**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

Logo

Description automatically generated**KHOA ĐIỆN- ĐIỆN TỬ**

**MÔN HỌC GIẢI THUẬT VÀ CẤU TRÚC DỮ LIỆU**

TỔ CHỨC THƯ VIỆN

|  |  |
| --- | --- |
| SVTH: NHÓM 1 | MSSV: |
| 1. PHẠM HUY HOÀNG | 20119341 |
| 2. CHÂU THÀNH ĐẠT | 20119332 |
| 3. NGUYỄN QUANG BÌNH | 20119063 |
| 4. NGUYỄN THANH DIỆN | 20119329 |

GVHD: HỒ NHỰT MINH

Lớp: Chiều thứ 4 – Tiết 9-12

**NHẬN XÉT CỦA GIÁO VIÊN**

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

*Tp. Hồ Chí Minh, ngày…. tháng….. năm 2023*

Giáo viên hướng dẫn

Hồ Nhựt Minh

**BẢNG PHÂN CHIA CÔNG VIỆC**

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên** | **Công việc thực hiện** |
| 1 | Châu Thành Đạt | - Tổ chức danh sách Danh mục sách  - Giao diện tổ chức Danh mục sách |
| 2 | Nguyễn Quang Bình | - Tổ chức danh sách Độc giả  - Giao diện quản lí độc giả |
| 3 | Nguyễn Thanh Diện | - Tổ chức danh sách Đầu sách  - Giao diện quản lí đầu sách |
| 4 | Phạm Huy Hoàng | - Tổ chức danh sách Mượn trả  - Giao diện quản lí mượn trả sách |

Mục lục

[**1 Tổ chức dữ liệu: 1**](#_Toc134997367)

[**1.1Tổ chức cấu trúc dữ liệu cho Đầu Sách 1**](#_Toc134997368)

[**1.1.1 Tổ chức cấu trúc và giải thuật cho Đầu Sách 1**](#_Toc134997369)

[**1.1.2 Tổ chức cơ sở dữ liệu cho đầu sách 7**](#_Toc134997370)

[**1.2Tổ chức cấu trúc dữ liệu cho DanhMucSach 7**](#_Toc134997371)

[**1.2.1 Tổ chức cấu trúc và giải thuật cho DanhMucSach 7**](#_Toc134997372)

[**1.2.2 Tổ chức dữ liệu cho DanhMucSach 15**](#_Toc134997373)

[**1.3Tổ chức cấu trúc dữ liệu cho DocGia 15**](#_Toc134997374)

[**1.3.1 Tổ chức cấu trúc cho DocGia 15**](#_Toc134997375)

[**1.4 Tổ chức cấu trúc dữ liệu cho MUONTRA 27**](#_Toc134997376)

[**1.4.1 Tổ chức cấu trúc cho MUONTRA 27**](#_Toc134997377)

[**1.4.2 Tổ chức dữ liệu cho MuonTra 32**](#_Toc134997378)

[**2. Giao diện 34**](#_Toc134997379)

[**2.1Giao diện quản lí đầu sách 34**](#_Toc134997380)

[**2.2Giao diện quản lí danh mục sách 44**](#_Toc134997381)

[**2.3 Giao diện quản lí độc giả 48**](#_Toc134997382)

[**2.4 Giao diện phần mượn trả sách: 61**](#_Toc134997383)

# 1 Tổ chức cấu trúc và dữ liệu:

## Tổ chức cấu trúc dữ liệu cho Đầu Sách

### 1.1.1 Tổ chức cấu trúc và giải thuật cho Đầu Sách

\* Tạo đối tượng DauSach bao gồm các kiểu dữ liệu sau:

public class DauSach

{

public string ISBN { get; set; }

public string TenSach { get; set; }

public int SoTrang { get; set; }

public string TacGia { get; set; }

public int NamXB { get; set; }

public string TheLoai { get; set; }

public int soluong;

public DauSach NextBook;

public DanhMucSach SachTrongThuVien = new DanhMucSach();

}

* ISBN: chuỗi định danh duy nhất của sách.
* TenSach: chuỗi tên của sách.
* SoTrang: số trang của sách.
* TacGia: chuỗi tên của tác giả.
* NamXB: số nguyên là năm xuất bản của sách.
* TheLoai: chuỗi thể loại của sách.
* soluong: số lượng sách có trong thư viện.
* NextBook: một tham chiếu đến đối tượng DauSach tiếp theo trong danh sách các đầu sách.
* SachTrongThuVien: một đối tượng DanhMucSach đại diện cho tất cả các bản sao của cuốn sách trong thư viện.

\* Tạo đối tượng Bookfunc để quản lí các đối tượng DauSach: Bookfunce sử dụng cấu trúc là mảng con trỏ

public class BookFunc

{

public static DauSach[] books;

public static List<DauSach> ListDauSach = new List<DauSach>();

public int soluong;

public BookFunc(int size)

{

books = new DauSach[size];

}

public IEnumerator<DauSach> GetEnumerator()

{

return ListDauSach.GetEnumerator();

}

Phương thức GetEnumerator được định nghĩa để cho phép lớp BookFunc có thể duyệt qua danh sách các đối tượng DauSach bằng cách sử dụng một vòng lặp foreach. Nó trả về một đối tượng IEnumerator<DauSach>, cho phép ta duyệt qua danh sách các đối tượng DauSach trong ListDauSach.

\* Các Hàm xử lí trong Bookfunc

**- Hàm GetIndexForNewBook -** Trả về chỉ số của một vị trí trống đầu tiên trong mảng

Giải thuật :

* Khởi tạo biến index bằng 0.
* Lặp qua mảng books, kiểm tra nếu phần tử tại vị trí index không rỗng, thì tăng giá trị của index lên 1 và tiếp tục vòng lặp.
* Nếu phần tử tại vị trí index trong mảng books là rỗng, gán book.NextBook bằng phần tử tiếp theo của mảng (có thể là null hoặc một đối tượng DauSach).
* Trả về giá trị hiện tại của biến index.

Vì các đối tượng DauSach được lưu trữ trong mảng books bằng cách sử dụng các con trỏ NextBook, hàm GetIndexForNewBook sẽ tìm vị trí trống đầu tiên trong mảng để có thể thêm một đối tượng DauSach mới và cập nhật con trỏ NextBook của đối tượng trước đó để trỏ đến đối tượng mới được thêm vào.

public static int GetIndexForNewBook(DauSach book)

{

int index = 0;

while (books[index] != null)

{

index++;

}

book.NextBook = books[index];

return index;

}

**- Hàm AddBook** - Thêm một đối tượng DauSach vào mảng con trỏ:

Giải thuật:

* Gọi hàm GetIndexForNewBook để tìm vị trí trống đầu tiên trong mảng books.
* Gán đối tượng book vào vị trí đó trong mảng books.
* Kết thúc hàm.

Vì vậy, hàm AddBook sẽ tìm vị trí trống đầu tiên trong mảng books bằng cách sử dụng hàm GetIndexForNewBook, sau đó thêm đối tượng book vào vị trí đó bằng cách gán nó cho phần tử tại vị trí đó của mảng.

public void AddBook(DauSach book)

{

int index = GetIndexForNewBook(book);

books[index] = book;

}

**- Hàm GetBookByISBN** - tìm kiếm và trả về đối tượng DauSach dựa trên mã ISBN của sách:

Giải thuật:

* Gọi hàm ReadDataFromFile để đọc dữ liệu từ file vào mảng books.
* Duyệt mảng books bằng vòng lặp foreach.
* Kiểm tra xem phần tử hiện tại trong mảng có khác null và có mã ISBN bằng với isbn không. Nếu có, trả về đối tượng book.
* Nếu không có đối tượng book nào có mã ISBN bằng với isbn, trả về null.
* Vì vậy, hàm GetBookByISBN sẽ trả về đối tượng DauSach có mã ISBN bằng với isbn, hoặc trả về null nếu không tìm thấy đối tượng nào có mã ISBN tương ứng.

public static DauSach GetBookByISBN(string isbn)

{

BookFunc.ReadDataFromFile();

foreach (DauSach book in books)

{

if (book != null && book.ISBN == isbn)

{

return book;

}

}

return null;

}

**- Hàm SortByISBN -** sắp xếp danh sách đầu sách theo ISBN theo thứ tự tăng dần.

Giải thuật: sử dụng thuật toán Bubble Sort

* Lấy danh sách đầu sách từ BookFunc.ListDauSach và lưu vào biến listDauSach.
* Khởi tạo biến n bằng số lượng phần tử trong danh sách listDauSach.
* Sử dụng hai vòng lặp lồng nhau để duyệt qua các phần tử trong danh sách listDauSach.
* So sánh ISBN của hai phần tử liên tiếp nhau, nếu ISBN của phần tử thứ j lớn hơn ISBN của phần tử thứ j+1, thực hiện hoán đổi vị trí hai phần tử.
* Khi vòng lặp bên trong kết thúc, phần tử cuối cùng của danh sách sẽ được sắp xếp đúng vị trí. Ta tăng giá trị của biến i lên 1 và quay lại bước 3 để tiếp tục sắp xếp các phần tử còn lại.
* Khi cả hai vòng lặp kết thúc, danh sách listDauSach sẽ được sắp xếp theo ISBN tăng dần.

public static void SortByISBN()

{

List<DauSach> listDauSach = BookFunc.ListDauSach;

int n = listDauSach.Count;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (listDauSach[j].ISBN.CompareTo(listDauSach[j + 1].ISBN) > 0)

{

// Hoán đổi vị trí hai phần tử

DauSach temp = listDauSach[j];

listDauSach[j] = listDauSach[j + 1];

listDauSach[j + 1] = temp;

}

}

}

}

**- Hàm ReadDataFromFile -** được định nghĩa trong lớp BookFunc. Nó cho phép đọc dữ liệu từ tệp văn bản chứa thông tin về các đầu sách và lưu chúng vào danh sách ListDauSach của lớp.

Thuật toán:

* Xóa toàn bộ phần tử trong danh sách ListDauSach của lớp BookFunc.
* Tạo một đối tượng dausach kiểu BookFunc với kích thước ban đầu là 100.
* Sử dụng lệnh using để mở tệp văn bản "danh-sach-sach.txt" và đọc từng dòng của tệp bằng vòng lặp while.
* Trong mỗi lần lặp, phân tích dòng đọc được thành các trường thông tin về đầu sách bằng cách sử dụng phương thức Split của chuỗi.
* Tạo một đối tượng book kiểu DauSach và thiết lập các thuộc tính của nó từ các trường thông tin phân tích được.
* Thêm đối tượng book vào danh sách dausach bằng cách gọi phương thức AddBook của đối tượng dausach.
* Thêm đối tượng book vào danh sách ListDauSach của lớp BookFunc.
* Đóng tệp văn bản.
* Nếu xảy ra lỗi trong quá trình đọc tệp văn bản, hiển thị thông báo lỗi.

Vì vậy, hàm ReadDataFromFile sẽ đọc thông tin về các đầu sách từ tệp văn bản và lưu chúng vào danh sách ListDauSach của lớp BookFunc.

public static void ReadDataFromFile()

{

BookFunc.ListDauSach.Clear();

BookFunc dausach = new BookFunc(100); // Tạo đối tượng BookFunc với kích thước 100

try

{

using (StreamReader reader = new StreamReader("danh-sach-sach.txt"))

{

string line;

while ((line = reader.ReadLine()) != null)

{

string[] fields = line.Split('|');

DauSach book = new DauSach();

book.ISBN = fields[0];

book.TenSach = fields[1];

book.SoTrang = int.Parse(fields[2]);

book.TacGia = fields[3];

book.NamXB = int.Parse(fields[4]);

book.TheLoai = fields[5];

dausach.AddBook(book); // Thêm sách vào danh sách

ListDauSach.Add(book);

}

reader.Close();

}

}

catch (Exception ex)

{

Console.WriteLine("Loi khi doc file: " + ex.Message);

}

}

**- Hàm SaveDataToFile** cho phép lưu thông tin của một quyển sách vào tệp tin.

* Tạo một chuỗi line để lưu thông tin của quyển sách theo định dạng chuỗi, gồm các thông tin ISBN, Tên sách, Số trang, Tác giả, Năm xuất bản và Thể loại. Chuỗi này sẽ được ghi vào tệp tin.
* Kiểm tra xem chuỗi line có rỗng hay không. Nếu không rỗng, mở tệp tin để ghi vào đó.
* Ghi chuỗi line vào tệp tin.
* Sau khi ghi xong, đóng tệp tin.

Vì vậy, hàm SaveDataToFile sẽ lưu thông tin của một quyển sách vào tệp tin nếu chuỗi line không rỗng.

public static void SaveDataToFile(string filePath, string isbn, string tenSach, string soTrang, string tacGia, string namXB, string theLoai)

{

string line = $"{isbn}|{tenSach}|{soTrang}|{tacGia}|{namXB}|{theLoai}";

if (!string.IsNullOrWhiteSpace(line))

{

using (StreamWriter writer = new StreamWriter(filePath, true))

{

writer.WriteLine(line);

}

}

}

### 1.1.2 Tổ chức cơ sở dữ liệu cho đầu sách

Cấu trúc lưu trữ thông tin đầu sách được tổ chức như sau :

ISBN|TenSach|SoTrang|TacGia|NamXB|TheLoai

A screenshot of a computer

Description automatically generated with low confidence

## Tổ chức cấu trúc dữ liệu cho DanhMucSach

### 1.2.1 Tổ chức cấu trúc và giải thuật cho DanhMucSach

**\* Tạo đối tượng Sach bao gồm các kiểu dữ liệu sau:**

public class Sach

{

public int masach { get; set; }

public int trangthai { get; set; }

public int vitri { get; set; }

public Sach next;

public MuonTra MuonTra;

public Sach(int masach, int trangthai, int vitri, Sach next)

{

this.masach = masach;

this.trangthai = trangthai;

this.vitri = vitri;

this.next = next;

}

}

* MaSach : kiểu int, đại diện cho mã của sách
* TrangThai : kiểu int, 0 là sách có trong kho , 1 là sách đã đucợ mượn
* ViTri : kiểu int, là vị trí của sách trong kho
* MuonTra : kiểu class Muontra cho biết thông tin người mượn ,ngày mượn nếu có

- next: trỏ đến sách tiếp theo trong danh sách liên kết

\* Tạo đối tượng DS\_MuonTra để quản lí các đối tượng MuonTra: DS\_MuonTra sử dụng cấu trúc danh sách liên kết đơn

public class DanhMucSach

{

public Sach head; //node đầu tiên trong danh sách

public Sach tail; //node cuối cùng trong danh sách

public int soluong; //số đối tượng Sach trong danh sách

public DanhMucSach()

{

head = null;

tail = null;

soluong = 0;

}

}

\* Các phương thức xử lí trong DanhMucSach

**- Phương thức IsEmpy():** Kiểm tra số đối tượng MuonTra trong danh sách có bằng 0 hay không

public bool isEmpty()

{

return soluong == 0;

}

Phương thức trả về true nếu số lượng bằng 0 và trả về false nếu ngược lại

**- Phương thức ThemSach:** Thêm các đối tượng Sach vào danh sách

Giải thuật:

* Kiểm tra nếu isEmpty là true thì cho newsach là node head và tail.

+ Nếu không thì gán tail.next là newsach rồi chuyển tail sang cho newsach tức là newsach trỏ thành tail mới.

public void ThemSach(int ms,int tt, int vt)

{

Sach newSach = new Sach(ms, tt, vt, null);

if(isEmpty())

{

head = newSach;

tail = newSach;

}

else

{

tail.next = newSach;

tail = newSach;

}

soluong++;

}

**- Phương thức GetSachBangMaSach:** dùng tìm kiếm Sach trong DanhMucSach bằng mã sách

Giải thuật:

* Tạo vòng while lặp đến khi hết danh sách liên kết.
* Trong vòng lặp nếu tìm thây Sach có masach giống với parameter ms đưa vào thì trả về Sach.
* Khi kết thúc vòng lặp mà không thấy Sach nào thì trả về null.

public Sach GetSachBangMaSach(int MS)

{

Sach newSach = head;

while (newSach != null)

{

if (newSach.masach == MS)

{

return newSach;

}

newSach = newSach.next;

}

return null;

}

**- Phương thức ThemSachKhiNapFile** tương tự ThemSach nhưng có xét điều kiện của trạng thái sách :

public void ThemSachKhiNapFile(Sach sach)

{

Sach newest = new Sach(sach.masach, sach.trangthai, sach.vitri, null);

if (sach.trangthai == 0 || sach.trangthai == 1 || sach.trangthai == 2)

{

if (isEmpty())

head = newest;

else

tail.next = newest;

tail = newest;

soluong++;

}

}

**- Phương thức LinkedListToList** là hàm chuyển đổi LinkedList thành List dùng hiển thị trên DataGridView của Winform.

{

List<Sach> listsach = new List<Sach>();

Sach sach = head; // khởi tạo node sach là tại node head

while (sach != null) // nếu sach chưa chạy hết danh sách

{

listsach.Add(sach);

sach = sach.next;

}

return listsach;

}

**- Phương thức ThemSachKhiNapFile** để đọc dữ liệu từ một tệp văn bản có tên là "danh-muc-sach-ton-kho.txt" và thêm nó vào danh sách liên kết.

Giải thuật :

* Hàm bắt đầu bằng cách mở một đối tượng StreamReader để đọc từ tệp văn bản.
* Hàm sau đó đọc dòng đầu tiên của tệp bằng cách sử dụng phương thức ReadLine () của StreamReader.
* Vòng lặp while được sử dụng để lặp lại từng dòng của tệp cho đến khi đến cuối cùng (tức là phương thức ReadLine () trả về null).
* Trong vòng lặp while, dòng hiện tại được phân tách thành một mảng các trường chuỗi bằng cách sử dụng phương thức Split ().
* Nếu trường đầu tiên của dòng là "DAUSACH", hàm biết rằng các dòng tiếp theo chứa thông tin về sách.
* Trường thứ hai của dòng "DAUSACH" chứa số lượng sách, vì vậy một vòng lặp for được sử dụng để lặp lại từng cuốn sách.
* Trong vòng lặp for, một dòng mới được đọc từ tệp, phân tách thành các trường và các trường được chuyển đổi sang kiểu số nguyên.
* Các số nguyên được chuyển đổi được chuyển làm đối số cho một hàm có tên là ThemSach (), hàm này thêm sách vào danh sách liên kết.
* Sau khi tất cả các cuốn sách cho "DAUSACH" hiện tại đã được thêm, hàm đọc dòng tiếp theo từ tệp (nó sẽ là dòng "DAUSACH" tiếp theo) để chuẩn bị cho vòng lặp while tiếp theo.
* Sau khi vòng lặp while kết thúc, đối tượng StreamReader được đóng.
* Nếu có lỗi xảy ra khi đọc từ tệp, một thông báo lỗi sẽ được in ra màn hình console.

public void ThemSachVaolinklist()

{

try

{

using (StreamReader reader = new StreamReader("danh-muc-sach-ton-kho.txt"))

{

string line = reader.ReadLine();

while(line != null)

{

string[] fields = line.Split('|');

if (fields[0] == "DAUSACH")

{

int soluong = int.Parse(fields[2]);

for (int i = 0; i < soluong; i++)

{

line = reader.ReadLine();

fields = line.Split('|');

int maSach = int.Parse(fields[0]);

int trangThai = int.Parse(fields[1]);

int viTri = int.Parse(fields[2]);

ThemSach(maSach, trangThai, viTri);

}

line = reader.ReadLine();

}

}

reader.Close();

}

}

catch (Exception ex)

{

Console.WriteLine("Loi khi doc file: " + ex.Message);

}

}

**- Phương thức ReadDataFromFileDanhMucSach** : Sử dụng để thêm các sách vào danh sách của đầu sách tương ứng

Giải thuật :

* Mở tệp "danh-muc-sach-ton-kho.txt" bằng đối tượng StreamReader.
* Đọc dòng đầu tiên của tệp và gán giá trị cho biến line.
* Sử dụng vòng lặp while để đọc từng dòng của tệp cho đến khi biến line bằng null.
* Phân tách giá trị trên dòng hiện tại thành một mảng chuỗi fields bằng cách sử dụng phương thức Split() với ký tự ” | ”
* Kiểm tra nếu giá trị đầu tiên của mảng fields bằng "DAUSACH" đúng thì thực hiện lệnh dưới , không thì lặp tiếp đến khi đúng
* Lấy giá trị mã số ISBN của đầu sách từ fields[1] trên dòng hiện tại.
* Lấy giá trị số lượng sách của đầu sách từ fields[2] thứ ba trên dòng hiện tại và chuyển đổi sang kiểu số nguyên.
* Sử dụng vòng lặp foreach để lặp qua danh sách đầu sách.
* Kiểm tra nếu mã số ISBN của đầu sách hiện tại trùng với mã số ISBN thì dùng vòng lặp for để đọc thông tin cho từng cuốn sách của đầu sách và thêm chúng vào danh sách sách của đầu sách đó bằng cách gọi phương thức ThemSachKhiNapFile()
* Đọc dòng tiếp theo để đọc đến đầu sách tiếp theo
* Thoát khỏi vòng lặp foreach bằng cách sử dụng câu lệnh break
* Nếu giá trị đầu tiên của mảng fields không bằng "DAUSACH" thì bỏ qua
* Đọc dòng tiếp theo để đọc đến dòng mới
* Đóng tệp bằng cách gọi phương thức Close() của đối tượng StreamReader
* Nếu có lỗi xảy ra trong quá trình đọc từ tệp, in ra một thông báo lỗi

public static void ReadDataFromFileDanhMucSach()

{

try

{

using (StreamReader reader = new StreamReader("danh-muc-sach-ton-

kho.txt"))

{

string line = reader.ReadLine();

while (line != null)

{

string[] fields = line.Split('|');

if (fields[0] == "DAUSACH")

{

string isbn = fields[1];

int soluong = int.Parse(fields[2]);

// đọc danh sách sách của đầu sách này

foreach (DauSach dausach in BookFunc.ListDauSach)

{

if (dausach.ISBN == isbn)

{

for (int i = 0; i < soluong; i++)

{

line = reader.ReadLine();

fields = line.Split('|');

int maSach = int.Parse(fields[0]);

int trangThai = int.Parse(fields[1]);

int viTri = int.Parse(fields[2]);

Sach sach = new Sach(maSach, trangThai,

viTri, null);

dausach.SachTrongThuVien.ThemSachKhiNapFile(sach);

}

line = reader.ReadLine();

break;

}

}

}

}

reader.Close();

}

}

catch (Exception ex)

{

Console.WriteLine("Loi khi doc file: " + ex.Message);

}

}

- **Phương thức UpdateMuonInFile và UpdateTraInFile** : Sử dụng để cập nhật danh sách sách lên file txt khi độc giả mượn và trả sách

Giải thuật:

* Tạo một biến đường dẫn đến file danh mục sách tồn kho là filePath.
* Đọc toàn bộ nội dung file và lưu vào một List<string> gọi là lines.
* Duyệt qua từng phần tử trong lines:
* Nếu phần tử đó không phải là thông tin đầu sách và có mã số sách trùng với mã số sách truyền vào (maso) thì:
* Lấy giá trị vị trí của sách hiện tại (được lưu trong parts[2]).
* Xóa phần tử đó ra khỏi List lines.
* Chèn một phần tử mới tương ứng với thông tin sách đã trả vào List lines. Phần tử mới này có định dạng là: maso + "|0|" + vitri nếu mượn sách maso + "|1|" + vitri nếu trả sách
* Kết thúc vòng lặp.
* Ghi lại toàn bộ nội dung của List lines vào file danh mục sách tồn kho.

public void UpdateMuonInFile(string maso)

{

string filePath = "danh-muc-sach-ton-kho.txt";

List<string> lines = File.ReadAllLines(filePath).ToList();

for (int i = 0; i < lines.Count; i++)

{

string[] parts = lines[i].Split('|');

if (parts[0] != "DAUSACH" && parts[0] == maso)

{

string vitri = parts[2];// Ghi lại vi tri

lines.RemoveAt(i);

lines.Insert(i, maso + "|1|" + vitri);

break;

}

}

File.WriteAllLines(filePath, lines);

}

public void UpdateTraInFile(string maso)

{

string filePath = "danh-muc-sach-ton-kho.txt";

List<string> lines = File.ReadAllLines(filePath).ToList();

for (int i = 0; i < lines.Count; i++)

{

string[] parts = lines[i].Split('|');

if (parts[0] != "DAUSACH" && parts[0] == maso)

{

string vitri = parts[2];// Ghi lại vi tri

lines.RemoveAt(i);

lines.Insert(i, maso + "|0|" + vitri);

break;

}

}

File.WriteAllLines(filePath, lines);

}

### 1.2.2 Tổ chức dữ liệu cho DanhMucSach

Cấu trúc lưu trữ thông tin sách được tổ chức như sau :

DAUSACH|DPTK|3

1001|0|1

1002|0|1

1003|0|1

DAUSACH|LVTH|3

2001|0|1

2002|0|1

Trong đó dòng DAUSACH là dấu cho biết các sách dòng tiếp theo thuộc đầu sách này với số lượng là số cuối cùng của hàng DAUSACH

## Tổ chức cấu trúc dữ liệu cho DocGia

### 1.3.1 Tổ chức cấu trúc cho DocGia

\* Tạo đối tượng DocGia với các dữ liệu sau

public class DocGia

{

public int MaDG { get; set; }

public string Ho { get; set; }

public string Ten { get; set; }

public string Phai { get; set; }

public int TrangThaiThe { get; set; }

public DS\_MuonTra SachDangMuon = new DS\_MuonTra();

public DocGia Left;

public DocGia Right;

public override string ToString()

{

return MaDG + ";" + Ho + ";" + Ten + ";" + Phai + ";" + TrangThaiThe;

}

}

* Ho: kiểu chuỗi, đại diện cho họ của độc giả.
* Ten: kiểu chuỗi, đại diện cho tên của độc giả.
* Phai: kiểu chuỗi, đại diện cho giới tính của độc giả.
* TrangThaiThe: kiểu số nguyên, đại diện cho trạng thái thẻ của độc giả.
* Danh sách các sách đang được mượn (SachDangMuon) trỏ về danh sách mượn trả.
* Tổ chức Node Left , Right trong sơ đồ cây nhị phân

\* Tạo đối tượng ReaderFunc để quản lí các đối tượng DocGia: DS\_MuonTra sử dụng cấu trúc danh sách liên kết đơn:

public class ReaderFunc

{

public static List<DocGia> ListDocGia = new List<DocGia>();

public static DocGia root;

public static string message;

public ReaderFunc()

{

root = null;

}

\* Các phương thức xử lí trong ReaderFunc

**Phương thức AddReader** - Thêm một đối tượng DocGia vào cây tìm kiếm nhị phân:

Giải thuật:

* Kiểm tra nút gốc, nếu cây chưa có độc giả nào thì gán nút gốc bằng độc giả đang được thêm và kết thúc hàm.
* Nếu cây đã có độc giả, khởi tạo biến current bằng nút gốc.
* Lặp lại cho đến khi tìm được vị trí thích hợp để thêm độc giả vào cây:

Nếu mã độc giả của độc giả mới nhỏ hơn mã độc giả của nút hiện tại, kiểm tra nút con bên trái của nút hiện tại:

* Nếu nút con bên trái không tồn tại, gán nút con bên trái của nút hiện tại bằng độc giả mới và kết thúc hàm.
* Nếu nút con bên trái đã tồn tại, gán nút hiện tại bằng nút con bên trái và lặp lại từ bước 3.
* Nếu mã độc giả của độc giả mới lớn hơn hoặc bằng mã độc giả của nút hiện tại, kiểm tra nút con bên phải của nút hiện tại:
* Nếu nút con bên phải không tồn tại, gán nút con bên phải của nút hiện tại bằng độc giả mới và kết thúc hàm.
* Nếu nút con bên phải đã tồn tại, gán nút hiện tại bằng nút con bên phải và lặp lại từ bước 3.

public static void AddReader(DocGia reader)

{

if (root == null)

{

root = reader;

return;

}

DocGia current = root;

while (current != null)

{

if (reader.MaDG < current.MaDG)

{

if (current.Left == null)

{

current.Left = reader;

return;

}

current = current.Left;

}

else

{

if (current.Right == null)

{

current.Right = reader;

return;

}

current = current.Right;

}

}

}

**- Phương thức GetReaderByMaDG**: Tìm kiếm độc giả trong cây tìm kiếm nhị phân dựa trên mã độc giả (MaDG):

Giải thuật:

* + Khởi tạo con trỏ current trỏ đến nút gốc root của cây.
  + Lặp lại các bước sau cho đến khi current trỏ đến nút cuối cùng (lá) hoặc tìm thấy độc giả có mã madg:
* So sánh mã độc giả cần tìm madg với mã độc giả của nút current.
* Nếu madg nhỏ hơn mã độc giả của current, thì ta chuyển current sang nút con trái của nút hiện tại.
* Nếu madg lớn hơn mã độc giả của current, thì ta chuyển current sang nút con phải của nút hiện tại.
* Nếu madg bằng với mã độc giả của current, ta trả về đối tượng DocGia ứng với nút hiện tại.
  + Nếu không tìm thấy độc giả có mã madg trong cây, ta trả về null.

public static DocGia GetReaderByMaDG(int madg)

{

DocGia current = root;

while (current != null)

{

if (madg < current.MaDG)

{

current = current.Left;

}

else if (madg > current.MaDG)

{

current = current.Right;

}

else

{

return current;

}

}

return null;

}

**- Phương Thức GetMaDG** - Tạo mã độc giả (MaDG) ngẫu nhiên không trùng lặp trong danh sách các độc giả hiện có :

Giải thuật:

- Tạo một đối tượng của lớp Random.

* Khởi tạo biến madg với giá trị ngẫu nhiên trong khoảng từ 10000 đến 99999.
* Tạo một danh sách existingMaDG chứa các mã độc giả đã tồn tại trong ListDocGia.
* Kiểm tra nếu danh sách existingMaDG chứa mã độc giả madg thì quay lại bước 2 để sinh lại giá trị ngẫu nhiên cho madg.
* Nếu madg không tồn tại trong danh sách existingMaDG thì trả về giá trị madg.

public static int GetMaDG()

{

Random rand = new Random();

int madg;

List<int> existingMaDG = ListDocGia.Select(dg => dg.MaDG).ToList(); // Danh sách mã độc giả đã tồn tại

do

{

madg = rand.Next(10000, 99999);

} while (existingMaDG.Contains(madg)); // Nếu mã độc giả đã tồn tại thì sinh lại

return madg;

}

**- Phương Thức EditReader(DocGia reader)** - cập nhật thông tin của một độc giả có sẵn trong cây tìm kiếm nhị phân:

Giải thuật:

* Khởi tạo nút current bằng nút root.
* Lặp lại các bước sau cho đến khi current bằng null:
* Kiểm tra xem reader có trùng với current không. Nếu có, cập nhật thông tin độc giả hiện tại với thông tin của reader và kết thúc hàm.
* Nếu reader.MaDG nhỏ hơn giá trị MaDG của current, di chuyển đến nút con trái của current.
* Nếu reader.MaDG lớn hơn giá trị MaDG của current, di chuyển đến nút con phải của current.
* Kết thúc hàm.

public static void EditReader(DocGia reader)

{

DocGia current = root;

while (current != null)

{

if (reader.MaDG == current.MaDG)

{

current.Ho = reader.Ho;

current.Ten = reader.Ten;

current.Phai = reader.Phai;

current.TrangThaiThe = reader.TrangThaiThe;

return;

}

else if (reader.MaDG < current.MaDG)

{

current = current.Left;

}

else

{

current = current.Right;

}

}

}

**- Phương thức XoaDocGia**: Xóa Độc giả theo mã độc giả cần xóa

Giải thuật:

* Khởi tạo nút cha (parent) là null và nút hiện tại (current) là nút gốc (root).
* Lặp qua cây tìm kiếm nhị phân (BST) bằng cách kiểm tra mã độc giả của nút hiện tại với mã độc giả được cung cấp (madg).
* Nếu mã độc giả được cung cấp (madg) nhỏ hơn mã độc giả của nút hiện tại (current), di chuyển sang nút bên trái của nút hiện tại và lưu nút hiện tại vào nút cha (parent).
* Nếu mã độc giả được cung cấp (madg) lớn hơn mã độc giả của nút hiện tại (current), di chuyển sang nút bên phải của nút hiện tại và lưu nút hiện tại vào nút cha (parent).
* Nếu mã độc giả của nút hiện tại (current) bằng mã độc giả được cung cấp (madg), kiểm tra xem độc giả đang mượn sách hay không. Nếu độc giả đang mượn sách, thông báo lỗi và thoát hàm. Nếu độc giả không mượn sách, xóa nút hiện tại (current) bằng cách gọi hàm DeleteNode(node) và lưu kết quả trở lại nút cha (parent).
* Nếu nút cha (parent) là null, tức là nút hiện tại (current) là nút gốc, cập nhật nút gốc (root) bằng nút trả về từ hàm DeleteNode(node).
* Nếu mã độc giả được cung cấp (madg) nhỏ hơn mã độc giả của nút cha (parent), cập nhật nút bên trái của nút cha (parent) bằng nút trả về từ hàm DeleteNode(node).
* Nếu mã độc giả được cung cấp (madg) lớn hơn mã độc giả của nút cha (parent), cập nhật nút bên phải của nút cha (parent) bằng nút trả về từ hàm DeleteNode(node).
* Trả về thông báo thành công hoặc lỗi

public static void XoaDocGia(int madg)

{

DocGia parent = null;

DocGia current = root;

while (current != null)

{

if (madg < current.MaDG)

{

parent = current;

current = current.Left;

}

else if (madg > current.MaDG)

{

parent = current;

current = current.Right;

}

else

{

if (current.SachDangMuon.Count() > 0)

{

message = "Khong the xoa doc gia vi doc gia dang muon sach";

return;

}

if (parent == null)

{

root = DeleteNode(current);

}

else if (madg < parent.MaDG)

{

parent.Left = DeleteNode(current);

}

else

{

parent.Right = DeleteNode(current);

}

message="Da xoa doc gia co ma la: "+ madg ;

return;

}

}

message="Khong tim thay doc gia co ma la :"+ madg;

}

**- Phương thức DeleteNode:** được sử dụng để xóa một nút node trong cây nhị phân tìm kiếm.

Giải thuật: Nếu nút node không có nút con nào, thì hàm trả về null. Nếu nút node chỉ có một nút con (trái hoặc phải), thì hàm trả về nút con đó. Nếu nút node có hai nút con, thì hàm tìm kiếm nút kế cận (successor) của nút node (nút kế cận là nút nhỏ nhất trong cây con bên phải của node). Sau đó, hàm thay thế nút node bằng nút successor và gán nút con trái của node cho nút con trái của successor.

private static DocGia DeleteNode(DocGia node)

{

if (node.Left == null && node.Right == null)

{

return null;

}

else if (node.Left == null)

{

return node.Right;

}

else if (node.Right == null)

{

return node.Left;

}

else

{

DocGia successor = GetSuccessor(node);

successor.Left = node.Left;

return successor;

}

}

* **Phương thức GetSuccessor:**

Giải thuật:

* Khởi tạo nút parent và nút current đều bằng nút trái của nút đầu vào node.
* Nếu current không phải là nút lá trái, thực hiện bước 3-4, ngược lại trả về current.
* Gán parent bằng current và current bằng nút lá trái của current.
* Lặp lại bước 2-3 cho đến khi current là nút lá trái.
* Gán nút lá trái của parent bằng nút phải của current.
* Gán nút phải của current bằng nút phải của nút đầu vào node.
* Trả về current là kết quả.

private static DocGia GetSuccessor(DocGia node)

{

DocGia parent = node;

DocGia current = node.Right;

while (current.Left != null)

{

parent = current;

current = current.Left;

}

parent.Left = current.Right;

current.Right = node.Right;

return current;

}

- **Phương thức InOrder** duyệt cây theo Inorder Traversal:

private static void InOrder(DocGia node)

{

if (node != null)

{

InOrder(node.Left);

HienThiThongTinDocGia(node);

InOrder(node.Right);

}

}

**- Phương thức InsertionSortByMa**: Sắp xếp độc giả theo mã độc giả sử dụng phương pháp sắp xếp Insertion Sort sắp xếp danh sách độc giả

Giải thuật:

* Lưu trữ phần tử hiện tại vào biến current và lưu trữ chỉ số j = i - 1.
* Tiến hành đẩy các phần tử lớn hơn phần tử hiện tại về phía sau trong danh sách bằng cách so sánh mã độc giả (MaDG) của chúng với MaDG của phần tử hiện tại. Trong quá trình này, các phần tử này sẽ được dịch chuyển sang bên phải cho đến khi tìm được vị trí thích hợp để chèn phần tử hiện tại vào.
* Sau khi xác định được vị trí thích hợp, chèn phần tử hiện tại vào danh sách.
* Lặp lại các bước trên cho tới khi danh sách được sắp xếp.

private static void InsertionSortByMa(List<DocGia> danhSachDocGia)

{

int n = danhSachDocGia.Count;

for (int i = 1; i < n; i++)

{

DocGia current = danhSachDocGia[i];

int j = i - 1;

while (j >= 0 && danhSachDocGia[j].MaDG.CompareTo(current.MaDG) > 0)

{

danhSachDocGia[j + 1] = danhSachDocGia[j];

j--;

}

danhSachDocGia[j + 1] = current;

}

}

* **Phương thức InsertionSortByTen:** Sắp xếp độc giả theo tên sử dụng thuật toán Insertion sort

Giải thuật

* Sắp xếp chèn bắt đầu bằng cách lặp qua từng phần tử trong danh sách độc giả, bắt đầu từ vị trí thứ 2 đến cuối danh sách (vòng lặp bên ngoài).
* Trong vòng lặp này, phần tử hiện tại được lưu trong biến current và vòng lặp bên trong được sử dụng để di chuyển các phần tử lớn hơn giá trị hiện tại sang phía sau của danh sách
* Nếu phần tử hiện tại có tên lớn hơn phần tử đang xét trong vòng lặp bên trong, thì phần tử đó được di chuyển sang vị trí sau để tạo chỗ cho phần tử hiện tại
* Nếu tên bằng nhau, thì phần tử đang xét trong vòng lặp bên trong sẽ được so sánh theo họ, nếu họ lớn hơn thì phần tử đó cũng sẽ được di chuyển sang vị trí sau để tạo chỗ cho phần tử hiện tại .
* Cuối cùng, phần tử hiện tại được chèn vào vị trí đúng trong danh sách.

private static void InsertionSortByTen(List<DocGia> danhSachDocGia)

{

int n = danhSachDocGia.Count;

for (int i = 1; i < n; i++)

{

DocGia current = danhSachDocGia[i];

int j = i - 1;

while (j >= 0 && string.Compare(danhSachDocGia[j].Ten, current.Ten) > 0)

{

danhSachDocGia[j + 1] = danhSachDocGia[j];

j--;

}

if (j >= 0 && danhSachDocGia[j].Ten == current.Ten && string.Compare(danhSachDocGia[j].Ho, current.Ho) > 0)

{

danhSachDocGia[j + 1] = danhSachDocGia[j];

j--;

}

danhSachDocGia[j + 1] = current;

}

}

**1.3.2** **Tổ chức cơ sở dữ liệu cho độc giả**

Cấu trúc lưu trữ thông tin độc giả được tổ chức như sau : ‘

Mã độc giả ; Họ ; Tên ; Phái ; Trạng thái thẻ :

A picture containing text, font, screenshot, white

Description automatically generated

Để sử dụng được dữ liệu trên ở đây ta tổ chức 2 phương pháp Đọc file và lưu file txt:

- Phương thức LuuDanhSachDocGiaVaoFile lưu các thông tin từ node của cây nhị phân

Giải thuật:

* Mở một file data\_docgia.txt để ghi dữ liệu danh sách độc giả vào.
* Sử dụng hàm InDanhSachDocGiaTheoMa() để lấy danh sách độc giả đã được sắp xếp theo mã.
* Duyệt qua từng phần tử DocGia trong danh sách lấy được ở bước 2 và ghi thông tin của nó vào file, mỗi phần tử trên một dòng. Thông tin được cấu trúc dạng MaDG;Ho;Ten;Phai;TrangThaiThe.
* Đóng file vừa ghi.

public static void LuuDanhSachDocGiaVaoFile()

{

using (StreamWriter writer = new StreamWriter("..\\data\_docgia.txt"))

{

List<DocGia> danhSachDocGia = InDanhSachDocGiaTheoMa();

foreach (DocGia docGia in danhSachDocGia)

{

string line = docGia.MaDG + ";" + docGia.Ho + ";" + docGia.Ten + ";" + docGia.Phai + ";" + docGia.TrangThaiThe;

writer.WriteLine(line);

}

}

}

- Phương thức DocFile():

Giải thuật:

* Đặt đường dẫn tệp thành "..\data\_docgia.txt".
* Tạo một đối tượng mới của lớp DocGia.
* Thử mở tệp bằng một đối tượng StreamReader và gán nó vào biến sr.
* Đọc dòng đầu tiên của tệp và gán nó vào biến line.
* Trong khi line không null, làm những việc sau:

o Tách line thành một mảng các chuỗi bằng ';' làm dấu phân cách và gán nó vào arr.

o Nếu arr có 5 phần tử, tạo một đối tượng mới của lớp DocGia, thiết lập các thuộc tính của nó dựa trên các phần tử trong arr, thêm nó vào cấu trúc dữ liệu của hệ thống bằng cách sử dụng phương thức AddReader của lớp ReaderFunc và thêm nó vào một danh sách gọi là ListDocGia.

o Đọc dòng tiếp theo của tệp và gán nó vào biến line.

* Đóng đối tượng StreamReader.
* Nếu có ngoại lệ được ném ra, ném nó tiếp tục.

public static void DocFile()

{

string path = "..\\data\_docgia.txt";

DocGia dsdg = new DocGia();

try

{

StreamReader sr = new StreamReader(path);

string line = sr.ReadLine();

while (line != null)

{

string[] arr = line.Split(';');

if (arr.Length == 5)

{

DocGia dg = new DocGia();

dg.MaDG = int.Parse(arr[0].Trim());

dg.Ho = arr[1].Trim();

dg.Ten = arr[2].Trim();

dg.Phai = arr[3].Trim();

dg.TrangThaiThe = int.Parse(arr[4].Trim());

ReaderFunc.AddReader(dg);

ListDocGia.Add(dg);

}

line = sr.ReadLine();

}

sr.Close();

}

catch (Exception ex)

{

throw ex;

}

}

## 1.4 Tổ chức cấu trúc dữ liệu cho MUONTRA

### 1.4.1 Tổ chức cấu trúc cho MUONTRA

\* Tạo đối tượng MuonTra bao gồm các kiểu dữ liệu sau:

public class MuonTra

{

public int MaSach { get; set; }

public string NgayMuon { get; set; }

public string NgayTra { get; set; }

public int TrangThai { get; set; }

public int soluotmuon { get; set; }

public int MaDG { get; set; }

public MuonTra next;

public MuonTra(int ms, string m, string t, int tt,int slm, int mdg, MuonTra n)

{

MaSach = ms;

NgayMuon = m;

NgayTra = t;

TrangThai = tt;

next = n;

soluotmuon = slm;

MaDG = mdg;

}

}

- MaSach : Mã sách đang mượn trả kiểu int

- NgayMuon: Ngày mượn kiểu string

- NgayTra: Ngày trả kiểu string

- TrangThai: Trạng thái sách mượn kiểu int, trả với 0 là sách đang mượn, 1 là sách đã trả, 2 là làm mất sách

- Soluotmuon: Số lượt mượn của sách

- MaDG: Mã độc giả đang mượn sách kiểu int, nếu độc giả mượn sách này thì MaDG tương ứng với mã của độc giả, còn sách không mươn thì MaDG bằng 0

- next : Con trỏ trỏ đến sách mượn trả tiếp theo trong danh sách mượn trả

\* Tạo đối tượng DS\_MuonTra để quản lí các đối tượng MuonTra: DS\_MuonTra sử dụng cấu trúc danh sách liên kết đơn

public class DS\_MuonTra

{

public MuonTra first; //node đầu tiên trong danh sách

public MuonTra last; //node cuối cùng trong danh sách

public int size; //số đối tượng MuonTra trong danh sách

public DS\_MuonTra()

{

first = null;

last = null;

size = 0;

}

}

\* Các phương thức xử lí trong DS\_MuonTra:

- Phương thức IsEmpy(): Kiểm tra số đối tượng MuonTra trong danh sách có bằng 0 hay không

public bool IsEmpty()

{

return size == 0;

}

Phương thức trả về true nếu size = 0 và trả về false nếu ngược lại

- Phương thức InsertLast\_MuonTra: Thêm các đối tượng MuonTra vào danh sách:

Giải thuật:

+ Tạo một đối tượng MuonTra là newNode mới với các thông tin được truyền vào (ms, m, t, tt, slm, mdg).

+ Kiểm tra trong danh sách, nếu rỗng tương đương với fisrt bằng null thì gán newNode cho first và last, nếu không rỗng gán con trỏ next của last cho newNode, sau đó gán last bằng newNode

+ Tăng size lên 1 để đồng bộ kích thước của danh sách sau khi kiểm tra điều kiện.

public void Insertlast\_MuonTra(int ms, string m, string t, int tt, int slm, int mdg)

{

MuonTra newNode = new MuonTra(ms, m, t, tt, slm,mdg, null);

if (first == null)

{

first = newNode;

last = newNode;

}

else

{

last.next = newNode;

last = newNode;

}

size++;

}

- Phương thức MuonSach

Giải thuật:

+ Tạo một biến dem với giá trị ban đầu = 0, biến đếm dùng để đánh dấu cho việc đã có đối tượng MuonTra có MaSach tương ứng với MaSach truyền vào hay chưa, tạo một đối tượng MuonTra là muon và gắn giá trị bằng first

+ Duyệt từng nút của danh sách từ muon bằng first cho đến khi muon bằng null, trong mỗi vòng lặp kiểm tra xem Masach của nút muon có bằng Masach được truyền vào hay không, nếu đúng thực hiện gán các giá trị: MaDG của muon bằng MaDG được truyền vào, TrangThai của muon bằng 0, soluotmuon của mượn tăng lên 1, NgayMuon của muon bằng ngày hiện tại, NgayTra của muon bằng rỗng, gán giá trị biến đếm bằng 1.

+ Sau khi duyệt, so sánh điều kiện biến dem, nếu dem bằng 0 tương ứng việc trong danh sách chưa có đối tượng MuonTra tương ứng với MaSach nhập vào, dùng hàm Insertlast\_MuonTra để thêm một đối tượng MuonTra mới vào cuối danh sách các thông tin được truyền vào hàm tương tự như trong vòng lặp trừ soluotmuon được truyền vào bằng 1.

public void MuonSach(int MaDG,int MaSach)

{

int dem = 0;

MuonTra muon = first;

while (muon != null)

{

if (muon.MaSach == MaSach)

{

muon.MaDG = MaDG;

muon.TrangThai = 0;

muon.soluotmuon++;

muon.NgayMuon = DateTime.Now.Date.ToString();

muon.NgayTra = "";

dem = 1;

}

muon = muon.next;

}

if(dem == 0)

{

Insertlast\_MuonTra(MaSach, DateTime.Now.Date.ToString(),"",0,1,MaDG);

}

}

**- Phương thức TraSach**

Giải thuật:

+ Tạo một đối tượng MuonTra và gán giá trị bằng first

+ Duyệt từng nút của danh sách từ muon bằng first cho đến khi muon bằng null, trong mỗi vòng lặp kiểm tra xem Masach của nút muon có bằng Masach được truyền vào hay không, nếu đúng thực hiện gán các giá trị: MaDG của muon bằng 0, NgayTra của muon bằng thời gian hiện tại.

public void TraSach(int MaDG,int MaSach)

{

MuonTra muon = first;

while(muon != null)

{

if(muon.MaSach == MaSach)

{

muon.MaDG = 0;

muon.TrangThai = 1;

muon.NgayTra = DateTime.Now.Date.ToString();

}

muon = muon.next;

}

}

**- Phương thức Xoahetdsmuon:** Xóa toàn bộ đối tượng MuonTra trong danh sách

Giải thuật:

+ Gán các giá trị first và last bằng null, đặt size bằng 0

public void Xoahetdsmuon()

{

first = null;

last = null;

size = 0;

}

**- Phương thức GetMuonTrabyMaSach:** Sử dụng để trả về một đối tượng MuonTra trong danh sách

Giải thuật

+ Tạo một đối tượng MuonTra là sach và gán giá trị bằng first

+ Duyệt từng nút của danh sách từ muon bằng first cho đến khi muon bằng null, trong mỗi vòng lặp kiểm tra xem Masach của nút muon có bằng Masach được truyền vào hay không, nếu đúng trả về sach.

+ Sau khi duyệt nếu không tìm được thì trả về null

public MuonTra GetMuonTrabyMaSach(int Masach)

{

MuonTra sach = first;

while(sach != null)

{

if(sach.MaSach == Masach)

{

return sach;

}

sach = sach.next;

}

return null;

}

### 1.4.2 Tổ chức dữ liệu cho MuonTra

Cấu trúc dữ liệu trong file file txt: MaSach,NgayMuon,NgayTra,trangthai,solanmuon,MaDG

Text

Description automatically generated

**- Phương thức Themsachvaolinklist:** Thêm các dữ liệu từ file txt vào DS\_MuonTra

Giải thuật:

+ Khởi tạo và đọc dữ liệu tới file txt

+ Duyệt lần lượt từng hàng, mỗi hàng có các dữ liệu tương ứng được ngăn cách bởi dấu phẩy, đổi các dữ liệu và lưu tương ứng vào từng biến. Dùng phương thức Insertlast\_MuonTra và truyền vào phương thức các biến tương ứng để thêm các đối tượng MuonSach của danh sách

+ Sau khi duyệt hết thì kết thúc đọc file

Code phương thức:

public void Themsachvaolinklist()

{

string filePath = "..\\data\_muontra.txt";

StreamReader reader = new StreamReader(filePath);

while (!reader.EndOfStream)

{

string[] line = reader.ReadLine().Split(',');

int ms = int.Parse(line[0]);

string m = line[1];

string t = line[2];

int tt = int.Parse(line[3]);

int slm = int.Parse(line[4]);

int mdg = int.Parse(line[5]);

Insertlast\_MuonTra(ms, m, t, tt, slm, mdg);

}

reader.Close();

}

**- Phương thức ReadMuonTraDataFromFile:** Sử dụng để thêm các đôi tượng MuonTra tương ứng mà Độc giả đang mượn vào DanhSachMuon của Độc giả đó

Giải thuật:

+ Khởi tạo và đọc dữ liệu tới file txt

+ Duyệt lần lượt từng hàng, mỗi hàng có các dữ liệu tương ứng được ngăn cách bởi dấu phẩy, đổi các dữ liệu và lưu tương ứng vào từng biến. Duyệt danh sách độc giả, nếu mã độc giả trùng với mã độc giả đọc được từ file txt dùng phương thức Insertlast\_MuonTra và truyền vào phương thức các biến tương ứng để thêm các đối tượng MuonSach của SachMuonTra của độc giả.

+ Sau khi duyệt hết thì kết thúc đọc file

**- Phương thức UpdateMuonTraDataToFile:** Dùng để cập nhật lên file txt từ danh sách mượn trả

Giải thuật:

+ Xóa hết dữ liệu trên file txt.

+ Khởi tạo để ghi dữ liệu tới file txt

+ Duyệt lần lượt từng phần tử MuonTra trong danh sách và ghi lên file txt

+ Sau khi duyệt xong thì kết thúc ghi.

Code phương thức:

public void UpdateMuonTraDataToFile()

{

string filePath = "..\\data\_muontra.txt";

File.WriteAllText(filePath, string.Empty);

{

using (StreamWriter writer = new StreamWriter(filePath))

{

MuonTra current = first;

while (current != null)

{

writer.WriteLine($"{current.MaSach},{current.NgayMuon},{current.NgayTra},{current.TrangThai},{current.soluotmuon},{current.MaDG}");

current = current.next;

}

}

}

}

# 2. Giao diện

## Giao diện quản lí đầu sách

Giao diện chính

A screenshot of a computer

Description automatically generated

* Sử dụng dataGrid để hiển thị danh sách DauSach.
* Sử dụng các textBox để nhập thông tin DauSach.
* Sử dụng các button để thực hiện chức năng như : hiệu chỉnh , tìm kiếm, thêm đầu sách.

Hiển thị thông tin độc giả lên dataGrid , giải thuật như sau :

* Đọc dữ liệu từ dữ liệu DauSach cho trước
* Đưa dữ liệu độc giả vào dataGrid theo cấu trúc linkedlist

Code :

private void Form1\_Load(object sender, EventArgs e)

{

BookFunc.ReadDataFromFile();

DanhMucSach.ReadDataFromFileDanhMucSach();

BookFunc.SortByISBN();

dgvDauS.DataSource = BookFunc.ListDauSach;

}

Kết quả :

- Khởi chạy chương trình và Nhấn vào button “Quản lý đầu sách:

A screenshot of a computer

Description automatically generated with medium confidence

- Giao diện sẽ hiển thị ra Danh sách các đầu sách được sắp xếp theo thứ tự tăng dần tên sách từ A -> Z :

A screenshot of a computer

Description automatically generated with medium confidence

Chuyển các thông tin đầu sách sang bảng thông tin đầu sách , Giải thuật như sau :

Phương thức **dgvDauS\_SelectionChanged** được gọi mỗi khi người dùng chọn một hàng trong bảng DataGridView dgvDauS. Hàm này thực hiện các bước sau:

* Kiểm tra xem có hàng được chọn hay không, bằng cách kiểm tra số lượng hàng được chọn (SelectedRows.Count), nếu lớn hơn 0 thì có ít nhất một hàng được chọn.
* Tạo một đối tượng DauSach mới, và lấy giá trị các ô trong hàng được chọn gán vào các thuộc tính của đối tượng này, đồng thời hiển thị giá trị này lên các ô văn bản tương ứng trên giao diện người dùng.
* Nếu không có hàng được chọn, không có thay đổi nào được thực hiện trên giao diện người dùng.

private void dgvDauS\_SelectionChanged(object sender, EventArgs e)

{

if (dgvDauS.SelectedRows.Count > 0)

{

DauSach dausach = new DauSach();

// Lấy dữ liệu từ hàng được chọn

DataGridViewRow row = dgvDauS.SelectedRows[0];

dausach.ISBN = row.Cells["ISBN"].Value.ToString();

dausach.TenSach = row.Cells["TenSach"].Value.ToString();

dausach.SoTrang = int.Parse(row.Cells["SoTrang"].Value.ToString());

dausach.TacGia = row.Cells["TacGia"].Value.ToString();

dausach.NamXB = int.Parse(row.Cells["NamXB"].Value.ToString());

dausach.TheLoai = row.Cells["TheLoai"].Value.ToString();

tbISBN.Text = dausach.ISBN;

tbTenSach.Text = dausach.TenSach;

tbSoTrang.Text = dausach.SoTrang.ToString();

tbTacGia.Text = dausach.TacGia;

tbNamXB.Text = dausach.NamXB.ToString();

tbTheLoai.Text = dausach.TheLoai;

}

}

- Phương thức **btnSuaDS\_Click** được gọi khi người dùng click vào nút "Sửa Đầu Sách" trong giao diện của chương trình. Các bước thực hiện của hàm như sau:

* Kiểm tra xem người dùng đã chọn một hàng trong DataGridView chưa.
* Nếu đã chọn, lấy dữ liệu từ hàng được chọn và cập nhật lại các giá trị mới được nhập vào TextBox.
* Sắp xếp lại danh sách theo ISBN.
* Cập nhật lại DataGridView.
* Lưu lại dữ liệu vào file "danh-sach-sach.txt".
* Hiển thị thông báo cho người dùng biết rằng dữ liệu đã được cập nhật và lưu thành công.

Về cách thực hiện, hàm truy cập vào các dữ liệu trong các ô DataGridView thông qua đối tượng DataGridViewRow. Sau đó, hàm sử dụng StreamWriter để ghi các dữ liệu mới vào file "danh-sach-sach.txt". Cuối cùng, hàm hiển thị thông báo thông qua MessageBox.

private void btnSuaDS\_Click(object sender, EventArgs e)

{

if (dgvDauS.SelectedRows.Count > 0)

{

// Lấy dữ liệu từ hàng được chọn

DataGridViewRow row = dgvDauS.SelectedRows[0];

row.Cells["ISBN"].Value = tbISBN.Text;

row.Cells["TenSach"].Value = tbTenSach.Text;

row.Cells["SoTrang"].Value = int.Parse(tbSoTrang.Text);

row.Cells["TacGia"].Value = tbTacGia.Text;

row.Cells["NamXB"].Value = int.Parse(tbNamXB.Text);

row.Cells["TheLoai"].Value = tbTheLoai.Text;

BookFunc.SortByISBN();

// Cập nhật lại DataGridView

dgvDauS.Refresh();

// Lưu lại dữ liệu vào file txt

StreamWriter sw = new StreamWriter("danh-sach-sach.txt");

foreach (DataGridViewRow r in dgvDauS.Rows)

{

string line = string.Format("{0}|{1}|{2}|{3}|{4}|{5}",

r.Cells["ISBN"].Value, r.Cells["TenSach"].Value, r.Cells["SoTrang"].Value,

r.Cells["TacGia"].Value, r.Cells["NamXB"].Value, r.Cells["TheLoai"].Value);

sw.WriteLine(line);

}

sw.Close();

MessageBox.Show("Đã cập nhật và lưu dữ liệu thành công!");

}

}

Kết quả :

- Chọn 1 đầu sách cần sửa thông tin : ở đây chúng ta chọn đầu sách có ISBN là “AHXD”

A screenshot of a computer

Description automatically generated

- Ta có thể sửa bất cứ thông tin nào của đầu sách này, ở đây ta sửa ISBN , tên sách và tác giả thành “TGHM” ; “THE GIOI HOAN MY” ; “THAN DONG” :

A screenshot of a computer

Description automatically generated

- Nhấn button “Sửa đầu sách” ta được kết quả khi sửa :

A screenshot of a computer

Description automatically generated

Phương thức **btnThemDS\_Click** - thêm một đầu sách mới vào danh sách, lưu thông tin của đầu sách vào file dữ liệu, đọc lại dữ liệu từ file và cập nhật lại bảng hiển thị danh sách đầu sách trên giao diện.

* Lấy thông tin của đầu sách mới từ các TextBox trên giao diện và lưu vào các biến tương ứng.
* Xóa nội dung của các TextBox để chuẩn bị cho việc nhập thông tin cho đầu sách tiếp theo.
* Lưu thông tin của đầu sách mới vào file dữ liệu bằng cách gọi phương thức SaveDataToFile của lớp BookFunc và truyền vào tên file và các thông tin của đầu sách.
* Đọc lại dữ liệu từ file bằng cách gọi phương thức ReadDataFromFile của lớp BookFunc.
* Sắp xếp lại danh sách các đầu sách theo mã ISBN bằng cách gọi phương thức SortByISBN của lớp BookFunc.
* Xóa nội dung của bảng hiển thị danh sách đầu sách bằng cách gán null cho thuộc tính DataSource của DataGridView.
* Cập nhật lại bảng hiển thị danh sách đầu sách bằng cách gán danh sách các đầu sách vào thuộc tính DataSource của DataGridView.
* Hiển thị thông báo thành công lên giao diện bằng hộp thoại MessageBox.

Code:

private void btnThemDS\_Click(object sender, EventArgs e)

{

string isbn = tbISBN.Text;

string tenSach = tbTenSach.Text;

string soTrang = tbSoTrang.Text;

string tacGia = tbTacGia.Text;

string namXB = tbNamXB.Text;

string theLoai = tbTheLoai.Text;

tbISBN.Clear();

tbTenSach.Clear();

tbSoTrang.Clear();

tbTacGia.Clear();

tbNamXB.Clear();

tbTheLoai.Clear();

BookFunc.SaveDataToFile("danh-sach-sach.txt", isbn, tenSach, soTrang, tacGia, namXB, theLoai);

BookFunc.ReadDataFromFile();

BookFunc.SortByISBN();

dgvDauS.DataSource = null; // Clear the data source

dgvDauS.DataSource = BookFunc.ListDauSach;

MessageBox.Show("Đã thêm và lưu dữ liệu thành công!");

}

Kết quả :  
- Nhập thông tin đầu sách muốn thêm vào danh sách đầu sách :

Graphical user interface, application

Description automatically generated

- Nhấn vào button “Thêm đầu sách” để thêm đầu sách vào danh sách :

A screenshot of a computer

Description automatically generated with medium confidence

- Đầu sách vừa được thêm đã cập nhật vào danh sách đầu sách :

A screenshot of a computer

Description automatically generated

Phương thức **btnSearch\_Click** - được sử dụng để tìm kiếm trong danh sách đầu sách.

* Lấy từ khóa tìm kiếm từ TextBox tbSearch và lưu vào biến tuKhoa.
* Kiểm tra nếu tuKhoa rỗng hoặc null, hiển thị thông báo cho người dùng và thoát khỏi hàm.
* Tạo một DataTable mới với các cột tương ứng với các trường của một DauSach và gán tên cột cho từng cột tương ứng.
* Duyệt qua danh sách các DauSach trong ListDauSach và kiểm tra xem tuKhoa có xuất hiện trong ISBN hoặc TenSach của từng DauSach không. Nếu có, tạo một dòng mới trong DataTable với các giá trị tương ứng của ISBN, TenSach, SoTrang, TacGia, NamXB, TheLoai.
* Gán DataTable vừa tạo vào DataSource của DataGridView dgvDauS để hiển thị kết quả tìm kiếm.

Code :

private void btnSearch\_Click(object sender, EventArgs e)

{

// Lấy dữ liệu từ TextBox tìm kiếm

string tuKhoa = tbSearch.Text.Trim();

// Kiểm tra xem người dùng đã nhập từ khóa tìm kiếm hay chưa

if (string.IsNullOrEmpty(tuKhoa))

{

MessageBox.Show("Vui lòng nhập từ khóa tìm kiếm", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

// Tạo DataTable để lưu trữ kết quả tìm kiếm

DataTable dt = new DataTable();

dt.Columns.Add("ISBN");

dt.Columns.Add("TenSach");

dt.Columns.Add("SoTrang");

dt.Columns.Add("TacGia");

dt.Columns.Add("NamXB");

dt.Columns.Add("TheLoai");

// Duyệt từng dòng trong DataGridView

foreach (DauSach ds in BookFunc.ListDauSach)

{

// Kiểm tra xem từ khóa có xuất hiện trong dòng hay không

if (ds.ISBN.ToLower().Contains(tuKhoa.ToLower()) || ds.TenSach.ToLower().Contains(tuKhoa.ToLower()))

{

string isbn = ds.ISBN;

string tensach = ds.TenSach;

string sotrang = ds.SoTrang.ToString();

string tacgia = ds.TacGia;

string namxb = ds.NamXB.ToString();

string theloai = ds.TheLoai;

// Nếu có, thêm dòng vào DataTable kết quả

dt.Rows.Add(isbn, tensach, sotrang, tacgia, namxb, theloai);

}

}

// Gán DataTable kết quả vào DataGridView

dgvDauS.DataSource = dt;

}

Kết quả :

- Nhập từ khóa tìm kiếm “ANH” và nhấn button “Tìm kiếm’ ta được các kết quả tìm kiếm có chứa “ANH” :

A screenshot of a computer

Description automatically generated

## Giao diện quản lí danh mục sách

\* Giao diện chính

**A screenshot of a computer

Description automatically generated with medium confidence**

* Sử dụng dataGrid để hiển thị danh sách DauSach
* Click vào DauSach bất kì hiển thị ra DanhMucSach của đầu sách đó
* Sử dụng các button để thực hiện chức năng thêm Sach vào DanhMucSach của đầu sách hiện tại

\* Hiển thị thông tin DanhMucSach lên dataGrid , giải thuật như sau :

* Đọc dữ liệu từ dữ liệu file lưu trữ DanhSachDoc cho trước
* Đưa dữ liệu vào dataGrid theo cấu trúc list

public void ThemSachVaoFileDanhMuc(string path, string ISBN, Sach sach)

{

try

{

List<string> lines = File.ReadAllLines(path).ToList();

for (int i = 0; i < lines.Count; i++)

{

string[] line = lines[i].Split('|');

if (line[0] == "DAUSACH" && line[1] == ISBN)

{

int soluong = int.Parse(line[2]);

soluong++;

line[2] = soluong.ToString();

string newLine = string.Join("|", line);

lines.Insert(i + soluong, sach.masach.ToString() + "|" +

sach.trangthai.ToString() + "|" + sach.vitri.ToString());

lines[i] = newLine;

break;

}

}

File.WriteAllLines(path, lines);

}

catch (Exception ex)

{

throw ex;

}

}

private void BtnThemSach\_Click(object sender, EventArgs e)

{

Sach sach = new Sach(int.Parse(tbMaSach.Text),

int.Parse(tbTrangThai.Text), int.Parse(tbViTri.Text), null);

ThemSachVaoFileDanhMuc("danh-muc-sach-ton-kho.txt", tbISBN.Text, sach);

XoasDanhMucSachCuaCacDauSach();

DanhMucSach.ReadDataFromFileDanhMucSach();

HienDanhMucSach();

dgvDanhMuc.DataSource = null;

dgvDanhMuc.DataSource = listsach;

dgvDanhMuc.Refresh();

}

Đây là phương thức trong lớp DanhMucSach và một sự kiện trong form (button click) để thêm một cuốn sách mới vào file danh mục sách.

* Phương thức ThemSachVaoFileDanhMuc nhận vào đường dẫn tệp, mã số ISBN của đầu sách và một đối tượng Sach để thêm vào file. Phương thức đọc tất cả các dòng trong tệp, tìm đến đầu sách có mã số ISBN trùng khớp với tham số truyền vào và tăng số lượng sách lên 1. Sau đó, phương thức chèn thông tin về cuốn sách mới vào file sau vị trí của đầu sách và cập nhật lại dữ liệu vào file.
* Trong sự kiện BtnThemSach\_Click, ta tạo một đối tượng Sach mới từ các giá trị nhập vào từ giao diện. Tiếp theo, ta gọi phương thức trên với đường dẫn và mã số ISBN của đầu sách tương ứng với cuốn sách mới để thêm sách vào file danh mục sách. Sau đó, ta cập nhật lại dữ liệu đọc từ file và hiển thị danh mục sách trên giao diện. Cuối cùng, ta refresh lại DataGridView để hiển thị thông tin mới nhất.

Kết Quả:

Với mỗi DauSach khác nhau thì có DanhMucSach khác nhau

- DanhMucSach của DauSach NOI GIONG RONG TIEN

A screenshot of a computer

Description automatically generated with medium confidence

- DanhMucSach của DauSach TAY DU KY

A screenshot of a computer

Description automatically generated with medium confidence

Tiếp theo kiểm tra nút Thêm Sách

- ­Thêm sách có mã 3005 vào DauSach AnhHungXaDieu

A screenshot of a computer

Description automatically generated with medium confidence

* Sau khi thêm thấy thành công thì kiểm tra lại file lưu trữ

A screenshot of a computer

Description automatically generated

- Cập nhật thêm sách thành công

## 2.3 Giao diện quản lí độc giả

\*Giao diện chính:

A screenshot of a computer

Description automatically generated

* Sử dụng dataGrid để hiển thị danh sách độc giả
* Sử dụng các textBox để nhập thông tin độc giả
* Sử dụng các button để thực hiện chức năng như : hiểu chỉnh , sắp xếp, thêm , xóa độc giả

Hiển thị thông tin độc giả lên dataGrid , giải thuật như sau :

* Đọc dữ liệu từ dữ liệu độc giả cho trước
* Đưa dữ liệu độc giả vào dataGrid theo cấu trúc linkedlist

Code :

private void Form1\_Load(object sender, EventArgs e)

{

ReaderFunc.DocFile();

DataGrid\_DocGia.DataSource = ReaderFunc.ListDocGia;

}

Chuyển các thông tin độc giả sang bảng thông tin độc giả , Giải thuật như sau :

* Nếu chỉ số hàng được chọn là -1, thoát khỏi hàm.
* Lấy dòng được chọn từ DataGridView bằng cách sử dụng chỉ số hàng được chọn trong đối số e của sự kiện CellClick.
* Lấy giá trị của các ô trong dòng được chọn và gán chúng vào các điều khiển TextBox tương ứng.
* Chuyển đổi kiểu dữ liệu của các giá trị cần thiết và gán chúng vào các điều khiển TextBox tương ứng.
* Chú ý rằng trong hàm, các giá trị được gán cho các điều khiển TextBox có thêm dấu cách ở đầu, điều này có thể gây ra sự cố nếu được sử dụng để phân tích sau này, vì vậy cần xóa các khoảng trắng này nếu không cần thiết.

Code :

private void DataGrid\_DocGia\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex == -1) return;

DataGridViewRow row = DataGrid\_DocGia.Rows[e.RowIndex];

int MaDG = int.Parse(row.Cells[0].Value.ToString());

txtMaDG.Text = " ";

txtMaDG.Text += MaDG;

string Ho = row.Cells[1].Value.ToString();

txtHo.Text = Ho;

string Ten = row.Cells[2].Value.ToString();

txtTen.Text = Ten;

int Phai = int.Parse(row.Cells[3].Value.ToString());

txtPhai.Text = " ";

txtPhai.Text += Phai;

int TrangThaiThe = int.Parse(row.Cells[4].Value.ToString());

txtTrangThai.Text = " ";

txtTrangThai.Text += TrangThaiThe;

}

Chuyển dạng 0/1 sang Nam/Nữ và Hoạt động/Không hoạt động : xử lý sự kiện định dạng ô cho một điều khiển DataGridView hiển thị thông tin về độc giả. Cụ thể, nó thay đổi giá trị hiển thị của hai cột (cột 3 và cột 4) thành văn bản dễ đọc hơn :

Nếu chỉ số cột là 3 và giá trị ô không null, thực hiện các bước sau:

* Nếu giá trị ô là "1", đặt giá trị ô thành "Nam".
* Ngược lại, đặt giá trị ô thành "Nữ".
* Đặt FormattingApplied thành true.

Nếu chỉ số cột là 4 và giá trị ô không null, thực hiện các bước sau:

* + Nếu giá trị ô là 1, đặt giá trị ô thành "Hoạt động".
  + Ngược lại, đặt giá trị ô thành "Không hoạt động".
  + Đặt FormattingApplied thành true.

Code :

private void DataGrid\_DocGia\_CellFormatting\_1(object sender, DataGridViewCellFormattingEventArgs e)

{

if (e.ColumnIndex == 3 && e.Value != null)

{

if ((string)e.Value == "1")

{

e.Value = "Nam";

}

else

{

e.Value = "Nữ";

}

e.FormattingApplied = true;

}

if (e.ColumnIndex == 4 && e.Value != null)

{

if ((int)e.Value == 1)

{

e.Value = "Hoạt động";

}

else

{

e.Value = "Bị khóa ";

}

e.FormattingApplied = true;

}

}

Button Nhập độc giả: thêm / xóa / hiệu chỉnh thông tin độc giả. Mã độc giả được chương trình tự động lấy ngẫu nhiên và không trùng với các mã độc giả cũ; Phái chỉ nhận ‘Nam’ hoặc ‘Nữ’; trạng thái thẻ = 0 khi thẻ bị khóa, trạng thái thẻ = 1 khi thẻ đang hoạt động (được quyền mượn sách).

Giải thuật :

* Ta ứng dụng các thuật toán của các hàm đã thiết lập từ trước như : GetMaDG() – tạo mã độc giả ngẫu nhiên không trùng , AddReader() – Thêm độc giả , EditReader() – Hiệu chỉnh độc giả , XoaDocGia(MaDG) – Xóa độc giả theo mã độc giả
* Phải chỉ nhận “ Nam “ hoặc “ Nữ” , ta xét giá trị đầu vào chỉ được nhận 0 hoặc 1 , nếu 0 là “Nữ” , 1 là “Nam” . Tương tự ở Trạng thái thẻ 1 – thẻ hoạt đông , 0 – “ Bị khóa “

Chức năng thêm độc giả :

Hàm này được sử dụng để thêm một độc giả mới vào hệ thống. Dưới đây là giải thuật của hàm:

* Thiết lập nguồn dữ liệu cho DataGridView là null.
* Tạo một danh sách mới để lưu thông tin độc giả (ReaderFunc.ListDocGia).
* Khởi tạo một đối tượng DocGia mới (newReader) với các thuộc tính được cung cấp từ các thành phần giao diện người dùng (txtHo, txtTen, txtPhai, txtTrangThai), và lấy mã độc giả bằng cách gọi phương thức ReaderFunc.GetMaDG().
* Thêm độc giả mới vào hệ thống bằng cách gọi phương thức ReaderFunc.AddReader(newReader).
* Lưu danh sách độc giả vào file bằng cách gọi phương thức ReaderFunc.LuuDanhSachDocGiaVaoFile().
* Xóa tất cả các độc giả khỏi danh sách độc giả bằng cách gọi phương thức ReaderFunc.XoaTatCaDocGia().
* Hiển thị thông báo cho người dùng về việc thêm độc giả mới thành công.
* Đọc lại danh sách độc giả từ file bằng cách gọi phương thức ReaderFunc.DocFile().
* Thiết lập nguồn dữ liệu cho DataGridView là danh sách độc giả mới nhất bằng cách gán ReaderFunc.ListDocGia cho DataGrid\_DocGia.DataSource.

Code :

private void btnThem\_Click(object sender, EventArgs e)

{

DataGrid\_DocGia.DataSource = null;

ReaderFunc.ListDocGia = new List<DocGia>();

DocGia newReader;

int a = ReaderFunc.GetMaDG();

newReader = new DocGia() { MaDG = a, Ho = txtHo.Text, Ten = txtTen.Text, Phai = txtPhai.Text, TrangThaiThe = int.Parse(txtTrangThai.Text) };

ReaderFunc.AddReader(newReader);

ReaderFunc.LuuDanhSachDocGiaVaoFile();

ReaderFunc.XoaTatCaDocGia();

MessageBox.Show("Thêm độc giả thành công !! ");

ReaderFunc.DocFile();

DataGrid\_DocGia.DataSource = ReaderFunc.ListDocGia;

}

Kết quả :

Trước khi thêm độc giả :

A screenshot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated with low confidence

- Thêm độc giả : PHAM THI LIN DA vào

A screenshot of a computer

Description automatically generated

- Kiểm tra độc giả LIN DA đã được thêm vào hay chưa :

A screenshot of a computer

Description automatically generated

- Kiểm tra trên file dữ liệu :

A screenshot of a computer

Description automatically generated with medium confidence

Hiệu chỉnh độc giả PHAM THI LIN DA :

A screenshot of a computer

Description automatically generated

- Hiệu chỉnh độc giả LIN DA : từ nữ - Nam , Hoạt động – bị khóa

A screenshot of a computer

Description automatically generated

- Kiểm tra xem đã cập nhật chưa :

A screenshot of a computer

Description automatically generated

- Xóa độc giả PHAM THI LIN DA :

A screenshot of a computer

Description automatically generated

- Kiểm tra xem đã xóa độc giả LIN DA hay chưa :

A screenshot of a computer

Description automatically generated

- Kiểm tra trên file dữ liệu :

A screenshot of a computer

Description automatically generated with low confidence

In danh sách độc giả : in ra màn hình theo thứ tự tên+họ tăng dần hoặc theo thứ tự mã độc giả tăng dần theo yêu của thủ thư.

Giải thuật :

* + Sử dụng giải thuật insertion sort từ các hàm InsertionSortByMa() và InsertionSortByTen() từ trước .
  + Lưu các Node thừ sơ đồ cây nhị phân sau khi được sắp xếp thành một linkedlist để hiển thị lên dataGrid – vì dataGrid chỉ hỗ trợ định dạng collection base

In độc giả theo thứ tự tên + họ tăng dần :

Giải thuật :

* + Xóa dữ liệu trong dataGrid từ trước
  + Thêm dữ liệu thừ list trong hàm InDanhSachDocGiaTheoTen()

Code :

private void btnSapTheoTen\_Click(object sender, EventArgs e)

{

DataGrid\_DocGia.DataSource = null;

DataGrid\_DocGia.DataSource= ReaderFunc.InDanhSachDocGiaTheoTen();

}

In độc giả theo thứ tự mã độc giả tăng dần :

Giải thuật :

* + Xóa dữ liệu trong dataGrid từ trước
  + Thêm dữ liệu thừ list trong hàm InDanhSachDocGiaTheoMa()

Code :

private void btnSapTheoMaThe\_Click(object sender, EventArgs e)

{

DataGrid\_DocGia.DataSource = null;

DataGrid\_DocGia.DataSource = ReaderFunc.InDanhSachDocGiaTheoMa();

}

Kết quả :

- In độc giả theo họ + tên :

A screenshot of a computer

Description automatically generated

- In độc giả theo Mã độc giả :

A screenshot of a computer

Description automatically generated with medium confidence

## 2.4 Giao diện phần mượn trả sách:

\* Giao diện chính:

Graphical user interface, application

Description automatically generated

Giao diện gồm các thành phần chính sau:

- 4 button:

+ btnTimMaDG: Dùng để xử lí và in ra thông tin độc giả

+ btnTimSach: Dùng để xử lí và in ra thông tin sách

+ btnMuonSach: Dùng để xử lí và mượn sách

+ btnTraSach: Dùng để xử lí và trả sách

* 2 textbox:

+ txtMaDG: Dùng để nhập vào mã độc giả

+ txtMaSach: Dùng để nhập vào mã sách

* 3 label

+ lbTen: Dùng để in ra họ tên độc giả

+ lbPhai: in ra phái của độc giả

+ lbTTT: in ra trạng thái thẻ của độc giả

* 2 Datagrid view:

+ GridDSM: In ra danh sách sách độc giả đang mượn

+ GridViewSach: In ra thông tin sách tìm kiếm

\* Khởi tạo các đối tượng mới

DanhMucSach DM\_S;

DS\_MuonTra DS\_MT;

private void MuonTraSach\_Load(object sender, EventArgs e)

{

DS\_MT = new DS\_MuonTra();// Khởi tạo Một Danh Sách Mượn Trả

DM\_S = new DanhMucSach();// Khởi tạo một Danh Mục Sách

}

\* btnTimMaDG:

- Code xử lí bntTimMaDG:

public void bntTimMaDG\_Click(object sender, EventArgs e)

{

ReaderFunc.XoaTatCaDocGia();//Xóa danh sách độc giả

ReaderFunc.DocFile(); // Lưu danh sách độc giả từ file

int MaDG = int.Parse(txtMaDG.Text);

//Tạo một đối tượng độc giả mới và lưu thông tin từ bằng phương thức GetReaderByMaDG(MaDG)

DocGia DG = ReaderFunc.GetReaderByMaDG(MaDG);

DG.SachDangMuon.Xoahetdsmuon(); //Xóa hết danh sách mượn của DG

DG.SachDangMuon.ReadMuonTraDataFromFile(); //Lưu Danh sách mượn mới cho DG

//Xử lí và in ra thông tin độc giả

if (DG != null)

{

lbTen.Text = DG.Ho + " " + DG.Ten;

if (DG.TrangThaiThe == 1)

{

lbPhai.Text = "NAM";

}

else

{

lbPhai.Text = "NU";

}

if (DG.TrangThaiThe == 1)

{

lbTTThe.Text = "Dang hoat dong";

}

else

{

lbTTThe.Text = "The bi khoa";

}

}

else

{

MessageBox.Show("Ma doc gia sai ");

}

GridDSM.DataSource = null;

//In danh sách độc giả đang mượn ra GridDSM

GridDSM.DataSource = DG.SachDangMuon.LinkedListToListMT();

}

- Quy trình xử lí của đoạn code trên:

+ Xóa danh sách độc giả hiện tại và đọc lại danh sách độc giả từ file và lưu vào danh sách độc giả.

+ Lấy mã độc giả từ textbox.

+ Tìm độc giả theo mã trong danh sách.

+ Xóa danh sách mượn hiện tại của độc giả.

+ Đọc danh sách mượn từ file và lưu vào danh sách mượn của độc giả.

+ Xét điều kiện của Mã độc giả nếu có trong danh sách hiển thị thông tin độc giả (Họ tên, Giới tính, Trạng thái thẻ) trên giao diện.

+ Xóa dữ liệu hiển thị trong GridDSM cũ và Hiển thị danh sách mượn của độc giả trên GridDSM.

\* btnTimSach:

- Code xử lí btnTimSach:

private void btnTimSach\_Click(object sender, EventArgs e)

{

int MaSach = int.Parse(txtMaSach.Text);

DM\_S.ThemSachVaolinklist();

Sach sach = DM\_S.GetSachBangMaSach(MaSach);

List<Sach> S = new List<Sach>();

S.Add(sach);

GridViewSach.DataSource = S;

}

- Quy trình xử lí của đoạn code trên

+ Lấy mã sách từ textbox và chuyển đổi sang kiểu số nguyên.

+ Thêm sách vào danh sách liên kết.

+ Tìm sách trong danh sách theo mã sách.

+ Tạo một danh sách mới chứa sách tìm được.

+ Gán danh sách mới này làm nguồn dữ liệu cho GridViewSach (hiển thị trên giao diện).

\* btnMuonSach:

- Code xử lí:

private void btnMuonSach\_Click(object sender, EventArgs e)

{

List<MuonTra> DSMT = new List<MuonTra>();

int MaDG = int.Parse(txtMaDG.Text);

int MaSach = int.Parse(txtMaSach.Text);

DocGia DG = ReaderFunc.GetReaderByMaDG(MaDG);

DM\_S.ThemSachVaolinklist();

Sach dsach = DM\_S.GetSachBangMaSach(MaSach);

DG.SachDangMuon.Xoahetdsmuon();

DG.SachDangMuon.ReadMuonTraDataFromFile();

DSMT.Clear();

DSMT = DG.SachDangMuon.LinkedListToListMT();

DS\_MT.Xoahetdsmuon();

DS\_MT.Themsachvaolinklist();

int Sachquahan = 0;

if (DG.TrangThaiThe == 0)

{

MessageBox.Show("Doc gia khong con hoat dong nen khong muon duoc sach");

}

else

{

if (DG.SachDangMuon.Count() == 3)

{

MessageBox.Show("Doc gia da muon du so luong sach");

}

else

{

if (dsach.trangthai == 1 || dsach.trangthai == 2)

{

MessageBox.Show("Sach da cho muon hoac thanh li nen khong the muon");

}

else

{

foreach (MuonTra mt in DSMT)

{

TimeSpan timeSpan = DateTime.Now - DateTime.Parse(mt.NgayMuon);

double totalDays = timeSpan.TotalDays;

if (totalDays > 15)

{

Sachquahan++;

}

}

if(Sachquahan == 0)

{

DS\_MT.MuonSach(MaDG, MaSach);

DS\_MT.UpdateMuonTraDataToFile();

DM\_S.UpdateMuonInFile(MaSach.ToString());

}

else

{

MessageBox.Show("Doc gia da muon sach qua han, hay tra sach qua han de duoc muon tiep");

}

}

}

}

DM\_S.XoaDanhSach();

DM\_S.ThemSachVaolinklist();

Sach sach = DM\_S.GetSachBangMaSach(MaSach);

GridViewSach.DataSource = null;

List<Sach> S = new List<Sach>();

S.Add(sach);

GridViewSach.DataSource = S;

DG.SachDangMuon.Xoahetdsmuon();

DG.SachDangMuon.ReadMuonTraDataFromFile();

GridDSM.DataSource = DG.SachDangMuon.LinkedListToListMT();

}

- Quy trình xử lí của đoạn code trên:

+ Lấy mã độc giả và mã sách từ các textbox.

+ Thêm sách vào danh sách liên kết của sách.

+ Tìm độc giả trong danh sách theo mã độc giả.

+ Xóa hết danh sách mượn của độc giả và đọc lại danh sách từ file.

+ Xóa danh sách mượn trả và đọc danh sách từ file.

+ Kiểm tra trạng thái và số lượng sách mượn của độc giả.

+ Kiểm tra trạng thái của sách.

+ Duyệt qua danh sách mượn trả để kiểm tra số sách quá hạn.

+ Nếu không có sách quá hạn, thực hiện mượn sách, cập nhật dữ liệu vào file và cập nhật mượn sách trong danh sách sách.

+ Nếu có sách quá hạn, thông báo cho người dùng.

+ Cập nhật lại danh sách sách và danh sách mượn của độc giả.

+ Hiển thị thông tin sách và danh sách mới mượn lên giao diện.

\* btnTraSach:

- Code xử lí:

private void btnTraSach\_Click(object sender, EventArgs e)

{

int MaDG = int.Parse(txtMaDG.Text);

int MaSach = int.Parse(txtMaSach.Text);

DocGia DG = ReaderFunc.GetReaderByMaDG(MaDG);

DM\_S.XoaDanhSach();

DM\_S.ThemSachVaolinklist();

DG.SachDangMuon.Xoahetdsmuon();

DG.SachDangMuon.ReadMuonTraDataFromFile();

MuonTra Sachdangmuon = DG.SachDangMuon.GetMuonTrabyMaSach(MaSach);

DS\_MT.Xoahetdsmuon();

DS\_MT.Themsachvaolinklist();

if(Sachdangmuon == null)

{

MessageBox.Show("Sach nay doc gia chua muon");

}

else

{

DS\_MT.TraSach(MaDG, MaSach);

DS\_MT.UpdateMuonTraDataToFile();

DM\_S.UpdateTraInFile(MaSach.ToString());

}

DM\_S.XoaDanhSach();

DM\_S.ThemSachVaolinklist();

Sach sach = DM\_S.GetSachBangMaSach(MaSach);

GridViewSach.DataSource = null;

List<Sach> S = new List<Sach>();

S.Add(sach);

GridViewSach.DataSource = S;

DG.SachDangMuon.Xoahetdsmuon();

DG.SachDangMuon.ReadMuonTraDataFromFile();

GridDSM.DataSource = DG.SachDangMuon.LinkedListToListMT();

}

- Quy trình xử lí của đoạn code trên:

* Lấy mã độc giả và mã sách từ các textbox.
* Tìm độc giả trong danh sách theo mã độc giả.
* Xóa danh sách sách và thêm sách vào danh sách từ file.
* Xóa hết danh sách mượn của độc giả và đọc lại danh sách từ file.
* Tìm thông tin sách đang mượn bằng mã sách.
* Xóa danh sách mượn trả và đọc danh sách từ file.
* Kiểm tra nếu sách đang mượn không tồn tại, thông báo cho người dùng.
* Nếu sách đang mượn tồn tại, thực hiện trả sách, cập nhật dữ liệu vào file và cập nhật trả sách trong danh sách sách.
* Cập nhật lại danh sách sách và danh sách mượn của độc giả.
* Hiển thị thông tin sách và danh sách mượn lên giao diện.

\* Kết quả:

Graphical user interface, application, Word

Description automatically generated

Nhập vào mã độc giả là 13477 và hiển thị ra thông tin độc giả và sách độc giả đang mượn

Graphical user interface

Description automatically generated

Nhập vào mã độc giả 15898, kiểm tra trong danh sách không có mã độc giả này và in ra Ma doc gia sai

Tìm sách với mã 1001 và in ra thông tin sách và mượn sách này cho độc giả có mã 13942

Graphical user interface, application, Word

Description automatically generated

Kiểm tra file txt để xem cập nhật

Graphical user interface, text, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated

- Các dữ liệu có đầu sách là 1001 trong file txt đã cập nhật thành công

Trả sách có mã 1001 cho độc giả có mã 13942

Graphical user interface, application, Word

Description automatically generated

Kiểm tra file txt để xem cập nhật dữ liệu

Graphical user interface

Description automatically generated with medium confidence

- Trạng thái sách 1001 được đặt về 0

Graphical user interface, text, application

Description automatically generated

- Sách có mã sách 1001 trong file data\_muontra đã cập nhật lại mã độc giả về 0, ngày mượn đúng với thời gian vừa trả

Độc giả có mã 13942 mượn thêm 2 sách đủ ba sách và mượn thêm sách nữa

Graphical user interface, application, Word

Description automatically generated

- Thông báo “độc giả đã mượn đủ số sách” và không thể mượn thêm sách

Graphical user interface, application, Word

Description automatically generated

* Độc giả có mã 15973 mượn sách có mã 1001, sách này đã cho mượn nên in ra thông báo “Sach đã cho mượn hoặc thanh lí nên không thể mượn”

Graphical user interface, application, Word

Description automatically generated

- Độc giả có mã 18802 mượn sách có mã 2003 ngày 24/4/2023 tính tới hiện tại là 14/5/2023 đã quá 15 ngày nên sách này đã quá hạn nên in ra thông báo “Doc gia muon sach qua han, hay tra sach qua han để được mượn tiếp”