数据结构课程设计

项目说明文档

考试报名系统

|  |  |
| --- | --- |
| 作者姓名： | 高逸轩 |
| 学 号： | 2053385 |
| 指导教师： | 张 颖 |
| 学院专业： | 软件学院 软件工程 |



同济大学

Tongji University

# 1项目分析

## 1.1 项目需求分析

本程序完成了以下需求：

* 功能完善

实现了建立、增加、查询、修改、删除的基本功能。

* 执行效率高

能够存储的数据条数应该足够多，使用的数据结构和算法应该具有较低的时间和空间复杂度。

* 健壮性

对于错误的输入如输入了已存在的考号等，程序应该做到不崩溃并且给予一定的提示。

# 1.2 项目要求

### 1.2.1 功能要求

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。同时实现健壮性。并提供了部分测试数据。

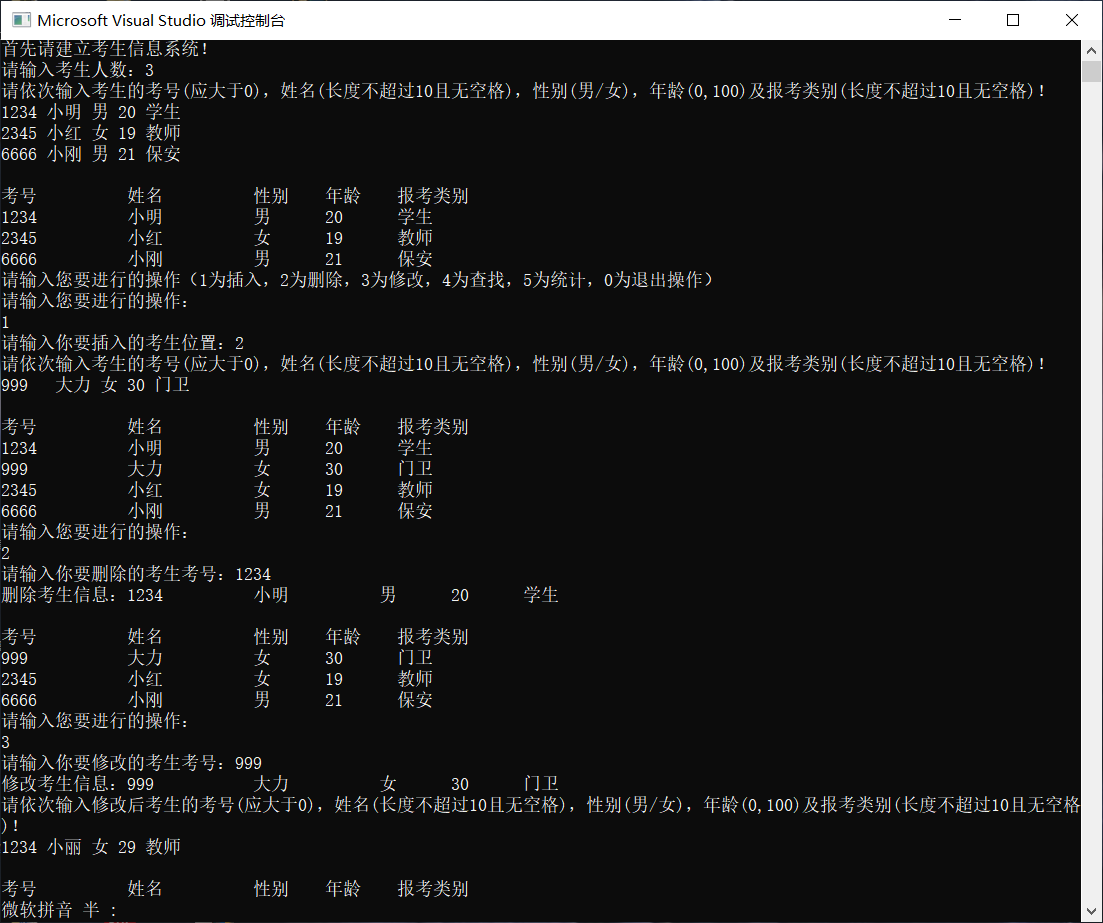
### 1.2.2 输入格式

每次输入需要执行的操作，以及对应考生的信息（包括准考证号、姓名、年龄和报考类别）。

### 1.2.3 输出格式

输出操作学生的信息，以及操作后的全部数据。

### 1.2.4 项目示例



# 2 项目设计

## 2.1 数据结构设计

本程序中需要快速进行大量对于数据的增加、删除、修改、查找等操作。因此，选择的数据结构要应该能够快速的对数据执行以上操作。本系统采用了单链表作为底层数据。

链表的优点有：

1. 链表是一个动态数据结构，因此它可以在运行时通过分配和取消分配内存来增长和收缩。 因此，无需给出链表的初始大小。
2. 节点的插入和删除较为容易。 与数组不同，在插入或删除元素后，我们不必移动元素。 在链表中，我们只需要更新节点的下一个指针中存在的地址即可。这样可以实现时间复杂度的插入和删除操作，比数组的要快很多。s
3. 由于链表的大小可以在运行时增加或减小，因此不会浪费内存。 在数组的情况下，会浪费大量内存.例如，如果我们声明一个大小为10的数组并在其中仅存储6个元素，那么就会浪费4个元素的空间。链表中没有这种问题，因为仅在需要时才分配内存。

本系统实现的链表结构各种操作的时间复杂度如下：

* 插入操作：O(n)
* 删除操作：O(n)
* 查询操作：O(n)
* 遍历操作：O(n)

## 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（List），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。以下为本程序的设计。

### 2.2.1 结点类（LinkedListNode）

// 节点，一个节点表示一个考生的信息

typedef struct LinkedListNode

{

int testNum = -1; // 考号

string name; // 姓名

string sex; // 性别 0为男，1为女

int age; // 年龄

string testMajor; // 报考职业

LinkedListNode\* next = NULL; // 指向下一个节点

// 构造函数，设置考生初始信息

LinkedListNode(const int num1 = NULL, const string str1 = "", const string str2 = "", int const num2 = NULL, const string str3 = "", LinkedListNode\* num3 = NULL);

// 输出考生信息

void Display();

}student;

### 2.2.2 链表类（LinkedList）

// 链表类

class LinkedList

{

public:

// 构造函数，设置初始信息

LinkedList()

// 析构函数，释放空间

~LinkedList()

// 计算带附加头结点的单链表的长度

int Length()

// 寻找链表中是否含有x考号考生，搜索成功返回该节点位置，否则返回-1

int Search(int x)

// 定位函数，返回表中第i个元素的地址

LinkedListNode\* Locate(int i)

// 删除第i个元素,x返回该元素的值

bool Remove(int i, LinkedListNode& x)

// 将新元素x插入到第i个节点后

bool Insert(int i, LinkedListNode& x)

// 将链表中信息依次输出

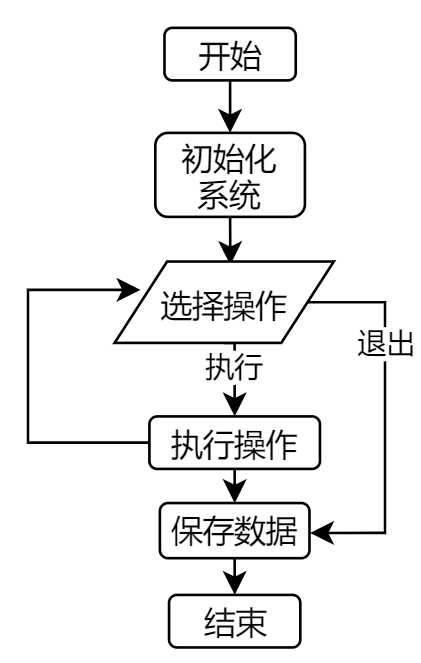
void Display()

private:

LinkedListNode\* head = NULL; // 链表头指针

};

## 2.3 项目功能流程图



# 3 核心代码详解

## 3.1 建立链表

在构造函数中，通过自行构造的getint()和getstring()函数，得到了考号、姓名等信息。同时对内存处理进行了健壮性处理。在这个过程中，若新输入的考生信息有重复存在，则应给出对应的错误提示，这个过程通过查询函数Search()实现。若输入信息合理，则应将新的节点插入链表末尾，用到了Insert()函数。

## 3.2 销毁链表

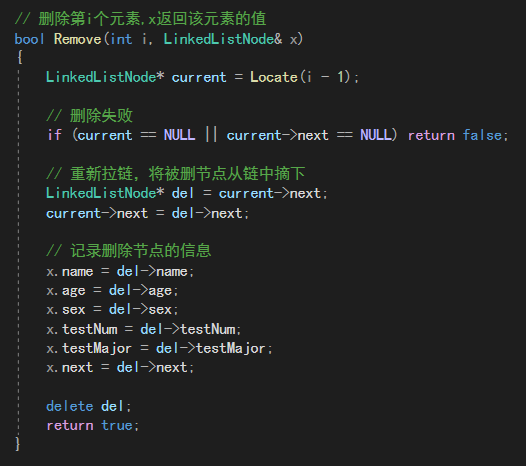
在析构函数中，我们从头结点开始向后遍历，在向后遍历的过程中同时释放经过节点的空间。由于采取了空表头链表的写法，最后我们还需要释放头节点的空间。

## 3.3 搜索数据

以节点的考号为关键字，对整个链表进行遍历，若查询成功，则返回其位置；若查询失败，则返回-1。

## 3.4 定位数据

## 3.5 删除数据



## 3.6 插入元素

如果插入位置不合法，则返回false；否则利用定位函数找到插入位置，进行插入操作。

## 3.7 展示链表内容

遍历链表，根据要求的格式输出各个节点内容。

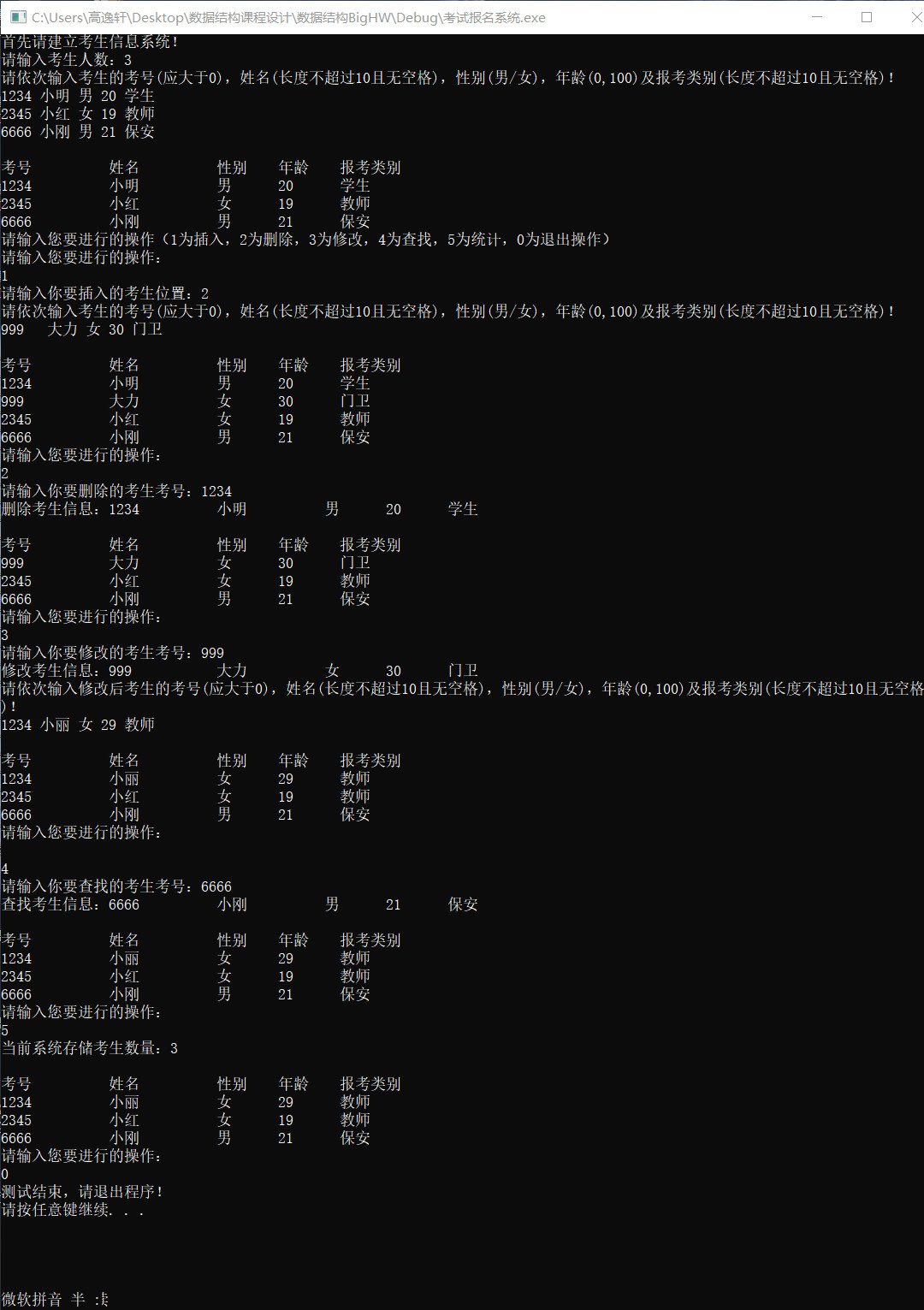
# 4 项目测试

## 4.1 健壮性测试



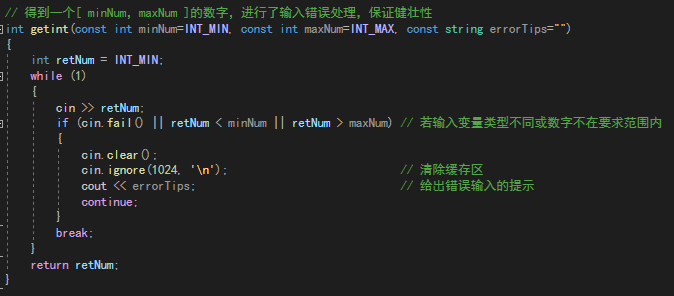
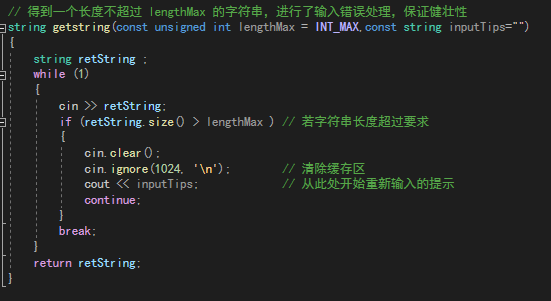
以上为两种错误信息的处理，一种为输入的数据类型与所需类型不同，如：年龄输入字符串变量，这种类型的输入错误通过自行编写的getint()函数和getstring()函数实现；一种为重复插入已经存在的考生信息或删除不存在的考生信息，通过链表的Search()函数实现。

## 4.2 功能测试

为方便老师测试，提供了文件1\_test.txt，内含一组数据，测试了本程序的全部功能。以下为一组测试数据：

# 5 心得与体会

本项目是链表这个数据结构的入门基础题目，主要涉及了链表的建立、插入、删除、修改、查询等基本操作，以及设计类的关系等知识内容。在本次作业中，我学习了带有空表头的链表操作编写：带有空表头的链表在编写功能函数时更加简洁方便，统一了写法，不需要写太多的特判。

另外，健壮性的保证也是重要内容。在本次作业中，我采用了自己编写的getint和getstring函数，以及其他操作，处理了错误输入、空间异常等问题，保证了程序的稳定性。

由于篇幅原因，报告内还有很多内容与解释没有展示，请老师和助教老师再移步源程序，在其中的注释写了每一步过程的详解。