2

Lab

File và I/O Stream trong C#

File and Stream I/O in C#

Thực hành Lập trình mạng căn bản - NT106

Giảng viên biên soạn: ThS. Đỗ Thị Hương Lan

Tháng 03/2023

Lưu hành nội bộ

A. TỔNG QUAN

1. Mục tiêu

- Cung cấp khả năng làm việc với File
  - **Khởi tạo**
  - **Đọc**
  - **Viết/Ghi**
  - **Cập nhật**
- Hiểu được luồng thông tin (Stream) trong C#.
- Có thể sử dụng được các lớp FileStream, lớp StreamReader, lớp StreamWriter v.v … để đọc và viết các đối tượng vào trong các File.

2. Môi trường

- IDE Microsoft Visual Studio 2010 – 2017

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

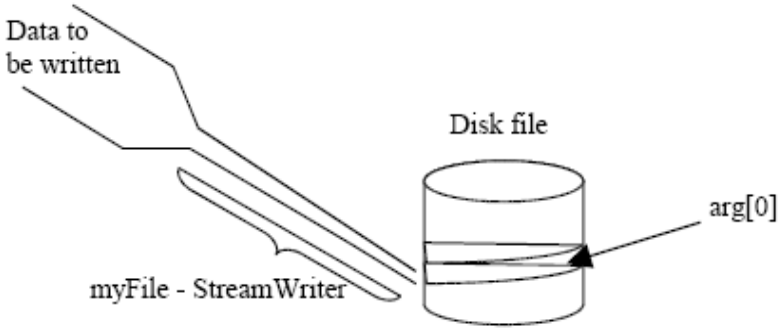
B. KIẾN THỨC NỀN TẢNG

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

Dữ liệu là một thành phần quan trọng và hiện diện ở hầu hết các ứng dụng trong các nền tảng công nghệ thông tin. Dữ liệu có thể được lưu trữ ở các **file** (tập tin), **Cơ sở dữ liệu** (CSDL) hoặc được gửi thông qua các **luồng** đến từ Mạng.

Luồng (Stream) là chuỗi các byte đi qua đường truyền, chứa thông tin sẽ được chuyển qua, còn tập tin (File) lưu trữ thông tin, dữ liệu. Khi một tập tin được mở lên để đọc hoặc ghi, nó sẽ trở thành một Stream. Có hai luồng chính là: Luồng vào (Input stream) và Luồng ra (Output Stream).

File và Stream I/O (Input/Output) dùng để đề cập đến việc truyền dữ liệu đến hoặc từ một phương tiện lưu trữ.



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

Dữ liệu được truyền theo hai hướng

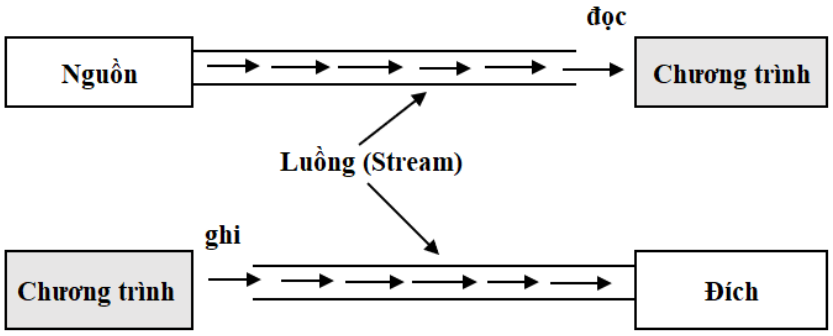
- Đọc dữ liệu: đọc dữ liệu từ bên ngoài vào chương trình.
- Ghi dữ liệu: đưa dữ liệu từ chương trình ra bên ngoài.

Streams

Lớp Stream hỗ trợ đọc và ghi byte. Tất cả các lớp đại diện cho các luồng đều kế thừa từ lớp Stream. Lớp Stream và các lớp dẫn xuất của nó 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 bao hàm ba thao tác cơ bản:

- **Đọc**: đưa dữ liệu từ một luồng vào một cấu trúc dữ liệu, ví dụ 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



Hình 1: Luồng dữ liệu (Stream)

Dưới đây là một số các lớp 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

Trong .NET Framework, namespace System.IO bao gồm các lớp khác nhau có khả năng đọc và ghi (đồng bộ và không đồng bộ) đối với các luồng dữ liệu và file. Những namespace này cũng chứa các loại namespace thực hiện việc nén và giải nén các file.

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

Loại Namespace	Mô tả
System.IO.File	Cung cấp các phương thức tĩnh cho việc tạo, sao chép, xóa, di chuyển và mở file, cũng như giúp khởi tạo một đối tượng FileStream
System.IO.FileInfo	Cung cấp các phương thức instance để tạo, sao chép, xóa, di chuyển và mở file cũng như giúp khởi tạo một đối tượng FileStream.
System.IO.Directory	Cung cấp các phương thức tĩnh để tạo, di chuyển và liệt kê thư mục và thư mục con.
System.IO.DirectoryInfo	Cung cấp các phương thức instance để tạo, di chuyển và liệt kê thư mục và thư mục con.
System.IO.Directory	Cung cấp các phương thức và thuộc tính để xử lý chuỗi thư mục theo cách đa nền tảng.

2. Các Class C# hỗ trợ liên quan

a) Lớp FileStream:

FileStream là một lớp dẫn xuất từ Lớp cơ sở Stream, được sử dụng đọc, viết dữ liệu và đóng file. FileStream cung cấp Stream cho File, hỗ trợ cả thao tác đọc và ghi đồng bộ và không đồng bộ.

Để sử dụng lớp FileStream, cần khai báo namespace System.IO.

Ví dụ: `using System.IO`

Để tạo ra một FileStream, cần cung cấp các thông tin sau:

- FileMode
- FileName
- FileAccess
- FileShare

Thuộc tính	Mô tả
FileMode	Được định nghĩa ở dạng **Enum**, gồm các chế độ khác nhau hỗ trợ cho việc mở File. **Append**: mở File hiện có, di chuyển con trỏ đến cuối File để viết tiếp nội dung vào hoặc nếu File không tồn tại thì tạo mới **Create**: tạo một File mới, nếu File đã tồn tại nó sẽ ghi đè lên **CreateNew**: tạo một File mới, nếu File đã tồn tại thì trả ra một lỗi ngoại lệ (Exception) **Open**: mở ra File hiện có **OpenOrCreate**: mở ra File hiện có, nếu File chưa tồn tại, nó sẽ tạo mới **Truncate**: mở File hiện có và cắt kích thước File trở về 0 byte
FileName	Tên của tập tin muốn truy xuất
FileAccess	Được định nghĩa ở dạng **Enum**, bao gồm các chế độ hỗ trợ cho việc truy xuất File. **Read**: cho phép đọc File **Write**: cho phép ghi File **ReadWrite**: cho phép đọc và ghi File
FileShare	Được định nghĩa ở dạng **Enum**, bao gồm các chế độ khác nhau để hỗ trợ cho việc chia sẻ File. **Inheritable**: cho phép một File truyền tính kế thừa đến các tiến trình con **None**: không cho phép chia sẻ File hiện tại **Read**: cho phép mở File để đọc **ReadWrite**: cho phép mở File để đọc, ghi **Write**: cho phép mở File để ghi

Bảng B: Danh sách các thuộc tính của FileStream

Ví dụ:

```
FileStream fs = new FileStream("demo.txt", FileMode.Open);
FileStream fs = new FileStream("a.txt", FileMode.CreateNew);
```

b) *StreamReader và StreamWritter*

StreamReader và **StreamWriter** là các lớp dẫn xuất từ Stream, được sử dụng để đọc và ghi dữ liệu. Trong đó, StreamReader đại diện cho luồng đọc văn bản, có thể đọc một chuỗi các ký tự. Ngược lại, StreamWriter đại diện cho luồng ghi văn bản, có thể ghi một chuỗi các ký tự. Các lớp này có thể áp dụng với nhiều loại Stream khác nhau như: File, Network, v.v…

Lớp StreamReader kế thừa từ lớp trừu tượng là TextReader.

Lớp StreamWriter kế thừa từ lớp trừu tượng là TextWriter.

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

Ví dụ:

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

c) *BinaryStream:*

Nếu chúng ta sử dụng một tập tin văn bản, thì khi chúng ta lưu dữ liệu kiểu số thì phải thực hiện việc chuyển đổi sang dạng chuỗi ký tự để lưu vào trong tập tin văn bản và khi lấy ra ta cũng lấy được giá trị chuỗi ký tự do đó ta phải chuyển sang dạng số. Đôi khi chúng ta muốn có cách thức nào đó tốt hơn để lưu trực tiếp giá trị vào trong tập tin và sau đó đọc trực tiếp giá trị ra từ tập tin.

Ví dụ: khi viết một số lượng lớn các số integer vào trong tập tin như là những số nguyên, thì khi đó ta có thể đọc các giá trị này ra như là số integer. Trường hợp nếu chúng được viết vào tập tin với dạng văn bản, thì khi đọc ra ta phải đọc ra văn bản và phải chuyển mỗi giá trị từ một chuỗi đến các số integer. Tốt hơn việc phải thực hiện thêm các bước chuyển đổi, ta có thể gắn một kiểu luồng nhị phân **BinaryStream** vào trong một tập tin, rồi 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 viết và đọc đối tượng từ trong luồng:

- ***Serialize***: chuyển đổi một đối tượng sang một định dạng, và có thể được viết vào File mà không 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

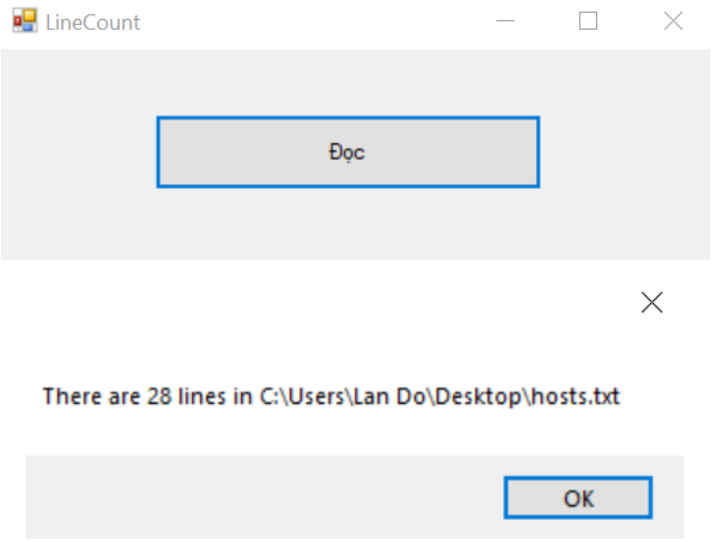
```
BinaryFormatter bf = new BinaryFormatter();
FileStream fs = File.OpenRead("..\\student.txt");
Student student = (Student)bf.Deserialize(fs);
```

C. VÍ DỤ MINH HỌA

1. Chương trình đếm số dòng trong file

Yêu cầu: Viết chương trình cho phép chọn một tập tin và hiển thị thông tin số dòng nội dung của tập tin đó.

Giao diện minh họa



Hình 3: Giao diện chương trình Đếm số dòng

Gợi ý:

- Sử dụng dụng vòng lặp kết hợp cùng phương thức ReadLine() của StreamReader để đếm số dòng.

```
string url = @"C:\Users\Lan Do\Desktop\hosts.txt";
StreamReader sr = new StreamReader(url);
while (sr.ReadLine() != null) { … }
```

- Sử dụng **MessageBox** để hiển thị thông báo số dòng.

```
MessageBox.Show("There are " + lCount +" lines in " +url+"");
```

D. BÀI TẬP

Các bài thực hành dưới đây yêu cầu viết chương trình dưới dạng *Windows Forms App*.

Sinh viên có thể tùy biến cách sắp xếp giao diện khác sao cho hợp lý.

Khuyến nghị: Tất cả các bài thực hành đặt chung trong 1 Project duy nhất.

1. Bài 01 – Đọc và ghi file cơ bản

Yêu cầu:

Viết chương trình cho phép đọc và ghi file cơ bản (file .txt)

Yêu cầu chi tiết

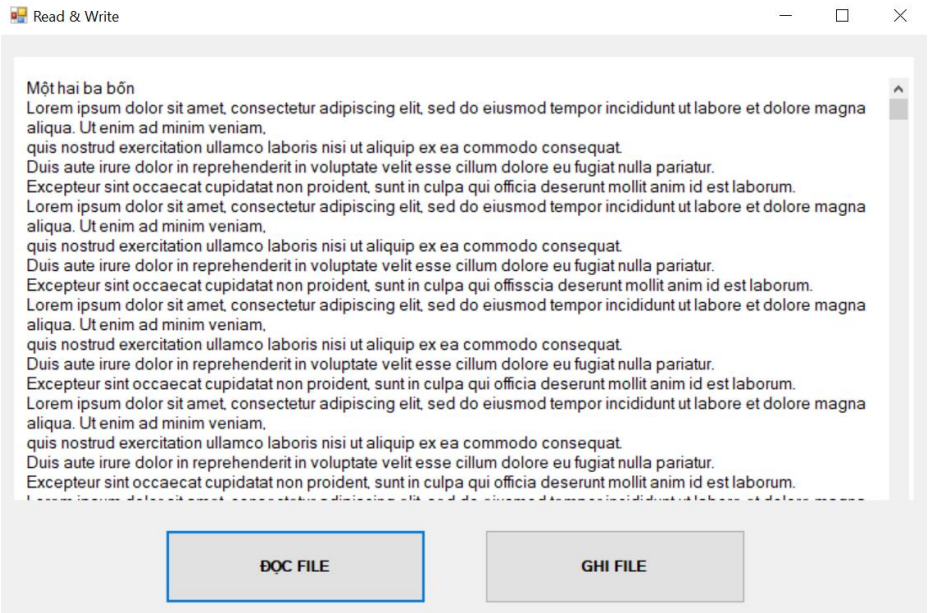
Tính năng Đọc file

- Hiển thị hộp thoại cho phép chọn file cần đọc
- Đọc và hiển thị nội dung vào vùng hiển thị

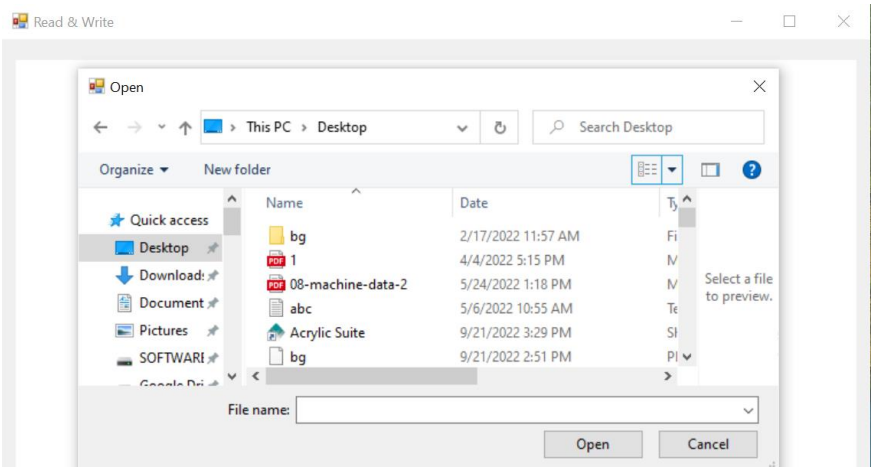
Tính năng Ghi file

- Hiển thị hộp thoại cho phép chọn vị trí lưu file
- Chuyển toàn bộ ký tự sang kiểu in hoa
- Cho phép ghi đè hoặc tạo mới file

Giao diện minh họa:



Hình 4: Giao diện chương trình Đọc và ghi file cơ bản



Gợi ý:

Tạo các phương thức liên quan đến sự kiện Click của các nút bấm "Đọc file", "Ghi file".

Đọc file

- Sử dụng Control OpenFileDialog để cho phép hiển thị Hộp thoại chọn file.

```
OpenFileDialog ofd = new OpenFileDialog();
ofd.ShowDialog();
```

- Sử dụng thuộc tính Filename của OpenFileDialog để có được đường dẫn của File cần chọn.
- Sử dụng FileStream hoặc StreamReader để đọc nội dung.

Ví dụ: sử dụng StreamReader

```
StreamReader sr = new StreamReader(ofd.FileName);
string content = sr.ReadToEnd();
```

Ghi file

- Sử dụng Control SaveFileDialog để cho phép hiển thị Hộp thoại chọn vị trí lưu trữ

```
SaveFileDialog sfd = new SaveFileDialog();
sfd.Filter = "Text (*.txt)|*.txt";
sfd.ShowDialog();
```

- Tùy chỉnh thuộc tính Filter để chọn loại file lưu trữ.
- Sử dụng FileStream hoặc StreamWriter để ghi nội dung.

```
FileStream fs = new FileStream(sfd.FileName, FileMode.OpenOrCreate);
byte[] ct = Encoding.UTF8.GetBytes(rtb_info.Text.Trim());
fs.Write(ct, 0, ct.Length);
```

2. Đọc thông tin tập tin (File .txt)

Yêu cầu:
Viết chương trình cho phép đọc thông tin một file .txt.

Yêu cầu chi tiết

Viết chương trình đọc file .txt

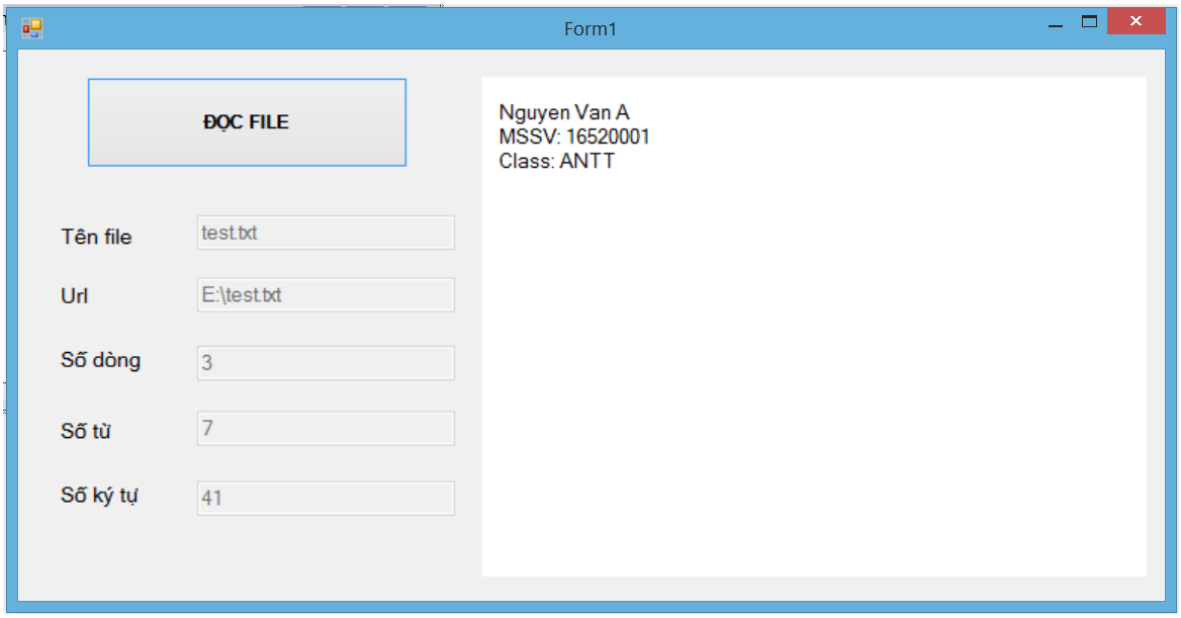
Tính năng Đọc file

- Hiển thị hộp thoại cho phép chọn file cần đọc
- Đọc và hiển thị nội dung vào vùng hiển thị

Tính năng Hiển thị các thông tin sau

- Tên file
- Đường dẫn Url
- Số dòng, số từ, số ký tự

Giao diện minh họa



Gợi ý:

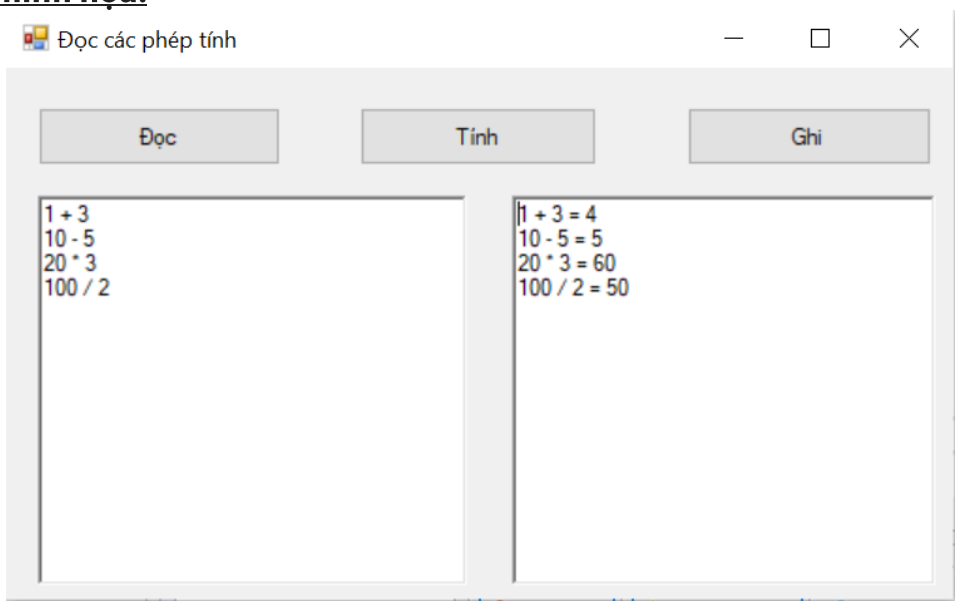
- Tính năng Đọc file: Tham khảo hướng dẫn của bài tập **Đọc và ghi file cơ bản**
- Hiển thị các thông tin thêm:
  - **Tên tập tin**: Sử dụng thuộc tính SafeFileName của OpenFileDialog
  - **Url**: sử dụng thuộc tính Name của FileStream hoặc Filename của OpenFileDialog
  - **Số ký tự**: sử dụng thuộc tính Length của nội dung đã đọc được
  - **Số dòng**: tham khảo hướng dẫn của ví dụ Đếm số dòng
  - **Số từ**: sử dụng phương thức Split(), với ký tự để tách chuỗi có thể là khoảng trắng, ký hiệu xuống hàng.

3. Bài 3 - Đọc và Ghi file (Mở rộng)

Yêu cầu:
Viết chương trình cho phép đọc các phép tính toán từ một file, thực hiện tính toán và ghi thành file mới

Ví dụ : Nội dung file "input.txt":	**Nội dung file "output.txt":**
1 +2	1 +2 = 3
12 –7	12 –7 = 5
10 *20	10 *20 = 200
200 /10	200 /10 = 200

Giao diện minh họa:



Hình 5: Giao diện chương trình Đọc và ghi file các phép tính

Gợi ý:

- Tham khảo cách tách chuỗi tại link sau:

https://learn.microsoft.com/en-us/dotnet/api/system.string.split?view=net-7.0&viewFallbackFrom=netframework-4.8%20

4. Bài 4 – Làm việc với File/CSDL

Yêu cầu:
Viết chương trình quản lý sinh viên có làm việc với file và CSDL

Yêu cầu chi tiết

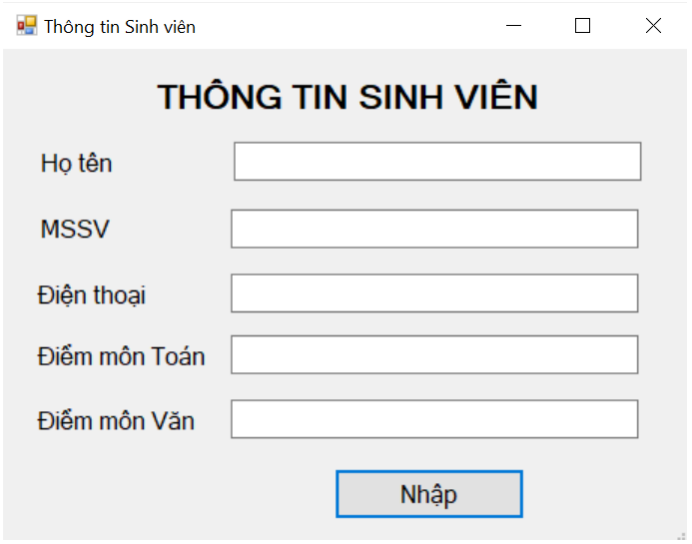
Viết chương trình cho phép nhập dữ liệu, lưu file, xuất file và hiển thị thông tin.

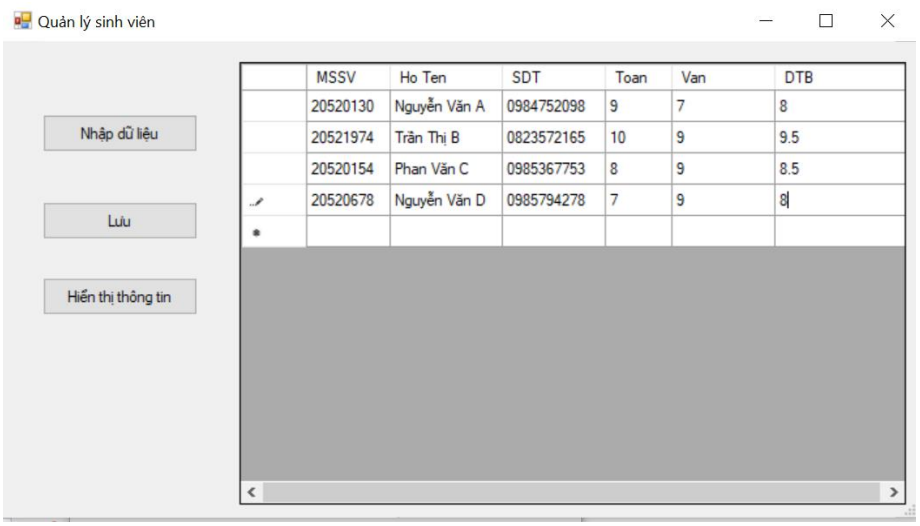
- **Nhập dữ liệu**: nhập từng thông tin SV, ghi xuống file "input.txt". Từng SV được lưu trong file dưới định dạng:

  MSSV1;HoTen;DienThoai;DiemToan;DiemVan
  MSSV2;HoTen;DienThoai;DiemToan;DiemVan …

- **Lưu file Excel**: đọc thông tin toàn bộ SV từ file "input.txt", tính ĐTB cho từng học viên, sau đó ghi xuống file Excel.
- **Hiển thị thông tin**: Đọc dữ liệu từ File Excel, hiển thị màn hình Danh sách sinh viên (bao gồm ĐTB đã được tính).

Giao diện minh họa:





5. Bài 5 – Duyệt thư mục

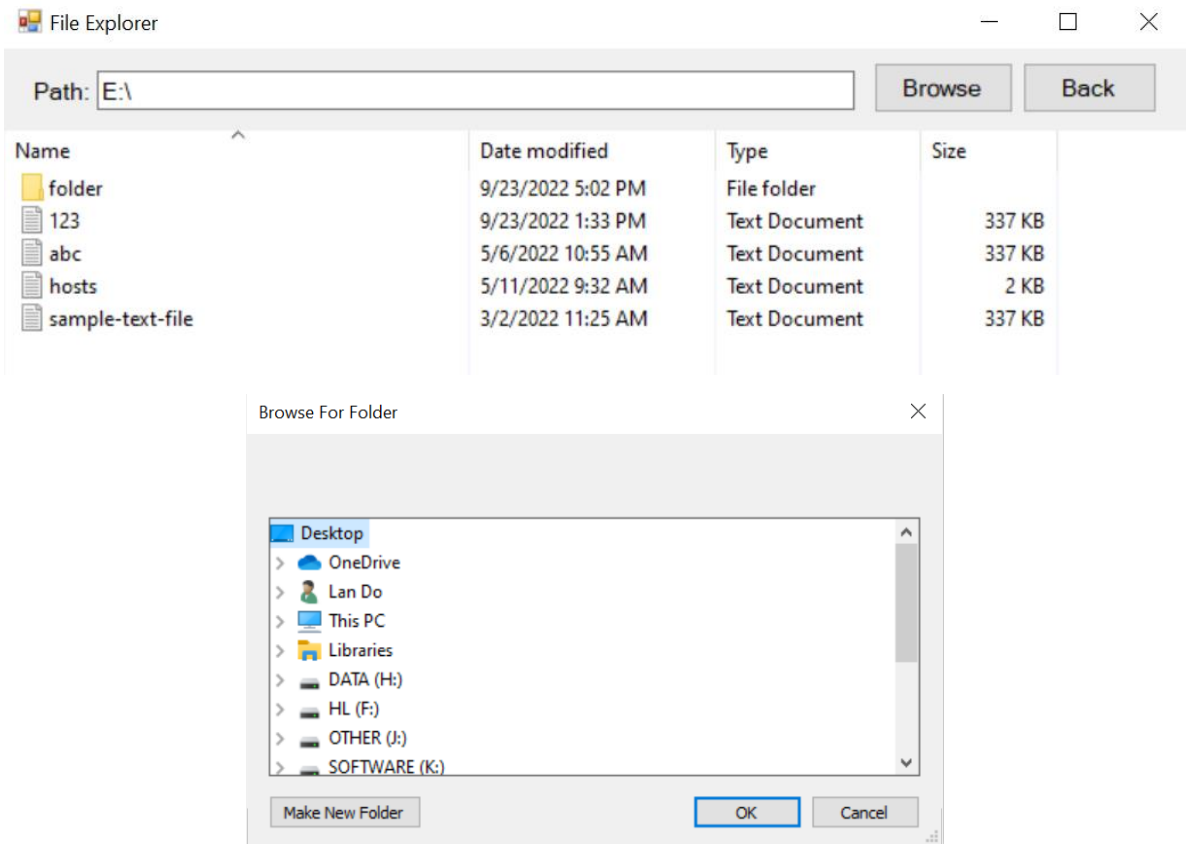
Yêu cầu:
Viết chương trình Duyệt thư mục đơn giản, cho phép duyệt tất cả các thư mục con và file có trong một thư mục.

Yêu cầu chi tiết:

Viết ứng dụng cho phép duyệt tất cả file có trong một thư mục, hiển thị danh sách các file gồm các thông tin sau:

- Tên file
- Kích thước
- Ngày tạo
- Đuôi mở rộng
- Các thông tin thêm v.v…

Giao diện minh họa:



Gợi ý:

- Sử dụng DirectoryInfo để lấy thông tin toàn bộ các file có trong 1 thư mục

```
DirectoryInfo di = new DirectoryInfo("E:\\TEST\\");
FileInfo[] fiArr = di.GetFiles();
```

- Sử dụng ListView để hiển thị danh sách file

E. YÊU CẦU & ĐÁNH GIÁ

1. Yêu cầu

- Sinh viên thực hành và nộp bài **cá nhân** theo thời gian quy định.
- Bài nộp: **Source code** (nén) và 1 **File .pdf** báo cáo tóm lược các bài tập liên quan kèm hình ảnh minh họa

Toàn bộ project đặt vào 1 file nén (.rar/.zip)	**File .PDF screenshot**
LabX-MSSV-HọTênSV	**LabX-MSSV-HọTênSV.pdf**
Ví dụ: *Lab1-16520901-NguyenVanA*	Ví dụ: *Lab1-16520901-NguyenVanA.pdf*

2. Đánh giá kết quả

- Tiêu chí đánh giá:
  - Chương trình chạy được, hoàn thành các yêu cầu cơ bản: **+70%.**
  - Có kiểm tra các điều kiện ràng buộc khi nhập dữ liệu, code "sạch" [2], đặt tên biến rõ ràng: **+30%.**
  - Nộp bài không đầy đủ; lỗi, không chạy được; nộp trễ; sao chép code bạn khác, nguồn có sẵn: *xử lý tùy theo mức độ* **(- (10 → 100)%)**

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.

HẾT