2

Lab

File và I/O Stream

File I/O and I/O Streams

Thực hành Lập trình mạng căn bản

Cập nhật: 09/2024

Lưu hành nội bộ

A. TỔNG QUAN

1. Mục tiêu

- Cung cấp khả năng khởi tạo, đọc, viết và khả năng cập nhật File.
- Hiểu được luồng thông tin (Stream) trong C#.
- Có thể sử dụng được các lớp FileStream, lớp StreamReader, lớp StreamWriter và lớp BinaryFormatter để đọc và viết các đối tượng vào trong các File.

2. Môi trường

- IDE Microsoft Visual Studio 2010 trở lên.

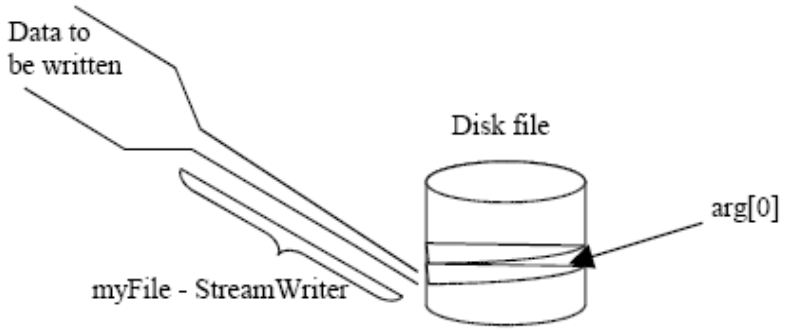
3. Liên quan

- Sinh viên cần nắm được các kiến thức nền tảng về lập trình. Các kiến thức này đã được giới thiệu trong các môn học trước và trong nội dung lý thuyết đã học do đó sẽ không được trình bày lại trong nội dung thực hành này.
- Tham khảo tài liệu (Mục E) để có kiến thức cơ bản về C#, Winforms.

B. KIẾN THỨC NỀN TẢNG

1. Luồng (Stream) và Tập tin (File) trong C#

Luồng (Stream) là luồng của thông tin, chứa thông tin sẽ được chuyển qua. Tập tin (File) được dùng để lưu trữ thông tin, dữ liệu. File và Stream I/O (Input/Output) đề cập đến việc truyền dữ liệu đến hoặc từ một phương tiện lưu trữ. Trong .NET Framework, namespace System.IO bao gồm các loại có khả năng đọc và ghi (đồng bộ và không đồng bộ) các luồng dữ liệu và file. Những namespace này chứa các loại namespace thực hiện việc nén và giải nén các file.



Hình 1 : Mô tả thực hiện tạo tập tin và đưa dữ liệu vào

Dữ liệu được truyền theo hai hướng:

- Đọc dữ liệu: đọc dữ liệu từ bên ngoài vào chương trình.
- Ghi dữ liệu: đưa dữ liệu từ chương trình ra bên ngoài.

Streams

Lớp Stream hỗ trợ đọc và ghi dữ liệu theo byte. Tất cả các lớp xử lý các luồng khác nhau hầu hết kế thừa từ lớp Stream. Lớp Stream và các lớp dẫn xuất cung cấp một cái nhìn chung về các nguồn dữ liệu và giúp lập trình viên không cần phải đi quá chi tiết về các đặc điểm của hệ điều hành và các thiết bị bên dưới.

Streams thường có ba thao tác cơ bản:

- **Đọc**: đưa dữ liệu từ một luồng vào một cấu trúc dữ liệu, chẳng hạn như một mảng byte.
- **Ghi**: đưa dữ liệu vào một luồng từ một nguồn dữ liệu.
- **Tìm kiếm**: truy vấn và sửa đổi vị trị hiện tại trong một luồng.

Dưới đây là một số các lớp liên quan đến stream thường được sử dụng:

- **FileStream** (System.IO.FileStream) – để đọc và ghi một file.
- **NetworkStream** (System.Net.Sockets.NetworkStream) – để đọc và ghi thông qua các socket mạng.

File và Thư mục

Chúng ta có thể sử dụng các loại dưới đây có trong namespace System.IO để tương tác với file và thư mục.

- ***System.IO.File*** – cung cấp các phương thức để tạo, sao chép, xóa, di chuyển, mở file và giúp khởi tạo một đối tượng FileStream.
- ***System.IO.FileInfo*** – cung cấp các phương thức instance để tạo, sao chép, xóa, di chuyển, mở file và giúp khởi tạo một đối tượng FileStream.
- ***System.IO.Directory*** – cung cấp các phương thức để tạo, di chuyển và liệt kê thư mục.
- ***System.IO.DirectoryInfo*** – cung cấp các phương thức để tạo, di chuyển và liệt kê thư mục và thư mục con.

2. Các lớp liên quan

a) FileStream:

Là một lớp dẫn xuất từ Stream, sử dụng để đọc dữ liệu từ một file hay ghi dữ liệu vào 1 file.

Ví dụ:

```
FileStream fs = new FileStream(FileName, mode);
```

Trong đó:

- FileName: tập tin mà chúng ta muốn truy xuất đến.
- Mode: chế độ thao tác file như thế nào (Append, Create, CreateNew, Open, OpenOrCreate…).

Ví dụ:

```
FileStream fs = new FileStream("thuchanh.txt", FileMode.CreateNew);
```

b) StreamReader và StreamWritter

Là các lớp dẫn xuất từ Stream, là luồng đọc và ghi file. Để đọc file ta dùng lớp StreamReader. Để ghi file ta dùng lớp StreamWriter.

```
StreamReader sr = new StreamReader(FileStream fileName);
StreamWriter sw = new StreamReader(FileStream fileName);
```

Ví dụ:

```
StreamReader sr = new StreamReader(fs);
```

c) BinaryStream:

Khi thao tác với tập tin văn bản, để lưu dữ liệu kiểu số thì phải thực hiện việc chuyển đổi sang dạng chuỗi ký tự để lưu. Dữ liệu khi lấy ra sẽ được biểu diễn dưới dạng chuỗi ký tự do đó cần phải chuyển sang dạng số. Vì vậy, cần có cách thức khác tốt hơn để lưu vào và lấy ra trực tiếp giá trị trong tập tin. Có thể sử dụng luồng nhị phân **BinaryStream** để thao tác với tập tin, sau đó đọc và ghi thông tin nhị phân từ luồng này.

Ghi chú: Thông tin nhị phân là thông tin đã được định dạng kiểu lưu trữ dữ liệu.

Ví dụ:

```
FileStream fs = new FileStream(sfd.FileName,FileMode.CreateNew);
BinaryWriter bw = new BinaryWriter(fs);
```

d) BinaryFormatter:

Sử dụng 2 phương thức Serialize và Deserialize để chuyển đổi tượng thành định dạng phù hợp cho quá trình lưu trữ file:

- ***Serialize***: chuyển đổi đối tượng sang định dạng phù hợp để ghi vào File, đảm bảo không làm mất dữ liệu.
- ***Deserialize***: đọc dữ liệu đã định dạng từ một File và chuyển nó về dạng ban đầu

Ví dụ:

Serialize

```
BinaryFormatter binaryFormatter = new BinaryFormatter();
FileStream fileName = File.Create("..\\student.txt");
binaryFormatter.Serialize(fileName, st);
```

Deserialize

```
BinaryFormatter bf = new BinaryFormatter();
FileStream fs = File.OpenRead("..\\student.txt");
Student student = (Student)bf.Deserialize(fs);
```

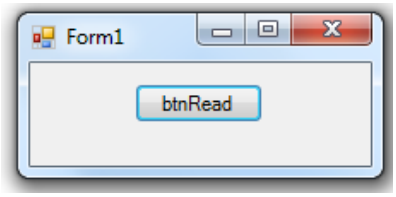
*** BinaryFormatter không an toàn và không được khuyến khích sử dụng để xử lý dữ liệu. Một số phiên bản .NET hoặc .NET Framework đã không hỗ trợ BinaryFormatter, sinh viên có thể chuyển sang sử dụng định dạng dữ liệu dạng JSON.**

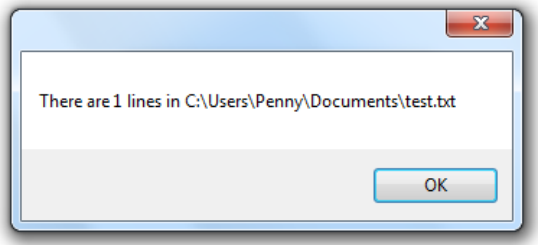
C. VÍ DỤ MINH HỌA

1. Chương trình đếm số dòng văn bản từ file

Khi nhấn vào nút **btnRead**, thực hiện đếm và thông báo số dòng có trong tập tin.

Giao diện minh họa.

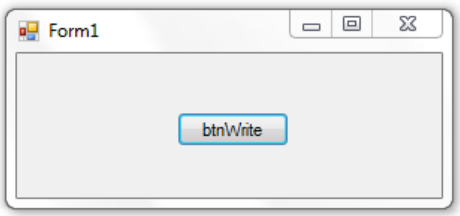




Gợi ý: bắt sự kiện cho nút btnRead, sử dụng lớp StreamReader.

```
private void btnRead_Click(object sender, System.EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.ShowDialog();
    FileStream fs = new FileStream(ofd.FileName,
FileMode.OpenOrCreate);
    StreamReader sr = new StreamReader(fs);
    int lineCount = 0;
    while (sr.ReadLine() != null)
    {
        lineCount++;
    }
    fs.Close();
    MessageBox.Show("There are " + lineCount + " lines in " +
ofd.FileName);
}
```

2. Chương trình ghi file nhị phân



Chương trình ghi thành file nhị phân với nội dung bất kỳ.

Gợi ý: bắt sự kiện cho nút btnWrite, sử dụng BinaryWriter

```
private void btnWrite_Click(object sender, EventArgs e)
{
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.ShowDialog();
    FileStream fs = new FileStream(sfd.FileName,  FileMode.CreateNew);
    BinaryWriter bw = new BinaryWriter(fs);
    int[] myArray = new int[1000];
    for (int i = 0; i < 1000; i++)
    {
        myArray[i] = i;
        bw.Write(myArray[i]);
    }
    bw.Close();
}
```

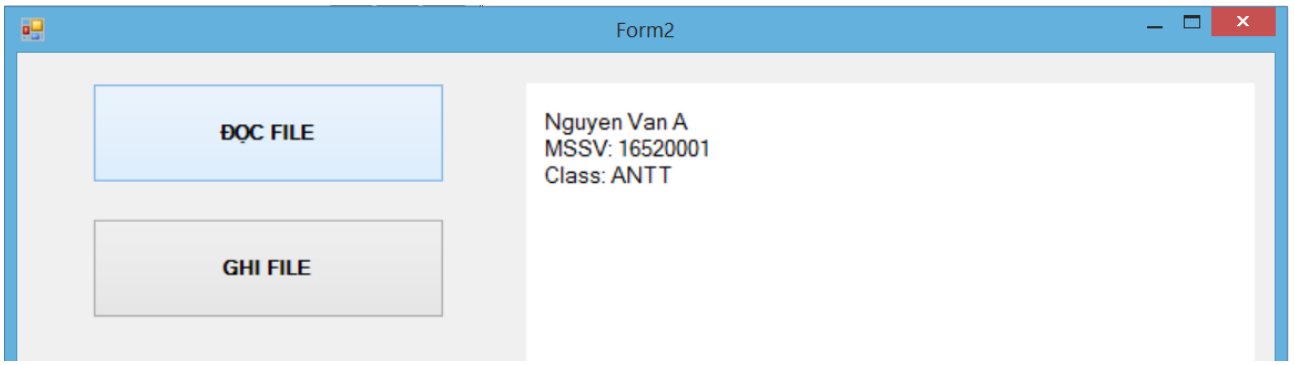
D. BÀI TẬP

Các bài thực hành yêu cầu viết chương trình dưới dạng *Windows Forms App.*
Sinh viên có thể tùy biến cách sắp xếp giao diện hợp lý.
Tất cả các bài thực hành đặt chung trong 1 Project duy nhất.

1. Ghi và Đọc file

Viết chương trình đọc nội dung từ file "input1.txt" và xuất ra màn hình. Sau đó chuyển toàn bộ nội dung sang kiểu in hoa và ghi xuống file "output1.txt".

Giao diện minh họa:



Gợi ý:

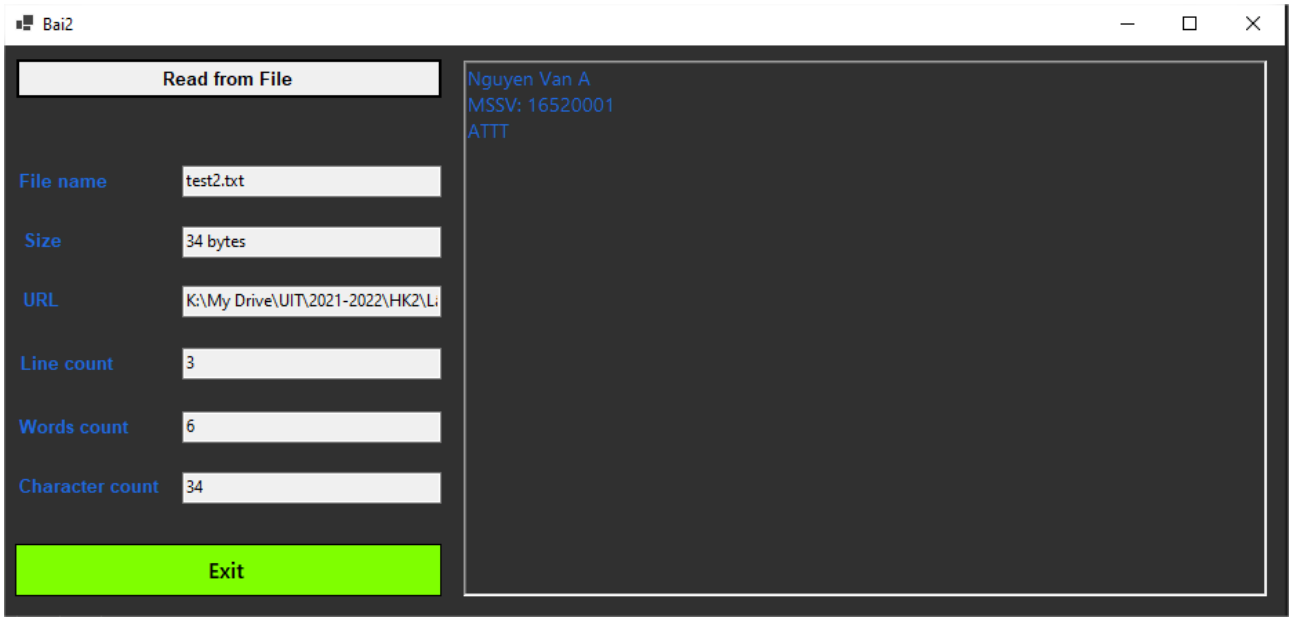
- Sử dụng hàm ReadToEnd để đọc toàn bộ dữ liệu và đưa dữ liệu vào RichTextBox.
- Sử dụng StreamWritter để ghi nội dung xuống file.

2. Đọc thông tin file

Viết chương trình đọc file và hiển thị các thông tin sau:

- Tên file
- Kích thước file
- Đường dẫn đến file
- Số dòng, số từ, số ký tự
- Nội dung của file

Giao diện minh họa.



Gợi ý: *bắt sự kiện click cho nút ĐỌC FILE, sử dụng lớp StreamReader*

- Có thể sử dụng thuộc tính .SafeFileName của OpenFileDialog để lấy tên file
  VD: name = ofd.SafeFileName.ToString();
- Có thể sử dụng thuộc tính .Name của FileStream để lấy đường dẫn
  VD: path = fs.Name.ToString();

3. Đọc, ghi file và tính toán

Đọc nội dung từ file "input3.txt" với nội dung theo định dạng, sau đó thực hiện các phép tính và ghi kết quả xuống file "output3.txt". Các phép tính bao gồm: cộng, trừ, nhân, chia, và hỗ trợ gom nhóm biểu thức bằng ngoặc đơn.

Lưu ý: Không sử dụng phương thức DataTable.Compute.

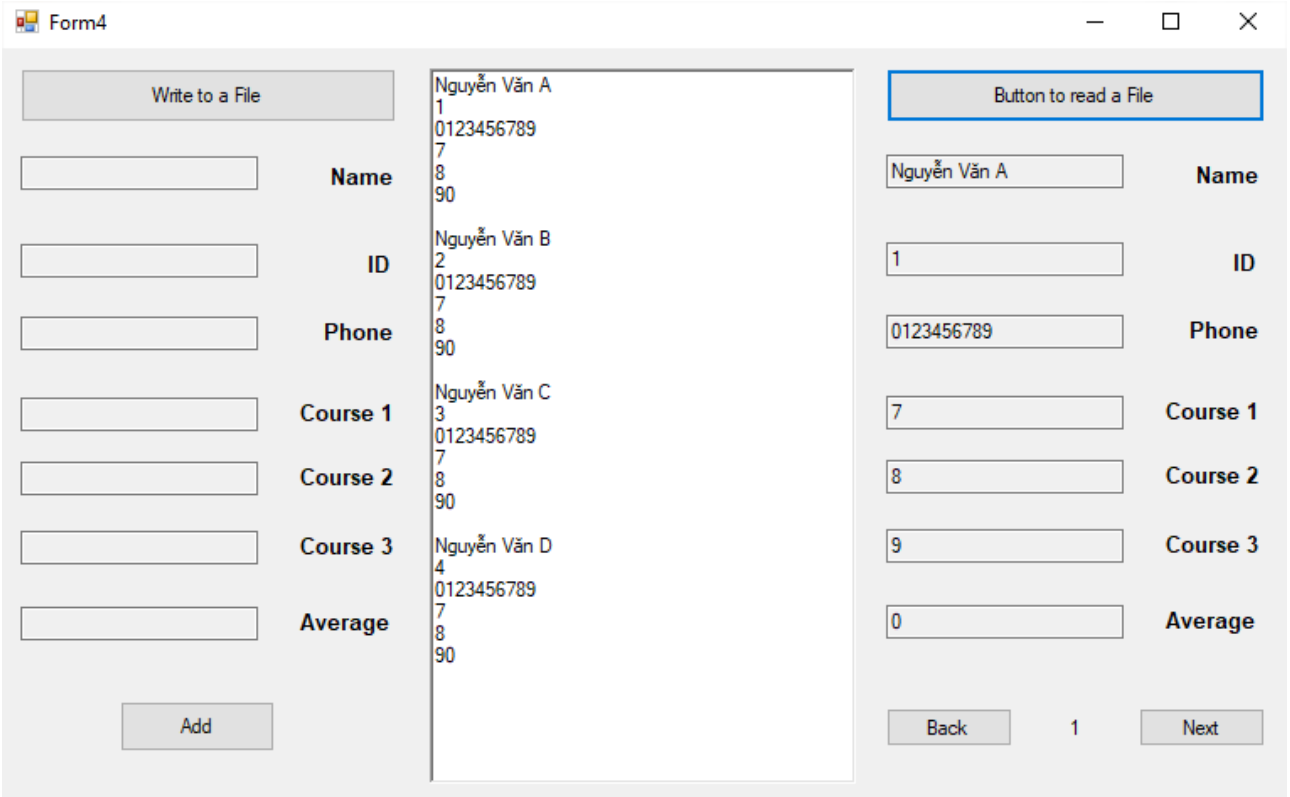
Ví dụ : Nội dung file "input3.txt" :	Nội dung file "output3.txt" :
1 + 2 + 3 + 4	1 + 2 + 3 + 4 = 10
12 – 7 – 5 + 2 – 3	12 – 7 – 5 + 2 – 3 = –1
2024 – 1 – 2 + 3	2024 – 1 – 2 + 3 = 2024
222 + 333 – 444.2 + 1	222 + 333 – 444.2 + 1 = 111.8

4. Đọc và ghi file sử dụng BinaryFormatter

Viết chương trình sử dụng BinaryFormatter cho phép : Nhập 1 mảng các sinh viên (không nhập điểm trung bình) và ghi xuống file "input4.txt". Cấu trúc của Sinh viên như sau :

- Họ và tên : String - MSSV : Int - Điện thoại : String - Điểm môn 1 : Float	- Điểm môn 2 : Float - Điểm môn 3 : Float - Điểm trung bình : Float

Đọc thông tin mảng Học Viên từ file "input4.txt" và tính điểm trung bình cho từng sinh viên sau đó ghi xuống file "output4.txt" và xuất ra màn hình.

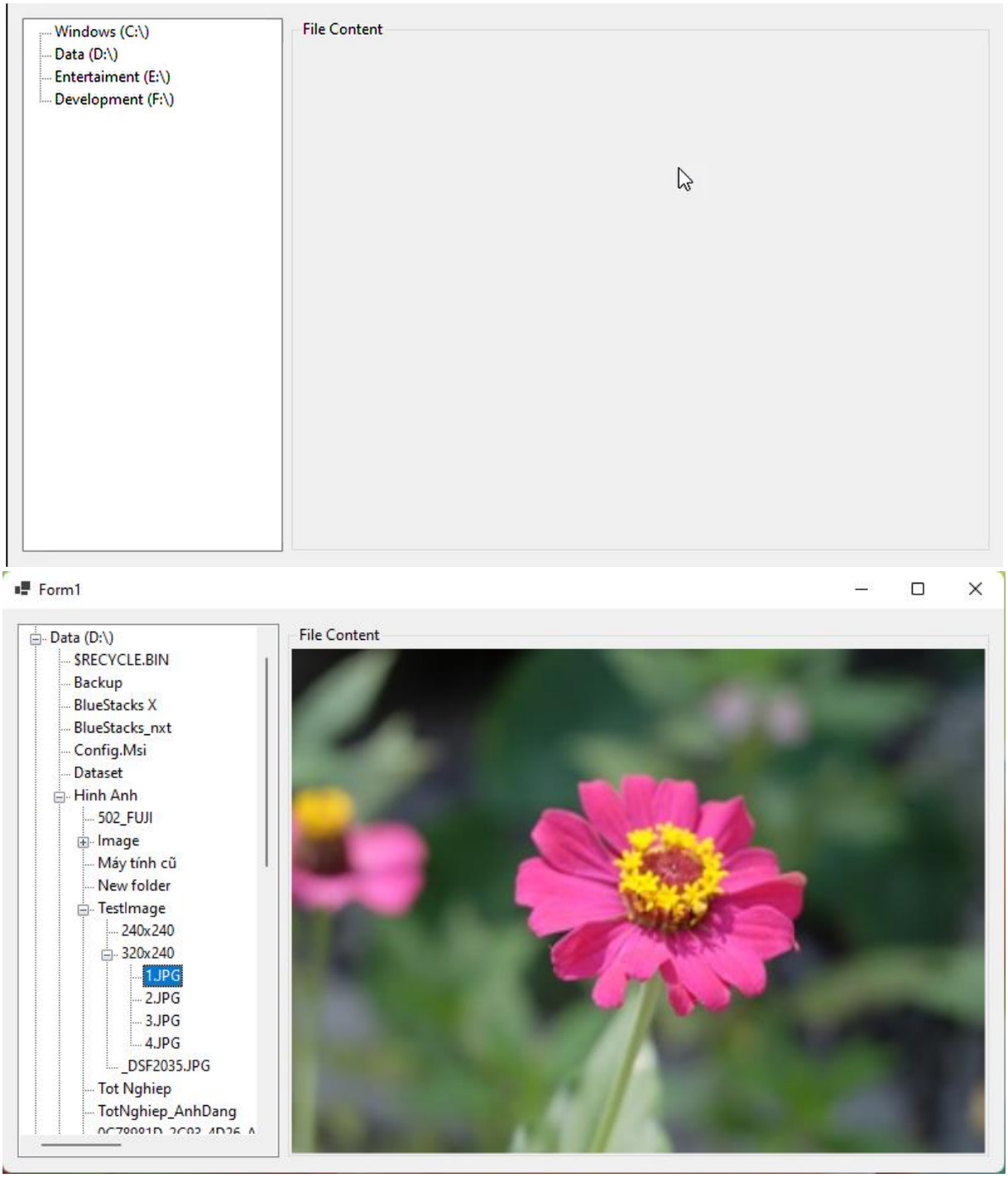


Lưu ý: Khi xuất ra màn hình, cần tính điểm trung bình của 3 môn học. Ràng buộc điều kiện số điện thoại phải có 10 chữ số và bắt đầu bởi số 0. Mã số sinh viên là một số có 8 chữ số. Các điểm của từng học phần từ 0 đến 10. Có chức năng hiển thị số trang hiện tại và có thể điều chỉnh để xem lần lượt danh sách sinh viên.

5. Bài 05 – Duyệt thư mục

Viết ứng dụng cho phép duyệt tất cả file có trong máy tính, hiển thị danh sách các file, folder. Cho phép đi đến folder tiếp theo khi nhấp nhấn đúp chuột và hiển thị nội dung của file khi bấm chọn (áp dụng với file văn bản, hình ảnh).

Giao diện minh họa



E. YÊU CẦU & NỘP BÀI

1. Yêu cầu

- Sinh viên có thể tuỳ biến các giao diện được gợi ý tại các yêu cầu. Tuy nhiên, cần đảm bảo bố cục hài hoà, thuận tiện cho thao tác của người sử dụng, thể hiện được các chức năng và thông tin cần thiết của yêu cầu.
- Có kiểm tra các điều kiện ràng buộc đối với dữ liệu nhập vào.
- Đặt tên biến rõ ràng, cấu trúc code hợp lý, thuận tiện trong quá trình kiểm tra, xem lại.
- Sinh viên viết báo cáo trình bày ý tưởng thuật toán, chức năng các đoạn code chính giải quyết yêu cầu bài toán, ảnh chụp màn hình minh hoạ quá trình hoạt động của chương trình.
- Các bài nộp sao chép của nhau sẽ không được chấm điểm.

2. Nộp bài

- Sinh viên thực hiện theo nhóm tối đa 03 thành viên. Tổng hợp bài làm và 01 sinh viên đại diện nộp bài tại website môn học theo thời gian quy định.
- Sinh viên đặt báo cáo (định dạng .docx), mã nguồn của chương trình, chương trình sau khi biên dịch và testcases vào thư mục như sau:

Đặt các file vào 1 thư mục và nén lại (định dạng .zip) với tên theo quy tắc sau:

Mã lớp-LabX-MSSV1_MSSV2

Ví dụ: *NT106.O21.1-Lab01_25520001_25520002*

F. THAM KHẢO

[1] Microsoft (2018). C# Guide. [Online] Available at: https://docs.microsoft.com/en-us/dotnet/csharp/

[2] Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.