项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 王加炜

学 号： 2150265

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 分析

## 1.1背景分析

考试报名工作给各高校报名工作带来了新的挑战，给教务管理部门增加了很大的工作量。本项目是对考试报名管理的简单模拟，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。

## 1.2功能分析

本项目旨在完成对于学生信息的录入，插入，删除，查找和修改功能的实现。

本项目的学生信息采用以链表为形式的数据结构存储，原因在于链表结构非常适合元素的增删查补功能，相较数组更加的简便快捷。而学生信息由于包含学号，性别等包含各种数据类型的元素，因此采用结构体的形式存储信息。同时，我新创建了一个类用来执行初始化输入和操作码输入的功能。同时，我也考虑到了程序的健壮性，考虑了各种错误信息的特判，尽可能地实现对于用户更加友好且可操作的用户界面。

# 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，相比起数组极为繁琐的增删操作，链表进行增加、删除等操作十分简便，因此考虑使用链表数据结构存储学生信息。

## 2.2类结构设计

经典的链表一般由链表头指针和各个结点组成。而每个结点中存储了data信息和下一个地点的link地址信息。由于本题的数据信息存储了student的个人信息，而个人信息并不是由单一的数据成员组成，因此考虑用结构体来存储学生的信息，将该结构体指针存到data里面，作为各个节点的信息存储下来。

## 2.3成员与操作设计

学生类：

struct student//存储学生信息

{

int rank;

char name[6];

char gender[3];

int age;

char job[11];

};

结点类：

template <class T>

struct newnode//定义结点类，包含数据和下一个结点的地址信息

{

T\* stu ;

newnode<T>\* link;

newnode(newnode<T>\* ptr=NULL)//结构体的构造函数,注意这里空的条件是必须加的，否则无法判断循环终止，而存储的data则必须初始化

{

stu = new T;

link = ptr;

}

};

链表类：

template <class T>

class list//存储包含所有学生信息的链表

{

protected:

newnode<T>\* first;

public:

int num=0;

list()

{

first = new newnode<T>;

}

~list()

{

empty();//删除链表

delete first;//删除链表首指针

}

void empty();

void input();

void output();

bool insert(int i);

bool del(int i);

bool find(int i);

bool edit(int i);

newnode<T>\* locate(int i);

};

## 2.4系统设计

创建一个操作系统类用来完成初始操作。系统首先定义一个该类的实例化对象，然后调用该对象的Initinput()函数实现对屏幕的初始化。实际上在定义类的实例化对象的时候就已经完成了对于链表的初始化工作。然后调用multioperator()函数根据用户所输入的操作码（operatorCode）执行链表对应的成员函数。

class system\_operation//定义一个操作系统专门用来存放学生管理系统，以及使用初始化输入和菜单操作的功能

{

public:

system\_operation()//构造函数，初始化一个学生管理系统

{

tj = new list<student>;

}

void init\_input();

void multioperator();

private:

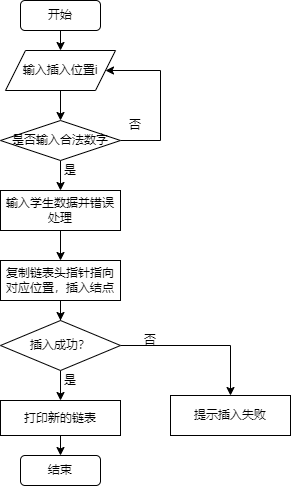
list<student>\* tj;

};

# 实现

## 3.1插入功能：

### 3.11流程图



### 3.12代码实现

bool list<T>::insert(int i)

{

newnode<T>\* current = locate(i);

if (current == NULL)

{

cout << "插入失败！！！" << endl;

return false;

}

newnode<T>\* newnode\_ = new newnode<T>;

if (newnode\_ == NULL)

{

cerr << "内存分配失败！！！" << endl;

exit(1);

}

if (num > defaultsize)

{

cout << "人数超过最大范围！！！" << endl;

return false;

}

newnode\_->link = current->link;

current->link = newnode\_;

cout << "请依次输入要插入的考生的考号，姓名，性别，年龄及报考类别！";

cin >> newnode\_->stu->rank;

cin >> newnode\_->stu->name;

cin >> newnode\_->stu->gender;

cin >> newnode\_->stu->age;

cin >> newnode\_->stu->job;

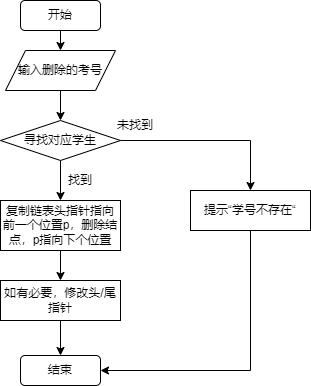
num++;

return true;

}

## 3.2删除功能：

### 3.2.1流程图

。

### 3.2.2代码实现

bool list<T>::del(int i)//删除操作，注意当删除第一个元素时

{

newnode<T>\* p=new newnode<T>;

newnode<T>\* current = new newnode<T>;

p = first;

while (p != NULL&&p->stu->rank != i)//如果找不到，即退出循环，只有一种可能，那就是指针为空，因为链表中如果指针有值说明该结点中有学生信息存在

{

current = p;

p = p->link;

}

if (p == NULL)

{

cerr << "您要删除的学生不存在" << endl;

return false;

}

else {

cout << "你删除的考生的信息是：";

cout << setw(8) << setiosflags(ios::left) << p->stu->rank << setw(8) << p->stu->name << setw(8) << p->stu->gender << setw(8) << p->stu->age << setw(8) << p->stu->job << endl;

current->link = p->link;

delete p;

num--;

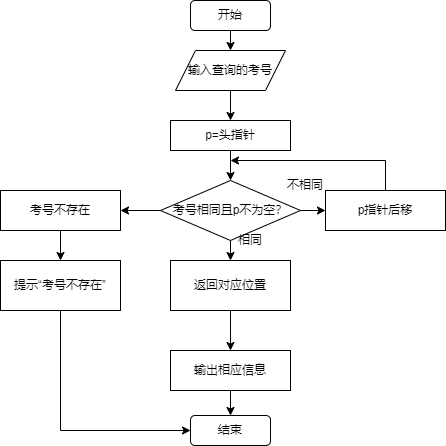
return true;

}

}

## 3.3查找功能：

### 3.3.1流程图



### 3.3.2代码实现

bool list<T>::find(int i)

{

newnode<T>\* current = first->link;

while (current != NULL && current->stu->rank != i )

{

current = current->link;

}

if (current == NULL)

{

cout << "您要查找的学生不存在" << endl;

return false;

}

cout << setw(8) << setiosflags(ios::left) << "考号" << setw(8) << "姓名" << setw(8) << "性别" << setw(8) << "年龄" << setw(8) << "报考类别" << endl;

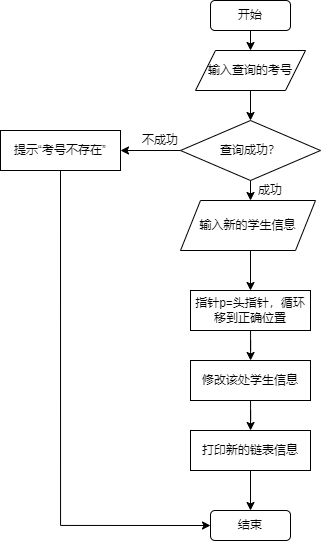
cout << setw(8) << setiosflags(ios::left) << current->stu->rank << setw(8) << current->stu->name << setw(8) << current->stu->gender << setw(8) << current->stu->age << setw(8) << current->stu->job << endl;

return true;

}

## 3.4修改功能：

### 3.4.1流程图



### 3.4.2代码实现

bool list<T>::edit(int i)

{

newnode<T>\* current = first->link;

while (current != NULL&&current->stu->rank != i )//这两句顺序不能反。因为空指针没有访问的数据，放在前半句直接就报错了

{

current = current->link;

}

if (current == NULL)

{

cerr << "您要修改的学生不存在" << endl;

return false;

}

cout << "请输入要修改的考生的考号，姓名，性别，年龄及报考类别！" << endl;

cin >> current->stu->rank;

cin >> current->stu->name;

cin >> current->stu->gender;

cin >> current->stu->age;

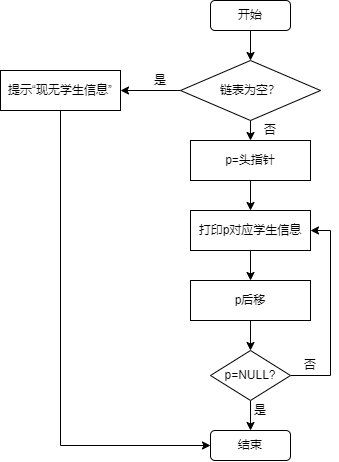
cin >> current->stu->job;

return true;

}

## 3.5统计功能：

### 3.5.1流程图



### 3.5.2代码实现

void list<T>::output()

{

newnode<T>\* current;

current = first;

if (first->link == NULL)

{

cout << "现无学生信息！" << endl;

}

else

{

cout << setw(8) << setiosflags(ios::left) << "考号" << setw(8) << "姓名" << setw(8) << "性别" << setw(8) << "年龄" << setw(8) << "报考类别" << endl;

while (current->link != NULL)

{

current = current->link;

cout << setw(8) << setiosflags(ios::left) << current->stu->rank << setw(8) << current->stu->name << setw(8) << current->stu->gender << setw(8) << current->stu->age << setw(8) << current->stu->job << endl;

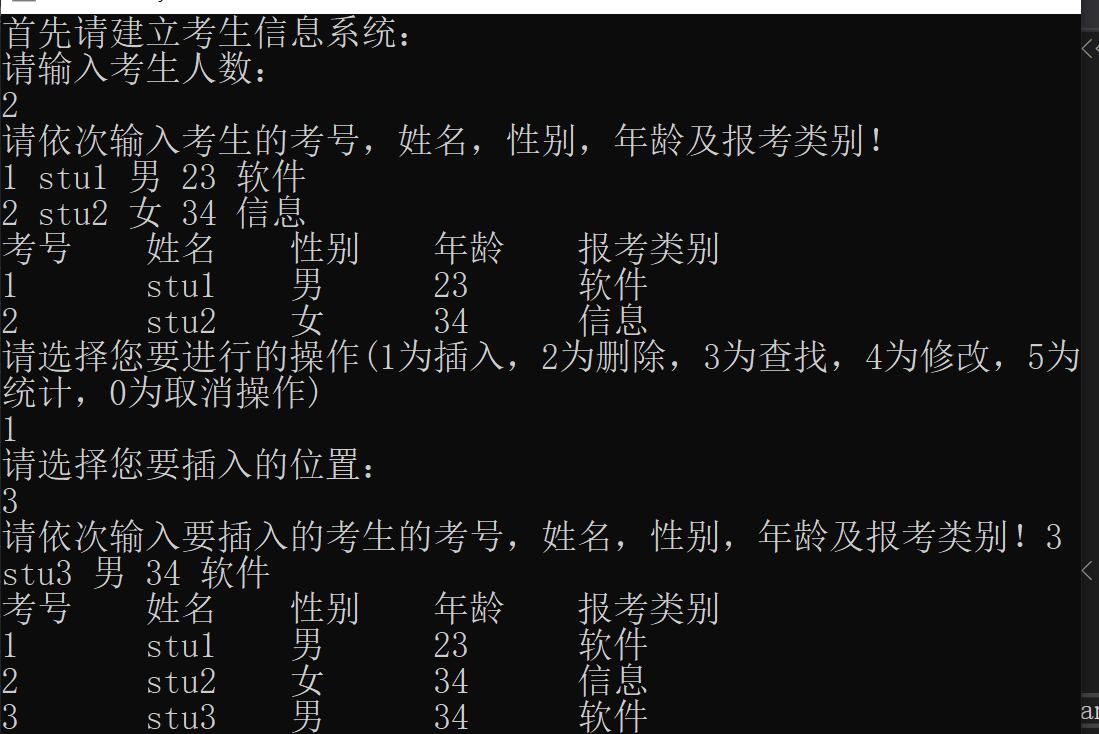
}

}

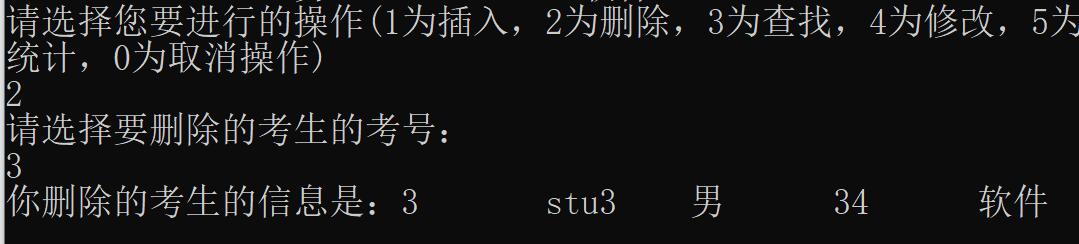
# 测试

## 4.1功能测试及边界测试

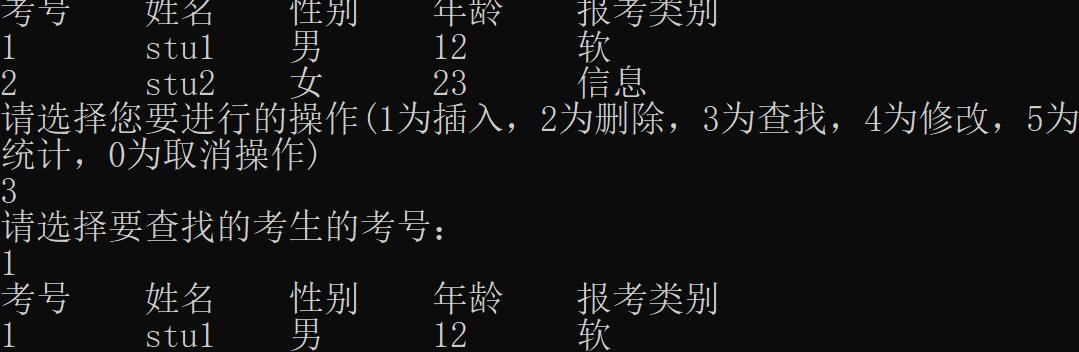
### 4.1.1插入功能：



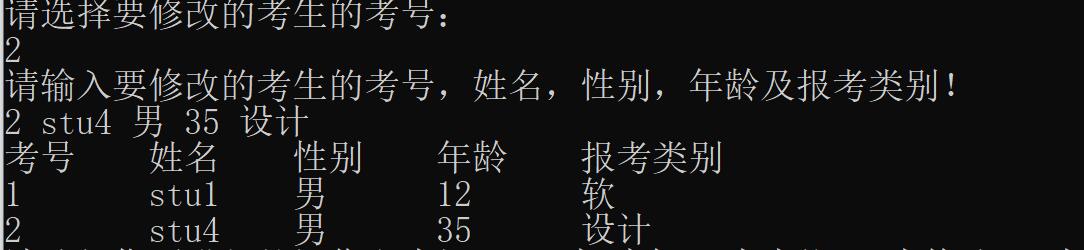
### 4.1.2删除功能：



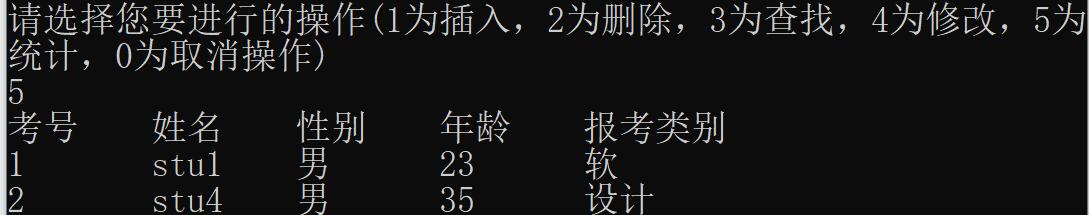
### 4.1.3查找功能：

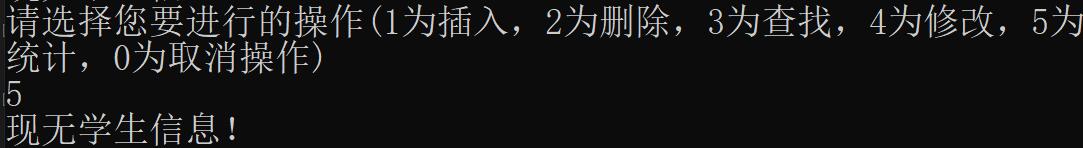


### 4.1.4修改功能：



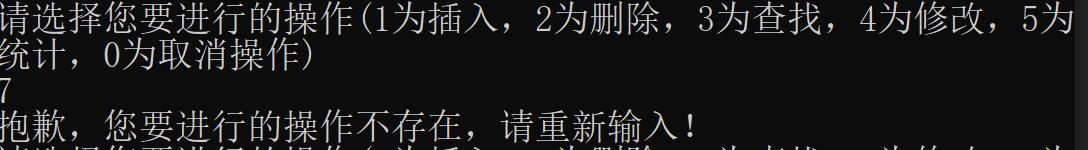
### 4.1.5统计功能：



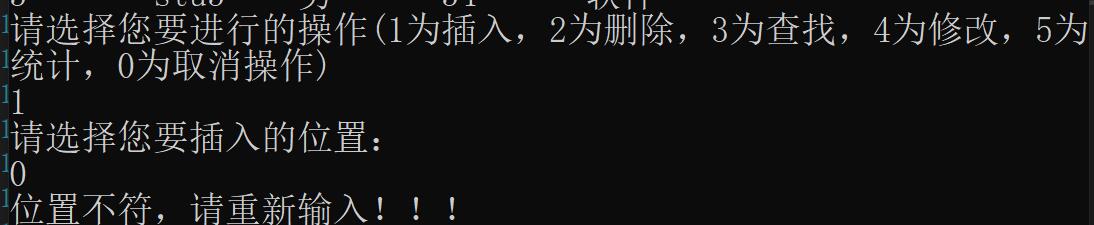


## 4.2错误测试

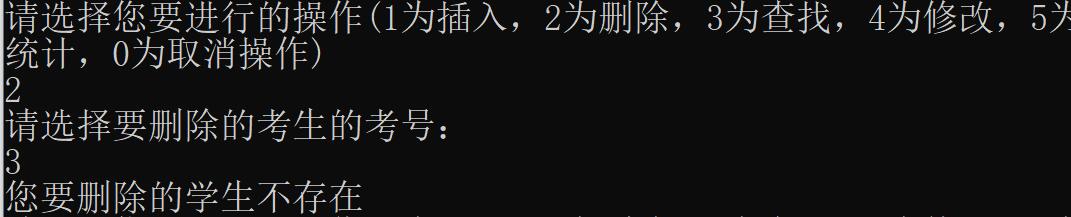
### 4.2.1操作码输入错误：



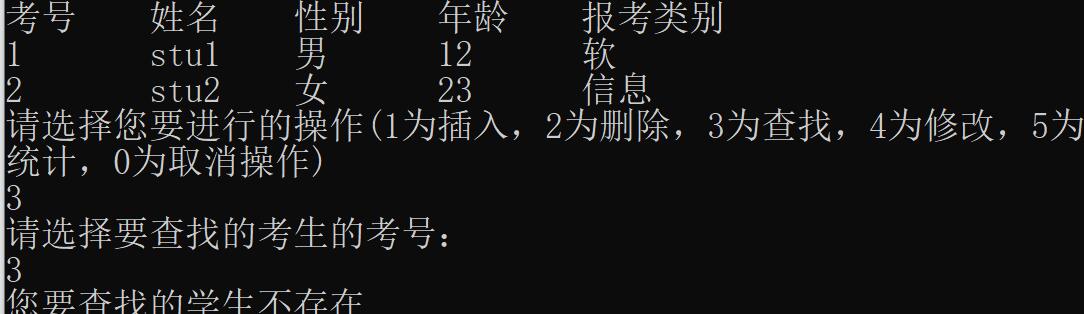
### 4.2.2插入位置错误：



### 4.2.3删除位置错误：



### 4.2.4查找位置错误：



### 4.2.5修改位置错误：

