项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 肖杨

学 号： 1950430

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 分析 3

2 设计 3

3 实现 8

4 测试 18

* 1分析

1.1背景分析

在当今的教育环境下，考试报名系统在学校管理当中扮演着一个很重要的角色。伴随着计算机技术的高速发展，相较于人工管理，使用计算机技术来进行考试信息的维护和管理，其具有管理容量上限高，管理快速，成本低等多个巨大优势，能极大提升考试报名管理的效率，有利于学校的发展。可以说，开发一个高效精准的考试报名系统是有必要，也有极大益处的。

1.2功能分析

一个完整的考试管理系统需要有输入、输出、插入、删除、修改、退出的功能。考试系统应当能将考生的对应信息进行输入并储存下来，同时其还应该能对已经储存的信息进行显示，对已经储存的信息进行维护（插入，删除，修改），保证不存在废弃信息以及方便考生对自己的信息进行修正。最后，这个考试管理系统还应该可以正常关闭。

* 2设计

2.1 数据结构设计

如上功能分析所述，该系统要求大量的增加、删除、修改操作，常用的可用数据结构包括数组，链表，BST，哈希表。考虑到需要对数据进行插入和删除，而数组在进行这种操作的时候很可能移动整个数组，太过耗费时间，而链表在进行查询操作时对于花费时间上界没有保证，同时考虑到查找单个学生的操作也较为花费时间，而一般情况下考生数量不会超出一定值，因此选择使用哈希表。每次修改、删除、插入操作都会进行一次查找操作，散列表的查找操作时间复杂度为O(1)。

2.2 类结构设计

类中选择使用一个哈希表进行储存，并基于哈希表实现类的其他操作。

由于考生数目未知，暂时选择使用大小为10099（大于10000的一个质数）的哈希表，同时使用链地址法处理哈希冲突，对于50000以内的考生数目都有很好的支持，如若需要支持更多考生可以另行修改哈希表大小。

类中所有数据均选择string储存，支持考生考号、姓名、性别、年龄、科目可以使用任何格式的输入。

2.3 成员与操作设计

哈希表与哈希节点：使用类模板，value支持任意数据类型

template<class v>

class Node

{

public:

friend class unordered\_map<v>;

Node<v>(string ikey, v ivalue);

Node<v>();

v getValue();

void setValue(v newvalue);

~Node<v>();

protected:

string key;

v value;

Node<v>\* next;

};

公有操作：

Node<v>(string ikey, v ivalue);

//含参构造函数

Node<v>();

//默认构造函数

v getValue();

//非友元类返回数据

void setValue(v newvalue);

//修改内部数据

template<class v>

class unordered\_map

{

public:

unordered\_map(int size);

~unordered\_map();

Node<v>\* find(string key);

void remove(string key);

void insert(string key, v value);

v& operator [] (string key);

void printValue();

private:

int hashSize;

Node<v>\* hashnode;

int hash(string key);

};

公有操作：

unordered\_map(int size);

//构造函数

Node<v>\* find(string key);

//查找是否存在，若存在返回指向元素的指针，否则返回nullptr

void remove(string key);

//移除键为key的元素

void insert(string key, v value);

//插入键值对于哈希表内

v& operator [] (string key);

//重载中括号，若不存在相应键，则会创建一个元素并返回引用

void printValue();

//输出哈希表储存的的所有非空值

私有操作：

int hash(string key);

//哈希化函数

说明：哈希表使用连续的结点而非地址，用于统一操作

考试系统类：包含一个内部类和考试系统的所有操作、成员

学生类：作为值存入哈希表，内部包含学生的所有数据

class TestSystem

{

private:

class Student

{

public:

string testnumber;

string name;

string sex;

string age;

string subject;

Student(string inumber, string iname, string isex, string iage, string isubjext):

testnumber(inumber), name(iname), sex(isex), age(iage), subject(isubjext) {}

Student() = default;

friend ostream& operator <<(ostream& os, Student stu)；

};

int studentNumber;

unordered\_map<Student> dataSystem;

public:

TestSystem();

bool getCommend();

bool find();

bool change();

bool insert();

bool remove();

void printSystem();

};

私有成员：

int studentNumber;

//当前学生数据

unordered\_map<Student> dataSystem;

//哈希表

公有操作：

TestSystem();

//初始化考试系统

bool getCommend();

//初始化后，直到退出系统，持续获取命令

bool find();

//执行查找操作并输出

bool change();

//执行修改操作并输出

bool insert();

//执行插入操作并输出

bool remove();

//执行插入操作并输出

void printSystem();

//输出整个考试系统的数据

2.4 系统设计

系统首先调用TestSystem的构造函数实现考试系统的初始建立，循环getCommand()函数来执行相应的成员函数。当希望执行的命令为退出时会退出循环并退出系统。

* 3实现

3.1 插入功能的实现

3.1.1 插入流程

首先输入考生各项信息，建立一个新考生。

之后检查考生号是否存在，若已存在，则输出错误信息并返回。

若不存在，则将考生号于对应考生存入哈希表。

3.1.2核心代码

getCommand():

case 1:

cout << "请按顺序输入考生考号、姓名、性别、年龄、科目" << endl;

if (insert())

{

studentNumber++;

}

return true;

insert():

string testnumber, name, sex, age, subject;

cin >> testnumber >> name >> sex >> age >> subject;

Student temp(testnumber, name, sex, age, subject);

if (dataSystem.find(testnumber))

{

cout << "不允许相同考号考生存在，插入失败" << endl;

return false;

}

else

{

dataSystem.insert(testnumber, temp);

cout << "考生插入成功,插入的考生信息为:" << temp;

return true;

}

哈希表中的插入函数

template<class v>

void unordered\_map<v>::insert(string key, v value)

{

Node<v>\* temp = find(key);

if (temp)

{

temp->value = value;

}

else

{

int index = hash(key);

temp = hashnode + index;

while (temp->next)

{

temp = temp->next;

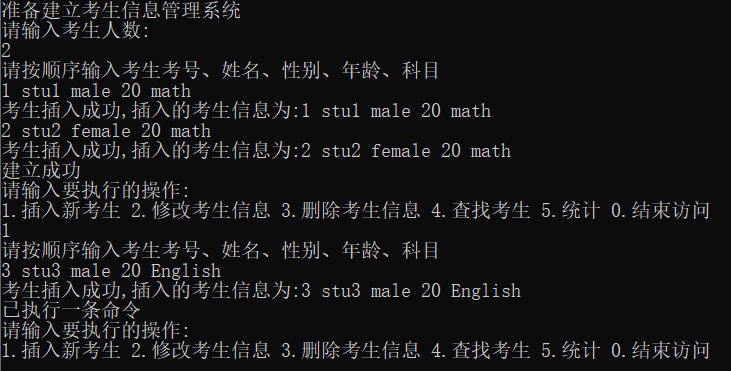
}

temp->next = new Node<v>(key, value);

}

}

3.1.3 插入功能截屏示例



3.2 删除功能的实现

3.2.1 删除流程

首先输入考生号，并储存。

之后检查考生号是否存在，若已存在，则在哈希表中删除该考生信息。

若不存在，则输出错误信息并返回。

3.2.2核心代码

getCommand():

case 3:

cout << "请输入想要删除信息的考生号" << endl;

remove();

return true;

remove():

string number;

cin >> number;

if (auto temp = dataSystem.find(number))

{

cout << "删除成功，删除的信息为" << temp->getValue();

studentNumber--;

dataSystem.remove(number);

return true;

}

else

{

cout << "删除失败，未找到对应考生" << endl;

return false;

}

哈希表中的删除函数

template<class v>

void unordered\_map<v>::remove(string key)

{

int index = hash(key);

Node<v>\* temp = hashnode + index;

while (temp->next)

{

if (temp->next->key == key)

{

Node<v>\* removeptr = temp->next;

temp->next = removeptr->next;

removeptr->next = nullptr;

delete removeptr;

return;

}

else

{

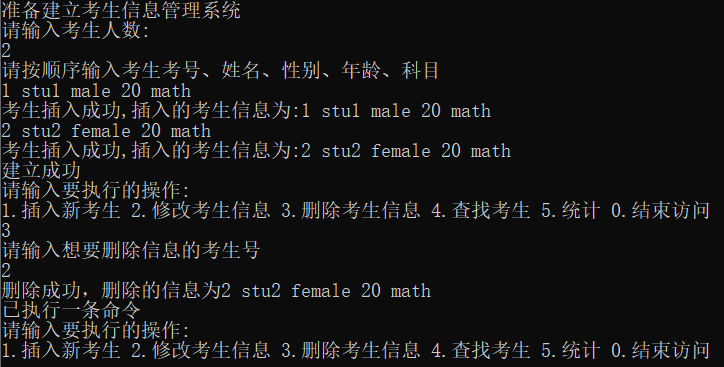
temp = temp->next;

}

}

}

3.2.3 删除功能截屏示例



3.3 修改功能的实现

3.3.1 修改流程

首先输入考生号，并储存。

之后检查考生号是否存在，若已存在，则在哈希表中查找该考生信息，之后输入修改后的信息。此处允许修改学号，若修改后的学号与之前不同且该学号未加入哈希表，则会删除原学生并添加至新学生，否则会报错并修改失败。若不修改学号，则会正常修改其他信息

若不存在，则输出错误信息并返回。

3.3.2核心代码

getCommand():

case 2:

cout << "请输入想要修改信息的考生号" << endl;

change();

return true;

change():

string number;

cin >> number;

if (dataSystem.find(number))

{

cout << "已找到考生,请输入修改后的信息" << endl;

string testnumber, name, sex, age, subject;

cin >> testnumber >> name >> sex >> age >> subject;

Student temp(testnumber, name, sex, age, subject);

if (testnumber == number)

{

dataSystem[number] = temp;

cout << "修改成功,修改后的信息为:" << temp;

return true;

}

else

{

if (dataSystem.find(testnumber))

{

cout << "不允许出现相同考号考生，修改失败" << endl;

return false;

}

else

{

dataSystem.remove(number);

dataSystem.insert(testnumber, temp);

cout << "修改成功,修改后的信息为:" << temp;

return true;

}

}

}

else

{

cout << "未找到相应考生，返回上一级" << endl;

return false;

}

哈希表中的修改函数

template<class v>

v& unordered\_map<v>:: operator [] (string key)

{

Node<v>\*temp = find(key);

if (temp)

{

return temp->value;

}

else

{

temp = hashnode + hash(key);

temp->next = new Node<v>(key, v());

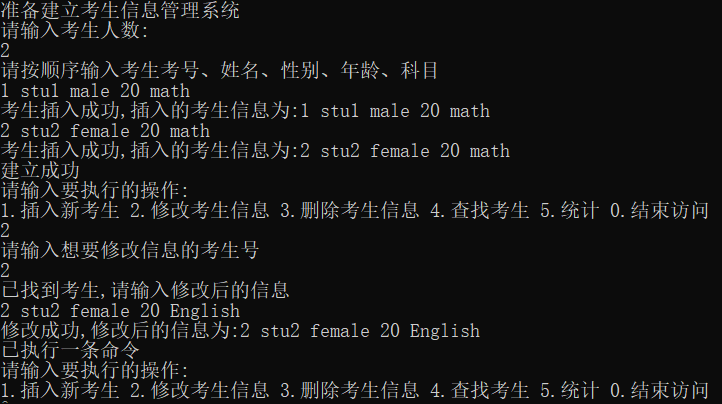
return temp->next->value;

}

}

不修改学号时选择使用中括号访问，方便操作。

3.3.3 修改功能截屏示例



3.4 查找功能的实现

3.4.1 查找流程

首先输入考生学号，并储存。

之后检查考生号是否存在，若存在，则输出考生信息。

若不存在，则输出错误信息并返回。

3.4.2核心代码

getOrder():

case 4:

cout << "请输入想要查找的考生号" << endl;

find();

return true;

find():

string number;

cin >> number;

auto temp = dataSystem.find(number);

if (temp)

{

cout << "已找到考生,考生信息为:" << temp->getValue();

return true;

}

else

{

cout << "未找到考生,返回上一级" << endl;

return false;

}

哈希表中的查找函数

template<class v>

Node<v>\* unordered\_map<v>::find(string key)

{

int index = hash(key);

Node<v>\* temp = hashnode + index;

while (temp)

{

if (temp->key == key)

{

return temp;

}

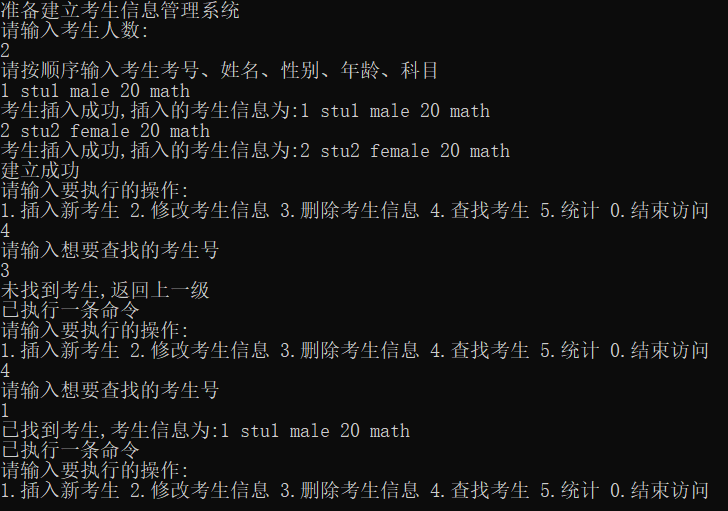
temp = temp->next;

}

return nullptr;

}

3.4.3 查找功能截屏示例



3.5 统计功能的实现

3.5.1 统计流程

输出一段内部信息后，输出整个哈希表，最后输出考生数。

3.5.2核心代码

getOrder():

case 5:

cout << "已经在数据库内的考生为:" << endl;

printSystem();

return true;

printSystem():

cout << "考生号 姓名 性别 年龄 科目" << endl;

dataSystem.printValue();

cout << "共" << studentNumber << "名考生，已全部显示" << endl;

哈希表中的输出函数

template<class v>

void unordered\_map<v> ::printValue()

{

Node<v>\*temp = hashnode;

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

{

if (temp->next)

{

Node<v>\* print = temp->next;

while (print)

{

cout << print->value;

print = print->next;

}

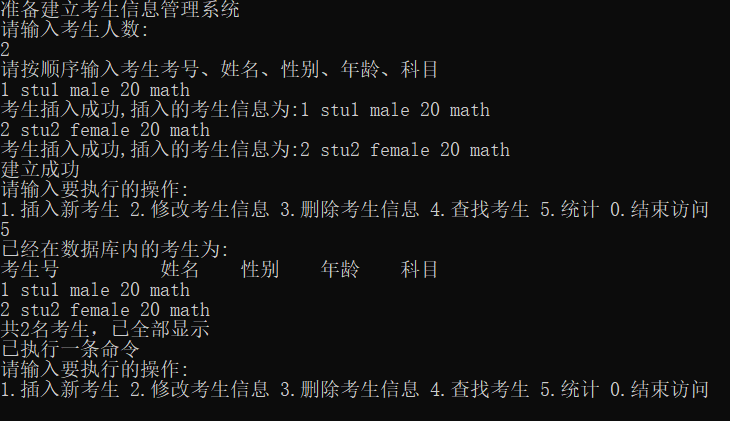
}

temp++;

}

}

3.5.3 统计功能截屏示例



3.6 总体功能的实现

3.6.1 工作流程

首先建立考试系统。

之后持续接受命令，直到收到退出指令。

3.6.2核心代码

int main()

{

TestSystem TS;

while (TS.getCommend())

{

cout << "已执行一条命令" << endl;

}

system("pause");

return 0;

}

getOrder():

int order = 0;

cout << "请输入要执行的操作:" << endl

<< "1.插入新考生 2.修改考生信息 3.删除考生信息 4.查找考生 5.统计 0.结束访问" << endl;

cin >> order;

switch (order)

{

case 0:

cout << "已离开考生系统" << endl;

return false;

case 1:

cout << "请按顺序输入考生考号、姓名、性别、年龄、科目" << endl;

if (insert())

{

studentNumber++;

}

return true;

case 2:

cout << "请输入想要修改信息的考生号" << endl;

change();

return true;

case 3:

cout << "请输入想要删除信息的考生号" << endl;

remove();

return true;

case 4:

cout << "请输入想要查找的考生号" << endl;

find();

return true;

case 5:

cout << "已经在数据库内的考生为:" << endl;

printSystem();

return true;

default:

cout << "请输入正确的指令!" << endl;

return true;

break;

}

* 4测试

4.1 功能测试

4.1.1 插入功能测试

**测试用例**：3 stu3 男 22 网络工程师

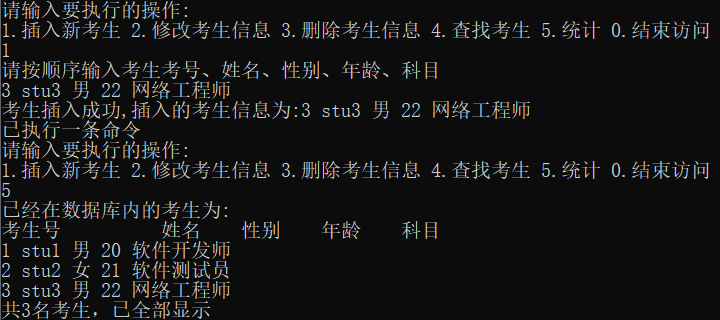
**预期结果**：

1 stu1 男 20 软件开发师

2 stu2 女 21 软件测试员

3 stu3 男 22 网络工程师

**实验结果：**

4.1.2 删除功能测试

**测试用例：**删除考号为4的考生

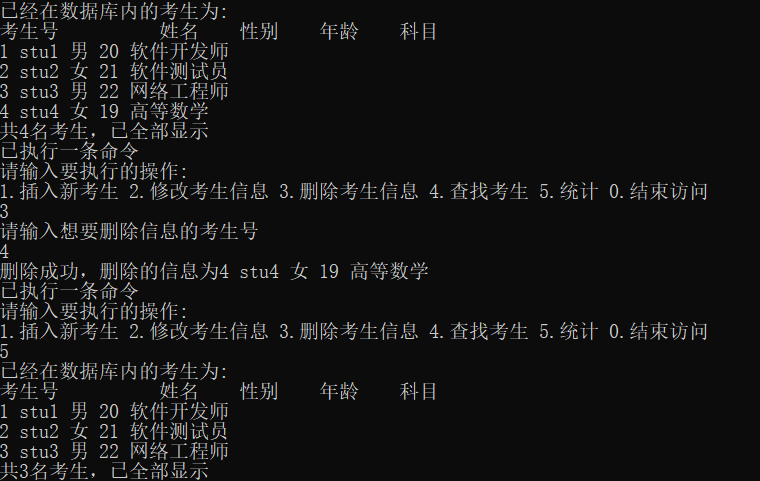
**预期结果：**

1 stu1 男 20 软件开发师

2 stu2 女 21 软件测试员

3 stu3 男 22 网络工程师

**实验结果：**

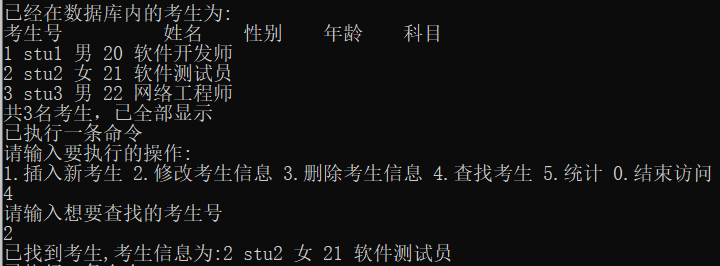
4.1.3 查找功能测试

**测试用例：**查找考号为2的考生

**预期结果：**

2 stu2 女 21 软件测试员

**实验结果：**

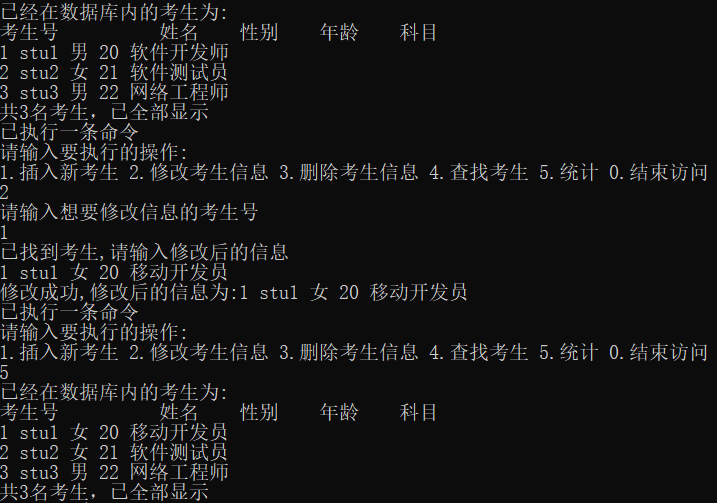
4.1.4 修改功能测试

**测试用例：**将考号1修改为性别女，年龄20，报考种类移动开发员。

**预期结果：**

1 stu1 女 20 移动开发员

**实验结果：**

4.1.5 统计功能测试

**测试用例：**统计当前数据

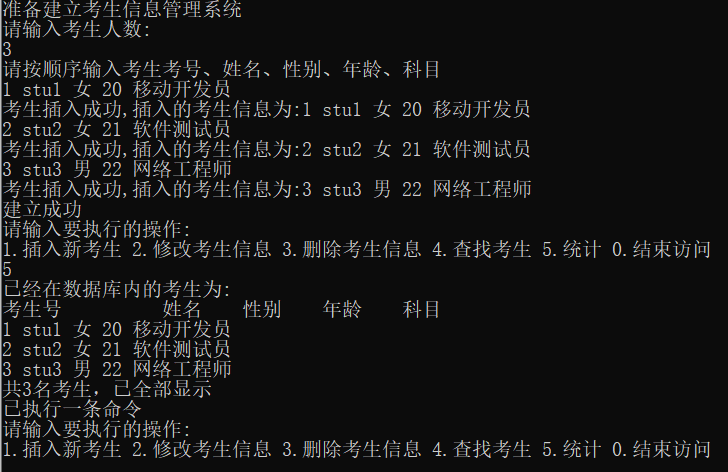
**预期结果：**

1 stu1 女 20 移动开发员

2 stu2 女 21 软件测试员

3 stu3 男 22 网络工程师

**实验结果：**



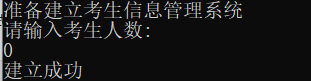
4.2 边界测试

4.2.1 初始化无输入数据

**测试用例：**初始无输入数据

**预期结果：**程序运行正常，允许初始库为空。

**实验结果：**



4.2.2 其他可能情况

由于所有数据均采取string保存，且哈希表使用链地址法实现，理论上除内存溢出外，不存在其他边界条件，使用时应注意考生数目极大的考试系统需要足够大的内存。

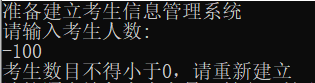
4.3 错误测试

4.3.1 考生人数错误

**测试用例：**输入考生人数为负数

**预期结果：**程序给出提示信息，不允许初始值为负数，需重新建立库。

**实验结果：**

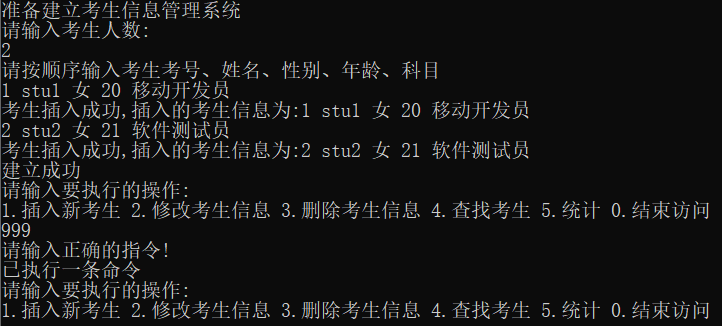


4.3.2 操作码错误

**测试用例：**输入操作码错误

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

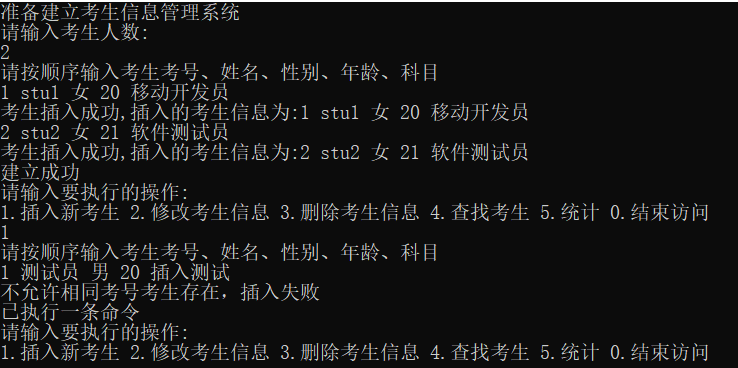
**实验结果：**

4.3.3 插入考生的考号已经存在

**测试用例：**哈希表里有两条记录，向哈希表里插入已经存在的考号

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

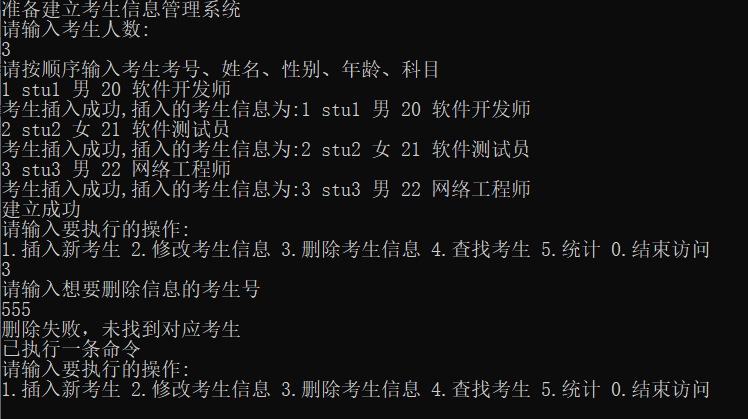


4.3.4 删除考号不存在

**测试用例：**要删除的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

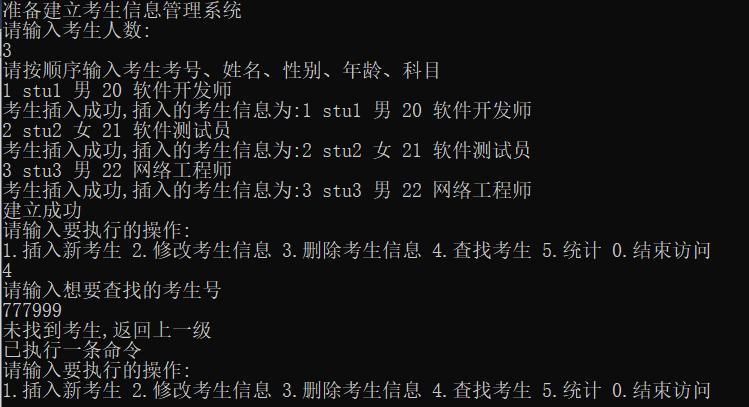


4.3.5 查找考号不存在

**测试用例：**要查找的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

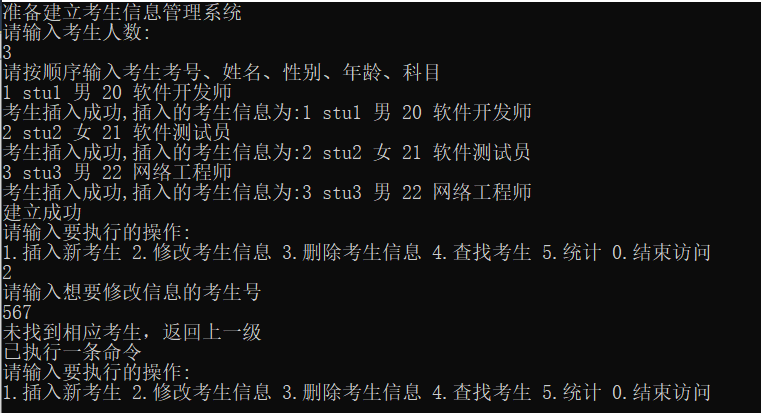


4.3.6 修改考号不存在

**测试用例：**要修改的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**



4.3.7 修改考号后与已存在考号相同

由于允许修改考号，可能存在该种情况。

**测试用例：**要修改的考号存在，修改后与已存在考号相同

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

