****

**《程序设计课程实践》设计文档**

# 学生程序管理系统程序设计报告

**团队成员：**

**16041321 黄继升**

**16042046 朱伯翰**

完成时间 2019 年 12 月

# 学生成绩管理系统设计报告

## 人员组成及分工

### 人员组成及分工

|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | **姓名** | **任务** | **完成工作量** |
| **16041321** | **黄继生** | **设计程序算法及程序主体** | **50%** |
| **16042046** | **朱伯翰** | **功能测试、界面优化及文案负责** | **50%** |

## 2．开发背景

为了更好的了解学生的学习情况，以对学生进行更详细的了解，对未来的教学有相应的规划，同时为了便于期末总结，我们设计了一套学生成绩管理系统。

在本次课程设计中，操作者根据清单上的提示对于不同的学生进行分数的录入、修改、查询以及获取相应的统计结果，并支持数据的输出。

## 3. 系统功能设计

### 系统功能模块图

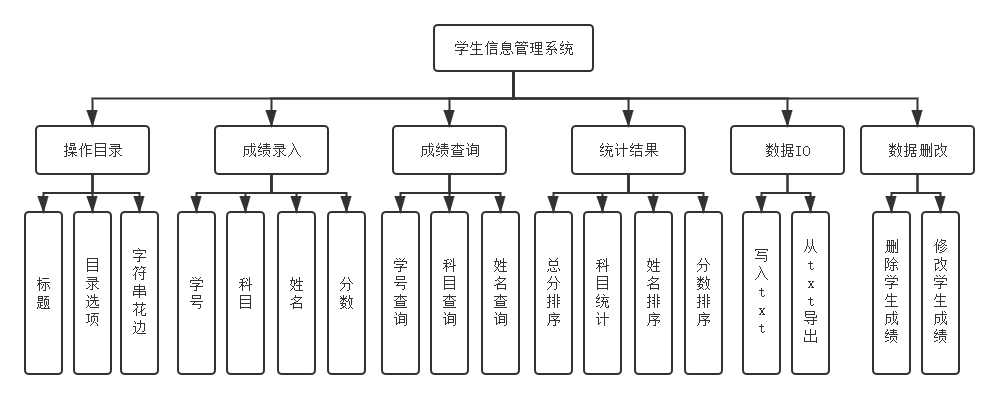


图3-1 系统功能模块图

### 系统业务流程图

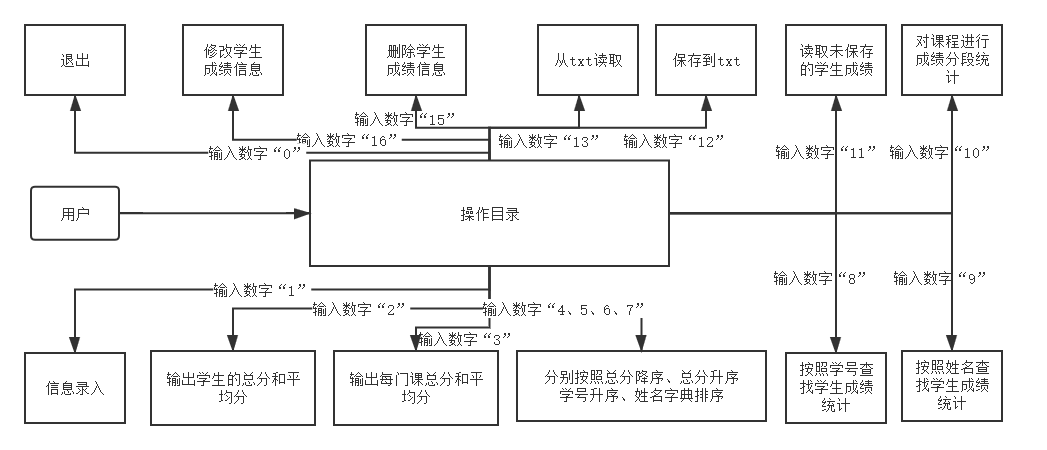


图3-2 业务流程图

## 项目创建

### 系统开发环境要求

本项目的开发及运行环境要求：

操作系统：windows 7, windows 10

开发工具：Code Blocks 17.12, Visual Studio Code

开发语言：C 语言

### 项目创建过程

1、打开Code Block

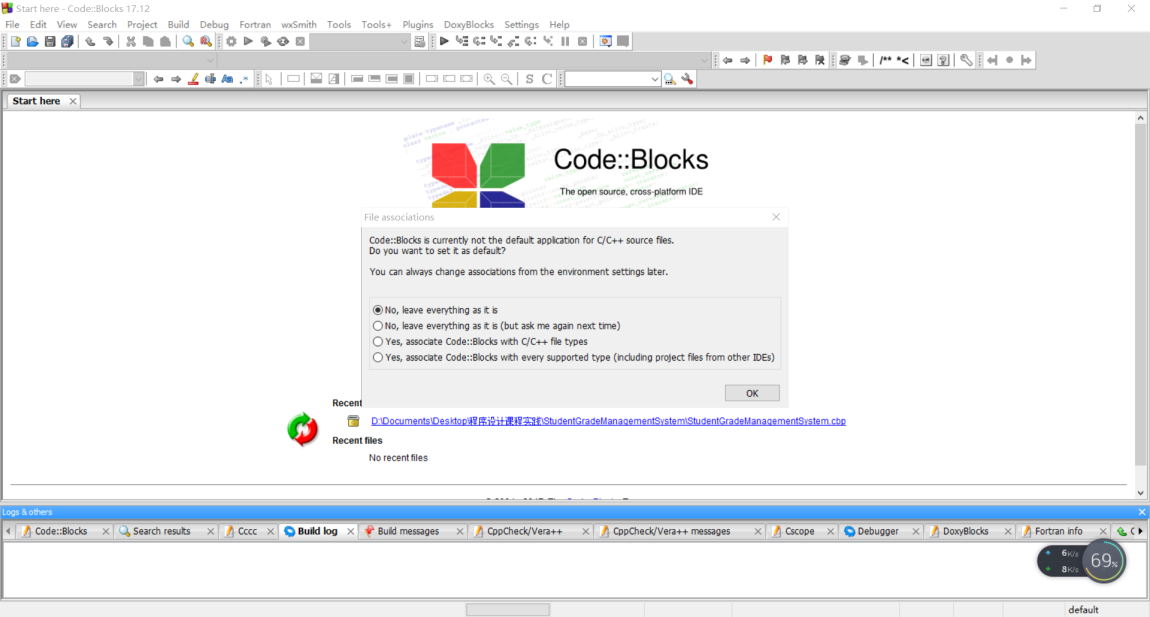


图4-1 Code Block使用界面

#### 2、点击“open an existing project”，并选中“Console application”

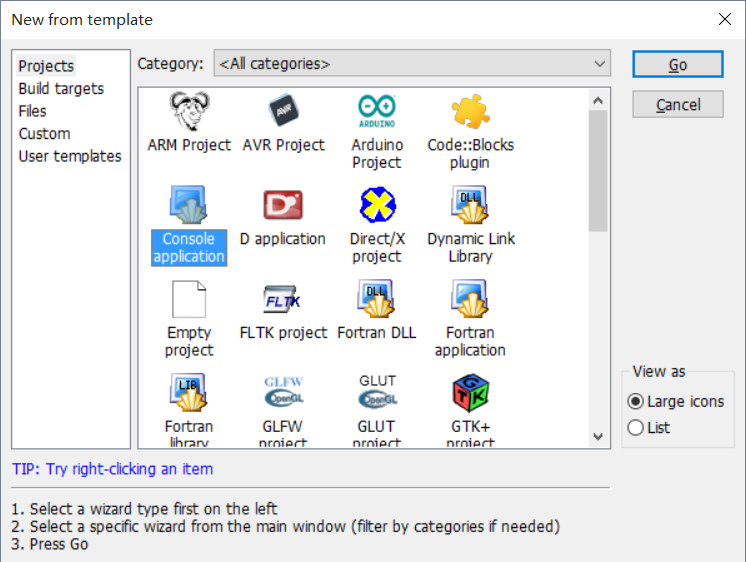


图4-2 工程创建界面

3、选择C，并点击next

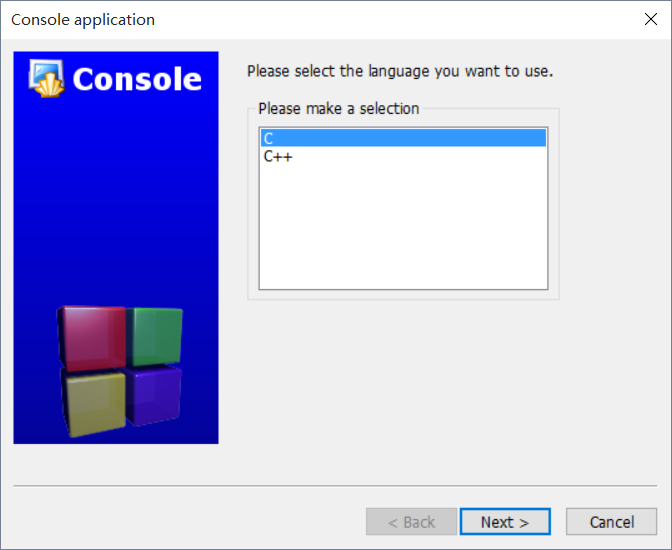


图4-3 创建C工程

4、填写项目名称以及路径。

#### 

图4-4 填写项目名称和工程路径

5、一直点击next默认配置，最后点击Finish完成工程创建。

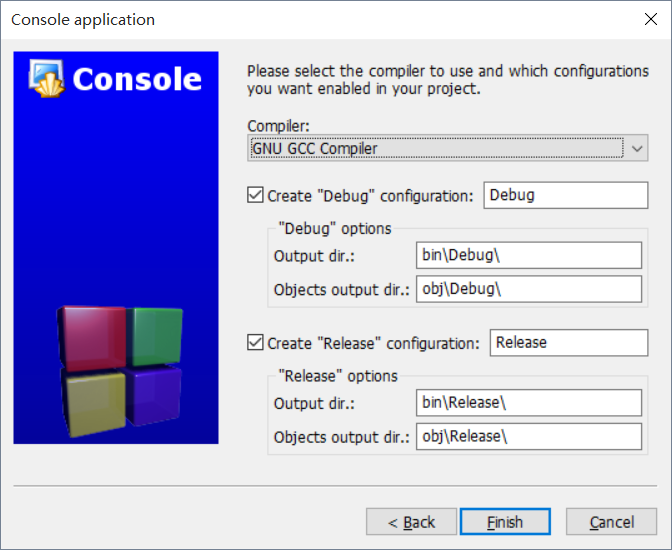


图4-4 完成工程创建

6、点击File->Empty file，创建并命名第一个C文件并编写代码

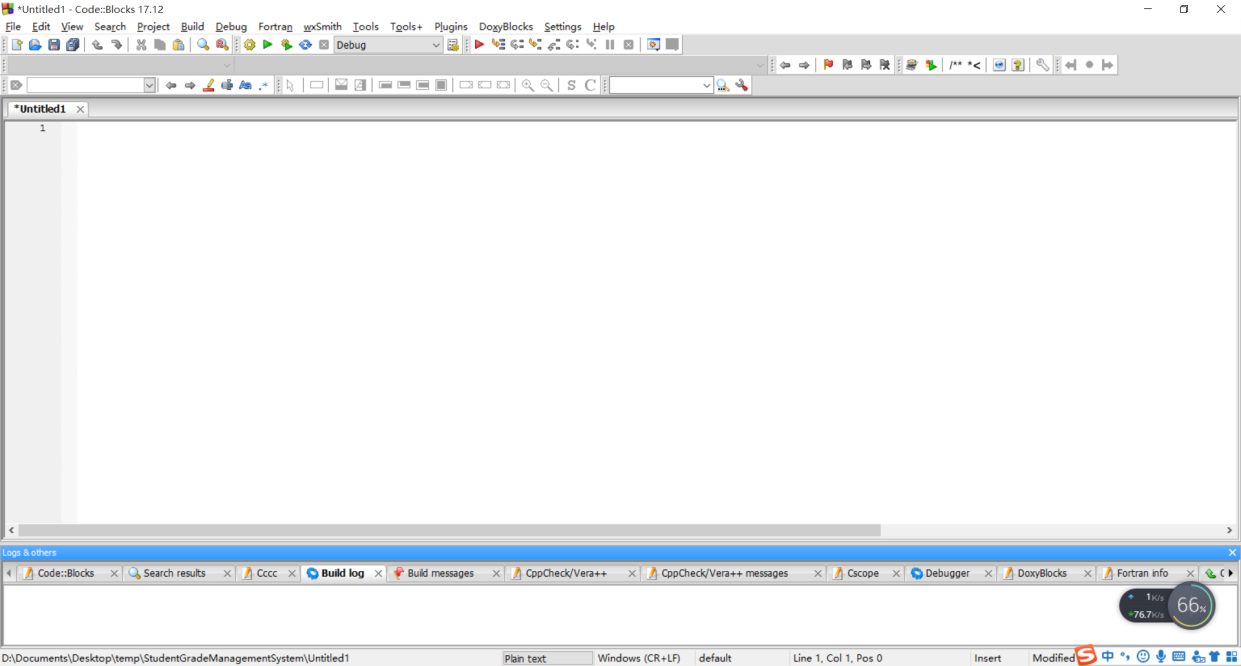


图4-4 创建C文件

## 预处理模块设计

### 文件引用

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <malloc.h>

#include <stdlib.h>

### 宏定义

#define true 1

#define false 0

#define MAX\_LEN 12 /\* 字符串最大长度 \*/

#define STU\_NUM 30 /\* 最大的学生人数 \*/

#define COURSE\_NUM 6 /\* 最大的考试科目数 \*/

#define LEN sizeof(struct Student) /\* 学生结构体所占的字节大小\*/

### 定义全局变量

int n, m; //输入的学生人数以及考试科目数

int i; //输入菜单选项编号

STU \*head; //定义内存中的学生链表头节点

### 函数声明

1. void Print(STU \*head, int n, int m)

输入参数：学生链表，学生数，考试科目数量

输出参数：null

实现功能：输出链表中所有节点

1. void AverSumofEveryStudent(STU \*head)

输入参数：学生链表

输出参数：null

实现功能：计算每门课程的总分和平均分

1. void AverSumofEveryCourse(STU \*head)

输入参数：学生链表

输出参数：null

实现功能：计算每个学生的总分和平均分

1. void StatisticAnalysis(STU \*head, int n, int m)

输入参数：学生链表，学生数，考试科目数量

输出参数：null

实现功能：按类别及比例输出

1. STU \*SortbyScore(STU \*head, int n)

输入参数：学生链表，学生数

输出参数：学生链表头结点

实现功能：按每个学生的总分由高到低排出名次表

1. STU \*SortbyScore1(STU \*head, int n)

输入参数：学生链表，学生数

输出参数：学生链表头结点

实现功能：按每个学生的总分由低到高排出名次表

1. STU \*SortbyNum(STU \*head)

输入参数：学生链表

输出参数：学生链表头结点

实现功能：按学号由小到大排出成绩表

1. STU \*SortbyName(STU \*head, int n)

输入参数：学生链表，学生数，考试科目数量

输出参数：学生链表头结点

实现功能：按姓名的字典顺序排出成绩表

1. STU \*Create(int n, int m)

输入参数：学生数，考试科目数量

输出参数：学生链表头结点

实现功能：创建链表并录入信息

1. void SearchbyNum(STU \*head, int n, int m)

输入参数：学生链表，学生数，考试科目数量

输出参数：null

实现功能：按学号查询学生排名及其考试成绩

1. void SearchbyName(STU \*head, int n, int m)

输入参数：学生链表，学生数，考试科目数量

输出参数：null

实现功能：按姓名查询学生排名及其考试成绩

1. void WritetoFile(STU \*head, int n, int m)

输入参数：学生链表，学生数，考试科目数量

输出参数：null

实现功能：将每个学生的纪录信息写入文件

1. STU \*ReadfromFile(STU \*head, int \*n, int \*m)

输入参数：学生链表，学生数，考试科目数量

输出参数：学生链表头结点

实现功能：从文件中读出每个学生的纪录信息并显示

1. int DeleteFromFile(STU \*head, int \*n, int \*m);

输入参数：学生链表，学生数，考试科目数量

输出参数：不存在数据或没有数据返回0，成功删除返回1

实现功能：物理删除(删除在文件中的数据)

1. STU \*getAllMessageFromFile(STU \*head, int count0[], int count1[], int \*count2)

输入参数：学生链表，学生数，考试科目数量

输出参数：学生链表头结点

实现功能：获取文件中所有的学生成绩数据，并组成单链表

1. STU \*ModifyScore(STU \*head)

输入参数：学生链表，学生数，考试科目数量

输出参数：学生链表头结点

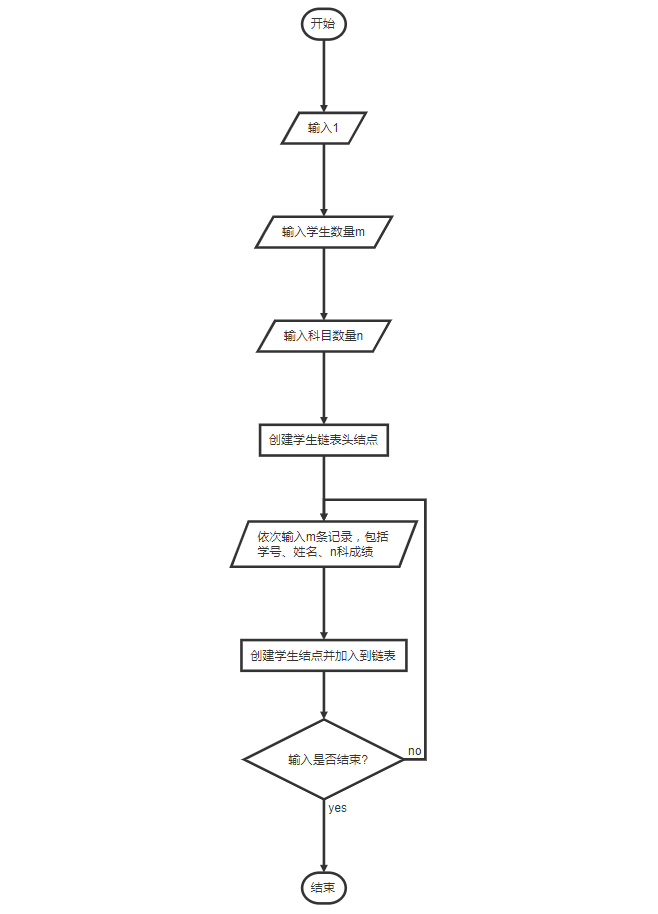
实现功能：用于修改学生某个科目的成绩信息

## 模块详细设计

#### 6.1 成绩录入模块

#### （1）录入新的学生成绩记录（临时区）

* 功能描述：用户在系统主界面输入数字 “1” ，进入录入成绩功能，根据提示依次输入学生数量、考试科目数量，并依次输入每个学生的学号、姓名和各科成绩，最后按下回车将成绩提交到临时区。
* 算法流程：

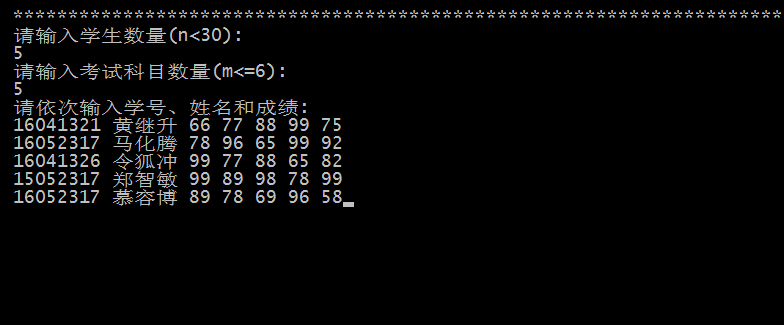


* 相关函数：

int Menu(void); //函数1：创建菜单函数

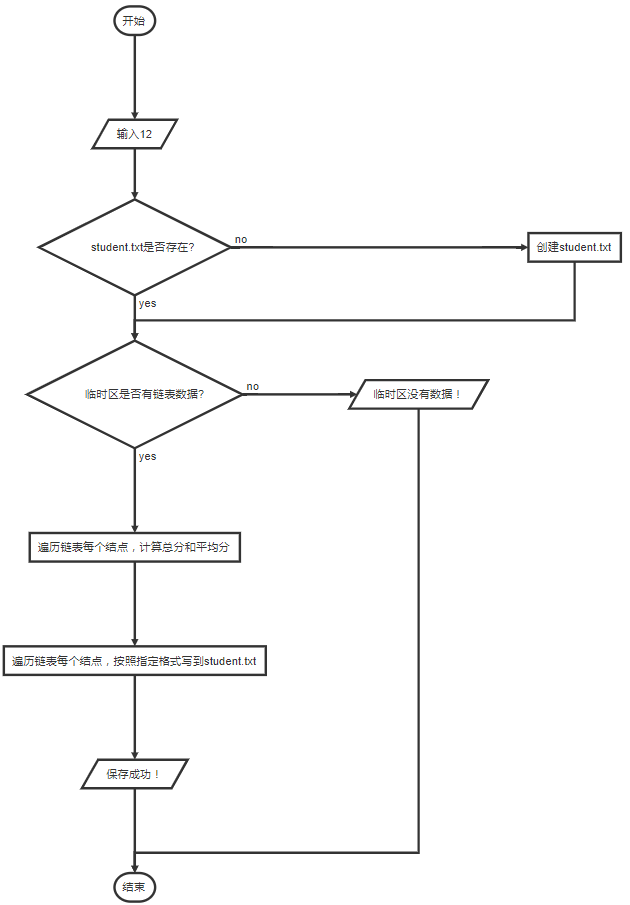
STU \*Create(int n, int m); //函数10：创建链表并录入信息

* 测试结果：



#### （2）将临时区数据保存到文件中

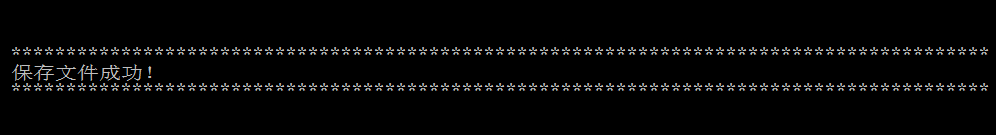
* 功能描述：用户在系统主界面输入数字 “12” ，即可将临时区的学生成绩信息保存到 student.txt 文件中
* 算法流程：

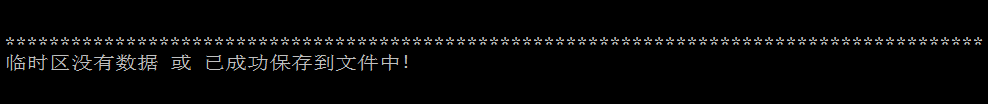


* 相关函数

void WritetoFile(STU \*head, int n, int m); //函数13：将每个学生的纪录信息写入文件

* 测试结果

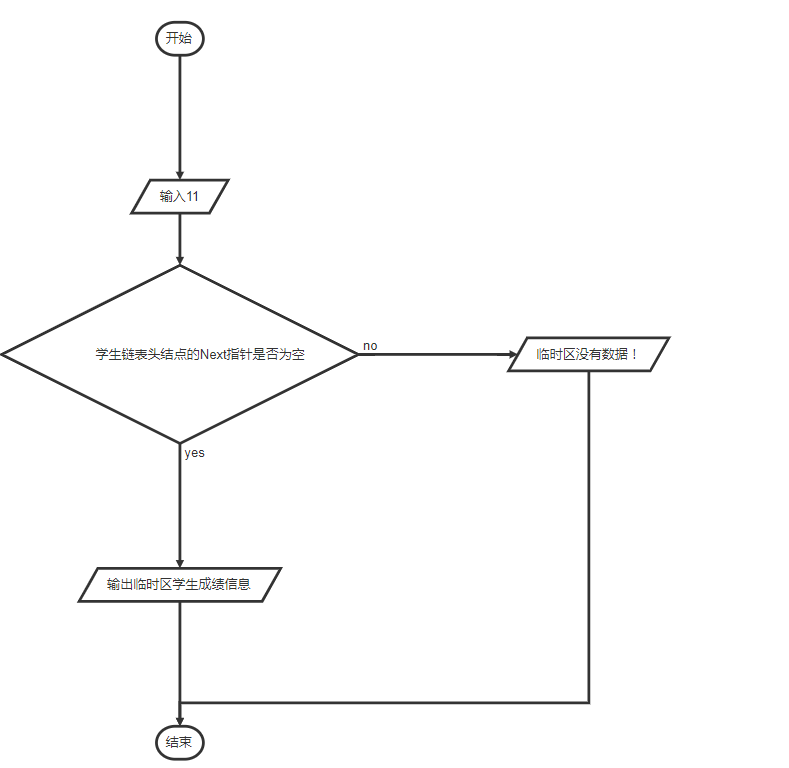




#### 6.2 成绩导出模块

#### （1）导出未保存的学生成绩信息（临时区）

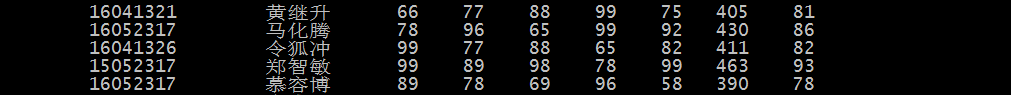
* 功能描述：用户在系统主界面输入数字 “11” ，进入临时区数据导出模块，此时界面会输出仍未保存的学生成绩数据，或临时区没有数据的提示信息。
* 算法流程：

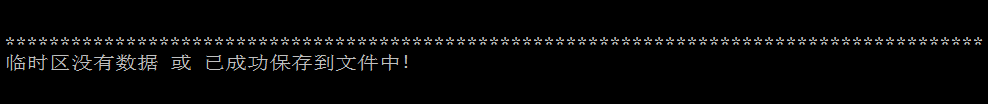


* 相关函数：

void Print(STU \*head, int n, int m); //函数2：打印函数

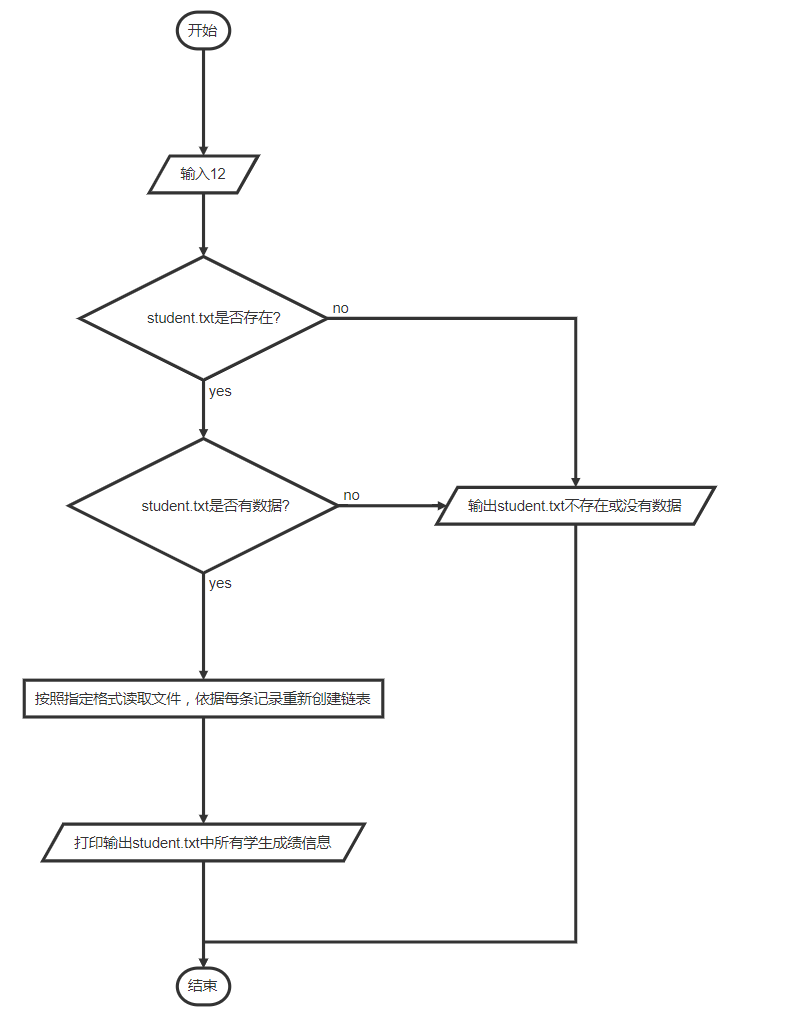
* 测试结果





#### （2）导出已保存的学生成绩信息（文件）

* 功能描述：用户在系统主界面输入数字 “12” ，进入文件数据导出模块，此时界面会输出保存在 student.txt 的所有学生成绩。如果文件不存在或没有数据，则会输出 “student.txt不存在或没有数据"的提示信息。
* 算法流程：

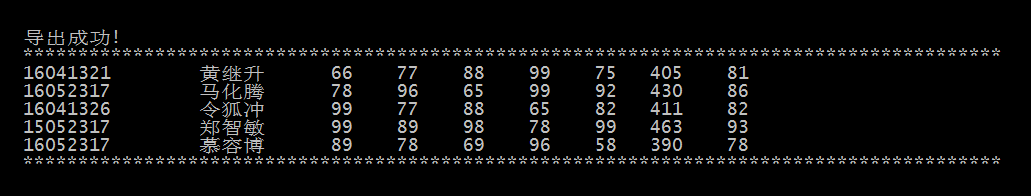


* 相关函数：

STU \*ReadfromFile(STU \*head, int \*n, int \*m); //函数14：从文件中读出每个学生的纪 录信息并显示

void Print(STU \*head, int n, int m); //函数2：打印函数void Print(STU \*head, int n, int m);

* 测试结果：



#### 6.3 排序模块

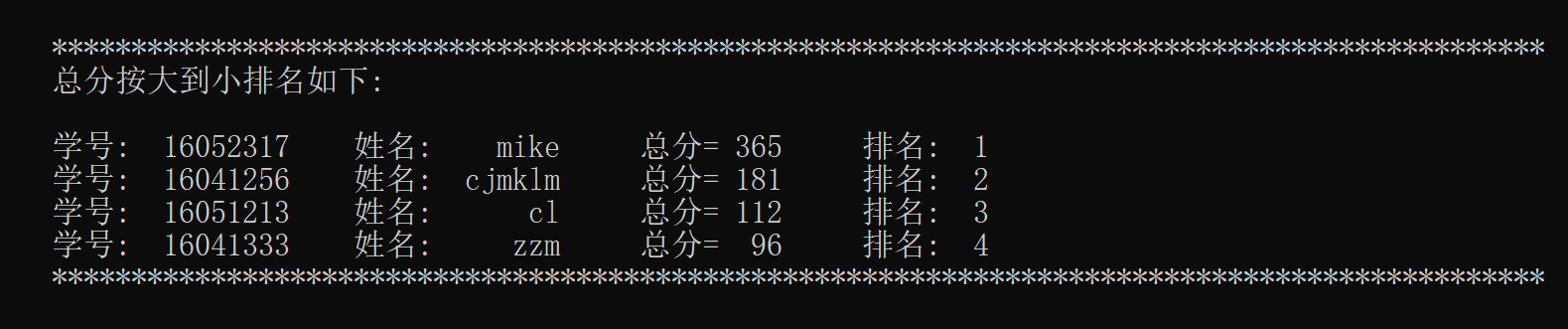
（1）根据成绩总分的升序、降序进行排序

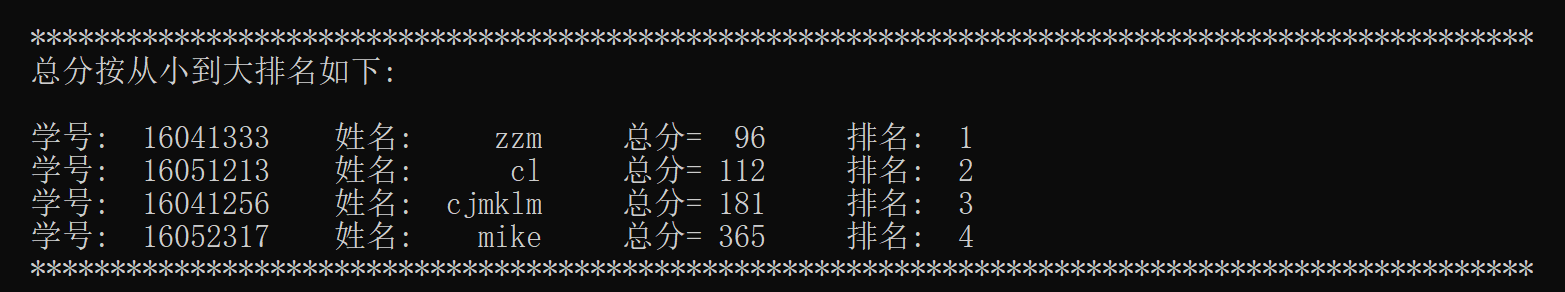
* 功能描述：用户在系统主界面输入“4”进入降序排列，输入“5”进入升序排列。
* 算法流程：首先调用getAllMessageFromFile函数从文件读取已经存入的信息，获得链表的头结点，再遍历链表，将总分作为对比依据进行冒泡排序，最后输出。
* 相关函数：

STU \*SortbyScore(STU \*head, int n); //函数6：按每个学生的总分由高到低排出名次表

STU \*SortbyScore1(STU \*head, int n); //函数7：按每个学生的总分由低到高排出名次表

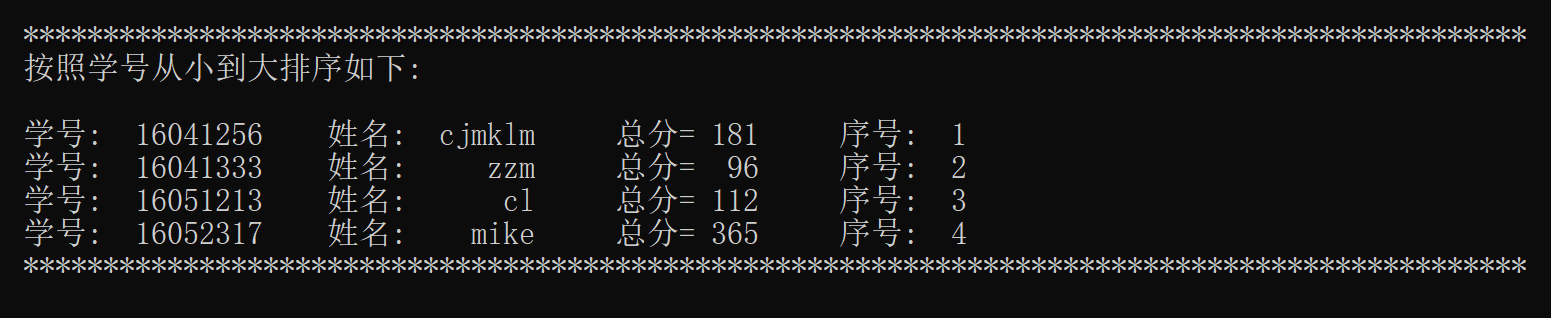
* 测试结果：





（2）根据学号的升序进行排序

* 功能描述：用户在系统主界面输入“6”进入。
* 算法流程：首先调用getAllMessageFromFile函数从文件读取已经存入的信息，获得链表的头结点，再遍历链表，将学号作为对比依据进行冒泡排序，最后输出。
* 相关函数：与7.1的算法相似
* 测试结果：

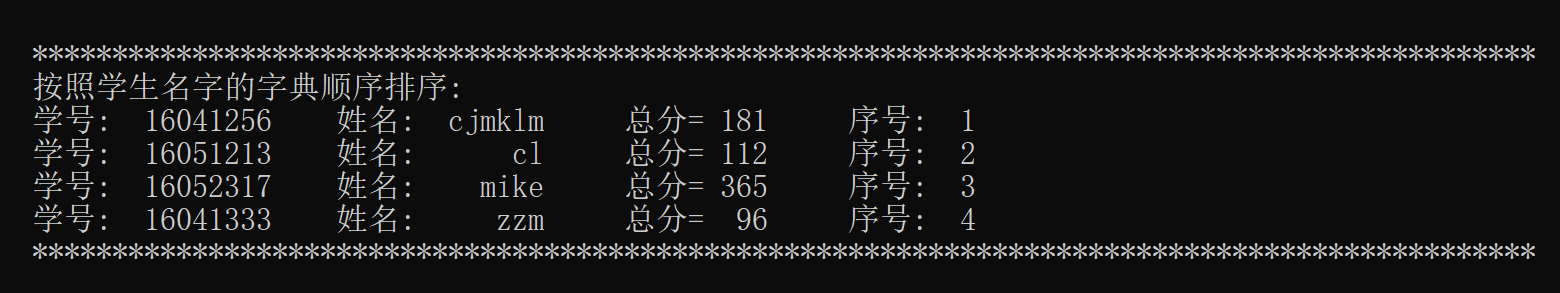


（3）根据学生姓名的字典序进行排序

* 功能描述：用户在系统主界面输入“7”进入。
* 算法流程：首先调用getAllMessageFromFile函数从文件读取已经存入的信息，获得链表的头结点，再遍历链表，将姓名首字母的字典序作为对比依据进行冒泡排序，最后输出。
* 相关函数：

STU \*SortbyNum(STU \*head); //函数8：按学号由小到大排出成绩表

* 测试结果：



## 6.4 UI界面设计模块

* 功能描述：展示操作界面
* 算法流程：

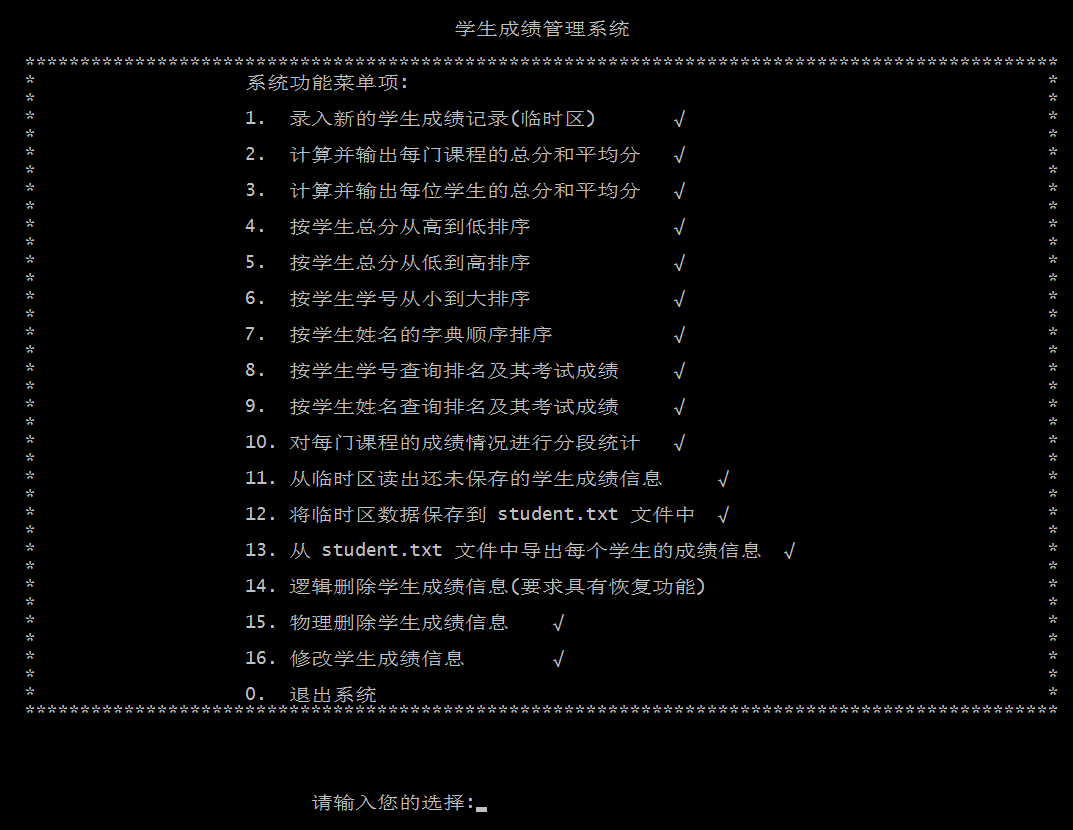
1、主界面根据序号展示相关操作，使用\*作为边框

2、具体操作界面首先清屏，以实现更加简洁的展示便于操作，并使用\*号分割

* 相关函数：

int Menu(void); //函数1：创建菜单函数

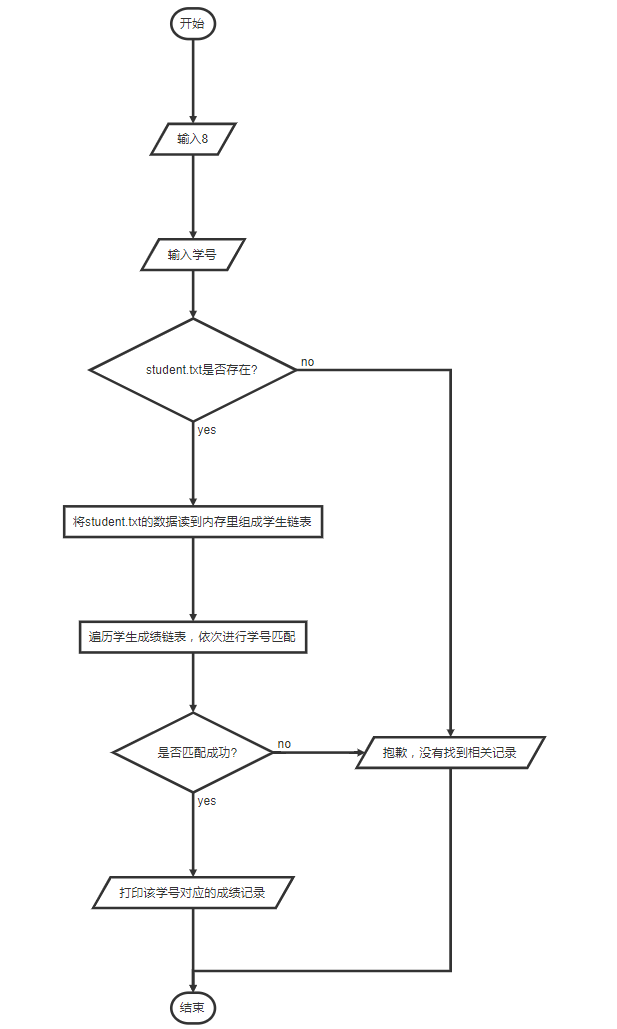
* 测试结果：



## 6.5 条件查询模块

### （1） 按学生学号查询排名及其考试成绩

* 功能描述：用户在系统主界面输入数字 “8” ，进入按学生学号查询排名及其考试成绩，此时按照界面提示输入查询的学号。如果学号不存在或 student.txt 没有数据，则会输出 “抱歉，没有找到相关记录"的提示信息。
* 算法流程：



* 相关函数：

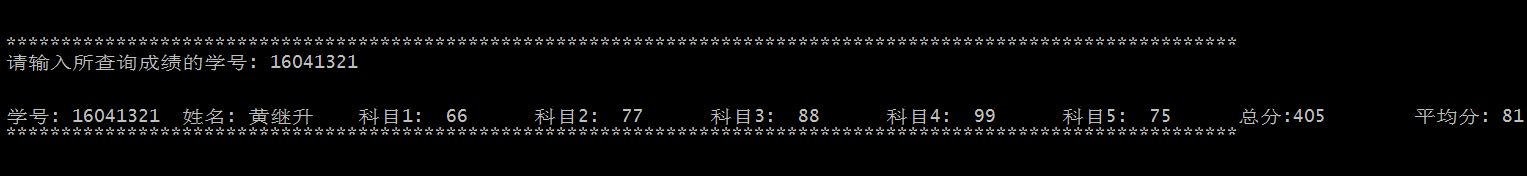
//函数11：按学号查询学生排名及其考试成绩

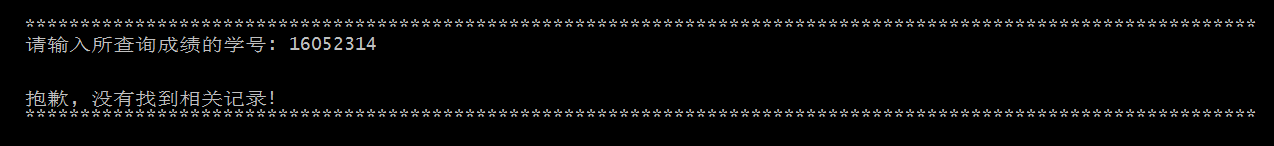
void SearchbyNum(STU \*head, int n, int m);

//函数16：获取文件中所有的学生成绩数据，并组成单链表

STU \*getAllMessageFromFile(STU \*head, int count0[], int count1[], int \*count2);

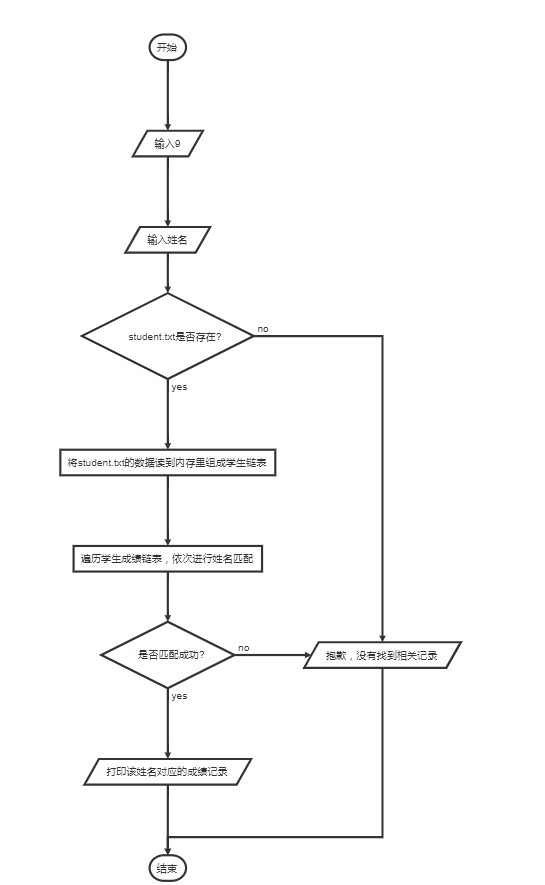
* 测试结果





### （2）按学生姓名查询排名及其考试成绩

* 功能描述：用户在系统主界面输入数字 “9” ，进入按学生学号查询排名及其考试成绩，此时按照界面提示输入查询的姓名。如果学号不存在或 student.txt 没有数据，则会输出 “抱歉，没有找到相关记录"的提示信息。
* 算法流程：



* 相关函数：

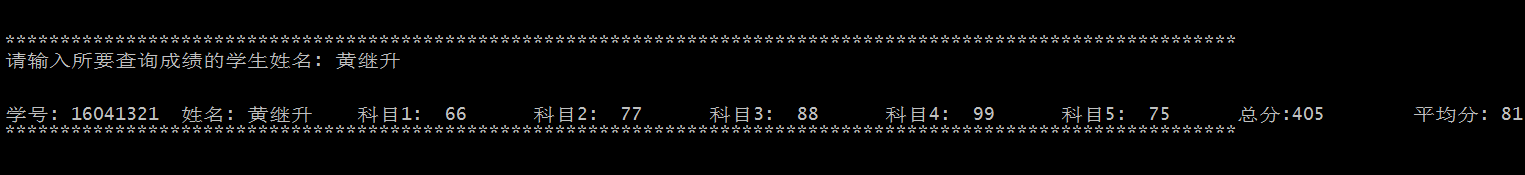
//函数12：按姓名查询学生排名及其考试成绩

void SearchbyName(STU \*head, int n, int m);

//函数2：打印函数

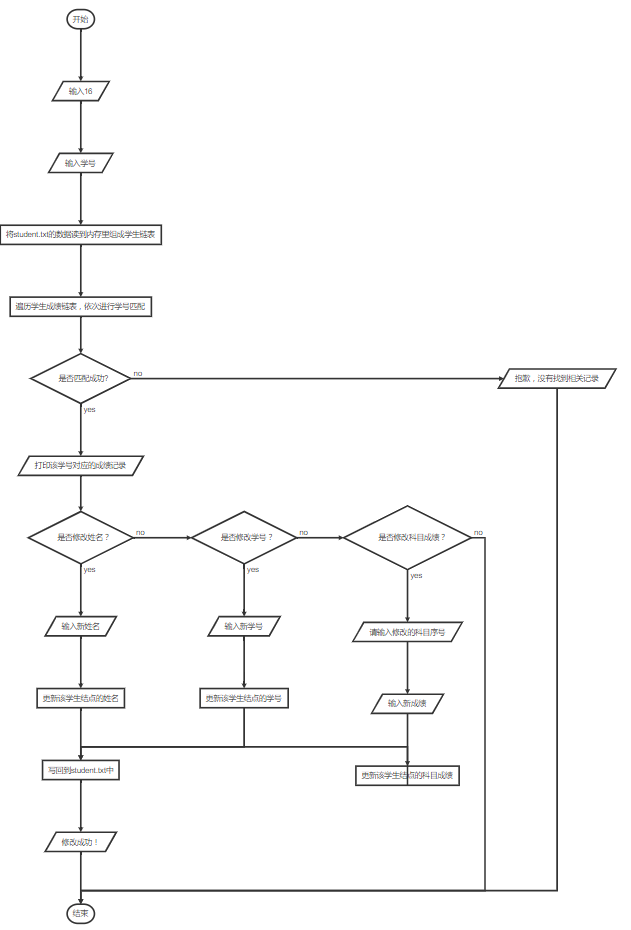
void Print(STU \*head, int n, int m);

* 测试结果：



## 6.6 成绩修改模块

* 功能描述：用户在系统主界面输入数字 “16” ，此时按照界面提示，输入即将修改的记录所对应的学号。如果学号不存在，则会输出 “抱歉，没有找到相关记录"的提示信息。否则，系统会打印出该学号对应的成绩记录并提示你需要修改的是学号、姓名还是科目成绩。如果是学号，则输入新的学号进行提交；如果是姓名，则输入新的姓名并进行提交；如果是科目成绩，则先输入是该科目的序号，再输入新的分数进行提交。
* 算法流程：



* 相关函数

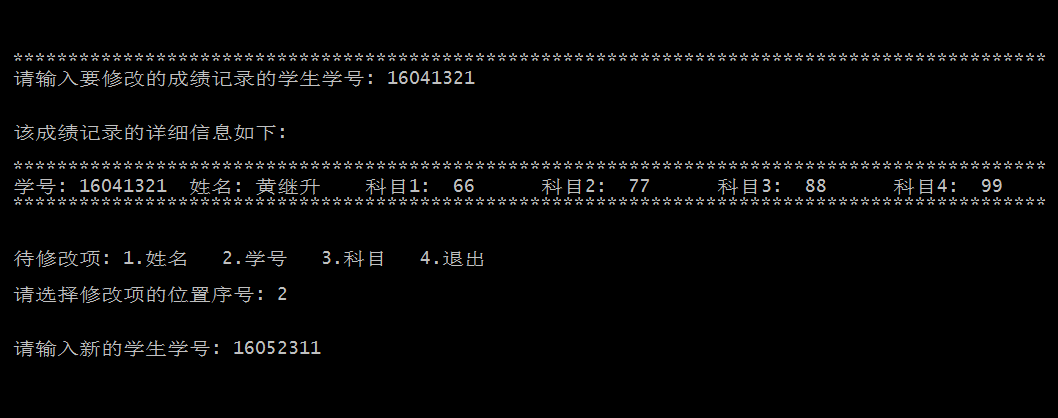
//函数17 用于修改学生成绩信息

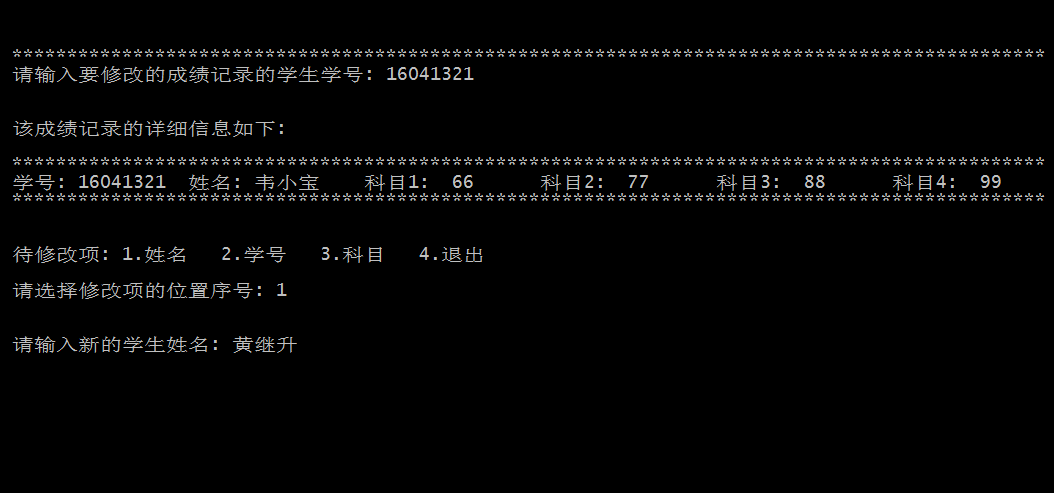
STU \*ModifyScore(STU \*head);

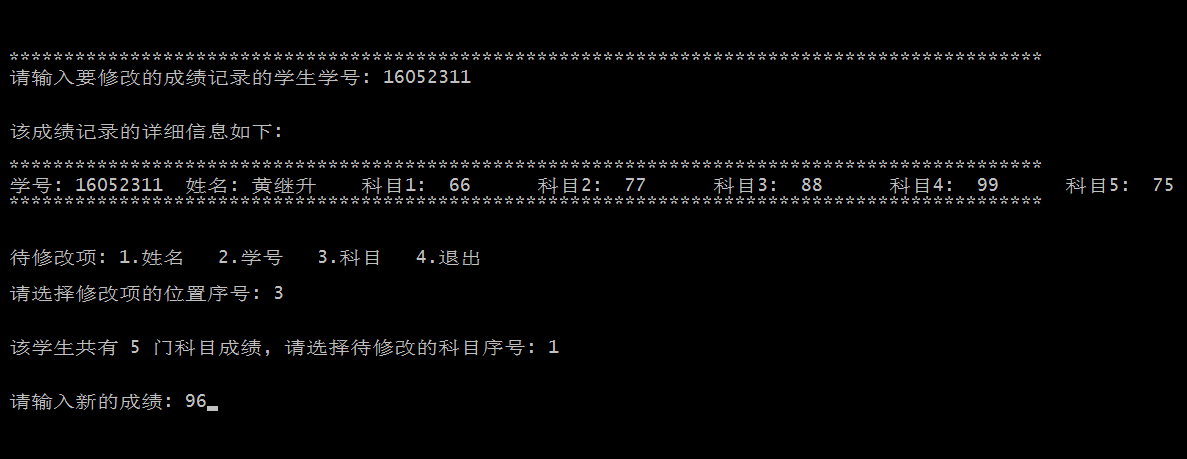
//函数16：获取文件中所有的学生成绩数据，并组成单链表

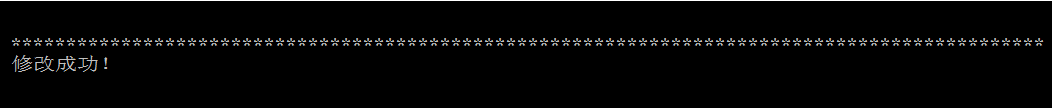
STU \*getAllMessageFromFile(STU \*head, int count0[], int count1[], int \*count2);

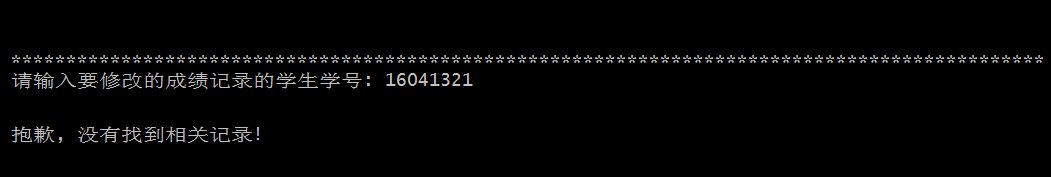
* 测试结果





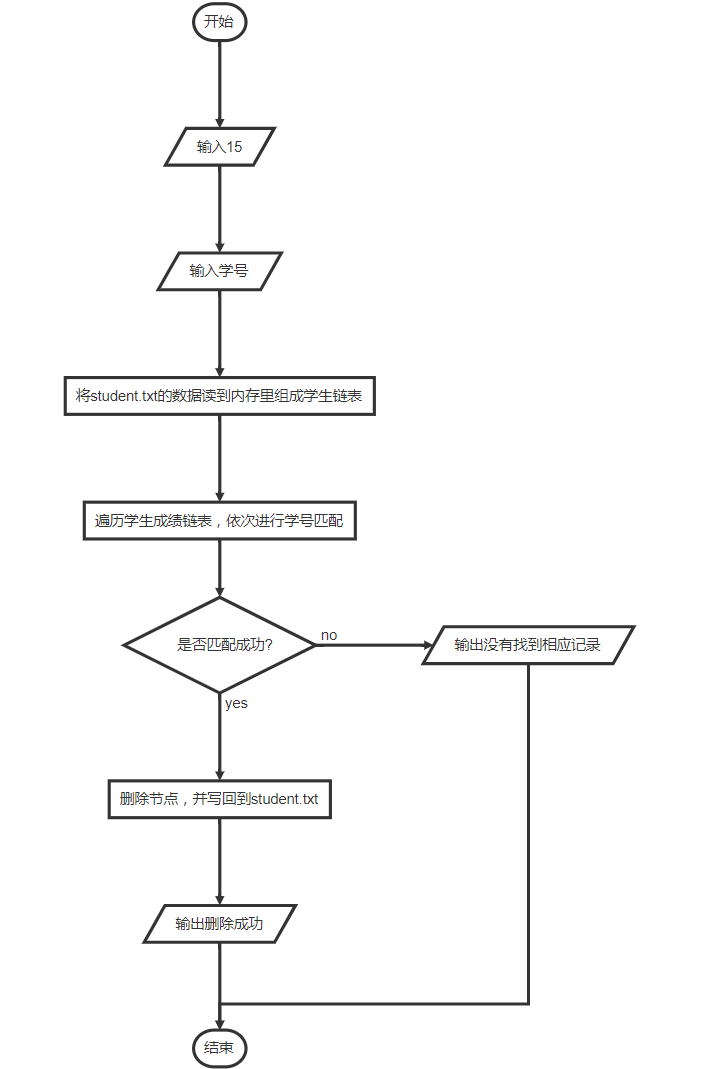






## 6.7 成绩删除

* 功能描述：用户在系统主界面输入数字 “15” ，此时按照界面提示，输入即将删除的记录所对应的学号。如果学号不存在，则会输出 “抱歉，没有找到相关记录"的提示信息。否则，系统将会删除文件中该学号所在的成绩记录，并打印出删除成功的提示。
* 算法流程：

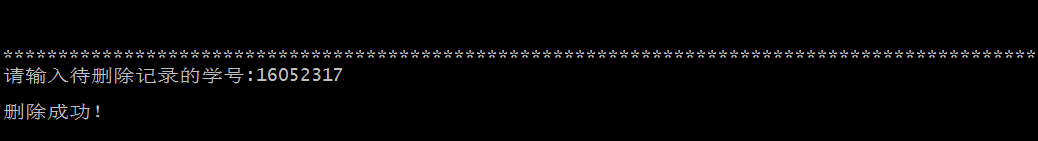


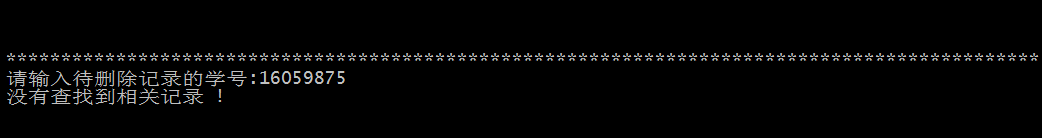
* 相关函数：

//函数15: 物理删除(删除在文件中的数据)

int DeleteFromFile(STU \*head, int \*n, int \*m);

* 测试结果：

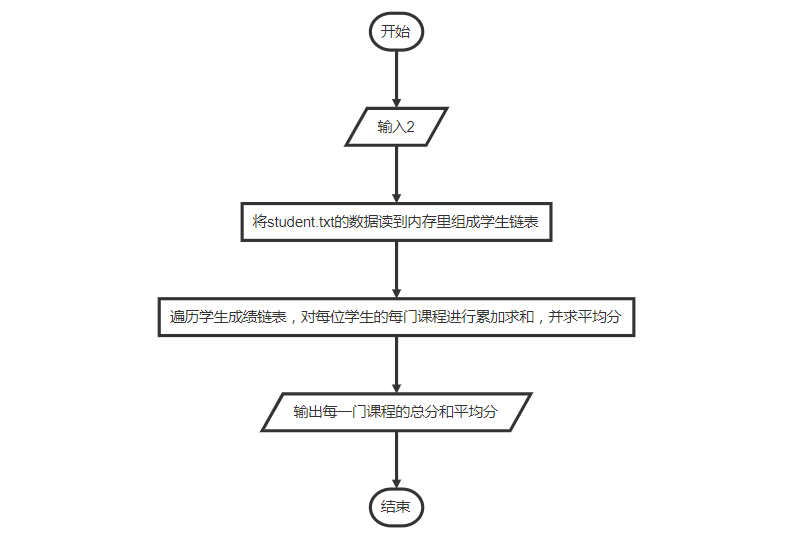




## 6.8 成绩统计计算

### （1） 计算并输出每门课程的总分和平均分

* 功能描述：用户在系统主界面输入数字 “2” ，系统会输出每一门课程的所有学生的总分和平均分
* 算法流程：



* 相关函数：

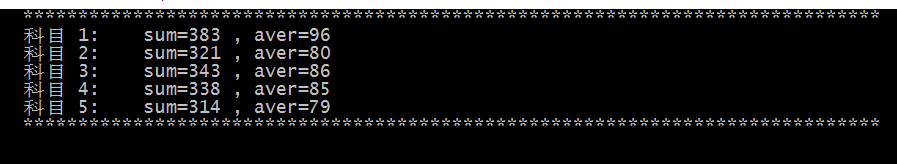
函数3：计算每门课程的总分和平均分

void AverSumofEveryStudent(STU \*head);

//函数16：获取文件中所有的学生成绩数据，并组成单链表

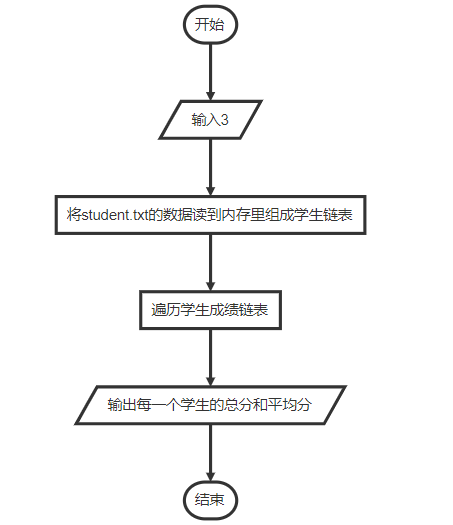
STU \*getAllMessageFromFile(STU \*head, int count0[], int count1[], int \*count2);

* 测试结果



### （2）计算并输出每位学生的总分和平均分

* 功能描述：用户在系统主界面输入数字 “3” ，系统会输出每一门课程的所有学生的总分和平均分
* 算法流程：



* 相关函数：

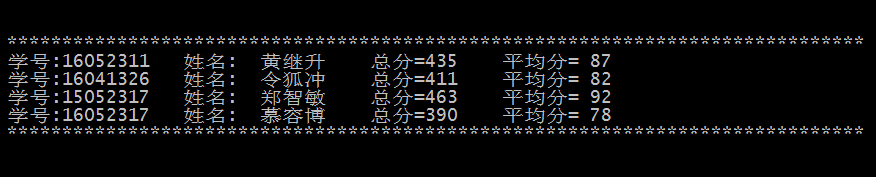
//函数4：计算每个学生的总分和平均分

void AverSumofEveryCourse(STU \*head);

//函数16：获取文件中所有的学生成绩数据，并组成单链表

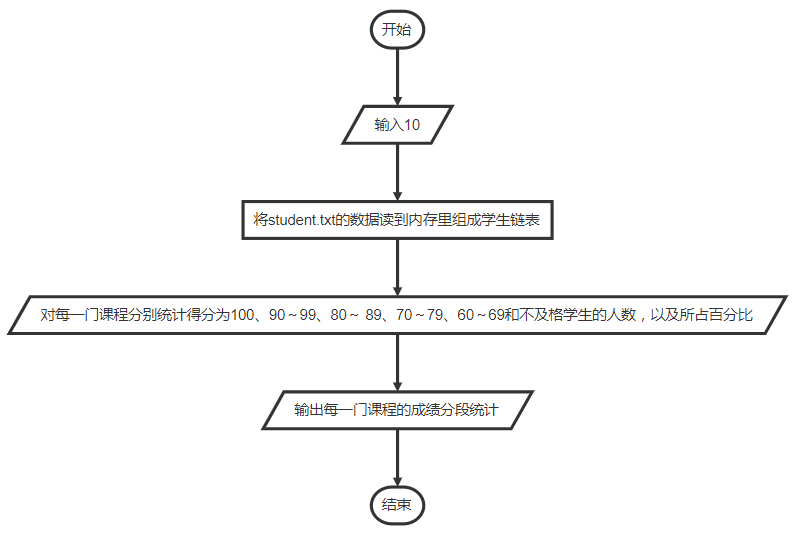
STU \*getAllMessageFromFile(STU \*head, int count0[], int count1[], int \*count2);

* 测试结果：



### （3） 对每门课程的成绩情况进行分段统计

* 功能描述：用户在系统主界面输入数字 “10” ，系统会输出按课程得分为100、90～99、80～ 89、70～79、60～69和不及格学生的人数，以及所占百分比
* 算法流程：



* 相关函数：

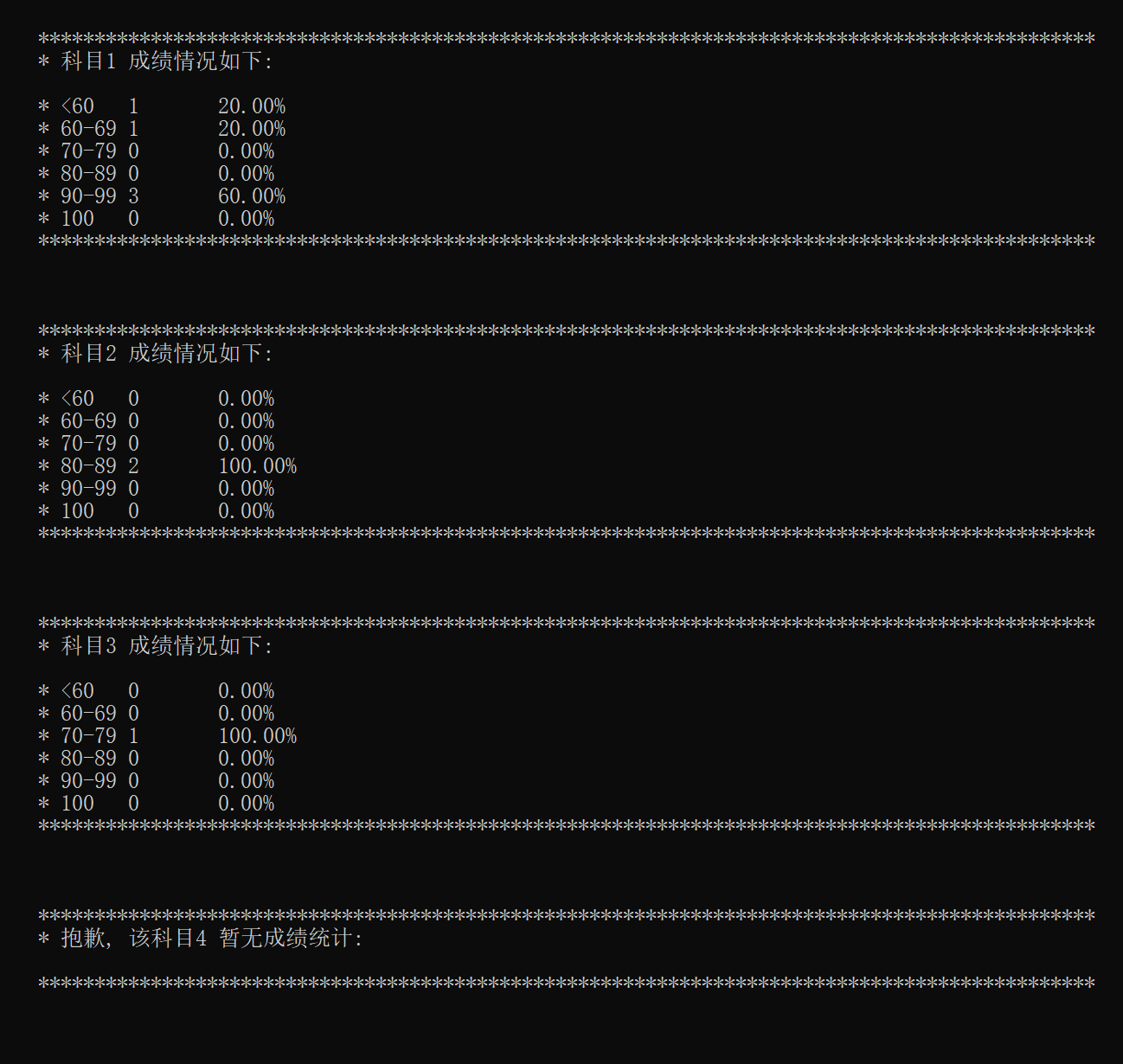
//函数5：按类别及比例输出

void StatisticAnalysis(STU \*head, int n, int m);

//函数16：获取文件中所有的学生成绩数据，并组成单链表

STU \*getAllMessageFromFile(STU \*head, int count0[], int count1[], int \*count2);

* 测试结果：



## 7 项目创新点

该项目的亮点在于：之前对学生的成绩进行各种条件的排序时，由于单链表不像数组那样，可以从后往前进行遍历（注：一开始没有专门去设置尾指针，所以前面已经写好的代码比较多，后面也不好为了增加个尾指针而进行更改），因此为了能够对单链表引入快速排序算法，提高排序效率，所以我们的具体思路如下：

两个指针p、q都从头结点开始往后遍历, 开始的时候p指针指向头结点(key), q指针指向p的下一个结点, q指针从前往后查找比key值小的元素, 当q指针遇到比key值小的元素的时候, p前进一位, 交换p、q指向的元素, 如果q指针遇到的元素比key大, 则p指针不动, q指针继续向前查找, 直到q指针查找到链表末端. 显然q指针比p指针移动速度快, 这就保证了p指针前面的元素都比key小, 此时p指针指向的位置就是key对应的元素在排序后的正确位置, 也就是partition的位置, 因此交换key对应的元素与p指针当前指向的元素, 并以此位置作为partition的临界点。

## 8 收获和建议

黄继升：这学期的程序实践课程设计，我们小组决定做一个学生成绩管理系统。自大一大二上完数据结构以及操作系统等课程之后，便很少继续深入C语言的学习，平时用的较多的是其他编程语言，因此通过这次大作业我们又重新对C语言进行了一次比较系统的复习，重点是指针、链表和文件I/O这一块，是C语言重点和难点，同时经过这次课程设计的开发，加深了对这一块内容的理解。

但是由于时间关系，最后的成果没能做到尽善尽美，该有的bug也没有能够消除：比如一次性录入超过10位学生，在写入到文件之前，打印出这10位学生的信息时最后两位学生的总分和平均分会出现一堆数字，可是写到文件里再读出却是正常的，这一点真的很困惑，但最终也没有解决）。另外对学生成绩的修改采用的方法也比较极端，是通过先将文件信息全部读到内存，再进行修改，最后把修改结果重新覆盖源文件。这是我们这次开发存在的两个最大的不足之处，此外UI界面的设计也值得吐槽。但是通过这次开发我们小组也是重新对基础语言进行了一次系统的复习，虽然我现在基本用的都是python和Java，但是这些编程语言的底层还是用C/C++来实现的，对C语言基础的巩固将会有助于我今后对开发框架底层、JVM和数据库等有更深的了解，让自己的知识积累更加丰富。

朱伯翰：这次的小组作业是建立一个学生成绩管理系统，我和搭档掌握了从功能设计开始构思整体框架，掌握了数组，指针，链表，排序，文件I/O等相关原理以及程序的编写。期间，查阅了相关的书籍和资料，也阅读了一些C语言相关的经典书籍，获得了对C语言更深刻的理解。

程序的编写过程中难免会出现一些bug，这也更利于去查找相关的资料以获得更官方和清晰的解释从而用最优的方法进行编写。比如，C语言内部，数组的首地址是不可以被更改的，所以不能用赋值的方式去交换。这样的困难会让自己积累经验，对C语言的掌握更深。

## 附：源代码清单（每个部分写明是哪位同学完成）

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <malloc.h>

#include <stdlib.h>

#define true 1

#define false 0

#define MAX\_LEN 12 /\* 字符串最大长度 \*/

#define STU\_NUM 30 /\* 最大的学生人数 \*/

#define COURSE\_NUM 6 /\* 最大的考试科目数 \*/

#define LEN sizeof(struct Student) /\* 学生结构体所占的字节大小\*/

//定义学生结构体

typedef struct Student

{

long num; /\* 每个学生的学号 \*/

char name[MAX\_LEN]; /\* 每个学生的姓名 \*/

float score[COURSE\_NUM]; /\* 每个学生COURSE\_NUM门功课的成绩 \*/

float sum; /\* 每个学生的总成绩 \*/

float aver; /\* 每个学生的平均成绩 \*/

struct Student \*next;

}STU;

int Menu(void); //函数1：创建菜单函数（朱伯翰）

void Print(STU \*head, int n, int m); //函数2：打印函数（朱伯翰）

void AverSumofEveryStudent(STU \*head); //函数3：计算每门课程的总分和平均分（朱伯翰）

void AverSumofEveryCourse(STU \*head); //函数4：计算每个学生的总分和平均分（朱伯翰）

void StatisticAnalysis(STU \*head, int n, int m); //函数5：按类别及比例输出（朱伯翰）

STU \*SortbyScore(STU \*head, int n); //函数6：按每个学生的总分由高到低排出名次表（朱伯翰）

STU \*SortbyScore1(STU \*head, int n); //函数7：按每个学生的总分由低到高排出名次表（朱伯翰）

STU \*SortbyNum(STU \*head); //函数8：按学号由小到大排出成绩表（朱伯翰）

STU \*SortbyName(STU \*head, int n); //函数9：按姓名的字典顺序排出成绩表（朱伯翰）

STU \*Create(int n, int m); //函数10：创建链表并录入信息（黄继升）

void SearchbyNum(STU \*head, int n, int m); //函数11：按学号查询学生排名及其考试成绩（黄继升）

void SearchbyName(STU \*head, int n, int m); //函数12：按姓名查询学生排名及其考试成绩（黄继升）

void WritetoFile(STU \*head, int n, int m); //函数13：将每个学生的纪录信息写入文件（黄继升）

STU \*ReadfromFile(STU \*head, int \*n, int \*m); //函数14：从文件中读出每个学生的纪录信息并显示（黄继升）

int DeleteFromFile(STU \*head, int \*n, int \*m); //函数15: 物理删除(删除在文件中的数据)（黄继升）

STU \*getAllMessageFromFile(STU \*head, int count0[], int count1[], int \*count2); //函数16：获取文件中所有的学生成绩数据，并组成单链表（黄继升）

STU \*ModifyScore(STU \*head); //函数17 用于修改学生某个科目的成绩信息（黄继升）

STU \*Creat1(int n, int m); //函数18 创建临时学生链表（黄继升）

int main()

{

int n, m;

int i;

STU \*head; //定义头节点

//head = (STU \*)malloc(LEN);

head = NULL;

while (true)

{

i = Menu(); //初始化菜单界面

if (i == 1) //菜单1

{

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t请输入学生数量(n<30):\n"); //输入学生数

printf("\t\t\t");

scanf("%d", &n);

printf("\t\t\t请输入考试科目数量(m<=6):\n");

printf("\t\t\t");

scanf("%d", &m);

}

switch (i)

{

case 1:

printf("\t\t\t请依次输入学号、姓名和成绩:\n");

head = Create(n, m);

system("cls"); //清屏

break;

case 2:

system("cls"); //清屏

AverSumofEveryStudent(head);

head = NULL;

break;

case 3:

system("cls"); //清屏

AverSumofEveryCourse(head);

head = NULL;

break;

case 4:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t总分按大到小排名如下:\n\n");

head = SortbyScore(head, n);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

head = NULL;

break;

case 5:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t总分按从小到大排名如下:\n\n");

head = SortbyScore1(head, n);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

head = NULL;

break;

case 6:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t按照学号从小到大排序如下:\n\n");

head = SortbyNum(head);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

head = NULL;

break;

case 7:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t按照学生名字的字典顺序排序:\n");

head = SortbyName(head, n);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

head = NULL;

break;

case 8:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t请输入所查询成绩的学号: ");

SearchbyNum(head, n, m);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

head = NULL;

break;

case 9:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t请输入所要查询成绩的学生姓名: ");

SearchbyName(head, n, m);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

head = NULL;

break;

case 10:

system("cls"); //清屏

StatisticAnalysis(head, n, m);

break;

case 11:

system("cls"); //清屏

printf("\n\n\n");

Print(head, n, m);

break;

case 12:

system("cls"); //清屏

WritetoFile(head, n, m);//持久化到文件中

head = NULL;//变野指针。清除临时区数据，避免二次保存重复数据

break;

case 13:

system("cls"); //清屏

printf("\n");

head = ReadfromFile(head, &n, &m);

head = NULL;

break;

case 14:

system("cls");

break;

case 15:

system("cls");

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

DeleteFromFile(head, &n, &m);

head = NULL;

break;

case 16:

system("cls");

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

head = ModifyScore(head);

head = NULL;

break;

case 0:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t已退出系统!\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

return 0;

default:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t输入错误!\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

return 0;

}

// 创建菜单

int Menu(void)

{

int i;

system("title 课程设计: 学生成绩管理系统 By 16041321黄继升 and 朱伯翰");

printf("\n\n");

printf("\t\t\t 学生成绩管理系统\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\* 系统功能菜单项: \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 1. 录入新的学生成绩记录(临时区) √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 2. 计算并输出每门课程的总分和平均分 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 3. 计算并输出每位学生的总分和平均分 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 4. 按学生总分从高到低排序 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 5. 按学生总分从低到高排序 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 6. 按学生学号从小到大排序 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 7. 按学生姓名的字典顺序排序 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 8. 按学生学号查询排名及其考试成绩 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 9. 按学生姓名查询排名及其考试成绩 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 10. 对每门课程的成绩情况进行分段统计 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 11. 从临时区读出还未保存的学生成绩信息 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 12. 将临时区数据保存到 student.txt 文件中 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 13. 从 student.txt 文件中导出每个学生的成绩信息 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 14. 逻辑删除学生成绩信息(要求具有恢复功能) \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 15. 物理删除学生成绩信息 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 16. 修改学生成绩信息 √ \*\n");

printf("\t\t\t\* \*\n");

printf("\t\t\t\* 0. 退出系统 \*\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n");

printf("\n");

printf("\n");

printf("\n");

printf("\t\t\t 请输入您的选择:");

scanf("%d", &i);

return i;

}

// 创建学生链表

// int n : 学生人数

// int m : 考试科目数

STU \*Create(int n, int m) {

STU \*head;

STU \*p1, \*p2;

int i, j;

for (i = 1; i < n + 1; i++)

{

p1 = (STU \*)malloc(LEN);

printf("\t\t\t");

scanf("%ld", &p1->num);//输入学号

scanf("%s", p1->name);//输入姓名

for (j = 0; j < m; j++)

{

scanf("%f", &p1->score[j]);

}

p1->next = NULL;

//如果是头结点

if (i == 1)

{

head = p2 = p1;

}

else

{

p2->next = p1; //临时结点

p2 = p1; //

}

}

//p1 = head;

//for(i = 0; i < n; i++){

// for(j = 0; j < m; j++) {

// printf("\t\t\t%f\n", p1->score[j]);

// }

// p1 = p1->next;

//}

return head;

}

STU \*Creat1(int n, int m)

{

STU \*head;

STU \*p1, \*p2;

int i, j;

head = NULL;

for (i = 0; i < n; i++)

{

p1 = (STU \*)malloc(LEN);

p1->next = NULL;

if (i == 0)

{

head = p2 = p1;

}

else

{

p2->next = p1;

p2 = p1;

}

}

p1 = head;

while(p1 != NULL)

{

for (i = 0; i < COURSE\_NUM; i++)

{

p1->score[i] = -1.0f;

}

p1 = p1->next;

}

return head;

}

void AverSumofEveryStudent(STU \*head)

{

int flag = 0;

int count = 0;

int count1[1000],count2[1000];

float score[COURSE\_NUM];

int num[COURSE\_NUM];

STU \*p;

int i, j, k = 0;

head = getAllMessageFromFile(head, count1, count2, &count);

p = head;

for(i = 0; i < COURSE\_NUM; i++)

{

score[i] = 0.0;

num[i] = 0;

}

//这里也没问题

//printf("count1[0] = %d, count2[0] = %d", count1[0], count2[0]); // 2 1

if (head != NULL)

{

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

for (i = 0; i < count; i++) //组数 1

{

for (j = 0; j < count1[i]; j++) //每组人数 2

{

for (k = 0; k < count2[i]; k++) //每个人的分数 1

{

score[k] = score[k] + p->score[k];

num[k]++;

}

p = p->next;

}

}

for(i = 0; i < COURSE\_NUM; i++)

{

if(num[i] > 0)

printf("\t\t\t科目 %d: sum=%.0f , aver=%.0f\n", i + 1, score[i], score[i] / num[i]);

}

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

void AverSumofEveryCourse(STU \*head)

{

int count = 0;

int count1[1000],count2[1000];

STU \*p;

head = getAllMessageFromFile(head, count1, count2, &count);

p = head;

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

while(p != NULL) {

printf("\t\t\t学号:%ld 姓名:%8s 总分=%3.0f 平均分=%3.0f\n", p->num, p->name, p->sum, p->aver);

p = p->next;

}

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

STU \*SortbyScore(STU \*head, int n)

{

int count = 0;

int flag = 0;

int rank = 1;

int count1[1000],count2[1000];

STU \*cur, \*tail, \*p;

float tempSum;

long tempNum;

char tempName[1024];

head = getAllMessageFromFile(head, count1, count2, &count);

tail = head;

cur = head;

p = head;

//tail指向为空

while (tail != NULL) {

tail = tail->next;

}

//外循环,循环次数为：链表长度-1次

while (p->next != tail) {

flag = 0;

cur = head;

while (cur->next != tail) {

if(cur->sum < cur->next->sum) {

//交换

tempSum = cur->sum;

cur->sum = cur->next->sum;

cur->next->sum = tempSum;

//C语言内部，数组的首地址是不可以被更改的。所以不能用赋值的方式去交换

strcpy(tempName, cur->name);

strcpy(cur->name, cur->next->name);

strcpy(cur->next->name, tempName);

tempNum = cur->num;

cur->num = cur->next->num;

cur->next->num = tempNum;

flag = 1;

}

cur = cur->next;

}

//尾指针向前移动

tail = cur;

}

//链表有序

while(p != NULL)

{

printf("\t\t\t学号:%10ld 姓名:%8s 总分=%4.0f 排名:%3d\n", p->num, p->name, p->sum, rank++);

p = p->next;

}

return head;

}

STU \*SortbyScore1(STU \*head, int n)

{

int count = 0;

int flag = 0;

int rank = 1;

int count1[1000],count2[1000];

STU \*cur, \*tail, \*p;

float tempSum;

long tempNum;

char tempName[1024];

head = getAllMessageFromFile(head, count1, count2, &count);

tail = head;

cur = head;

p = head;

//tail指向为空

while (tail != NULL) {

tail = tail->next;

}

//外循环,循环次数为：链表长度-1次

while (p->next != tail) {

flag = 0;

cur = head;

while (cur->next != tail) {

if(cur->sum > cur->next->sum) {

//交换

tempSum = cur->sum;

cur->sum = cur->next->sum;

cur->next->sum = tempSum;

//C语言内部，数组的首地址是不可以被更改的。所以不能用赋值的方式去交换

strcpy(tempName, cur->name);

strcpy(cur->name, cur->next->name);

strcpy(cur->next->name, tempName);

tempNum = cur->num;

cur->num = cur->next->num;

cur->next->num = tempNum;

flag = 1;

}

cur = cur->next;

}

//尾指针向前移动

tail = cur;

}

//链表有序

while(p != NULL)

{

printf("\t\t\t学号:%10ld 姓名:%8s 总分=%4.0f 排名:%3d\n", p->num, p->name, p->sum, rank++);

p = p->next;

}

return head;

}

STU \*SortbyNum(STU \*head)

{

int count = 0;

int flag = 0;

int rank = 1;

int count1[1000],count2[1000];

STU \*cur, \*tail, \*p;

float tempSum;

long tempNum;

char tempName[1024];

head = getAllMessageFromFile(head, count1, count2, &count);

tail = head;

cur = head;

p = head;

//tail指向为空

while (tail != NULL) {

tail = tail->next;

}

//外循环,循环次数为：链表长度-1次

while (p->next != tail) {

flag = 0;

cur = head;

while (cur->next != tail) {

if(cur->num > cur->next->num) {

//交换

tempSum = cur->sum;

cur->sum = cur->next->sum;

cur->next->sum = tempSum;

strcpy(tempName, cur->name);

strcpy(cur->name, cur->next->name);

strcpy(cur->next->name, tempName);

tempNum = cur->num;

cur->num = cur->next->num;

cur->next->num = tempNum;

flag = 1;

}

cur = cur->next;

}

//尾指针向前移动

tail = cur;

}

//链表有序

while(p != NULL)

{

printf("\t\t\t学号:%10ld 姓名:%8s 总分=%4.0f 序号:%3d\n", p->num, p->name, p->sum, rank++);

p = p->next;

}

return head;

}

STU \*SortbyName(STU \*head, int n)

{

int count = 0;

int flag = 0;

int rank = 1;

int count1[1000],count2[1000];

STU \*cur, \*tail, \*p;

float tempSum;

long tempNum;

char tempName[1024];

head = getAllMessageFromFile(head, count1, count2, &count);

tail = head;

cur = head;

p = head;

//tail指向为空

while (tail != NULL) {

tail = tail->next;

}

//外循环,循环次数为：链表长度-1次

while (p->next != tail) {

flag = 0;

cur = head;

while (cur->next != tail) {

if (strcmp(cur->name, cur->next->name) > 0) {

//交换

tempSum = cur->sum;

cur->sum = cur->next->sum;

cur->next->sum = tempSum;

strcpy(tempName, cur->name);

strcpy(cur->name, cur->next->name);

strcpy(cur->next->name, tempName);

tempNum = cur->num;

cur->num = cur->next->num;

cur->next->num = tempNum;

flag = 1;

}

cur = cur->next;

}

//尾指针向前移动

tail = cur;

}

//链表有序

while(p != NULL)

{

printf("\t\t\t学号:%10ld 姓名:%8s 总分=%4.0f 序号:%3d\n", p->num, p->name, p->sum, rank++);

p = p->next;

}

return head;

}

void Print(STU \*head, int n, int m)

{

STU \*p;

int i;

p = head;

if (head != NULL) //只要不是空链表，就输出链表中所有节点

{

do

{

printf("\t\t\t%ld\t%s\t", p->num, p->name);

for (i = 0; i<m; i++)

{

printf("%6.0f", p->score[i]);

}

printf("%6.0f%6.0f\n", p->sum, p->sum / m);//

p = p->next; //移到下一个节点

} while (p != NULL);

}

}

void SearchbyNum(STU \*head, int n, int m)

{

int count = 0;

int flag2 = 1;

int count1[1000],count2[1000];

STU \*p;

long num;

scanf("%ld", &num);

head = getAllMessageFromFile(head, count1, count2, &count);

p = head;

if (head != NULL) {

do {

if (p->num == num) {

printf("\t\t\t\n\n");

printf("\t\t\t学号: %ld\t姓名: %s\t", p->num, p->name);

for (int i = 0; i < COURSE\_NUM; i++) {

if (p->score[i] != -1.0f) {

printf("科目%d: %3.0f\t",i+1, p->score[i]);

}

}

printf("总分:%3.0f\t平均分:%3.0f\n", p->sum, p->aver);

flag2 = 0;

}

p = p->next;

} while(p != NULL);

}

if (flag2 != 0)

{

printf("\n\n");

printf("\t\t\t抱歉, 没有找到相关记录!\n");

}

}

void SearchbyName(STU \*head, int n, int m)

{

int count = 0;

int flag2 = 1;

int count1[1000],count2[1000];

STU \*p;

char name[MAX\_LEN];

scanf("%s", name);

head = getAllMessageFromFile(head, count1, count2, &count);

p = head;

if (head != NULL) {

do {

if (strcmp(name, p->name) == 0) {

printf("\t\t\t\n\n");

printf("\t\t\t学号: %ld\t姓名: %s\t", p->num, p->name);

for (int i = 0; i < COURSE\_NUM; i++) {

if (p->score[i] != -1.0f) {

printf("科目%d: %3.0f\t",i+1, p->score[i]);

}

}

printf("总分:%3.0f\t平均分:%3.0f\n", p->sum, p->aver);

flag2 = 0;

}

p = p->next;

} while(p != NULL);

}

if (flag2 != 0)

{

printf("\n\n");

printf("\t\t\t抱歉, 没有找到相关记录!\n");

}

}

void StatisticAnalysis(STU \*head, int n, int m)

{

int flag;

int count = 0;

int count1[1000],count2[1000];

int a[6];

STU \*p;

head = getAllMessageFromFile(head, count1, count2, &count);

p = head;

int i, j;

for (i = 0; i<6; i++)

{

p = head;

for (j = 0; j<6; j++) // 初始化

{

a[j] = 0;

}

flag = 0;

do

{

if (p->score[i] != -1.0f)

{

if (p->score[i] < 60)

{

a[0]++;

}

else if (p->score[i]<70)

{

a[1]++;

}

else if (p->score[i]<80)

{

a[2]++;

}

else if (p->score[i]<90)

{

a[3]++;

}

else if (p->score[i]<100)

{

a[4]++;

}

else

{

a[5]++;

}

flag = 1;

p = p->next;

}

else

{

p = p->next;

}

}

while (p != NULL);

if(flag == 1)

{

count = 0;

for (int t = 0; t < 6; t++)

count = count + a[t];

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\* 科目%d 成绩情况如下: \n\n", i + 1);

printf("\t\t\t\* <60\t%d\t%.2f%%\n", a[0], 100 \* (float) a[0]/count);

printf("\t\t\t\* %d-%d\t%d\t%.2f%%\n", 60, 69, a[1], (float) 100 \* a[1] / count);

printf("\t\t\t\* %d-%d\t%d\t%.2f%%\n", 70, 79, a[2], (float) 100 \* a[2] / count);

printf("\t\t\t\* %d-%d\t%d\t%.2f%%\n", 80, 89, a[3], (float) 100 \* a[3] / count);

printf("\t\t\t\* %d-%d\t%d\t%.2f%%\n", 90, 99, a[4], (float) 100 \* a[4] / count);

printf("\t\t\t\* %d\t%d\t%.2f%%\n", 100, a[5], 100 \* (float) a[5] / count);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}else {

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\* 抱歉, 该科目%d 暂无成绩统计: \n\n", i + 1);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

}

//STU \*head 学生链表头指针

//int n: 学生人数

//int m: 考试科目人数

void WritetoFile(STU \*head, int n, int m)

{

float sum;

STU \*p,\*p1;

p = head;

p1 = head;

sum = 0.0f;

//for(int i = 0; i < n; i++){

// for(int j = 0; j < m; j++) {

// printf("\t\t\t%f\n", p1->score[j]);

// }

// p1 = p1->next;

//}

FILE \*fp; //文件指针

int i, j;

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

//if ((fp = fopen("student.txt", "w")) == NULL)//不存在则以指定文件名创建;已存在则内容全部清空

if ((fp = fopen("student.txt", "a")) == NULL)//不存在则创建，存在则追加

{

printf("\t\t\t抱歉, 无法创建/打开文件 student.txt\n");

return;

}

//printf("\t\t\tdebug1");

if (p == NULL)

{

printf("\t\t\t临时区没有数据 或 已成功保存到文件中!\n");

return;

}

//1[tab]2

fprintf(fp, "%d\t%d\n", n, m);

//printf("\t\t\tdebug2\n");

for (i = 0; i<n; i++)

{

fprintf(fp, "%12ld%12s", p->num, p->name);//一个汉字占2个字符

//printf("%12ld%12s", p->num, p->name);

for (j = 0; j < m; j++)

{

sum = sum + p->score[j];

fprintf(fp, "%12.0f", p->score[j]);//输出宽度12，小数点后保留0位。即有小数的也默认取整数部分

//printf("%12.0f", p->score[j]);

}

//fprintf(fp, "%12.0f%12.0f\n", p->sum, p->sum / m);

fprintf(fp, "%12.0f%12.0f\n", sum, sum / m);

//printf("%12.0f%12.0f\n", p->sum, p->sum / m);

sum = 0.0f;

p = p->next;

}

printf("\t\t\t保存文件成功！ \n");

fclose(fp);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

//这里用了直接生成了一个新的链表，确保可以直接读取存好的数据

STU \*ReadfromFile(STU \*head, int \*n, int \*m)

{

STU \*p;

FILE \*fp;

int i, j;

if ((fp = fopen("student.txt", "r")) == NULL)

{

printf("\t\t\t不存在 student.txt!\n");

return NULL;

}

if (fgetc(fp) == EOF) {

printf("\t\t\tstudent.txt 没有数据!\n");

return NULL;

}

fclose(fp);

fp = fopen("student.txt", "r");

printf("\n\t\t\t导出成功!\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

while(!feof(fp))

{

fscanf(fp, "%d\t%d\n", n, m);//读出学生人数和考试科目数·

head = Creat1(\*n, \*m); //创建了一个空链表，并且赋给head

p = head;

for (i = 0; i<\*n; i++)

{

fscanf(fp, "%12ld", &p->num);

fscanf(fp, "%12s", &p->name);

for (j = 0; j<\*m; j++)

{

fscanf(fp, "%12f", &p->score[j]);

}

fscanf(fp, "%12f%12f\n", &p->sum, &p->aver);

p = p->next;

}

i = \*n;

j = \*m;

//fclose(fp);

Print(head, i, j);

//在这里需要将释放链表节省空间,时间太紧张了，所以暂时就直接赋予野指针了。等之后有时间再写一个函数到这里进行优化

//debug:

//if(feof(fp)) {

// printf("\t\t\t已经到文件末尾\n");

//}

}

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

fclose(fp);

return head;

}

//物理删除(替代法,方法比较繁琐)

int DeleteFromFile(STU \*head, int \*n, int \*m) {

int count0[1000], count1[1000];//用于保存学生人数和课程数,支持1000次记录存取

int count2;

STU \*head1;//head1

STU \*p,\*p1,\*p2;

FILE \*fp;

int i,j,k;

long num;

if ((fp = fopen("student.txt", "r")) == NULL) {

printf("\t\t\t不存在文件 student.txt!\n");

return 0;

}

if (fgetc(fp) == EOF) {

printf("\t\t\tstudent.txt 没有数据!\n");

return 0;

}

fclose(fp);

fp = fopen("student.txt", "r");

printf("\t\t\t请输入待删除记录的学号:");

scanf("%ld", &num);

//先把文件里的所有数据全部读到内存中，并组成一个学生链表，总共有count2次提交记录

count2 = 0;

fscanf(fp, "%d\t%d\n", n, m);

count0[count2] = \*n;//count0[0] = 1

count1[count2] = \*m;//count1[0] = 1

count2++;//1

head1 = head = Creat1(\*n, \*m); //创建一个空链表，并且赋给head

p = head;

for(i = 0; i < \*n; i++)

{

fscanf(fp, "%12ld", &p->num);//16041321

fscanf(fp, "%12s", &p->name);//hjs

for (j = 0; j<\*m; j++)

{

fscanf(fp, "%12f", &p->score[j]);//100

}

fscanf(fp, "%12f%12f\n", &p->sum, &p->aver);//100 100

if(p->num == num) {

count0[count2 - 1]--;//count0[0] = 0

}

p2 = p;//保存下当前节点

p = p->next;

}

while(!feof(fp))

{

fscanf(fp, "%d\t%d\n", n, m);// 1 1

count0[count2] = \*n;// 1

count1[count2] = \*m;// 1

++count2;//2

head = Creat1(\*n, \*m); //创建一个空链表，并且赋给head

p2->next = head;

p = head;

for(i = 0; i < \*n; i++)

{

fscanf(fp, "%12ld", &p->num);//16041333

fscanf(fp, "%12s", &p->name);//hjj

for (j = 0; j<\*m; j++)

{

fscanf(fp, "%12f", &p->score[j]);//89

}

fscanf(fp, "%12f%12f\n", &p->sum, &p->aver);//89 89

if(p->num == num){

count0[count2-1]--;//count0[1] = 1

}

p2 = p;

p = p->next;

}

}

fclose(fp);

p = p1 = head1;

if (head1 != NULL)

{

if(p->num == num)//如果删除的是头结点

{

head1 = p->next;

}

else

{

while(p != NULL && p->num != num) //注释这里要先判空，顺序不能颠倒...在这里debug了很久

{

p1 = p;//保存下当前节点

p = p->next;

}

if(p == NULL)

{

printf("\t\t\t没有查找到相关记录 !\n");

return 0;

}

p1->next = p->next;

}

free(p);//删除节点

}

fp = fopen("student.txt", "w");//以清空的方式打开文件

p = head1;

for (i = 0; i < count2; i++)

{

fprintf(fp, "%d\t%d\n", count0[i], count1[i]);

for (j = 0; j < count0[i]; j++)

{

fprintf(fp, "%12ld%12s", p->num, p->name);//一个汉字占2个字符

for (k = 0; k < count1[i]; k++)

{

fprintf(fp, "%12.0f", p->score[k]);//输出宽度12，小数点后保留0位。即有小数的也默认取整数部分

}

fprintf(fp, "%12.0f%12.0f\n", p->sum, p->aver);

p = p->next;

}

}

printf("\n\t\t\t删除成功！ \n");

fclose(fp);

return 1;

}

STU\* getAllMessageFromFile(STU \*head, int count0[], int count1[], int \*count2) {

int n, m;

STU \*head1;

STU \*d,\*p1,\*p2;

FILE \*fp;

int i,j,k;

int num;

if ((fp = fopen("student.txt", "r")) == NULL) {

printf("\t\t\t不存在文件 student.txt!\n");

return NULL;

}

if (fgetc(fp) == EOF) {

printf("\t\t\tstudent.txt 没有数据!\n");

return NULL;

}

fclose(fp);

//先把文件里的所有数据全部读到内存中，并组成一个学生链表，总共有count2次提交记录

fp = fopen("student.txt", "r");

(\*count2) = 0;

fscanf(fp, "%d\t%d\n", &n, &m);

count0[(\*count2)] = n;//count0[0] = 2

count1[(\*count2)] = m;//count1[0] = 1

(\*count2)++;//1

head1 = head = Creat1(n, m); //创建一个空链表，并且赋给head

d = head;

for(i = 0; i < n; i++)

{

fscanf(fp, "%12ld", &d->num);//16041333 16051213

fscanf(fp, "%12s", &d->name);//zzm cl

for (j = 0; j<m; j++)

{

fscanf(fp, "%12f", &d->score[j]);//96 16

}

fscanf(fp, "%12f%12f\n", &d->sum, &d->aver);//100 100

p2 = d;//保存下当前节点

d = d->next;

}

while(!feof(fp))

{

fscanf(fp, "%d\t%d\n", &n, &m);// 1 1

count0[(\*count2)] = n;// 1

count1[(\*count2)] = m;// 1

++(\*count2);//2

head = Creat1(n, m); //创建一个空链表，并且赋给head

p2->next = head;

d = head;

for(i = 0; i < n; i++)

{

fscanf(fp, "%12ld", &d->num);//16041333

fscanf(fp, "%12s", &d->name);//hjj

for (j = 0; j<m; j++)

{

fscanf(fp, "%12f", &d->score[j]);//89

}

fscanf(fp, "%12f%12f\n", &d->sum, &d->aver);//89 89

p2 = d;

d = d->next;

}

}

fclose(fp);

return head1;

}

STU \*ModifyScore(STU \*head)

{

int count = 0;

int flag2 = 1;

int choose;

int count1[1000],count2[1000];

STU \*p;

long num;

char newName[MAX\_LEN];

FILE \*fp;

int i, j, k;

int count3 = 0;

float newSum = 0.0f;

printf("\t\t\t请输入要修改的成绩记录的学生学号: ");

scanf("%ld", &num);

head = getAllMessageFromFile(head, count1, count2, &count);//从student.txt中获取学生链表

p = head;

if (head != NULL)

{

do

{

if (p->num == num)

{

printf("\n\n");

printf("\t\t\t该成绩记录的详细信息如下:\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t学号: %ld\t姓名: %s\t", p->num, p->name);

for (int i = 0; i < COURSE\_NUM; i++)

{

if (p->score[i] != -1.0f)

{

count3++;

printf("科目%d: %3.0f\t",i+1, p->score[i]);

}

}

printf("总分:%3.0f\t平均分:%3.0f\n", p->sum, p->aver);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

flag2 = 0;

break;

}

p = p->next;

}

while(p != NULL);

}

if (flag2 != 0)

{

printf("\n\n");

printf("\t\t\t抱歉, 没有找到相关记录!\n");

}

else

{

printf("\n\n\t\t\t待修改项: 1.姓名 2.学号 3.科目 4.退出\n\n");

printf("\t\t\t请选择修改项的位置序号: ");

scanf("%d", &choose);

printf("\n\n");

fp = fopen("student.txt", "w");//以清空的方式打开文件

switch (choose)

{

case 1:

printf("\t\t\t请输入新的学生姓名: ");

scanf("%s", p->name);

printf("\n");

p = head;

for (i = 0; i < count; i++)

{

fprintf(fp, "%d\t%d\n", count1[i], count2[i]);

for (j = 0; j < count1[i]; j++)

{

fprintf(fp, "%12ld%12s", p->num, p->name);//一个汉字占2个字符

for (k = 0; k < count2[i]; k++)

{

fprintf(fp, "%12.0f", p->score[k]);//输出宽度12，小数点后保留0位。即有小数的也默认取整数部分

}

fprintf(fp, "%12.0f%12.0f\n", p->sum, p->aver);

p = p->next;

}

}

system("cls");

printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t修改成功！\n");

break;

case 2:

printf("\t\t\t请输入新的学生学号: ");

scanf("%ld", &p->num);

printf("\n");

p = head;

for (i = 0; i < count; i++)

{

fprintf(fp, "%d\t%d\n", count1[i], count2[i]);

for (j = 0; j < count1[i]; j++)

{

fprintf(fp, "%12ld%12s", p->num, p->name);//一个汉字占2个字符

for (k = 0; k < count2[i]; k++)

{

fprintf(fp, "%12.0f", p->score[k]);//输出宽度12，小数点后保留0位。即有小数的也默认取整数部分

}

fprintf(fp, "%12.0f%12.0f\n", p->sum, p->aver);

p = p->next;

}

}

system("cls");

printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t修改成功！\n");

break;

case 3:

printf("\t\t\t该学生共有 %d 门科目成绩, 请选择待修改的科目序号: ", count3);

scanf("%ld", &choose);

while(choose <= 0 || choose > count3)

{

printf("\n\t\t\t输入错误, 请重新输入: ");

scanf("%ld", &choose);

}

printf("\n\n\t\t\t请输入新的成绩: ");

scanf("%f", &p->score[choose-1]);

printf("\n");

p = head;

for (i = 0; i < count; i++)

{

fprintf(fp, "%d\t%d\n", count1[i], count2[i]);

for (j = 0; j < count1[i]; j++)

{

fprintf(fp, "%12ld%12s", p->num, p->name);

for (k = 0; k < count2[i]; k++)

{

newSum = newSum + p->score[k];

fprintf(fp, "%12.0f", p->score[k]);//输出宽度12，小数点后保留0位。即有小数的也默认取整数部分

}

fprintf(fp, "%12.0f%12.0f\n", newSum, newSum/count2[i]);

p = p->next;

}

}

system("cls");

printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t修改成功！\n");

break;

case 4:

break;

default:

printf("\t\t\t输入错误!\n");

}

}

fclose(fp);

return head;

}