目录

[一、课程设计目的 1](#_Toc136682485)

[1、设计目的概述 1](#_Toc136682486)

[二、总体设计思路 1](#_Toc136682487)

[1、总体介绍 1](#_Toc136682488)

[2、需求分析 2](#_Toc136682489)

[三、图示及流程图 3](#_Toc136682490)

[1、程序功能划分 3](#_Toc136682491)

[三、程序设计、运行结果 6](#_Toc136682492)

[1、程序设计代码 6](#_Toc136682493)

[2、运行结果截图 16](#_Toc136682494)

[四、心得体会 17](#_Toc136682495)

[五、参考文献 18](#_Toc136682496)

# 一、课程设计目的

## 1、设计目的概述

设计一个学生成绩管理程序的目的是为了有效地管理学生的成绩信息，提供方便快捷的查询和统计功能，以及支持教师、学生和学校管理者对学生成绩进行分析和评估。该程序旨在提供以下功能：

1.学生信息管理：记录学生的基本信息，包括姓名、学号、班级等。这样可以方便地查找和更新学生的成绩信息。

2.成绩录入和计算：允许教师输入学生的成绩信息，并自动计算总分、平均分和排名等统计数据。通过自动计算成绩，减少了手工计算的错误和工作量。

3.成绩查询和统计：提供学生和教师查询学生个人成绩和班级整体成绩的功能。学生可以方便地了解自己的成绩情况，教师可以快速获取班级的统计数据，如平均分和及格率等。

4.成绩分析和报告：支持生成成绩分析报告，帮助教师和学校管理者对学生成绩进行分析和评估。这些报告可以包括各科目的成绩分布、成绩趋势等信息，有助于制定教学改进计划和提供个性化辅导。

5.数据安全和权限控制：确保学生成绩数据的安全性和隐私性，只有授权人员才能访问和修改成绩信息。教师可以管理自己所教班级的成绩，而学校管理者可以管理全校的成绩数据。

通过这个学生成绩管理程序，学校可以更好地管理学生的成绩信息，提供及时准确的成绩数据，帮助教师和学校管理者进行教育决策和学生发展评估。

# 二、总体设计思路

## 1、总体介绍

学生成绩管理系统是一种基于计算机技术和数据库管理的软件系统，旨在帮助学校、教师和学生有效管理和维护学生的成绩信息。该系统提供了一系列功能，包括学生信息管理、成绩录入和计算、成绩查询和统计、成绩分析和报告等，以实现对学生成绩的全面管理。

学生成绩管理系统的总体目标是提供一个方便、高效、准确和安全的平台，以满足教育机构对学生成绩管理的需求。通过该系统，教师可以轻松录入学生的成绩，进行自动计算和分析，以及生成各种报告和统计数据。学生和家长可以通过系统查询和了解自己的成绩情况，以及与其他学生进行比较和评估。

以下是学生成绩管理系统的主要特点和功能：

1. 学生信息管理：系统提供学生信息的录入和管理功能，包括学生的基本信息、班级信息、联系方式等。这样可以方便地查询和更新学生的成绩信息，并支持学生信息的批量导入和导出。

2. 成绩录入和计算：教师可以通过系统录入学生的各科成绩，系统会自动计算总分、平均分和排名等统计数据。教师还可以设置权重和考试类型，以灵活地进行成绩计算。

3. 成绩查询和统计：学生、家长和教师可以通过系统查询学生的个人成绩和班级整体成绩。系统提供多种查询方式和筛选条件，方便用户根据需要进行成绩查询和统计。

4. 成绩分析和报告：系统支持生成各种成绩分析报告，如学科成绩分布图、成绩趋势图等。这些报告可以帮助教师和学校管理者对学生成绩进行深入分析，以制定教学改进计划和提供个性化辅导。

5. 数据安全和权限控制：系统采用安全的数据库管理技术，确保学生成绩数据的安全性和隐私性。系统还提供多级权限控制，确保只有授权人员才能访问和修改成绩信息。

学生成绩管理系统的实施可以帮助学校提高成绩管理的效率和准确性，为教师和学生提供更好的教学和学习支持。同时，系统还可以为学校管理者提供决策依据，进行学生发展评估和教育质量监控，促进学校的整体发展。

## 2、需求分析

学生成绩管理系统的需求分析是为了明确系统的功能和特性，以满足学校、教师和学生的实际需求。以下是学生成绩管理系统的需求分析：

1. 学生信息管理需求：

- 系统应该能够记录和管理学生的基本信息，包括姓名、学号、班级、联系方式等。

- 教师和管理员应该能够轻松添加、更新和删除学生信息。

- 系统应该支持批量导入和导出学生信息，以提高效率。

2. 成绩录入和计算需求：

- 教师应该能够方便地录入学生的各科成绩，包括考试成绩、作业成绩等。

- 系统应该能够自动计算学生的总分、平均分和排名等统计数据。

- 教师应该能够灵活设置权重和考试类型，以适应不同科目和评估方式。

3. 成绩查询和统计需求：

- 学生、家长和教师应该能够通过系统方便地查询学生的个人成绩。

- 教师和管理员应该能够查询和统计班级整体成绩，包括平均分、及格率等。

- 系统应该支持多种查询方式和筛选条件，以满足不同用户的需求。

4. 成绩分析和报告需求：

- 系统应该能够生成各科目的成绩分布图、趋势图等成绩分析报告。

- 教师和管理员应该能够根据需要生成个性化的成绩分析报告。

- 报告应该提供清晰的数据展示和图表，以便教师和管理员进行教学和评估分析。

5. 数据安全和权限控制需求：

- 系统应该具备高级的数据安全保护措施，确保学生成绩数据的安全性和隐私性。

- 系统应该支持多级权限控制，确保只有授权人员才能访问和修改成绩信息。

6. 用户友好性需求：

- 系统应该具备良好的用户界面和操作流程，使用户能够轻松上手和使用。

- 系统应该提供清晰的操作指导和帮助文档，以解决用户在使用过程中的问题。

7. 扩展性和灵活性需求：

- 系统应该具备良好的扩展性，能够根据学校需求进行定制和功能拓展。

- 系统应该能够适应不同规模和类型的学校，包括小学、中学和大学等。

通过满足上述需求，学生成绩管理系统能够提供高效、准确、安全的学生成绩管理服务，支持学校教育管理的决策和改进。

# 三、图示及流程图

## 1、程序功能划分

该代码的功能划分如下：

1. 定义常量和结构体：

- 定义了常量 `SWN` 表示课程数，`NAMELEN` 表示姓名最大字符数，`CODELEN` 表示学号最大字符数，`FNAMELEN` 表示文件名最大字符数，`BUFLEN` 表示缓冲区最大字符数。

- 定义了结构体 `record`，包含学生的姓名、学号、各门功课的成绩和总分。

- 定义了结构体 `node`，用于构造链表，包含和 `record` 相同的字段和一个指向下一个节点的指针。

2. 声明函数：

- `readrecord`：从文件中读取一个记录到结构体中。

- `writerecord`：将一个记录写入文件中。

- `displaystu`：显示一个学生的记录信息。

- `totalmark`：计算各门课程的总分。

- `liststu`：列出所有学生的记录。

- `makelist`：根据总分构造一个链表。

- `displaylist`：顺序显示链表中的所有记录。

- `retrievebyn`：按学生姓名查找学生的记录。

- `retrievebyc`：按学生学号查找学生的记录。

3. 主函数 `main`：

- 获取输入的学生成绩记录文件名。

- 如果文件不存在，询问是否创建新文件并获取要写入的记录数量，然后逐个输入学生的姓名、学号和各门功课的成绩，并将记录写入文件中。

- 关闭文件。

- 显示功能菜单，根据用户输入的命令执行相应的功能，包括计算平均分、计算总分、按姓名或学号查找记录、列出所有记录以及按总分排序并显示记录。

- 根据用户输入的命令执行相应的功能，直到用户选择退出。

以上是该代码的主要功能划分。它实现了统计学生考试分数的管理程序，能够根据需求计算平均分、总分，按姓名或学号查找记录，列出所有记录，以及按总分排序显示学生信息等功能。

在这个程序中，以下是一些其他函数的功能划分：

1. 函数 `totalmark`：

- 参数：接收学生成绩记录文件的文件名。

- 功能：计算各门课程的总分，并返回记录的数量。

2. 函数 `liststu`：

- 参数：接收学生成绩记录文件的文件名。

- 功能：列出所有学生的记录。

3. 函数 `makelist`：

- 参数：接收学生成绩记录文件的文件名。

- 功能：根据学生的总分构造一个链表，并返回链表的头节点。

4. 函数 `displaylist`：

- 参数：接收链表的头节点。

- 功能：顺序显示链表中的所有学生记录。

5. 函数 `retrievebyn`：

- 参数：接收学生成绩记录文件的文件名和要查找的学生姓名。

- 功能：按照学生姓名查找学生的记录，并显示找到的记录。

6. 函数 `retrievebyc`：

- 参数：接收学生成绩记录文件的文件名和要查找的学生学号。

- 功能：按照学生学号查找学生的记录，并显示找到的记录。

这些函数的作用是进一步实现具体的功能，例如计算总分、列出所有记录、按姓名或学号查找记录等。通过调用这些函数，可以更灵活地管理学生的考试分数记录。

还有一个重要的函数是 `readrecord`，它有以下功能：

1. 函数 `readrecord`：

- 参数：接收一个文件指针和一个指向 `struct record` 结构的指针。

- 功能：从文件中读取一条学生记录，并将其存储在 `struct record` 结构中。

- 返回值：如果成功读取一条记录，则返回 1；如果文件已结束，返回 0。

这个函数在程序中被多次调用，用于从文件中逐行读取学生记录，并将相关信息存储在结构体中。每次调用该函数，它会读取一行记录的姓名、学号和各门课程的成绩，并计算出该学生的总分。这样，程序就能够在后续的操作中使用这些学生记录进行各种统计和查询操作。

通过以上这些功能的划分，整个学生成绩管理程序能够实现对学生记录的读取、存储、计算和查询等操作。用户可以通过不同的命令选择所需的功能，对学生成绩进行管理和统计。

**2、流程图**

开始

├─ 输入学生记录文件名

├─ 检查文件是否存在

│ ├─ 文件不存在

│ │ ├─ 询问是否创建新文件

│ │ │ ├─ 是

│ │ │ │ ├─ 创建文件

│ │ │ │ ├─ 输入要写入的记录数

│ │ │ │ │ ├─ 循环：

│ │ │ │ │ │ ├─ 输入学生姓名、学号和各科成绩

│ │ │ │ │ ├─ 写入记录到文件

│ │ │ │ └─ 关闭文件

│ │ │ ├─ 否

│ │ │ └─ 结束

│ ├─ 文件存在

│ │ ├─ 关闭文件

│ └─ 进入主循环

│ ├─ 输入命令

│ ├─ 检查命令类型

│ │ ├─ 平均分命令

│ │ │ ├─ 计算各科平均分和总记录数

│ │ │ └─ 显示各科平均分

│ │ ├─ 总分命令

│ │ │ ├─ 计算各科总分和总记录数

│ │ │ └─ 显示各科总分

│ │ ├─ 按姓名查找命令

│ │ │ ├─ 输入要查找的姓名

│ │ │ └─ 根据姓名查找记录并显示

│ │ ├─ 按学号查找命令

│ │ │ ├─ 输入要查找的学号

│ │ │ └─ 根据学号查找记录并显示

│ │ ├─ 列出记录命令

│ │ │ ├─ 列出所有学生记录并逐条显示

│ │ │ └─ 每条记录显示完后等待用户按回车键继续

│ │ ├─ 按总分排序命令

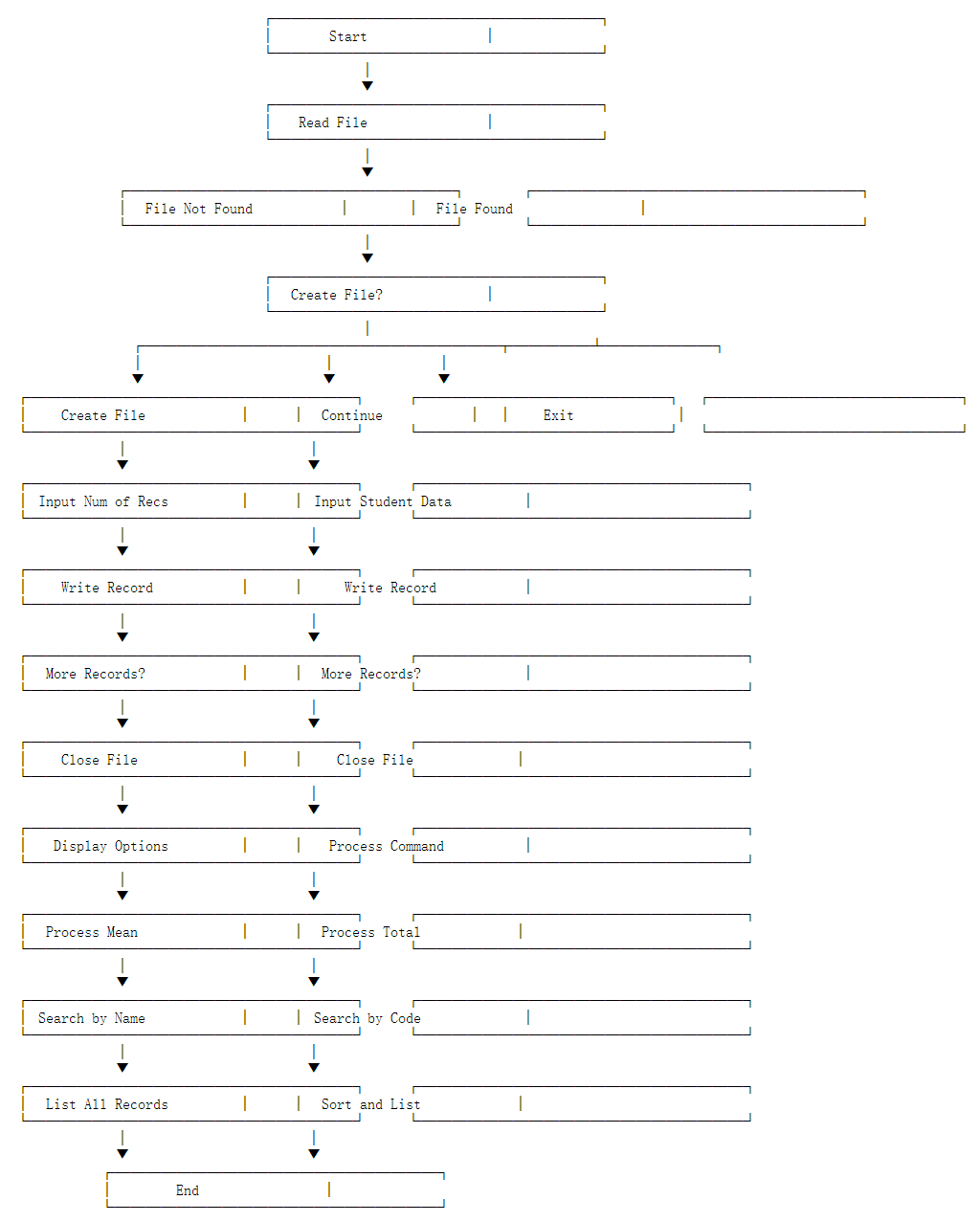
│ │ │ ├─ 构造链表，按总分从高到低排列

│ │ │ └─ 逐条显示排序后的学生记录

│ │ └─ 无效命令

│ └─ 重复主循环

结束



# 三、程序设计、运行结果

## 1、程序设计代码

编制一个统计学生考试分数的管理程序。

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SWN 3 /\* 课程数 \*/

#define NAMELEN 20 /\* 姓名最大字符数 \*/

#define CODELEN 10 /\* 学号最大字符数 \*/

#define FNAMELEN 80 /\* 文件名最大字符数 \*/

#define BUFLEN 80 /\* 缓冲区最大字符数 \*/

/\* 课程名称表 \*/

char schoolwork[SWN][NAMELEN + 1] = {"Chinese", "Mathematic", "English"};

struct record {

char name[NAMELEN + 1]; /\* 姓名 \*/

char code[CODELEN + 1]; /\* 学号 \*/

int marks[SWN]; /\* 各课程成绩 \*/

int total; /\* 总分 \*/

};

struct node {

char name[NAMELEN + 1]; /\* 姓名 \*/

char code[CODELEN + 1]; /\* 学号 \*/

int marks[SWN]; /\* 各课程成绩 \*/

int total; /\* 总分 \*/

struct node \*next; /\* 后续表元指针 \*/

};

while (1) {

puts("Please input command:");

scanf(" %c", &c);

if (c == 'q' || c == 'Q') {

puts("\nThank you for using.");

break;

}

switch (c) {

case 'm': /\* 计算平均分 \*/

case 'M':

if ((n = totalmark(stuf)) == 0) {

puts("Error!");

break;

}

printf("\n");

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

printf("%-15s's average is: %.2f.\n", schoolwork[i], (float) total[i] / n);

}

break;

case 't': /\* 计算总分 \*/

case 'T':

if ((n = totalmark(stuf)) == 0) {

puts("Error!");

break;

}

printf("\n");

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

printf("%-15s's total mark is: %d.\n", schoolwork[i], total[i]);

}

break;

case 'n': /\* 按学生的姓名寻找记录 \*/

case 'N':

printf("Please input the student's name you want to search: ");

scanf("%s", stuf);

retrievebyn(stuf, stuf);

break;

case 'c': /\* 按学生的学号寻找记录 \*/

case 'C':

printf("Please input the student's code you want to search: ");

scanf("%s", stuf);

retrievebyc(stuf, stuf);

break;

case 'l': /\* 列出所有学生记录 \*/

case 'L':

liststu(stuf);

break;

case 's': /\* 按总分从高到低排列显示 \*/

case 'S':

if ((head = makelist(stuf)) != NULL) {

displaylist(head);

}

break;

default:

break;

}

}

return 0;

}

设学生成绩已以一个学生一个记录的形式存储在文件中

int readrecord(FILE \*fpt, struct record \*rpt) {

char buf[BUFLEN];

int i;

if (fscanf(fpt, "%s", buf) != 1) {

return 0; /\* 文件结束 \*/

}

strncpy(rpt->name, buf, NAMELEN);

fscanf(fpt, "%s", buf);

strncpy(rpt->code, buf, CODELEN);

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

fscanf(fpt, "%d", &rpt->marks[i]);

}

for (rpt->total = 0, i = 0; i < SWN; i++) {

rpt->total += rpt->marks[i];

}

return 1;

}

/\* 对指定文件写入一个记录 \*/

void writerecord(FILE \*fpt, struct record \*rpt) {

int i;

fprintf(fpt, "%s\n", rpt->name);

fprintf(fpt, "%s\n", rpt->code);

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

fprintf(fpt, "%d\n", rpt->marks[i]);

}

}

/\* 显示学生记录 \*/

void displaystu(struct record \*rpt) {

int i;

printf("\nName : %s\n", rpt->name);

printf("Code : %s\n", rpt->code);

printf("Marks :\n");

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

printf(" %-15s : %4d\n", schoolwork[i], rpt->marks[i]);

}

printf("Total : %4d\n", rpt->total);

}

/\* 计算各单科总分 \*/

int totalmark(char \*fname) {

FILE \*fp;

struct record s;

int count, i;

if ((fp = fopen(fname, "r")) == NULL) {

printf("Can't open file %s.\n", fname);

return 0;

}

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

total[i] = 0;

}

count = 0;

while (readrecord(fp, &s) != 0) {

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

total[i] += s.marks[i];

}

count++;

}

fclose(fp);

return count; /\* 返回记录数 \*/

}

/\* 列表显示学生信息 \*/

void liststu(char \*fname) {

FILE \*fp;

struct record s;

if ((fp = fopen(fname, "r")) == NULL) {

printf("Can't open file %s.\n", fname);

return;

}

while (readrecord(fp, &s) != 0) {

displaystu(&s);

printf("\n Press ENTER to continue...\n");

while (getchar() != '\n');

}

fclose(fp);

}

/\* 从指定文件读入一个记录 \*/

int readrecord(FILE \*fpt, struct record \*rpt) {

char buf[BUFLEN];

int i;

if (fscanf(fpt, "%s", buf) != 1) {

return 0; /\* 文件结束 \*/

}

strncpy(rpt->name, buf, NAMELEN);

fscanf(fpt, "%s", buf);

strncpy(rpt->code, buf, CODELEN);

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

fscanf(fpt, "%d", &rpt->marks[i]);

}

for (rpt->total = 0, i = 0; i < SWN; i++) {

rpt->total += rpt->marks[i];

}

return 1;

}

/\* 对指定文件写入一个记录 \*/

void writerecord(FILE \*fpt, struct record \*rpt) {

int i;

fprintf(fpt, "%s\n", rpt->name);

fprintf(fpt, "%s\n", rpt->code);

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

fprintf(fpt, "%d\n", rpt->marks[i]);

}

}

/\* 显示学生记录 \*/

void displaystu(struct record \*rpt) {

int i;

printf("\nName : %s\n", rpt->name);

printf("Code : %s\n", rpt->code);

printf("Marks :\n");

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

printf(" %-15s : %4d\n", schoolwork[i], rpt->marks[i]);

}

printf("Total : %4d\n", rpt->total);

}

/\* 计算各单科总分 \*/

int totalmark(char \*fname) {

FILE \*fp;

struct record s;

int count, i;

if ((fp = fopen(fname, "r")) == NULL) {

printf("Can't open file %s.\n", fname);

return 0;

}

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

total[i] = 0;

}

count = 0;

while (readrecord(fp, &s) != 0) {

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

total[i] += s.marks[i];

}

count++;

}

fclose(fp);

return count; /\* 返回记录数 \*/

}

/\* 列表显示学生信息 \*/

void liststu(char \*fname) {

FILE \*fp;

struct record s;

if ((fp = fopen(fname, "r")) == NULL) {

printf("Can't open file %s.\n", fname);

return;

}

while (readrecord(fp, &s) != 0) {

displaystu(&s);

printf("\n Press ENTER to continue...\n");

while (getchar() != '\n');

}

fclose(fp);

}

/\* 构造链表 \*/

struct node \*makelist(char \*fname) {

FILE \*fp;

struct record s;

struct node \*p, \*u, \*v, \*h;

int i;

if ((fp = fopen(fname, "r")) == NULL) {

printf("Can't open file %s.\n", fname);

return NULL;

}

h = NULL;

p = (struct node \*)malloc(sizeof(struct node));

while (readrecord(fp, (struct record \*)p) != 0) {

v = h;

while (v && p->total <= v->total) {

u = v;

v = v->next;

}

if (v == h) {

h = p;

} else {

u->next = p;

}

p->next = v;

p = (struct node \*)malloc(sizeof(struct node));

}

free(p);

fclose(fp);

return h;

}

/\* 顺序显示链表各表元 \*/

void displaylist(struct node \*h) {

while (h != NULL) {

displaystu((struct record \*)h);

printf("\n Press ENTER to continue...\n");

while (getchar() != '\n');

h = h->next;

}

}

/\* 按学生姓名查找学生记录 \*/

int retrievebyn(char \*fname, char \*key) {

FILE \*fp;

int c;

struct record s;

if ((fp = fopen(fname, "r")) == NULL) {

printf("Can't open file %s.\n", fname);

return 0;

}

c = 0;

while (readrecord(fp, &s) != 0) {

if (strcmp(s.name, key) == 0) {

displaystu(&s);

c++;

}

}

fclose(fp);

if (c == 0) {

printf("The student's name %s doesn't exist.\n", key);

}

return c;

}

/\* 按学生学号查找学生记录 \*/

int retrievebyc(char \*fname, char \*key) {

FILE \*fp;

int c;

struct record s;

if ((fp = fopen(fname, "r")) == NULL) {

printf("Can't open file %s.\n", fname);

return 0;

}

c = 0;

while (readrecord(fp, &s) != 0) {

if (strcmp(s.code, key) == 0) {

displaystu(&s);

c++;

}

}

fclose(fp);

if (c == 0) {

printf("The student's code %s doesn't exist.\n", key);

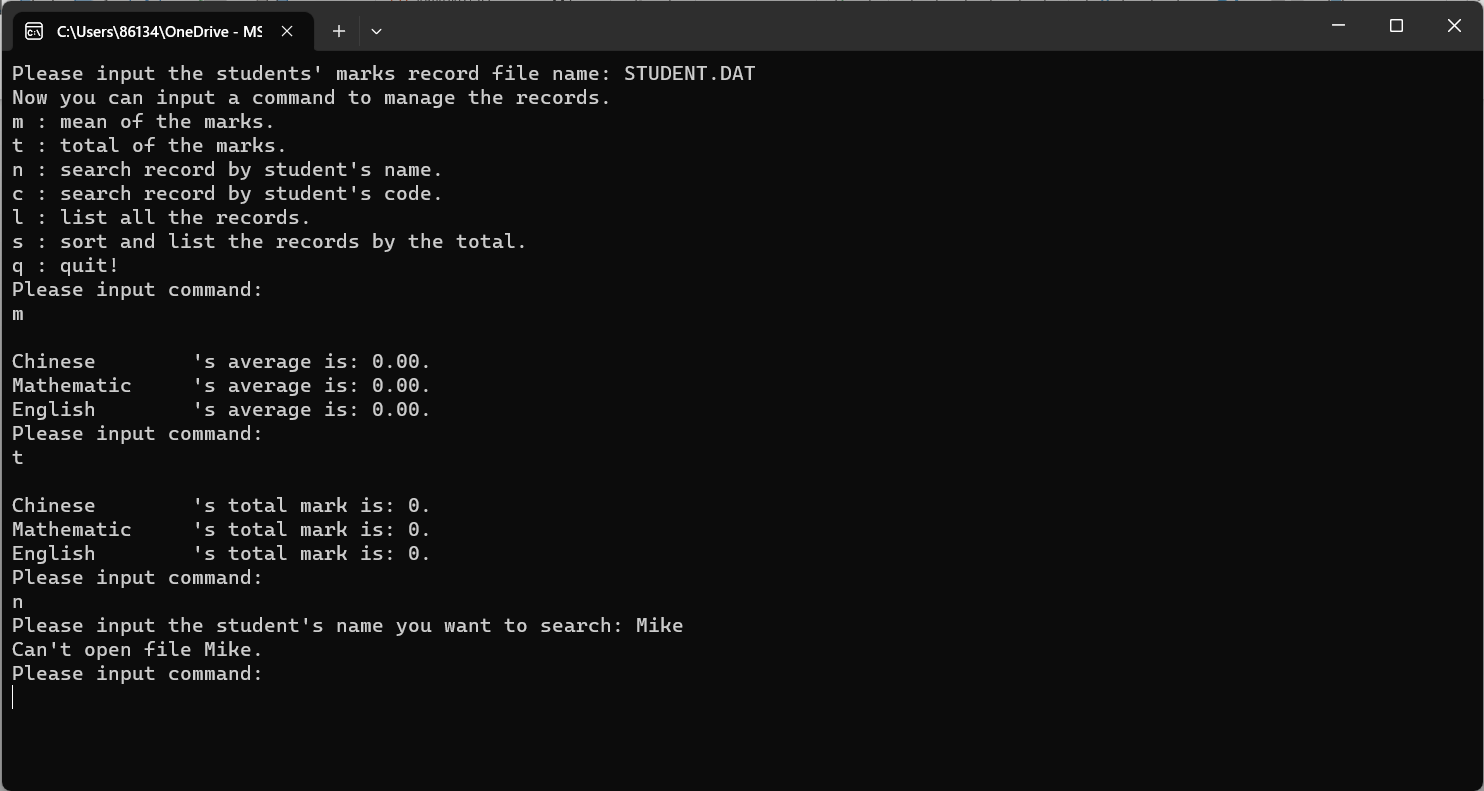
}

return c;

}

## 2、运行结果截图

运行结果



# 四、心得体会

心得体会

我学习和应用数据结构的过程中，我得出了以下几点心得体会：

1.数据结构是解决问题的基础：在编写程序时，选择合适的数据结构对于解决问题非常关键。不同的数据结构适用于不同类型的问题，例如数组适用于顺序访问和随机访问，链表适用于频繁的插入和删除操作。了解各种数据结构的特点和适用场景，可以提高程序的效率和性能。

2.数据结构的选择影响算法的效率：数据结构和算法是相辅相成的。合理选择数据结构可以帮助我们设计出高效的算法。例如，使用哈希表可以实现快速的查找操作，使用树结构可以实现高效的排序和搜索操作。因此，在解决问题时，需要综合考虑数据结构和算法之间的关系，以实现最佳的性能和效率。

3.抽象和封装是数据结构的核心思想：数据结构的设计应该注重抽象和封装的原则。通过定义合适的数据结构和操作，可以将数据和操作进行逻辑上的封装，提高代码的可读性和可维护性。良好的抽象可以隐藏底层的实现细节，使代码更具可复用性。

4.数据结构的学习需要理论和实践相结合：数据结构是一门理论学科，但仅仅停留在理论上是远远不够的。实践是深入理解和掌握数据结构的关键。通过编写代码实现各种数据结构，可以加深对其原理和应用的理解。同时，通过解决实际问题来应用数据结构，可以锻炼自己的编程能力和思维方式。

总的来说，数据结构是计算机科学中非常重要的基础知识。了解和掌握各种数据结构的特点和应用场景，对于编写高效和可维护的程序至关重要。通过不断学习和实践，我相信在数据结构方面的知识和技能会为我在编程领域的发展和成长带来很大的帮助。

# 五、参考文献

1. 严蔚敏，吴伟民.数据结构（c语言版）【M】.北京：清华大学出版社，2011.
2. 严蔚敏，陈文博.数据结构及应用算法教程【M】.北京：清华大学出版社，2011.
3. 王红梅，胡明，王涛.数据结构（c++版）【M】.北京：清华大学出版社，2011.