



TRƯỜNG ĐẠI HỌC KINH TẾ TP. HỒ CHÍ MINH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

BÁO CÁO



Môn học:

Cấu trúc dữ liệu và Giải thuật

Giảng viên hướng dẫn: ĐẶNG NGỌC HOÀNG THÀNH

Lớp:

Khoa học máy tính CS0001

Sinh viên thực hiện:

ĐINH HOÀNG NHÃ	-	31241022351
NGUYỄN THÀNH NAM	-	31241020817
LÊ HỮU NGHĨA	-	31241021147
NGUYỄN ĐỨC THÀNH	-	31241022260

MỤC LỤC

Chương 1. GIỚI THIỆU CHUNG	4
I. Lời Mở Đầu.....	4
II. Lý do chọn đề tài	5
III. Phương hướng tiếp cận.....	6
IV. Lời cảm ơn	7
CHƯƠNG 2: CÂY NHỊ PHÂN TÌM KIẾM.....	8
I. Giới Thiệu Chung.....	8
1. Cây nhị phân tìm kiếm là gì	8
2. Cấu trúc Cây BST.....	8
3. Các thao tác cơ bản trong cây BST	9
II. Các thuật toán trên cây BST trong định danh điện tử.....	9
1. Thuật Toán Chèn Thông Tin Người Dùng	9
2. Thuật toán tìm kiếm	10
3. Thuật Toán Xoá.....	10
4. Tổng Kết	10
CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG ĐỊNH DANH ĐIỆN TỬ	12
I. Định danh điện tử là gì ?	12
1. Bài toán định danh điện tử.....	12
2. Cây BST trong định danh điện tử.....	13
II. Sơ Đồ Lớp (Class Diagram) và cài đặt các lớp trong hệ thống	13
1. Tổng Quan.....	13
2. Các lớp chính trong hệ thống và chức năng	13
III. Mô tả thuật toán và mối quan hệ giữa các lớp	17
1. Mô tả thuật toán.....	17
2. Mô tả mối liên hệ giữa các lớp	18
3. Kết Luận	18
CHƯƠNG 3: THIẾT KẾ GIAO DIỆN ĐỊNH DANH ĐIỆN TỬ	19
I. Tổng Quan.....	19
II. Thiết Kế	19
1. Đăng nhập và xác thực	19
2. Giao diện tạo tài khoản.....	20
3. Giao diện định danh	21

CHƯƠNG 4: THẢO LUẬN VÀ ĐÁNH GIÁ 23

I. Kết quả đạt được 23

II. Hạn chế và tồn tại 23

III. Hướng phát triển..... 23

Chương 1. GIỚI THIỆU CHUNG

I. Lời Mở Đầu

Trong thời đại công nghệ phát triển mạnh mẽ như hiện nay, khoa học máy tính không chỉ là một lĩnh vực nghiên cứu chuyên sâu mà còn là nền tảng cho hầu hết các hệ thống hiện đại – từ ứng dụng di động, trí tuệ nhân tạo đến quản lý thông tin và bảo mật dữ liệu. Một trong những môn học cốt lõi, giữ vai trò quan trọng trong chương trình đào tạo ngành này chính là **Cấu trúc dữ liệu và Giải thuật**. Môn học này không chỉ cung cấp kiến thức lý thuyết mà còn rèn luyện cho sinh viên tư duy logic, khả năng phân tích và giải quyết vấn đề một cách tối ưu. Trong đó, các cấu trúc dữ liệu như cây nhị phân tìm kiếm (Binary Search Tree – BST) đóng vai trò đặc biệt trong việc lưu trữ và truy xuất dữ liệu hiệu quả.

Song song với sự phát triển của khoa học dữ liệu và công nghệ bảo mật, **định danh điện tử** đang trở thành xu hướng tất yếu trong quản lý thông tin cá nhân. Đây là phương thức xác thực người dùng thông qua các yếu tố điện tử như tên đăng nhập, mật khẩu, mã OTP, hay dữ liệu sinh trắc học.

Khi kết hợp với các cấu trúc dữ liệu hiệu quả như BST, hệ thống không chỉ có khả năng xác thực nhanh chóng mà còn tối ưu hóa việc tìm kiếm và quản lý tài khoản người dùng. Bài viết này sẽ làm rõ mối liên hệ giữa lý thuyết cấu trúc dữ liệu – giải thuật với thực tiễn xây dựng hệ thống định danh điện tử, qua đó cho thấy vai trò to lớn của kiến thức học thuật trong việc giải quyết các bài toán công nghệ thực tiễn.

II. Lý do chọn đề tài

Trong thời đại chuyển đổi số mạnh mẽ như hiện nay, nhu cầu định danh và xác thực người dùng trên các hệ thống điện tử ngày càng trở nên quan trọng và phổ biến. Định danh điện tử không chỉ giúp người dùng truy cập an toàn vào hệ thống, mà còn đóng vai trò then chốt trong việc quản lý dữ liệu cá nhân, đảm bảo tính bảo mật và nâng cao trải nghiệm người dùng.

Hiện nay, đa số các hệ thống định danh sử dụng cơ chế xác thực truyền thống như kết hợp tên đăng nhập và mật khẩu. Tuy nhiên, khi số lượng người dùng tăng lên, việc lưu trữ và truy xuất thông tin người dùng một cách nhanh chóng, hiệu quả là một thách thức lớn. Xuất phát từ thực tế đó, đề tài này nghiên cứu và xây dựng ứng dụng định danh điện tử sử dụng **cấu trúc dữ liệu cây nhị phân tìm kiếm (Binary Search Tree - BST)** để lưu trữ và tìm kiếm thông tin tài khoản. Cây nhị phân không chỉ giúp tăng tốc độ truy xuất mà còn hỗ trợ tổ chức dữ liệu khoa học, dễ dàng kiểm soát, mở rộng và tối ưu hóa thao tác tìm kiếm trong hệ thống.

Bên cạnh đó, đề tài được triển khai trên nền tảng **WinForms**, một công nghệ giao diện đồ họa phổ biến trong lập trình ứng dụng Windows, nhằm tạo ra một giao diện trực quan, thân thiện với người dùng. WinForms không chỉ hỗ trợ tốt cho việc hiển thị dữ liệu mà còn giúp sinh viên rèn luyện kỹ năng thiết kế giao diện người dùng và xử lý sự kiện trong ứng dụng thực tế.

Với mục tiêu kết hợp giữa thuật toán và giao diện ứng dụng, đề tài "Ứng dụng cây nhị phân trong định danh điện tử" không chỉ có ý nghĩa thực tiễn trong việc nâng cao hiệu quả xác thực người dùng, mà còn là cơ hội để sinh viên vận dụng kiến thức về cấu trúc dữ liệu, lập trình hướng đối tượng và phát triển phần mềm vào một bài toán thực tế có tính ứng dụng cao.

III. Phương hướng tiếp cận

Để đảm bảo tính logic, khoa học và hiệu quả trong quá trình thực hiện đề tài, nhóm đã thống nhất triển khai nội dung các chương theo hai phần chính:

- **Cây nhị phân**
- **Ứng dụng thực tiễn:** Vận dụng các kiến thức đã học để phân tích yêu cầu, thiết kế hệ thống và lập trình giao diện ứng dụng định danh điện tử. Phần này gắn liền với các câu hỏi, nhiệm vụ cụ thể được giảng viên hướng dẫn, và được triển khai theo từng chương như sau:
 - **Chương I:** Trình bày tổng quan về cây nhị phân tìm kiếm (BST) – một cấu trúc dữ liệu quan trọng giúp tổ chức, lưu trữ và truy xuất dữ liệu hiệu quả. BST tuân thủ nguyên tắc: nút trái có giá trị nhỏ hơn, nút phải có giá trị lớn hơn nút cha. Các thao tác cơ bản gồm chèn, tìm kiếm, xóa và duyệt cây. Ba thuật toán chính – **Insert**, **Search** và **Delete** – được ứng dụng trực tiếp để quản lý thông tin tài khoản người dùng trong hệ thống định danh điện tử. Nhờ đặc tính sắp xếp theo ID, BST giúp tăng tốc độ truy xuất và kiểm tra thông tin. Tuy nhiên, hiệu quả thuật toán phụ thuộc vào độ cân bằng của cây, ảnh hưởng đến độ phức tạp khi xử lý dữ liệu lớn.
 - **Chương II:** Trình bày hệ thống định danh điện tử dùng cây nhị phân tìm kiếm (BST) để lưu trữ và xử lý thông tin người dùng. Mỗi tài khoản có một mã ID duy nhất giúp xác minh, truy xuất và bảo mật dữ liệu. Lớp `TreeNode` lưu thông tin người dùng, còn lớp `CayNhịPhan` thực hiện các thao tác chèn, tìm kiếm và xóa. Nhờ cấu trúc BST, hệ thống cho hiệu suất cao và dễ mở rộng khi số lượng tài khoản lớn.
 - **Chương III:** Trình bày giao diện người dùng của hệ thống định danh điện tử trên nền tảng WinForms. Giao diện gồm ba phần chính: **Đăng nhập**, **Đăng ký**, và **Quản lý thông tin định danh**. Người dùng có thể đăng nhập bằng ID và mật khẩu, đăng ký tài khoản mới với đầy đủ thông tin cá nhân, và sau khi xác thực, có thể xem, chỉnh sửa hoặc xóa thông tin. Hệ thống kết nối trực tiếp với cấu trúc cây nhị phân tìm kiếm để xử lý dữ liệu nhanh chóng và an toàn.
- Cách tiếp cận này không chỉ giúp nhóm bám sát các yêu cầu đặt ra trong đề tài, mà còn bảo đảm tính hệ thống, dễ theo dõi và thuận tiện cho việc kiểm tra, đánh giá trong quá trình thực hiện.

IV. Lời cảm ơn

Lời đầu tiên, chúng em xin cảm ơn Đại học Kinh tế TP.Hồ Chí Minh đã cho chúng em cơ hội được học tiếp xúc và tiếp thu kiến thức về bộ môn Cấu trúc dữ liệu và giải thuật, một môn học vô cùng thú vị và bổ ích vì nó đã giúp em có được thêm những kinh nghiệm, bài học mà trước đây chưa từng được giảng dạy, đặc biệt là với sinh viên năm nhất như chúng em. Đặc biệt, chúng em cũng xin chân thành cảm ơn người đã trực tiếp giảng dạy môn này đó là thầy Đặng Ngọc Hoàng Thành. Cảm ơn thầy đã ở bên chúng em trong suốt thời gian qua, nhiệt tình giúp đỡ, giảng dạy, giúp chúng em tiếp thu những kiến thức mới cực kì bổ ích thông qua những bài giảng đầy tâm huyết. Những kiến thức ấy cực kỳ quý báu đối với chúng em trên con đường học tập và cũng là hành trang tuyệt vời đồng hành cùng chúng em trên hành trình tương lai phía trước.

Trân trọng!

CHƯƠNG 2: CÂY NHỊ PHÂN TÌM KIẾM

I. Giới Thiệu Chung

1. Cây nhị phân tìm kiếm là gì

Cây nhị phân tìm kiếm (Binary Search Tree - BST) là một cấu trúc dữ liệu dạng cây nhị phân, trong đó mỗi nút có tối đa hai nút con: nút con trái và nút con phải. BST được thiết kế dựa trên một nguyên tắc cốt lõi:

Tính chất của BST:

Mọi nút trong cây con bên trái của một nút cha có giá trị nhỏ hơn giá trị của nút cha. Mọi nút trong cây con bên phải của một nút cha có giá trị lớn hơn giá trị của nút cha. Không có giá trị trùng lặp (hoặc nếu có, thường được quy định nằm ở một bên, ví dụ: nhỏ hơn hoặc bằng nằm bên trái).

Một số ứng dụng của BST:

Tổ chức và quản lý dữ liệu: BST giúp lưu trữ và truy xuất dữ liệu một cách nhanh chóng và hiệu quả.

Hệ thống tệp và cơ sở dữ liệu: BST được sử dụng để xây dựng các cấu trúc dữ liệu như bảng băm, cây chỉ mục, giúp tăng tốc độ truy vấn dữ liệu.

Thuật toán tìm kiếm và sắp xếp: BST là nền tảng cho nhiều thuật toán tìm kiếm và sắp xếp, như tìm kiếm nhị phân, sắp xếp cây.

2. Cấu trúc Cây BST

Một cây BST gồm các phần tử gọi là node, trong đó mỗi node bao gồm: Giá trị dữ liệu (data), liên kết đến node con trái, liên kết đến node con phải. Node đầu tiên của cây gọi là gốc (root), từ đó phân nhánh thành các node con bên trái và bên phải.

3. Các thao tác cơ bản trong cây BST

Thêm phần tử: So sánh giá trị cần thêm với node hiện tại. Nếu nhỏ hơn, chuyển sang nhánh trái. Nếu lớn hơn, chuyển sang nhánh phải. Lặp lại cho đến khi tìm được vị trí phù hợp để thêm vào.

Tìm Kiếm Phần Tử: Tương tự thao tác thêm, so sánh giá trị cần tìm với node hiện tại, di chuyển sang trái hoặc phải tùy theo giá trị. Dừng khi tìm thấy hoặc đến node rỗng.

Xoá Phần Tử: Phần này có 3 trường hợp, đầu tiên nếu node cần xóa là lá (không có con) ta có thể xóa trực tiếp, nếu node cần xóa có một con thay thế node đó bằng node con, và cuối cùng nếu node cần xóa có hai con tìm node kế cận (in-order successor hoặc predecessor) để thay thế.

Duyệt cây: Có ba cách phổ biến: In-order (trái – gốc – phải) trả kết quả theo thứ tự tăng dần. Pre-order (gốc – trái – phải) thường dùng để sao chép cây. Post-order (trái – phải – gốc) thường dùng để xóa toàn bộ cây.

II. Các thuật toán trên cây BST trong định danh điện tử

1. Thuật Toán Chèn Thông Tin Người Dùng

Mô tả thuật toán

Thuật toán chèn (Insert) trong cây nhị phân tìm kiếm (BST) giúp thêm một tài khoản mới vào hệ thống định danh điện tử. Mỗi tài khoản bao gồm ID (mã định danh), Họ tên, Email, và được lưu trữ dưới dạng một nút trong cây BST.

Bước thực hiện:

Nếu cây rỗng, tạo một nút mới làm gốc.

Nếu cây không rỗng: Nếu ID của tài khoản nhỏ hơn ID của nút hiện tại, di chuyển sang cây con bên trái. Nếu ID của tài khoản lớn hơn ID của nút hiện tại, di chuyển sang cây con bên phải. Khi đến vị trí thích hợp (node con trống), thêm tài khoản mới.

Đánh giá độ phức tạp: $O(\log n)$ (trong cây cân bằng), $O(n)$ (trong cây mất cân bằng).

2. Thuật toán tìm kiếm

Mô tả thuật toán

Thuật toán tìm kiếm (Search) giúp kiểm tra xem một tài khoản có tồn tại trong hệ thống không, dựa vào ID định danh.

Bước thực hiện:

Đầu tiên ta bắt đầu từ gốc của cây. So sánh ID cần tìm với ID của nút hiện tại: Nếu bằng, trả về thông tin người dùng. Nếu nhỏ hơn, tiếp tục tìm kiếm ở cây con bên trái. Nếu lớn hơn, tiếp tục tìm kiếm ở cây con bên phải. Nếu duyệt đến một nút rỗng, kết luận tài khoản không tồn tại.

Đánh giá độ phức tạp: $O(\log n)$ (cây cân bằng), $O(n)$ (cây mất cân bằng).

3. Thuật Toán Xóa

Mô tả thuật toán

Thuật toán xóa (Delete) dùng để loại bỏ một tài khoản khỏi hệ thống. Khi xóa một nút khỏi cây BST, có ba trường hợp xảy ra:

Bước thực hiện: Đầu tiên tìm nút cần xóa. Nếu ID nhỏ hơn, tiếp tục tìm ở cây con bên trái. Nếu ID lớn hơn, tiếp tục tìm ở cây con bên phải. Xử lý các trường hợp xóa:

Trường hợp 1: Nút cần xóa là lá, xóa trực tiếp.

Trường hợp 2: Nút cần xóa có một con, nối con với cha.

Trường hợp 3: Nút cần xóa có hai con ta có thể tìm nút nhỏ nhất bên phải (hoặc lớn nhất bên trái), thay thế ID, sau đó xóa nút thay thế.

4. Tổng Kết

Ba thuật toán cơ bản: **Chèn (Insert)**, **Tìm kiếm (Search)** và **Xóa (Delete)** đóng vai trò then chốt trong việc quản lý và truy xuất thông tin tài khoản trên hệ thống định danh điện tử thông qua cấu trúc **cây nhị phân tìm kiếm (BST)**.

Nhờ đặc tính sắp xếp theo ID, BST cho phép thao tác với dữ liệu một cách hiệu quả, đảm bảo truy xuất nhanh và chính xác. Tuy nhiên, hiệu suất của các thuật toán này phụ thuộc vào độ cân bằng của cây. Trong trường hợp cây bị mất cân bằng (ví dụ: nghiêng hoàn toàn về một phía), độ phức tạp có thể tăng lên đến $O(n)$, làm giảm hiệu quả hệ thống.

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

ĐỊNH DANH ĐIỆN TỬ

I. Định danh điện tử là gì ?

Định danh điện tử (electronic identification - eID) là quá trình xác minh danh tính của một cá nhân hoặc một tổ chức trong môi trường kỹ thuật số. Ví dụ: Căn cước công dân điện tử, tài khoản ngân hàng số, hệ thống đăng nhập trực....

Mỗi danh tính đều có một mã định danh duy nhất (ID), giúp truy xuất, rà soát thông tin một cách nhanh chóng và tiện lợi.

Ứng dụng: Được sử dụng rộng rãi trong các lĩnh vực như chính phủ điện tử, ngân hàng trực tuyến, thương mại điện tử hoặc các dịch vụ y tế số.

1. Bài toán định danh điện tử

Bài toán định danh điện tử liên quan đến việc xác minh danh tính của người dùng trong các hệ thống số. Một hệ thống định danh điện tử thường yêu cầu mỗi người dùng có một mã số (ID) duy nhất để xác nhận thông tin cá nhân của họ, giúp phục vụ cho nhiều mục đích khác nhau như dịch vụ công, ngân hàng, bảo mật hệ thống, và hơn thế nữa.

Mục tiêu của hệ thống định danh điện tử:

Xác minh danh tính người dùng: Xác định danh tính một cá nhân bằng cách kiểm tra mã ID. Quản lý danh tính hiệu quả: Lưu trữ và truy xuất thông tin cá nhân của người dùng một cách nhanh chóng và chính xác. Bảo mật và bảo vệ thông tin: Đảm bảo rằng thông tin cá nhân được bảo vệ khỏi sự truy cập trái phép và chỉ có thể truy xuất khi nhập đúng ID.

2. Cây BST trong định danh điện tử

Cây tìm kiếm nhị phân (BST) là một cấu trúc dữ liệu lý tưởng để giải quyết bài toán này vì khả năng tổ chức dữ liệu theo cách có thể tìm kiếm, thêm và xóa một cách nhanh chóng (với độ phức tạp trung bình là $O(\log n)$).

Ưu điểm :

BST là một cây nhị phân trong đó mỗi nút có một giá trị và được sắp xếp theo quy tắc: Nút con trái chứa giá trị nhỏ hơn nút cha và nút con phải chứa giá trị lớn hơn nút cha. Vì vậy, Tìm kiếm nhanh chóng với việc lưu trữ thông tin ID của một người dùng theo thứ tự, BST giúp tìm kiếm một ID trong hệ thống chỉ mất thời gian $O(\log n)$ trong trường hợp cây cân bằng. Có thể thêm và sửa nhanh chóng khi thêm hoặc sửa danh tính, BST giúp thực hiện các thao tác này với độ phức tạp thấp nhờ vào cách tổ chức dữ liệu theo quy tắc so sánh. Sắp xếp thông tin: Các dữ liệu (danh tính người dùng) có thể được duyệt theo thứ tự từ nhỏ đến lớn một cách tự động bằng phương thức duyệt cây, giúp quản lý và thống kê thông tin dễ dàng.

II. Sơ Đồ Lớp (Class Diagram) và cài đặt các lớp trong hệ thống

1. Tổng Quan

Sơ đồ lớp (Class Diagram) mô tả mối quan hệ giữa các thành phần chính trong hệ thống, đồng thời phản ánh cách tổ chức và xử lý dữ liệu danh tính một cách logic và hiệu quả.

2. Các lớp chính trong hệ thống và chức năng

Mô tả tổng quan nhiệm vụ của các lớp như sau:

Lớp	Nhiệm Vụ
TreeNode	Lưu trữ dữ liệu của người dùng và liên kết cây
CayNhiPhan	Xử lý cây nhị phân (chèn, tìm, xóa)

a. Lớp TreeNode

```
20 references
public class TreeNode
{
    public int ID;
    public string Ten;
    public string MatKhai;
    public string GioiTinh;
    public string DiaChi;
    public string QuocTich;
    public string QueQuan;
    public DateTime NgaySinh;
    public byte[] anh;
    public TreeNode Left;
    public TreeNode Right;

    1 reference
    public TreeNode(int id, string ten, string matKhai, string gioiTinh,
        string diaChi, string queQuan, string quocTich, DateTime ngaySinh, byte[] anh)
    {
        ID = id;
        Ten = ten;
        MatKhai = matKhai;
        GioiTinh = gioiTinh;
        DiaChi = diaChi;
        QueQuan = queQuan;
        QuocTich = quocTich;
        NgaySinh = ngaySinh;
        this.anh = anh;
        Left = null;
        Right = null;
    }
}
```

Hình. Hàm TreeNode để lưu trữ thông tin người dùng

Lớp Treenode đại diện cho mỗi người dùng trong hệ thống. Mỗi đối tượng thuộc lớp này sẽ chứa toàn bộ thông tin định danh bao gồm:

ID: Khóa để phân biệt người dùng, dùng để sắp xếp trong cây.

Ten: Họ tên người dùng.

MatKhai: Mật khẩu đăng nhập.

ThôngTinCaNhan: Mô tả thông tin cá nhân.

DiaChi: Địa chỉ nơi ở.

Left, Right: Con trỏ đến cây con trái và phải.

Cuối cùng là hàm khởi tạo **public TreeNode(int id, string ten, string matKhai, string thôngTinCaNhan, string diaChi)** với vai trò và nhiệm vụ là gán dữ liệu từ các tham số truyền vào (có thể đến từ bàn phím, file, hay bất kỳ đâu) vào các thuộc tính (biến) của đối tượng TreeNode mới được tạo ra.

b. Lớp CayNhiPhan

```
public class CayNhiPhan
{
    public TreeNode root;

    6 references
    public void ChenThongTin(int id, string ten, string matKhai, string gioiTinh,
        string diaChi, string queQuan, string quocTich, DateTime ngaySinh, byte[] anh)
    {
        root = ThemThongTinRec(root, id, ten, matKhai, gioiTinh, diaChi, queQuan, quocTich, ngaySinh, anh);
    }

    3 references
    private TreeNode ThemThongTinRec(TreeNode node, int id, string ten, string matKhai, string gioiTinh,
        string diaChi, string queQuan, string quocTich, DateTime ngaySinh, byte[] anh)
    {
        if (node == null)
        {
            return new TreeNode(id, ten, matKhai, gioiTinh, diaChi, queQuan, quocTich, ngaySinh, anh);
        }

        int sosanh = id.CompareTo(node.ID); // So sánh ID

        if (sosanh > 0)
        {
            node.Right = ThemThongTinRec(node.Right, id, ten, matKhai, gioiTinh, diaChi, queQuan, quocTich, ngaySinh, anh);
        }
        else if (sosanh < 0)
        {
            node.Left = ThemThongTinRec(node.Left, id, ten, matKhai, gioiTinh, diaChi, queQuan, quocTich, ngaySinh, anh);
        }
        else
        {
            MessageBox.Show("Tài khoản đã tồn tại!", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

        return node;
    }
}
```

Hình. Hàm ChenThongTin gọi đệ quy hàm ThemThongTinRec để chèn thông tin

Hàm ChenThongTin: Đây là hàm chèn một người dùng mới vào đúng vị trí trong cây dựa theo thứ tự ID và gọi hàm đệ quy ThemThongTinRec để chèn dữ liệu vào cây.

Tiếp theo là hàm ThemThongTinRec: Đây hàm đệ quy để thực hiện việc chèn vào đúng vị trí theo cấu trúc cây BST với điều kiện là không được trùng số ID nếu trùng sẽ không cho phép thêm.

```

1 reference
public TreeNode TimKiemThongTin(int id, string matKhai)
{
    return TimKiemThongTinRec(root, id, matKhai);
}

3 references
private TreeNode TimKiemThongTinRec(TreeNode node, int id, string matKhai)
{
    if (node == null) return null;

    int sosanh = id.CompareTo(node.ID); // So sánh ID

    if (sosanh == 0)
    {
        if (node.MatKhai == matKhai)
        {
            return node;
        }
        else
        {
            return null;
        }
    }
    else if (sosanh < 0)
    {
        return TimKiemThongTinRec(node.Left, id, matKhai);
    }
    else
    {
        return TimKiemThongTinRec(node.Right, id, matKhai);
    }
}

```

**Hình. Hàm TimKiemThongTin gọi đệ quy hàm
TimKiemThongTinRec để tìm thông tin**

Tiếp đến là Hàm TimKiemThongTin: Với chức năng là gọi hàm đệ quy
TimKiemThongTinRec để tìm kiếm người dùng dựa trên ID và mật khẩu.

Kế đến là Hàm TimKiemThongTinRec: Đây là hàm đệ quy để tìm một nút có ID
khớp, sau đó kiểm tra MatKhai.

Tiếp Theo Là Hàm XoaTaiKhoan: Có nhiệm vụ là xóa một tài khoản người dùng

theo ID từ cây BST và Gọi hàm **XoaTaiKhoanRec()** xóa đệ quy để xử lý việc xóa từ root.

Cuối cùng là Hàm XoaTaiKhoanRec() và Hàm TimNodeNhoNhat(): Hai hàm này có chức năng lần lượt là đệ quy để xóa node theo ID trong cây và tìm node có ID nhỏ nhất trong cây (hoặc cây con).

III. Mô tả thuật toán và mối quan hệ giữa các lớp

1. Mô tả thuật toán

Thuật toán chèn dữ liệu (ChenThongTin):

Nếu cây rỗng khởi tạo nút mới làm root. Nếu $id < node.ID$ thực hiện đệ quy chèn vào cây con trái. Nếu $id > node.ID$ thực hiện đệ quy chèn vào cây con phải. Nếu $id == node.ID$ chương trình sẽ báo “Tài khoản đã tồn tại” (không chèn trùng ID).

Thuật Toán Tìm Kiếm (TimKiemThongTin):

Bắt đầu từ root, so sánh id nhập vào với node.ID. Nếu bằng tiến hành so sánh mật khẩu tới đây sẽ xảy ra 2 trường hợp như sau nếu trùng sẽ xác thực thành công sẽ trả về node. Ngược lại, nếu sai mật khẩu chương trình sẽ trả về null. Nếu $id < node.ID$ chương trình tiếp tục tìm bên trái. Nếu $id > node.ID$ chương trình sẽ tiếp tục tìm bên phải. Nếu đến nút rỗng vẫn không tìm thấy máy sẽ trả về null.

Thuật Toán Xóa (XoaTaiKhoan):

Đầu tiên bạn cần nhập id và đảm bảo rằng id đã đăng ký trên hệ thống. Nếu node hiện tại là null máy sẽ trả về cây rỗng hoặc không tìm thấy. Tiếp theo tiến hành so sánh id với node.ID :

Nếu $id < node.ID$: Gọi đệ quy để xóa node trong cây con trái: $node.Left = XoaTaiKhoanRec(node.Left, id)$.

Nếu $id > node.ID$: Gọi đệ quy để xóa node trong cây con phải: $node.Right = XoaTaiKhoanRec(node.Right, id)$.

Nếu $id == node.ID$: Đã tìm thấy node cần xóa, tiến hành xử lý.

Khi xóa sẽ xảy ra những trường hợp sau:

Trường hợp 1: Node không có con trái và con phải máy sẽ trả về null (xóa node lá).

Trường hợp 2: Node chỉ có một con trái với trường hợp này trả về node.Left (nối node cha với con trái).

Trường hợp 3: Node chỉ có một con phải
Trả về node.Right (nối node cha với con phải).

Trường hợp 4: Đây là trường hợp rắc rối nhất khi node có cả hai con: Tìm node nhỏ nhất trong cây con phải. Sao chép dữ liệu từ node hiện tại và cuối cùng nút đã được sao chép ra khỏi cây.

2. Mô tả mối liên hệ giữa các lớp

Lớp **CayNhiPhan** và lớp **TreeNode** có mối quan hệ thành phần (composition), trong đó **TreeNode** đóng vai trò là đơn vị cấu thành của cấu trúc dữ liệu cây nhị phân, còn **CayNhiPhan** là lớp quản lý toàn bộ cây.

Lớp **CayNhiPhan** chứa một trường (field) là root – đại diện cho nút gốc của cây, và từ nút gốc này, toàn bộ các **TreeNode** khác được liên kết lại thông qua các thuộc tính Left (trái) và Right (phải) của từng nút.

Mỗi **TreeNode** chứa dữ liệu người dùng và đóng vai trò là nút của cây, còn **CayNhiPhan** chịu trách nhiệm thực hiện các thao tác trên cây như chèn, tìm kiếm, xóa và sửa đổi thông tin, thông qua việc truy cập và thay đổi các **TreeNode** được kết nối với nhau trong cây.

3. Kết Luận

Việc thiết kế hệ thống định danh theo hướng đối tượng với cấu trúc cây tìm kiếm nhị phân cho phép: Tăng hiệu suất truy xuất và xử lý dữ liệu khi số lượng tài khoản người dùng lớn, nhờ khả năng tìm kiếm nhanh theo ID trong cây nhị phân. Tổ chức hệ thống rõ ràng, với mỗi người dùng được biểu diễn bằng một lớp **TreeNode**, và toàn bộ dữ liệu được quản lý bởi lớp **CayNhiPhan**. Phân chia nhiệm vụ rõ ràng: **TreeNode** chịu trách nhiệm lưu trữ thông tin người dùng, còn **CayNhiPhan** đảm nhiệm các thao tác xử lý như thêm, tìm kiếm, và cập nhật thông tin. Thuận tiện trong việc bảo trì, mở rộng, nhờ thiết kế hướng đối tượng dễ tái sử dụng và nâng cấp.

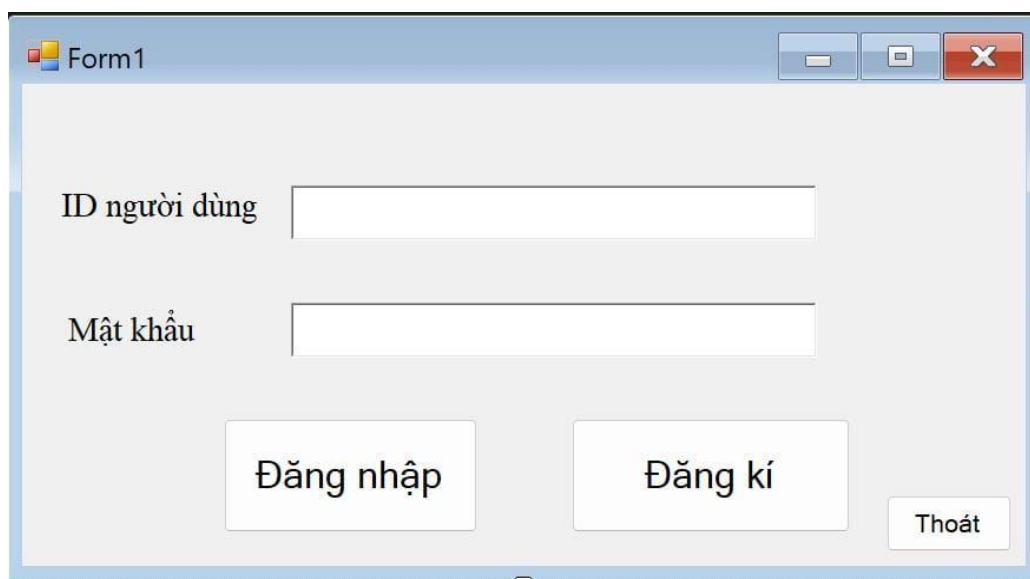
CHƯƠNG 3: THIẾT KẾ GIAO DIỆN ĐỊNH DANH ĐIỆN TỬ

I. Tổng Quan

Hệ thống định danh điện tử được thiết kế theo hướng đối tượng, sử dụng cấu trúc dữ liệu cây nhị phân tìm kiếm (BST) để lưu trữ và quản lý thông tin người dùng một cách hiệu quả. Việc kết hợp giao diện người dùng WinForms với mô hình tổ chức dữ liệu bằng cây BST mang lại trải nghiệm mượt mà, đồng thời đảm bảo khả năng mở rộng và bảo trì hệ thống trong thực tế.

II. Thiết Kế

1. Đăng nhập và xác thực

The image shows a Windows Forms application window titled 'Form1'. Inside the window, there is a login interface. It features two text boxes: the first is labeled 'ID người dùng' (User ID) and the second is labeled 'Mật khẩu' (Password). Below these text boxes, there are three buttons: 'Đăng nhập' (Login), 'Đăng kí' (Register), and 'Thoát' (Exit). The window has a standard Windows title bar with minimize, maximize, and close buttons.

Hình . Giao diện đăng nhập

Trường nhập ID người dùng:

Nhận mã định danh duy nhất của người dùng. Dữ liệu sẽ được gửi để xác thực trong cây nhị phân.

Trường nhập mật khẩu:

Cho phép nhập chuỗi mật khẩu bảo mật. Dùng để so khớp với dữ liệu đã lưu.

Nút “Đăng nhập”:

Khi nhấn, hệ thống sẽ kiểm tra thông tin ID và mật khẩu: Nếu hợp lệ: truy cập thành công. Nếu không: hiển thị thông báo lỗi.

Nút “Đăng kí”:

Mở form mới để thêm tài khoản mới. Người dùng sẽ nhập thông tin và hệ thống sẽ chèn node mới vào cây BST theo ID.

Nút “Thoát”:

Đóng ứng dụng WinForms.

2. Giao diện tạo tài khoản



Formdangki

Tạo tài khoản

Mã định danh

Mật khẩu

Tên người dùng

Giới tính

Ngày sinh

Quê quán Quốc tịch

Địa chỉ

Đăng ký

Giao diện đăng ký

Giao diện "Formdangki" là nơi người dùng nhập thông tin cá nhân để đăng ký một tài khoản định danh điện tử. Giao diện bao gồm:

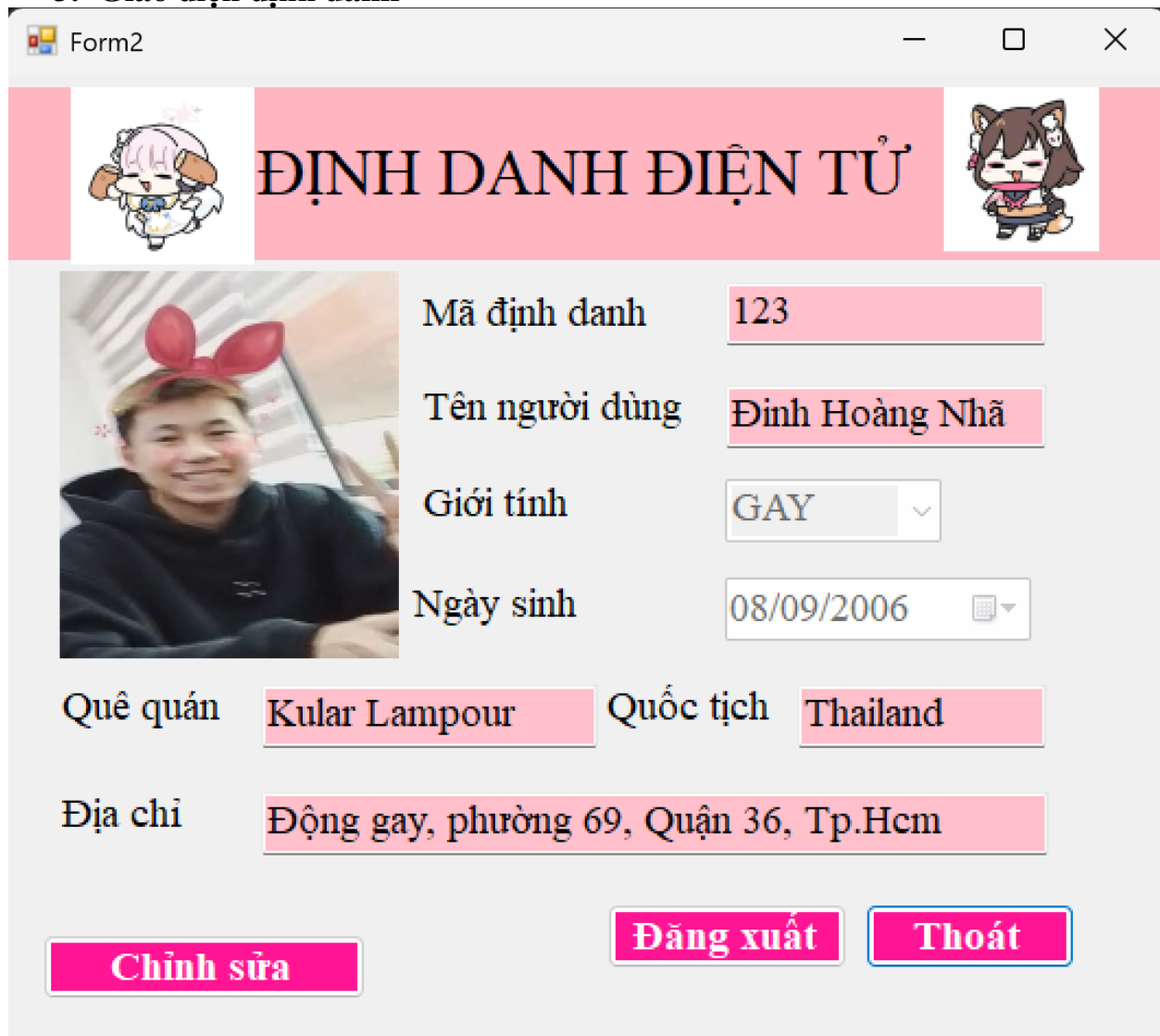
Mã định danh: Trường để nhập ID duy nhất cho người dùng. Mật khẩu: Nhập mật khẩu để bảo vệ tài khoản.

Tên người dùng: Họ tên đầy đủ của người đăng ký. Giới tính: Chọn giới tính từ danh sách (dropdown). Ngày sinh: Chọn ngày sinh bằng DateTimePicker.

Quê quán, Quốc tịch, Địa chỉ: Các trường văn bản để khai báo thông tin cư trú. Ảnh đại diện: Có thể có tùy chọn thêm ảnh đại diện người dùng.

Nút "Xác nhận": Sau khi nhập đầy đủ thông tin, người dùng bấm nút này để hoàn tất đăng ký. Hệ thống kiểm tra và thêm vào cây BST nếu ID chưa tồn tại.

3. Giao diện định danh



The screenshot shows a Windows application window titled "Form2". The main header is pink with the text "ĐỊNH DANH ĐIỆN TỬ" in the center, flanked by two anime-style character avatars. Below the header, the form is organized into several sections:

- Profile Picture:** A placeholder image of a person wearing a red bunny ear headband.
- Registration Fields:**
 - Mã định danh (ID Number): 123
 - Tên người dùng (Username): Đinh Hoàng Nhã
 - Giới tính (Gender): GAY (selected from a dropdown)
 - Ngày sinh (Date of Birth): 08/09/2006 (selected from a date picker)
 - Quê quán (Hometown): Kular Lampour
 - Quốc tịch (Nationality): Thailand
 - Địa chỉ (Address): Động gay, phường 69, Quận 36, Tp.Hcm
- Action Buttons:** Three buttons are located at the bottom: "Chỉnh sửa" (Edit), "Đăng xuất" (Logout), and "Thoát" (Exit).

Giao diện thông tin người dùng

Giao diện “Form2” thể hiện thông tin định danh điện tử sau khi đăng nhập thành công, đồng thời cung cấp chức năng chỉnh sửa thông tin: Các trường hiển thị giống Form đăng ký: Gồm mã định danh, tên người dùng, giới tính, ngày sinh, quê quán, quốc tịch, địa chỉ.

Ảnh đại diện: hiển thị ảnh người dùng.

Nút “Chỉnh sửa”: Cho phép chỉnh sửa thông tin cá nhân.

Nút “Lưu”: Lưu lại thông tin sau khi chỉnh sửa.

Nút “Đăng xuất”: Thoát khỏi phiên làm việc.

Nút “Thoát”: Đóng ứng dụng.

Nút “Xoá”: Xoá thông tin đã lưu.

CHƯƠNG 4: THẢO LUẬN VÀ ĐÁNH GIÁ

I. Kết quả đạt được

Đầu tiên là giúp chúng ta quản lý hiệu quả thông tin người dùng dưới dạng cấu trúc cây nhị phân tìm kiếm (BST), giúp dữ liệu được tổ chức khoa học theo thứ tự ID. Thêm thông tin của người dùng nhanh chóng vào đúng vị trí của cây đảm bảo không trùng lặp id. Cung cấp một lớp bảo mật bằng cách nhập id và mật khẩu để có bảo mật thông tin. Duy trì cấu trúc BST ổn định sau mỗi thao tác chèn hoặc xóa, đảm bảo hiệu năng tìm kiếm cũng như là tác vụ luôn ở mức tối ưu (nếu cây cân bằng).

Và cuối cùng có thể mở rộng thêm các chức năng khác như duyệt cây, in danh sách, bảo mật mạnh hơn bằng cách biến mã hoá,... nếu cần trong tương lai.

II. Hạn chế và tồn tại

Sau những kết quả đạt được khá ổn thì những hạn chế trước mắt như sau:

Đầu tiên là hệ thống chưa hỗ trợ ghi dữ liệu ra file hoặc đọc dữ liệu từ file, nên khi tắt ứng dụng thì toàn bộ dữ liệu sẽ mất. Không có kiểm tra định dạng dữ liệu đầu vào (như ngày sinh, email, số điện thoại), dễ dẫn đến nhập sai dữ liệu. Giao diện còn khá thô sơ, chưa thân thiện với người dùng. Mật khẩu người dùng được lưu dưới dạng thô, không mã hóa dễ gây rò rỉ dữ liệu nếu bị truy cập trái phép. Cây nhị phân chưa được tự cân bằng, nên nếu dữ liệu thêm theo thứ tự tăng hoặc giảm dần thì cây sẽ lệch, gây giảm hiệu suất (giống danh sách liên kết).

III. Hướng phát triển

Nhóm tụi em sẽ phát triển thêm một phiên bản dành cho những người quản lý (phục vụ cho những doanh nghiệp hoặc tổ chức) với các chức năng lưu cây BST ra file (JSON/XML) và đọc lại từ file khi khởi động phần mềm. Phát triển thêm chức năng thống kê dữ liệu: tổng số tài khoản, phân loại theo giới tính, quê quán, quốc tịch. Hỗ trợ tìm kiếm nâng cao: tìm theo tên, quê quán, có hỗ trợ tìm gần đúng. Có thể mở rộng lên giao diện Web hoặc ứng dụng di động sử dụng nền tảng ASP.NET hoặc trên một số nền tảng khác.