#include <stdio.h> // 引入标准输入输出库

#include <stdlib.h> // 引入标准库

#include <string.h> // 引入字符串处理库

// 定义学生结构体（结构体是定义一种类型）

typedef struct Student {

int studentID; // 学号

char name[50]; // 姓名

char gender[10]; // 性别

char birthDate[20]; // 出生日期

char major[50]; // 专业

char className[50]; // 班级

char college[50]; // 所属学院

float moralEducationScore; // 德育分

float physicalEducationScore; // 体育分

float intellectualEducationScore; // 智育分

float comprehensiveScore; // 综合分

struct Student\* next; // 指向下一个学生的指针

} Student;（student是类型！！）

// 链表头指针

Student\* head = NULL;

// 计算综合分数

float calculateComprehensiveScore(Student\* s) { 有返回值有参数的自定义函数

return 0.7 \* s->intellectualEducationScore + 0.2 \* s->moralEducationScore +0.1 \* s->physicalEducationScore;

}

// 添加学生

void addStudent() {

Student\* newStudent = (Student\*)malloc(sizeof(Student)); // 动态分配内存给新学生

printf("请输入学生信息：\n"); // 提示输入学生信息

printf("学号: ");

scanf("%d", &newStudent->studentID); // 输入学号

printf("姓名: ");

scanf("%s", newStudent->name); // 输入姓名

printf("性别: ");

scanf("%s", newStudent->gender); // 输入性别

printf("出生日期: ");

scanf("%s", newStudent->birthDate); // 输入出生日期

printf("专业: ");

scanf("%s", newStudent->major); // 输入专业

printf("班级: ");

scanf("%s", newStudent->className); // 输入班级

printf("所属学院: ");

scanf("%s", newStudent->college); // 输入所属学院

printf("德育分: ");

scanf("%f", &newStudent->moralEducationScore); // 输入德育分

printf("体育分: ");

scanf("%f", &newStudent->physicalEducationScore); // 输入体育分

printf("智育分: ");

scanf("%f", &newStudent->intellectualEducationScore); // 输入智育分

newStudent->comprehensiveScore = calculateComprehensiveScore(newStudent); // 计算综合分

newStudent->next = head; // 将新学生插入链表头

head = newStudent; // 更新链表头指针

printf("学生信息添加成功。\n"); // 提示添加成功

}

// 删除学生

void deleteStudentByID() {

int studentID; // 学号

printf("请输入要删除的学生学号："); // 提示输入要删除的学号

scanf("%d", &studentID); // 输入学号

Student \*current = head, \*previous = NULL; // 当前学生和前一个学生指针（把current指针放到首位head，以便接下来遍历整个链表！！，previous和current连在一起理解）

while (current != NULL) { // 遍历链表

if (current->studentID == studentID) { // 如果找到学号匹配的学生

if (previous == NULL) { // 如果是链表头

head = current->next; // 更新链表头指针

} else {

previous->next = current->next; // 前一个学生指向下一个学生

}

free(current); // 释放当前学生内存

printf("学生信息删除成功。\n"); // 提示删除成功

return;

}

previous = current; // 更新前一个学生指针

current = current->next; // 更新当前学生指针

}

printf("未找到该学号的学生。\n"); // 提示未找到学生

}

// 更新学生信息

void updateStudentInfoByID() {

int studentID; // 学号

printf("请输入要更新的学生学号："); // 提示输入学号

scanf("%d", &studentID); // 输入学号

Student\* current = head; // 当前学生指针

while (current != NULL) { // 遍历链表

if (current->studentID == studentID) { // 如果找到学号匹配的学生

printf("请输入新的信息：\n"); // 提示输入新信息

printf("德育分: ");

scanf("%f", &current->moralEducationScore); // 输入新的德育分

printf("体育分: ");

scanf("%f", &current->physicalEducationScore); // 输入新的体育分

printf("智育分: ");

scanf("%f", &current->intellectualEducationScore); // 输入新的智育分

current->comprehensiveScore = calculateComprehensiveScore(current); // 计算新的综合分

printf("信息更新成功。\n"); // 提示更新成功

return;

}

current = current->next; // 更新当前学生指针

}

printf("未找到该学号的学生。\n"); // 提示未找到学生

}

// 查询学生信息

void queryStudentInfo() {

int choice; // 查询方式选择

printf("请选择查询方式：\n");

printf("1. 按学号查询\n");

printf("2. 按姓名查询\n");

printf("3. 按专业查询\n");

scanf("%d", &choice); // 输入选择

Student\* current = head; // 当前学生指针

switch (choice) {

case 1: {

int studentID; // 学号

printf("请输入学号："); // 提示输入学号

scanf("%d", &studentID); // 输入学号

while (current != NULL) { // 遍历链表

if (current->studentID == studentID) { // 如果找到学号匹配的学生

printf("查询结果：\n学号:%d 姓名:%s 性别:%s 出生日期:%s 专业:%s 班级:%s 所属学院:%s 德育分:%.2f 体育分:%.2f 智育分:%.2f 综合分:%.2f\n",

current->studentID, current->name, current->gender, current->birthDate,

current->major, current->className, current->college,

current->moralEducationScore, current->physicalEducationScore,

current->intellectualEducationScore, current->comprehensiveScore); // 输出学生信息

return;

}

current = current->next; // 更新当前学生指针

}

printf("未找到该学号的学生。\n"); // 提示未找到学生

break;

}

case 2: case 3: {

char queryStr[50]; // 查询字符串

printf("请输入姓名或专业："); // 提示输入姓名或专业

scanf("%s", queryStr); // 输入查询字符串

while (current != NULL) { // 遍历链表

if ((choice == 2 && strcmp(current->name, queryStr) == 0) ||

(choice == 3 && strcmp(current->major, queryStr) == 0)) { // 如果找到匹配的学生

printf("查询结果：\n学号:%d 姓名:%s 性别:%s 出生日期:%s 专业:%s 班级:%s 所属学院:%s 德育分:%.2f 体育分:%.2f 智育分:%.2f 综合分:%.2f\n",

current->studentID, current->name, current->gender, current->birthDate,

current->major, current->className, current->college,

current->moralEducationScore, current->physicalEducationScore,

current->intellectualEducationScore, current->comprehensiveScore); // 输出学生信息

}

current = current->next; // 更新当前学生指针

}

break;

}

default: printf("无效的选择。\n"); // 提示无效选择

}

}

// 排序学生信息

void sortStudentsByID() {

if (head == NULL || head->next == NULL) return; // 如果链表为空或只有一个节点，直接返回

Student\* sorted = NULL; // 已排序的链表 这里是student类型定义\*sorted的意思

Student\* current = head; // 当前学生指针

while (current != NULL) { // 遍历链表

Student\* next = current->next; // 下一个学生指针

if (sorted == NULL || sorted->studentID >= current->studentID) { // 如果已排序链表为空或当前学生学号小于或等于已排序链表头

current->next = sorted; // 将当前学生插入已排序链表头

sorted = current; // 更新已排序链表头

} else {

Student\* temp = sorted; // 临时指针

while (temp->next != NULL && temp->next->studentID < current->studentID) { // 找到插入位置

temp = temp->next; // 更新临时指针

}

current->next = temp->next; // 将当前学生插入已排序链表

temp->next = current; // 更新已排序链表

}

current = next; // 更新当前学生指针

}

head = sorted; // 更新链表头指针

printf("排序完成。\n"); // 提示排序完成

}

// 浏览所有学生信息

void browseAllStudents() {

printf("所有学生信息如下：\n"); // 提示浏览所有学生信息

Student\* current = head; // 当前学生指针

while (current != NULL) { // 遍历链表

printf("学号:%d 姓名:%s 性别:%s 出生日期:%s 专业:%s 班级:%s 所属学院:%s 德育分:%.2f 体育分:%.2f 智育分:%.2f 综合分:%.2f\n",

current->studentID, current->name, current->gender, current->birthDate,

current->major, current->className, current->college,

current->moralEducationScore, current->physicalEducationScore,

current->intellectualEducationScore, current->comprehensiveScore); // 输出学生信息

current = current->next; // 更新当前学生指针

}

}

// 统计不及格学生人数

void countFailingStudentsInCollegeAndClass() {

char college[50], className[50]; // 学院和班级

printf("请输入学院和班级："); // 提示输入学院和班级

scanf("%s %s", college, className); // 输入学院和班级

int count = 0; // 不及格学生人数

Student\* current = head; // 当前学生指针

while (current != NULL) { // 遍历链表

if (strcmp(current->college, college) == 0 && strcmp(current->className, className) == 0 && current->comprehensiveScore < 60) { // 如果找到匹配的学生且综合分小于60

count++; // 计数加1

}

current = current->next; // 更新当前学生指针

}

printf("不及格学生人数：%d\n", count); // 输出不及格学生人数

}

// 保存数据到文件

void saveDataToFile() {

FILE \*file = fopen("students.dat", "wb"); // 打开文件

if (file == NULL) { // 如果文件打开失败

printf("无法打开文件。\n"); // 提示无法打开文件

return;

}

Student\* current = head; // 当前学生指针

while (current != NULL) { // 遍历链表

fwrite(current, sizeof(Student), 1, file); // 将学生数据写入文件

current = current->next; // 更新当前学生指针

}

fclose(file); // 关闭文件

printf("数据保存成功。\n"); // 提示数据保存成功

}

// 从文件读取数据

void loadDataFromFile() {

FILE \*file = fopen("students.dat", "rb"); // 打开文件

if (file == NULL) { // 如果文件打开失败

printf("无法打开文件。\n"); // 提示无法打开文件

return;

}

Student\* newStudent; // 新学生指针

while (!feof(file)) { // 循环读取文件

newStudent = (Student\*)malloc(sizeof(Student)); // 动态分配内存

if (fread(newStudent, sizeof(Student), 1, file) == 1) { // 如果读取成功

newStudent->next = head; // 将新学生插入链表头

head = newStudent; // 更新链表头指针

} else {

free(newStudent); // 释放内存

}

}

fclose(file); // 关闭文件

printf("数据读取成功。\n"); // 提示数据读取成功

}

int main() {

// 主菜单

while (1) {

printf("\n学生考评信息管理系统\n"); // 输出系统标题

printf("1. 添加学生\n");

printf("2. 删除学生\n");

printf("3. 更新学生信息\n");

printf("4. 查询学生信息\n");

printf("5. 排序学生信息\n");

printf("6. 浏览所有学生信息\n");

printf("7. 统计不及格学生人数\n");

printf("8. 保存数据\n");

printf("9. 读取数据\n");

printf("0. 退出\n");

int choice; // 用户选择

scanf("%d", &choice); // 输入选择

switch (choice) {

case 1: addStudent(); break; // 添加学生

case 2: deleteStudentByID(); break; // 删除学生

case 3: updateStudentInfoByID(); break; // 更新学生信息

case 4: queryStudentInfo(); break; // 查询学生信息

case 5: sortStudentsByID(); break; // 排序学生信息

case 6: browseAllStudents(); break; // 浏览所有学生信息

case 7: countFailingStudentsInCollegeAndClass(); break; // 统计不及格学生人数

case 8: saveDataToFile(); break; // 保存数据

case 9: loadDataFromFile(); break; // 读取数据

case 0: return 0; // 退出

default: printf("无效的选择\n"); // 提示无效选择

}

}

return 0;

}