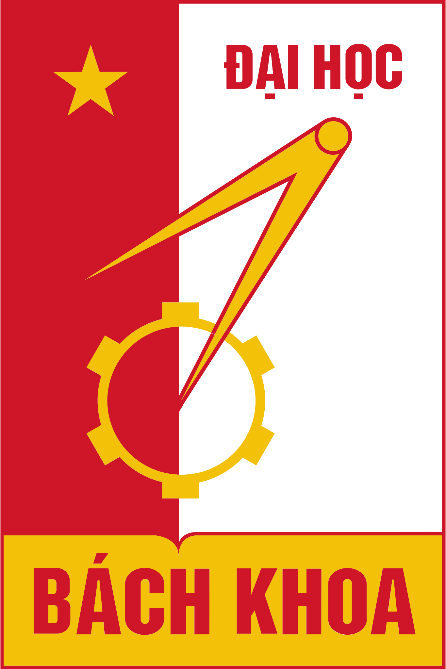
**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**NGÀNH: ĐIỆN TỬ VIỄN THÔNG**

--------------\*--------------



**BÁO CÁO MÔN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

(Mã học phần: ET2100)

**ĐỀ TÀI: HỆ THỐNG MÔ PHỎNG VÀ QUẢN LÝ TỆP TIN**

GVHD: PGS.TS Trần Thị Thanh Hải

Sinh viên thực hiện:

Hoàng Quang Vinh - 20224201

Phan Thanh Bình - 20223683

Phạm Trường Giang - 20223944

Mã lớp: 154832

**Mục lục**

[LỜI MỞ ĐẦU 4](#_Toc186240578)

[I: NỘI DUNG ĐỀ TÀI 5](#_Toc186240579)

[**1. Giới thiệu đề tài** 5](#_Toc186240580)

[**2. Mục tiêu cho đề tài** 5](#_Toc186240582)

[II: PHÂN TÍCH BÀI TOÁN: 5](#_Toc186240583)

[III: PHÂN CÔNG NHIỆM VỤ: 5](#_Toc186240584)

[IV: QUÁ TRÌNH THIẾT KẾ VÀ XÂY DỰNG 6](#_Toc186240585)

[**1. Khai báo thư viện** 6](#_Toc186240586)

[**2. Class Node** 6](#_Toc186240587)

[**3. Hàm nameExists: Kiểm tra tên đã tồn tại trong danh sách con của một nút** 6](#_Toc186240588)

[**4. Hàm addNode: Thêm một thư mục hoặc tệp vào cây** 7](#_Toc186240589)

[**5. Hàm deleteNode: xóa thư mục hoặc tệp khỏi cây** 9](#_Toc186240590)

[**6. Hàm renameNode: đổi tên một thư mục hoặc tệp trong cây** 10](#_Toc186240591)

[**7. Hàm displayTree: hiển thị cấu trúc cây DFS, hỗ trợ phân trang** 11](#_Toc186240592)

[**8. Hàm safeInputInt: Kiểm tra nhập liệu an toàn** 12](#_Toc186240593)

[**9. Hàm main** 13](#_Toc186240594)

[V: KẾT QUẢ 20](#_Toc186240595)

[VI: KẾT LUẬN 23](#_Toc186240596)

[VII: LỜI KẾT 24](#_Toc186240597)

# **LỜI MỞ ĐẦU**

Đối với môn Cấu trúc dữ liệu & Giải thuật, chúng em đã được tiếp thu những kiến thức mới về các giải thuật dựa trên nền tảng ngôn ngữ C/C++, học và áp dụng cấu trúc cây vào trong bài tập lớn.

Qua đề tài xây dựng hệ thống mô phỏng và quản lý tệp tin mà cô giao, chúng em đã có cơ hội tương tác, tìm hiểu cách xử lý các thuật toán để tạo nên giao diện mong muốn, cũng là để tạo cho chúng em làm quen dần đến môn học về sau.

Chúng em xin gửi lời cảm ơn chân thành đến cô Trần Thị Thanh Hải đã nhiệt tình hướng dẫn, góp ý cũng như giải đáp thắc mắc cho nhóm trong suốt quá trình làm đề tài, và giúp chúng em hiểu thêm về đề tài cũng như cách tiếp cận một bài toán được đưa ra.

Sản phẩm của chúng em đã hoàn thành nhưng vẫn còn sự sai sót, chúng em rất mong nhận được những lời nhận xét, đánh giá chân thành từ cô. Nhóm chúng em xin chân thành cảm ơn.

# **I: NỘI DUNG ĐỀ TÀI**

1. **Giới thiệu đề tài**

Hệ thống mô phỏng và quản lý tệp tin là hệ thống giúp người quản lý có thể thu hẹp không gian lưu trữ, tránh thất lạc dữ liệu, xử lý thông tin nhanh, tra cứu tìm kiếm, truy xuất dữ liệu một cách nhanh chóng với độ chính xác cao, tiết kiệm được nhiều thời gian và công sức cho người quản lý

1. **Mục tiêu cho đề tài**

Hiểu rõ và nắm bắt được quy trình quản lý thư mục và tệp tin. Bao gồm việc cập nhật dữ liệu, thêm tệp/ thư mục, xóa tệp/ thư mục, duyệt thư mục theo chiều sâu (DFS), hiển thị cấu trúc cây của thư mục dưới dạng cây hoặc danh sách,…

# **II: PHÂN TÍCH BÀI TOÁN:**

**1. Phân tích hệ thống**

Hệ thống mô phỏng và quản lý tệp tin là hệ thống là hệ thống cho phép truy cập, thêm và xóa tệp tin và hiển thị cấu trúc cây thư mục dưới dạng cây hoặc danh sách

**2. Các chức năng cơ bản của hệ thống bao gồm:**

- Thêm thư mục hoặc tệp tin

- Xóa thư mục hoặc tệp tin

- Hiển thị cấu trúc cây thư mục

# **III: PHÂN CÔNG NHIỆM VỤ:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | NHIỆM VỤ | Giang | Vinh | Bình |
| 1 | Lên ý tưởng | x | x | x |
| 2 | Code | x | x | x |
| 3 | Làm báo cáo | x | x | x |
| 4 | Làm slide | x | x | x |

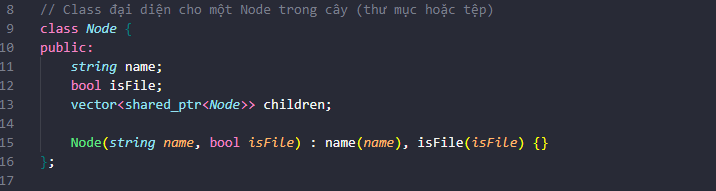
# **IV: QUÁ TRÌNH THIẾT KẾ VÀ XÂY DỰNG**

## **Khai báo thư viện**



Hình 4.1 Khai báo thư viện

## **Class Node**



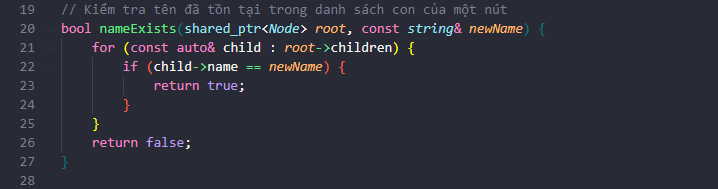
Hình 4.2. Class Node

Chức năng của Class Node: đại diện cho một nút trong cây, có thể là một thư mục hoặc tệp tin

- IsFile: biến bool để xác định nút là tệp tin (true) hay là thư mục (fales)

- children: Danh sách các con của nút hiện tại được lưu bằng “ shared\_ptr<node> “

## **Hàm nameExists: Kiểm tra tên đã tồn tại trong danh sách con của một nút**



Hình 4.3: Hàm nameExists

1. **Chức năng**: Kiểm tra xem tên cần kiểm tra (newName) có tồn tại trong danh sách con của một nút (thư mục cha) không.
2. **Cách hoạt động**:

+) Duyệt toàn bộ các nút con (children) của nút gốc (root).

+) So sánh child->name với newName.

+) Nếu tìm thấy, trả về true. Nếu không, trả về false.

1. Giải thuật:

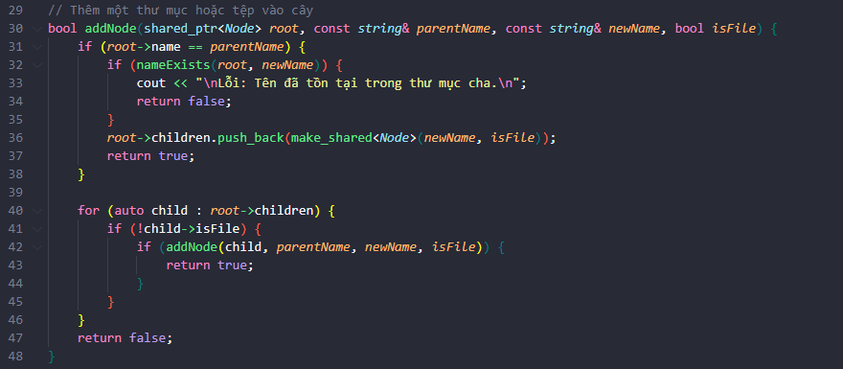
1. Nhận vào một nút gốc và tên cần kiểm tra.

2. Duyệt qua từng nút con của nút gốc.

3. Nếu có nút con nào có tên trùng với tên cần kiểm tra, trả về true.

4. Nếu không tìm thấy, trả về false.

## **Hàm addNode: Thêm một thư mục hoặc tệp vào cây**



Hình 4.4: Hàm addNode

1. **Chức năng:** Thêm một thư mục hoặc tệp tin mới vào cây.
2. **Cách hoạt động:**

- Kiểm tra nút cha: Nếu tên nút gốc (root->name) khớp với parentName:

+) Kiểm tra trùng lặp tên bằng hàm nameExists.

+) Nếu không trùng, thêm nút mới vào danh sách children.

- Duyệt đệ quy:

+) Nếu nút cha chưa được tìm thấy, tiếp tục tìm trong các nút con (đệ quy).

+) Bỏ qua các tệp tin vì chúng không thể chứa nút con.

1. **Giải thuật:**

1. Nhận vào nút gốc, tên của nút cha, tên mới và kiểu (thư mục hoặc tệp tin).

2. Nếu tên của nút gốc trùng với tên nút cha:

+) Kiểm tra xem tên mới đã tồn tại trong danh sách con bằng cách gọi nameExists.

+) Nếu tồn tại, in thông báo lỗi và trả về false.

+) Nếu không, tạo một nút mới và thêm vào danh sách con, trả về true.

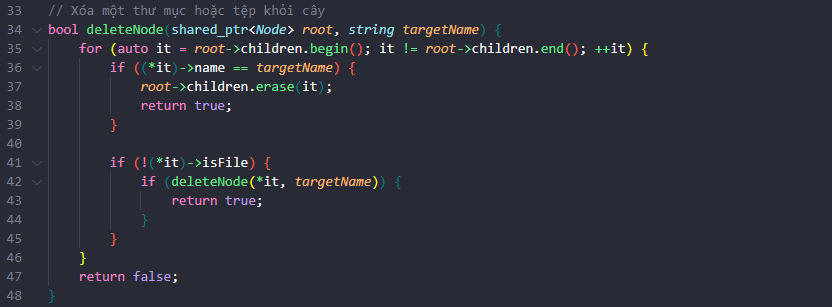
3. Nếu không, duyệt qua các nút con:

+) Nếu nút con không phải là tệp tin, gọi đệ quy addNode cho nút con.

+) Nếu thêm thành công, trả về true.

4. Nếu không tìm thấy nút cha, trả về false.

## **Hàm deleteNode: xóa thư mục hoặc tệp khỏi cây**



Hình 4.5: Hàm xóa thư viện hoặc tệp khỏi cây

1. **Chức năng:** Xóa một thư mục hoặc tệp tin khỏi cây.

**b) Cách hoạt động:**

- Duyệt qua danh sách children của nút hiện tại bằng iterator.

- Nếu tìm thấy nút có tên trùng với targetName:

+ Sử dụng erase để xóa nút khỏi danh sách children.

+ Trả về true (xóa thành công).

- Nếu nút hiện tại là thư mục:

=> Tiếp tục gọi đệ quy trên các nút con để tìm và xóa.

- Nếu không tìm thấy nút cần xóa, trả về false.

**c) Giải thuật**

1. Nhận tham số: root (nút gốc), targetName (tên của thư mục hoặc tệp cần xóa).

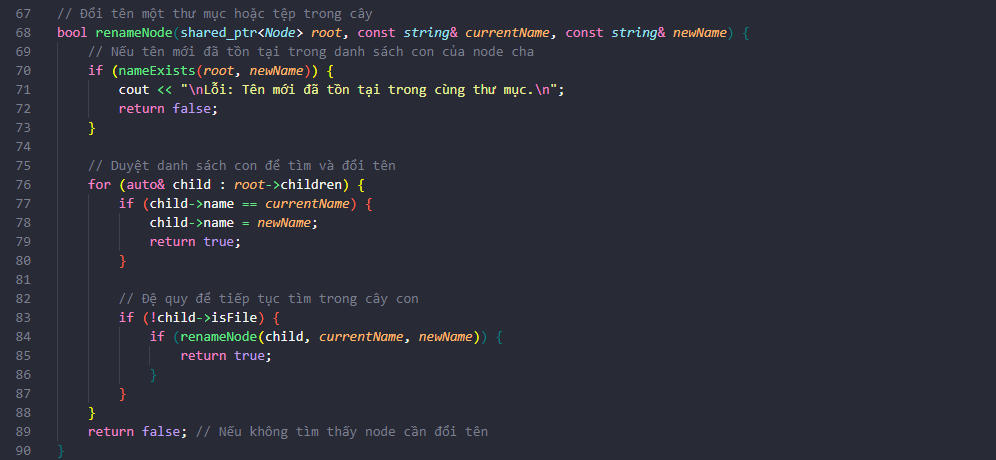
2. Duyệt qua danh sách children của root:

- Nếu tên của nút con bằng với targetName, xóa nút này khỏi danh sách children và trả về true.

- Nếu nút con là thư mục (không phải tệp), gọi đệ quy deleteNode trên nút con

3. Nếu không tìm thấy nút cần xóa trong danh sách con, trả về false.

## **Hàm renameNode: đổi tên một thư mục hoặc tệp trong cây**



Hình 4.6: Hàm renameNode

1. **Chức năng:** Đổi tên một nút trong cây.
2. **Cách hoạt động:**

- Kiểm tra trùng lặp tên mới:

+) Sử dụng nameExists để kiểm tra tên mới có trùng trong thư mục cha không.

+) Nếu có, trả về lỗi.

- Tìm nút cần đổi tên:

+) Duyệt qua các nút con của nút gốc.

+) Nếu tìm thấy nút cần đổi tên (currentName), cập nhật thành newName.

1. **Đệ quy:** Nếu nút không phải tệp tin, tiếp tục tìm trong các thư mục con.
2. **Giải thuật:**

1. Nhận vào nút gốc, tên hiện tại và tên mới.

2. Kiểm tra xem tên mới đã tồn tại trong danh sách con bằng cách gọi nameExists. Nếu có, in thông báo lỗi và trả về false.

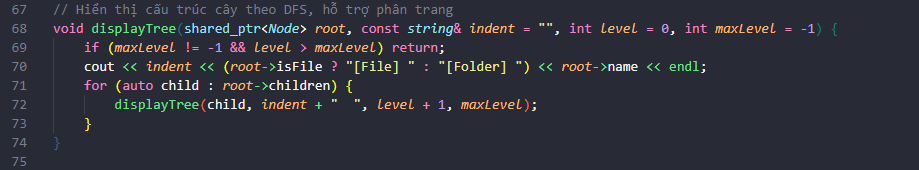
3. Duyệt qua danh sách các nút con:

- Nếu tên của nút con trùng với tên hiện tại, đổi tên nó với tên mới và trả về true.

- Nếu không, nếu nút con không phải là tệp tin, gọi đệ quy renameNode cho nút con.

4. Nếu không tìm thấy nút cần đổi tên, trả về false.

## **Hàm displayTree: hiển thị cấu trúc cây DFS, hỗ trợ phân trang**



Hình 4.7: Hàm hiển thị cấu trúc cây

1. **Chức năng:** Hiển thị cấu trúc cây thư mục theo dạng phân cấp (DFS).
2. **Cách hoạt động:**

- In thông tin nút hiện tại (root) với thụt lề tùy theo cấp độ (level).

- Đệ quy gọi lại hàm với từng nút con, tăng mức thụt lề.

- Nếu có maxLevel, dừng hiển thị nếu vượt quá mức này.

1. **Giải thuật**

- Nhận vào nút gốc, chuỗi indent (để hiển thị cấu trúc) và mức độ hiện tại.

- Nếu mức độ tối đa được chỉ định và mức độ hiện tại lớn hơn mức độ tối đa, dừng lại.

- In tên của nút với định dạng (thư mục hoặc tệp tin).

- Duyệt qua các nút con và gọi đệ quy displayTree cho từng nút, tăng mức độ hiện tại.

## **Hàm safeInputInt: Kiểm tra nhập liệu an toàn**



Hình 4.8 Hàm safeInputInt

1. **Chức năng**: Đảm bảo việc nhập số nguyên từ người dùng là an toàn.
2. **Cách hoạt động**:
3. Sử dụng cin.fail() để phát hiện lỗi khi nhập liệu.
4. Nếu có lỗi:

+) Xóa trạng thái lỗi bằng cin.clear().

+) Bỏ qua phần còn lại của dòng nhập bằng cin.ignore().

1. Lặp lại cho đến khi nhận được một số nguyên hợp lệ.
2. **Giải thuật:**

1. Nhận vào một thông báo để hiển thị.

2. Trong vòng lặp:

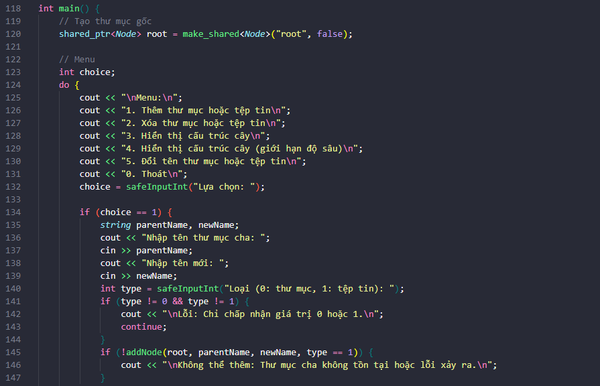
- Yêu cầu người dùng nhập một số nguyên.

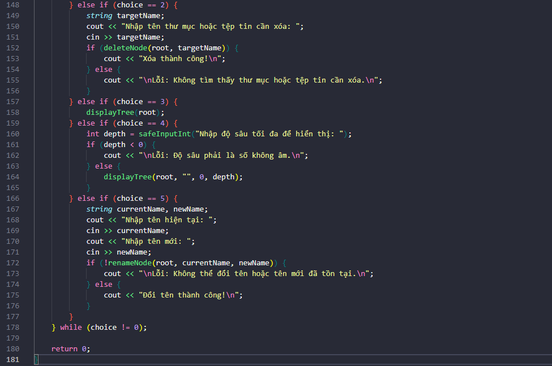
- Kiểm tra xem việc nhập liệu có thất bại không.

- Nếu thất bại, xóa lỗi và yêu cầu nhập lại.

- Nếu thành công, trả về giá trị đã nhập.

## **Hàm main**





Hình 4.9 Hàm main

**9.1 Chức năng:** Giao diện điều khiển chính của chương trình, cho phép người dùng thao tác trên cây thư mục qua menu.

**9.2 Các bước chính:**

- Tạo thư mục gốc (root).

- Hiển thị menu và nhận lựa chọn từ người dùng.

- Dựa vào lựa chọn, thực hiện các thao tác:

+) Thêm, xóa, đổi tên nút.

+) Hiển thị cây (toàn bộ hoặc giới hạn độ sâu).

- Lặp lại cho đến khi người dùng chọn thoát (choice == 0).

**9.3 Giải thuật:**

1. Tạo một nút gốc cho cây.

2. Hiển thị menu cho người dùng với các tùy chọn (thêm, xóa, hiển thị, đổi tên).

3. Dựa vào lựa chọn của người dùng, gọi các hàm tương ứng.

4. Lặp lại cho đến khi người dùng chọn thoát.

**9.4 Chi tiết:**

**9.4.1 Tạo thư mục gốc ( root):**



Hình 4.10: Tạo thư mục gốc

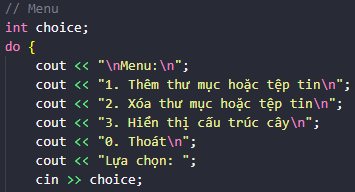
- Mục đích: Tạo một thư mục gốc với tên "root".

- Ý nghĩa:

+) shared\_ptr<Node> để đảm bảo quản lý bộ nhớ tự động, tránh rò rỉ bộ nhớ.

+) false được truyền vào constructor Node để xác định root là một thư mục (không phi tệp tin).

**9.4.2 Hiển thị menu và nhận lựa chọn từ người dùng**



Hình 4.11: Hiển thị menu và nhận lựa chọn từ người dùng

**- Cách hoạt động**:

+) Hiển thị các tùy chọn của chương trình, mỗi số tương ứng với một chức năng.

+) Người dùng nhập số tương ứng với lựa chọn của họ.

+) Hàm safeInputInt đảm bảo giá trị nhập vào là số nguyên hợp lệ.

**- Chức năng menu**:

+) 1: Thêm thư mục hoặc tệp tin.

+) 2: Xóa thư mục hoặc tệp tin.

+) 3: Hiển thị toàn bộ cấu trúc cây.

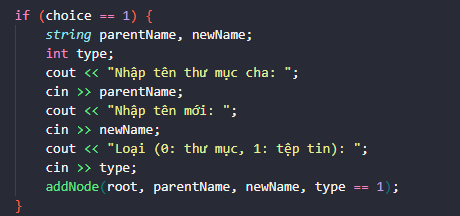
+) 4: Hiển thị cấu trúc cây với độ sâu giới hạn.

+) 5: Đổi tên thư mục hoặc tệp tin.

+) Thoát chương trình.

**9.4.3 Xử lý lựa chọn của người dùng**

*a) Thêm thư mục hoặc tệp tin (choice == 1)*



Hình 4.12: Thêm thư mục hoặc tệp tin

**Các bước xử lý**:

1. Nhập parentName: Tên của thư mục cha mà nút mới sẽ được thêm vào.
2. Nhập newName: Tên của nút mới.
3. Nhập type: Loại nút mới (0 là thư mục, 1 là tệp tin).

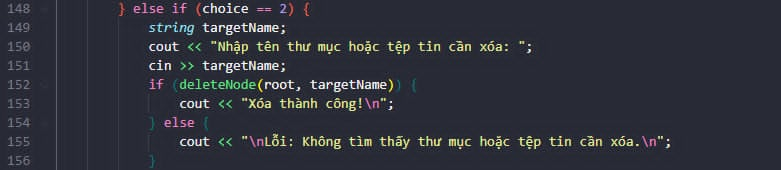
+) Nếu nhập sai giá trị (khác 0 hoặc 1), thông báo lỗi và quay lại menu.

1. Gọi hàm addNode để thêm nút mới.

+) Nếu thêm thành công, nút mới được lưu vào cây.

+) Nếu thất bại (tên trùng hoặc thư mục cha không tồn tại), thông báo lỗi.

*b) Xóa thư mục hoặc tệp tin (choice == 2)*



Hình 4.13: Xóa thư mục hoặc tệp tin

**Các bước xử lý**:

1. Nhập targetName: Tên của nút cần xóa.
2. Gọi hàm deleteNode để tìm và xóa nút này khỏi cây.

+) Nếu tìm thấy và xóa thành công, thông báo "Xóa thành công!".

+) Nếu không tìm thấy, thông báo lỗi.

*c) Hiển thị cấu trúc cây ( choice == 3)*



Hình 4.14: Hiển thị cấu trúc cây

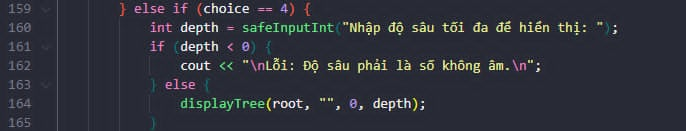
**Mục đích**: Gọi hàm displayTree để hiển thị toàn bộ cấu trúc cây, bắt đầu từ thư mục gốc (root).

**Hiển thị**:

+) Mỗi thư mục/tệp tin sẽ được in kèm với thông tin loại (thư mục hoặc tệp tin).

+) Cấu trúc cây được hiển thị theo chiều sâu (DFS).

*d) Hiển thị cấu trúc cây với độ sâu giới hạn ( choice == 4)*



Hình 4.15: Hiển thị cấu trúc cây với độ sâu giới hạn

**Mục đích**: Hiển thị cấu trúc cây nhưng chỉ hiển thị đến một độ sâu nhất định (depth).

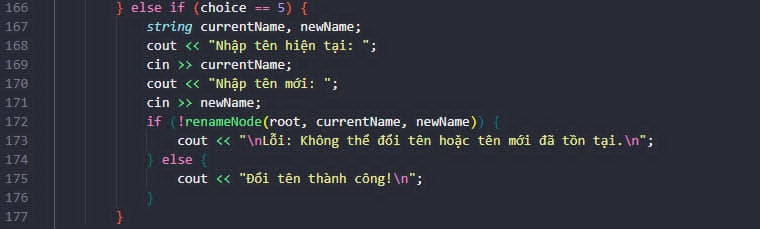
**Các bước xử lý**:

+) Nhập depth: Độ sâu tối đa. Nếu depth < 0, thông báo lỗi.

+) Gọi hàm displayTree với tham số maxLevel = depth.

+) Cây sẽ hiển thị các nút từ mức gốc đến mức depth.

*e) Đổi tên thư mục hoặc tệp tin ( choice == 5)*



Hình 4.16: Đổi tên thư mục hoặc tệp tin

**Các bước xử lý**:

- Nhập currentName: Tên hiện tại của nút cần đổi tên.

- Nhập newName: Tên mới cho nút.

- Gọi hàm renameNode:

+) Nếu đổi tên thành công, thông báo "Đổi tên thành công!".

+) Nếu thất bại (tên mới trùng hoặc không tìm thấy nút), thông báo lỗi.

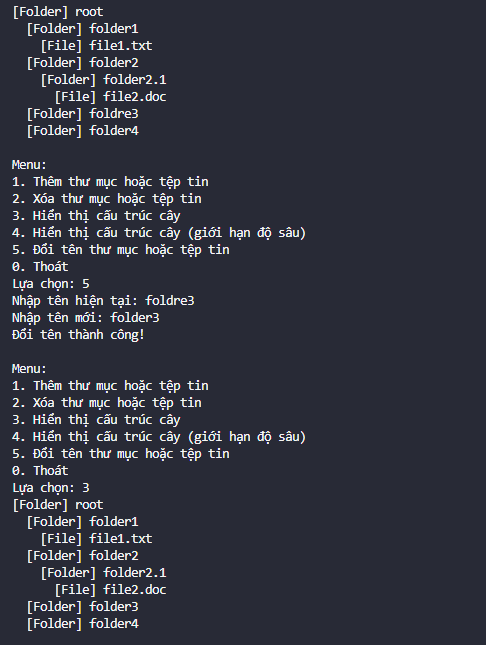
**9.4.4 Thoát chương trình (choice == 0)**

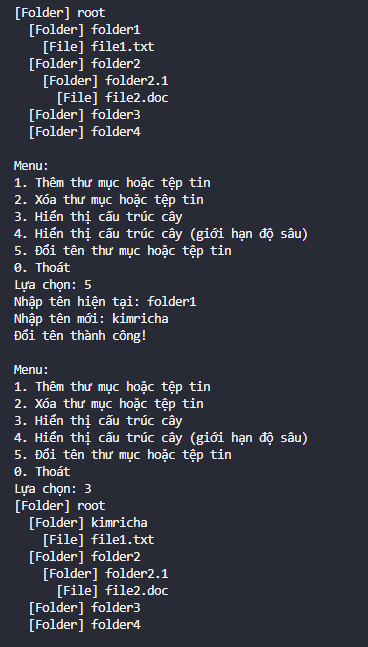


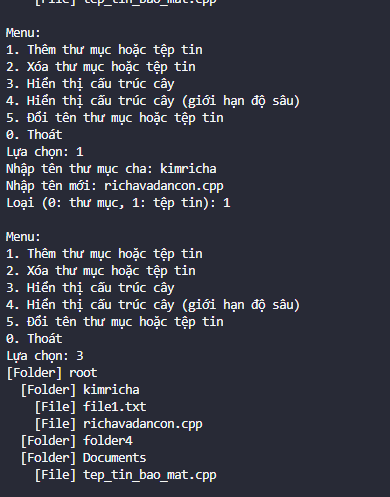
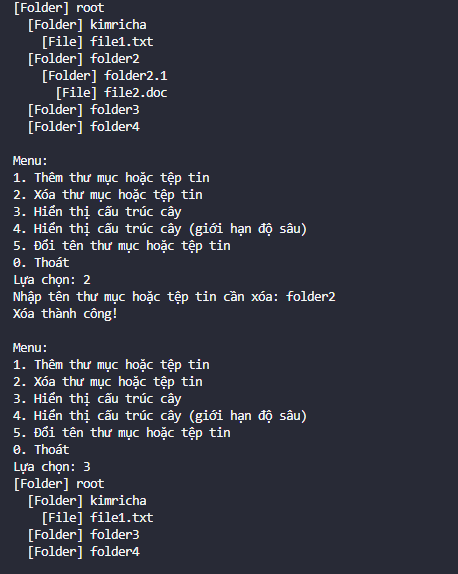
Hình 4.17: Thoát chương trình

-) Hoạt động: Khi người dùng nhập 0, chương trình thoát khỏi vòng lặp và kết thúc.

# **V: KẾT QUẢ**







# **VI: KẾT LUẬN**

Bài tập lớn trên đã xây dựng một hệ thống quản lý cây thư mục đơn giản bằng ngôn ngữ C++. Với cấu trúc dữ liệu cây và các tính năng thiết yếu, chương trình có thể mô phỏng hoạt động của một hệ thống tệp tin cơ bản.

**Các tính năng chính bao gồm:**

- Thêm thư mục hoặc tệp tin: Người dùng có thể tạo mới các node trong cây, với việc kiểm tra trùng tên đảm bảo tính hợp lệ.

- Xóa thư mục hoặc tệp tin: Hỗ trợ xóa node trong cây, bao gồm cả node cha và các node con liên quan.

- Đổi tên node: Cho phép người dùng thay đổi tên của node, với việc kiểm tra trùng tên để đảm bảo tính chính xác.

- Hiển thị cấu trúc cây: Cung cấp cái nhìn trực quan về cây thư mục với định dạng phân cấp rõ ràng, hỗ trợ giới hạn độ sâu hiển thị.

- Xử lý nhập liệu an toàn: Đảm bảo rằng chương trình hoạt động ổn định khi người dùng nhập sai hoặc không hợp lệ.

**Ưu điểm:**

- Tính đệ quy mạnh mẽ: Các thao tác như duyệt, thêm, xóa, và đổi tên được thực hiện hiệu quả nhờ sử dụng đệ quy phù hợp với cấu trúc cây.

- Quản lý bộ nhớ an toàn: Sử dụng shared\_ptr để quản lý bộ nhớ, giảm thiểu nguy cơ rò rỉ và giúp chương trình dễ dàng mở rộng.

- Tính dễ mở rộng: Với thiết kế hướng đối tượng, các chức năng mới có thể được thêm vào mà không ảnh hưởng lớn đến cấu trúc hiện tại.

**Nhược điểm và Hạn chế:**

- Hiệu năng hạn chế: Việc duyệt cây theo cách đệ quy có thể không tối ưu khi cây trở nên rất lớn.

- Khả năng xử lý lỗi chưa toàn diện: Các lỗi như nhập liệu phức tạp (tên chứa ký tự đặc biệt) hoặc vòng lặp vô tận có thể chưa được xử lý triệt để.

- Chưa hỗ trợ lưu trữ lâu dài: Cấu trúc cây hiện tại chỉ tồn tại trong bộ nhớ, không thể lưu và tải từ tệp.

# **VII: LỜI KẾT**

Qua đây bọn em muốn gửi lời cảm ơn cô Trần Thị Thanh Hải đã tận tình hướng dẫn, giảng dạy giúp chúng em hoàn thành bài tập này. Sản phẩm của chúng em tuy đã hoàn thành nhưng vẫn còn nhiều thiếu xót, chúng em rất mong sẽ nhận được những lời nhận xét chỉ bảo của cô. Chúng em xin chân thành cảm ơn.