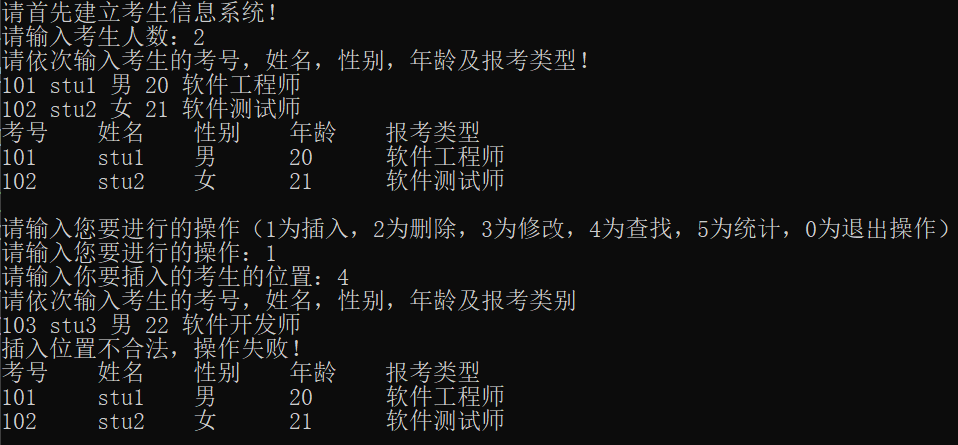
**4.2.2 插入考生信息的代码健壮性测试**

**测试用例：** 103 stu3 男 22 软件开发师

1.但是在输入插入位置的时候插入的位置实际并不存在

**预期结果：**给出提示(插入的位置非法，操作失败)

**实验结果**：

****2.但是在输入要插入考生的基本信息数据的时候考号、姓名、性别、年龄和报考类型分别输入不合法数据。

**预期结果和实验结果：**给出提示(插入考生的信息哪一个信息出错)，从输入错误的那一项开始重新输入考生基本信息

**(同4.1.2中的第一个错误方式，这里不再给出截图)**

**4.3 删除考生信息的测试**

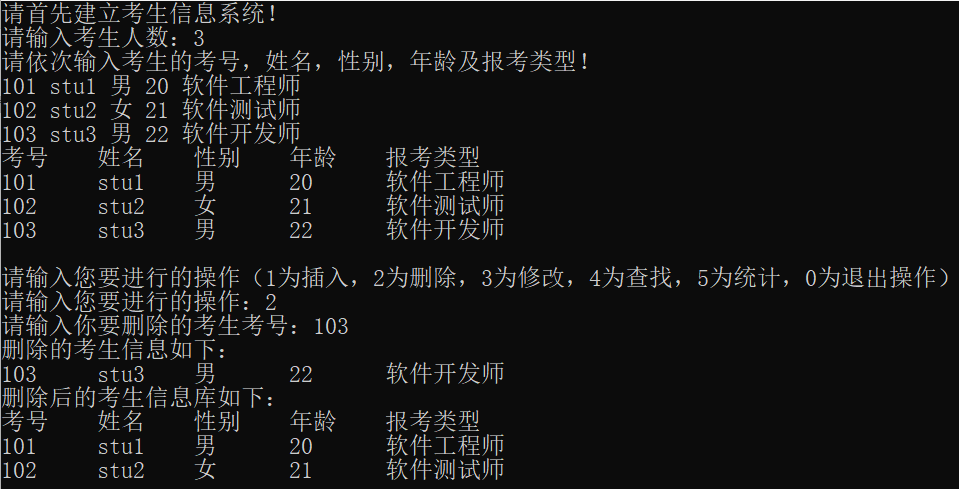
**4.3.1 删除考生信息的功能测试**

**测试用例：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

输入要删除的考生的考号：103

**预期结果：**显示要删除考生的基本信息和删除后的考试报名系统

**实验结果：**

**4.3.2 删除考生信息的代码健壮性测试**

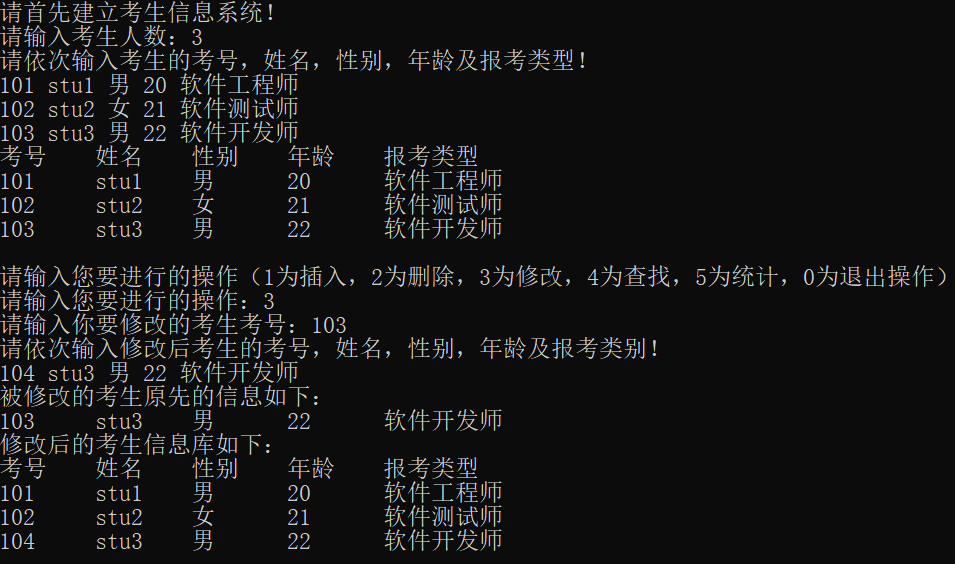
**测试用例：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

输入要删除的考生的考号：104

**预期结果：**考试报名系统给出提示，该考生不存在

**实验结果：**

**4.4 修改考生信息的测试**

**4.4.1 修改考生信息的功能测试**

**测试用例：** 101 stu1 男 20 软件工程师

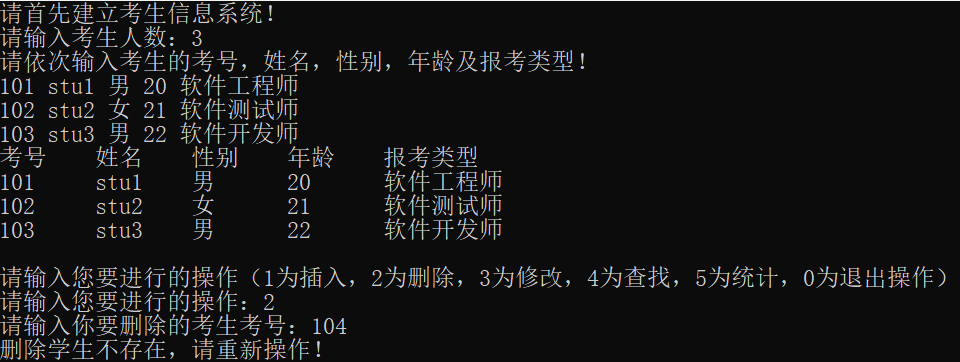
102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

输入要修改的考生的考号：103

输入修改后的考生信息：104 stu3 男 22 软件开发师

**预期结果：**显示修改前这位考生的基本信息和修改后的考试报名系统

**实验结果：**

**4.4.2 修改考生信息的代码健壮性测试**

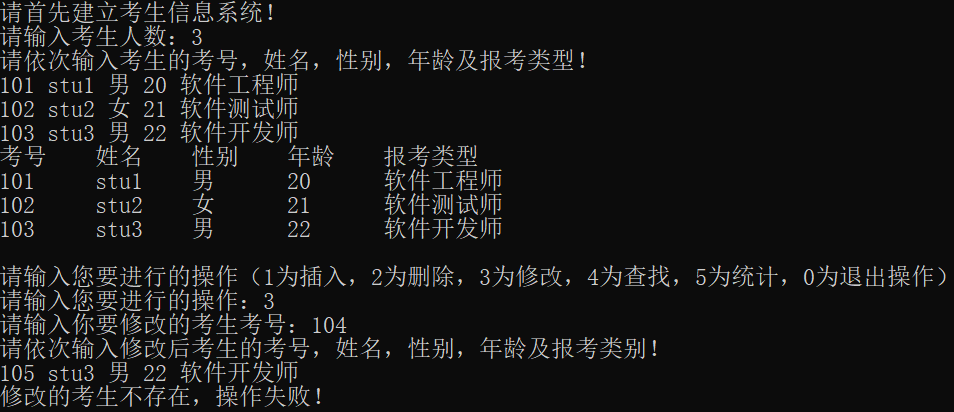
**测试用例：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

1.但是在输要删除考号的时候输入的考号实际并不存在

**预期结果：**给出提示(要修改的考生不存在，操作失败)

**实验结果**：

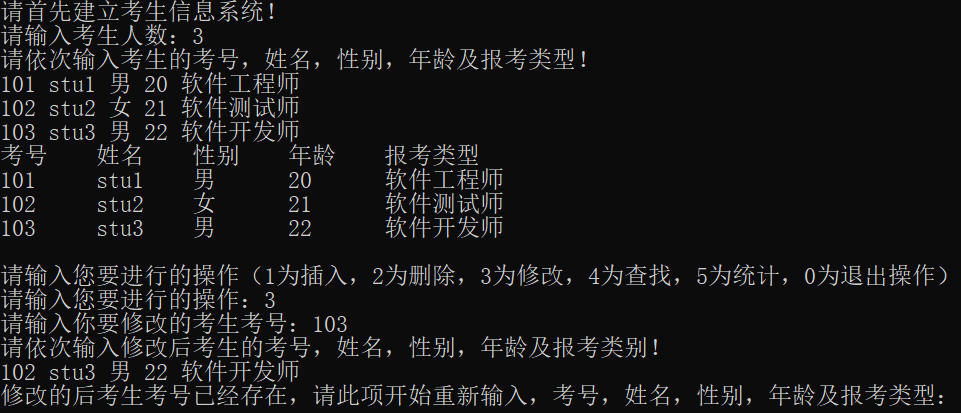
2.但是在输入要修改考生的基本信息数据的时候考号、姓名、性别、年龄和报考类型分别输入不合法数据。

**预期结果和实验结果：**给出提示(修改考生的信息哪一个信息出错)，从输入错误的那一项开始重新输入修改考生基本信息

**(同4.1.2中的第一个错误方式，这里不再给出截图)**

3.但是在输入修改后的考生考号的时候输入了与前面考生相同的考号。

**预期结果：**给出提示(修改的考生考号已经存在)，从输入错误的那一项开始重新输入考生基本信息

**实验结果：**

**4.5 查找考生信息的测试**

**4.5.1 查找考生信息的功能测试**

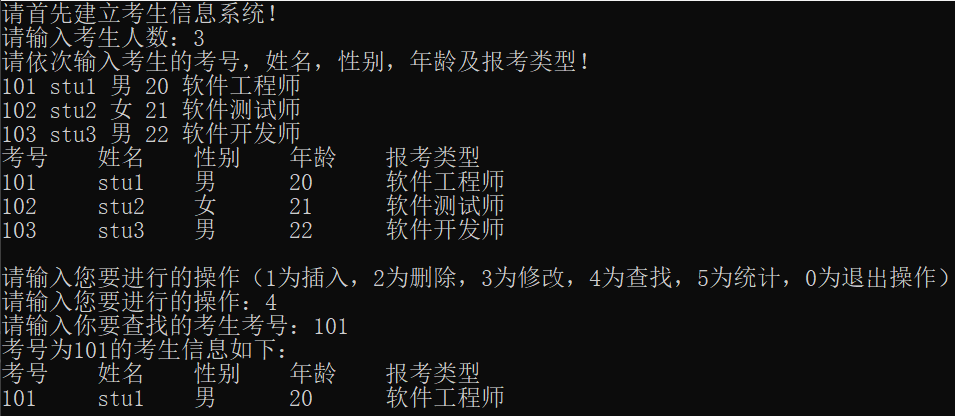
**测试用例：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

输入要查找的考生的考号：101

**预期结果：** 101 stu1 男 20 软件工程师

**实验结果：**

**4.5.2 查找考生信息的代码健壮性测试**

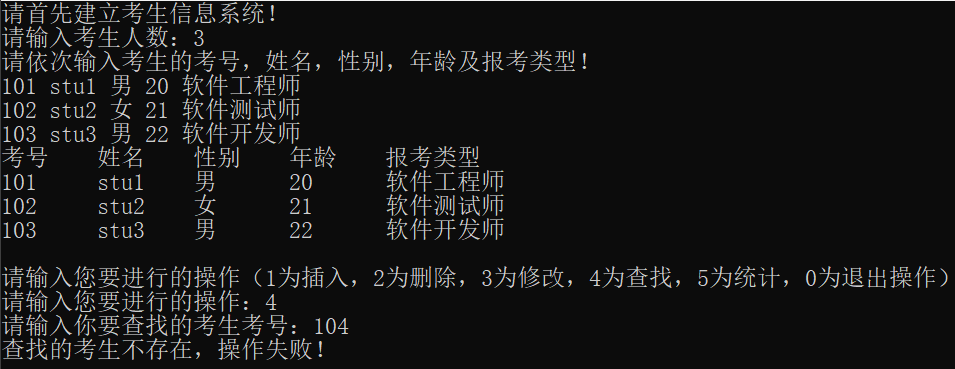
**测试用例：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

但是在输要查找考号的时候输入的考号实际并不存在

**预期结果：**给出提示(要查找的考生不存在，操作失败)

**实验结果：**

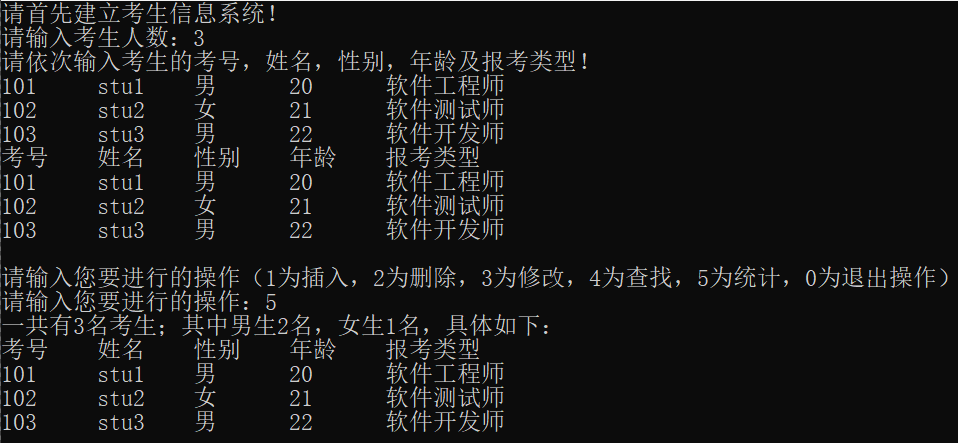
**4.6 统计考生信息的测试**

**测试用例：** 101 stu1 男 20 软件工程师

102 stu2 女 21 软件测试师

103 stu3 男 22 软件开发师

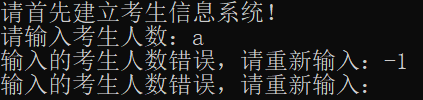
**预期结果：**统计考试报名系统的总人数和男女生分别人数

**实验结果：**

**4.7 其他基本输入的健壮性测试**

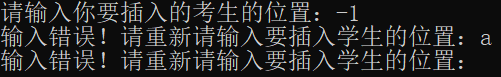
本考试报名系统在所有需要输入的数据类型应该是数字或是字符串的地方都做了相应的健壮性检验，即如果需要的是一个在一定范围内的数字，如果输入的是字符串或者超过范围的数字，都会提示输入错误，让其重新输入；在需要输入字符串的地方，如果输入其他数据类型，也会提示输入错误，让其重新输入(这个在4.1.2的第一个健壮性检验的时候已经给出输入姓名、性别或报考类型错误的截图了，下文不再举例)，下文举例一些要求输入数字但是非法的举例。

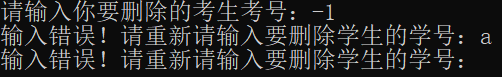
**4.7.1 建立考试报名系统输入人数数字的健壮性**

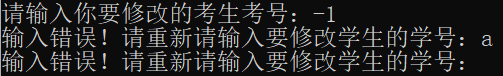
****

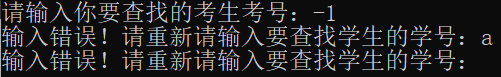
这里需要输入的是一个正整数，输入字符或是非正整数则会提示。

**4.7.2 插入操作时输入的位置、删除、修改、查找操作时输入考号数字的健壮性**

****

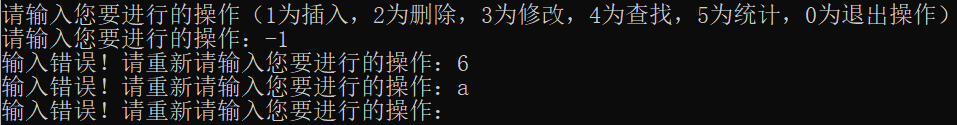
****

****

****

插入操作时输入的位置、删除、修改、查找操作时输入考号数字的数据类型都应该是int型而且是正整数，如果输入的是字符、不为正整数都会提示，至于输入的位置和考号存不存在，在上文每一个功能的健壮性检验里都阐释过了。

**4.7.3 在main函数中选择操作数的健壮性**

在main函数中选择操作数应该是0，1，2，3，4，5这几个整数，如果输入的不是这几个数字则会给出提示。

**5.项目心得与体会**

考试报名系统是本人到现在完成的最长篇幅的代码之一。从中我学习到了很多编程经验和技巧。第一，我对链表的插入、删除等等最基本的操作有了更加深入的理解，也对这些操作的代码实现更加熟练；第二，这是本人第一次用C++类的方式来完成这一程序，在完成该程序的过程中，我查阅了很多资料，学习了关于面向对象的很多方法，如何定义一个类，如何选择私有成员，公有成员，如何调用类的共有操作等等，是我对C++面向对象编程进一步了解；第三，写了很多关于链表类的删除、插入、查找、计算长度的代码，可以将这些函数保存起来，对于后续的题目，有用到这些类似操作的我就可以直接利用，使得后续的编程更加方便。总之，我在这次项目中获得了很多编程想法和编程方法。