BÁO CÁO THỰC HÀNH

Môn học: Nhập môn mạng máy tính

Buổi báo cáo: Lab 02

Tên chủ đề: File and Stream I/O in C#

GVHD: Nguyễn Xuân Hà

Ngày thực hiện: 30/03/2023

THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)
Lớp: NT106.O23.1

STT	Họ và tên	MSSV	Email
1	Trần Thiện Mạnh	22520853	22520853@gm.uit.edu.vn

1. ĐÁNH GIÁ KHÁC:

Nội dung	Kết quả
Tổng thời gian thực hiện bài thực hành trung bình: 10 ngày	21/03/2024 – 30/03/2024
Link Video thực hiện *(nếu có)*	
Ý kiến *(nếu có)* + Khó khăn + Đề xuất …	
Điểm tự đánh giá	8/10

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện.

BÁO CÁO CHI TIẾT

Bài 1: Ghi và đọc File

Dùng richTextBox để hiển thị ra nội dung file xử lí các button "Đọc File", "Ghi File", "Thoát", "Reset"

OpenFileDialog:
- Được sử dụng để mở hộp thoại chọn file từ hệ thống tệp tin.
- Thuộc tính **Filter** được sử dụng để chỉ định loại các file mà người dùng có thể chọn (trong trường hợp này, chỉ các file có đuôi **.txt**).

SaveFileDialog:
- Được sử dụng để mở hộp thoại lưu file đến vị trí được chỉ định.
- Thuộc tính **Filter** được sử dụng để chỉ định loại các file mà người dùng có thể lưu (trong trường hợp này, chỉ các file có đuôi **.txt**).
- Thuộc tính **FileName** được sử dụng để chỉ định tên mặc định của file khi nó được lưu.

StreamReader:
- Được sử dụng để đọc nội dung của một file văn bản.
- Phương thức **ReadToEndAsync()** được sử dụng để đọc toàn bộ nội dung của file và trả về dưới dạng một chuỗi.

File.WriteAllTextAsync:
- Được sử dụng để ghi một chuỗi nào đó vào một file văn bản.
- Tham số đầu tiên là đường dẫn đến file mà chuỗi sẽ được ghi vào.
- Tham số thứ hai là chuỗi cần được ghi vào file.

MessageBox:
- Được sử dụng để hiển thị thông báo cho người dùng.
- Phương thức **Show()** được sử dụng để hiển thị một thông báo với nội dung được chỉ định.

RichTextBox:
- Là một điều khiển giao diện người dùng dùng để hiển thị và chỉnh sửa văn bản.
- Thuộc tính **Text** được sử dụng để đặt hoặc lấy nội dung của **RichTextBox**.

async/await:
- **async** được sử dụng để chỉ định phương thức bất đồng bộ.
- **await** được sử dụng để chờ cho một tác vụ bất đồng bộ kết thúc mà không làm đóng băng luồng thực thi của ứng dụng.

Bài 02 – Đọc thông tin một file .txt

Dùng richTextBox để hiển thị ra nội dung file xử lí các button "Read from File", "Exit", hiên thị các mục chi tiết của file hiển thị ở richTextBox ra các TextBox

Mở hộp thoại để chọn file:
- Tạo một instance mới của **OpenFileDialog**.
- Thiết lập thuộc tính **Filter** để chỉ định loại các file mà người dùng có thể chọn (trong trường hợp này, chỉ các file có đuôi **.txt**).
- Hiển thị hộp thoại và chờ người dùng chọn một file bằng cách gọi **ShowDialog().** Nếu người dùng chọn OK, tiếp tục thực hiện các thao tác khác.

Đọc nội dung file và hiển thị lên RichTextBox:
- Sử dụng **StreamReader** để đọc nội dung của file đã chọn và gán nó cho thuộc tính **Text** của **RichTextBox**.
- Lấy tên của file và hiển thị nó trong một TextBox khác.

Lấy thông tin về file:
- Tạo một instance của **FileInfo** từ đường dẫn của file.
- Lấy kích thước của file thông qua thuộc tính **Length** của **FileInfo** và hiển thị nó trong một TextBox.

Lấy URL của file:
- Đơn giản là hiển thị đường dẫn đến file (tức là **FileName** của **OpenFileDialog**) trong một TextBox.

Đếm số dòng của file:
- Sử dụng **File.ReadAllLines** để đọc tất cả các dòng của file và lưu chúng vào một mảng string.
- Đếm số phần tử trong mảng string để biết số dòng và hiển thị nó trong một TextBox.

Đếm số từ của file:
- Lặp qua mỗi dòng của file đã đọc và sử dụng một vòng lặp for để đếm số khoảng trắng hoặc ký tự kết thúc dòng để xác định số từ trong mỗi dòng.
- Tổng hợp số từ của tất cả các dòng để đếm tổng số từ trong file và hiển thị nó trong một TextBox.

Đếm số ký tự của file:
- Lặp qua từng ký tự của mỗi dòng của file và đếm tổng số ký tự.
- Đếm số ký tự xuống dòng (\n) để tính số dòng và thêm vào tổng số ký tự.
- Hiển thị tổng số ký tự trong một TextBox.

Bài 03 - Đọc và Ghi file và tính toán

Dùng hai richTextBox, 1 richTextBox_input để hiển thị ra file muốn thực thi và sử dụng 1 richTextBox_ouput để hiển thị file sau thực thi để xuất ra nội dung ở file input nhập ở richTextBox_input ở file input vừa đọc ra và lưu ở file output

Phương thức btn_Read_Click:
- Mục đích: Được gọi khi người dùng nhấn nút "Read", phương thức này cho phép người dùng chọn một file văn bản và hiển thị nội dung của file đó trong một **RichTextBox**.
- **try-catch**: Sử dụng để bắt lỗi trong quá trình đọc file.

- **OpenFileDialog**: Tạo một hộp thoại mở file để người dùng chọn file văn bản.
- **if (openFile.ShowDialog() == DialogResult.OK):** Kiểm tra nếu người dùng đã chọn file và nhấn OK trong hộp thoại, tiếp tục thực hiện.
- **StreamReader**: Đọc nội dung của file đã chọn và gán vào **RichTextBox**.
- **MessageBox**: Hiển thị thông báo khi đã đọc xong file hoặc gặp lỗi.

Phương thức CalculateExpression:

- Mục đích: Tính toán giá trị của một biểu thức toán học được cung cấp dưới dạng chuỗi.
- Đầu vào: Chuỗi biểu thức toán học.
- Quá trình tính toán:
  - Loại bỏ khoảng trắng từ biểu thức.
  - Tách biểu thức thành các toán hạng và toán tử.
  - Thực hiện các phép tính cơ bản như cộng, trừ, nhân, chia và trả về kết quả.

Phương thức btn_Cal_Click:

- Mục đích: Được gọi khi người dùng nhấn nút "Calculate", phương thức này tính toán các biểu thức trong **RichTextBox** và lưu kết quả vào một file mới.
- **if (string.IsNullOrEmpty(richTextBox_input.Text))**: Kiểm tra xem **RichTextBox** có nội dung hay không. Nếu không, hiển thị thông báo lỗi.
- **SaveFileDialog:** Tạo một hộp thoại lưu file để người dùng chọn vị trí và tên cho file được ghi.
- Tính toán và lưu kết quả:
  - Duyệt qua mỗi dòng của **RichTextBox** và tính toán giá trị của từng dòng bằng cách gọi phương thức **CalculateExpression**.
  - Lưu kết quả tính toán vào một danh sách.
- Ghi kết quả tính toán xuống file mới và hiển thị nội dung của file đó trong một **RichTextBox**.
- **MessageBox**: Hiển thị thông báo khi đã ghi xong file hoặc gặp lỗi.

Bài 4 - Đọc và Ghi file sử dụng BinaryFormatter (JsonSerializer)

Constructor bai4():

- Trong constructor này, **InitializeComponent()** được gọi để khởi tạo tất cả các thành phần giao diện người dùng được thiết kế trong Visual Studio.
- **List<Student> _students** được khởi tạo để lưu trữ danh sách các đối tượng sinh viên. Điều này giúp ứng dụng theo dõi và quản lý thông tin của mỗi sinh viên.

Class Student:

- Là một lớp đại diện cho sinh viên, có các thuộc tính như tên, ID, số điện thoại, điểm các môn học và điểm trung bình.
- Sử dụng các thuộc tính để lưu trữ thông tin của từng sinh viên.

Các phương thức kiểm tra thông tin nhập vào (IsEmpty, IsPhoneValid, IsStudentIDValid, IsScoreValid):

- Các phương thức này kiểm tra xem thông tin nhập vào từ giao diện người dùng có hợp lệ không. Ví dụ: kiểm tra xem các ô nhập liệu có trống không, kiểm tra định dạng số điện thoại, mã sinh viên, điểm số có đúng không.

- Quan trọng để đảm bảo rằng dữ liệu được nhập vào là hợp lệ trước khi tiếp tục xử lý.

Biến và danh sách _students:

- **List<Student> _students** là một danh sách để lưu trữ đối tượng sinh viên. Mỗi sinh viên được thêm vào danh sách sau khi thông tin của họ đã được nhập vào và xác thực.
- Biến **_index** được sử dụng để theo dõi chỉ mục hiện tại của sinh viên đang được hiển thị trong danh sách.

Phương thức btn_Add_Click:

- Xử lý sự kiện khi người dùng nhấn nút "Add".
- Kiểm tra tính hợp lệ của thông tin được nhập vào từ các ô nhập liệu.
- Tạo một đối tượng **Student** mới từ thông tin đã nhập và thêm vào danh sách **_students**.
- Làm sạch các ô nhập liệu sau khi thêm sinh viên.

Phương thức btn_Read_Click:

- Xử lý sự kiện khi người dùng nhấn nút "Read".
- Mở một hộp thoại để chọn file văn bản.
- Đọc thông tin sinh viên từ file văn bản và thêm vào danh sách **_students**.
- Cập nhật giao diện hiển thị sau khi đọc xong.

Phương thức btn_Write_Click:

- Xử lý sự kiện khi người dùng nhấn nút "Write".
- Mở một hộp thoại để lưu file văn bản.
- Ghi thông tin sinh viên từ danh sách **_students** vào file văn bản.

Các phương thức ClearTextBoxInput và UpdateStatus:

- **ClearTextBoxInput**: Xóa nội dung của các ô nhập liệu trong giao diện.
- **UpdateStatus**: Cập nhật giao diện hiển thị danh sách sinh viên sau khi thêm hoặc đọc dữ liệu.

Phương thức UpdateOutputTextBox:

- Cập nhật thông tin của sinh viên được chọn vào các ô hiển thị thông tin sinh viên.

Các phương thức xử lý sự kiện khi nhấn nút "Exit", "Back", "Next":

- **btn_Exit_Click**: Đóng form hiện tại.
- **btn_Back_Click**: Hiển thị thông tin của sinh viên trước đó trong danh sách.
- **btn_Next_Click**: Hiển thị thông tin của sinh viên tiếp theo trong danh sách

Bài 07 – Duyệt thư mục

Phương thức bai7_Load:

- Phương thức này được gọi khi form được load lên.
- **DriveInfo.GetDrives()** trả về một mảng các đối tượng **DriveInfo**, mỗi đối tượng đại diện cho một ổ đĩa trong máy tính.
- Vòng lặp foreach duyệt qua mỗi ổ đĩa và tạo một nút treenode đại diện cho ổ đĩa đó.
- Nếu ổ đĩa tồn tại, gọi hai hàm **LoadFolder** và **LoadFile** để tải các thư mục và tệp tin của ổ đĩa đó.
- Cuối cùng, thêm nút treenode của ổ đĩa vào treeView.

Phương thức treeView_AfterSelect:

- Phương thức này được gọi khi một nút trong treeView được chọn.

- Kiểm tra xem nút được chọn có tham chiếu đến một đường dẫn không. Nếu không, thoát khỏi phương thức.
- Nếu nút đó tham chiếu đến một tệp tin, và tệp tin không phải là tệp nhị phân, thì đọc nội dung của tệp tin và hiển thị nó trong richTextBoxFile.
- Nếu nút đó tham chiếu đến một hình ảnh, thì đọc hình ảnh từ đường dẫn và hiển thị nó trong pictureBoxFile.
- Nếu nút đó không phải là tệp tin hoặc thư mục, thì tải thêm các thư mục và tệp tin bên trong thư mục đó bằng cách gọi lại hàm LoadFolder và LoadFile.

Phương thức LoadFolder:

- Hàm này được sử dụng để tải các thư mục con trong một thư mục cụ thể.
- Sử dụng **Directory.GetDirectories(path)** để lấy danh sách các thư mục con trong thư mục được chỉ định.
- Đối với mỗi thư mục con, tạo một treenode mới và thêm vào treeNodeCollection (tức là các nút con của nút hiện tại).

Phương thức LoadFile:

- Hàm này được sử dụng để tải các tệp tin trong một thư mục cụ thể.
- Sử dụng **Directory.GetFiles(path)** để lấy danh sách các tệp tin trong thư mục được chỉ định.
- Đối với mỗi tệp tin, tạo một treenode mới và thêm vào treeNodeCollection.

Phương thức IsImageFile:

- Hàm này kiểm tra xem một tệp tin có phải là tệp hình ảnh không.
- Đọc một số byte đầu của tệp tin để xác định loại hình ảnh (JPEG, PNG, GIF, BMP, TIFF, WAV).
- Trả về true nếu là tệp hình ảnh, ngược lại trả về false.

Phương thức IsBinaryFile:

- Hàm này kiểm tra xem một tệp tin có phải là tệp nhị phân không.
- Đọc dữ liệu từ tệp tin và kiểm tra xem có byte nào có giá trị là 0 không.
- Trả về true nếu là tệp nhị phân, ngược lại trả về false.

YÊU CẦU CHUNG

1) Đánh giá

- Chuẩn bị tốt các yêu cầu đặt ra trong bài thực hành.
- Sinh viên hiểu và tự thực hiện được bài thực hành, trả lời đầy đủ các yêu cầu đặt ra.
- Nộp báo cáo kết quả chi tiết những đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả *(nếu có)*; giải thích cho quan sát *(nếu có)*.
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

2) Báo cáo

- File .PDF hoặc .docx. Tập trung vào nội dung, giải thích.
- Nội dung trình bày bằng Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Avo)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
- Đặt tên theo định dạng: LabX_MSSV1_MSSV2. (trong đó X là Thứ tự buổi Thực hành).

  Ví dụ: Lab01_21520001_21520002
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT