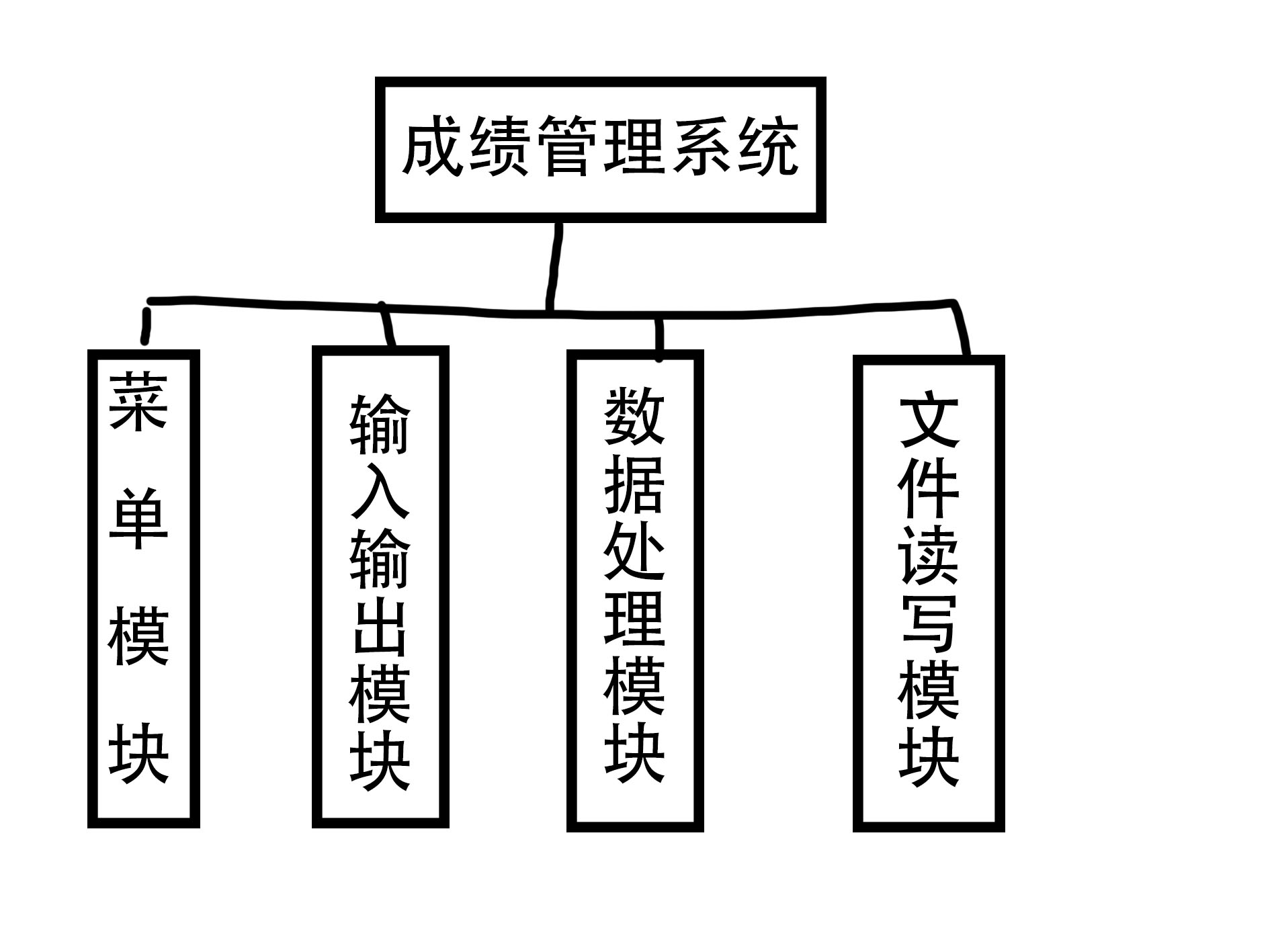
|  |
| --- |
| **课程设计——小型成绩管理系统**  姓名： \*\*\* 学号： \*\*\*\*\*\*\*\*\* 任课教师： \*\*\*\*\*\*\*\*\*\*  一、程序功能  编程实现一个基础的成绩管理系统。  二、设计目的  通过本程序综合掌握数组、函数、结构体类型、指针等知识的综合使用。  三、功能要求（可扩充或整合功能）  (1)实现简单的菜单设计，如下所示：  1 添加学生信息  2 显示学生信息  3 按学号查找学生信息  4 插入一个学生信息  5 删除一个学生信息  6 按总分降序输出  7 退出   1. 每个学生的信息至少包括姓名、学号、3门课程的成绩等信息。   (3) 添加学生信息：输入新的学生信息（按学号升序）以后，系统将存储相关信息。  (4) 显示学生信息：系统将按要求显示所有的学生信息。要求输出清晰。  (5) 按学号查找学生信息：按学号查找。  (6) 插入一个学生信息：根据学号顺序进行插入。  (7) 删除一个学生信息：根据学号删除。删除前显示该条记录，并提交用户确认。  (8) 按总分降序输出：计算总分，按总分降序。  四、要求采用模块化设计，独立的功能（如添加、显示、查询、插入、删除等）应在各个自定义函数中实现。  五、撰写课程设计报告，内容包括：功能结构图、函数列表及功能简介，完整的源程序（包含必要的注释）、程序运行结果、小结等。  六、报告以word文档形式提交，以学号+姓名作为文件名。 |

## 一、功能结构图



## 二、函数列表及功能简介

AddStd(struct STU \*student):

功能：添加学生信息，包括学号、姓名和三门课的分数。

简介：该函数接收一个指向结构体STU的指针，通过标准输入获取学生信息，并将其写入到score.txt文件中。

QueryById(struct STU \*student):

功能：按学号查询学生详细信息。

简介：该函数接收一个指向结构体STU的指针，输入学号后，从score.txt文件中查找对应学生的信息并输出。

ListAll(struct STU \*student):

功能：按学号排序输出所有学生信息。

简介：该函数接收一个指向结构体STU的指针，读取score.txt文件中的所有学生信息，按学号排序后输出。

AvgScore(struct STU \*student):

功能：输出平均分或单科的平均分。

简介：该函数接收一个指向结构体STU的指针，根据用户选择的科目，计算并输出平均分。

MaxScore(struct STU \*student):

功能：输出平均分或单科最高分的学生信息。

简介：该函数接收一个指向结构体STU的指针，根据用户选择的科目，找出最高分的学生信息并输出。

removeID(const char \*source):

功能：按学号删除学生信息。

简介：该函数接收一个源文件路径，删除score.txt文件中指定学号的学生信息，并通过重命名操作保持文件的完整性。

menu(struct STU \*student):

功能：显示功能菜单并处理用户的选择。

简介：该函数提供一个循环，用于显示各种操作选项，并根据用户的选择调用相应的函数。

## 三、源程序（有注释）

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct STU {

char sno[20];

char name[20];

double score1;

double score2;

double score3;

double score;

};

// 函数声明

void AddStd(struct STU \*student);

void QueryById(struct STU \*student);

void ListAll(struct STU \*student);

void AvgScore(struct STU \*student);

void MaxScore(struct STU \*student);

void menu(struct STU \*student);

int main() {

struct STU std;

menu(&std); // 调用菜单函数启动程序

return 0;

}

// 1、输入学生信息函数AddStd实现添加学生相关信息；

void AddStd(struct STU \*student) {

char op;

printf("请依次输入学生的学号、姓名（汉语拼音）和三门课分数（回车键隔开）：\n");

scanf("%s", student->sno);

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

scanf("%lf", &student->score1);

scanf("%lf", &student->score2);

scanf("%lf", &student->score3);

student->score = (student->score1+student->score2+student->score3)/3;

// 检查学号、姓名是否为空

if (strlen(student->sno) == 0 || strlen(student->name) == 0) {

printf("学号和姓名不能为空，请重新输入。\n");

return;

}

FILE \*info = fopen("score.txt", "a+");

if (info == NULL) {

printf("文件操作失败\n");

return;

}

// 写入学生信息到文件

fprintf(info, "%s\n%s\n%0.2f\n%0.2f\n%0.2f\n%0.2f\n", student->sno, student->name, student->score, student->score1, student->score2 ,student->score3);

fclose(info);

printf("是否继续添加学生信息，y/n\n");

getchar();

scanf("%c", &op);

if (op == 'y')

AddStd(student);

else

menu(student); // 假设这是返回到主菜单的函数

}

//2、按学号查询学生详细信息（包括成绩）QueryById;

void QueryById(struct STU \*student) {

char sno1[20], op;

char line[100]; // 修改了变量名以提高可读性

int flag = 0;

printf("请输入您要查询的学号：\n");

scanf("%s", sno1);

FILE \*info = fopen("score.txt", "r");

if (info == NULL) {

printf("读取文件失败\n");

return;

}

// 检查学号是否存在

while (fgets(line, sizeof(line), info)) {

int i = 0;

char sno\_chk[20];

sscanf(line, "%s", sno\_chk);

while (sno\_chk[i] != '\0' && sno\_chk[i] != ',') {

if (sno\_chk[i] == sno1[i]) {

flag = 1;

break;

}

i++;

}

if (flag) {

printf("学号为%s的学生的详细信息如下：\n%s\n\n", sno1, line);

break;

}

}

if (!flag)

printf("未查询到学号为%s的学生信息。\n\n", sno1);

fclose(info);

printf("是否继续查询学生信息，y/n\n");

getchar(); // 清除输入缓冲区

scanf("%c", &op);

if (op == 'y')

QueryById(student);

else

menu(student); // 假设这是返回到主菜单的函数

}

//3、成绩列表函数ListAll按学号排序输出学生信息；

void ListAll(struct STU \*student) {

struct STU stu[100]; // 定义一个足够大的数组来存储所有学生信息

int i, j, len = 0;

FILE \*info = fopen("score.txt", "r");

if (info == NULL) {

printf("读取文件失败\n");

return;

}

// 读取学生信息并存入数组

while (fscanf(info, "%s %s %lf %lf %lf %lf", stu[len].sno, stu[len].name, &stu[len].score, &stu[len].score1, &stu[len].score2, &stu[len].score3) != EOF) {

len++;

}

fclose(info);

// 按学号排序

for (i = 0; i < len - 1; i++) {

for (j = 0; j < len - i - 1; j++) {

if (strcmp(stu[j].sno, stu[j + 1].sno) > 0) {

struct STU temp = stu[j];

stu[j] = stu[j + 1];

stu[j + 1] = temp;

}

}

}

// 输出学生信息

for (i = 0; i < len; i++) {

printf("%s %s %0.2f %0.2f %0.2f %0.2f\n", stu[i].sno, stu[i].name,&stu[i].score, &stu[i].score1, &stu[i].score2, &stu[i].score3 != EOF);

}

printf("请按任意键继续...\n");

getchar(); // 等待用户按键，以便清屏

}

//4.

void ListScore(struct STU \*student) {

printf("请选择科目1-3，或输入0为平均分\n");

int nol;

scanf("%d", &nol);

struct STU stu[100]; // 定义一个足够大的数组来存储所有学生信息

int i, j, len = 0;

FILE \*info = fopen("score.txt", "r");

if (info == NULL) {

printf("读取文件失败\n");

return;

}

// 读取学生信息并存入数组

while (fscanf(info, "%s %s %lf %lf %lf %lf", stu[len].sno, stu[len].name, &stu[len].score, &stu[len].score1, &stu[len].score2, &stu[len].score3) != EOF) {

len++;

}

fclose(info);

switch (nol)

{

case 0:

for (i = 0; i < len - 1; i++) {

for (j = 0; j < len - i - 1; j++) {

if (stu[j].score < stu[j + 1].score) {

struct STU temp = stu[j];

stu[j] = stu[j + 1];

stu[j + 1] = temp;

}

}

}

break;

case 1:

for (i = 0; i < len - 1; i++) {

for (j = 0; j < len - i - 1; j++) {

if (stu[j].score1 < stu[j + 1].score1) {

struct STU temp = stu[j];

stu[j] = stu[j + 1];

stu[j + 1] = temp;

}

}

}

break;

case 2:

for (i = 0; i < len - 1; i++) {

for (j = 0; j < len - i - 1; j++) {

if (stu[j].score2 < stu[j + 1].score2) {

struct STU temp = stu[j];

stu[j] = stu[j + 1];

stu[j + 1] = temp;

}

}

}

break;

case 3:

for (i = 0; i < len - 1; i++) {

for (j = 0; j < len - i - 1; j++) {

if (stu[j].score3 < stu[j + 1].score3) {

struct STU temp = stu[j];

stu[j] = stu[j + 1];

stu[j + 1] = temp;

}

}

}

break;

default:

break;

}

// 输出学生信息

for (i = 0; i < len; i++) {

printf("%s %s %0.2f %0.2f %0.2f %0.2f\n", stu[i].sno, stu[i].name,&stu[i].score, &stu[i].score1, &stu[i].score2, &stu[i].score3 != EOF);

}

printf("请按任意键继续...\n");

getchar();

}

//4、输出平均分AvgScore；

void AvgScore(struct STU \*student) {

printf("请选择科目1-3，或输入0为平均分\n");

int nol;

scanf("%d", &nol);

struct STU stu[100]; // 定义一个足够大的数组来存储所有学生信息

int i, j;

double totalSum = 0;

int numStudents = 0;

FILE \*info = fopen("score.txt", "r");

if (info == NULL) {

printf("读取文件失败\n");

return;

}

// 读取学生信息并存入数组

while (fscanf(info, "%s %s %lf", stu[numStudents].sno, stu[numStudents].name, &stu[numStudents].score) != EOF) {

numStudents++;

}

fclose(info);

// 计算总分

switch (nol)

{

case 0:

for (i = 0; i < numStudents; i++) {

totalSum += stu[i].score;

}

break;

case 1:

for (i = 0; i < numStudents; i++) {

totalSum += stu[i].score1;

}

break;

case 2:

for (i = 0; i < numStudents; i++) {

totalSum += stu[i].score2;

}

break;

case 3:

for (i = 0; i < numStudents; i++) {

totalSum += stu[i].score3;

}

break;

default:

break;

}

// 计算平均分

float average = totalSum / numStudents;

// 输出平均分

printf("\n平均分数为：%.2f\n\n", average);

printf("请按任意键继续...\n");

getchar(); // 等待用户按键，以便清屏

}

//5、输出最高分MaxScore学生信息（需要注意的是最高分的学生可能不只一人）。

void MaxScore(struct STU \*student) {

printf("请选择科目1-3，或输入0为平均分\n");

int nol;

struct STU stu[100]; // 定义一个足够大的数组来存储所有学生信息

int i, j, maxScore = 0;

char maxID[20];

int numStudents = 0;

FILE \*info = fopen("score.txt", "r");

if (info == NULL) {

printf("读取文件失败\n");

return;

}

// 读取学生信息并存入数组

while (fscanf(info, "%s %s %lf %lf %lf %lf", stu[numStudents].sno, stu[numStudents].name, &stu[numStudents].score, &stu[numStudents].score1, &stu[numStudents].score2, &stu[numStudents].score3) != EOF) {

numStudents++;

}

fclose(info);

switch (nol)

{

case 0:

// 查找最高分

for (i = 0; i < numStudents; i++) {

if (stu[i].score > maxScore) {

maxScore = stu[i].score;

strcpy(maxID, stu[i].sno); // 假设学号是用来识别学生的唯一标识符

}

}

break;

case 1:

// 查找最高分

for (i = 0; i < numStudents; i++) {

if (stu[i].score1 > maxScore) {

maxScore = stu[i].score1;

strcpy(maxID, stu[i].sno); // 假设学号是用来识别学生的唯一标识符

}

}

break;

case 2:

// 查找最高分

for (i = 0; i < numStudents; i++) {

if (stu[i].score2 > maxScore) {

maxScore = stu[i].score2;

strcpy(maxID, stu[i].sno); // 假设学号是用来识别学生的唯一标识符

}

}

break;

case 3:

// 查找最高分

for (i = 0; i < numStudents; i++) {

if (stu[i].score2 > maxScore) {

maxScore = stu[i].score2;

strcpy(maxID, stu[i].sno); // 假设学号是用来识别学生的唯一标识符

}

}

break;

default:

break;

}

// 输出最高分和对应学生的基本信息

printf("获得最高成绩学生的基本信息如下：\n");

for (i = 0; i < numStudents; i++) {

if (strcmp(stu[i].sno, maxID) == 0) {

printf("学号：%s\n", stu[i].sno);

printf("姓名：%s\n", stu[i].name);

printf("平均分：%.2f 各科分数：%.2f %.2f %.2f\n", stu[i].score, stu[i].score1, stu[i].score2, stu[i].score3);

break;

}

}

printf("\n请按任意键继续...\n");

getchar(); // 等待用户按键，以便清屏

}

void removeID(const char \*source) {

char targetID[20];

printf("输入需要删除的ID\n");

getchar();

gets(targetID);

FILE \*info = fopen(source, "r+");

if (info == NULL) {

printf("读取文件失败\n");

return;

}

// 读取所有学生信息并存入数组

struct STU stu[100];

int numStudents = 0;

while (fscanf(info, "%s %s %lf %lf %lf %lf", stu[numStudents].sno, stu[numStudents].name, &stu[numStudents].score, &stu[numStudents].score1, &stu[numStudents].score2, &stu[numStudents].score3) != EOF) {

numStudents++;

}

fclose(info);

// 打开文件进行写操作

FILE \*tempFile = fopen("temp.txt", "w");

if (tempFile == NULL) {

printf("创建临时文件失败\n");

return;

}

// 遍历数组，将除了要删除的数据之外的所有数据写入临时文件

int i, j;

for (i = 0; i < numStudents; i++) {

if (strcmp(stu[i].sno, targetID) != 0) {

fprintf(tempFile, "%s\n%s\n%lf\n", stu[i].sno, stu[i].name, stu[i].score);

} else {

// 找到要删除的数据，跳过这一行

for (j = i; j < numStudents - 1; j++) {

stu[j] = stu[j + 1];

}

numStudents--; // 数组中的元素数量减一

i--; // 重新遍历这一行

}

}

// 关闭文件

fclose(tempFile);

fclose(info);

// 删除原文件，重命名临时文件

remove(source);

rename("temp.txt", source);

printf("学生信息已成功从 %s 文件中删除。\n", source);

}

//利用while循环实现功能界面输出，根据用户选择调用对应的功能选项。

void menu(struct STU \*student) {

int flag;

while (1) {

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*学生成绩管理系统\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("0、退出\n");

printf("1、添加学生信息\n");

printf("2、查询学生详细信息\n");

printf("3、按学号排序输出学生信息\n");

printf("4、按单科或平均分排序输出学生信息\n");

printf("5、输出平均分或单科的平均分\n");

printf("6、输出平均分或单科最高分的学生信息\n");

printf("7、按学号删除学生\n");

printf("8、删除文件\n");

printf("请输入您要进行的操作：\n");

scanf("%d", &flag);

switch (flag) {

case 0:

printf("退出系统。\n");

exit(0);

break;

case 1:

AddStd(student);

break;

case 2:

QueryById(student);

break;

case 3:

ListAll(student);

break;

case 4:

ListScore(student);

case 5:

AvgScore(student);

break;

case 6:

MaxScore(student);

break;

case 7:

removeID("score.txt");

case 8:

remove("source.txt");

break;

default:

printf("输入错误，请重新输入。\n");

getchar(); // Clear the input buffer

break;

}

}

}

## 四、程序运行结果

程序运行时，会首先显示一个功能菜单，用户可以根据自己的需求选择相应的功能选项进行操作。

添加学生信息：用户可以输入学生的学号、姓名和三门课的分数，然后程序将把这些信息存储到一个名为"score.txt"的文件中。

查询学生详细信息：用户需要输入要查询的学生学号，程序会根据学号在"score.txt"文件中查找对应的学生信息，并将其显示出来。

按学号排序输出学生信息：程序会读取"score.txt"文件中的所有学生信息，并按照学号进行排序，然后将排序后的信息输出显示。

按单科或平均分排序输出学生信息：用户可以选择按单科成绩或平均分进行排序，程序会按照用户的选择在"score.txt"文件中查找并输出排序后的学生信息。

输出平均分或单科的平均分：用户可以选择输出所有学生的平均分或单科的平均分，程序会根据用户的选择计算并输出相应的平均分。

输出平均分或单科最高分的学生信息：用户可以选择输出平均分最高的学生信息或单科最高分的学生信息，程序会根据用户的选择在"score.txt"文件中查找并输出相应的信息。

按学号删除学生信息：用户需要输入要删除的学生学号，程序会在"score.txt"文件中查找并删除对应的学生信息。

删除文件：用户可以选择删除"score.txt"文件，程序会删除该文件。

此外，程序还包含了一些辅助功能，如输入检查、文件操作错误处理等。

## 五、小结

该程序是一个以文件为基础的学生成绩管理系统的程序。它提供了添加学生信息、查询学生详细信息、按学号排序输出学生信息、按单科或平均分排序输出学生信息、输出平均分或单科最高分的学生信息、按学号删除学生信息以及删除文件等功能。同时，该程序还具备一定的检查输入输出的能力，且能够实时保存，防止意外打断。