项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 杨滕超

学 号： 2151298

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 项目概述 3](#_Toc495668153)

[1.1 项目背景 3](#_Toc495668154)

[1.2 项目功能 3](#_Toc495668155)

[1.3 项目分析 4](#_Toc495668154)

[2 项目设计 4](#_Toc495668156)

[2.1 总体设计 4](#_Toc495668157)

[2.2 类设计 5](#_Toc495668158)

[3 项目实现 6](#_Toc495668161)

[3.1 总体实现 6](#_Toc495668162)

[3.1.1 总体实现流程图 6](#_Toc495668163)

[3.1.2 总体实现思路 7](#_Toc495668163)

[3.1.3 总体实现主要代码 8](#_Toc495668164)

[3.2 初始化实现 9](#_Toc495668166)

[3.2.1 初始化实现流程图 9](#_Toc495668167)

[3.2.2 初始化实现思路 1](#_Toc495668167)0

[3.2.3 初始化实现主要代码 1](#_Toc495668168)0

[3.3 插入实现 1](#_Toc495668170)1

[3.3.1 插入实现流程图 1](#_Toc495668171)1

[3.3.2 插入实现思路 1](#_Toc495668171)2

[3.3.3 插入实现主要代码 1](#_Toc495668172)2

[3.4 删除实现 13](#_Toc495668174)

[3.4.1 删除实现流程图 13](#_Toc495668175)

[3.4.2 删除实现思路 14](#_Toc495668175)

[3.4.3 删除实现主要代码 14](#_Toc495668176)

[3.5 修改实现 15](#_Toc495668178)

[3.5.1 修改实现流程图 15](#_Toc495668179)

[3.5.2 修改实现思路 15](#_Toc495668179)

[3.5.3 修改实现主要代码 15](#_Toc495668180)

[3.6 查找实现 16](#_Toc495668182)

[3.6.1 查找实现流程图 16](#_Toc495668183)

[3.6.2 查找实现思路 16](#_Toc495668183)

[3.6.3 查找实现主要代码 16](#_Toc495668184)

[3.7 统计实现 17](#_Toc495668182)

[3.6.1 统计实现流程图 17](#_Toc495668183)

[3.6.2 统计实现思路 17](#_Toc495668183)

[3.6.3 统计实现主要代码 18](#_Toc495668184)

[3.8 存入文件实现 1](#_Toc495668182)8

[3.8.1 存入文件实现主要代码 18](#_Toc495668184)

[3.8.2 存入文件实现主要代码 18](#_Toc495668184)

[3.8.3 存入文件实现主要代码 18](#_Toc495668184)

[4 项目测试 19](#_Toc495668186)

[4.1 初始化测试 19](#_Toc495668187)

[4.1.1 文件读入 19](#_Toc495668188)

[4.1.2 文件不存在 19](#_Toc495668189)

[4.1.3 手动输入 1](#_Toc495668190)9

[4.1.4 考生人数错误 1](#_Toc495668190)9

[4.2 插入测试](#_Toc495668193) 20

[4.2.1 正确位置](#_Toc495668194) 20

[4.2.2 非法位置输入 2](#_Toc495668195)0

[4.3 删除测试](#_Toc495668197) 20

[4.3.1 正确输入存在的考生考号 2](#_Toc495668198)0

[4.3.2 输入不存在的考生考号 2](#_Toc495668199)0

[4.4 查找测试 2](#_Toc495668197)1

[4.4.1 输入存在的考生考号 2](#_Toc495668198)1

[4.4.2 输入不存在的考生考号 2](#_Toc495668198)1

[4.5 修改测试 2](#_Toc495668197)1

[4.5.1 修改前后考号不匹配 2](#_Toc495668198)1

[4.6 统计测试 2](#_Toc495668197)1

[4.7 存入文件测试 2](#_Toc495668197)2

[4.8 输入操作错误测试 2](#_Toc495668197)2

1 项目概述

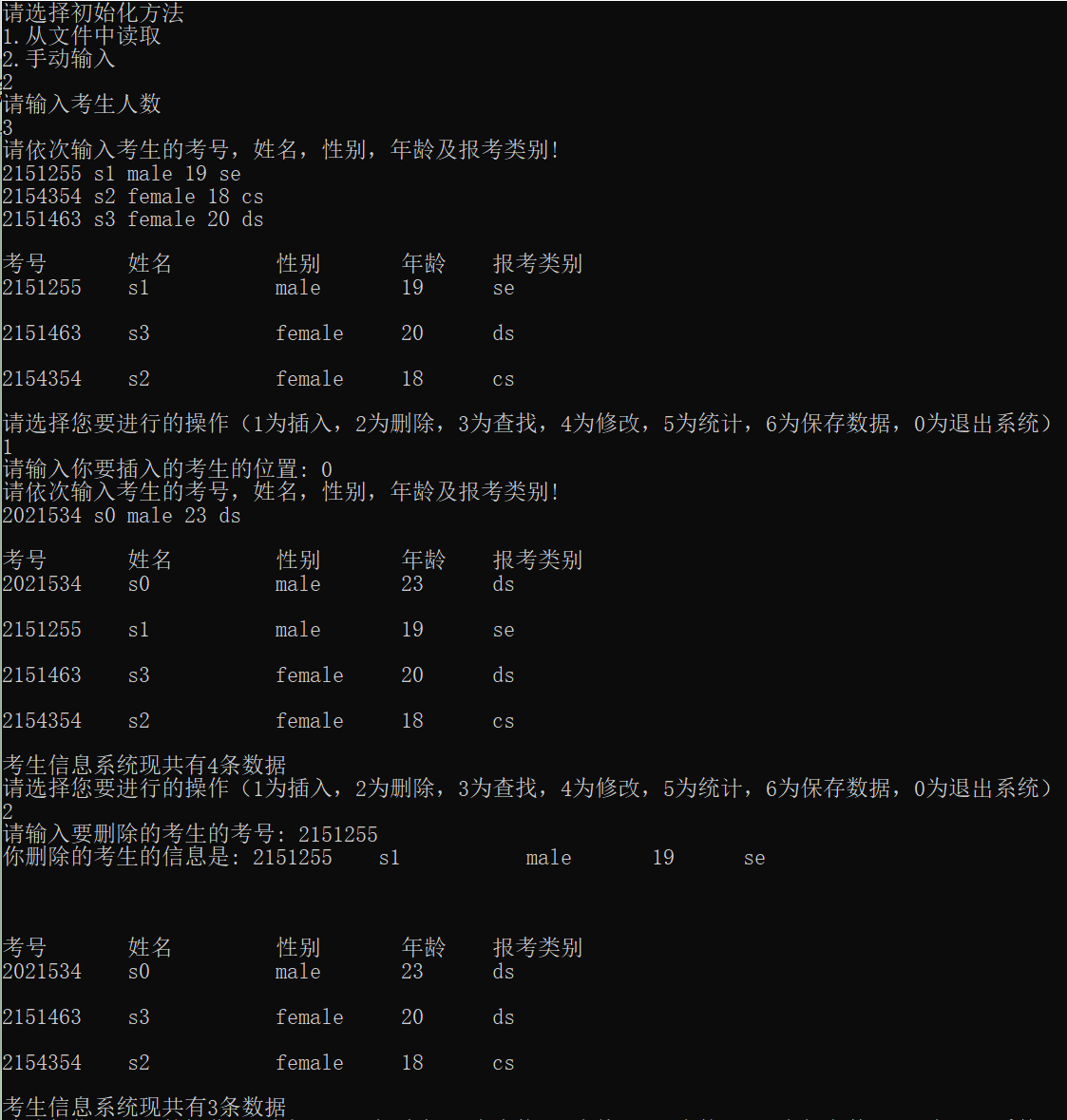
* 1. 项目背景

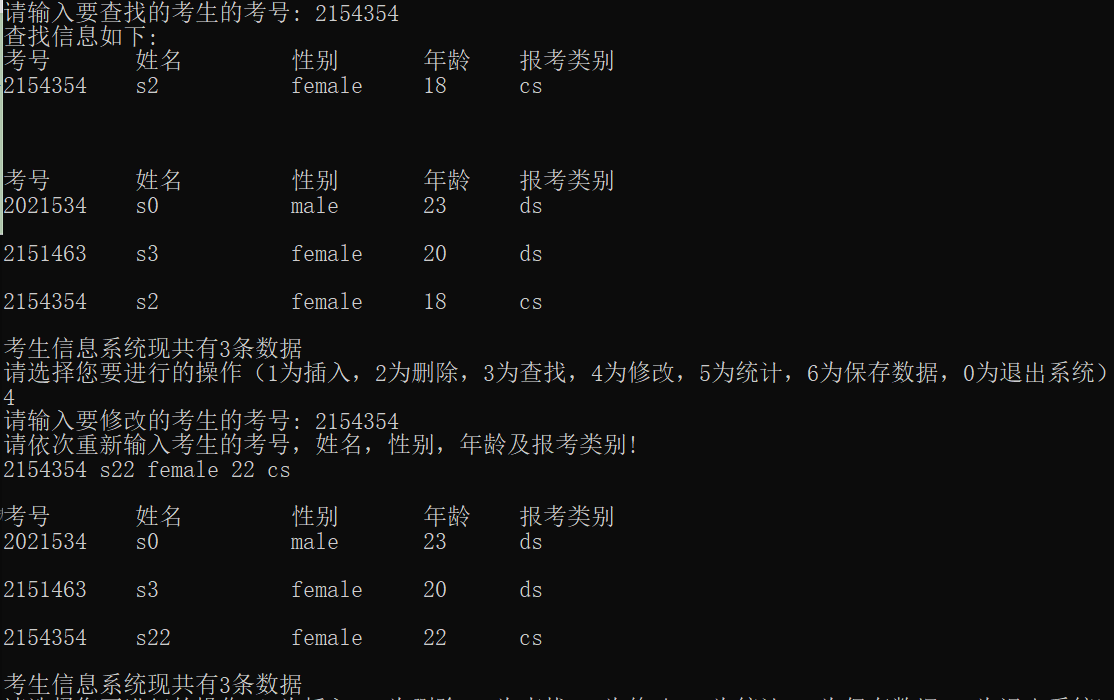
在日常生活中，每个人都需要经历大大小小的考试，这对于个人来说可能只是参加作答即为绝大部分的流程，但对于学校来说，统计成千上万考生的信息确实是一个巨大的挑战。随着信息技术的深化发展与应用，考试报名系统对于学校恐怕是一个必不可少的程序，它的出现极大地减少了学校教务的工作量。通过对考试报名系统的最大利用，在面临日益增加的考试数量与考试人数，学校可以以最快时间处理其中考生的信息，这对于一个学校来说具有重大意义，例如对考场分布的管理、考试时间的安排以及监考老师的分配等等事务具有巨大的作用。因此，在我们的学习过程中，模拟开发一套考试报名系统十分具有现实意义，不仅从实际问题出发，还更加深入对数据结构的选择与应用。

* 1. 项目功能

该项目需要完成对考生信息的搜集与整合，其中重点是完成对考生信息的输入存储、插入、删除、查找、修改、统计等功能，其中考生信息包括考生的考号、姓名、性别、年龄以及报考类别。

项目主要功能实例如下图所示：





* 1. 项目分析

除了基本的增删改查功能外，对程序功能的完善十分重要。

1. 对于统计功能，本人使用以红黑树为底层的Map结构进行，实现按照性别、报考类别的人数统计，丰富该系统的分类筛查功能。
2. 通过对文件的操作，实现保存于读取文件，极大地保证了输入数据的持久性，避免了再次启动程序而需要重新输入之前数据的繁琐。减少了使用系统人员的工作量。
3. 对于错误输入等操作和不了解如何操作，该系统具有一定的健壮性，为使用者提供相当的提示，使用者在未事先学习过的情况下，可以根据提示进行正确操作，而不会产生系统的崩溃。
4. 对于数据的呈现，在显示生活中，考生的数据成千上万，而并不需要在执行操作的过程中全部展示完，我们通常根据学考进行排序，选取前十条进行展示。当然，使用者也可以通过手动调整，使得特定学号的学生信息的排列不按照默认排序方法。
5. 考虑到考生数据的大小，选取合适的数据结构进行存储十分重要。因此，基于链表的考生数据库正符合了该要求，在较小的时间复杂度和空间复杂度下，该数据结构能够良好地适应考试报名系统的要求。

2 项目设计

2.1 总体设计

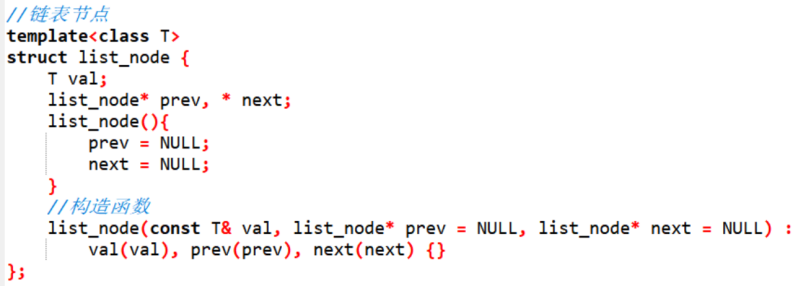
考虑到存储数据成千上万，我们需要用到链表作为考生系统数据库的底层。其中链表的结点具有分散的空间，通过指针相互连接，具有空间的灵活性。同时对于需要实现的各种功能，时间复杂度均为O(n)。

其中将链表结点(list\_node)声明为struct类型，链表(List)声明为class类型，由链表类可以访问链表结点中的数据成员。

为了实现链表的可复用性，我们利用模板template定义上述类，保证了链表中的数据可以为任意类型。

2.2 类设计

·链表结点类设计：其中包括需要储存的数据，前驱节点和后继节点。如下图所示：



·链表类设计：数据成员包括，头结点，尾结点以及记录节点个数数据。

主要功能(与本题有关)包括：

·//返回链表大小inline int size() const;

·//判断链表是否为空inline bool empty() const;

·//清空链表void clear();

·//最后添加元素void emplace\_back(const T& x)；

·//最后删除元素void pop\_back();

·//开头添加元素void emplace\_front(const T& x);

·//开头删除元素void pop\_front();

·//插入void insert(const List<T>::list\_iterator& it, const T& x);

·//删除void remove(const List<T>::list\_iterator& it);

·//返回迭代器的查找typename List<T>::list\_iterator find(const T& x);

·//迭代器begin inline List<T>::list\_iterator begin();

·//迭代器end inline List<T>::list\_iterator end();

·游标类设计：数据成员包括指向链表结点的指针

主要功能包括：

·//重载\* T& operator\*();

·//重载-> T\* operator->();

·//重载自加inline List<T>::list\_iterator& operator++();

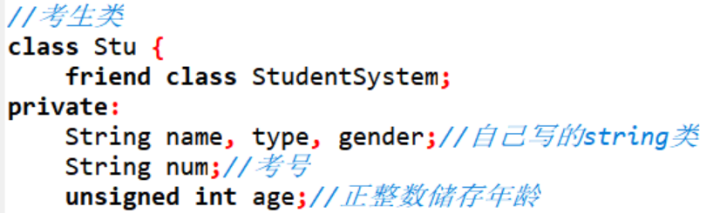
·//重载自减inline List<T>::list\_iterator& operator--();

·//重载==inline bool operator==(const list\_iterator& it) const;

·//重载!=inline bool operator!=(const list\_iterator& it) const;

此处不完全列出所有功能，但基本按照C++迭代器标准实现其功能。

·学生类设计：数据成员包括如下图所示：



其中考虑用String类储存考号，为了处理以’0’打头的考号出现

主要功能包括：

·//重载输入流运算符 friend istream& operator>>(istream& in, Stu& s);

·//重载输出流运算符friend ostream& operator<<(ostream& out, const Stu& s);

·考试系统数据库类设计：数据成员包括学生类为节点数据的链表List<Stu> data;

主要功能包括：

·//插入正确位置void insertProperly(Stu& s);

·//内部查找函数，返回迭代器

List<Stu>::list\_iterator mySearch(const Stu& s);

List<Stu>::list\_iterator mySearch(const String& num);

·//初始化bool ini();

·//查找void search();

·//插入void insert();

·//修改void change();

·//保存入文件void save();

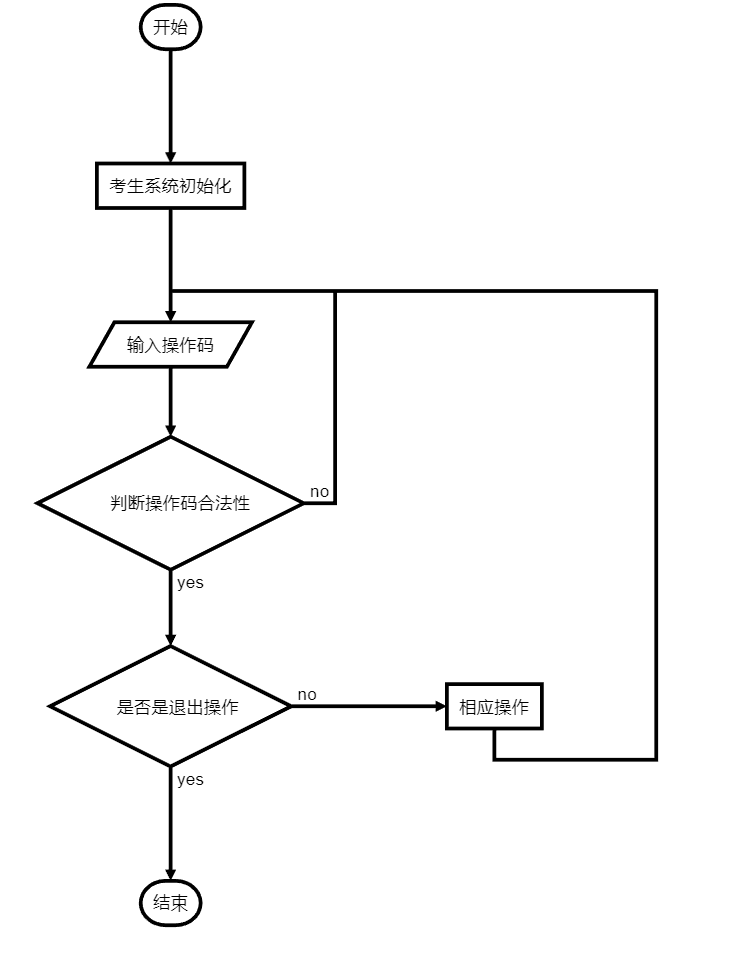
·//删除void remove();

·//统计void statistics();

3 项目实现

3.1 总体实现

3.1.1 总体实现流程图



3.1.2 总体实现思路

首先进行考试报名系统类的初始化，初始化成功则选择系统的功能，不成功则直接退出。选择系统功能执行操作后，（除了选择退出系统后）重新回到选择系统的功能。这里需要判断输入的选择是否在列出范围内。最后输入前十条考生的信息。

我们认为能区分考生的独特的信息的考生的考号，因此利用考号进行查找和比较两个考生的差别。

同时，每项功能均具备首先查看考试报名系统中是否存在数据，若不存在，则直接退出，并输出提示。

3.1.3 总体实现主要代码

if (data.empty()){

cout << "考生信息系统中暂无数据，XXXXXXX" << endl;

return;

}

说明：代码展示时为了整洁美观，做了一定调整。

主要代码如下：

while (loop){

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，6为保存数据，0为退出系统）\n";

cin >> op;failOp = false;

switch (op[0]){

case '0':

loop = false;

failOp = true;break;

case '1':

stuSys.insert();break;

case '2':

stuSys.remove();break;

case '3':

stuSys.search();break;

case '4':

stuSys.change();break;

case '5':

stuSys.statistics();break;

case '6':

stuSys.save();break;

default:

failOp = true;

cin.ignore(65536, '\n');

cin.clear();

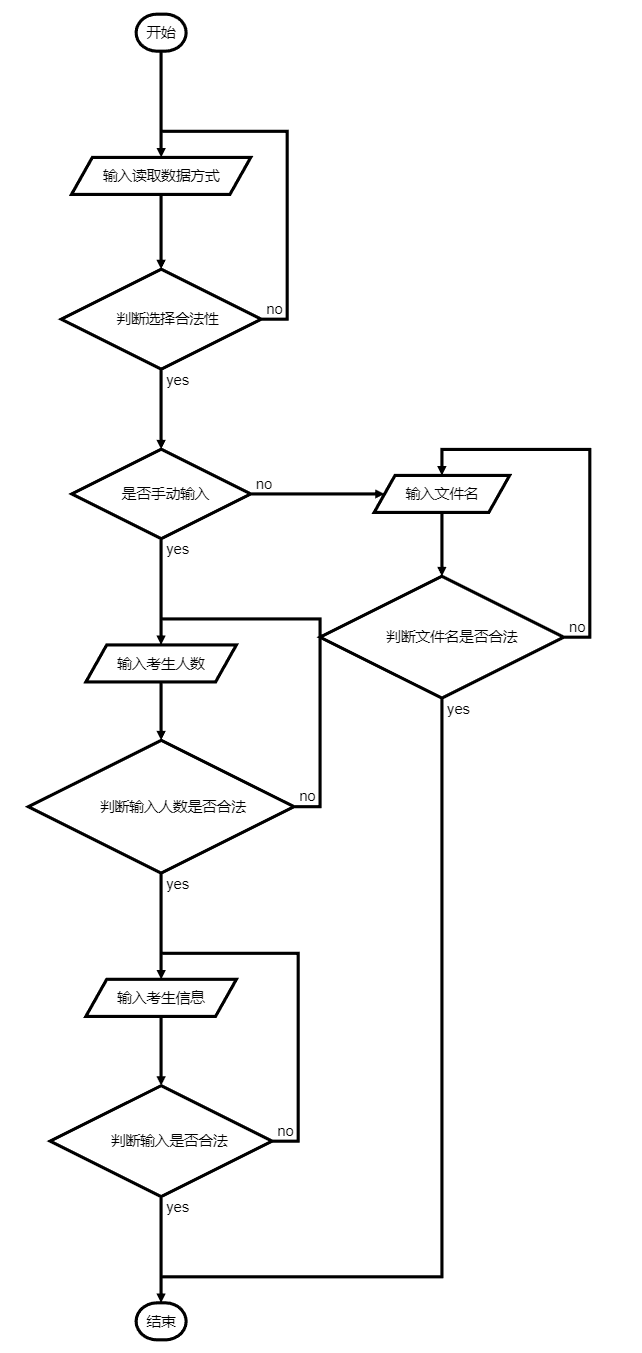
cout << "非法操作，请重新输入" << endl;

cout << "温馨提示：1为插入，2为删除，3为查找，4为修改，5为统计，6为保存数据，0为退出系统" << endl;break;

}

3.2 初始化实现

3.2.1 初始化实现流程图



3.2.2 初始化实现思路

给予用户选择，分为同目录下的文件中读取数据和手动输入数据，由于打开文件可能失败，在此需要判断。

手动输入数据首先需要先输入考生信息的条数，利用循环依次输入考生的考号、姓名、性别、年龄和报考专业。其中值得注意的是，除了年龄以外其他数据成员的类型为String(自己写的字符串的类)，因此需要对年龄进行输入错误的判断和负数的判断。

对于每次读取到的学生类的实例化对象，插入链表的时候，需要根据考号判断插入的位置，最后的链表为根据考号的字典序从小到大排列。

3.2.3 初始化实现主要代码

//从文件中读取

if (op == 1){

String fileName;

cout << "请输入文件名称（如: Stu)" << endl;

cin >> fileName;

if (fileName.length() == 0)

fileName = "studentData.txt";

fileName += ".txt";//加上文件名后缀

ifstream file(fileName.c\_str(), ios::in);

if (!file.is\_open()){

cerr << "文件打开失败" << endl;

return false;

}

Stu temp;

while (file >> temp)

data.emplace\_back(temp);

file.close();

show();

return true;//打开文件成功

}

cout << "请输入考生人数" << endl;

while (1){//正确输入才退出循环

cin >> cnt;

if (cin.fail() || cnt < 0){

cout << "请输入正整数作为考生人数!" << endl;

cin.clear();

cin.ignore(65536, '\n');

}

else//正确输入

break;

}

if (cnt == 0)//一开始不想有学生

return true;

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

Stu temp;

for (int i = 0; i < cnt; ++i){

cin >> temp;

//考号已经存在，Search()找不到则返回NULL

if (mySearch(temp) != NULL)

cout << "数据中已存在该考生!" << endl;

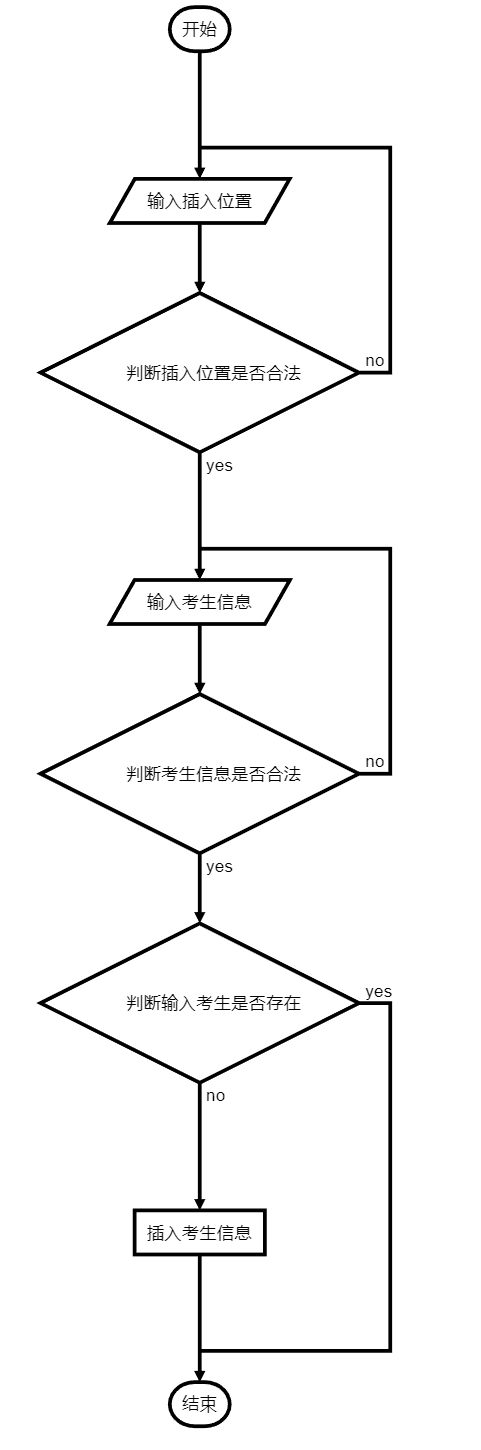
else

insertProperly(temp);

}

3.3 插入实现

3.3.1 插入实现流程图



3.3.2 插入实现思路

首先输入正确的插入的位置，从0到考生信息总条数。然后输入考生信息，这时候我们需要调用查找函数，若已经存在该项考生信息，则插入失败，并输出提示，否则根据迭代器进行信息的插入。

3.3.3 插入实现主要代码

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

Stu temp;

cin >> temp;

//找不到则返回NULL

auto tempIter = mySearch(temp);

//获得插入位置

List<Stu>::list\_iterator iter = data.begin();

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

++iter;

//如果没有考生

if (tempIter == NULL){

if (loc == Size)

data.emplace\_back(temp);

else

data.insert(iter, temp);

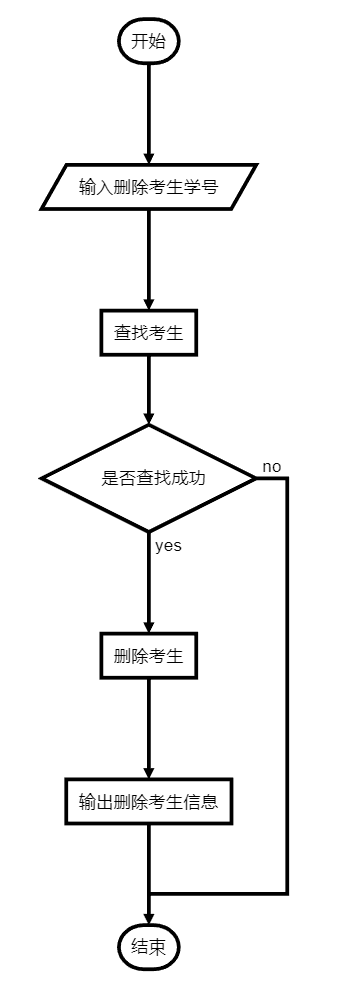
}

else

cout << "数据中已存在该考生!不能再次插入!" << endl << endl;

3.4 删除实现

3.4.1 删除实现流程图



3.4.2 删除实现思路

输入需要删除的考生的学号，调用查找函数，找到则利用迭代器删除，否则删除失败，输出删除失败的信息提示。

3.4.3 删除实现主要代码

if (data.empty()){

cout << "考生信息系统中暂无数据，不能删除" << endl;

return;

}

cout << "请输入要删除的考生的考号: ";

String num; cin >> num;

auto tempIter = mySearch(num);

if (tempIter == NULL)

cout << "不存在考号为" << num << "的考生！" << endl << endl;

else{

cout << "你删除的考生的信息是: ";

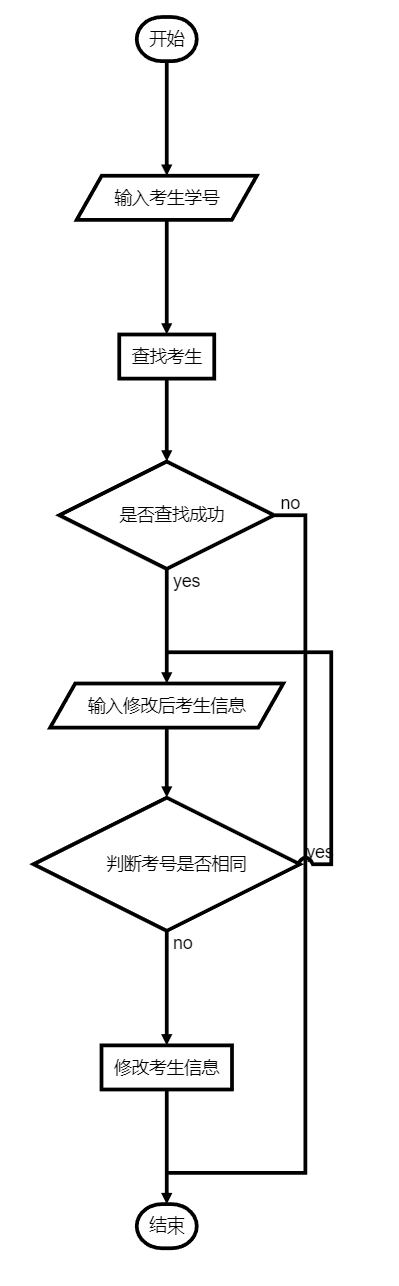
cout << \*tempIter << endl << endl;

data.remove(tempIter);

}

3.5 修改实现

3.5.1 修改实现流程图



3.5.2 修改实现思路

输入需要修改的考生信息的考号，调用查找函数，得到查找到的迭代器，查找不到则输出提示信息，否则输入修改后的考生信息，这里需要注意修改前后的考号应该为一致的，修改不能改变该学生的特有的和其他学生区分的标识，否则需要重新输入。

3.5.3 修改实现主要代码

auto tempIter = mySearch(num);

if (tempIter == NULL){

cout << "不存在考号为" << num << "的考生！" << endl << endl;

}

else{

while (1){

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

cin >> \*tempIter;

if (tempIter->num == num)

break;

else{

cout << "考号不可更改，请重新输入" << endl;

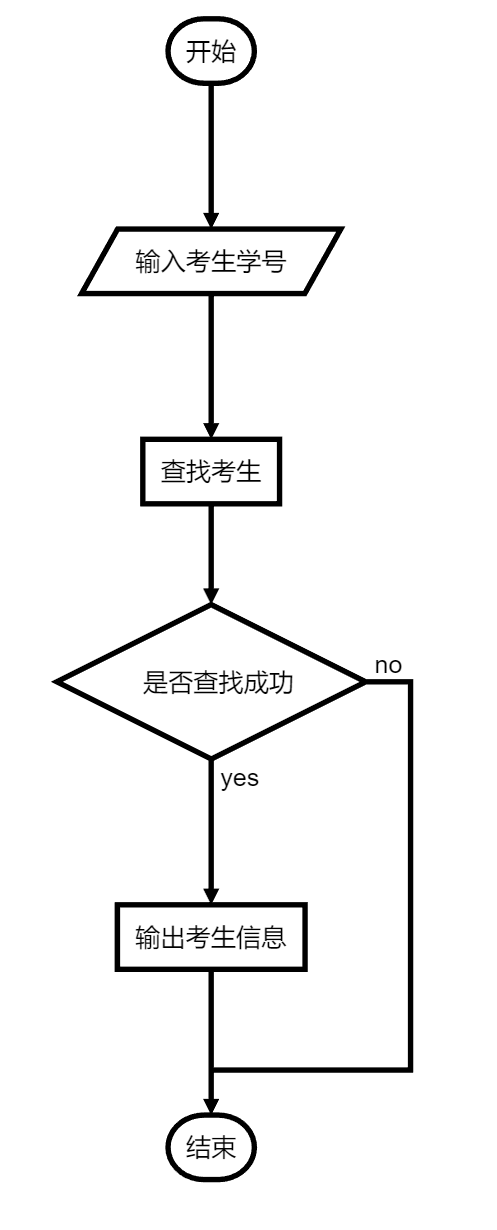
cin.clear();

cin.ignore(65536, '\n');

}}}

3.6 查找实现

3.6.1 查找实现流程图



3.6.2 查找实现思路

通过输入得到的考号，查询考生的信息。调用查找的函数返回迭代器，迭代器为空则输出提示，否则输出考生的所有信息

3.6.1 查找实现主要代码

cout << "请输入要查找的考生的考号: ";

String num;

cin >> num;

auto tempIter = mySearch(num);

if (tempIter == NULL)

cout << "不存在考号为" << num << "的考生！" << endl << endl;

Else{

cout << "查找信息如下: " << endl;

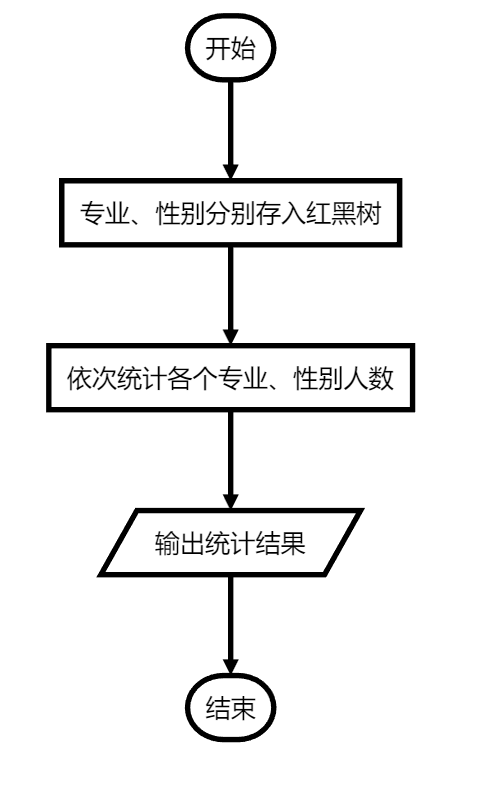
printf("考号\t 姓名\t\t性别\t 年龄\t 报考类别\n");

cout << \*tempIter << endl << endl;

}

3.7 统计实现

3.7.1 统计实现流程图



3.7.2 统计实现思路

将实例化Map<String, int>，分为性别和报考类别的统计人数，遍历链表，放入两个Map对象。每遍历到一个性别和报考类别，对其映射的个数增加，从0开始。利用红黑树优良的性质，我们通过String的比较方法，利用迭代器从头到尾输出（中序遍历），得到的即为按String有序的排列。并且通过红黑树的性质，改进了AVL树，BST树的不足，使得增删改查平均上的性能较好。

3.7.3 统计实现主要代码

for (auto it = data.begin(); it != data.end(); ++it){

++gender[it->gender];

++type[it->type];}

cout << "\n性别统计: " << endl;

for (auto it = gender.begin(); it != gender.end(); ++it)

cout << it->first << it->second << endl;

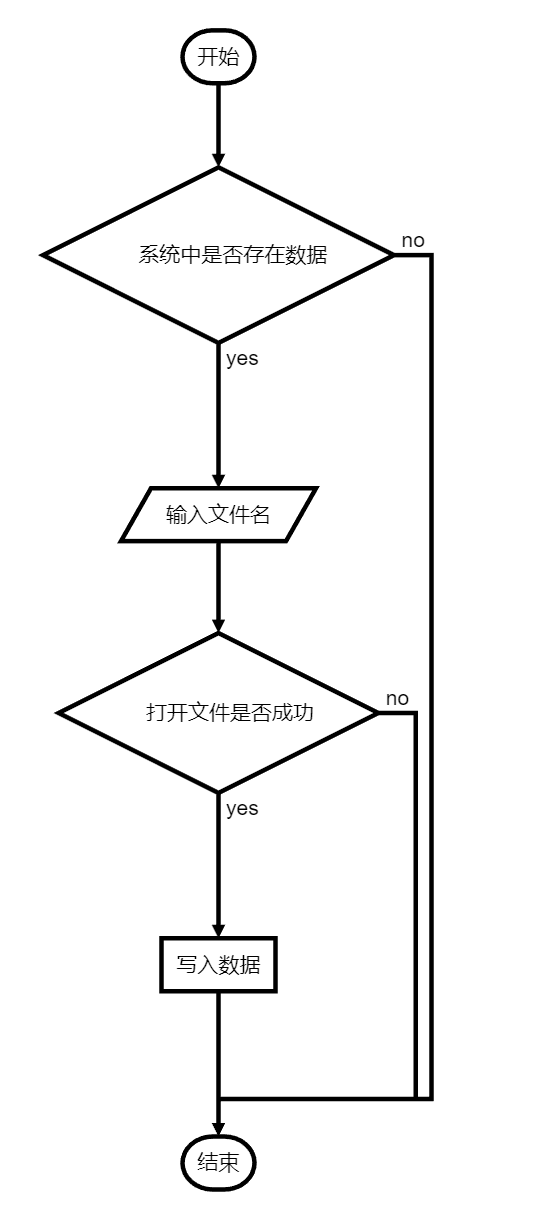
cout << "专业统计: " << endl;

for (auto it = type.begin(); it != type.end(); ++it)

cout << it->first << it->second << endl;

3.8 存入文件实现

3.8.1 存入文件实现流程图



3.8.2 存入文件实现思路

若系统中存在考生数据，通过输入文件名，将数据库中的每条数据输入至打开的文件中。

3.8.3 存入文件实现主要代码

ofstream file(fileName.c\_str(), ios::out);

if (!file){

cerr << "文件打开失败" << endl;

exit(1);}

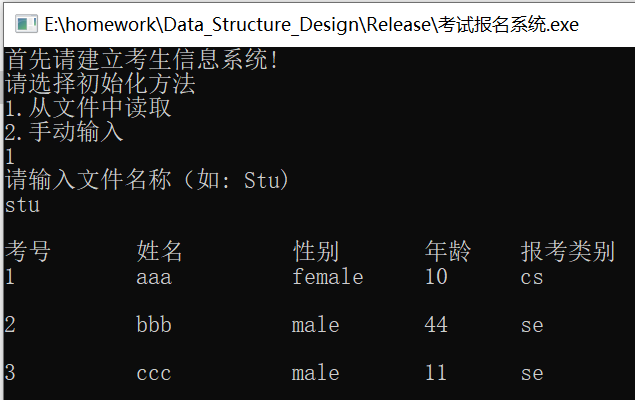
for (auto it = data.begin(); it != data.end(); ++it){

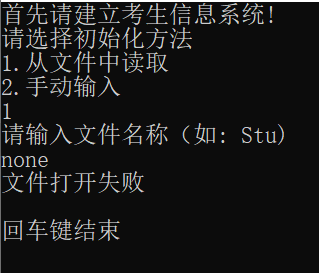
file << it->num << ' ' << it->name << ' ' << it->gender<< ' ' << it->age << ' ' << it->type << endl;}

4 项目测试

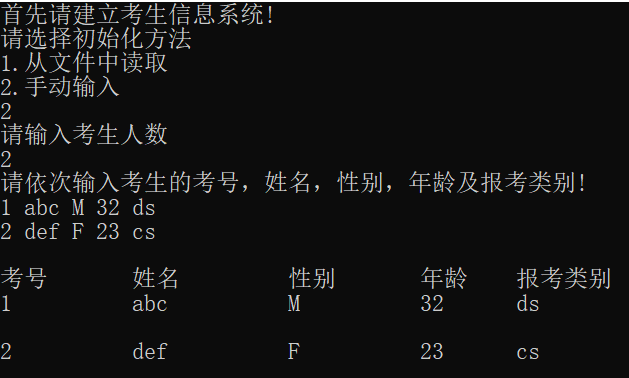
4.1 初始化测试

4.1.1 文件读入

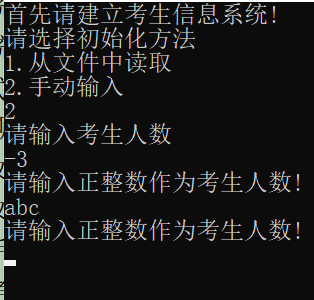
4.1.2 文件不存在



4.1.3 手动输入

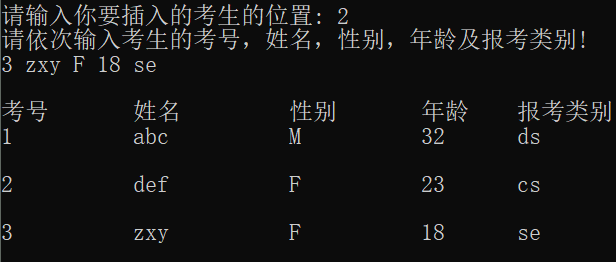


4.1.4 考生人数错误

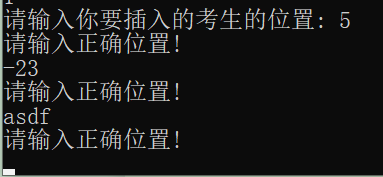


4.2 插入测试

4.2.1 正确位置插入

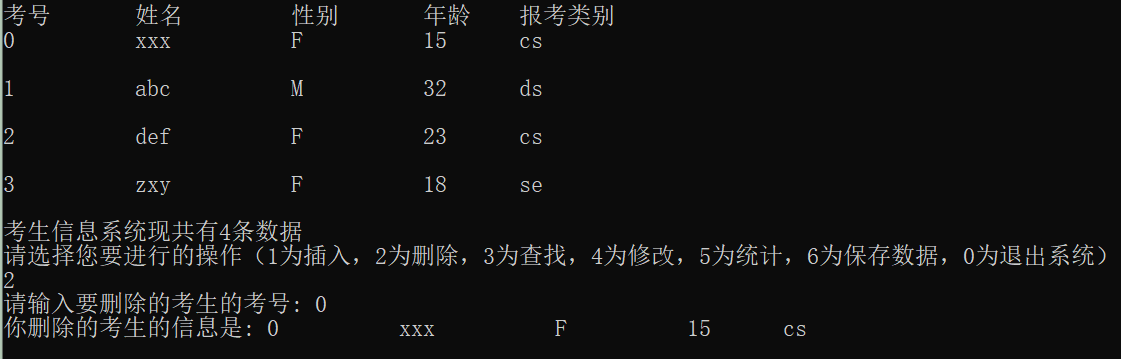


4.2.1 非法位置插入

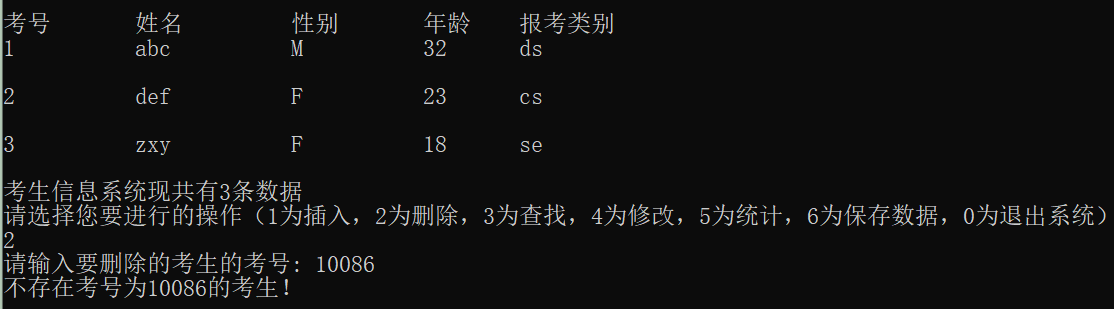


4.3 删除测试

4.3.1 正确输入存在的考生考号

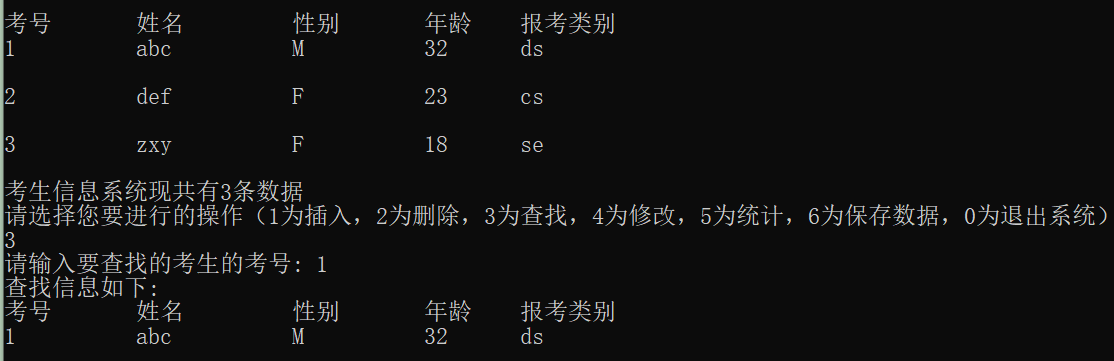


4.3.2 输入不存在的考生考号

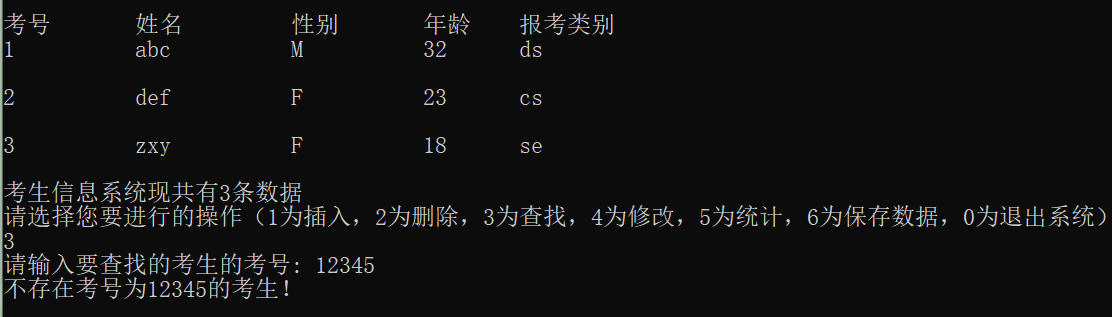


4.4 查找测试

4.4.1 输入存在的考生考号

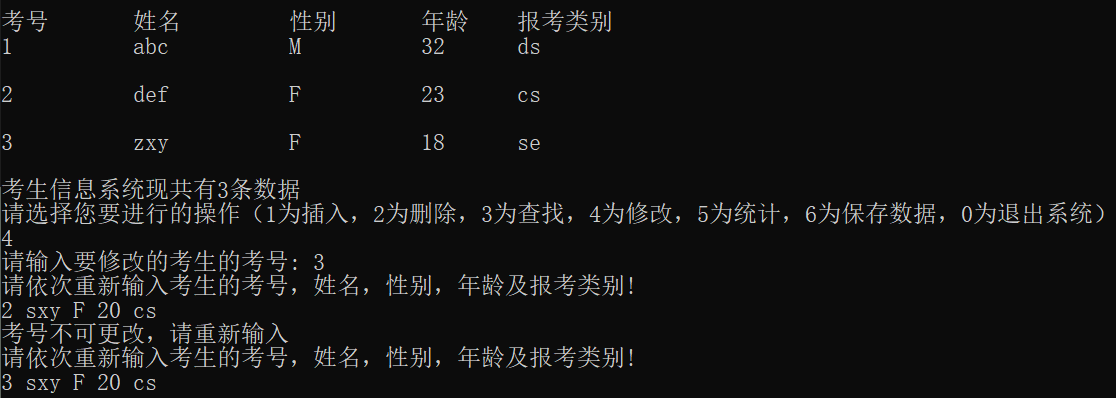


4.4.2 输入不存在的考号考号

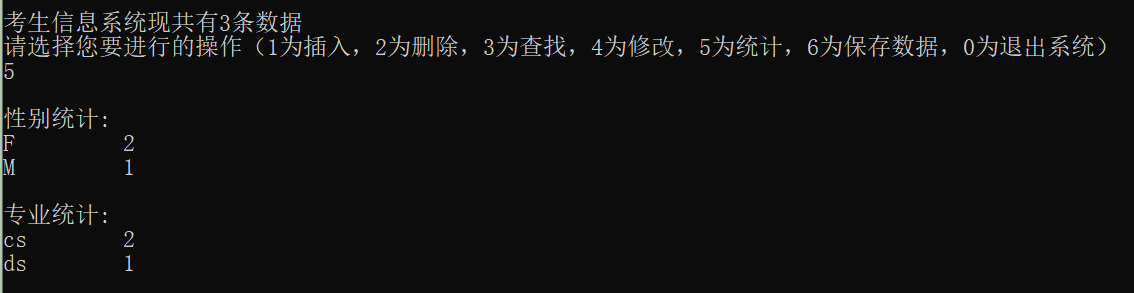


4.5 修改测试

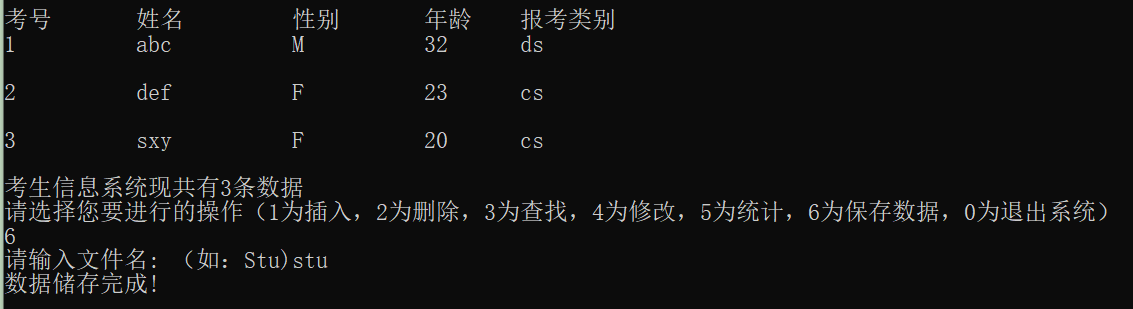
4.5.1 修改前后考号需要匹配，（输入不存在考号测试与上类似，不再展示）

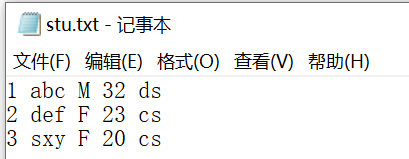


4.6 统计测试



4.7 存入文件测试





4.8 输入操作错误测试

