**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

A red and white rectangular sign with a yellow and red design

AI-generated content may be incorrect.

**BÁO CÁO BÀI TẬP LỚN**

**Đề Tài: Hệ Thống Quản Lý Dự Án**

Sinh viên thực hiện

|  |  |  |
| --- | --- | --- |
| **Họ và Tên** | **MSSV** | **Lớp** |
| **Nguyễn Đức Phúc** | **20234032** | **ET-E9 01 K68** |
| **Lê Minh Triết** | **20234045** | **ET-E9 01 K68** |
| **Tạ Văn Việt** | **20234050** | **ET-E9 01 K68** |
| **Hoàng Quốc Việt** | **20234048** | **ET-E9 01 K68** |

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | TS. Nguyễn Thị Kim Thoa |
| **Bộ môn:** | Cấu Trúc Dữ Liệu và Giải Thuật |
| **Viện:** | Điện-Điện Tử |
| **HÀ NỘI, 6/2025** | |

**MỤC LỤC**

**PHÂN CÔNG NHIỆM VỤ……………………………………………………..4**

**CHƯƠNG I: TỔNG QUAN ĐỀ TÀI...................................................5**

* **1.1. Lý do chọn đề tài................................................................................5**
* **1.2. Mục tiêu của hệ thống........................................................................5**

**CHƯƠNG II: PHÂN TÍCH CHỈ TIÊU..............................................6**

* **2.1. Chỉ tiêu chức năng...............................................................................6**
* **2.2. Chỉ tiêu phi chức năng.........................................................................7**
* **2.3.** Phân tích rủi ro và hạn chế**..................................................................7**

**Chương III: THIẾT KẾ TỔNG THỂ HỆ THỐNG.....................8**

* **3.1.** Kiến trúc hệ thống**.............................................................................8**
* **3.2.** Sơ đồ luồng dữ liệu (Data Flow Diagram)**.......................................8**
* **3.3.** Cơ sở dữ liệu và cấu trúc lưu trữ**.....................................................9**
* **3.4.** Các cấu trúc dữ liệu chủ đạo**............................................................9**

**CHƯƠNG IV: MÔ TẢ CHI TIẾT MODULE NHÂN SỰ.......11**

* **4.1.** Mục tiêu và chức năng của module.**..................................................11**
* **4.2.** Thiết kế dữ liệu...................**..............................................................11**
* **4.2.1.** Định nghĩa struct **Employee** và **EmployeeNode.....................11**
* **4.2.2.** Cấu trúc cây nhị phân (BST) theo ID nhân viên **.. ................12**
* **4.3.** Chức năng chính và thuật toán **......................................................12**
* **4.4.** Mô tả giao diện của module Nhân sự............................**..................20**

**CHƯƠNG V: MÔ TẢ CHI TIẾT MODULE DỰ ÁN...............21**

* **5.1.** Mục tiêu và chức năng của module................................................21
* **5.2.** Thiết kế dữ liệu................................................................................21
* **5.2.1.** Định nghĩa struct Project và ProjectNode.........................21
* **5.2.2.** Cấu trúc danh sách liên kết (LList) cho các dự án.........22
* **5.2.3.** Cây BST phụ (EmployeeNode) chứa nhân sự tham gia dự án....................................................................................................22
* **5.3.** Chức năng chính và thuật toán.......................................................22
* **5.4.** Mô tả giao diện (Console) của module Dự án................................31

**CHƯƠNG VI: TÍCH HỢP VÀ CÁCH VẬN HÀNH HỆ THỐNG...........................................................................................................32**

* **6.1.** Flow chính của chương trình (main.c)............................................32
* **6.2.** Xác định điểm khởi tạo và giải phóng tài nguyên.........................33

**CHƯƠNG VII: KIỂM THỬ VÀ ĐÁNH GIÁ...............................34**

* **7.1.** Mục tiêu kiểm thử............................................................................34
* **7.2.** Phương pháp kiểm thử....................................................................34
* **7.3.** Kịch bản và kết quả kiểm thử.........................................................35
* **7.4.** Phân tích hiệu năng..........................................................................36
* **7.5.** Đánh giá mức độ hoàn thiện và hạn chế hiện tại...........................36

**CHƯƠNG VIII: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN....37**

* **8.1.** Kết luận chung về hệ thống.............................................................37
* **8.2.** Thành tựu đạt được.........................................................................37

**PHÂN CÔNG NHIỆM VỤ**

|  |  |
| --- | --- |
| **Thành viên** | **Nhiệm vụ chính** |
| **Lê Minh Triết – 20234045** | * Lên ý tưởng * Xây dựng cấu trúc dữ liệu BST * Mô tả chức năng vào ra của 2 module chính * Xây dựng các hàm cơ bản liên quan đến nhân viên (insert, tìm kiếm, in ra) |
| **Tạ Văn Việt – 20234050** | * Phân chia công việc * Xây dựng các hàm cơ bản liên quan đến nhân viên (sửa, xóa) * Hoàn thiện struct project và xây dụng hàm thêm nhân sự vào dự án * Làm báo cáo |
| **Hoàng Quốc Việt – 20234048** | * Xây dựng các hàm cơ bản liên quan đến dự án (2 hàm sắp xếp, xóa, thêm dự án) |
| **Nguyễn Đức Phúc – 20234032** | * Xây dựng các hàm cơ bản liên quan đến dự án (in, sửa, tìm kiếm dự án) * Xây dựng giao diện đầu ra, lấy status * Làm báo cáo |

**CHƯƠNG I: TỔNG QUAN ĐỀ TÀI**

**1.1.** Lý do chọn đề tài  
Trong thời đại thông tin phát triển như vũ bão, việc quản trị dự án ngày càng trở nên quan trọng. Các công ty, tổ chức luôn cần một công cụ giúp theo dõi, lưu trữ và xử lý thông tin về nhân sự, cũng như tiến độ và chi phí đầu tư cho từng dự án. Hiện nay, phần lớn các doanh nghiệp vừa và nhỏ vẫn đang quản lý qua Excel hoặc sổ sách truyền thống, dẫn đến khó khăn khi tìm kiếm, báo cáo và dễ nhầm lẫn dữ liệu. Đề tài “Hệ thống Quản trị Dự án (Phần mềm)” được xây dựng với mục đích:

* Tăng cường tính minh bạch trong việc quản lý nhân sự và tiến độ dự án.
* Giảm thiểu lỗi do thao tác thủ công khi quản lý danh sách, tìm kiếm hoặc xóa/sửa thông tin.
* Cung cấp giao diện console đơn giản, dễ sử dụng, phù hợp cho doanh nghiệp vừa và nhỏ, không yêu cầu môi trường đồ họa phức tạp.

**1.2.** Mục tiêu của hệ thống

* Xây dựng một chương trình cho phép người dùng:
  + Quản trị nhân sự: thêm, sửa, xóa, tìm kiếm và lọc nhân viên theo phòng ban.
  + Quản trị dự án: thêm, sửa, xóa, tìm kiếm và lọc dự án theo trạng thái, sắp xếp dự án theo ID hoặc vốn đầu tư, đồng thời gán nhân sự vào từng dự án.
* Sử dụng cấu trúc dữ liệu hiệu quả (cây nhị phân tìm kiếm cho nhân viên, danh sách liên kết cho dự án) để tối ưu về tốc độ tìm kiếm và sắp xếp.
* Cung cấp hướng dẫn cài đặt, cách biên dịch và sử dụng cho người dùng cuối.

**CHƯƠNG II: PHÂN TÍCH CHỈ TIÊU**

**2.1. Chỉ tiêu** chức năng

* Quản trị Nhân sự (Employee Module)
  1. Nhập thông tin nhân viên: ID (kiểu số nguyên), tên (chuỗi kí tự), mức lương (số thực), trạng thái đóng BHXH (có/không), phòng ban (chuỗi kí tự).
  2. Hệ thống tự động phân cấp nhân viên theo ID:
     + ID bắt đầu bằng 1xxx → “Quản lý cấp cao”
     + ID bắt đầu bằng 2xxx → “Quản lý”
     + ID bắt đầu bằng 3xxx → “Nhân viên”
     + Các ID khác → “Không rõ”
  3. Lưu trữ nhân viên vào cây BST (Binary Search Tree)
  4. Hiển thị danh sách nhân viên (phương thức in-order traversal) gồm đầy đủ các thông tin: ID, tên, chức vụ, lương, BHXH, phòng ban.
  5. Tìm kiếm và hiển thị nhân viên theo ID.
  6. Xóa thông tin nhân viên theo ID.
  7. Chỉnh sửa thông tin nhân viên: cho phép thay đổi tên, lương, BHXH, phòng ban; giữ nguyên ID (vì ID là khóa chính).
  8. Lọc và hiển thị nhân viên theo phòng ban bất kỳ (so sánh chuỗi).
* Quản trị Dự án (Project Module)
  1. Nhập thông tin dự án: ID (số nguyên), tên (chuỗi kí tự), vốn điều lệ (số thực), trạng thái (0: Đang phát triển, 1: Đang kiểm thử, 2: Đã bàn giao).
  2. Lưu trữ dự án vào danh sách liên kết đơn (singly linked list), mới nhất được thêm vào đầu danh sách.
  3. Hiển thị danh sách toàn bộ dự án, kèm theo danh sách nhân sự tham gia ở mỗi dự án (dựa trên cây BST con của dự án).
  4. Tìm kiếm dự án theo ID.
  5. Xóa dự án theo ID.
  6. Chỉnh sửa thông tin dự án (tên, vốn điều lệ, trạng thái).
  7. Sắp xếp danh sách dự án theo ID (bubble sort).
  8. Sắp xếp danh sách dự án theo vốn điều lệ (bubble sort).
  9. Lọc và hiển thị dự án theo trạng thái bất kỳ.
  10. Thêm (gán) nhân viên vào dự án: sau khi tìm được dự án theo ID, yêu cầu nhập ID nhân viên, nếu tồn tại trong cây BST chung, chèn vào cây BST con của dự án (chỉ thêm khi nhân viên chưa tồn tại trong cây con).

**2.2. Chỉ tiêu** phi chức năng

* Hiệu năng và hiệu suất
  + Thao tác tìm kiếm, chèn, xóa nhân viên trong BST có độ phức tạp trung bình O(log n) (với cây cân bằng tương đối).
  + Hiển thị toàn bộ danh sách nhân viên (in-order) có độ phức tạp O(n).
  + Thao tác trên danh sách liên kết dự án như thêm/xóa tại đầu danh sách có độ phức tạp O(1), tìm kiếm O(m) với m là số dự án.
  + Sắp xếp bubble sort trên danh sách dự án có độ phức tạp O(m²); do m khá nhỏ (thường dưới vài trăm), vẫn chấp nhận được.
* Khả năng mở rộng
  + Thiết kế tách biệt module nhân sự và dự án, dễ dàng mở rộng thêm tính năng như xuất file, chuyển sang GUI, hoặc tích hợp cơ sở dữ liệu.
  + Cấu trúc dữ liệu linh hoạt, có thể thay thế BST bằng các cấu trúc tự cân bằng (AVL, Red–Black Tree) nếu cần tối ưu cho lượng dữ liệu lớn.
* Giao diện người dùng
  + Ứng dụng chạy trên console (Terminal).
  + Hướng dẫn rõ ràng các lựa chọn và thông báo lỗi khi đầu vào không hợp lệ.

**2.3.** Phân tích rủi ro và hạn chế

* BST không tự cân bằng: nếu dữ liệu nhân viên nhập vào tuần tự tăng (ID sắp xếp sẵn), cây sẽ trở thành dạng đường thẳng, làm độ phức tạp tìm kiếm xuống O(n).
* Bubble sort trên danh sách dự án chưa hiệu quả nếu số lượng dự án quá lớn. Nên cân nhắc thuật toán sắp xếp hiệu quả hơn (quick sort, merge sort) hoặc cấu trúc dữ liệu khác (mảng động + sort).

**Chương III: Thiết kế tổng thể hệ thống**

**3.1.** Kiến trúc hệ thống  
Hệ thống gồm ba thành phần chính:

* Giao diện:
  + Hiển thị menu, nhận nhập lựa chọn từ người dùng, gọi vào các hàm tương ứng của module Nhân sự hoặc Dự án.
* Module Nhân sự (Employee Module):
  + Quản lý dữ liệu nhân viên, bao gồm các hàm thao tác với cây BST.
* Module Dự án (Project Module):
  + Quản lý dữ liệu dự án, bao gồm danh sách liên kết dự án và cây BST con lưu trữ nhân sự tham gia.

**3.2.** Sơ đồ luồng dữ liệu (Data Flow Diagram)

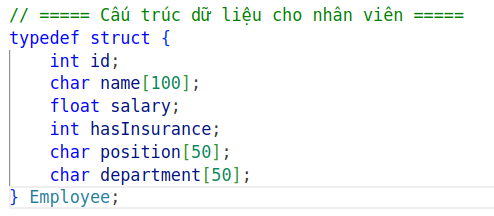
* DFD cấp 0 (Context Diagram)
  + Nguồn (External Entity): Người dùng (User)
  + Hệ thống (System): Hệ thống Quản trị Dự án
  + Luồng dữ liệu chính:
    1. User gửi lệnh (chọn menu, nhập ID, nhập tên, v.v.) → System.
    2. System trả về thông báo, kết quả (Danh sách, thông báo thành công, v.v.) → User.
* DFD cấp 1 (Các chức năng con)
  + Quản trị Nhân sự
    1. Nhập dữ liệu nhân viên → Cập nhật BST
    2. Tìm/Hiển thị/Xóa/Sửa/Lọc nhân viên → Thao tác BST
    3. Kết quả trả về màn hình Console
  + Quản trị Dự án
    1. Nhập dữ liệu dự án → Cập nhật danh sách liên kết
    2. Tìm/Hiển thị/Xóa/Sửa/Sắp xếp/Lọc dự án → Thao tác trên danh sách liên kết
    3. Gán nhân sự → Tương tác với module Nhân sự
    4. Kết quả trả về màn hình Console

**3.3.** Cơ sở dữ liệu và cấu trúc lưu trữ

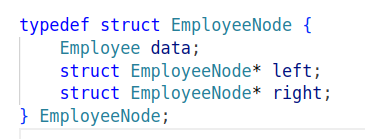
* Lưu trữ tạm thời (In-Memory):
  + BST Nhân sự (Employee BST): Dạng nhị phân; mỗi node chứa struct Employee (id, tên, lương, hasInsurance, position, department) và 2 con left/right.
  + Danh sách liên kết đơn (Project Linked List): Mỗi node chứa struct Project (id, tên, capital, status, pointer employees đến cây BST con) và next.
  + BST con của mỗi dự án (Project’s Employee BST): Giống BST chính, lưu Employee nhưng chỉ các nhân viên đã được gán.

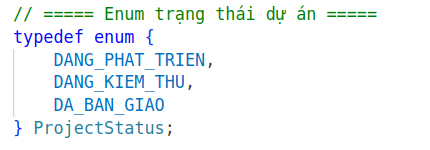
**3.4.** Các cấu trúc dữ liệu chủ đạo

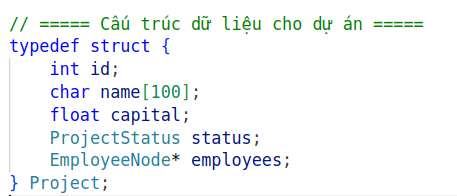
* Employee (struct):



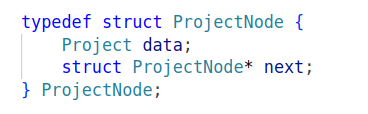
* EmployeeNode **(cây BST cho nhân sự):**



* ProjectStatus **(enum trạng thái dự án):**
* ****
* Project **(struct):**

****

* ProjectNode **(danh sách liên kết dự án):**

****

Chương IV. Mô tả chi tiết module Nhân sự

**4.1.** Mục tiêu và chức năng của module

* Cung cấp các hàm để quản lý toàn bộ thông tin nhân viên trong hệ thống dưới dạng cây nhị phân tìm kiếm.
* Đảm bảo các thao tác: thêm, tìm, xóa, in-order, chỉnh sửa, lọc theo phòng ban được thực hiện hiệu quả.

**4.2.** Thiết kế dữ liệu **4.2.1.** Định nghĩa struct **Employee** và **EmployeeNode**

* **Employee (Struct):**

A screenshot of a computer code

AI-generated content may be incorrect.

* Employee lưu trữ tất cả trường thông tin của nhân viên:
  + id (int): Mã số duy nhất, khóa chính.
  + name[100] (char[]): Tên đầy đủ.
  + salary (float): Mức lương.
  + hasInsurance (int): 1 nếu đã đóng BHXH, 0 nếu chưa.
  + position[50] (char[]): Chức vụ phân cấp tự động từ ID.
  + department[50] (char[]): Tên phòng ban.
* **EmployeeNode (Struct):**

A screenshot of a computer code

AI-generated content may be incorrect.

* EmployeeNode đại diện một node trong cây BST:
  + data (Employee): Thông tin nhân viên.
  + left (EmployeeNode\*): Con trái.
  + right (EmployeeNode\*): Con phải.

**4.2.2.** Cấu trúc cây nhị phân (BST) theo ID nhân viên

* Tiêu chí sắp xếp (Key): data.id của mỗi EmployeeNode.
* Khi chèn: so sánh emp.id với root->data.id; nếu nhỏ hơn, đi vào left; lớn hơn, đi vào right.
* Khi tìm kiếm hoặc xóa, cũng dựa vào so sánh ID để di chuyển xuống cây.

**4.3.** Chức năng chính và thuật toán

* Hàm getPositionFromID(int id):

A computer screen shot of a code

AI-generated content may be incorrect.

* + **Mục đích**: Từ 4 chữ số đầu (prefix = id/1000), trả về chức vụ tương ứng:
    - 1 → “Quản lý cấp cao”
    - 2 → “Quản lý”
    - 3 → “Nhân viên”
    - Khác → “Không rõ”
  + Cài đặt sử dụng switch cho prefix.
* Hàm inputEmployee()

A screenshot of a computer code

AI-generated content may be incorrect.

* + **Mục đích:** Hiển thị lời nhắc, đọc thông tin từ bàn phím, gán chức vụ dựa vào ID, trả về struct Employee.
  + **Thuật toán:**
    - Yêu cầu nhập id.
    - Nhập name (dòng), salary, hasInsurance (1 hoặc 0), department (dòng).
    - Gọi getPositionFromID(emp.id) để gán emp.position.
    - Return emp.
* Hàm insertEmployee(EmployeeNode\* root, Employee emp)

A computer screen shot of text

AI-generated content may be incorrect.

* + **Mục đích:** Chèn một node chứa emp vào cây BST; nếu tồn tại ID trùng, thông báo lỗi và không chèn.
  + **Thuật toán (đệ quy):**
    - Nếu root == NULL: tạo node mới, gán data = emp, left = right = NULL, trả về node mới.
    - Ngược lại, gọi findEmployeeByID(root, emp.id). Nếu không NULL → ID đã tồn tại → in thông báo “ID nhân viên đã tồn tại” → return root.
    - So sánh emp.id < root->data.id:
      * Nếu đúng: root->left = insertEmployee(root->left, emp).
      * Nếu sai: root->right = insertEmployee(root->right, emp).
    - Return root.
* Hàm findEmployeeByID(EmployeeNode\* root, int id)

A screen shot of a computer code

AI-generated content may be incorrect.

* + **Mục đích:** Tìm và trả về con trỏ đến node có data.id == id; nếu không tìm thấy, trả về NULL.
  + **Thuật toán (đệ quy):**
    - Nếu root == NULL → trả về NULL.
    - Nếu id == root->data.id → trả về root.
    - Nếu id < root->data.id: đệ quy bên trái; ngược lại, đệ quy bên phải.
* Hàm deleteEmployeeByID(EmployeeNode\* root, int id)

A screenshot of a computer program

AI-generated content may be incorrect.

* + **Mục đích:** Xóa node có id khỏi cây BST, sau đó trả về cây đã chỉnh sửa.
  + **Thuật toán (đệ quy, tiêu chuẩn xóa node BST):**
    - Nếu root == NULL → return NULL.
    - Nếu id < root->data.id → root->left = deleteEmployeeByID(root->left, id).
    - Nếu id > root->data.id → root->right = deleteEmployeeByID(root->right, id).
    - Nếu id == root->data.id:
      * **Trường hợp 1: Node chỉ có một con hoặc không con:**
        + Nếu root->left == NULL: lưu temp = root->right, free(root), return temp.
        + Nếu root->right == NULL: lưu temp = root->left, free(root), return temp.
      * **Trường hợp 2: Node có hai con:**
        + Tìm node successor (phần tử nhỏ nhất bên phải): đi temp = root->right; while (temp->left) temp = temp->left;.
        + Gán root->data = temp->data.
        + Gọi root->right = deleteEmployeeByID(root->right, temp->data.id).
    - Return root.
* Hàm printEmployees(EmployeeNode\* root)

A screenshot of a computer code

AI-generated content may be incorrect.

* + **Mục đích:** In toàn bộ nhân viên theo thứ tự tăng dần ID (in-order traversal).
  + **Thuật toán (đệ quy):**
    - Nếu root == NULL → return.
    - printEmployees(root->left).
    - In thông tin của root→data:
    - printEmployees(root->right).
* Hàm printEmployeesByDepartment(EmployeeNode\* root, const char\* department)

A computer code with text

AI-generated content may be incorrect.

* **Mục đích:** Duyệt in-order, nhưng chỉ in khi strcmp(root->data.department, department) == 0.
* **Thuật toán (đệ quy):**
  1. Nếu root == NULL → return.
  2. printEmployeesByDepartment(root->left, department).
  3. Nếu strcmp(root->data.department, department) == 0: in thông tin như ở hàm printEmployees.
  4. printEmployeesByDepartment(root->right, department).
* Hàm editEmployee(EmployeeNode\* root, int id)

A computer code with many text

AI-generated content may be incorrect.

* **Mục đích:** Cho phép chỉnh sửa tên, lương, BHXH, phòng ban của nhân viên có id.
* **Thuật toán:**
  1. Gọi findEmployeeByID(root, id). Nếu NULL → in “Không tìm thấy nhân sự với ID đã nhập” → return.
  2. Nếu tìm thấy: hiển thị lời nhắc nhập “Tên mới”, “Lương mới”, “Đã đóng BHXH (1/0)”, “Phòng ban mới”.
  3. Gán các trường tương ứng; giữ nguyên id (không đổi).
  4. Gọi lại strcpy(empNode->data.position, getPositionFromID(empNode->data.id)) để cập nhật chức vụ (thực tế ID không đổi nên giá trị position cũ vẫn đúng, tuy nhiên vẫn gọi để đảm bảo tính nhất quán).
  5. In “Đã sửa thông tin nhân sự.”

**4.4.** Mô tả giao diện của module Nhân sự

* Menu chính của Quản trị Nhân sự ( hàm printEmployeeMenuFrame() ):

A screenshot of a computer

AI-generated content may be incorrect.

Chương V. Mô tả chi tiết module Dự án

**5.1.** Mục tiêu và chức năng của module

* Quản lý toàn bộ thông tin dự án, cho phép thao tác thêm, xóa, tìm kiếm, hiển thị, sắp xếp và lọc theo trạng thái.
* Cho phép gán nhân viên (từ cây BST chung) vào mỗi dự án; lưu dưới dạng cây BST con để dễ dàng hiển thị và tìm kiếm danh sách nhân sự tham gia.

**5.2.** Thiết kế dữ liệu **5.2.1.** Định nghĩa struct Project và ProjectNode

* **Project (struct):**

A screenshot of a computer code

AI-generated content may be incorrect.

* Project gồm:
  + id (int): Mã dự án.
  + name[100] (char[]): Tên dự án.
  + capital (float): Vốn điều lệ (invested capital).
  + status (ProjectStatus): Trạng thái hiện tại (0, 1, 2).
  + employees (EmployeeNode\*): Con trỏ đến cây BST con lưu nhân viên tham gia dự án. Ban đầu NULL.
* **ProjectNode (struct):**

**A white background with blue text

AI-generated content may be incorrect.**

* ProjectNode (danh sách liên kết):
  + data (Project): Thông tin dự án.
  + next (ProjectNode\*): Node kế tiếp.

**5.2.2.** Cấu trúc danh sách liên kết (Linked List) cho các dự án

* Sử dụng danh sách liên kết đơn cho phép:
  + Thêm mới ở đầu danh sách (thao tác O(1)).
  + Lặp qua toàn bộ danh sách để hiển thị, tìm kiếm, xóa, sắp xếp.

**5.2.3.** Cây BST phụ (EmployeeNode) chứa nhân sự tham gia dự án

* Cây này cũng dùng struct EmployeeNode, thừa kế cấu trúc từ module Nhân sự, nhưng chỉ lưu các Employee đã được gán.
* Khi gán: nếu findEmployeeByID(rootCon, emp.id) != NULL → nhân viên đã có trong dự án → in “Nhân viên đã có trong dự án!”. Ngược lại, chèn node mới.

**5.3.** Chức năng chính và thuật toán

* Hàm getStatusName(ProjectStatus status)

A screen shot of a computer code

AI-generated content may be incorrect.

* + **Mục đích:** Chuyển ProjectStatus thành chuỗi tương ứng:
    - DANG\_PHAT\_TRIEN → “Dang phat trien”
    - DANG\_KIEM\_THU → “Dang kiem thu”
    - DA\_BAN\_GIAO → “Da ban giao”
    - Mặc định → “Khong xac dinh”
* Hàm addProject(ProjectNode\* head)

A screen shot of a computer code

AI-generated content may be incorrect.

* + **Mục đích:** Tạo node dự án mới, điền thông tin, chèn vào đầu danh sách.
  + **Thuật toán:**
    - Cấp phát newNode = (ProjectNode\*)malloc(sizeof(ProjectNode)).
    - Nhập newNode->data.id, newNode->data.name, newNode->data.capital.
    - Nhập stt (int), gán newNode->data.status = (ProjectStatus)stt.
    - newNode->data.employees = NULL.
    - newNode->next = head.
    - Return newNode (trở thành head mới).
* Hàm findProjectByID(ProjectNode\* head, int projectID)
* A screen shot of a computer code

  AI-generated content may be incorrect.
  + **Mục đích:** Trả về con trỏ đến node dự án có data.id == projectID.
  + **Thuật toán:**
    - Duyệt while (head): nếu head->data.id == projectID, return head. Ngược lại, head = head->next.
    - Nếu duyệt hết chưa tìm, return NULL.
* Hàm deleteProjectByID(ProjectNode\* head, int id)

A screen shot of a computer code

AI-generated content may be incorrect.

* + **Mục đích:** Xóa node dự án đầu tiên có data.id == id.
  + **Thuật toán:**
    - Khởi tạo prev = NULL, curr = head.
    - Duyệt while (curr):
      * Nếu curr->data.id == id:
        + Nếu prev != NULL: prev->next = curr->next.
        + Ngược lại (curr là head): head = curr->next.
        + (Chú ý): Nếu dự án có cây nhân sự con (curr->data.employees != NULL), cần giải phóng toàn bộ cây đó trước khi free(curr). (Trong mã mẫu, phần giải phóng BST con được để chú thích “TODO”.)
        + free(curr), in “Đã xóa dự án.”, return head.
      * Ngược lại, prev = curr; curr = curr->next.
    - Nếu kết thúc vòng lặp mà không tìm thấy, in “Không tìm thấy dự án.”, return head.
* Hàm printProjectEmployees(EmployeeNode\* root)

A computer code with text

AI-generated content may be incorrect.

* + **Mục đích:** In-order traversal cây BST con, hiển thị danh sách nhân viên tham gia.
  + **Thuật toán (đệ quy):** nếu root == NULL → return; else in-order bên trái, in thông tin node, in-order bên phải.
* Hàm printProjects(ProjectNode\* head)

A screenshot of a computer code

AI-generated content may be incorrect.

* **Mục đích:** Duyệt toàn bộ danh sách liên kết, in chi tiết từng dự án và danh sách nhân sự (nếu có).
* **Thuật toán:**
  1. Nếu head == NULL: in “Danh sách dự án rỗng!” và return.
  2. Duyệt while (head): ( chụp ảnh đoạn code bên dưới )
     + **ID: %d | Ten: %s | Von: %.2f | Trang thai: %s**
     + **Danh sach nhan su:**
     + **Nếu** head->data.employees != NULL**: gọi** printProjectEmployees(head->data.employees)**.**
       - Ngược lại: in “ (không có nhân sự)”.
       - head = head->next.
* Hàm addEmployeeToProject(EmployeeNode\* root, Employee emp)

A computer screen shot of text

AI-generated content may be incorrect.

* + **Mục đích:** Thêm một Employee (đã lấy từ BST chung) vào cây BST con của dự án.
  + **Thuật toán (tương tự insertEmployee):**
    1. Nếu findEmployeeByID(root, emp.id) != NULL: in “Nhân viên đã có trong dự án!” → return root.
    2. Nếu root == NULL: tạo node mới chứa emp, trả về node.
    3. Nếu emp.id < root->data.id: root->left = addEmployeeToProject(root->left, emp).
    4. Ngược lại: root->right = addEmployeeToProject(root->right, emp).
    5. Return root.
* Hàm bubbleSortProjectsByID(ProjectNode\* head)

A screenshot of a computer code

AI-generated content may be incorrect.

* + **Mục đích:** Sắp xếp tăng dần theo data.id (thuật toán bubble sort trên danh sách liên kết).
  + **Thuật toán:**
    1. Kiểm tra nếu head == NULL hoặc chỉ có 1 phần tử → return.
    2. Sử dụng hai vòng lặp:
       - lptr = NULL; swapped = 0; do { ... } while (swapped);
       - Duyệt ptr1 từ head đến node trước lptr:
         * Nếu ptr1->data.id > ptr1->next->data.id: hoán đổi toàn bộ struct Project giữa hai node, swapped = 1.
         * ptr1 = ptr1->next.
       - Sau một lần duyệt, cập nhật lptr = ptr1 (node cuối đã sắp xếp).
* Hàm bubbleSortProjectsByCapital(ProjectNode\* head)

A computer code with text

AI-generated content may be incorrect.

* + Tương tự bubbleSortProjectsByID, nhưng điều kiện: ptr1->data.capital > ptr1->next->data.capital.
* Hàm editProject(ProjectNode\* head, int id)

A computer code with text

AI-generated content may be incorrect.

* + **Mục đích:** Chỉnh sửa thông tin name, capital, status của dự án có ID tương ứng.
  + **Thuật toán:**
    1. Gọi findProjectByID(head, id). Nếu NULL: in “Không tìm thấy dự án với ID đã nhập.”, return.
    2. Nếu tìm thấy: in “Nhập thông tin mới cho dự án (ID=<id>):”, sau đó:
       - Nhập tên mới, scanf(" %[^\n]", prj->data.name);
       - Nhập vốn mới, scanf("%f", &prj->data.capital);
       - Nhập trạng thái mới (0/1/2), gán prj->data.status = (ProjectStatus)stt;
       - In “Đã sửa thông tin dự án.”
* Hàm printProjectsByStatus(ProjectNode\* head, ProjectStatus status)

A computer screen shot of a code

AI-generated content may be incorrect.

* + **Mục đích:** Duyệt danh sách liên kết, in các dự án có data.status == status.
  + **Thuật toán:**
    1. Khai báo found = 0. In tiêu đề “Danh sách dự án có trạng thái: <getStatusName(status)>”.
    2. Duyệt while (head):
       - Nếu head->data.status == status: in “ID: %d | Ten: %s | Von: %.2f”, found = 1.
       - head = head->next.
    3. Nếu found == 0: in “Không tìm thấy dự án nào có trạng thái này.”

**5.4.** Mô tả giao diện (Console) của module Dự án

* Menu chính của Quản trị Dự án ( hàm printProjectMenuFrame() )

A screenshot of a cell phone

AI-generated content may be incorrect.

Chương VI. Tích hợp và cách vận hành hệ thống

**6.1.** Flow chính của chương trình (main.c)

* Khi khởi động, các con trỏ empTree (cây BST nhân sự chung) và prjList (danh sách liên kết dự án) đều được khởi tạo bằng NULL.
* Vòng lặp chính (do…while) hiển thị Menu chính (hàm printMainMenuFrame()):

A screenshot of a computer

AI-generated content may be incorrect.

* Đọc lựa chọn mainChoice (int). Nếu nhập sai định dạng, vòng lặp vẫn tiếp tục (đã xử lý while (getchar()!='\n')).
  1. Nếu mainChoice == 1:
     + Vào vòng lặp con cho Menu Nhân sự (printEmployeeMenuFrame()) cho đến khi người dùng chọn 0 (Quay lại).
     + Mỗi lựa chọn (1–6) được xử lý tương ứng, gọi hàm trong module Nhân sự.
  2. Nếu mainChoice == 2:
     + Vào vòng lặp con cho Menu Dự án (printProjectMenuFrame()), cho đến khi chọn 0.
     + Mỗi lựa chọn (1–9) gọi hàm trong module Dự án.
  3. Nếu mainChoice == 0:
     + In khung “ĐANG THOÁT CHƯƠNG TRÌNH” rồi break ra khỏi vòng lặp.
  4. Nếu mainChoice không hợp lệ: in “Lựa chọn không hợp lệ. Vui lòng chọn lại.” và quay lại.

**6.2.** Xác định điểm khởi tạo và giải phóng tài nguyên

* Khởi tạo:
  + empTree = NULL; prjList = NULL.
* Giải phóng khi thoát:
  + Trong mã hiện tại, khi người dùng chọn “0. Thoát”, chương trình return 0; và kết thúc, nhưng chưa giải phóng toàn bộ bộ nhớ đã cấp phát cho cây BST và danh sách liên kết. Để tránh rò rỉ bộ nhớ, cần thêm:
    - Hàm đệ quy giải phóng toàn bộ cây BST nhân sự (freeEmployeeTree(EmployeeNode\* root)), xử lý hai nhánh con rồi free(root).
    - Trước khi free(curr) trong deleteProjectByID, phải giải phóng BST con của dự án đó.
    - Khi thoát hoàn toàn, duyệt prjList, gọi giải phóng BST con và free(ProjectNode\*).

Chương VII. Kiểm thử và đánh giá

**7.1.** Mục tiêu kiểm thử

* Đảm bảo toàn bộ các chức năng của hai module Nhân sự và Dự án hoạt động đúng như yêu cầu.
* Phát hiện và xử lý ngoại lệ khi người dùng nhập dữ liệu không hợp lệ.
* Đánh giá hiệu năng của các thao tác cơ bản (tìm, chèn, xóa) với lượng dữ liệu thử nghiệm tương đối lớn.

**7.2.** Phương pháp kiểm thử

* Kiểm thử đơn vị (Unit Test): Viết các test case cho từng hàm quan trọng:
  + insertEmployee(), findEmployeeByID(), deleteEmployeeByID(), printEmployees(), printEmployeesByDepartment(), editEmployee().
  + addProject(), findProjectByID(), deleteProjectByID(), printProjects(), addEmployeeToProject(), bubbleSortProjectsByID(), bubbleSortProjectsByCapital(), editProject(), printProjectsByStatus().

**7.3.** Kịch bản và kết quả kiểm thử

****

**7.4.** Phân tích hiệu năng

* Nhân sự:
  + Với n = 10.000 nhân viên, nếu cây BST cân bằng gần như hoàn hảo, tìm kiếm/chèn/xóa có độ phức tạp trung bình ~ O(log 10.000) ≈ 14.
  + In-order toàn bộ 10.000 nhân viên có O(n) = O(10.000) (cho phép hiển thị).
  + Nếu dữ liệu nhập đã sắp xếp (ví dụ ID tăng dần), cây có thể chuyển thành cây “gãy” khiến độ phức tạp giảm xuống O(n) cho tìm/chèn. Để khắc phục, có thể cân nhắc sử dụng BST tự cân bằng.
* Dự án:
  + Với m = 1.000 dự án, tìm kiếm dự án theo ID (duyệt danh sách liên kết) mất O(m) = O(1.000). Đối với ứng dụng doanh nghiệp nhỏ chứa dưới 100 dự án, điều này chấp nhận được.
  + Sắp xếp bubble sort có độ phức tạp O(m²) = O(1.000²) = 1.000.000 bước so sánh/hoán đổi; trong thực tế, với m ≈ 100, chỉ tốn khoảng 10.000 bước, tốc độ vẫn nhanh.

**7.5.** Đánh giá mức độ hoàn thiện và hạn chế hiện tại

* Hoàn thiện:
  + Đầy đủ chức năng theo yêu cầu: quản lý nhân sự, quản lý dự án, gán nhân sự.
  + Giao diện console rõ ràng, có menu điều hướng và thông báo lỗi cơ bản.
  + Sử dụng cấu trúc dữ liệu BST và Linked List phù hợp, đơn giản, dễ hiểu.

Chương VIII. Kết luận và hướng phát triển

**8.1.** Kết luận chung về hệ thống  
Hệ thống Quản trị Dự án (Phần mềm) hoàn thiện hai module: Nhân sự (Employee) và Dự án (Project). Các chức năng cơ bản như thêm, sửa, xóa, tìm kiếm, hiển thị và lọc đều hoạt động đầy đủ và chính xác. Cấu trúc dữ liệu BST cho phép quản lý nhân viên hiệu quả, trong khi danh sách liên kết kết hợp cây BST con giúp hiển thị danh sách nhân sự cho từng dự án rõ ràng. Giao diện console đơn giản, dễ sử dụng, phù hợp cho doanh nghiệp vừa và nhỏ không cần đến giải pháp phức tạp.

**8.2.** Thành tựu đạt được

* Hoàn thành tất cả yêu cầu chức năng đề ra:
  + Quản lý nhân sự: thêm, xóa, tìm, sửa, lọc theo phòng ban.
  + Quản lý dự án: thêm, xóa, tìm, sửa, sắp xếp, lọc theo trạng thái.
  + Gán nhân sự vào dự án, hiển thị danh sách nhân sự tham gia.
* Tổ chức cấu trúc mã nguồn rõ ràng, tách biệt theo module, dễ bảo trì và mở rộng.
* Kiểm thử đầy đủ với lượng dữ liệu mô phỏng, đảm bảo các hàm chính hoạt động đúng.