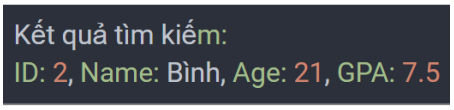
Chức năng tra cứu sinh viên. Nhiệm vụ:

1. Sử dụng Cursor AI viết hàm nhập ID cần tìm.
2. In thông tin nếu tìm thấy, ngược lại báo “Không tìm thấy”.
3. Nhờ AI sinh docstring cho hàm tìm kiếm.
4. Prompt AI tạo test case: ID tồn tại, ID không tồn tại, ID âm.

- prompt gợi ý: *“Hãy bổ sung vào Student Manager hàm tìm kiếm sinh viên theo ID. Thêm docstring cho hàm. Sinh test case cho các trường hợp: ID tồn tại, không tồn tại, âm.”*

  Input: 

  Output : 

Prompt :

"Hãy viết một chương trình C hoàn chỉnh để quản lý sinh viên với các yêu cầu chi tiết sau:

1. Cấu trúc dữ liệu và Khởi tạo:

- Định nghĩa `struct Student` bao gồm: `id` (int), `name` (chuỗi), `age` (int), và `gpa` (float).
- Trong hàm `main`, khởi tạo sẵn một mảng chứa 3 sinh viên mẫu. In danh sách này ra màn hình dưới dạng bảng có tiêu đề cột.

2. Chức năng Nhập liệu:

- Bổ sung chức năng cho phép người dùng nhập thêm tối đa 5 sinh viên mới từ bàn phím. Sau khi nhập xong, in toàn bộ danh sách sinh viên hiện có ra màn hình.

3. Xử lý File:

- Viết hàm `saveToFile`: Lưu toàn bộ danh sách sinh viên hiện tại vào file `students.txt`. Mỗi sinh viên nằm trên một dòng, các trường cách nhau bằng khoảng trắng.
- Viết hàm đọc dữ liệu từ `students.txt` và lưu vào mảng `Student`.

4. Chức năng Tìm kiếm & Docstring:

- Viết hàm `findStudentByID` để tìm kiếm sinh viên trong mảng dựa trên ID truyền vào.
- Yêu cầu Docstring: Thêm chú thích cho hàm này theo phong cách chuyên nghiệp (mô tả chức năng, các tham số đầu vào, và giá trị trả về).

5. Kiểm thử (Unit Test):

- Trong hàm `main`, sau khi thực hiện các chức năng trên, hãy tự động chạy và in kết quả kiểm thử cho hàm tìm kiếm với 3 kịch bản:
    1. ID tồn tại: Tìm một ID có trong danh sách.
    2. ID không tồn tại: Tìm một ID không có trong danh sách.
    3. ID là số âm: Truyền vào một giá trị ID âm để kiểm tra tính hợp lệ.

6. Định dạng Output (Quan trọng):

- Khi in danh sách đọc được từ file, định dạng phải khớp chính xác với mẫu sau:
    - Dòng tiêu đề: `Danh sách đọc từ file:`
    - Mỗi dòng dữ liệu: `[id] [name] [age] [gpa]`
    - Điểm GPA phải hiển thị đúng 1 chữ số thập phân (ví dụ: `8.0`, `7.5`).

7. Yêu cầu kỹ thuật:

- Sử dụng `#define` để quản lý kích thước mảng và tên file.
- Có kiểm tra lỗi khi mở file và kiểm tra tràn mảng khi đọc/nhập dữ liệu."

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX_STUDENTS 8 // 3 sample + up to 5 from input
#define FILE_NAME "students.txt"

struct Student {
    int id;
    char name[50];
    int age;
    float gpa;
};

// In header
void printHeader() {
    printf("\n%-10s %-30s %-10s %-10s\n", "ID", "Ten", "Tuoi", "GPA");
    printf("-------------------------------------------------------------------\n");
}

// Print a single student
void printStudent(struct Student s) {
    printf("%-10d %-30s %-10d %-10.1f\n", s.id, s.name, s.age, s.gpa);
}

// Print a student list in table format
void printStudentList(struct Student students[], int count) {
    printHeader();
    for (int i = 0; i < count; i++) {
        printStudent(students[i]);
    }
    printf("\n");
}

// Lưu danh sách sinh viên vào file
void saveToFile(struct Student students[], int count, const char* filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        printf("Khong the mo file de ghi!\n");
        return;
    }
    for (int i = 0; i < count; i++) {
        fprintf(file, "%d %s %d %.1f\n",
                students[i].id,
                students[i].name,
                students[i].age,
                students[i].gpa);
    }
    fclose(file);
    printf("Da luu %d sinh vien vao file %s\n", count, filename);
}
```

```c
// Đọc dữ liệu sinh viên từ file vào mảng. Trả về số sinh viên đã đọc
int readFromFile(struct Student students[], int maxCount, const char* filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("Khong the mo file de doc!\n");
        return 0;
    }
    int count = 0;
    while (count < maxCount &&
           fscanf(file, "%d %49s %d %f",
                  &students[count].id,
                  students[count].name,
                  &students[count].age,
                  &students[count].gpa) == 4) {
        count++;
    }
    fclose(file);
    return count;
}

// Hàm in danh sách sinh viên theo định dạng yêu cầu khi đọc từ file
void printListFromFile(struct Student students[], int count) {
    printf("Danh sach doc tu file:\n");
    for (int i = 0; i < count; i++) {
        printf("%d %s %d %.1f\n",
               students[i].id, students[i].name, students[i].age, students[i].gpa);
    }
}

// Nhập thêm sinh viên từ bàn phím, trả về số lượng đã nhập
int inputStudents(struct Student students[], int currentCount, int maxCount) {
    int nAdd = 0;
    printf("\nNhap so luong sinh vien muon them (toi da %d): ", maxCount - currentCount);
    scanf("%d", &nAdd);
    if (nAdd < 1) {
        printf("Khong them sinh vien moi.\n");
        return 0;
    }
    if (nAdd > maxCount - currentCount) nAdd = maxCount - currentCount;
    for (int i = 0; i < nAdd; i++) {
        printf("\nNhap thong tin sinh vien thu %d:\n", currentCount + i + 1);
        printf("ID: ");
        scanf("%d", &students[currentCount + i].id);
        printf("Ten: ");
        scanf("%49s", students[currentCount + i].name);
        printf("Tuoi: ");
        scanf("%d", &students[currentCount + i].age);
        printf("GPA: ");
        scanf("%f", &students[currentCount + i].gpa);
    }
    return nAdd;
}
```

```c
 * Tìm kiếm sinh viên theo ID trong mảng.
 * @param students Mảng các sinh viên.
 * @param count Số lượng sinh viên trong mảng.
 * @param id ID của sinh viên cần tìm.
 * @return Con trỏ tới sinh viên nếu tìm thấy, NULL nếu không tìm thấy hoặc ID không hợp lệ.
 */
struct Student* findStudentByID(struct Student students[], int count, int id) {
    if (id < 0) return NULL;
    for (int i = 0; i < count; i++) {
        if (students[i].id == id) {
            return &students[i];
        }
    }
    return NULL;
}

int main() {
    struct Student students[MAX_STUDENTS] = {
        {1, "An", 20, 8.0},
        {2, "Binh", 21, 7.5},
        {3, "Cuong", 19, 8.5},
    };
    int studentCount = 3;
    printf("\n=== DANH SACH SINH VIEN BAN DAU ===\n");
    printStudentList(students, studentCount);

    // Nhập thêm sinh viên
    int nAdded = inputStudents(students, studentCount, MAX_STUDENTS);
    studentCount += nAdded;

    printf("\n=== DANH SACH SINH VIEN SAU KHI THEM ===\n");
    printStudentList(students, studentCount);

    // Lưu vào file
    saveToFile(students, studentCount, FILE_NAME);

    // Đọc lại từ file để kiểm chứng
    struct Student fromFile[MAX_STUDENTS];
    int nFromFile = readFromFile(fromFile, MAX_STUDENTS, FILE_NAME);
    printListFromFile(fromFile, nFromFile);

    // Kiểm thử findStudentByID với 3 kịch bản
    printf("\n=== TEST TIM KIEM SINH VIEN THEO ID ===\n");
```

```c
    // 1. ID tồn tại
    int test_id = students[1].id; // Chắc chắn tồn tại
    struct Student* found = findStudentByID(students, studentCount, test_id);
    printf("\nTest case 1: Tim ID = %d (ton tai)\n", test_id);
    if (found) {
        printf("Tim thay sinh vien: ID=%d, Ten=%s, Tuoi=%d, GPA=%.1f\n", found->id, found->name, found->age, found->gpa);
    } else {
        printf("Khong tim thay sinh vien co ID = %d\n", test_id);
    }

    // 2. ID không tồn tại
    int not_exist_id = 999;
    found = findStudentByID(students, studentCount, not_exist_id);
    printf("\nTest case 2: Tim ID = %d (khong ton tai)\n", not_exist_id);
    if (found) {
        printf("Tim thay sinh vien: ID=%d, Ten=%s, Tuoi=%d, GPA=%.1f\n", found->id, found->name, found->age, found->gpa);
    } else {
        printf("Khong tim thay sinh vien co ID = %d\n", not_exist_id);
    }

    // 3. ID âm
    int neg_id = -5;
    found = findStudentByID(students, studentCount, neg_id);
    printf("\nTest case 3: Tim ID = %d (so am)\n", neg_id);
    if (found) {
        printf("Tim thay sinh vien: ID=%d, Ten=%s, Tuoi=%d, GPA=%.1f\n", found->id, found->name, found->age, found->gpa);
    } else {
        printf("ID = %d khong hop le (so am hoac khong ton tai)\n", neg_id);
    }

    return 0;
}
```

Kết quả :

```
C:\Users\Lenovo\OneDrive\D
ID          Ten                                 Tuoi        GPA
-----------------------------------------------------------------
1           An                                  20          8.0
2           Binh                                21          7.5
3           Cuong                               19          8.5
4           Dung                                20          8.0
5           Ethan                               21          9.0

Da luu 5 sinh vien vao file students.txt
Danh sach doc tu file:
1 An 20 8.0
2 Binh 21 7.5
3 Cuong 19 8.5
4 Dung 20 8.0
5 Ethan 21 9.0

=== TEST TIM KIEM SINH VIEN THEO ID ===

Test case 1: Tim ID = 2 (ton tai)
Tim thay sinh vien: ID=2, Ten=Binh, Tuoi=21, GPA=7.5

Test case 2: Tim ID = 999 (khong ton tai)
Khong tim thay sinh vien co ID = 999

Test case 3: Tim ID = -5 (so am)
ID = -5 khong hop le (so am hoac khong ton tai)

--------------------------------
Process exited after 39.43 seconds with return value 0
Press any key to continue . . .
```