Лабораторная работа №6

Обработка исключений и работа с файлами в C#

Краткое описание: В данной лабораторной работе мы познакомимся с механизмами обработки исключений (try-catch-finally) и научимся работать с файлами и каталогами в языке C#. Мы создадим простое WPF-приложение "Ежедневник", которое позволит создавать, редактировать и удалять записи, сохраняя их в текстовых файлах. Также мы рассмотрим методы чтения и записи файлов, обработку возможных ошибок.

Состав работы: Данная лабораторная работа состоит из 5 заданий:

1) Создать WPF-приложение "Ежедневник", реализовать функционал добавления новой записи в ежедневник и отображение ее в списке.
2) Научиться сохранять данные в заранее определенный текстовый файл.
3) Реализовать загрузку ранее сохраненных записей из файла по нажатию кнопки и обрабатывать возможные исключения при работе с файлами.
4) Добавить возможность сохранения и загрузки в/из произвольного файла по нажатию кнопок.
5) Добавить автоматическую загрузку из последнего сохраненного файла при запуске приложения.

Задание 1.

Создать WPF-приложение "Ежедневник", реализовать функционал добавления новой записи в ежедневник и отображение ее в списке.

Для выполнения данного задания вам нужно сделать заготовку приложения: расположите в окне два “StackPanel” для списков дел на два дня. Под каждым из них расположите кнопку добавления новой стоки (“TextBox”) в список.

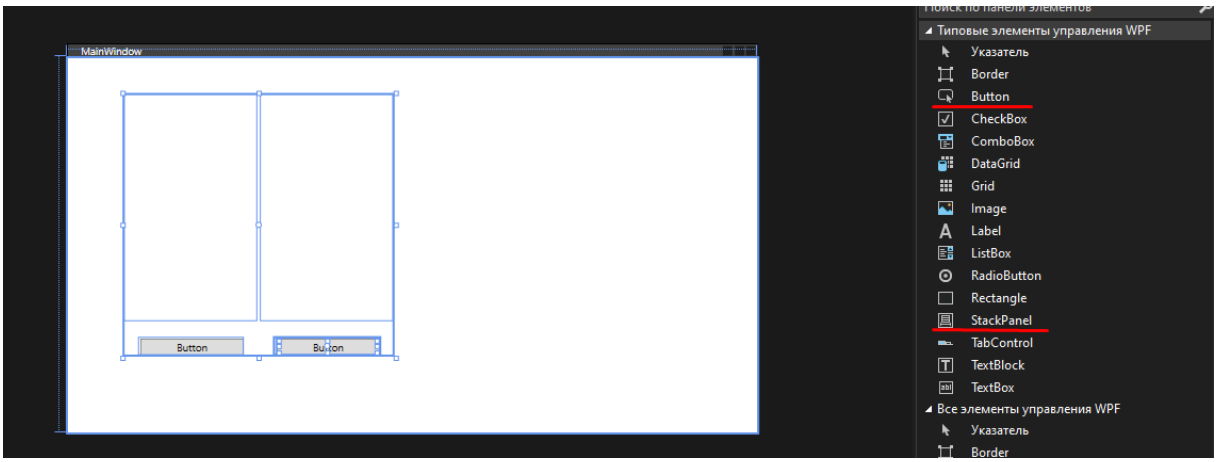

Рис. 1 – заготовка приложения.

Ранее вы работали с переменными и обращались к ним по имени:

```
int a = 0; // Создание переменной с именем "a", типом данных int и значением 0

a = a + 1; // Использование переменной: увеличение её значения на 1
```

Теперь, по условию задания вам требуется создавать некоторое количество (неизвестно какое) строк в “StackPanel”. Вы не можете присвоить каждому объекту “TextBox” имя, поскольку (как минимум) до нажатия кнопки, добавляющей строку, этой переменной ещё нет. Классическим решением такой проблемы является использование листов: вы можете хранить объекты “TextBox” в листе следующим образом:

1) сначала создайте лист, пригодный для хранения “TextBox”:

```
List<TextBox> ListOfTextBoxes = new List<TextBox>();
```

2) затем добавьте в лист новый объект “TextBox”:

```
ListOfTextBoxes.Add(new TextBox());
```

3) поскольку вы создаёте “TextBox” прямо в коде, в процессе выполнения программы, вам необходимо также куда-то «поместить» этот “TextBox”, чтобы он начал отображаться в программе. В обычной ситуации, когда у вас есть “StackPanel” с именем “Monday” и “TextBox” с именем “FirstBox”, вы могли бы добавить “TextBox” следующим образом:

```
Monday.Children.Add(FirstBox);
```

4) в данном случае, вместо имени “FirstBox” у объекта будет его номер в листе:

```
Monday.Children.Add(ListOfTextBoxes[0]);
string TextFromTextBox = ListOfTextBoxes[0].Text;
```

Используя вышеописанный подход, вы можете добавить обработчик нажатий на кнопку таким образом, чтобы каждый раз в соответствующем “StackPanel” создавался новый “TextBox”. Следите за тем, чтобы “ListOfTextBoxes” был глобальной переменной (т.е. создавался вне какой-либо функции).

Задание 2.

Научиться сохранять данные в заранее определенный текстовый файл.

Для выполнения данного задания вам требуется добавить кнопку «Сохранить в файл» через графический редактор. По нажатию кнопки создается текстовый файл, который хранит все строки обоих “StackPanel”.

В теории описан синтаксис для записи текста в файл. Единственное, на что здесь стоит обратить внимание – вместо “filename.txt” может быть указан полный путь к файлу, например: “C:\ProgramData\filename.txt”. Именно в таком формате хранится “FileName”, который возвращает OpenFileDialog.

Задание 3.

Реализовать загрузку ранее сохраненных записей из файла по нажатию кнопки и обрабатывать возможные исключения при работе с файлами.

Для выполнения данного задания вам требуется добавить кнопку «Загрузить из файла» через графический редактор. По нажатию кнопки в обоих “StackPanel” сначала удаляются все элементы, а затем создается нужное количество “TextBox” (точно такое же, которое было до нажатия кнопки «Сохранить в файл»). Каждый из созданных объектов “TextBox” заполняется соответствующим текстом из файла.

Используйте try-catch блок для того, чтобы обрабатывать случай, когда вы пытаетесь загрузить файл, которого не существует: в таком случае кнопка просто ничего не добавляет, а не завершает программу с ошибкой. В процессе работы с файлом вы можете столкнуться с двумя проблемами:

1) Непонятно, как разделяются строки в файле после сохранения в текстовую переменную.
2) Непонятно, как узнать какое количество строк принадлежит первому “StackPanel”, а какое – второму.

1: существуют специальные символы ‘\r’, ‘\n’ (один символ, для обозначения которого фактически используется два символа: \ и, например n). Первый символ обозначает возврат в начало строки, а второй обозначает переход на новую строку. WriteLine использует оба этих символа, чтобы обозначить переход на новую строку. В вашем случае, выполнение двух команд

```
writer.WriteLine("Строка 1");
writer.WriteLine("Строка 2");
```

фактически создаст одну строку: “Строка 1\r\nСтрока 2\r\n”. При этом, в блокноте или другом текстовом редакторе вы будете видеть две строки без специальных символов.

2: для того, чтобы загружать из одного файла сразу два “StackPanel” можно придумать несколько решений на этапе сохранения в файл: создать строку-разделитель (по аналогии с “\r\n”), которая обозначала бы переход к следующему списку; указывать в начале файла первыми двумя числами количество строк в каждом списке; сериализовать два листа, хранящих несколько “TextBox”, используя Json.

Задание 4.

Добавить возможность сохранения и загрузки в/из произвольного файла по нажатию кнопок.

Для выполнения данного задания, измените функции созданных ранее кнопок таким образом, чтобы при нажатии на них открывалось окно вида:

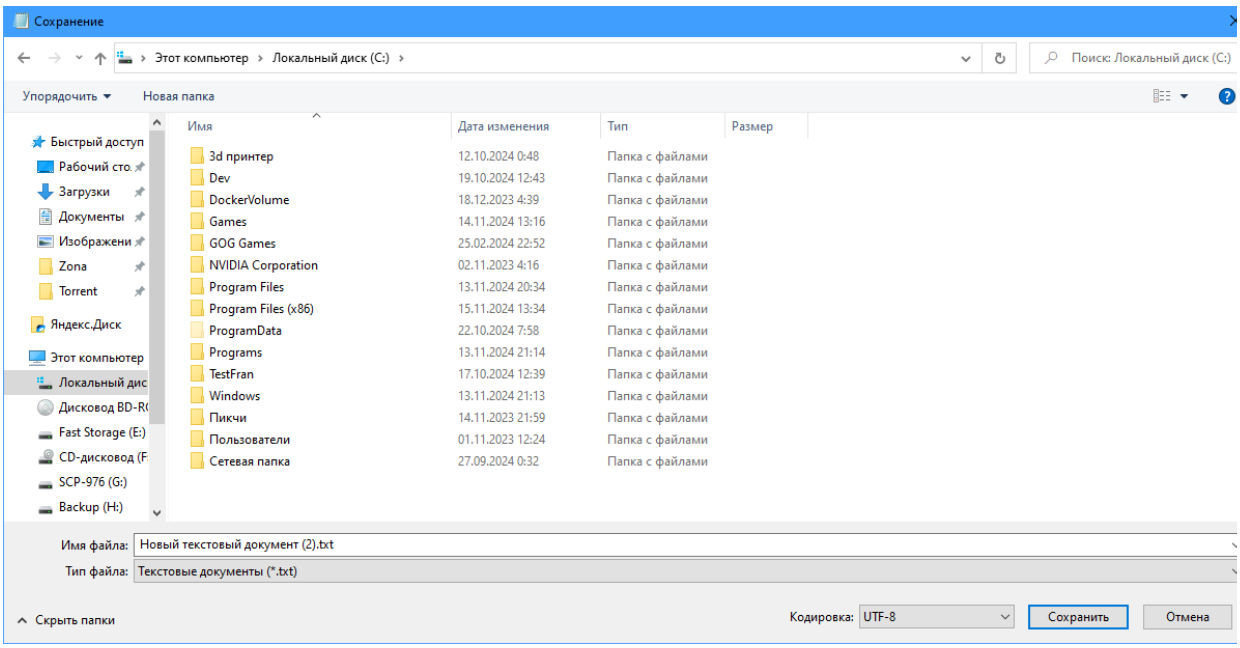

Рис. 2 – вид окна, создаваемого при вызове “SaveFileDialog”.

После выбора пользователем файла (сохраняемого или загружаемого) выполняется соответствующее действие.

Задание 5.

Добавить автоматическую загрузку из последнего сохраненного файла при запуске приложения.

Во время запуска приложения (до отображения графического интерфейса), пользователь никак не может указать файл, который будет использоваться при загрузке, а сохранить он мог его в любое место на компьютере. Следовательно, вам нужно каким-то образом сохранять путь к последнему сохраненному файлу. Для этого вам необходимо создать файл по известному вам (программе) пути. Этот файл должен хранить путь к последнему сохраненному файлу.

Теория.

Обработка исключений (try-catch-finally)

Обработка исключений позволяет перехватывать и обрабатывать ошибки, возникающие во время выполнения программы, предотвращая ее аварийное завершение.

1) try блок: содержит код, который может вызвать исключение (Например, если пользователь ввел дробное число)

2) catch блок: перехватывает исключение и позволяет обработать его.
3) finally блок (опционально): содержит код, который выполняется гарантированно (т.е. даже в случае, если в блоке catch был вызван return).

Пример:

```
Console.WriteLine("Введите целое число");
string input = Console.ReadLine();
try
{
    Convert.ToInt32(input);
    return;
}
catch (Exception ex)
{
    Console.WriteLine("Число может быть только целым!");
}
finally
{
    Console.WriteLine("Вы ввели {0}",input);
}
Console.WriteLine("Программа завершена");
```

В данном примере, если на вход программы подать целое число (допустим, 5) вывод будет следующий:

1) Введите целое число
2) Вы ввели 5

В случае, если подать дробное число (допустим, 0.5) вывод будет длиннее:

1) Введите целое число
2) Число может быть только целым!
3) Вы ввели 0.5
4) Программа завершена

Работа с файлами и каталогами

В C# для работы с файлами и каталогами используются пространства имен System.IO.

StreamWriter: используется для записи данных в файл.

StreamReader: используется для чтения данных из файла.

File: предоставляет статические методы для работы с файлами.

Directory: предоставляет статические методы для работы с каталогами.

Пример записи в файл:

```
using (StreamWriter writer = new StreamWriter("filename.txt"))
{
    writer.WriteLine("Текст для записи");
}
```

Пример чтения из файла:

```
using (StreamReader reader = new StreamReader("filename.txt"))
{
    string content = reader.ReadToEnd();
}
```

Открытие и сохранение файлов с диалогом

Для открытия и сохранения файлов с помощью диалоговых окон используются классы:

OpenFileDialog из пространства имен Microsoft.Win32 для открытия файлов.

SaveFileDialog из того же пространства имен для сохранения файлов.

Пример использования OpenFileDialog:

```
Microsoft.Win32.OpenFileDialog openFileDialog = new Microsoft.Win32.OpenFileDialog();
if (openFileDialog.ShowDialog() == true)
{
    string filename = openFileDialog.FileName;
    // Дальнейшая обработка
}
```