**2019～2020年春季学期**

**序号：**

# 学术与工程实践(计算机)

# 课设报告

**课设题目：** 学生成绩管理系统

**指导老师：** 束劼

班级 计专起本19 学号 19951060119 姓名 章磊

|  |
| --- |
| **总成绩：**  **评语：**  **指导教师签字： 日期：** |

目录

[1 课题主要内容介绍 1](#_Toc49795501)

[1.1 课题概述 1](#_Toc49795502)

[1.2 运行环境说明 2](#_Toc49795503)

[2 系统设计与实现 3](#_Toc49795504)

[2.1 系统设计 3](#_Toc49795505)

[2.1.1 设计原则与设计思路 3](#_Toc49795506)

[2.1.2 功能设计 3](#_Toc49795507)

[2.1.3 数据结构设计 5](#_Toc49795508)

[2.2 系统实现 6](#_Toc49795509)

[2.2.1 功能 6](#_Toc49795510)

[2.2.2 学生添加功能 12](#_Toc49795511)

[2.2.3 显示所有学生功能 20](#_Toc49795513)

[2.2.4 查询功能 22](#_Toc49795514)

[2.2.3 删除学生功能 29](#_Toc49795515)

[2.3.1 添加课程功能 36](#_Toc49795516)

[2.4.1 添加成绩功能 42](#_Toc49795517)

[2.4.2 查询成绩功能 47](#_Toc49795518)

[2.4.1 修改功能 54](#_Toc49795519)

[2.4.3 删除成绩功能 60](#_Toc49795520)

[3 系统测试和结果分析 64](#_Toc49795521)

[3.1 学生-添加功能 64](#_Toc49795522)

[3.2 学生-删除功能 64](#_Toc49795523)

[3.3 学生-查询功能 65](#_Toc49795524)

[4 课题收获与总结 69](#_Toc49795525)

[4.1 理论知识应用体会 69](#_Toc49795526)

[4.2 课题应用软件环境的体会 69](#_Toc49795527)

[4.3 其它体会 69](#_Toc49795528)

[5 参考文献 70](#_Toc49795529)

# 课题主要内容介绍

本章共分为两节，其中第一节为课题概述，该节包括实训目的、项目需求、项目要求详尽介绍。第二节为运行环境的说明，该节对在项目开发环境和所使用的语言进行具体说明。

## 课题概述

1. 实训目的：

通过本实训，学生应创建一个命令行模式的学生成绩系统。

1、掌握结构体类型数据的定义方法和引用方法。

2、掌握有关C语言程序设计的文件打开，关闭，输入，输出以及定位。

3、培养C语言程序编程规范。

1. 项目需求：

**（1）通过对文件操作可以实现如下基本功能（总分85分）**

1. 用户登录界面和用户信息管理（15分）
2. 数据信息的录入（15分）
3. 学生、课程及成绩全部显示（15分）
4. 成绩修改及删除（15分）
5. 查询功能 （15分）
6. 成绩统计 （5分）
7. 退出系统（5分）

**（2）通过链表知识完成功能1~6中的至少一个功能（15分）**

注：学生成绩系统作为一个用来进行学生成绩管理的应用软件，在对其进行需求分析的过程中，需要充分考虑系统的可用性和有效性，联系实际应用，从而得到合理的软件。

1. 项目要求：

1）画出功能模块图。

2）整个程序中不能只有main函数，必须按照功能模块图进行功能函数编写。功能函数不能少于10个。

3）学生信息、课程信息和成绩信息必须以结构体形式储存。

4）学生信息、课程信息和成绩信息必须存储在各自文件中。对文件进行操作**正确的**实现对所需信息的增加、删除、修改、查询和计算。

## 运行环境说明

操作系统： Windows系列

开发语言： C语言

# 系统设计与实现

本章分为两节。在第一节中为系统设计，该节包括设计原则与设计思路、功能设计、数据结构设计。第二节中系统实现，包括各功能的流程图、功能描述、以及源代码、运行截图。

## 系统设计

### 设计原则与设计思路

设计原则：

* + - 1. 实用性原则，采用成熟可靠的技术，达到实用、经济和有效的目的。
      2. 高可用性/高可靠性原则，确保程序的兼容荣较好，不会因为一些错误值，系统直接无法运行
      3. 安全性原则，确保程序中不会别插入一些与项目无关的数据
      4. 易用性原则，项目具有友好的交互式界面不会，让操作者在不熟悉系统的情况下无从下手

设计思路：

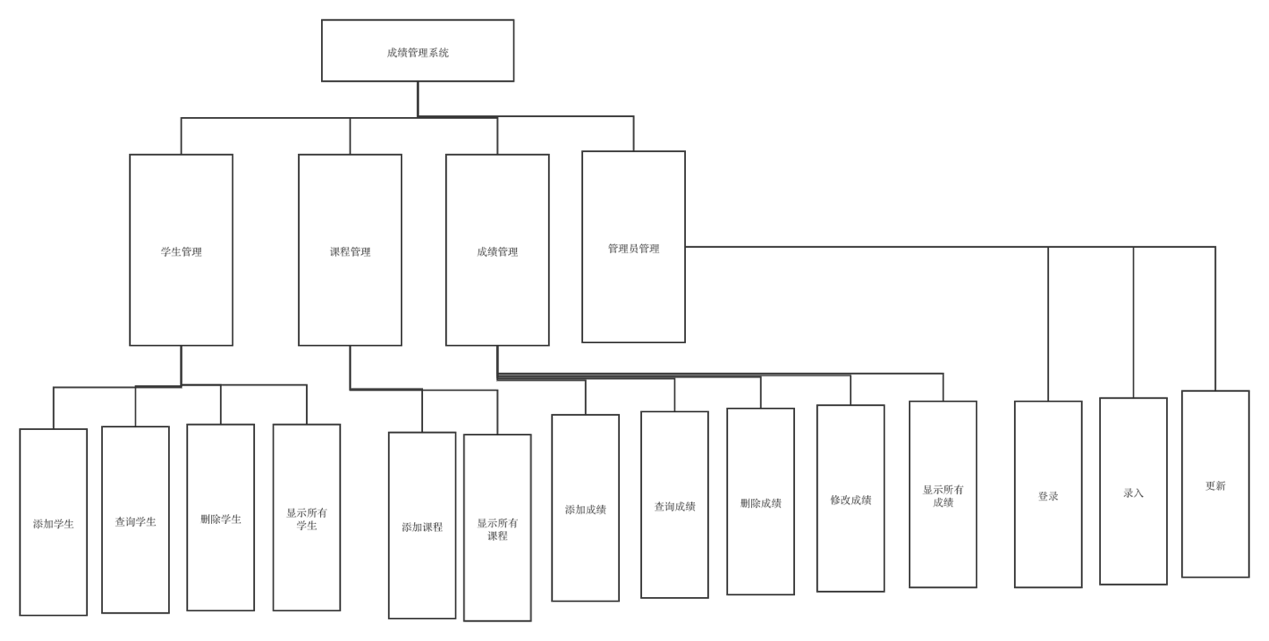
将项目大致分为学生、课程、成绩、管理员模块

使用链表对每个模块的数据进行临时存储，增强每个模块独立性，避免个模块过度耦合

使用大量的 if else 对各种情况进行判断，增强系统的容错行，以及提供错误提示信息，增强用户体验感

### 功能设计

根据项目需求，该程序至少要有7个功能，即用户登录界面和用户信息管理、数据信息的录入、学生、课程及成绩的全部显示、成绩修改及删除、查询、成绩统计、退出系统等功能。系统功能结构图如下所示：



下面是对程序功能设计的描述。

1. 添加学生添

管理员通过输入学号、姓名、性别、年龄、所在系等字段可实现对学生数据的添加。

1. 删除学生

管理员可以通过输入学号，删除某个学生，同时也会删除成绩中与他相关联的成绩。

1. 查询学生

管理员可通过学号、姓名进行学生数据查询。

1. 添加课程

管理员可通过输入正确的课号，对课程进行添加。

1. 添加成绩

管理员通过输入学号、课号、平时成绩、卷面成绩、等字段可实现对课程数据的添加。

1. 删除成绩

管理员通过输入学号能删除对应学生的成绩。

1. 修改学生

管理员输入密码后，再通过输入学号和课号，能有修改某一条成绩记录，当修改了平时成绩和卷面成绩后，系统会根据课程类型自动计算出综合成绩

1. 显示所有学生

显示所有学生信息

1. 显示所有课程功能

显示所有的课程信息。

1. 管理员登录

通过输入用户名和密码，实现系统登录，只有登录后，才能显示功能界面。

### 数据结构设计

数据结构设计如下所示：

**学生信息**

char no[20]; // 学号

char name[16]; // 姓名

char gender[4]; // 性别

int age; // 年龄

char dep[24]; // 系

struct Student \*next; 指向下一个学生

**课程信息**

char no[20]; // 课号

char cName[16]; // 课程名

char tName[20]; // 任课教师

struct Course \*next;

**成绩信息**

char sno[20]; // 学号

char cno[20]; // 课号

int uGrade; // 平时成绩

int pGrade; // 卷面成绩

int tGrage; // 综合成绩

struct Score \*next;

**管理员信息**

char username[20]; // 用户名

int password; //密码

struct Admin \*next;

学生文件： student.txt

课程文件： course.txt

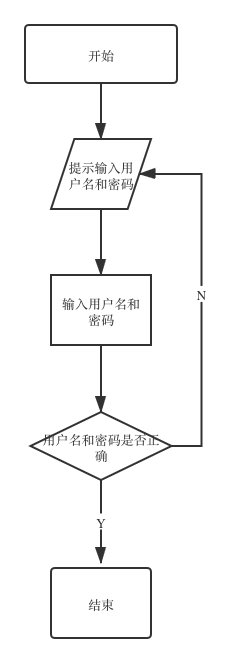
成绩文件： medicine\_sell.dat

## 系统实现

该节报告系统个各功能的流程图、功能描述、源代码界面实现。

### 功能

* + - 1. **用户登录功能流程图**

****

* + - 1. **功能描述**

提示管理员输入账号和密码，管理员输入账号和密码后，与管理链表中的每个管理员进行用户和密码进行比对，如果没有找到该管理员，给出相应的错误提示，如果存在则进入功能选择界面。

* + - 1. **源代码**

**```C**

typedef struct Admin

{

char username[20];

int password;

struct Admin \*next;

} ADMIN\_LINK\_LIST;

// 检测管理员是否存在

ADMIN\_LINK\_LIST \*getAdmin (ADMIN\_LINK\_LIST \*head, char username[20], int password)

{

ADMIN\_LINK\_LIST \*tmp = head->next;

ADMIN\_LINK\_LIST \*result = NULL;

while (tmp != NULL)

{

If (strcmp(tmp->username,username) == 0 &&

tmp.password == password)

{

Result = tmp;

}

tmp = tmp->next;

}

return result;

}

// 校验密码格式

int isValidPassword (int password)

{

int status;

regmatch\_t pmatch[1];

int cflags = REG\_EXTENDED;

const size\_t nmatch = 1;

regex\_t reg;

const char \*pattern =” \\d+”;

regcomp(&reg, pattern, cflags);

status = regexec(&reg, password, nmatch, pmatch, 0);

regfree(&reg);

return status == 0 ? 0 : 1;

}

Int main()

{

Char username[20];

Int password;

Int isAdmin = 0;

ADMIN\_LINK\_LIST \*pAdminList = createAdminLinkList();

…

for (int i = 0; i < 2; i++)

{

switch (i)

{

case 0:

printf("请输入用户名：\n");

scanf("%s", username);

break;

default:

printf("请输入密码：\n");

scanf("%d", &password);

break;

}

}

If (isValidPassword(password) != 0)

{

Printf(“密码格式不正确”);

}

Else

{

IsAdmin = (pAdminList, username, password) == NULL ? 1 : 0;

If (isAdmin == 1)

{

… 显示操作

}

Else

{

Printf(“用户名或密码错误\n”);

}

}

Return 0;

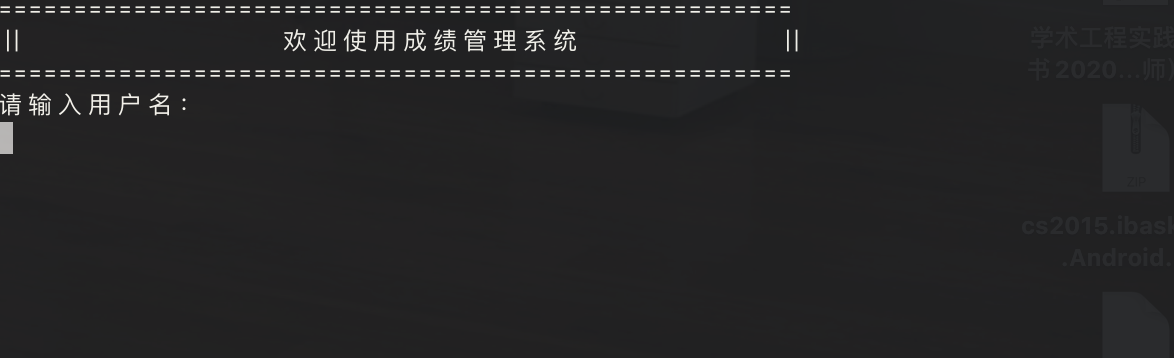
}

**```**

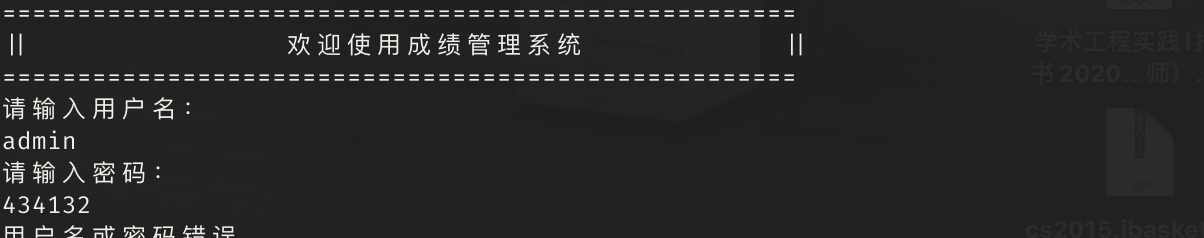
* + - 1. **界面实现**

**录入界面：**

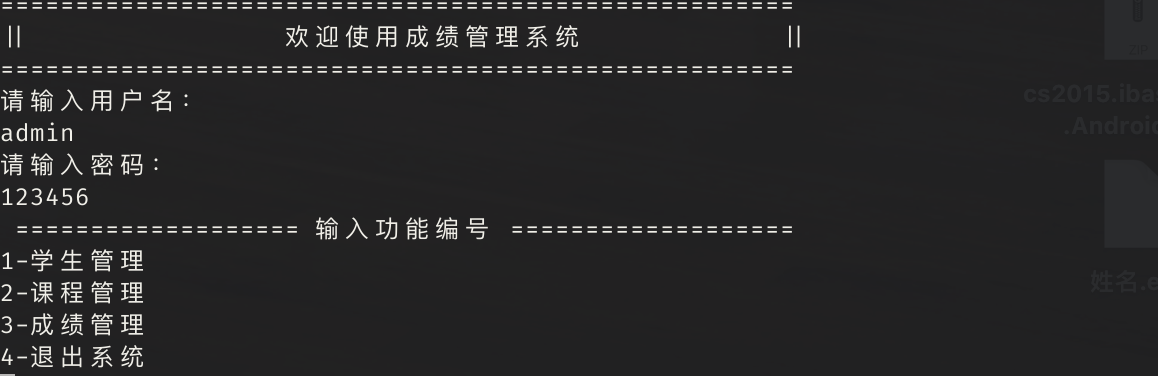
**/用户名和密码正确性检测**

****

**错误提示：**

****

**正确展示：**



* + - 1. **使用说明**

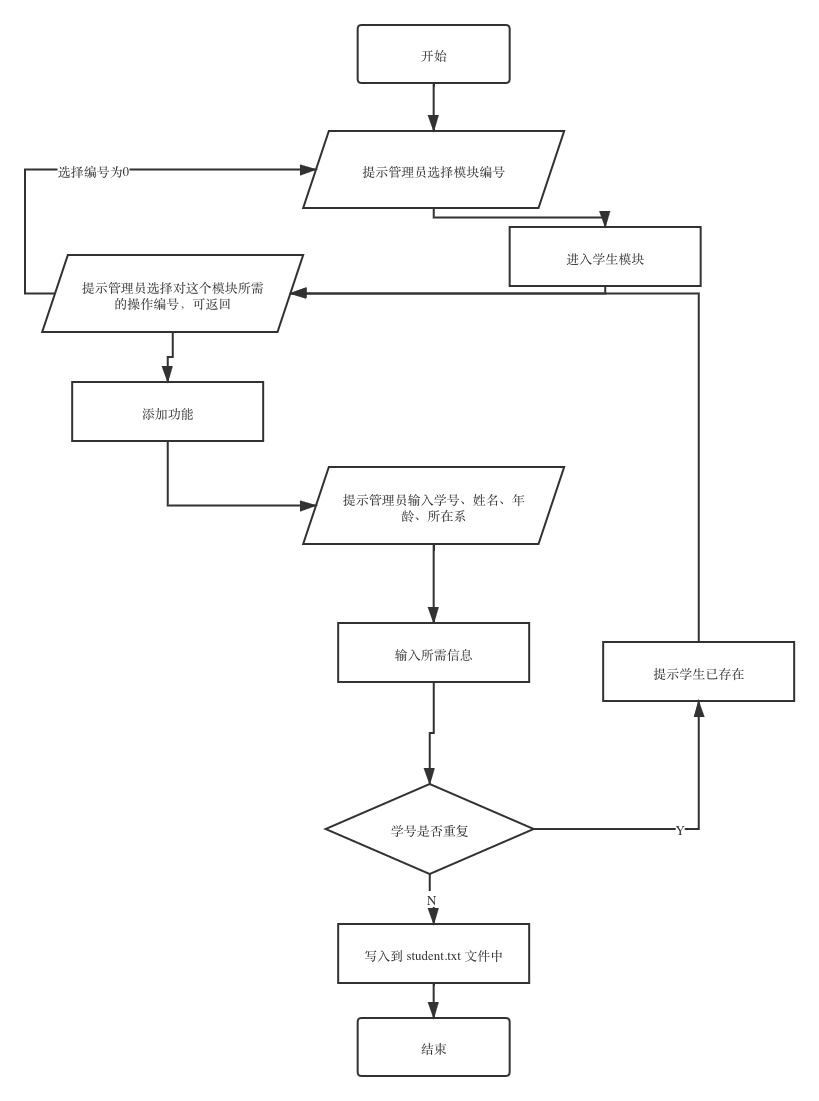
用户需要在给定的提示下，输入所对应的内容，本项目并没有对用户的用户名错特别的要去，但密码必须是数字，否则提示密码格式不正确。

* + - 1. **技术难点**

使用者可能会误输入非数字的密码，此时需要使用 C 的正则表达式方式，使用 /d+ 模式来匹配是否符合密码格式，当检测出密码格式不正确时，直接忽略，输出错误提示，用此方法来减少不必要的链表遍历和数据的正确性比较。

### 学生添加功能

* + - 1. **学生添加流程图**

****

* + - 1. **功能描述**

管理员根据提示选择对管理员进行学生管理进行操作，然后会出现对学生管理的具体操作，其中选择学生添加功能后，提示输入学生学号、姓名、年龄、性别和所在系，当出现重复的学号是会提示学生已存在，并返回学生功能选择界面，如果没有则成功插入到学生链表中，并写入到 student.txt 文件中。

* + - 1. **源代码**

```C

// 添加学生

void insertStu(STU\_LINK\_LIST \*head)

{

// 是否重复

STU\_LINK\_LIST \*tmp = head;

STU\_LINK\_LIST \*node = (STU\_LINK\_LIST \*)malloc(sizeof(STU\_LINK\_LIST));

for (int i = 0, len = 5; i < len; i++)

{

switch (i)

{

case 0:

printf("请输入学号：\n");

scanf("%s", &node->no);

break;

case 1:

printf("请输入姓名：\n");

scanf("%s", &node->name);

break;

case 2:

printf("请输入性别：\n");

scanf("%s", &node->gender);

break;

case 3:

printf("请输入年龄：\n");

scanf("%d", &node->age);

break;

default:

printf("请输入所在系：\n");

scanf("%s", &node->dep);

break;

}

}

if (hasStu(head, node->no) == 1)

{

printf("!!!该学生已经存在\n");

}

else

{

while (tmp->next != NULL)

{

tmp = tmp->next;

}

tmp->next = node;

}

}

int hasStu(STU\_LINK\_LIST \*head, char no[20])

{

STU\_LINK\_LIST \*tmp = head;

int result = 0;

printf("1----%s\n", no);

while (tmp != NULL)

{

if (strcmp(tmp->no, no) == 0)

{

result = 1;

break;

}

tmp = tmp->next;

}

return result;

}

void studentManage(STU\_LINK\_LIST \*head, SCORE\_LINK\_LIST \*head2)

{

int op\_choice;

STU\_LINK\_LIST \*result = NULL;

STU\_LINK\_LIST \*delStu = NULL;

while ((op\_choice = operateSelectStu(" =================== 学生管理 =================== \n")) != 0)

{

switch (op\_choice)

{

case 1:

printf(" =================== 学生管理 (添加学生) =================== \n");

insertStu(head);

writeFileStudent(head);

break;

case 2:

printf(" =================== 学生管理 (显示所有学生) =================== \n");

dispalyStu(head);

break;

case 3:

printf(" =================== 学生管理 (查询学生) =================== \n");

result = searchStu(head);

if (result->next == NULL)

{

printf("没有找到该学生\n");

}

else

{

dispalyStu(result);

}

break;

case 4:

printf(" =================== 学生管理 (删除学生) =================== \n");

delStu = deleteStu(head);

if (delStu != NULL)

{

deleteScore(head2, delStu->no);

}

writeFileStudent(head);

writeFileScore(head2);

break;

default:

printf("无效的选择");

break;

}

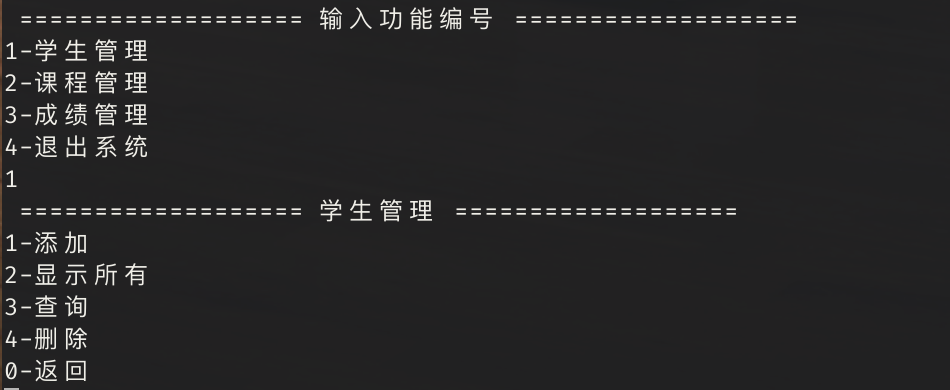
}

}

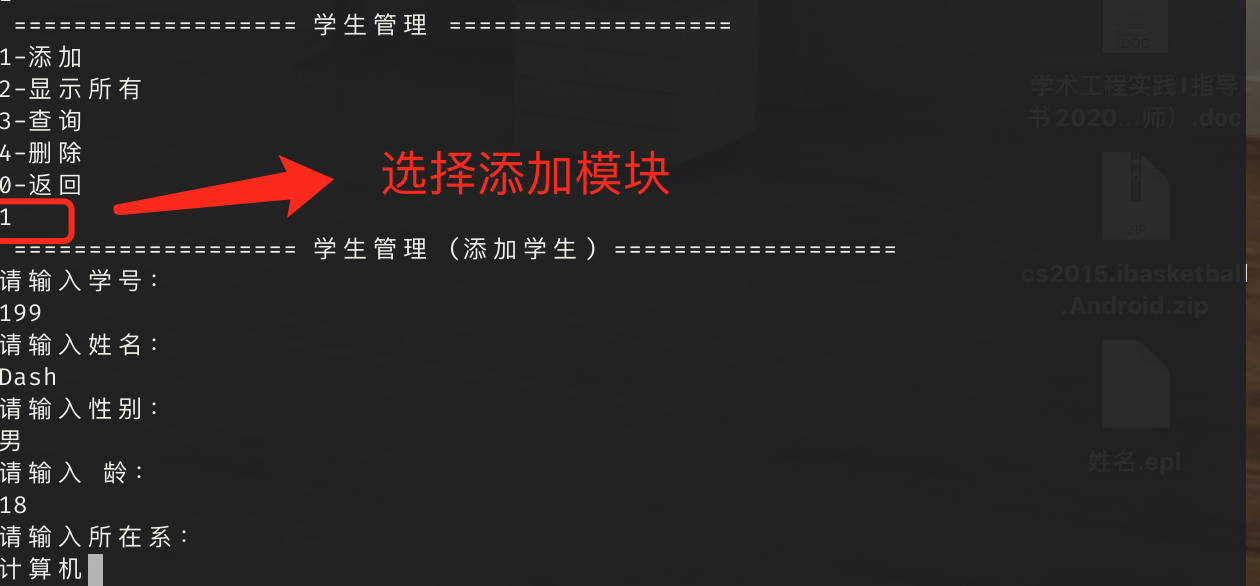
```

* + - 1. **界面实现**

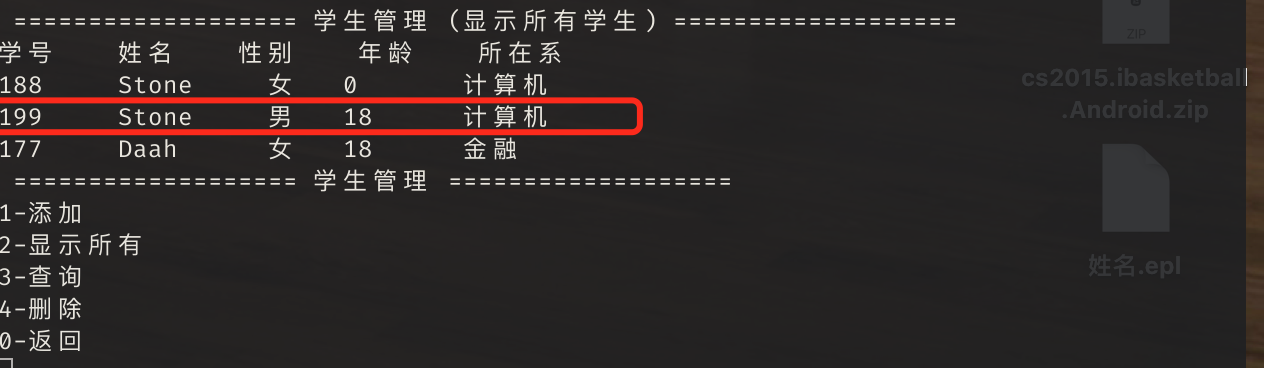
**学生模块选择**

****

**输入学生信息**

****

**添加信息**

****

* + - 1. **使用说明**

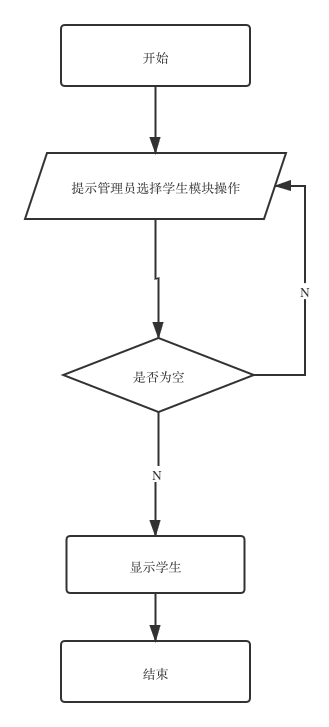
管理员选择添加学生功能编号，进入学生信息输入界面，根据提示会先后输入学号、姓名、性别、年龄、所在系，只有当学生数据库中没有该学号的学生时，才能正确写入 student.txt 文件，否则提示错误学生已存在，并且进入之前的该模块的功能选择界面，重新操作。

* + - 1. **技术难点**

### 需要找到链表的最后一个节点，由于一开始对链表结构不是很熟悉，导致自己无法入手，由于链表指针是一个引用类型，所以不能轻易求该节点属性，否则会影响其他指向该链表节点的变量。

### 2.2.3 显示所有学生功能

* + - 1. **显示所有学生流程图**

****

* + - 1. **功能描述**

管理员选择显示所有学生的功能编号，按规范打印当前学生链表里的所有节点数据。

* + - 1. **源代码**

**```C**

// 显示学生

void dispalyStu(STU\_LINK\_LIST \*head)

{

STU\_LINK\_LIST \*tmp = head->next;

if (tmp == NULL)

{

printf("1111111111\n");

}

printf("%-10s%-10s%-10s%-10s%-0s\n", "学号", "姓名", "性别", "年龄", "所在系");

while (tmp != NULL)

{

printf("%-8s%-10s%-6s%-8d%-0s\n", tmp->no, tmp->name, tmp->gender, tmp->age, tmp->dep);

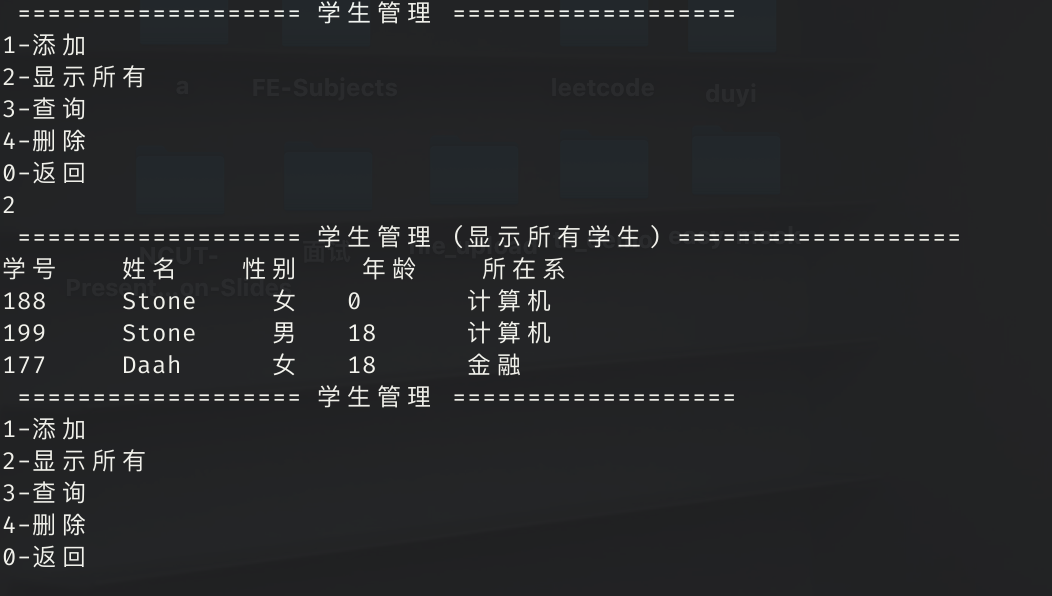
tmp = tmp->next;

}

}

**```**

* + - 1. **界面实现**

****

* + - 1. **使用说明**

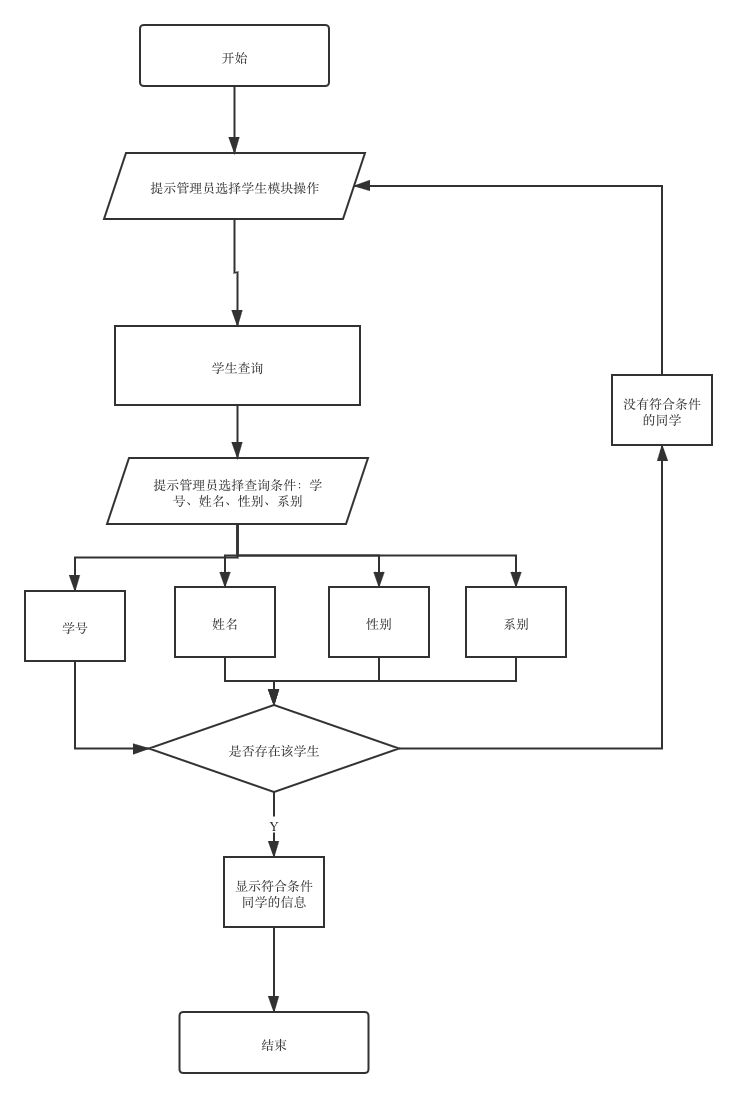
在学生管理里面，选择 2 号功能即可，进行当前所有学生的信息打印。

* + - 1. **技术难点**

数据的格式化打印需要估算一下大致距离，比较费时。

### 2.2.4 查询功能

* + - 1. **查询学生流程图**

****

* + - 1. **功能描述**

管理员根据提示选择用于查询的条件，当管理员选择通过学号进行查询时，结果是唯一的；当选择姓名、性别、系别作为条件进行查询时，所查询出来的结果可能是多条，当没有查询到是显示没有符合条件的同学。

* + - 1. **源代码**

```C

// 用于对链表节点进行 clone，避免使原数据受到影响

STU\_LINK\_LIST \*cloneStu(STU\_LINK\_LIST \*stu)

{

STU\_LINK\_LIST \*node = (STU\_LINK\_LIST \*)malloc(sizeof(STU\_LINK\_LIST));

strcpy(node->no, stu->no);

strcpy(node->name, stu->name);

strcpy(node->gender, stu->gender);

strcpy(node->dep, stu->dep);

node->age = stu->age;

node->next = NULL;

return node;

}

// 查找学生，返回查找结果

STU\_LINK\_LIST \*searchStu(STU\_LINK\_LIST \*head)

{

int menu\_choice = menu\_choice = searchMenuStu();

STU\_LINK\_LIST \*tmp = head->next;

STU\_LINK\_LIST \*result = (STU\_LINK\_LIST \*)malloc(sizeof(STU\_LINK\_LIST));

STU\_LINK\_LIST \*rHead = result;

STU\_LINK\_LIST \*newNode = NULL;

char value[40]; // 搜索内容

int flag = 1;

int isSearchName = 0;

while (tmp != NULL)

{

newNode = cloneStu(tmp);

switch (menu\_choice)

{

case 1:

if (flag == 1)

{

printf("请输入学号：");

scanf("%s", &value);

}

printf("-----%d %s %s\n", strcmp(tmp->no, value), tmp->no, value);

if (strcmp(tmp->no, value) == 0)

{

rHead->next = newNode;

isSearchName = 1;

}

break;

case 2:

if (flag == 1)

{

printf("请输入姓名：");

scanf("%s", &value);

}

if (strcmp(tmp->name, value) == 0)

{

rHead->next = newNode;

rHead = rHead->next;

}

break;

case 3:

if (flag == 1)

{

printf("请输入性别：");

scanf("%s", &value);

}

if (strcmp(tmp->gender, value) == 0)

{

rHead->next = newNode;

rHead = rHead->next;

}

break;

case 4:

if (flag == 1)

{

printf("请输入系别：");

scanf("%s", &value);

}

if (strcmp(tmp->dep, value) == 0)

{

rHead->next = newNode;

rHead = rHead->next;

}

break;

default:

printf("无效的选择");

break;

}

flag = 0;

tmp = tmp->next;

if (isSearchName == 1)

{

break;

}

}

return result;

}

void studentManage(STU\_LINK\_LIST \*head, SCORE\_LINK\_LIST \*head2)

{

int op\_choice;

STU\_LINK\_LIST \*result = NULL;

STU\_LINK\_LIST \*delStu = NULL;

while ((op\_choice = operateSelectStu(" =================== 学生管理 =================== \n")) != 0)

{

switch (op\_choice)

{

case 1:

printf(" =================== 学生管理 (添加学生) =================== \n");

insertStu(head);

writeFileStudent(head);

break;

case 2:

printf(" =================== 学生管理 (显示所有学生) =================== \n");

dispalyStu(head);

break;

case 3:

printf(" =================== 学生管理 (查询学生) =================== \n");

result = searchStu(head);

if (result->next == NULL)

{

printf("没有找到该学生\n");

}

else

{

dispalyStu(result);

}

break;

case 4:

printf(" =================== 学生管理 (删除学生) =================== \n");

delStu = deleteStu(head);

if (delStu != NULL)

{

deleteScore(head2, delStu->no);

}

writeFileStudent(head);

writeFileScore(head2);

break;

default:

printf("无效的选择");

break;

}

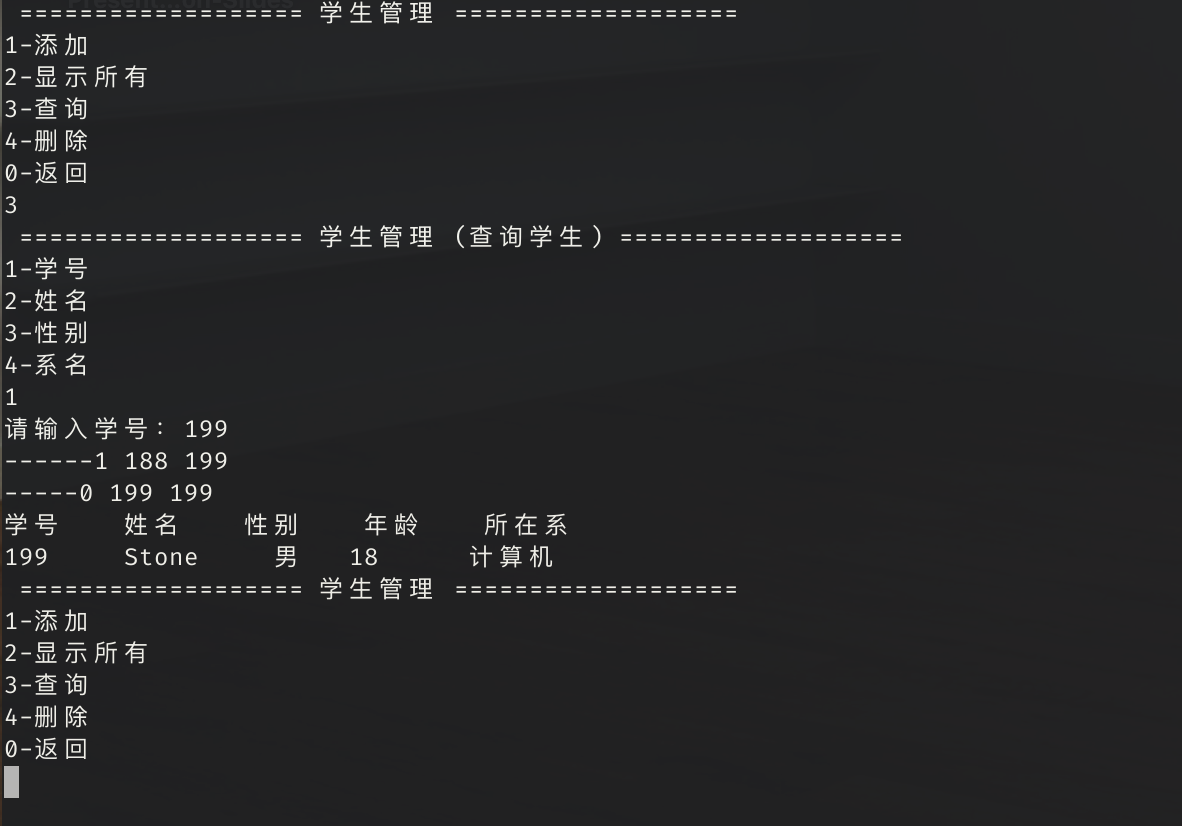
}

}

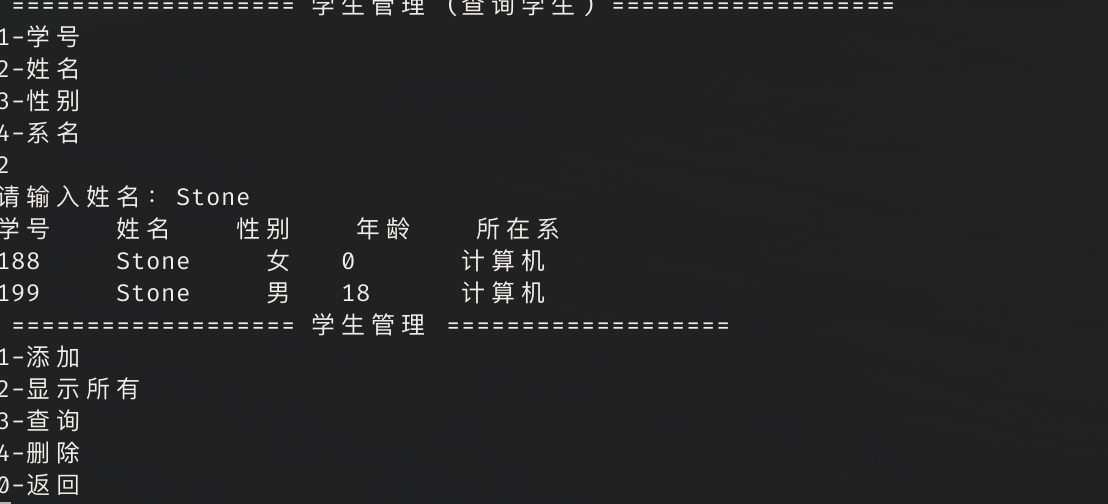
```

* + - 1. **界面实现**

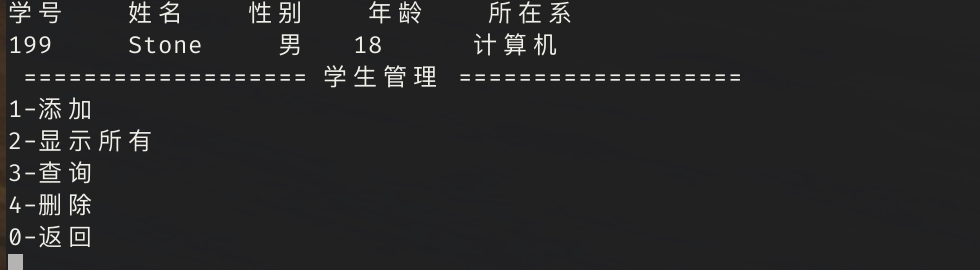
**学号**

****

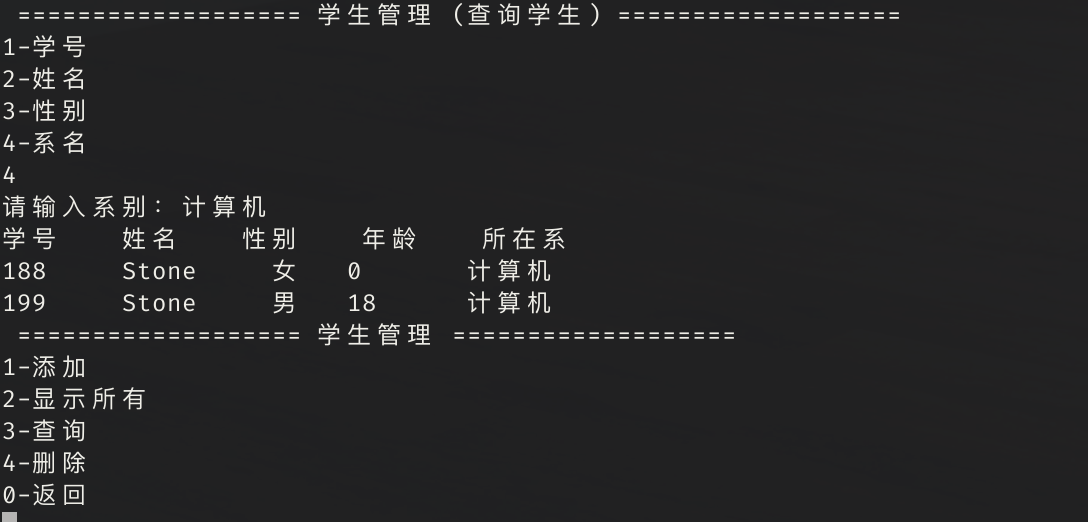
**姓名**

****

**性别**

****

**所在系**

****

* + - 1. **使用说明**

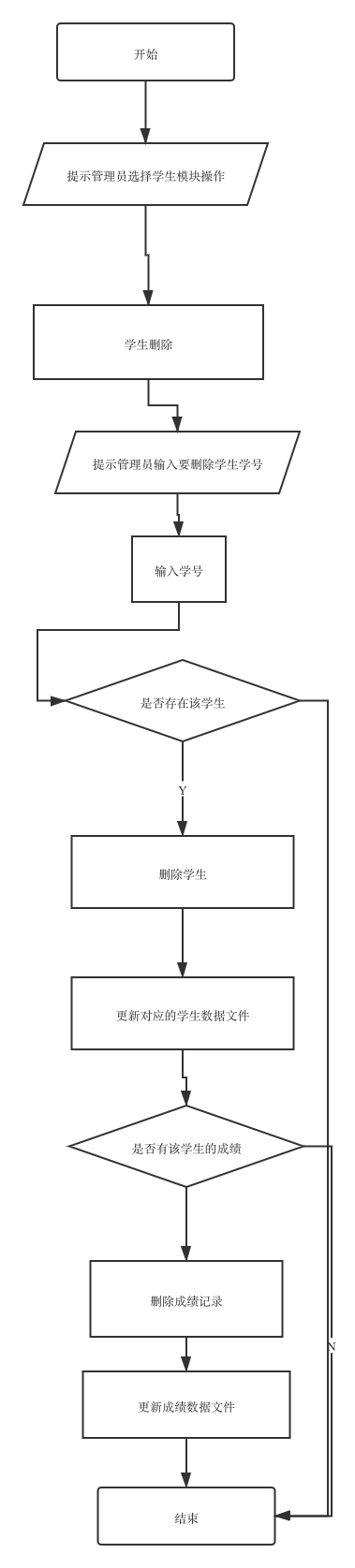
首先管理员要进入学生模块，然后选择 3-查询功能，然后会进入查询条件，然后在对应条件的编号，根据提示输入要查询学生的具体条件值即可。

* + - 1. **技术难点**

通过学号查询是比其他几个条件特殊的，由于学号是不可能重复的，所以如果查询到了，那么可以直接终止循环，而不需要一直循环下去，但是其他几个条件是可以重复出现的，所有在对学生链表进行循环的同时，还需要对用于收集结果的数据结构也进行操作，保证每次都是最后一个。

### 删除学生功能

* + - 1. **删除流程图**



* + - 1. **功能描述**

根据管理员输入的学号，判断学生是否存在，如果不存在给出不存在提示，如果存在在删除学生后，会自动删除该学生的有关的成绩。

* + - 1. **源代码**

```C

// 删除学生

STU\_LINK\_LIST \*deleteStu(STU\_LINK\_LIST \*head)

{

STU\_LINK\_LIST \*tmp = head;

STU\_LINK\_LIST \*prev = NULL;

STU\_LINK\_LIST \*result = NULL;

char no[20];

printf("请输入要删除学生的学号：\n");

scanf("%s", no);

while (tmp != NULL)

{

if (strcmp(tmp->no, no) == 0)

{

result = tmp;

if (prev == NULL)

{

head = head->next;

}

else

{

prev->next = tmp->next;

tmp = tmp->next;

}

break;

}

else

{

prev = tmp;

tmp = tmp->next;

}

}

return result;

}

void studentManage(STU\_LINK\_LIST \*head, SCORE\_LINK\_LIST \*head2)

{

int op\_choice;

STU\_LINK\_LIST \*result = NULL;

STU\_LINK\_LIST \*delStu = NULL;

while ((op\_choice = operateSelectStu(" =================== 学生管理 =================== \n")) != 0)

{

switch (op\_choice)

{

case 1:

printf(" =================== 学生管理 (添加学生) =================== \n");

insertStu(head);

writeFileStudent(head);

break;

case 2:

printf(" =================== 学生管理 (显示所有学生) =================== \n");

dispalyStu(head);

break;

case 3:

printf(" =================== 学生管理 (查询学生) =================== \n");

result = searchStu(head);

if (result->next == NULL)

{

printf("没有找到该学生\n");

}

else

{

dispalyStu(result);

}

break;

case 4:

printf(" =================== 学生管理 (删除学生) =================== \n");

delStu = deleteStu(head);

// 删除学生的成绩记录

if (delStu != NULL)

{

deleteScore(head2, delStu->no);

}

writeFileStudent(head);

writeFileScore(head2);

break;

default:

printf("无效的选择");

break;

}

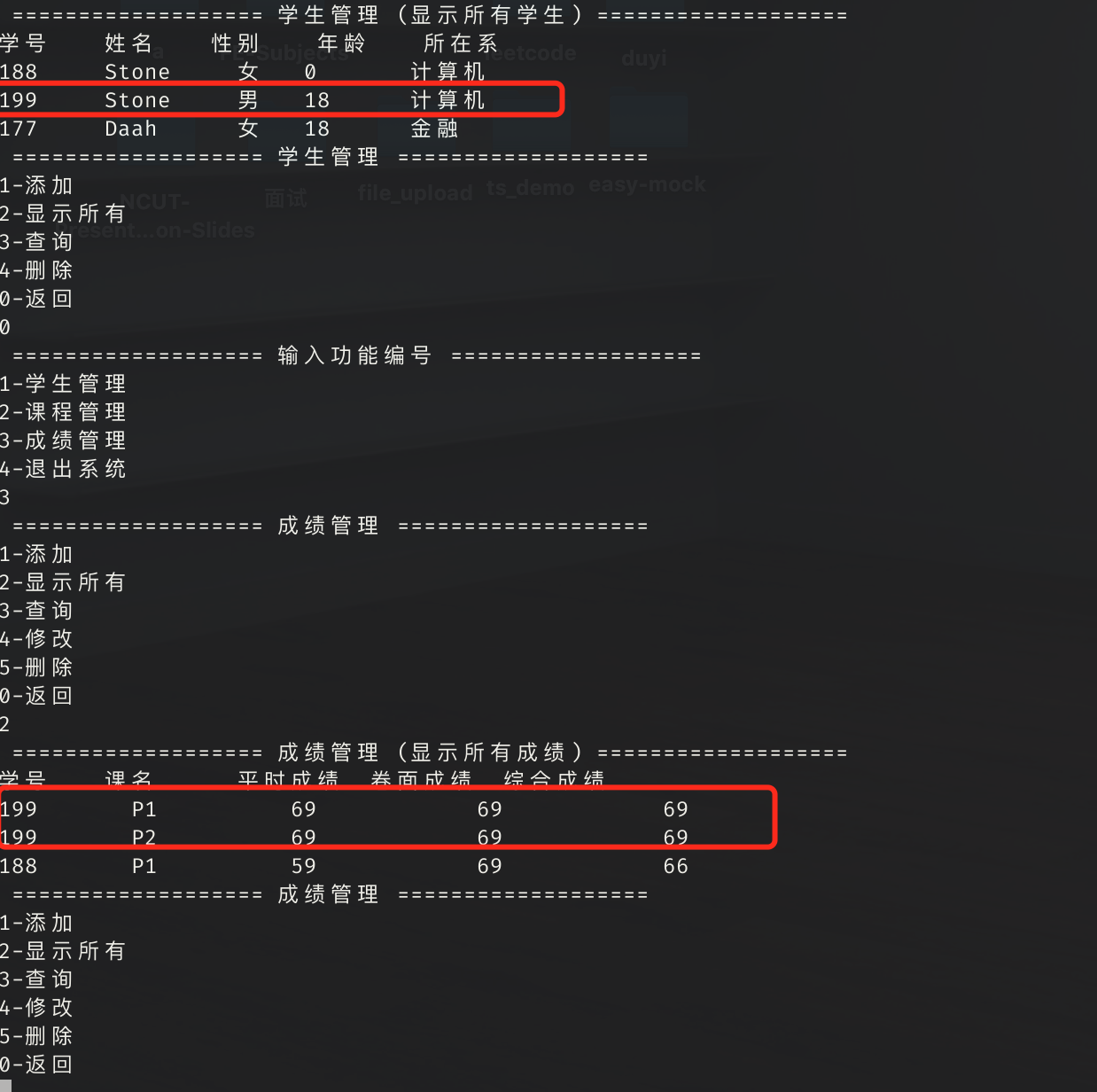
}

}

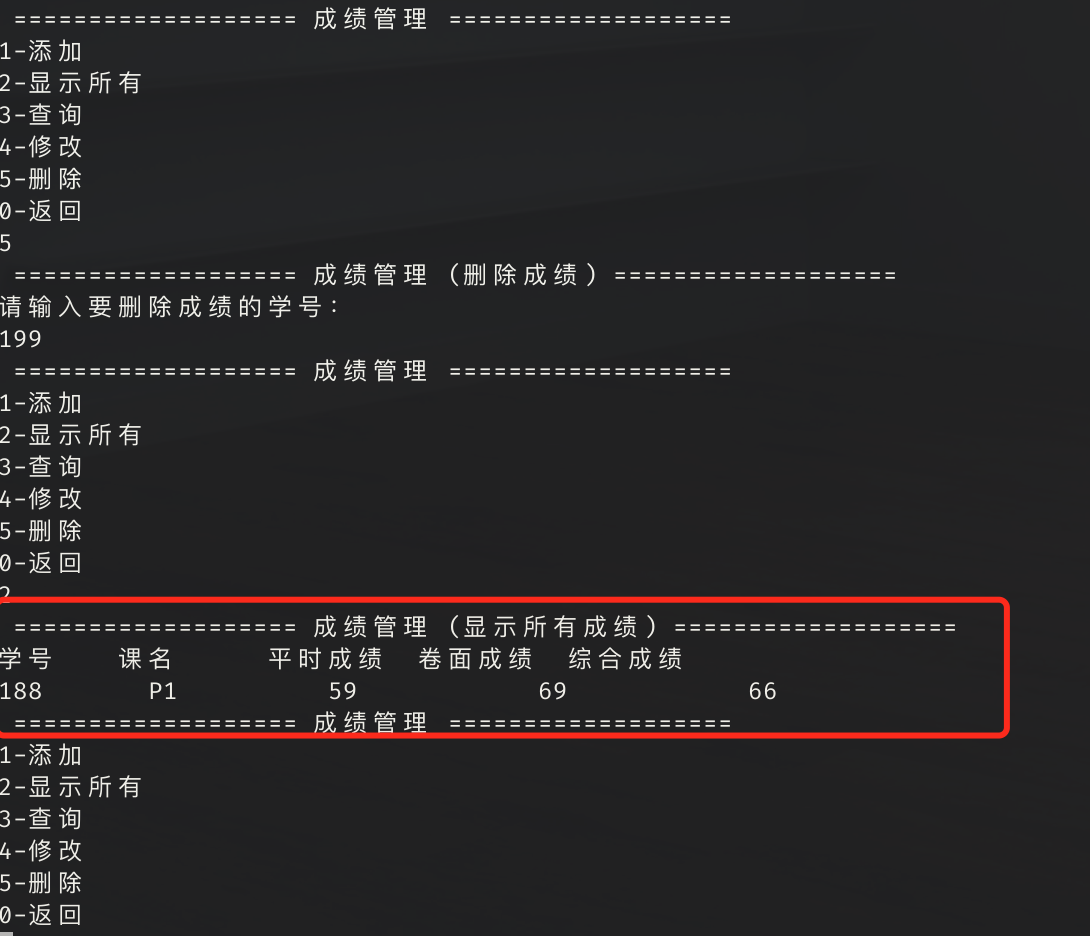
```

* + - 1. **界面实现**

**删除 199**

****

**关联的成绩删除**

****

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* + - 1. **使用说明**

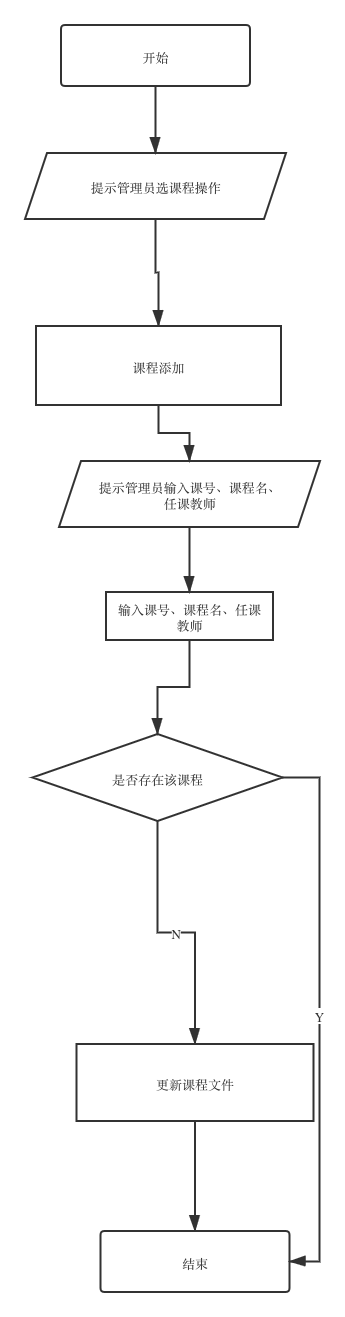
管理员进入学生模块，选择 4-删除学生功能，提示管理员输入要删除的学生学号，然后自动查询是否存在该学生，如果不存在，提示学生不存在；反之在删除学生后，自动删除该学生的成绩。

* + - 1. **技术难点**

链表的删除是通过将当前节点的 prev 节点的.next 指针指向 current 节点.next，所以需要一个 prev 记录上一个节点，一开没有了解链表，导致实现比较困难。

### 2.3.1 添加课程功能

* + - 1. **添加流程图**

****

* + - 1. **功能描述**

根据管理员输入的课程号，先进行课程是否存在判断，如果存在，给出课程已存在提示；反之，将课程添加到课程链表尾，并且更新 course.txt 文件。

* + - 1. **源代码**

```C

// 判断是否存课程

int hasCourse(COURSE\_LINK\_LIST \*head, char value[20], int tag)

{

COURSE\_LINK\_LIST \*tmp = head;

int result = 0;

while (tmp != NULL)

{

if (tag == 0 && strcmp(tmp->no, value) == 0)

{

result = 1;

break;

}

if (tag == 1 && strcmp(tmp->cName, value) == 0)

{

result = 1;

break;

}

tmp = tmp->next;

}

return result;

}

void insertCourse(COURSE\_LINK\_LIST \*head)

{

COURSE\_LINK\_LIST \*tmp = head;

COURSE\_LINK\_LIST \*node = (COURSE\_LINK\_LIST \*)malloc(sizeof(COURSE\_LINK\_LIST));

node->next = NULL;

for (int i = 0; i < 3; i++)

{

switch (i)

{

case 0:

printf("请输入课号：\n");

scanf("%s", &node->no);

break;

case 1:

printf("请输入课名：\n");

scanf("%s", &node->cName);

break;

default:

printf("请输入任课教师：\n");

scanf("%s", &node->tName);

break;

}

}

if (hasCourse(head, node->no) == 1)

{

printf("!!!该课程已经存在\n");

}

else

{

while (tmp->next != NULL)

{

tmp = tmp->next;

}

tmp->next = node;

}

}

void courseManage(COURSE\_LINK\_LIST \*head)

{

int op\_choice;

while ((op\_choice = operateSelectCourse(" =================== 课程管理 =================== \n")) != 0)

{

switch (op\_choice)

{

case 1:

printf(" =================== 课程管理 (添加课程) =================== \n");

insertCourse(head);

writeFileCourse(head);

break;

case 2:

printf(" =================== 课程管理 (显示所有课程) =================== \n");

displayCourse(head);

break;

default:

printf("无效的选择");

break;

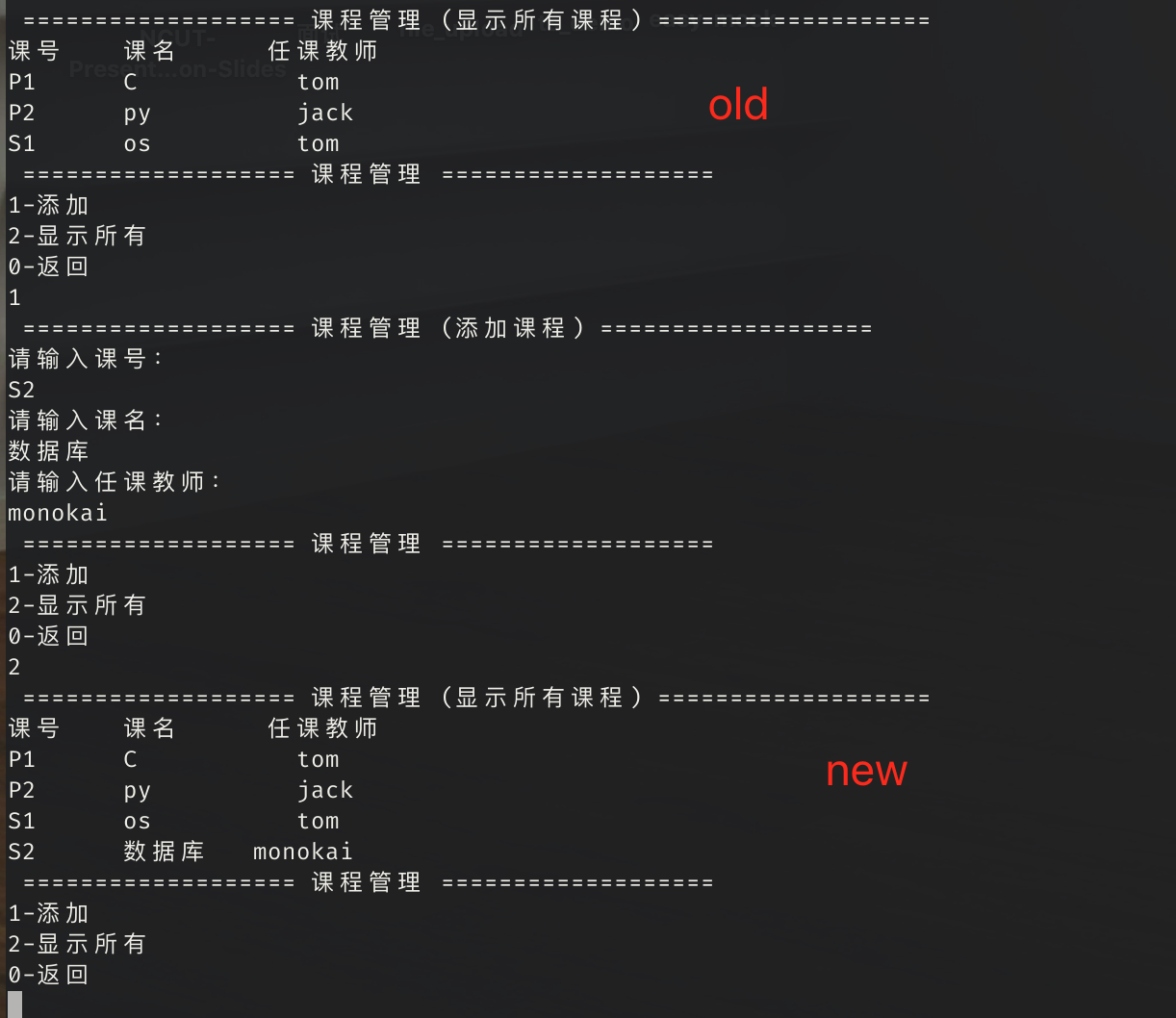
}

}

}

```

* + - 1. **界面实现**

****

* + - 1. **使用说明**

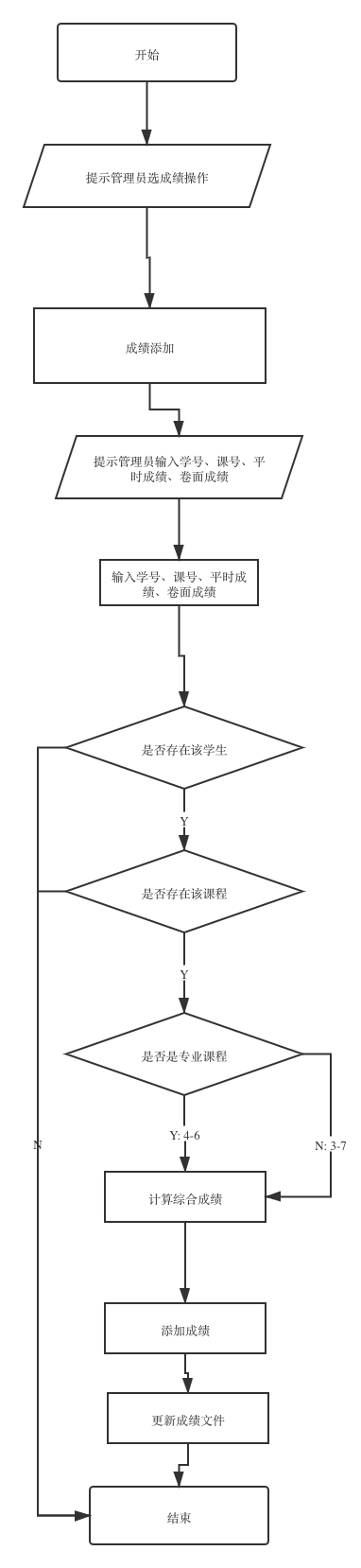
管理员先选择2-成绩管理，进入成绩管理，然后选择 1- 添加课程，根据提示输入课号、课名、任课教师数据，如果存在会进行提示课程已存在，如果没有成功新增。

* + - 1. **技术难点**

需要找到链表的最后一个节点，对链表结构仍然较为陌生，导致指针指向经常出错，数据一会添加成功，一会又添加失败了。

### 2.4.1 添加成绩功能

添加成绩流程图

****

* + - 1. **功能描述**

管理员选择成绩添加功能，根据管理员输入的学号和课号，分别进行学生和课号存在进行的判断，其一不存在，给出添加失败提示；如果都存在，在对课程进行类型判断，如果是专业课，使用4-6 计算方式，计算综合成绩，反之 使用3-7 方式计算综合成绩。

* + - 1. **源代码**

**`**``C

// 校验课号格式

int isValidCourse(char \*match\_text, char \*match\_pattern)

{

int status;

regmatch\_t pmatch[1];

int cflags = REG\_EXTENDED;

const size\_t nmatch = 1;

regex\_t reg;

const char \*pattern = match\_pattern;

regcomp(&reg, pattern, cflags);

status = regexec(&reg, match\_text, nmatch, pmatch, 0);

regfree(&reg);

return status == 0 ? 0 : 1;

}

void insertScore(SCORE\_LINK\_LIST \*head, STU\_LINK\_LIST \*head2, COURSE\_LINK\_LIST \*head3)

{

SCORE\_LINK\_LIST \*tmp = head;

SCORE\_LINK\_LIST \*node = (SCORE\_LINK\_LIST \*)malloc(sizeof(SCORE\_LINK\_LIST));

node->next = NULL;

for (int i = 0; i < 4; i++)

{

switch (i)

{

case 0:

printf("请输入学号：\n");

scanf("%s", &node->sno);

break;

case 1:

printf("请输入课号：\n");

scanf("%s", &node->cno);

break;

case 2:

printf("请输入平时成绩：\n");

scanf("%d", &node->uGrade);

break;

case 3:

printf("请输入卷面成绩：\n");

scanf("%d", &node->pGrade);

break;

default:

printf("请输入综合成绩：\n");

scanf("%d", &node->tGrage);

break;

}

}

if (hasStu(head2, node->sno) == 1 && hasCourse(head3, node->cno, 0) == 1)

{

char \*cno = node->cno;

char \*cParrtern = "P\\d\*";

char \*sParrtern = "S\\d\*";

int isCCourse = isValidCourse(cno, cParrtern);

int isSCourse = isValidCourse(cno, sParrtern);

if (isCCourse == 0 || isSCourse == 0)

{

if (isSCourse == 0)

{

node->tGrage = node->uGrade \* 0.4 + node->pGrade \* 0.6;

}

if (isCCourse == 0)

if (hasStu(head2, node->sno) == 1 && hasCourse(head3, node->cno, 0) == 1)

{

char \*cno = node->cno;

char \*cParrtern = "P\\d\*";

char \*sParrtern = "S\\d\*";

int isCCourse = isValidCourse(cno, cParrtern);

int isSCourse = isValidCourse(cno, sParrtern);

if (isCCourse == 0 || isSCourse == 0)

{

if (isSCourse == 0)

{

node->tGrage = node->uGrade \* 0.4 + node->pGrade \* 0.6;

}

if (isCCourse == 0)

{

node->tGrage = node->uGrade \* 0.3 + node->pGrade \* 0.7;

}

while (tmp->next != NULL)

{

tmp = tmp->next;

}

tmp->next = node;

}

else

{

printf("非法课号\n");

}

}

else

{

printf("学号或课号无效\n");

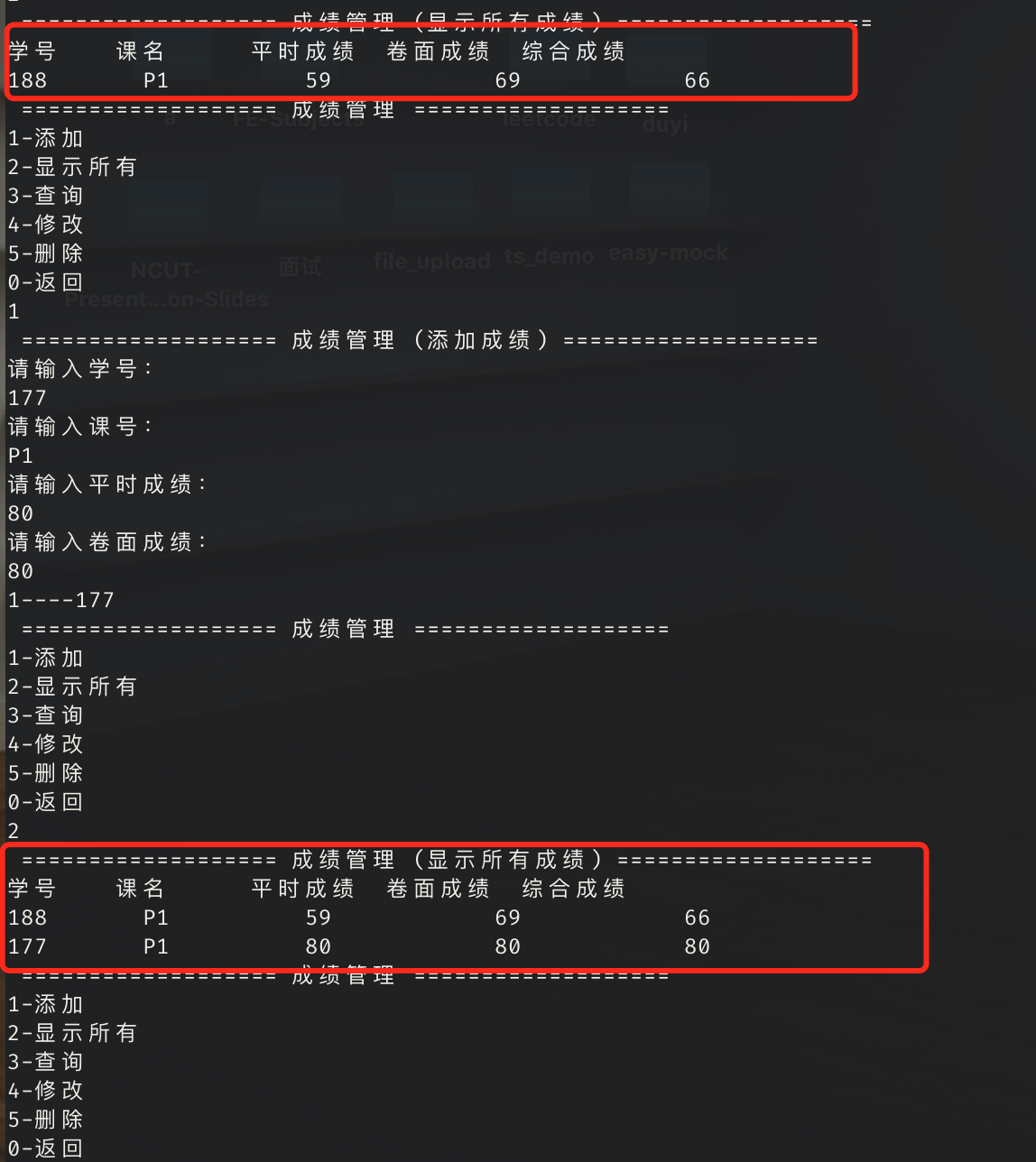
}

}

```

* + - 1. **界面实现**

// 添加成功



* + - 1. **使用说明**

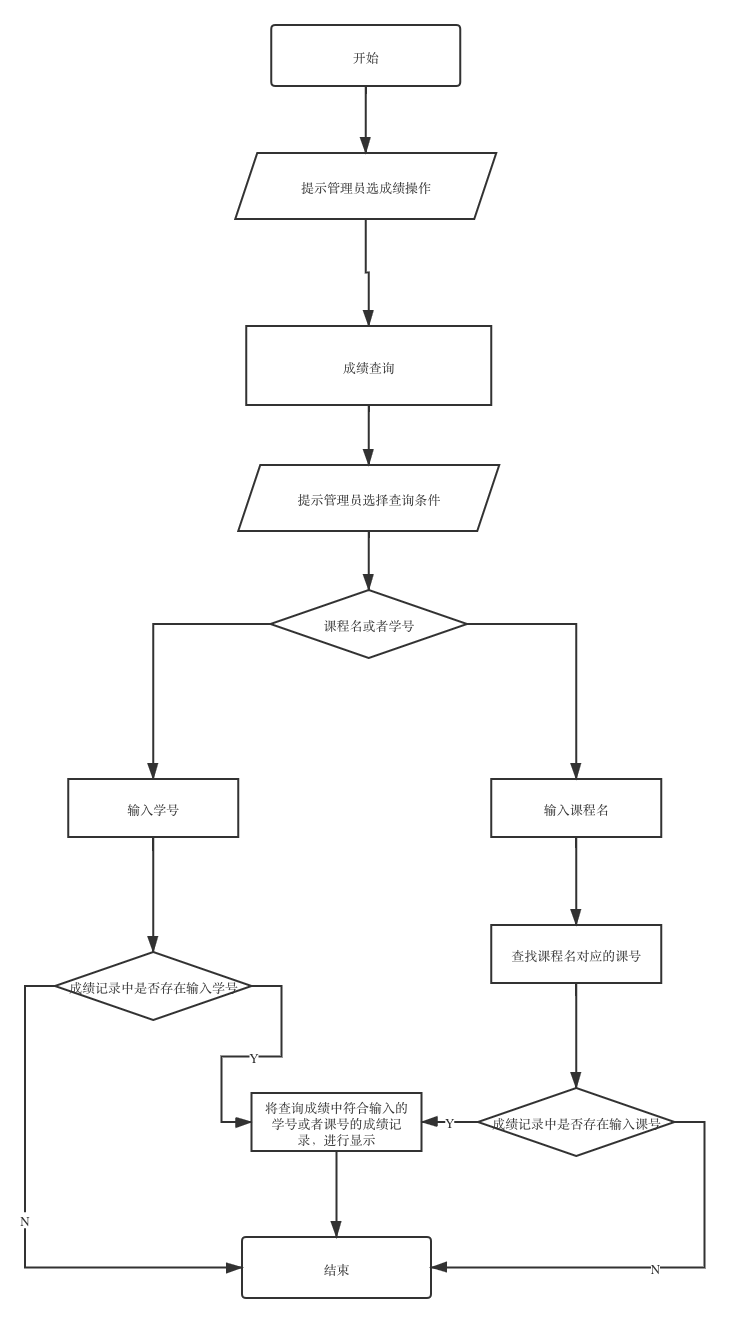
管理员选择添加成绩，根据提示数据课学号、课号、平时成绩、卷面成绩；课程必须是使用 P 或 S 开头的课号，不然会提示课程格式错误。

* + - 1. **技术难点**

项目中使用正则表达式来校验课号的格式，但由于正则表达式不熟悉，使用了对课号逐个字符比较咱时解决，后来查阅了一些相关案例，找到 C 语言中正则的使用，替换了之前的方案。

### 2.4.2 查询成绩功能

* + - 1. **查询成绩流程图**

****

* + - 1. **功能描述**

管理员选择学号或课程名作为查询条件，按提示输入对应的学号或者课程名，如学号查询，则直接成绩文件中找到符合条件的成绩记录，如果以课程名为查询条件则先查询到课程名对应的课号，然后在从成绩文件中找符合条件的成绩记录。

* + - 1. **源代码**

```C

void searchScore(SCORE\_LINK\_LIST \*head, COURSE\_LINK\_LIST \*head2)

{

int menu\_choice = searchMenuScore();

SCORE\_LINK\_LIST \*tmp = head->next;

SCS\_LINK\_LIST \*result = (SCS\_LINK\_LIST \*)malloc(sizeof(SCS\_LINK\_LIST));

SCS\_LINK\_LIST \*rHead = result;

SCS\_LINK\_LIST \*newNode = NULL;

STU\_LINK\_LIST \*tmpNode = NULL;

COURSE\_LINK\_LIST \*tmpNode2 = NULL;

char value[20]; // 搜索内容

int flag = 1;

while (tmp != NULL)

{

switch (menu\_choice)

{

case 1:

if (flag == 1)

{

printf("请输入学号：");

scanf("%s", value);

}

if (strcmp(tmp->sno, value) == 0)

{

if (hasCourse(head2, tmp->cno, 0) == 1)

{

tmpNode2 = searchCourse(head2, tmp->cno, 0)->next;

newNode = cloneSCSBySno(tmp, value, tmpNode2->cName);

rHead->next = newNode;

rHead = rHead->next;

tmpNode2 = NULL;

}

}

break;

case 2:

if (flag == 1)

{

printf("请输入课程名：\n");

scanf("%s", value);

}

if (hasCourse(head2, value, 1) == 1)

{

tmpNode2 = searchCourse(head2, value, 1)->next;

if (strcmp(tmp->cno, tmpNode2->no) == 0)

{

newNode = cloneSCSByCname(tmp);

rHead->next = newNode;

rHead = rHead->next;

tmpNode2 = NULL;

}

}

else

{

printf("课程不存在");

}

break;

default:

printf("无效的选择");

break;

}

flag = 0;

tmp = tmp->next;

}

int flag2 = 1;

result = result->next;

while (result != NULL)

{

switch (menu\_choice)

{

case 1:

if (flag2 == 1)

{

printf("%-8s%-10s%-12s%-14s%-0s\n", "学号", "课程名", "平时成绩", "卷面成绩", "综合成绩");

}

if (result != NULL)

{

printf("%-8s%-8s%-8d%-8d%-0d\n", result->sno, result->cName, result->uGrade, result->pGrade, result->tGrage);

}

break;

case 2:

if (flag2 == 1)

{

printf("%-8s%-10s%-0s\n", "学号", "课号", "综合成绩");

}

if (result != NULL)

{

printf("%-8s%-10s%-0d\n", result->sno, result->cno, result->tGrage);

}

break;

default:

break;

}

flag2 = 0;

result = result->next;

}

}

```

* + - 1. **界面实现**

****

* + - 1. **使用说明**

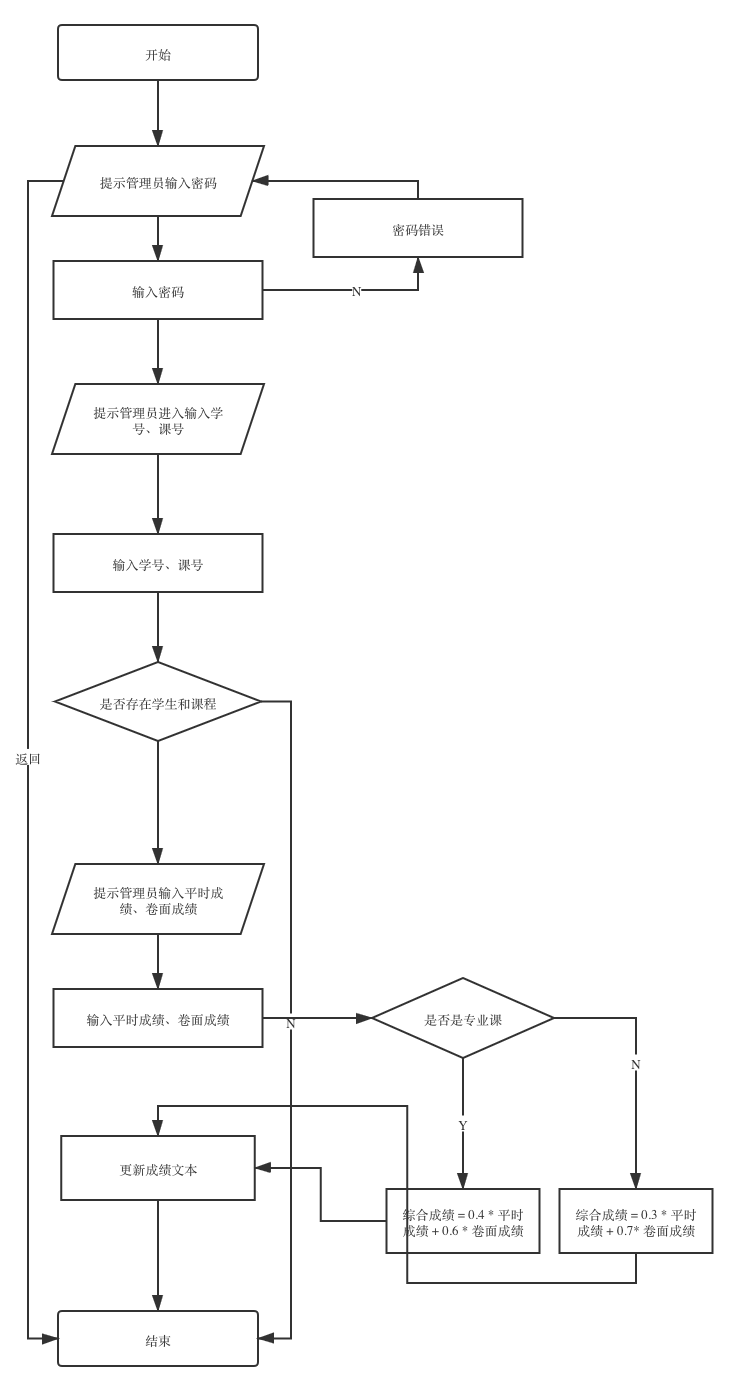
管理员进入成绩管理，选择 3-查询功能，给出查询条件，管理员可以进行 1-学号查询、2-课程名查询，当成绩文件中包含查询学号时，打印所有这个学号同学的成绩记录；如果选择通过课程名查询当输入的课程名不包含在课程文件中，会提示课程不存在，如存在再查询成绩文件中课号与之对应的记录，最后打印记录。

* + - 1. **技术难点**

由于查询功能牵扯到多链表遍历，比如搜索条件为课程名，则需要先查询寻到该课程名对应的课号，不存在则会给出查询失败；如存在遍历成绩链表查询该课号的记录，比较繁琐。

### 2.4.1 修改功能

* + - 1. **修改功能流程图**

****

* + - 1. **功能描述**

修改前，输入管理员密码，如果成功，再根据管理员输入的学号和课号，找到对应的成绩记录节点，然后将平时成绩和卷面成绩进行修改，当修改完成后，更新 score.txt 文件；如果没对应的记录，提示修改失败。

* + - 1. **源代码**

```C

void updateScore(SCORE\_LINK\_LIST \*head)

{

SCORE\_LINK\_LIST \*tmp = head->next;

int grade = 0;

char sno[20];

char cno[20];

int uGrade;

int pGrade;

int flag = 0;

char \*cParrtern = "P\\d\*";

char \*sParrtern = "S\\d\*";

printf("请输入要修改成绩的学号：\n");

scanf("%s", sno);

printf("请输入要修改成绩的课号：\n");

scanf("%s", cno);

printf("请输入平时成绩：\n");

scanf("%d", &uGrade);

printf("请输入平时成绩：\n");

scanf("%d", &pGrade);

int isCCourse = isValidCourse(cno, cParrtern);

int isSCourse = isValidCourse(cno, sParrtern);

while (tmp != NULL)

{

if (strcmp(tmp->sno, sno) == 0 && strcmp(tmp->cno, cno) == 0)

{

tmp->uGrade = uGrade;

tmp->pGrade = pGrade;

if (isSCourse == 0)

{

tmp->tGrage = tmp->uGrade \* 0.4 + tmp->pGrade \* 0.6;

}

if (isCCourse == 0)

{

tmp->tGrage = tmp->uGrade \* 0.3 + tmp->pGrade \* 0.7;

}

flag = 1;

break;

}

tmp = tmp->next;

}

if (flag == 1)

{

printf("修改成功\n");

}

else

{

printf("没有找到修改的记录");

}

}

void scoreManage(SCORE\_LINK\_LIST \*head, STU\_LINK\_LIST \*head2, COURSE\_LINK\_LIST \*head3)

{

int op\_choice;

char sno[20];

char password2[10];

while ((op\_choice = operateSelectScore(" =================== 成绩管理 =================== \n")) != 0)

{

switch (op\_choice)

{

case 1:

printf(" =================== 成绩管理 (添加成绩) =================== \n");

insertScore(head, head2, head3);

writeFileScore(head);

break;

case 2:

printf(" =================== 成绩管理 (显示所有成绩) =================== \n");

displayScore(head);

break;

case 3:

printf(" =================== 成绩管理 (查询成绩) =================== \n");

searchScore(head, head3);

break;

case 4:

printf("请输入密码：\n");

scanf("%s", &password2);

if (strcmp(&password, &password2) != 0)

{

printf("用户名或密码错误\n");

}

else

{

printf(" =================== 成绩管理 (修改成绩) =================== \n");

updateScore(head);

writeFileScore(head);

}

break;

case 5:

printf(" =================== 成绩管理 (删除成绩) =================== \n");

printf("请输入要删除成绩的学号：\n");

scanf("%s", sno);

deleteScore(head, sno);

writeFileScore(head);

break;

default:

printf("无效的选择\n");

break;

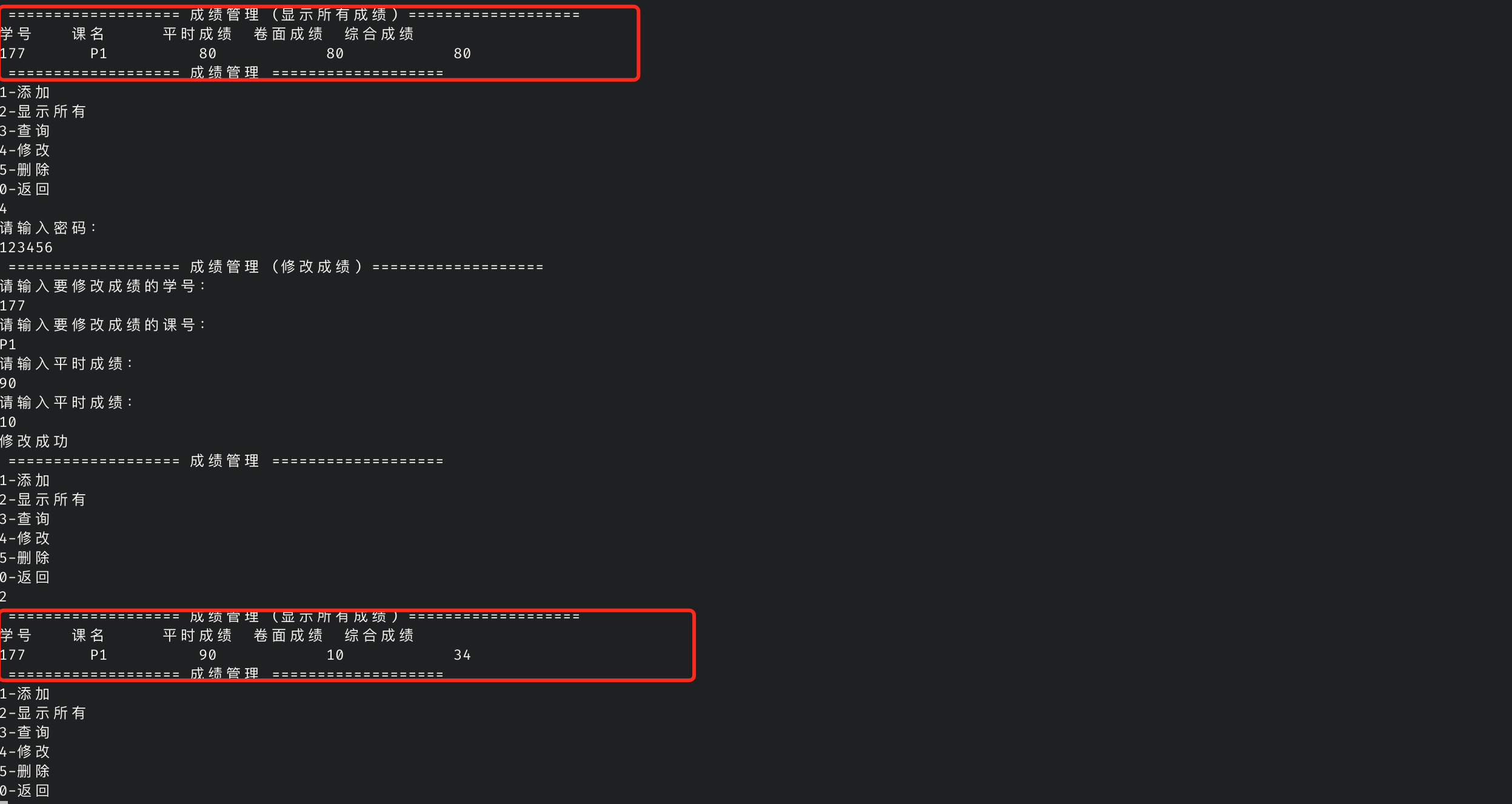
}

}

}

```

* + - 1. **界面实现**

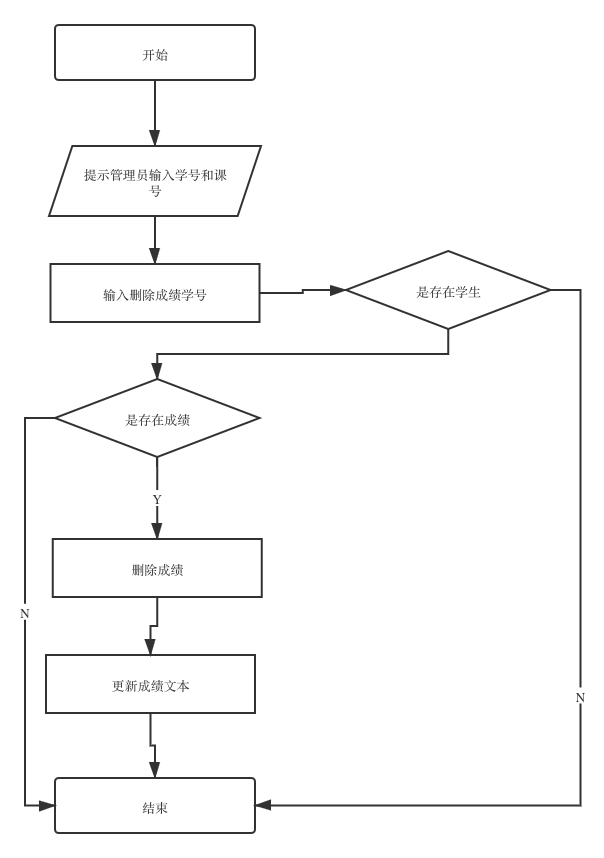
****

* + - 1. **使用说明**

提示管理员输入密码，当输入的密码不正确，提示密码错误，通过后管理员选择 4-修改功能，提示管理员输入要修改的学号和课号，接下来再输入要修改的平时成绩和卷面成绩，系统自动计算新的综合成绩。如果记录未存在，则出现修改失败提示，反之返回功能选择界面。

### 2.4.3 删除成绩功能

* + - 1. **删除流程图**

****

* + - 1. **功能描述**

根据管理员输入的学号，遍历成绩链表，找出其中学号符合的进行删除，并且将删除后的链表结果更新到 score.txt 中

* + - 1. **源代码**

```C

void deleteScore(SCORE\_LINK\_LIST \*head, char sno[20])

{

SCORE\_LINK\_LIST \*tmp = head->next;

SCORE\_LINK\_LIST \*prev = NULL;

while (tmp != NULL)

{

if (strcmp(tmp->sno, sno) == 0)

{

if (prev == NULL)

{

head->next = head->next;

prev = head;

tmp = head->next;

}

else

{

prev->next = tmp->next;

tmp = tmp->next;

}

}

else

{

prev = tmp;

tmp = tmp->next;

}

}

}

```

* + - 1. **界面实现**

****

* + - 1. **使用说明**

管理员选择5-删除功能，管理员根据提示输入要删除的成绩学号，如果没有该学生会提示删除失败，反之，当删除了数据后会提示管理员删除成功。

* + - 1. **技术难点**

由于要与学生的删除功能相关联所以，不能使用进行简单的封装，在封装的同时，还需要提供一个接口，能后接受学生删除时，传递的 sno 参数，从而在删除学生的同时，也能删除他的成绩。

# 系统测试和结果分析

本章是介绍系统的每个模块的功能测试，查看编写的程序是否具有较强的容错性。

## 学生-添加功能

测试1：

输入：录入重复学号 199

输出：系统给出错误提示。

结果分析：学生学号是唯一的。

测试2：

输入：性别录入 0.5

输出：系统给出错误提示。

结果分析：性别必须是字串类型。

测试3：

输入：年龄输入 - 1

输出：系统给出错误提示。

结果分析：年龄最小为 0 岁。

## 学生-删除功能

测试1：

输入：输入不存在学号 177

输出：系统给出错误提示。

结果分析：无法删除学号不在学生列表的数据。

## 学生-查询功能

测试1：

输入：输入不存在学号 177

输出：系统给出错误提示。

结果分析： 无法查询到学号不在学生列表的数据。

测试2：

输入：输入错误的姓名 Dash

输出：系统给出错误提示。

结果分析： 无法查询到姓名不在学生列表的数据。

测试3：

输入：输入错误的性别 -1

输出：系统给出错误提示。

结果分析：性别必须是字串类型。

* 1. **课程-添加功能**

测试1：

输入：输入错误的课号 D3

输出：系统给出错误提示。

结果分析：课号必须以 P 或 S 开头。

测试2：

输入：输入重复的课号 P1

输出：系统给出错误提示。

结果分析：课程的课号必须唯一。

* 1. **成绩-添加功能**

测试1：

输入：输入错误的课号 D3

输出：系统给出错误提示。

结果分析：课号必须以 P 或 S 开头。

测试2：

输入：输入不存在的课号 P2

输出：系统给出错误提示。

结果分析：必须添加已存在的课程的成绩。

测试3：

输入：输入不存在的学号 177

输出：系统给出错误提示。

结果分析：必须添加已存在的学生的成绩。

* 1. **成绩-添加功能**

测试1：

输入：输入错误的课号 D3

输出：系统给出错误提示。

结果分析：课号必须以 P 或 S 开头。

测试2：

输入：输入不存在的课号 P2

输出：系统给出错误提示。

结果分析：必须添加已存在的课程的成绩。

测试3：

输入：输入不存在的学号 177

输出：系统给出错误提示。

结果分析：必须添加已存在的学生的成绩。

* 1. **成绩-修改功能**

测试1：

输入：输入错误的课号 D3

输出：系统给出错误提示。

结果分析：课号必须以 P 或 S 开头。

测试2：

输入：输入不存在的课号 P2

输出：系统给出错误提示。

结果分析：被修改成绩的课号必须是以存在课程的课号。

测试3：

输入：输入不存在的学号 177

输出：系统给出错误提示。

结果分析：被修改成绩的学号必须是以存在学生的学号。

* 1. **成绩-删除功能**

测试1：

输入：输入不存在的学号 177

输出：系统给出错误提示。

结果分析：被删除成绩的学号必须是以存在学生的学号。

测试2：

输入：输入成绩中不存在的学号 199

输出：系统给出错误提示。

结果分析：被删除成绩的学号必须是成绩表中存在的数据。

# 课题收获与总结

## 理论知识应用体会

项目中多次使用了链表和文件的方式进行数据存取，但在实习的代码编写中，我发现链表并不是很适应用于数据的查找，反而是文件的读取操作更容易些；而在增删操作上，又是链表更是适合，文件则没有这么方便，而我的体悟就是项目中最好能合理的使用多种数据结构，取长补短，就比如，使用HashMap来缓存每个学号这样就能提供常量级别的查询时间复杂度，而不需要每次都遍历一遍链表来查询是否存在学生。

## 课题应用软件环境的体会

本次课程设计是在 windows 的环境下使用 C 语言开发，其中编辑器使用 vs 进行 C 代码的编写，感觉一个好的编辑器也会大大提升工作效率，但也不能过分依赖编辑器，做到就算脱离编辑器也能写代码。

## 其它体会

此次小学期课程设计，在指导教师的精心教导下，我学会了如何用 C 编写一个简单的应用程序。在 C 语言的指针运用更加的熟练，也对数据结构中的链表也有了更深的体会。

# 参考文献

[1] 唐浩强.C程序设计（第五版）清华大学出版社.2017.7

[2] [Stephen Prata](https://book.douban.com/search/Stephen%20Prata). C Primer Plus（第五版）中文版 人民邮电出版社. 2005.2