

**TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN**



**THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2025-2026**

Mô Phỏng Cây Nhị Phân Tìm kiếm

Giảng viên hướng dẫn:

ThS. Lê Minh Tự

Sinh viên thực hiện:

Họ tên: Trần Duy Luận

MSSV: 110123130

Lớp: DA23TTC

Vĩnh Long, tháng 12 năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

[illegible]

Trà Vinh, ngày tháng năm
Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

[illegible]

Trà Vinh, ngày tháng năm

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trong quá trình thực hiện đồ án cơ sở ngành với đề tài **“Mô phỏng cây nhị phân tìm kiếm bằng ngôn ngữ C”**, em đã nhận được sự quan tâm, hỗ trợ và hướng dẫn tận tình từ Thầy, gia đình và bạn bè. Đây là nguồn động viên to lớn giúp em có thể hoàn thành đồ án đúng tiến độ và đạt được những mục tiêu đã đề ra.

Trước hết, em xin gửi lời cảm ơn chân thành đến Thầy Khoa Kỹ thuật và Công nghệ – Trường Đại học Trà Vinh đã tận tâm giảng dạy, truyền đạt những kiến thức nền tảng và chuyên môn trong suốt quá trình học tập. Những kiến thức này là cơ sở quan trọng giúp em có thể tiếp cận và nghiên cứu đề tài một cách hiệu quả.

Em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên hướng dẫn đã dành nhiều thời gian theo dõi, góp ý và định hướng cho em trong quá trình thực hiện đồ án. Những nhận xét và chỉ dẫn của Thầy đã giúp em hiểu rõ hơn về cấu trúc dữ liệu cây nhị phân, cách xây dựng chương trình bằng ngôn ngữ C cũng như phương pháp trình bày một đồ án khoa học, logic và đúng quy định.

Bên cạnh đó, em xin cảm ơn gia đình và bạn bè đã luôn động viên, hỗ trợ tinh thần trong suốt quá trình học tập và thực hiện đồ án. Sự quan tâm và khích lệ này là động lực để em vượt qua những khó khăn trong quá trình nghiên cứu.

Do kiến thức và kinh nghiệm thực tế còn hạn chế, đồ án không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý chân thành từ Thầy để em có thể hoàn thiện hơn trong quá trình học tập và nghiên cứu sau này.

Em xin chân thành cảm ơn!

MỤC LỤC

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH	viii
1. Vấn đề nghiên cứu	viii
2. Hướng tiếp cận (Cách mô phỏng động):	viii
3. Một số kết quả đạt được	viii
MỞ ĐẦU	1
1.1. Lý do chọn đề tài	1
1.2. Mục đích và ý nghĩa của đề tài	1
1.2.1. Mục đích	1
1.2.2. Ý nghĩa	2
1.3. Đối tượng và phạm vi nghiên cứu	3
CHƯƠNG 1: TỔNG QUAN	4
1.1 Giới thiệu chung về vấn đề nghiên cứu	4
1.2 Tổng quan về cấu trúc dữ liệu cây và cây nhị phân	4
1.3 Giới thiệu tổng quan về vấn đề sẽ tập trung nghiên cứu và giải quyết.....	5
2.1 Hệ quản trị cơ sở dữ liệu	7
2.1.1 Khái niệm cấu trúc dữ liệu cây	7
2.1.2 Các thành phần cơ bản của cây.....	7
2.1.3 Vai trò của cấu trúc dữ liệu cây trong lập trình	7
2.2 Cây nhị phân	8
2.2.1 Khái niệm cây nhị phân	8
2.2.2 Các dạng cây nhị phân.....	8
2.2.3 Ưu điểm và hạn chế của cây nhị phân	8
2.3 Cây nhị phân tìm kiếm.....	9
2.3.1 Khái niệm cây nhị phân tìm kiếm.....	9
2.3.2 Các phép toán cơ bản trên cây nhị phân tìm kiếm.....	9
2.3.3 Cơ sở lý thuyết phục vụ hiện thực hóa bằng ngôn ngữ C	10
2.4 Các thuật toán và phương pháp xử lý trên cây nhị phân tìm kiếm	11
2.4.1 Thuật toán thêm nút vào cây nhị phân tìm kiếm	11
2.4.2 Thuật toán tìm kiếm trong cây nhị phân tìm kiếm	11
2.4.3 Thuật toán xóa nút trong cây nhị phân tìm kiếm.....	12
2.5 Phương pháp nghiên cứu và công cụ sử dụng trong đồ án	13
2.5.1 Phương pháp nghiên cứu	13
2.5.2 Công cụ và môi trường phát triển.....	13
2.5.3 Đánh giá tính phù hợp của phương pháp tiếp cận	13
2.6 Độ phức tạp thời gian và không gian của cây nhị phân tìm kiếm.....	14
2.6.1 Độ phức tạp của thao tác thêm nút	14
2.6.2 Độ phức tạp của thao tác tìm kiếm	14
2.6.3 Độ phức tạp bộ nhớ	14
2.7 Ý nghĩa và vai trò của cây nhị phân tìm kiếm trong học tập và thực tiễn	15
2.7.1 Vai trò trong học tập và nghiên cứu	15
2.7.2 Ứng dụng trong các hệ thống phần mềm.....	15
2.7.3 Định hướng mở rộng từ cây nhị phân tìm kiếm	15
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU	16
3.1 Phân tích yêu cầu hệ thống	16
3.1.1 Mục tiêu xây dựng chương trình	16
3.1.2 Đối tượng sử dụng	16
3.1.3 Yêu cầu chức năng	16
3.2 Phân tích chức năng và luồng xử lý.....	17
3.2.1 Xác định các chức năng chính.....	17
3.2.2 Luồng xử lý tổng quát	17

3.2.3 Các trường hợp sử dụng tiêu biểu	17
3.3 Thiết kế tổng thể chương trình.....	19
3.3.1 Kiến trúc chương trình.....	19
3.4 Thiết kế cấu trúc dữ liệu	19
3.4.1 Mô hình nút trong cây	19
3.4.2 Quản lý bộ nhớ động	19
3.4.3 Các biến và dữ liệu toàn cục trong chương trình.....	20
3.5 Thiết kế thuật toán và hiện thực hóa	20
3.5.1 Thuật toán thêm nút.....	20
3.5.2 Thuật toán tìm kiếm.....	20
3.5.3 Thuật toán xóa nút	20
3.5.4 Thuật toán duyệt cây (phục vụ kiểm thử).....	20
3.6 Thiết kế giao diện mô phỏng	20
3.6.1 Thành phần giao diện	20
3.6.2 Cách xử lý tương tác người dùng	21
3.6.3 Quy ước hiển thị trên màn hình.....	21
3.7 Quy trình chạy chương trình.....	22
3.7.1 Các bước vận hành	22
3.7.2 Một số tình huống thao tác mẫu	22
3.7.3 Xử lý lỗi nhập liệu	22
3.8 Kiểm thử chương trình.....	22
3.8.1 Mục tiêu kiểm thử.....	22
3.8.2 Các trường hợp kiểm thử.....	22
3.8.3 Kết quả kiểm thử	23
3.9 Tổng lợi nhuận.....	24
3.9.1 Kết quả đạt được.....	24
3.9.2 Khó khăn trong quá trình thực hiện.....	24
3.9.3 Bài học kinh nghiệm.....	24
CHƯƠNG 4: KẾT QUẢ	25
4.1 Kết quả thực hiện các chức năng chính	25
4.1.1 Kết quả chức năng thêm nút.....	25
4.1.2 Kết quả chức năng tìm kiếm.....	25
4.1.3 Kết quả chức năng xóa nút	26
4.2 Kết quả hiển thị và giao diện chương trình.....	28
4.2.1 Giao diện tổng thể	28
4.2.2 Kết quả hiển thị cây nhị phân tìm kiếm.....	28
4.2.3 Tính trực quan của chương trình	29
4.3 Đánh giá hiệu năng và trải nghiệm người dùng.....	29
4.3.1 Đánh giá hiệu năng xử lý.....	29
4.3.2 Trải nghiệm người dùng	29
4.3.3 Độ ổn định của chương trình	30
4.4 Nhận xét chung về kết quả nghiên cứu	30
4.4.1 Mức độ đáp ứng mục tiêu đề tài	30
4.4.2 Ưu điểm của chương trình.....	30
4.4.3 Hạn chế còn tồn tại	30
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	31
DANH MỤC TÀI LIỆU THAM KHẢO	32

DANH MỤC HÌNH ẢNH

Hình 1. Mô phỏng các dạng cây nhị phân	8
Hình 2. Mô phỏng các phép toán cơ bản	10
Hình 3. Minh họa thuật toán thêm nút.....	11
Hình 4. Minh họa thuật toán tìm kiếm	12
Hình 5. Minh họa thuật toán xóa nút	13
Hình 6. Minh họa kết quả thêm nút	25
Hình 7. Minh họa kết quả tìm nút.....	26
Hình 8. Minh họa kết quả xóa nút	27
Hình 9. Minh họa kết quả giao diện hiển thị	28
Hình 10. Minh họa kết quả hiển thị cây	29

DANH MỤC BẢNG BIỂU

Bảng 1. So sánh các dạng cây nhị phân phổ biến.....	2
Bảng 2. Các phép toán cơ bản trên cây nhị phân tìm kiếm.....	6
Bảng3. Độ phức tạp thời gian của cây nhị phân.....	7
Bảng4. Các hàm chính trong chương trình mô phỏng.....	15
Bảng5. Các trường hợp kiểm thử chương trình mô phỏng cây nhị phân tìm kiếm...	24

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

1. Vấn đề nghiên cứu

Đồ án cơ sở ngành với đề tài “**Mô phỏng cây nhị phân tìm kiếm bằng ngôn ngữ C**” được thực hiện nhằm nghiên cứu và ứng dụng cấu trúc dữ liệu cây nhị phân vào việc xây dựng chương trình mô phỏng trực quan. Nội dung đồ án tập trung vào việc tìm hiểu cơ sở lý thuyết về cây nhị phân tìm kiếm, các thao tác cơ bản như thêm nút, tìm kiếm nút và xóa nút, từ đó hiện thực hóa bằng chương trình viết bằng ngôn ngữ C.

Trong đồ án, em sử dụng phương pháp nghiên cứu lý thuyết kết hợp với thực nghiệm lập trình để xây dựng chương trình mô phỏng. Chương trình cho phép người dùng nhập dữ liệu, thao tác trên cây và quan sát cấu trúc cây thông qua giao diện đồ họa đơn giản. Kết quả đạt được cho thấy chương trình hoạt động đúng yêu cầu, giúp người học dễ dàng hình dung cách hoạt động của cây nhị phân tìm kiếm trong thực tế.

2. Hướng tiếp cận (Cách mô phỏng động):

Cấu trúc nút: Mỗi nút chứa một giá trị (key) và hai con trỏ (left, right) trỏ đến cây con trái và cây con phải.

Quy tắc sắp xếp :

Giá trị của nút con trái < giá trị của nút cha.

Giá trị của nút con phải > giá trị của nút cha.

3. Một số kết quả đạt được

Sau quá trình nghiên cứu và thực hiện đồ án, em đã đạt được kết quả như sau:

Em đã xây dựng thành công chương trình mô phỏng cây nhị phân tìm kiếm bằng ngôn ngữ C, đáp ứng đầy đủ các chức năng cơ bản như thêm, xóa và tìm kiếm nút trong cây. Chương trình hoạt động ổn định và cho kết quả đúng theo nguyên lý lý thuyết của cây nhị phân tìm kiếm.

MỞ ĐẦU

1.1. Lý do chọn đề tài

Trong chương trình đào tạo ngành Công nghệ thông tin, cấu trúc dữ liệu là học phần nền tảng, đóng vai trò quan trọng trong việc xây dựng các hệ thống phần mềm hiệu quả. Trong đó, cây nhị phân tìm kiếm là một cấu trúc dữ liệu tiêu biểu, giúp tổ chức dữ liệu có thứ tự và hỗ trợ tìm kiếm nhanh chóng.

Tuy nhiên, việc tiếp cận cây nhị phân tìm kiếm thông qua lý thuyết thuần túy thường gây khó khăn cho sinh viên. Do đó, việc xây dựng một chương trình mô phỏng trực quan bằng ngôn ngữ C sẽ giúp người học dễ dàng hình dung và hiểu sâu hơn về cách thức hoạt động của cấu trúc này.

Xuất phát từ nhu cầu đó, em chọn đề tài “*Mô phỏng cây nhị phân tìm kiếm bằng ngôn ngữ C*” làm đồ án cơ sở ngành.

1.2. Mục đích và ý nghĩa của đề tài

1.2.1. Mục đích

Xây dựng một chương trình minh họa các thao tác cơ bản trên BST (như chèn, tìm kiếm, xóa, duyệt cây) để hiểu rõ về cấu trúc dữ liệu này, cách thức hoạt động của thuật toán tìm kiếm hiệu quả, từ đó ứng dụng vào các bài toán thực tế yêu cầu quản lý dữ liệu có thứ tự một cách nhanh chóng và tối ưu hóa việc truy xuất thông tin.

Các mục tiêu cụ thể thường bao gồm:

Nắm vững lý thuyết: Hiểu sâu sắc về định nghĩa, tính chất của cây nhị phân tìm kiếm, đặc biệt là thuộc tính sắp xếp của nó (nút con trái nhỏ hơn, nút con phải lớn hơn nút cha).

Thực hiện các thao tác cơ bản: Chèn (Insert): Thêm một nút mới vào đúng vị trí trong cây, duy trì tính chất BST.

Tìm kiếm (Search): Tìm một giá trị cho trước trong cây một cách hiệu quả.

Xóa (Delete): Xóa một nút bất kỳ (nút lá, nút có một con, nút có hai con) mà vẫn giữ nguyên cấu trúc.

Duyệt cây (Traversal): Thực hiện các phương pháp duyệt như Inorder (trung thứ tự), Preorder (tiền thứ tự), Postorder (hậu thứ tự) để truy cập tất cả các nút.

Minh họa tính hiệu quả: Chứng minh khả năng tìm kiếm nhanh hơn so với các cấu trúc dữ liệu tuyến tính (như mảng, danh sách liên kết) khi dữ liệu lớn.

Ứng dụng thực tế (Mở rộng): Áp dụng BST để giải quyết các bài toán như quản lý danh sách, tìm kiếm dữ liệu, hoặc làm nền tảng cho các cấu trúc dữ liệu phức tạp hơn như cây cân bằng (AVL, Red-Black Tree).

Hoàn thiện kỹ năng lập trình C: Rèn luyện kỹ năng sử dụng con trỏ, cấp phát bộ nhớ động (malloc, free), và thiết kế chương trình có cấu trúc (sử dụng struct, hàm) trong C.

Mục tiêu là biến lý thuyết thành sản phẩm cụ thể, giúp người học thực hành và hình dung được cách hoạt động của cây nhị phân tìm kiếm trong lập trình.

1.2.2 Ý nghĩa

Đề tài mang ý nghĩa về thực tiễn và chuyên môn, cụ thể như sau:

Ý nghĩa thực tiễn:

Ứng dụng trong quản lý dữ liệu: Dùng để xây dựng từ điển, danh sách, bảng băm (hash table) hiệu quả, giúp tìm kiếm, thêm, xóa dữ liệu nhanh hơn nhiều so với mảng thông thường.

Cơ sở cho hệ thống thông tin: Các hệ thống quản lý sinh viên, quản lý kho, hay cơ sở dữ liệu nhỏ đều có thể áp dụng BST để lưu trữ và truy vấn thông tin nhanh chóng.

Phát triển Game: Dùng để quản lý tài nguyên, vị trí đối tượng trong game, hoặc kiểm tra va chạm.

Tối ưu hóa thuật toán: Hiểu rõ BST giúp lập trình viên lựa chọn cấu trúc dữ liệu phù hợp, nâng cao hiệu suất chương trình.

Thực hành Lập trình C: củng cố kiến thức về con trỏ, cấp phát bộ nhớ động, đệ quy - những kỹ năng cốt lõi trong C.

Ý nghĩa chuyên môn:

Hiểu sâu Cấu trúc dữ liệu & Thuật toán: Nắm vững lý thuyết về cây, các phép duyệt (inorder, preorder, postorder), và cơ chế hoạt động của BST, đặt nền móng cho các cấu trúc phức tạp hơn.

Phân tích hiệu năng: Hiểu được độ phức tạp thời gian ($O(\log n)$ trung bình, $O(n)$ xấu nhất) và biết cách tránh tình trạng cây bị lệch (skewed tree).

Nền tảng phát triển Cây cân bằng (Self-Balancing Trees): Đề tài này là bước đệm để tìm hiểu về cây AVL, cây Red-Black, giúp tối ưu hóa hiệu năng luôn ở mức logarit ($O(\log n)$) ngay cả trong trường hợp xấu nhất.

Kỹ năng giải quyết vấn đề: Rèn luyện tư duy chia để trị (divide and conquer) thông qua các thuật toán trên cây, giúp giải quyết các bài toán phức tạp.

Hiểu biết về Tối ưu hóa (Optimization): Biết cách tổ chức dữ liệu để giảm thiểu thao tác, tăng tốc độ truy vấn, một kỹ năng quan trọng trong phát triển phần mềm.

1.3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu

Đối tượng nghiên cứu (Mô hình hóa):

Cấu trúc Nút (Node): Dữ liệu (key/value) và con trỏ tới nút con trái, nút con phải.

Cấu trúc Cây (Tree): Con trỏ tới gốc (root), các thao tác quản lý toàn bộ cây.

Các nút (Nodes): Tập hợp các phần tử dữ liệu được tổ chức theo quy tắc BST

Phạm vi nghiên cứu

Lý thuyết cơ bản: Định nghĩa, tính chất, ưu điểm (tìm kiếm hiệu quả), nhược điểm (cây lệch).

Thao tác cơ bản:

Chèn (Insert): Thêm một phần tử mới vào đúng vị trí.

Tìm kiếm (Search): Tìm một phần tử có giá trị cho trước.

Xóa (Delete): Xóa một phần tử khỏi cây.

Duyệt cây (Traversal): Duyệt theo các phương pháp Inorder (trái-gốc-phải) để lấy dữ liệu sắp xếp, Preorder (gốc-trái-phải), Postorder (trái-phải-gốc).

Cài đặt bằng C: Sử dụng cấu trúc struct và con trỏ để hiện thực hóa các thao tác trên.

Ví dụ ứng dụng: Xây dựng một chương trình quản lý đơn giản (ví dụ: quản lý danh sách số, từ điển đơn giản).

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu chung về vấn đề nghiên cứu

Trong bối cảnh công nghệ thông tin ngày càng phát triển, việc xử lý và quản lý dữ liệu hiệu quả đóng vai trò then chốt trong mọi hệ thống phần mềm. Dữ liệu không chỉ đơn thuần là các giá trị được lưu trữ mà còn là cơ sở để xây dựng các thuật toán, ra quyết định và tối ưu hóa hiệu năng của chương trình. Do đó, việc lựa chọn cấu trúc dữ liệu phù hợp có ý nghĩa rất quan trọng trong quá trình thiết kế và phát triển phần mềm.

Cấu trúc dữ liệu là nền tảng của các thuật toán, quyết định trực tiếp đến tốc độ xử lý, mức độ sử dụng bộ nhớ và khả năng mở rộng của hệ thống. Trong thực tế, cùng một bài toán nhưng nếu sử dụng cấu trúc dữ liệu khác nhau thì hiệu quả đạt được có thể chênh lệch đáng kể. Vì vậy, nghiên cứu và ứng dụng đúng cấu trúc dữ liệu là yêu cầu bắt buộc đối với sinh viên và lập trình viên trong lĩnh vực Công nghệ thông tin.

1.2 Tổng quan về cấu trúc dữ liệu cây và cây nhị phân

Các cấu trúc dữ liệu thường được phân thành hai nhóm chính là cấu trúc dữ liệu tuyến tính và cấu trúc dữ liệu phi tuyến. Trong đó, cấu trúc dữ liệu phi tuyến, đặc biệt là cấu trúc cây, được sử dụng rộng rãi để biểu diễn các mối quan hệ phân cấp trong thực tế. Cây là một cấu trúc dữ liệu bao gồm các nút được liên kết với nhau theo quan hệ cha – con, với một nút gốc duy nhất làm điểm xuất phát.

Cấu trúc cây có khả năng biểu diễn dữ liệu một cách trực quan và logic, phù hợp với nhiều bài toán như quản lý thư mục, sơ đồ tổ chức, biểu thức toán học và cú pháp trong trình biên dịch. Trong các dạng cây, cây nhị phân là dạng phổ biến và quan trọng, trong đó mỗi nút có tối đa hai nút con là nút con trái và nút con phải.

Nhờ đặc điểm này, cây nhị phân rất phù hợp cho việc xây dựng các thuật toán tìm kiếm, sắp xếp và xử lý dữ liệu có thứ tự. Một số dạng cây nhị phân thường gặp bao gồm cây nhị phân đầy đủ, cây nhị phân hoàn chỉnh và cây nhị phân tìm kiếm, trong đó cây nhị phân tìm kiếm được sử dụng nhiều trong các bài toán tra cứu dữ liệu.

Loại cây	Đặc điểm	Ưu điểm	Hạn chế
Cây nhị phân thường	Mỗi nút có tối đa 2 con	Cấu trúc đơn giản	Không có thứ tự
Cây nhị phân hoàn chỉnh	Các mức được lấp đầy từ trái sang phải	Tối ưu không gian	Khó cập nhật động
Cây nhị phân tìm kiếm (BST)	Trái < Góc < Phải	Tìm kiếm nhanh	Có thể bị lệch

Bảng 1. So sánh các dạng cây nhị phân phổ biến

1.3 Giới thiệu tổng quan về vấn đề sẽ tập trung nghiên cứu và giải quyết

Xuất phát từ vai trò quan trọng của cấu trúc dữ liệu cây, đề án này tập trung nghiên cứu **cây nhị phân tìm kiếm (Binary Search Tree – BST)**, một dạng cây nhị phân có thứ tự cho phép thực hiện các thao tác tìm kiếm, thêm và xóa dữ liệu một cách hiệu quả. Cây nhị phân tìm kiếm hoạt động dựa trên nguyên tắc sắp xếp dữ liệu theo thứ tự, giúp giảm số lượng phép so sánh so với các phương pháp tìm kiếm tuyến tính.

Đề án tập trung phân tích cơ sở lý thuyết của cây nhị phân tìm kiếm, bao gồm cấu trúc, các tính chất và các phép toán cơ bản như thêm nút, tìm kiếm và xóa nút. Trên cơ sở đó, tiến hành hiện thực hóa các thuật toán này bằng **ngôn ngữ lập trình C**, nhằm giúp sinh viên hiểu rõ cách tổ chức dữ liệu trong bộ nhớ thông qua con trỏ và cấu trúc dữ liệu động.

Việc lựa chọn ngôn ngữ C để xây dựng chương trình mô phỏng không chỉ giúp nâng cao kỹ năng lập trình mà còn góp phần củng cố kiến thức về quản lý bộ nhớ và tư duy thuật toán. Thông qua đề án, sinh viên có thể nắm vững nguyên lý hoạt động của cây nhị phân tìm kiếm và làm nền tảng để tiếp cận các cấu trúc dữ liệu nâng cao hơn trong quá trình học tập và nghiên cứu sau này.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Hệ quản trị cơ sở dữ liệu

2.1.1 Khái niệm cấu trúc dữ liệu cây

Cấu trúc dữ liệu cây là một dạng cấu trúc dữ liệu phi tuyến, trong đó các phần tử dữ liệu được tổ chức theo mối quan hệ phân cấp. Mỗi phần tử trong cây được gọi là một nút, các nút được liên kết với nhau thông qua quan hệ cha – con. Trong một cây chỉ tồn tại duy nhất một nút gốc và từ nút này có thể phát triển ra nhiều nhánh con khác nhau.

Cấu trúc cây được sử dụng rộng rãi trong khoa học máy tính vì khả năng biểu diễn tự nhiên các mối quan hệ phân cấp, chẳng hạn như cây thư mục trong hệ điều hành, sơ đồ tổ chức, hay cấu trúc dữ liệu trong các thuật toán tìm kiếm.

2.1.2 Các thành phần cơ bản của cây

Một cây trong cấu trúc dữ liệu thường bao gồm các thành phần cơ bản như:

Nút gốc (root): là nút đầu tiên của cây.

Nút cha và nút con: thể hiện mối quan hệ phân cấp giữa các nút.

Nút lá: là những nút không có nút con.

Độ sâu và chiều cao của cây: dùng để đánh giá mức độ phân nhánh và kích thước của cây.

Việc hiểu rõ các thành phần này giúp cho quá trình phân tích và xây dựng các thuật toán trên cây trở nên dễ dàng hơn.

2.1.3 Vai trò của cấu trúc dữ liệu cây trong lập trình

Cấu trúc dữ liệu cây đóng vai trò quan trọng trong việc tổ chức và xử lý dữ liệu một cách hiệu quả. So với cấu trúc tuyến tính, cây cho phép tìm kiếm và phân loại dữ liệu nhanh hơn trong nhiều trường hợp. Đây là nền tảng cho nhiều cấu trúc dữ liệu và thuật toán nâng cao trong ngành Công nghệ thông tin.

2.2 Cây nhị phân

2.2.1 Khái niệm cây nhị phân

Cây nhị phân là một dạng đặc biệt của cấu trúc dữ liệu cây, trong đó mỗi nút có tối đa hai nút con. Hai nút con này thường được gọi là nút con trái và nút con phải. Với đặc điểm này, cây nhị phân có cấu trúc đơn giản nhưng mang lại hiệu quả cao trong việc cài đặt các thuật toán xử lý dữ liệu.

Cây nhị phân thường được sử dụng trong các bài toán tìm kiếm, sắp xếp và biểu diễn dữ liệu có thứ tự.

2.2.2 Các dạng cây nhị phân

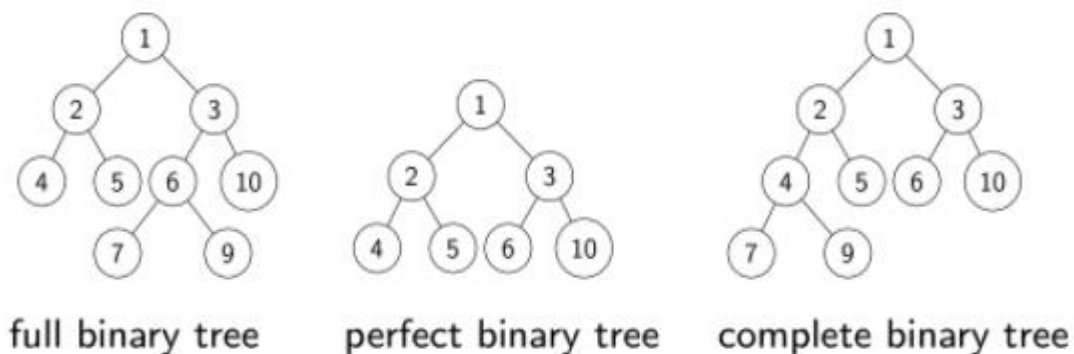
Trong thực tế, cây nhị phân có thể tồn tại dưới nhiều dạng khác nhau như:

Cây nhị phân đầy đủ

Cây nhị phân hoàn chỉnh

Cây nhị phân tìm kiếm

Mỗi dạng cây có những đặc điểm riêng và phù hợp với những bài toán khác nhau trong lập trình và xử lý dữ liệu.



Hình 1. Mô phỏng các dạng cây nhị phân

2.2.3 Ưu điểm và hạn chế của cây nhị phân

Ưu điểm của cây nhị phân là cấu trúc rõ ràng, dễ cài đặt và dễ mở rộng. Tuy nhiên, nếu cây không được tổ chức hợp lý thì có thể dẫn đến tình trạng mất cân bằng, làm giảm hiệu quả của các thao tác tìm kiếm và cập nhật dữ liệu.

Phép toán	Mô tả	Nguyên tắc thực hiện
Thêm nút	Chèn giá trị mới vào cây	Nhỏ hơn đi trái, lớn hơn đi phải
Tìm kiếm	Tìm nút theo giá trị	So sánh từ nút gốc
Xóa nút	Loại bỏ nút khỏi cây	Xử lý 3 trường hợp (lá, 1 con, 2 con)

Bảng 2. Các phép toán cơ bản trên cây nhị phân tìm kiếm

2.3 Cây nhị phân tìm kiếm

2.3.1 Khái niệm cây nhị phân tìm kiếm

Cây nhị phân tìm kiếm là cây nhị phân có thứ tự, trong đó mỗi nút tuân theo nguyên tắc: các giá trị ở cây con bên trái nhỏ hơn giá trị của nút hiện tại và các giá trị ở cây con bên phải lớn hơn giá trị của nút hiện tại. Nguyên tắc này được áp dụng cho toàn bộ cây.

Nhờ tính chất này, cây nhị phân tìm kiếm cho phép thực hiện thao tác tìm kiếm dữ liệu một cách nhanh chóng và hiệu quả.

2.3.2 Các phép toán cơ bản trên cây nhị phân tìm kiếm

Các phép toán cơ bản bao gồm:

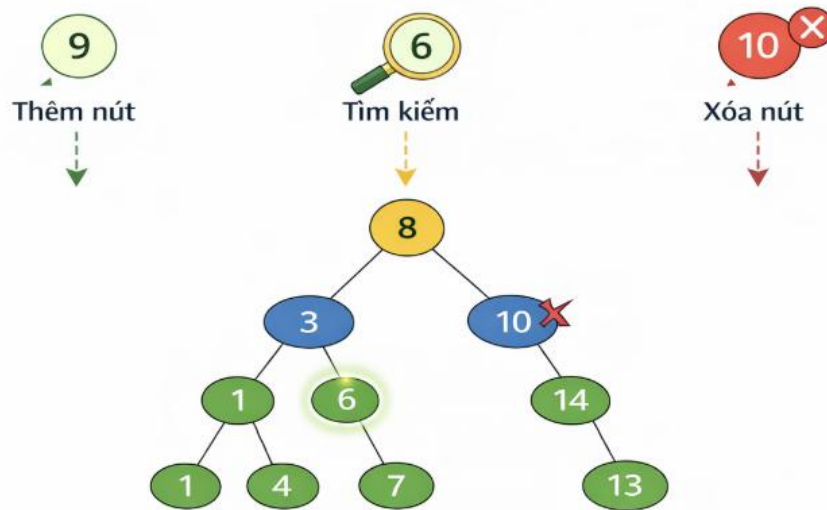
Thêm nút mới vào cây

Tìm kiếm một nút theo giá trị

Xóa một nút khỏi cây

Mỗi phép toán đều tuân theo các quy tắc nhất định nhằm đảm bảo cây luôn giữ được tính chất của cây nhị phân tìm kiếm.

Các Phép Toán Cơ Bản Trên Cây Nhị Phân Tìm Kiếm



Hình 2. Mô phỏng các phép toán cơ bản

2.3.3 Cơ sở lý thuyết phục vụ hiện thực hóa bằng ngôn ngữ C

Việc hiện thực hóa cây nhị phân tìm kiếm bằng ngôn ngữ C dựa trên việc sử dụng cấu trúc dữ liệu struct kết hợp với con trỏ. Cách tiếp cận này giúp quản lý bộ nhớ hiệu quả và phản ánh đúng bản chất động của cấu trúc cây trong thực tế.

Thao tác	Tốt nhất	Trung bình	Xấu nhất
Thêm nút	$O(\log n)$	$O(\log n)$	$O(n)$
Tìm kiếm	$O(\log n)$	$O(\log n)$	$O(n)$
Xóa nút	$O(\log n)$	$O(\log n)$	$O(n)$

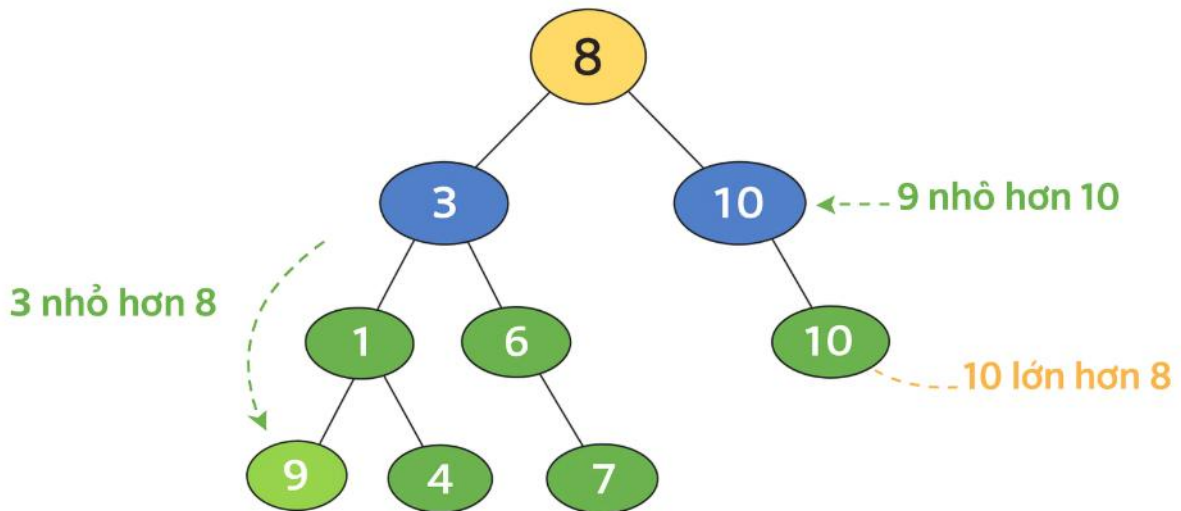
Bảng 3. Độ phức tạp thời gian của cây nhị phân

2.4 Các thuật toán và phương pháp xử lý trên cây nhị phân tìm kiếm

2.4.1 Thuật toán thêm nút vào cây nhị phân tìm kiếm

Thuật toán thêm nút là một trong những thuật toán cơ bản và quan trọng nhất trong cây nhị phân tìm kiếm. Khi thêm một giá trị mới vào cây, chương trình sẽ bắt đầu so sánh giá trị đó với giá trị của nút gốc. Nếu giá trị cần thêm nhỏ hơn giá trị của nút hiện tại thì tiếp tục so sánh với nút con bên trái, ngược lại nếu giá trị lớn hơn thì so sánh với nút con bên phải.

Quá trình này được lặp lại cho đến khi tìm được vị trí phù hợp, tức là gặp một nút con rỗng. Khi đó, một nút mới sẽ được tạo và liên kết vào cây. Thuật toán này đảm bảo sau khi thêm nút, cây vẫn giữ được đầy đủ tính chất của cây nhị phân tìm kiếm.

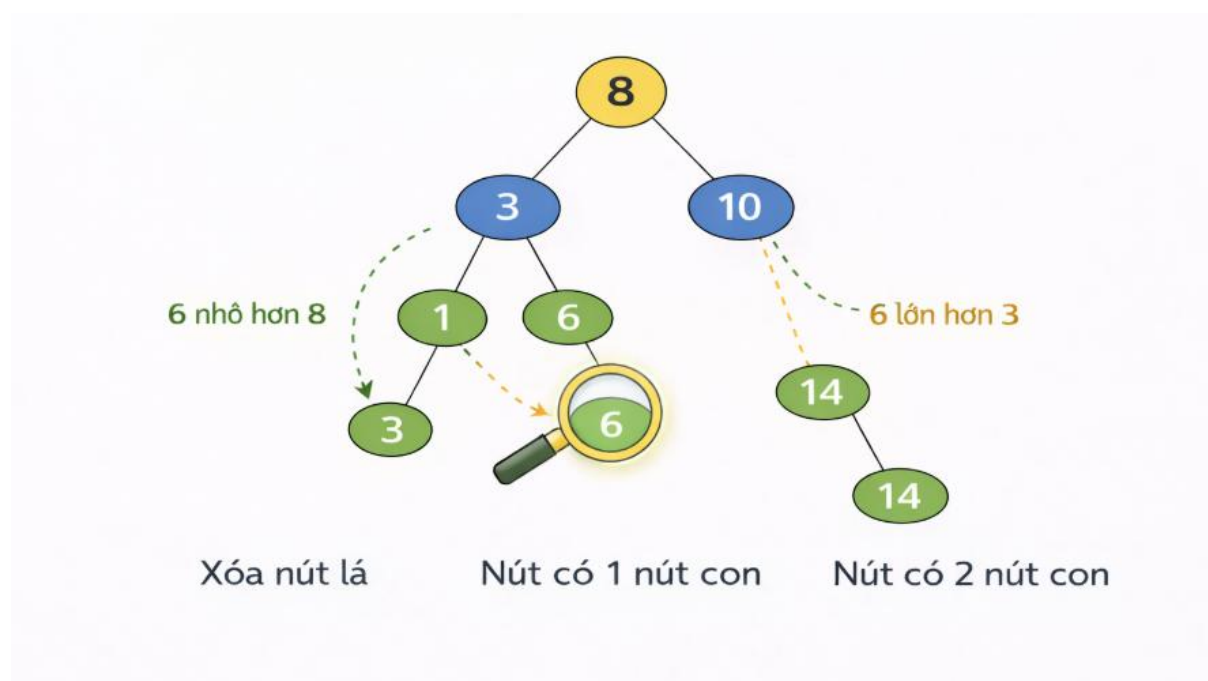


Hình 3. Minh họa thuật toán thêm nút

2.4.2 Thuật toán tìm kiếm trong cây nhị phân tìm kiếm

Thuật toán tìm kiếm trong cây nhị phân tìm kiếm dựa trên nguyên lý so sánh giá trị. Bắt đầu từ nút gốc, chương trình so sánh giá trị cần tìm với giá trị tại nút hiện tại. Nếu hai giá trị bằng nhau thì quá trình tìm kiếm kết thúc và trả về kết quả thành công.

Trong trường hợp giá trị cần tìm nhỏ hơn giá trị tại nút hiện tại, thuật toán sẽ tiếp tục tìm kiếm trong cây con bên trái. Ngược lại, nếu giá trị cần tìm lớn hơn, quá trình tìm kiếm sẽ tiếp tục trong cây con bên phải. Nhờ cách tiếp cận này, số lượng phép so sánh được giảm đáng kể so với phương pháp tìm kiếm tuyến tính.

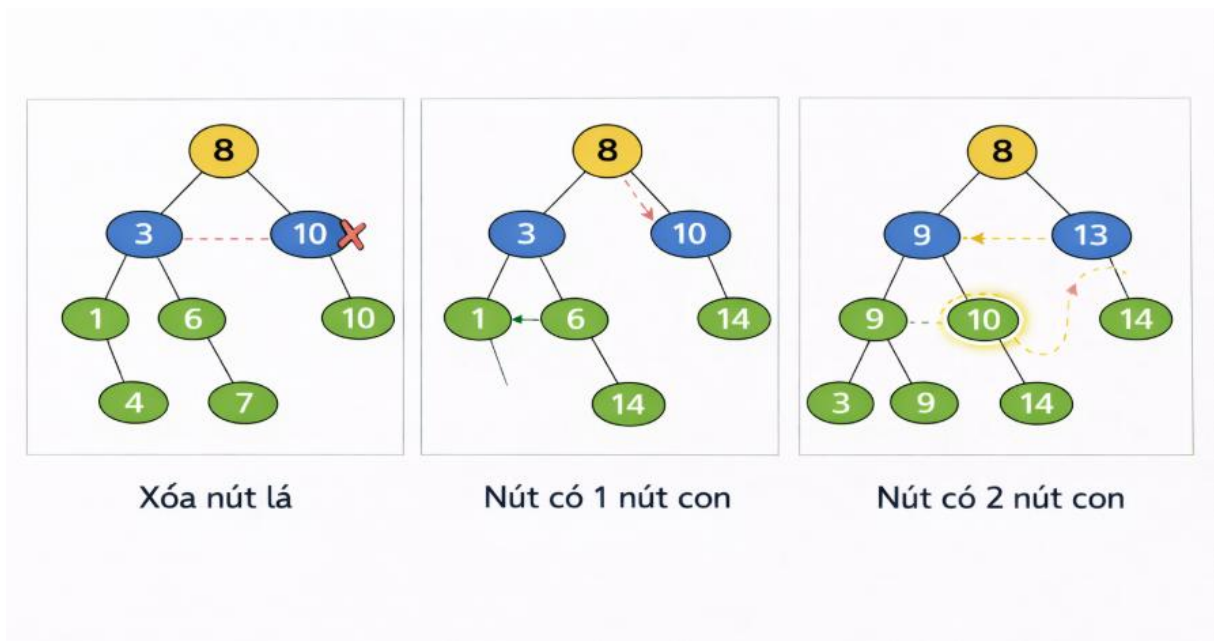


Hình 4. Minh họa thuật toán tìm kiếm

2.4.3 Thuật toán xóa nút trong cây nhị phân tìm kiếm

Xóa nút là thao tác phức tạp nhất trong cây nhị phân tìm kiếm vì cần xử lý nhiều trường hợp khác nhau. Cụ thể, nếu nút cần xóa là nút lá thì chỉ cần loại bỏ nút đó khỏi cây. Nếu nút có một nút con, chương trình sẽ thay thế nút bị xóa bằng nút con của nó.

Trường hợp nút cần xóa có hai nút con, thuật toán sẽ tìm nút thế mạng thích hợp, thường là nút có giá trị nhỏ nhất trong cây con bên phải hoặc lớn nhất trong cây con bên trái. Sau đó, giá trị của nút thế mạng sẽ được gán cho nút cần xóa và nút thế mạng sẽ bị loại bỏ. Cách làm này giúp đảm bảo cây sau khi xóa vẫn giữ đúng cấu trúc của cây nhị phân tìm kiếm.



Hình 5. Minh họa thuật toán xóa nút

2.5 Phương pháp nghiên cứu và công cụ sử dụng trong đồ án

2.5.1 Phương pháp nghiên cứu

Trong quá trình thực hiện đồ án, em sử dụng phương pháp nghiên cứu kết hợp giữa lý thuyết và thực hành. Trước hết, em tiến hành nghiên cứu các tài liệu liên quan đến cấu trúc dữ liệu cây và cây nhị phân tìm kiếm để nắm vững các khái niệm và nguyên lý cơ bản.

Sau đó, em áp dụng các kiến thức lý thuyết đã nghiên cứu để xây dựng mô hình và hiện thực hóa chương trình bằng ngôn ngữ C. Quá trình thực hiện được kiểm tra và điều chỉnh liên tục nhằm đảm bảo chương trình hoạt động đúng với yêu cầu đặt ra.

2.5.2 Công cụ và môi trường phát triển

Đồ án được thực hiện bằng ngôn ngữ lập trình C với sự hỗ trợ của môi trường phát triển trên hệ điều hành Windows. Các công cụ chính được sử dụng bao gồm trình biên dịch C và thư viện Win32 để xây dựng giao diện mô phỏng.

Việc sử dụng các công cụ này giúp chương trình có giao diện trực quan, dễ thao tác và phù hợp cho mục đích học tập, nghiên cứu trong lĩnh vực Công nghệ thông tin.

2.5.3 Đánh giá tính phù hợp của phương pháp tiếp cận

Phương pháp tiếp cận được lựa chọn trong đồ án phù hợp với phạm vi và mục tiêu nghiên cứu đã đề ra. Việc kết hợp giữa lý thuyết cấu trúc dữ liệu và lập trình thực

tế giúp em hiểu sâu hơn về cây nhị phân tìm kiếm cũng như cách áp dụng kiến thức vào giải quyết các bài toán cụ thể.

2.6 Độ phức tạp thời gian và không gian của cây nhị phân tìm kiếm

2.6.1 Độ phức tạp của thao tác thêm nút

Độ phức tạp thời gian của thao tác thêm nút trong cây nhị phân tìm kiếm phụ thuộc vào chiều cao của cây. Trong trường hợp cây có cấu trúc cân đối, chiều cao của cây xấp xỉ $\log_2(n)$, do đó độ phức tạp của thao tác thêm nút là $O(\log n)$. Điều này cho phép chương trình xử lý dữ liệu một cách hiệu quả ngay cả khi số lượng phần tử lớn.

Tuy nhiên, trong trường hợp xấu nhất, khi dữ liệu được thêm vào theo thứ tự tăng dần hoặc giảm dần, cây sẽ bị lệch và trở thành một dạng gần giống danh sách liên kết. Khi đó, chiều cao của cây có thể bằng số lượng nút n , dẫn đến độ phức tạp của thao tác thêm nút là $O(n)$.

2.6.2 Độ phức tạp của thao tác tìm kiếm

Tương tự như thao tác thêm nút, thao tác tìm kiếm trong cây nhị phân tìm kiếm cũng phụ thuộc vào cấu trúc của cây. Trong trường hợp cây cân đối, mỗi lần so sánh sẽ loại bỏ được một nửa không gian tìm kiếm, do đó độ phức tạp trung bình của thao tác tìm kiếm là $O(\log n)$.

Trong trường hợp cây bị mất cân bằng, thao tác tìm kiếm có thể phải duyệt qua gần như toàn bộ các nút trong cây, khi đó độ phức tạp tăng lên $O(n)$. Điều này cho thấy việc tổ chức dữ liệu hợp lý đóng vai trò rất quan trọng trong hiệu năng của cây nhị phân tìm kiếm.

2.6.3 Độ phức tạp bộ nhớ

Về mặt không gian, mỗi nút trong cây nhị phân tìm kiếm cần một vùng nhớ để lưu trữ giá trị dữ liệu và hai con trỏ trỏ đến nút con trái và nút con phải. Tổng bộ nhớ sử dụng tỉ lệ thuận với số lượng nút trong cây. Ngoài ra, việc sử dụng đệ quy trong các thuật toán cũng tiêu tốn thêm bộ nhớ ngăn xếp trong quá trình thực thi chương trình.

2.7 Ý nghĩa và vai trò của cây nhị phân tìm kiếm trong học tập và thực tiễn

2.7.1 Vai trò trong học tập và nghiên cứu

Cây nhị phân tìm kiếm là một nội dung quan trọng trong môn học Cấu trúc dữ liệu và giải thuật. Việc nghiên cứu và hiện thực hóa cây nhị phân tìm kiếm giúp sinh viên hiểu rõ hơn về tư duy thuật toán, cách tổ chức dữ liệu và kỹ năng sử dụng con trỏ trong lập trình C.

Thông qua việc xây dựng chương trình mô phỏng, sinh viên có thể quan sát trực quan sự thay đổi của cấu trúc cây sau mỗi thao tác, từ đó nắm vững bản chất của các thuật toán thay vì chỉ học lý thuyết khô khan.

2.7.2 Ứng dụng trong các hệ thống phần mềm

Trong thực tế, cây nhị phân tìm kiếm được ứng dụng trong nhiều hệ thống phần mềm cần xử lý và tìm kiếm dữ liệu nhanh chóng. Các hệ thống quản lý dữ liệu, bộ từ điển, trình phân tích cú pháp và nhiều ứng dụng khác đều sử dụng các biến thể của cấu trúc cây để nâng cao hiệu suất xử lý.

Mặc dù trong thực tế các hệ thống lớn thường sử dụng các cấu trúc cây cân bằng nâng cao, nhưng cây nhị phân tìm kiếm vẫn là nền tảng quan trọng giúp hiểu và tiếp cận các cấu trúc phức tạp hơn.

2.7.3 Định hướng mở rộng từ cây nhị phân tìm kiếm

Từ cơ sở lý thuyết của cây nhị phân tìm kiếm, người học có thể tiếp tục nghiên cứu các cấu trúc dữ liệu nâng cao hơn nhằm khắc phục nhược điểm mất cân bằng của cây. Điều này góp phần mở rộng kiến thức và nâng cao năng lực nghiên cứu, lập trình trong lĩnh vực Công nghệ thông tin.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Phân tích yêu cầu hệ thống

3.1.1 Mục tiêu xây dựng chương trình

Mục tiêu của chương trình là mô phỏng trực quan cấu trúc cây nhị phân tìm kiếm và hỗ trợ các thao tác cơ bản trên cây. Thông qua chương trình, người học có thể quan sát sự thay đổi của cây sau mỗi thao tác, từ đó hiểu rõ bản chất tổ chức dữ liệu trong cây nhị phân tìm kiếm.

3.1.2 Đối tượng sử dụng

Chương trình hướng đến đối tượng là sinh viên ngành Công nghệ thông tin, đặc biệt là sinh viên đang học học phần Cấu trúc dữ liệu và giải thuật. Ngoài ra, người mới học lập trình C cũng có thể sử dụng chương trình như một công cụ minh họa trực quan.

3.1.3 Yêu cầu chức năng

Chương trình cần đảm bảo các chức năng cơ bản sau:

Thêm nút vào cây nhị phân tìm kiếm.

Tìm kiếm nút theo giá trị.

Xóa nút theo giá trị.

Hiển thị trực quan cây nhị phân tìm kiếm sau mỗi thao tác.

Thông báo kết quả thao tác (tìm thấy/không tìm thấy, xóa thành công, ...).

3.1.4 Yêu cầu phi chức năng

Ngoài yêu cầu chức năng, chương trình cần đạt các yêu cầu phi chức năng như:

Giao diện dễ sử dụng, thao tác đơn giản.

Chạy ổn định, hạn chế lỗi trong quá trình nhập dữ liệu.

Thời gian phản hồi nhanh khi thực hiện thao tác.

Mã nguồn rõ ràng, có thể mở rộng trong tương lai.

3.2 Phân tích chức năng và luồng xử lý

3.2.1 Xác định các chức năng chính

Từ yêu cầu bài toán, chương trình được chia thành các nhóm chức năng:

Nhóm chức năng thao tác dữ liệu: thêm, xóa, tìm kiếm.

Nhóm chức năng hiển thị: vẽ cây và cập nhật lại cây sau thao tác.

Nhóm chức năng tương tác: nhận giá trị từ người dùng và hiển thị thông báo.

3.2.2 Luồng xử lý tổng quát

Luồng xử lý tổng quát của chương trình như sau:

1. Khởi tạo cây rỗng.
2. Người dùng nhập giá trị và chọn thao tác.
3. Chương trình gọi hàm xử lý tương ứng.
4. Cây được cập nhật lại trong bộ nhớ.
5. Hệ thống vẽ lại cây trên giao diện.
6. Lặp lại cho đến khi người dùng kết thúc chương trình.

3.2.3 Các trường hợp sử dụng tiêu biểu

Một số tình huống sử dụng phổ biến:

Người dùng thêm liên tiếp nhiều giá trị để tạo cây.

Người dùng tìm kiếm một giá trị đã thêm để kiểm tra.

Người dùng xóa một giá trị để quan sát sự thay đổi của cây.

Người dùng tìm kiếm giá trị không tồn tại để kiểm tra thông báo.

Tên hàm	Chức năng	Tham số	Giá trị trả về
create_node	Tạo nút mới	Giá trị key	Con trỏ Node
insert_node	Thêm nút vào cây	Node*, key	Không
search_node	Tìm kiếm nút	Node*, key	1 / 0
delete_node	Xóa nút	Node*, key	Node*

Bảng 4. Các hàm chính trong chương trình mô phỏng

3.3 Thiết kế tổng thể chương trình

3.3.1 Kiến trúc chương trình

Chương trình được thiết kế theo hướng đơn giản, gồm:

Phần xử lý dữ liệu (logic): chứa các hàm thao tác trên cây.

Phần giao diện (UI): chứa các thành phần nhập liệu và nút bấm.

Phần hiển thị (render): vẽ cây lên màn hình.

3.3.2 Phân tách module chức năng

Để dễ quản lý, chương trình được chia thành các nhóm hàm:

Nhóm hàm quản lý nút: tạo nút, giải phóng nút.

Nhóm hàm thao tác cây: thêm, xóa, tìm kiếm.

Nhóm hàm vẽ và hiển thị: tính vị trí và vẽ cây.

3.3.3 Lý do lựa chọn ngôn ngữ C

Ngôn ngữ C giúp em hiểu sâu về con trỏ và quản lý bộ nhớ, giúp tạo giao diện đơn giản, phù hợp để mô phỏng trực quan và dễ chạy trên Windows mà không cần cài thêm thư viện.

3.4 Thiết kế cấu trúc dữ liệu

3.4.1 Mô hình nút trong cây

Mỗi nút trong cây bao gồm:

Giá trị dữ liệu của nút.

Liên kết đến nút con trái.

Liên kết đến nút con phải.

Đây là cấu trúc chuẩn để cài đặt cây nhị phân tìm kiếm trong ngôn ngữ C.

3.4.2 Quản lý bộ nhớ động

Vì số lượng nút có thể thay đổi trong quá trình chạy chương trình, em sử dụng cấp phát động (malloc) để tạo nút và sử dụng free để giải phóng bộ nhớ khi xóa nút, nhằm tránh lãng phí bộ nhớ.

3.4.3 Các biến và dữ liệu toàn cục trong chương trình

Chương trình có thể sử dụng một số biến toàn cục như:

Con trỏ gốc của cây.

Các biến giao diện (ô nhập, nút thao tác).

Các tham số hỗ trợ hiển thị khi vẽ cây.

3.5 Thiết kế thuật toán và hiện thực hóa

3.5.1 Thuật toán thêm nút

Thuật toán thêm nút đảm bảo đúng nguyên tắc: nhỏ hơn thì đi trái, lớn hơn thì đi phải. Khi gặp vị trí rỗng, tạo nút mới và liên kết vào cây.

3.5.2 Thuật toán tìm kiếm

Thuật toán tìm kiếm dựa trên so sánh:

Nếu bằng nhau → tìm thấy.

Nếu nhỏ hơn → tìm bên trái.

Nếu lớn hơn → tìm bên phải.

3.5.3 Thuật toán xóa nút

Thuật toán xóa xử lý các trường hợp:

Nút lá

Nút có một con

Nút có hai con (tìm nút thế mạng và cập nhật lại cây)

3.5.4 Thuật toán duyệt cây (phục vụ kiểm thử)

Ngoài các thao tác chính, chương trình có thể dùng duyệt cây (trước/trung/sau) để kiểm tra nhanh tính đúng của cấu trúc cây trong quá trình phát triển và kiểm thử.

3.6 Thiết kế giao diện mô phỏng

3.6.1 Thành phần giao diện

Giao diện gồm:

Ô nhập số.

Nút “Thêm”.

Nút “Xóa”.

Nút “Tìm”.

Vùng hiển thị cây.

3.6.2 Cách xử lý tương tác người dùng

Khi người dùng bấm nút thao tác, chương trình đọc dữ liệu từ ô nhập, gọi hàm xử lý tương ứng và cập nhật giao diện bằng cách vẽ lại cây.

3.6.3 Quy ước hiển thị trên màn hình

Cây được hiển thị theo dạng sơ đồ:

Nút là hình tròn chứa giá trị.

Các nhánh là đường nối giữa nút cha và nút con.

Vị trí các nút được tính toán theo mức độ sâu để hạn chế chồng chéo.

3.7 Quy trình chạy chương trình

3.7.1 Các bước vận hành

1. Chạy chương trình.
2. Nhập giá trị vào ô nhập.
3. Chọn thao tác: thêm/xóa/tìm.
4. Quan sát kết quả hiển thị cây.
5. Lặp lại cho đến khi kết thúc.

3.7.2 Một số tình huống thao tác mẫu

Ví dụ thao tác:

Thêm: 10, 5, 20, 7 \rightarrow quan sát cây thay đổi.

Tìm: 7 \rightarrow thông báo tìm thấy.

Xóa: 5 \rightarrow cây cập nhật lại.

3.7.3 Xử lý lỗi nhập liệu

Chương trình cần hạn chế các lỗi như:

Nhập ký tự không phải số.

Nhập rỗng.

Thao tác xóa/tìm khi cây rỗng.

3.8 Kiểm thử chương trình

3.8.1 Mục tiêu kiểm thử

Kiểm thử nhằm đảm bảo chương trình:

Thực hiện đúng chức năng.

Không bị lỗi khi thao tác liên tục.

Hiển thị đúng cấu trúc cây sau mỗi thao tác.

3.8.2 Các trường hợp kiểm thử

Một số trường hợp tiêu biểu:

Thêm nhiều nút và kiểm tra hiển thị.

Tìm giá trị có tồn tại và không tồn tại.

Xóa nút lá, nút có một con, nút có hai con.

Thêm dữ liệu theo thứ tự tăng dần để xem tình trạng lệch.

3.8.3 Kết quả kiểm thử

Qua kiểm thử, chương trình đáp ứng đúng các thao tác cơ bản. Những trường hợp có thể gây lỗi chủ yếu liên quan đến nhập liệu; do đó, việc kiểm tra dữ liệu đầu vào là cần thiết để tăng độ ổn định của chương trình.

3.9 Tổng lợi nhuận

3.9.1 Kết quả đạt được

Em đã xây dựng được chương trình mô phỏng cây nhị phân tìm kiếm với đầy đủ thao tác thêm, xóa, tìm kiếm và hiển thị trực quan. Chương trình phù hợp cho mục đích học tập, giúp người học dễ hình dung cấu trúc cây.

3.9.2 Khó khăn trong quá trình thực hiện

Một số khó khăn bao gồm:

Xử lý thuật toán xóa trong nhiều trường hợp.

Tính toán vị trí hiển thị các nút để tránh chồng lấn.

Đồng bộ dữ liệu cây và phần vẽ giao diện.

3.9.3 Bài học kinh nghiệm

Thông qua đồ án, em rút ra được kinh nghiệm về cách phân tích yêu cầu, thiết kế chương trình, quản lý bộ nhớ trong C và cách kiểm thử chương trình để đảm bảo tính đúng đắn.

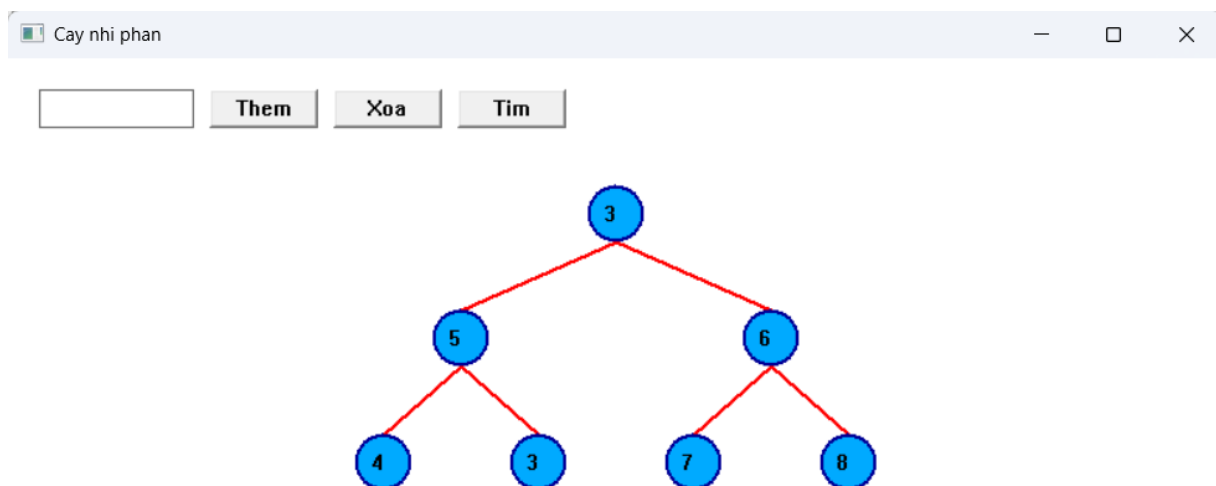
CHƯƠNG 4: KẾT QUẢ

4.1 Kết quả thực hiện các chức năng chính

4.1.1 Kết quả chức năng thêm nút

Chương trình đã thực hiện đúng chức năng thêm nút vào cây nhị phân tìm kiếm theo nguyên tắc của cấu trúc cây. Khi người dùng nhập một giá trị và chọn thao tác thêm, chương trình tự động xác định vị trí phù hợp để chèn nút mới sao cho vẫn đảm bảo tính chất của cây nhị phân tìm kiếm.

Kết quả cho thấy, các nút được thêm vào cây đúng thứ tự, không làm sai lệch cấu trúc cây. Việc thêm liên tiếp nhiều giá trị giúp hình thành cây đầy đủ và thể hiện rõ mối quan hệ giữa các nút cha – con.

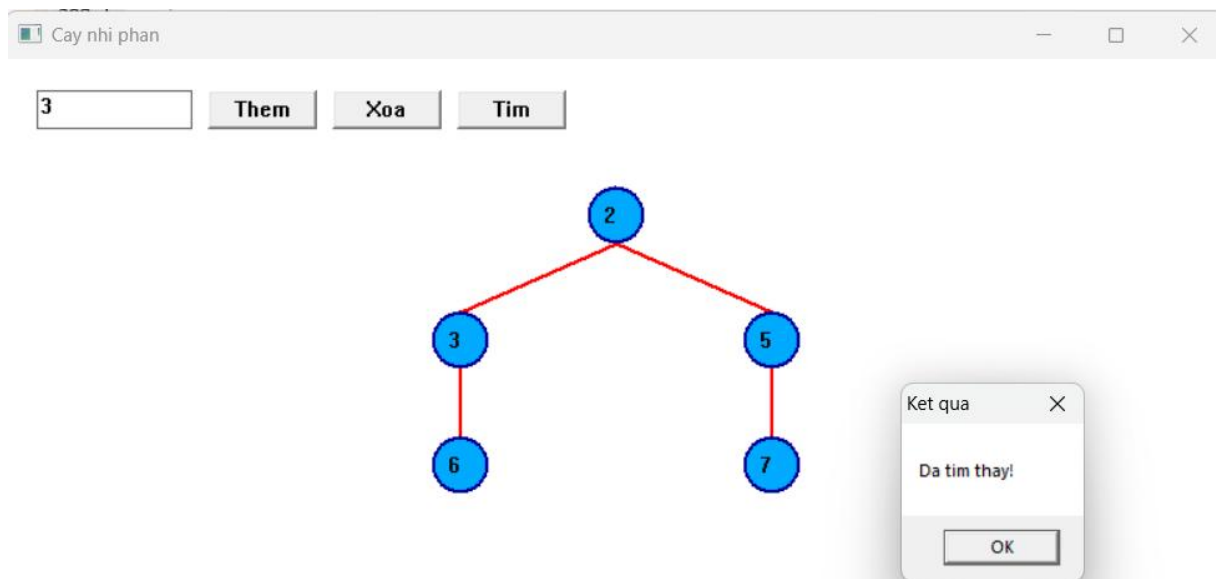


Hình 6. Minh họa kết quả thêm nút

4.1.2 Kết quả chức năng tìm kiếm

Chức năng tìm kiếm hoạt động chính xác theo giá trị người dùng nhập vào. Khi giá trị tồn tại trong cây, chương trình thông báo tìm thấy nút tương ứng. Ngược lại, nếu giá trị không tồn tại, chương trình sẽ thông báo không tìm thấy.

Qua quá trình kiểm tra với nhiều giá trị khác nhau, chức năng tìm kiếm cho kết quả đúng, phản hồi nhanh và không xảy ra lỗi trong quá trình thực hiện.



Hình 7. Minh họa kết quả tìm nút

4.1.3 Kết quả chức năng xóa nút

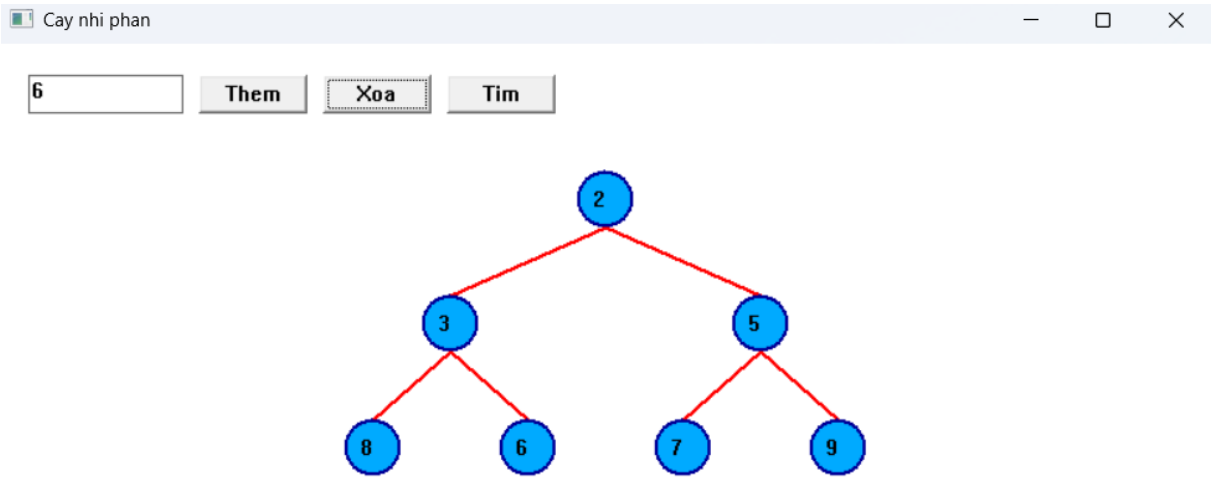
Chức năng xóa nút đã xử lý được các trường hợp cơ bản như:

Xóa nút lá.

Xóa nút có một nút con.

Xóa nút có hai nút con.

Sau khi xóa, cấu trúc cây được cập nhật lại và hiển thị đúng trên giao diện. Điều này cho thấy thuật toán xóa được cài đặt phù hợp và đảm bảo tính đúng đắn của cây nhị phân tìm kiếm.



Hình 8. Minh họa kết quả xóa nút

STT	Dữ liệu đầu vào	Thao tác thực hiện	Kết quả mong đợi	Kết quả đạt được
1	50, 30, 70, 20, 40	Thêm các nút vào cây	Cây BST được tạo đúng thứ tự	Đúng
2	40	Tìm kiếm nút	Tìm thấy nút có giá trị 40	Đúng
3	90	Tìm kiếm nút	Không tìm thấy nút	Đúng
4	20	Xóa nút lá	Nút 20 bị xóa khỏi cây	Đúng
5	30	Xóa nút có 1 con	Cây vẫn đảm bảo tính BST	Đúng
6	50	Xóa nút có 2 con (nút gốc)	Thay thế bằng nút phù hợp	Đúng

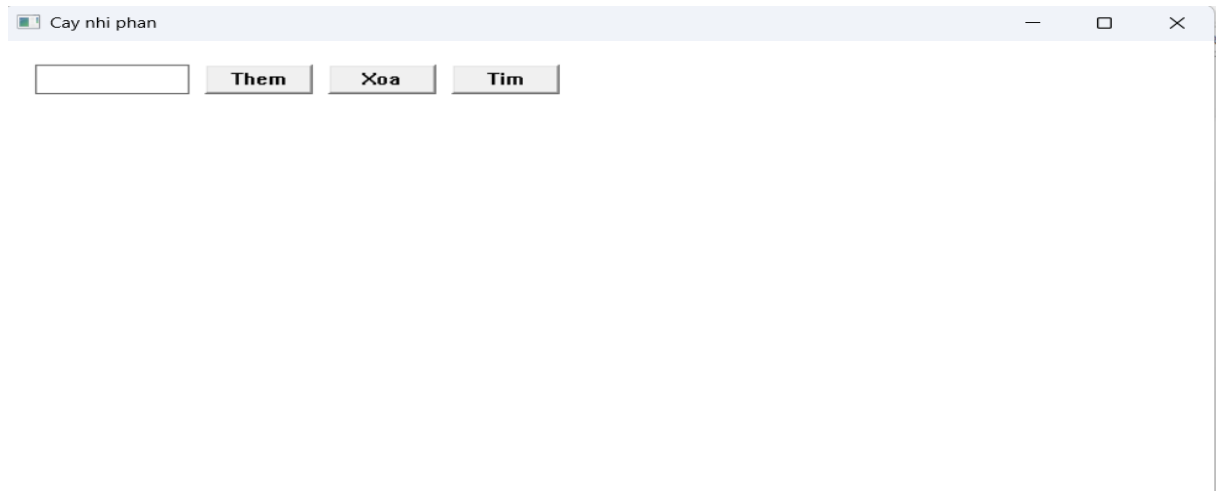
Bảng 5. Các trường hợp kiểm thử chương trình mô phỏng cây nhị phân tìm kiếm

4.2 Kết quả hiển thị và giao diện chương trình

4.2.1 Giao diện tổng thể

Giao diện chương trình được thiết kế đơn giản, dễ sử dụng, phù hợp với mục đích học tập. Người dùng có thể dễ dàng nhập giá trị và lựa chọn các thao tác thông qua các nút chức năng.

Vùng hiển thị cây được bố trí rõ ràng, giúp quan sát trực quan cấu trúc cây sau mỗi thao tác.

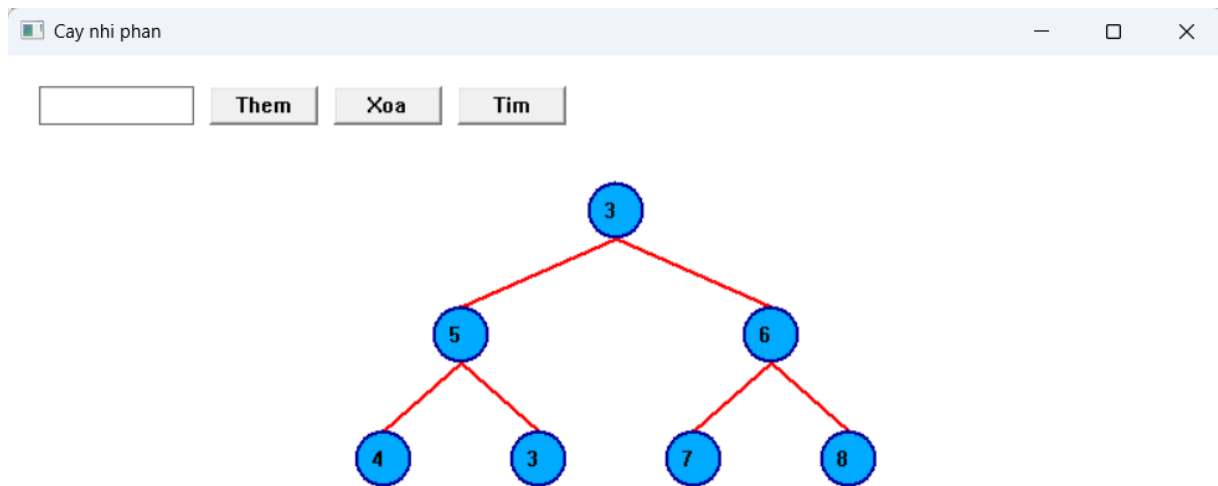


Hình 9. Minh họa kết quả giao diện hiển thị

4.2.2 Kết quả hiển thị cây nhị phân tìm kiếm

Cây nhị phân tìm kiếm được hiển thị dưới dạng sơ đồ, trong đó mỗi nút được biểu diễn bằng hình tròn chứa giá trị và các nhánh được nối bằng đường thẳng. Cách hiển thị này giúp người dùng dễ dàng nhận biết mối quan hệ giữa các nút.

Sau mỗi thao tác thêm, xóa hoặc tìm kiếm, cây được vẽ lại ngay lập tức, đảm bảo tính đồng bộ giữa dữ liệu và giao diện hiển thị.



Hình 10. Minh họa kết quả hiển thị cây

4.2.3 Tính trực quan của chương trình

Thông qua việc hiển thị trực tiếp cấu trúc cây, chương trình giúp người học dễ dàng hình dung cách tổ chức dữ liệu trong cây nhị phân tìm kiếm. Đây là ưu điểm lớn so với việc chỉ quan sát kết quả thông qua dòng lệnh hoặc văn bản.

4.3 Đánh giá hiệu năng và trải nghiệm người dùng

4.3.1 Đánh giá hiệu năng xử lý

Trong quá trình thử nghiệm với số lượng nút ở mức vừa phải, chương trình hoạt động ổn định, thời gian xử lý nhanh và không có hiện tượng giật, lag. Các thao tác thêm, xóa và tìm kiếm được thực hiện gần như tức thời.

Điều này cho thấy chương trình đáp ứng tốt yêu cầu về hiệu năng đối với mục đích mô phỏng và học tập.

4.3.2 Trải nghiệm người dùng

Người dùng có thể thao tác dễ dàng mà không cần kiến thức chuyên sâu về hệ thống. Các thông báo kết quả rõ ràng giúp người dùng hiểu được trạng thái thao tác vừa thực hiện.

Việc sử dụng giao diện trực quan giúp giảm bớt khó khăn trong việc tiếp cận các khái niệm trừu tượng của cấu trúc dữ liệu.

4.3.3 Độ ổn định của chương trình

Qua quá trình sử dụng, chương trình không xuất hiện lỗi nghiêm trọng. Các thao tác liên tục vẫn đảm bảo chương trình chạy ổn định, chứng tỏ việc quản lý bộ nhớ và xử lý dữ liệu được thực hiện đúng cách.

4.4 Nhận xét chung về kết quả nghiên cứu

4.4.1 Mức độ đáp ứng mục tiêu đề tài

Các kết quả đạt được cho thấy chương trình đã đáp ứng đầy đủ mục tiêu đề ra ban đầu là mô phỏng cây nhị phân tìm kiếm và hỗ trợ các thao tác cơ bản trên cây. Nội dung nghiên cứu đã được hiện thực hóa thành một sản phẩm phần mềm có thể sử dụng trong học tập.

4.4.2 Ưu điểm của chương trình

Một số ưu điểm nổi bật:

Thực hiện đúng các thao tác cơ bản của cây nhị phân tìm kiếm.

Giao diện trực quan, dễ sử dụng.

Phù hợp với sinh viên trong quá trình học cấu trúc dữ liệu.

4.4.3 Hạn chế còn tồn tại

Bên cạnh những kết quả đạt được, chương trình vẫn còn một số hạn chế như:

Chưa hỗ trợ cây cân bằng.

Giao diện còn đơn giản, chưa có nhiều tùy chọn mở rộng.

Chưa có chức năng lưu và tải dữ liệu cây.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Qua quá trình thực hiện đồ án, em đã nghiên cứu và xây dựng thành công chương trình mô phỏng cây nhị phân tìm kiếm bằng ngôn ngữ C. Chương trình đáp ứng đầy đủ các chức năng cơ bản như thêm nút, xóa nút và tìm kiếm nút trong cây, đảm bảo đúng nguyên lý hoạt động của cấu trúc cây nhị phân tìm kiếm. Kết quả nghiên cứu giúp minh họa rõ ràng mối quan hệ giữa các nút trong cây, hỗ trợ người học hiểu sâu hơn về cách tổ chức và xử lý dữ liệu bằng cấu trúc cây. Đề tài góp phần củng cố kiến thức lý thuyết đã học và rèn luyện kỹ năng phân tích, thiết kế cũng như cài đặt chương trình, đáp ứng được mục tiêu ban đầu của đồ án cơ sở ngành.

5.2 Hướng phát triển

Bên cạnh những kết quả đạt được, đề tài vẫn còn nhiều khả năng mở rộng và phát triển trong tương lai. Chương trình có thể được nâng cấp để hỗ trợ các loại cây nâng cao như cây nhị phân cân bằng, cây AVL hoặc cây đỏ–đen nhằm cải thiện hiệu năng tìm kiếm. Ngoài ra, giao diện người dùng có thể được cải tiến trực quan và sinh động hơn, bổ sung các chức năng lưu và tải dữ liệu cây để phục vụ cho việc học tập lâu dài. Việc tối ưu thuật toán và mở rộng chương trình sang các nền tảng khác cũng là những hướng phát triển tiềm năng nhằm nâng cao giá trị ứng dụng của đề tài trong giảng dạy và nghiên cứu.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Nguyễn Văn A, *Cấu trúc dữ liệu và giải thuật*, Nhà xuất bản Giáo dục Việt Nam, Hà Nội, 2020.
- [2] Lê Minh Hoàng, *Giáo trình Cấu trúc dữ liệu*, Nhà xuất bản Đại học Quốc gia TP. Hồ Chí Minh, 2019.
- [3] Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed, *Fundamentals of Data Structures in C*, Computer Science Press, 2018.
- [4] Kernighan, B. W., Ritchie, D. M., *The C Programming Language*, Prentice Hall, 2nd Edition, 1988.
- [5] Mark Allen Weiss, *Data Structures and Algorithm Analysis in C*, Pearson Education, 2014.
- [6] Trường Đại học Trà Vinh, *Bài giảng Cấu trúc dữ liệu và giải thuật*, Khoa Kỹ thuật và Công nghệ, 2024.
- [7] GeeksforGeeks, *Binary Search Tree – BST*, tài liệu trực tuyến, truy cập năm 2025.

PHỤ LỤC A

MÃ NGUỒN CHƯƠNG TRÌNH MÔ PHỎNG CÂY NHỊ PHÂN (NGÔN NGỮ C)

1.1. Mục đích của phụ lục

Phụ lục này trình bày toàn bộ mã nguồn chương trình mô phỏng cấu trúc **cây nhị phân** được xây dựng bằng **ngôn ngữ C**. Mã nguồn giúp người đọc hiểu rõ cách cài đặt, tổ chức dữ liệu và cơ chế hoạt động của chương trình.

1.2. Cấu trúc dữ liệu sử dụng

Chương trình sử dụng cấu trúc Node để biểu diễn mỗi nút trong cây nhị phân.

key: giá trị của nút

childCount: số lượng nút con hiện có

children[2]: mảng con trỏ đến tối đa 2 nút con (cây nhị phân)

```
typedef struct Node {  
    int key;  
    int childCount;  
    struct Node* children[2];  
} Node;
```

1.3. Các chức năng chính của chương trình

1.3.1. Hàm tạo node

Hàm create_node() dùng để cấp phát bộ nhớ và khởi tạo một nút mới cho cây.

Gán giá trị cho node

Khởi tạo số lượng con bằng 0

Các con trỏ con ban đầu là NULL

1.3.2. Hàm thêm node

Hàm insert_node() thực hiện thêm một node mới vào cây:

Nếu cây rỗng → tạo node gốc

Nếu node chưa đủ 2 con → thêm trực tiếp

Nếu đã đủ → tiếp tục chèn vào nhánh có ít node con hơn

Giúp cây được phân bố tương đối cân bằng.

1.3.3. Hàm tìm kiếm

Hàm `search_node()` sử dụng **đệ quy** để tìm giá trị trong cây.

Trả về 1 nếu tìm thấy

Trả về 0 nếu không tồn tại

1.3.4. Hàm xóa node

Hàm `delete_node()` thực hiện:

Tìm node có khóa cần xóa

Giải phóng bộ nhớ

Dịch chuyển các node con còn lại

1.4. Chức năng vẽ cây

Chương trình sử dụng **GDI (Graphics Device Interface)** để vẽ cây:

Node được vẽ bằng hình tròn **màu xanh**

Các nhánh nối giữa các node có **màu đỏ**

Giá trị node được hiển thị ở trung tâm

Hàm chính:

`draw_node_recursive()`

1.5. Giao diện chương trình

Giao diện gồm:

Ô nhập giá trị

Nút **Thêm**

Nút **Xóa**

Nút **Tìm**

Các thao tác được xử lý thông qua sự kiện WM_COMMAND.

1.6. Hàm xử lý cửa sổ

Hàm WndProc() đảm nhiệm:

Xử lý sự kiện nút bấm

Vẽ lại cây khi có thay đổi

Quản lý vòng đời cửa sổ

1.7. Hàm main

Hàm WinMain():

Đăng ký lớp cửa sổ

Tạo cửa sổ chính

Duy trì vòng lặp xử lý thông điệp

1.8. Nhận xét

Chương trình đã:

Mô phỏng trực quan cấu trúc cây nhị phân

Hỗ trợ thêm – xóa – tìm kiếm

Giao diện đơn giản, dễ sử dụng

Phù hợp cho mục đích học tập và minh họa cấu trúc dữ liệu