Có nhiều lớp học khác nhau. Nhiệm vụ:

1. Sử dụng Cursor AI tạo mỗi lớp có file riêng (classA.txt, classB.txt).
2. Người dùng nhập tên file để đọc dữ liệu.
3. Prompt AI thêm comment chi tiết cho từng bước xử lý nhiều file.
4. Nhờ AI sinh dữ liệu giả lập cho 3 lớp.

Input:

```
Nhập tên file: classA.txt
```

Output :

```
Danh sách sinh viên lớp A:
1 An 20 8.0
2 Bình 21 7.5
```

Prompt :

"Hãy đóng vai một kỹ sư phần mềm chuyên nghiệp, viết một chương trình C quản lý sinh viên hoàn chỉnh với các module sau:

1. Cấu trúc dữ liệu & Khởi tạo:

- Định nghĩa `struct Student` gồm: `id` (int), `name` (chuỗi), `age` (int), và `gpa` (float).
- Khởi tạo sẵn 3 sinh viên mẫu.
- Giả lập dữ liệu: Sinh ngẫu nhiên dữ liệu cho 10 sinh viên với GPA khác nhau để phục vụ kiểm thử.
- Sinh dữ liệu giả lập cho 3 lớp học riêng biệt (lưu vào 3 mảng hoặc chuẩn bị sẵn để ghi vào file `classA.txt`, `classB.txt`, `classC.txt`).

2. Quản lý File & Đa lớp (Multi-file Handling):

- Viết hàm `saveToFile` và `readFromFile`.
- Tính năng chọn lớp: Người dùng nhập tên file (ví dụ: `classA.txt`) để hệ thống đọc dữ liệu từ file đó vào mảng `Student`.
- Yêu cầu AI: Thêm comment cực kỳ chi tiết cho từng bước xử lý file (mở file, kiểm tra tồn tại, đọc định dạng, đóng file).

3. Chức năng Mở rộng:

- Nhập liệu: Cho phép nhập thêm tối đa 5 sinh viên từ bàn phím.
- Tìm kiếm: Viết hàm `findStudentByID`. Yêu cầu có Docstring (Doxygen style) mô tả chi tiết tham số và giá trị trả về.
- Thống kê: Viết hàm `calculateAverageGPA` để tính điểm trung bình toàn lớp.
- Sắp xếp: Viết hàm `sortStudentsByGPA` (sắp xếp giảm dần).

4. Giải thuật & Logic:

- Yêu cầu AI: Hãy giải thích chi tiết logic của thuật toán sắp xếp bạn sử dụng (chọn Bubble Sort hoặc Selection Sort) bằng ngôn ngữ dễ hiểu nhất.

5. Kịch bản Kiểm thử (Test Cases):

- Hãy viết các đoạn mã hoặc hàm kiểm thử tự động (Unit Test) cho các trường hợp:
    - Tìm kiếm: ID có trong danh sách, ID không tồn tại, và ID âm.
    - Thống kê: Xử lý trường hợp danh sách rỗng (không có sinh viên) để tránh lỗi chia cho 0.
    - Sắp xếp: Kiểm thử với các bộ dữ liệu: GPA bằng nhau hoàn toàn, danh sách đã được sắp xếp tăng dần sẵn, và danh sách có GPA ngẫu nhiên.

6. Định dạng Output:

- In danh sách dưới dạng bảng đẹp mắt.
- Khi đọc từ file, in ra mẫu: `[id] [name] [age] [gpa]` với GPA lấy 1 chữ số thập phân.

7. Yêu cầu kỹ thuật:

- Sử dụng `#define` để quản lý các hằng số.
- Sử dụng con trỏ (pointer) để quản lý mảng sinh viên hiệu quả.
- Có xử lý ngoại lệ (Exception Handling) khi người dùng nhập sai tên file."

```c
/*
    ==============================
    CHUONG TRINH QUAN LY SINH VIEN
    ==============================

    - Ho tro quan ly 3 lop/hoc phan rieng biet thong qua 3 file: classA.txt, classB.txt, classC.txt
    - Sinh ngau nhien du lieu kiem thu cho tung lop
    - Cho phep nhap them sinh vien tu ban phim vao lop hien tai
    - Cac chuc nang: luu file, doc file, tim kiem, thong ke GPA, sap xep GPA, test case tu dong
*/

// --------- 1. Cau truc va Hang so ---------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define MAX_STUDENTS 18                 // Toi da cho 1 danh sach
#define MAX_NAME_LENGTH 30
#define MAX_FILENAME_LENGTH 30
#define INIT_STUDENTS 3                 // So sinh vien khoi tao san
#define MOCK_STUDENTS 10                // So sinh vien gia lap
#define CLASSES 3

// Ten file cho moi lop
#define CLASSA "classA.txt"
#define CLASSB "classB.txt"
#define CLASSC "classC.txt"

// --------- Cau truc du lieu ---------
typedef struct {
    int id;
    char name[MAX_NAME_LENGTH+1];
    int age;
    float gpa;
} Student;

// ======== Module Tien ich In bang ========

void printHeader() {
    printf("\n%-6s | %-30s | %-8s | %-5s\n", "ID", "Ten", "Tuoi", "GPA");
    printf("-----------------------------------------------------------------\n");
}

// In mot sinh vien dang bang
void printStudent(Student s) {
    printf("%-6d | %-30s | %-8d | %-5.1f\n", s.id, s.name, s.age, s.gpa);
}
```

```c
// In danh sach dang bang
void printStudentList(Student *students, int count) {
    if (count <= 0) {
        printf("(Khong co sinh vien!)\n"); return;
    }
    printHeader();
    for (int i = 0; i < count; ++i)
        printStudent(students[i]);
    printf("\n");
}

// In danh sach dang thap (doc tu file)
void printListFromFile(Student *students, int count) {
    printf("Noi dung doc tu file:\n");
    if (count <= 0) {
        printf("(Tap tin rong hoac khong co du lieu)\n");
        return;
    }
    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);
    }
}

// --------- 2. MOCKUP DATA: Sinh mo phong sinh vien va 3 lop ---------

// Ten gia lap cho du lieu random
const char *SAMPLE_NAMES[] = {
    "An", "Bao", "Cuong", "Dung", "Huyen", "Lan", "Mai", "Nam", "Phuc", "Quang",
    "Son", "Thao", "Trang", "Tuan", "Vy", "Yen", "Viet", "Hang", "Tung", "Luong"
};
#define SAMPLE_NAME_COUNT (sizeof(SAMPLE_NAMES)/sizeof(SAMPLE_NAMES[0]))

// Ham sinh 1 sinh vien ngau nhien
Student generateRandomStudent(int id) {
    Student s;
    s.id = id;
    // Lay ten ngau nhien
    strcpy(s.name, SAMPLE_NAMES[rand()%SAMPLE_NAME_COUNT]);
    s.age = 18 + rand() % 5; // 18-22
    // GPA tu 5.0 den 10.0, lam tron 1 chu so TP
    s.gpa = (float)(rand()%51 + 50)/10.0f; // 5.0->10.0
    return s;
}

// Sinh mang sinh vien ngau nhien
void generateStudents(Student *arr, int from_id, int n) {
    for (int i=0; i<n; ++i) {
        arr[i] = generateRandomStudent(from_id+i);
    }
}
```

```c
// Ghi du lieu mock vao file (ghi de toan bo)
void writeMockDataToFile(const char* filename, Student* students, int count) {
    // ===== File Writing Detail BEGIN =====
    FILE *fp = fopen(filename, "w");    // Mo che do ghi de toan bo
    if (fp == NULL) {
        printf("Khong the tao file %s!\n", filename); return;
    }
    // Ghi tung dong voi format: id name age gpa
    for (int i = 0; i < count; ++i) {
        fprintf(fp, "%d %s %d %.1f\n", students[i].id, students[i].name,
                                       students[i].age, students[i].gpa);
    }
    fclose(fp); // Nhut tap tin lai
    // ===== File Writing Detail END =====
}

// Ham khoi tao du lieu mo phong cho tat ca cac lop
void initializeAllClasses() {
    Student classA[MOCK_STUDENTS], classB[MOCK_STUDENTS], classC[MOCK_STUDENTS];
    generateStudents(classA, 1001, MOCK_STUDENTS);
    generateStudents(classB, 2001, MOCK_STUDENTS);
    generateStudents(classC, 3001, MOCK_STUDENTS);
    writeMockDataToFile(CLASSA, classA, MOCK_STUDENTS);
    writeMockDataToFile(CLASSB, classB, MOCK_STUDENTS);
    writeMockDataToFile(CLASSC, classC, MOCK_STUDENTS);
    printf("Du lieu MOCK cho 3 lop da duoc tao file.\n");
}

// --------- 3. Luu va Doc File (chi tiet) ---------

/**
 * Luu danh sach sinh vien vao file
 * @param students: Mang sinh vien
 * @param count: So luong
 * @param filename: Ten file dich
 */
void saveToFile(Student* students, int count, const char* filename) {
    // Mo file che do ghi de
    FILE *fp = fopen(filename, "w");
    // Kiem tra file mo thanh cong
    if (fp == NULL) {
        printf("[ERROR] Khong the mo file de ghi (%s)!\n", filename); return;
    }
    // Duyet va ghi tung sinh vien (id name age gpa)
    for (int i=0; i<count; ++i) {
        fprintf(fp, "%d %s %d %.1f\n", students[i].id, students[i].name,
                                       students[i].age, students[i].gpa);
    }
    // Dong file
    fclose(fp);
    printf("Da luu %d sinh vien vao file %s thanh cong.\n", count, filename);
}
```

```c
/**
 * Doc danh sach sinh vien tu file vao mang
 * @param students: Mang sinh vien de doc vao
 * @param max: So luong phan tu co the luu
 * @param filename: Ten tap tin
 * @return So luong doc duoc
 */
int readFromFile(Student* students, int max, const char* filename) {
    // ============ File Reading Step-by-step ============
    // 1. Mo file che do doc
    FILE *fp = fopen(filename, "r");
    if (fp == NULL) {
        // File khong ton tai/sai ten
        printf("[ERROR] Khong the mo file %s de doc!\n", filename);
        return 0;
    }
    int count = 0;
    // 2. Doc tung dong cho den khi het file hoac het MAX phan tu
    while (count < max &&
           fscanf(fp, "%d %30s %d %f",
                  &students[count].id,
                  students[count].name,
                  &students[count].age,
                  &students[count].gpa) == 4) {
        count++;
    }
    // 3. Dong file
    fclose(fp);
    // 4. Tra ve so luong doc duoc
    return count;
}

// --------- 4. Chon file lop & Doc du lieu ---------
/**
 * Cho nguoi dung nhap vao ten file lop va doc du lieu
 * @param students Mang de doc vao
 * @param max So SV toi da
 * @return So SV doc duoc
 */
int inputClassAndLoad(Student* students, int max) {
    char filename[MAX_FILENAME_LENGTH+1];
    printf("Nhap ten file lop de mo (vi du: classA.txt): ");
    scanf("%s", filename);
    int n = readFromFile(students, max, filename);
    if (n == 0) {
        printf("Doc file that bai hoac file rong!\n");
    } else {
        printf("Doc file thanh cong. Co %d sinh vien trong lop.\n", n);
    }
    return n;
}
```

```c
// --------- 5. THEM, TIM KIEM, THONG KE, SAP XEP ---------

/**
 * Nhap them toi da 5 sinh vien tu ban phim
 * @param students: Mang sinh vien
 * @param current So luong da co
 * @param max   Toi da cac phan tu mang
 * @return So sv nhap them
 */
int inputStudents(Student* students, int current, int max) {
    int nAdd;
    printf("\nNhap so sv muon them (toi da %d): ", (max-current>5)?5:(max-current));
    scanf("%d", &nAdd);
    if (nAdd < 1) return 0;
    if (nAdd > 5) nAdd = 5;
    if (nAdd > max-current) nAdd = max-current;
    for (int i=0; i<nAdd; ++i) {
        printf("\nNhap SV thu %d:\n", current+i+1);
        printf("ID: "); scanf("%d", &students[current+i].id);
        printf("Ten: "); scanf("%30s", students[current+i].name);
        printf("Tuoi: "); scanf("%d", &students[current+i].age);
        printf("GPA: "); scanf("%f", &students[current+i].gpa);
    }
    return nAdd;
}

/**
 * Tim sinh vien theo ID (DOXYGEN CHU GIAI)
 * @param students  Mang sinsh vien can tim
 * @param count     So sinh vien trong mang
 * @param id        ID sinh vien can tim
 * @return  Con tro SV trung khop, NULL neu khong tim thay hoac id khong hop le
 */
Student* findStudentByID(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;
}
```

```cpp
/**
 * Tinh diem trung binh GPA
 * @param students Mang SV
 * @param n So luong
 * @return -1 neu khong co SV
 */
float calculateAverageGPA(Student* students, int n) {
    if (n <= 0) return -1.0f;
    float sum = 0;
    for (int i=0; i<n; ++i)
        sum += students[i].gpa;
    return sum/n;
}

/**
 * Sap xep mang SV theo GPA giam dan (Selection Sort)
 * @param students Mang SV
 * @param n So luong
 */
void sortStudentsByGPA(Student* students, int n) {
    /* ------ GIAI THICH THUAT TOAN: SELECTION SORT ------
        - Duyet tu dau toi cuoi mang, moi lan chon SV co GPA cao nhat
        - Swap SV do ve vi tri hien tai, tiep tuc cho cac vi tri con lai
        - Khong can them bo nho phu
        - Do phuc tap: n*(n-1)/2 lan so sanh
        - Thich hop list nho/khong can on dinh thu tu
    */
    for (int i=0; i<n-1; ++i) {
        int maxIdx = i;
        for (int j=i+1; j<n; ++j) {
            if (students[j].gpa > students[maxIdx].gpa)
                maxIdx = j;
        }
        if (maxIdx != i) {
            // Swap 2 SV
            Student tmp = students[i];
            students[i] = students[maxIdx];
            students[maxIdx] = tmp;
        }
    }
}
```

```
// --------- 6. Unit Test, Test case tu dong ---------
void test_findStudentByID() {
    printf("\n--[Test findStudentByID]--\n");
    Student dummy[4] = {
        {1,"An",19,8.5},
        {2,"Binh",21,7.2},
        {3,"Cuong",18,9.0}
    };
    int size = 3;

    // 1. Tim ton tai
    Student* p = findStudentByID(dummy, size, 3);
    printf("Test ton tai (ID=3): %s\n", (p!=NULL && p->id==3) ? "PASS" : "FAIL");

    // 2. Khong ton tai
    p = findStudentByID(dummy, size, 99);
    printf("Test khong ton tai (ID=99): %s\n", (p==NULL) ? "PASS" : "FAIL");

    // 3. Am
    p = findStudentByID(dummy, size, -7);
    printf("Test ID am: %s\n", (p==NULL) ? "PASS" : "FAIL");
}

void test_calculateAverageGPA() {
    printf("\n--[Test calculateAverageGPA]--\n");
    Student list1[3] = {{1,"A",18,9.0},{2,"B",20,7.0},{3,"C",19,8.0}};
    float avg = calculateAverageGPA(list1, 3);
    printf("Test avg list 3 SV (kq=8.0)  : %s\n", (avg>=7.99f && avg<=8.01f)?"PASS":"FAIL");

    float empty = calculateAverageGPA(list1, 0);
    printf("Test avg voi ds rong (kq=-1): %s\n", (empty < 0.0f)?"PASS":"FAIL");
}

void test_sortStudentsByGPA() {
    printf("\n--[Test sortStudentsByGPA]--\n");
    // a. GPA bang nhau
    Student eq[4] = {{1,"A",18,7.0},{2,"B",19,7.0},{3,"C",20,7.0}};
    sortStudentsByGPA(eq, 3);
    int pass = 1;
    for (int i=0; i<2; ++i)
        if (eq[i].gpa != eq[i+1].gpa) pass=0;
    printf("Test GPA bang nhau : %s\n", (pass ? "PASS" : "FAIL"));

    // b. Tang dan san
    Student asc[4] = {{1,"A",18,4.1},{2,"B",19,7.2},{3,"C",20,9.5}};
    sortStudentsByGPA(asc, 3);
    printf("Test tang dan san   : %s\n", (asc[0].gpa==9.5f && asc[1].gpa==7.2f && asc[2].gpa==4.1f)?"PASS":"FAIL");
```

```
    // c. GPA ngau nhien
    Student ran[4] = {{1,"X",20,7.9},{2,"Y",19,5.1},{3,"Z",20,9.3}};
    sortStudentsByGPA(ran, 3);
    printf("Test GPA random     : %s\n", (ran[0].gpa==9.3f && ran[2].gpa==5.1f)?"PASS":"FAIL");
}

// ======= 7. MAIN PROGRAM ========
int main() {
    srand((unsigned)time(NULL));

    // Dieukien khoi tao du lieu mock chi 1 lan (co the bo neu da co file)
    initializeAllClasses();

    // 1. Khoi tao 3 SV san
    Student students[MAX_STUDENTS]={
        {101, "An",    19, 8.8},
        {102, "Binh",  21, 7.9},
        {103, "Cuong", 18, 7.5}
    };
    int n = INIT_STUDENTS;

    printf("\n========= CHAO MUNG DEN PHAN MEM QUAN LY SINH VIEN =========\n");

    // 2. Buoc doc tu file lop (da duoc khoi tao O tren)
    printf("\nChon lop hoc ban muon lam viec (classA.txt, classB.txt, classC.txt):\n");
    n = inputClassAndLoad(students, MAX_STUDENTS);

    // 3. In danh sach sau khi doc file
    printf("\n===== DANH SACH HOC SINH CUUA LOP =====\n");
    printStudentList(students, n);

    // 4. Nhap them tu ban phim
    int added = inputStudents(students, n, MAX_STUDENTS);
    n += added;
    if (added > 0) {
        printf("\nBan vua nhap bo sung %d sinh vien!\n", added);
    }

    // 5. LUu file => Ten file van GIU NGUYEN
    printf("\nLuu danh sach hien tai vao file vua doc lai:\n");
    char fnameSave[MAX_FILENAME_LENGTH+1];
    printf("Nhap lai ten file de ghi (hoac trung file cu): ");
    scanf("%s", fnameSave);
    saveToFile(students, n, fnameSave);
```

```c
    // 6. Test chuc nang tim, thong ke, sap xep

    // -- Tim kiem
    printf("\n====== TEST CASE: Tim kiem sinh vien ======\n");
    printf("Nhap ID can tim: ");
    int idquery;
    scanf("%d", &idquery);
    Student* found = findStudentByID(students, n, idquery);
    if (found) {
        printf(">>> Tim thay SV: ID=%d | Ten=%s | Tuoi=%d | GPA=%.1f\n",
               found->id, found->name, found->age, found->gpa);
    } else {
        printf("Khong tim thay SV co ID = %d hoac ID KHONG HOP LE!\n", idquery);
    }

    // -- Thong ke diem TB
    float avgGPA = calculateAverageGPA(students, n);
    if (avgGPA >= 0)
        printf("\n>>> GPA trung binh cua lop: %.2f\n", avgGPA);
    else
        printf("\n>>> (Khong co du lieu sinh vien -> GPA TB khong tinh duoc)\n");

    // -- Sap xep
    printf("\n====== SAP XEP SINH VIEN THEO GPA GIAM DAN ======\n");
    sortStudentsByGPA(students, n);
    printStudentList(students, n);

    // -- Doc tu file & in raw de minh hoa
    Student fromFile[MAX_STUDENTS];
    int nf = readFromFile(fromFile, MAX_STUDENTS, fnameSave);
    printListFromFile(fromFile, nf);

    // Unit Test
    printf("\n========== KICH BAN TEST CASE UNIT ==========\n");
    test_findStudentByID();
    test_calculateAverageGPA();
    test_sortStudentsByGPA();

    // Ngoai le cho file sai
    printf("\nTest doc tu file sai ten....\n");
    readFromFile(fromFile, MAX_STUDENTS, "filenotexist123.txt");

    printf("\n===== CHUONG TRINH KET THUC! =====\n");
    return 0;
}
```