

ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

канд. техн. наук, доцент

должность, уч. степень, звание

подпись, дата

Н. В. Богословская

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 5

ИСТОЧНИКИ ДАННЫХ. ЧТЕНИЕ И ЗАПИСЬ ТЕКСТОВЫХ ФАЙЛОВ:
КЛАССЫ STREAMREADER И STREAMWRITER

по курсу:

ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326

подпись, дата

Г. С. Томчук

инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1	Цель работы	3
2	Задача.....	3
3	Ключевые позиции.....	4
3.1	Реализация методов	4
3.2	Создание интерфейса.....	6
4	Тестирование программы.....	7
	ВЫВОДЫ	11

1 Цель работы

Выполнение работы имело следующие цели:

- Изучить и закрепить навыки работы с текстовыми файлами в C# с помощью классов `StreamReader` и `StreamWriter`, а также научиться использовать диалоговые окна для выбора файлов.
- Освоить работу с перегруженными методами и параметрами различных типов, включая ссылочные (`ref`), выходные (`out`) и необязательные параметры.
- Ознакомиться с особенностями передачи массивов как параметров и влиянием этого на их изменение.

2 Задача

11. В приложении пользователь может создать объект класса Текстовый файл, используя классы Файл, Директория. Методы для работы с файлами должны полностью обеспечить пользователя возможностями создания, удаления, изменения, переименования файлов. Подзадачи лабораторной работы № 5 включают в себя:

- Разработать и протестировать перегруженные методы для чтения и записи текстовых файлов (`EditFile()` и `ReadFile()`) с использованием объектов классов `OpenFileDialog` и `SaveFileDialog` для выбора файлов.
- Реализовать чтение и сохранение метаданных о файле, включая авторов, тип файла, дату создания и атрибут доступности (`IsReadOnly`), используя текстовые файлы.
- Создать метод `ChangeAuthors` для изменения массива авторов и продемонстрировать, что изменения сохраняются после выполнения метода, подтверждая передачу массива как ссылочного типа.
- Передать параметр по ссылке (`ref`) в одном из методов и убедиться, что изменения, сделанные в методе, остаются после его выполнения.
- Добавить метод с двумя выходными параметрами (`out`), протестировать его и продемонстрировать использование выходных параметров.

- Добавить метод с необязательным параметром и проверить его работу, показав, как необязательные параметры влияют на вызовы метода.
- Разработать пользовательский интерфейс для тестирования методов, используя различные элементы управления, подходящие для отображаемых данных, и задействовать разнообразные обработчики событий, помимо кнопок, для повышения удобства и современности интерфейса.

3 Ключевые позиции

3.1 Реализация методов

На рис. 1, 2 показаны перегруженные методы ReadFile, EditFile, использующие объекты классов OpenFileDialog, SaveFileDialog, StreamReader, StreamWriter для возможности открывать и сохранять файлы как в пределах рабочей папки, так и любой другой директории.

```
public string ReadFile()
{
    if (File.Exists(Path))
    {
        return File.ReadAllText(Path);
    }
    return string.Empty;
}

public string ReadFile(OpenFileDialog openFileDialog)
{
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        Path = openFileDialog.FileName;
        using (StreamReader reader = new StreamReader(Path))
        {
            return reader.ReadToEnd();
        }
    }
    return null;
}
```

Рисунок 1

```

public void EditFile(string content)
{
    if (File.Exists(Path))
    {
        File.WriteAllText(Path, content);
    }
}

public void EditFile(SaveFileDialog saveFileDialog, string content)
{
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        Path = saveFileDialog.FileName;
        using (StreamWriter writer = new StreamWriter(Path))
        {
            writer.Write(content);
        }
    }
}

```

Рисунок 2

На рис. 3 изображены методы сохранения и загрузки файлов метаданных о файлах. Метаданные хранятся рядом с рабочими файлами в файлах с расширением meta.

```

// Сохранение и чтение метаданных
public void SaveMetadata()
{
    string metadataPath = System.IO.Path.ChangeExtension(Path, ".meta");
    using (StreamWriter writer = new StreamWriter(metadataPath))
    {
        writer.WriteLine(string.Join(", ", Authors));
        writer.WriteLine(FileTypeProperty);
        writer.WriteLine(DateCreated);
        writer.WriteLine(IsReadOnly);
    }
}

public void LoadMetadata()
{
    string metadataPath = System.IO.Path.ChangeExtension(Path, ".meta");
    if (File.Exists(metadataPath))
    {
        using (StreamReader reader = new StreamReader(metadataPath))
        {
            // Здесь, поскольку BindingList является "оберткой" для передаваемого при его инициализации списка,
            // нужно использовать копию списка, чтобы избежать ошибки "Read-Only List" при попытке изменения BindingList
            BindingList<string> parsedAuthors = new BindingList<string>(reader.ReadLine().Split(',').ToList());
            if (parsedAuthors[0] != "") Authors = parsedAuthors;

            FileTypeProperty = reader.ReadLine();
            DateCreated = DateTime.Parse(reader.ReadLine());
            IsReadOnly = bool.Parse(reader.ReadLine());
        }
    }
    else
    {
        SaveMetadata();
        LoadMetadata();
    }
}

```

Рисунок 3

На рис. 4 показаны методы изменения списка авторов, изменения типа файла, чтения деталей о файле и добавления автора в список. Методы соответственно позволяют протестировать ссылочное изменение массива при непосредственной подаче его в метод, изменение аргумента, поданного через ref, получение нескольких выходных значений с помощью out, а также

поведение метода при подаче в него необязательного параметра.

```
// Метод изменения авторов с использованием массива
public void ChangeAuthors(List<string> newAuthors)
{
    newAuthors.Add("Добавленный автор");
    Authors = new BindingList<string>(newAuthors);
}

// Метод с параметром по ссылке (ref)
public void ModifyFileType(ref string type)
{
    type = "Польз.: " + type;
    FileTypeProperty = type;
}

// Метод с двумя выходными параметрами (out)
public void GetFileDetails(out string type, out DateTime creationDate)
{
    type = FileTypeProperty;
    creationDate = DateCreated;
}

// Метод с необязательным параметром
public void AddAuthor(string authorName = "Неизвестный автор")
{
    Authors.Add(authorName);
}
```

Рисунок 4

3.2 Создание интерфейса

На рис. 5 изображен метод обработчика добавления автора в список авторов. При неимении значения имени автора в список добавляется автор со стандартным именем.

```
private void buttonFileAuthorsAdd_Click(object sender, EventArgs e)
{
    if (textBox_FileAuthorsToAdd.Text != "")
    {
        string authorToAdd = textBox_FileAuthorsToAdd.Text;
        currentFile.AddAuthor(authorToAdd);
        textBox_FileAuthorsToAdd.Text = "";
    }
    else
    {
        currentFile.AddAuthor();
    }
}
```

Рисунок 5

На рис. 6 показаны методы обработчиков изменения типа файла по ref, чтения деталей файла через out и изменения всего списка авторов на предуказанные значения через прямую передачу списка в метод.

```

private void buttonChangeTypeByRef_Click(object sender, EventArgs e)
{
    string fileType = comboBox_FileType.Text;
    string initialFileType = fileType;
    currentFile.ModifyFileType(ref fileType);
    MessageBox.Show($"Переданный тип до вызова метода: {initialFileType}\nПереданный тип после вызова метода: {fileType}");
    fileList.ResetBindings();
}

private void buttonReadFileDetails_Click(object sender, EventArgs e)
{
    currentFile.GetFileDetails(out string type, out DateTime creationDate);
    MessageBox.Show($"Тип файла: {type}\nСоздан: {creationDate}");
}

private void buttonChangeAuthorsList_Click(object sender, EventArgs e)
{
    List<string> newAuthors = new[] { "Автор 1", "Автор 2", "Автор 3" };
    List<string> initialNewAuthors = newAuthors.ToList();
    currentFile.ChangeAuthors(newAuthors);
    MessageBox.Show($"Переданный список до вызова метода: {String.Join(", ", initialNewAuthors)}\n\nПереданный список после вызова метода: {String.Join(", ", newAuthors)}");
    fileList.ResetBindings();
}

```

Рисунок 6

4 Тестирование программы

На рис. 7, 8, 9, 10, 11 представлены результаты тестирования программы. На рис. 7 изображено окно открытия файла для редактирования.

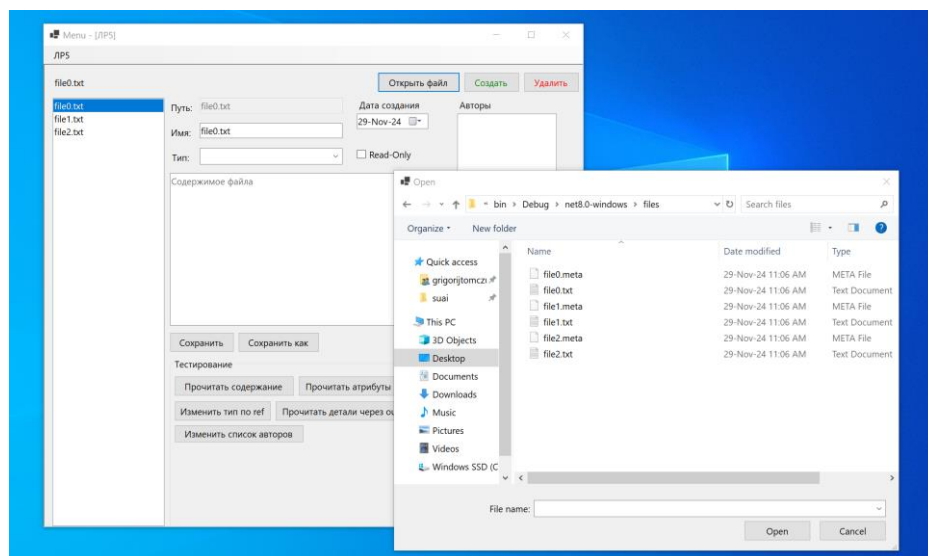


Рисунок 7

На рис. 8 показана структура рабочей папки программы, в которой находятся файлы метаданных.







Name	Date modified	Type	Size
 file0.meta	29-Nov-24 11:06 AM	META File	1 KB
 file0.txt	29-Nov-24 11:06 AM	Text Document	0 KB
 file1.meta	29-Nov-24 11:06 AM	META File	1 KB
 file1.txt	29-Nov-24 11:06 AM	Text Document	0 KB
 file2.meta	29-Nov-24 11:06 AM	META File	1 KB
 file2.txt	29-Nov-24 11:06 AM	Text Document	0 KB

Рисунок 8

На рис. 9 показано окно тестирования метода с ссылочным параметром ref.

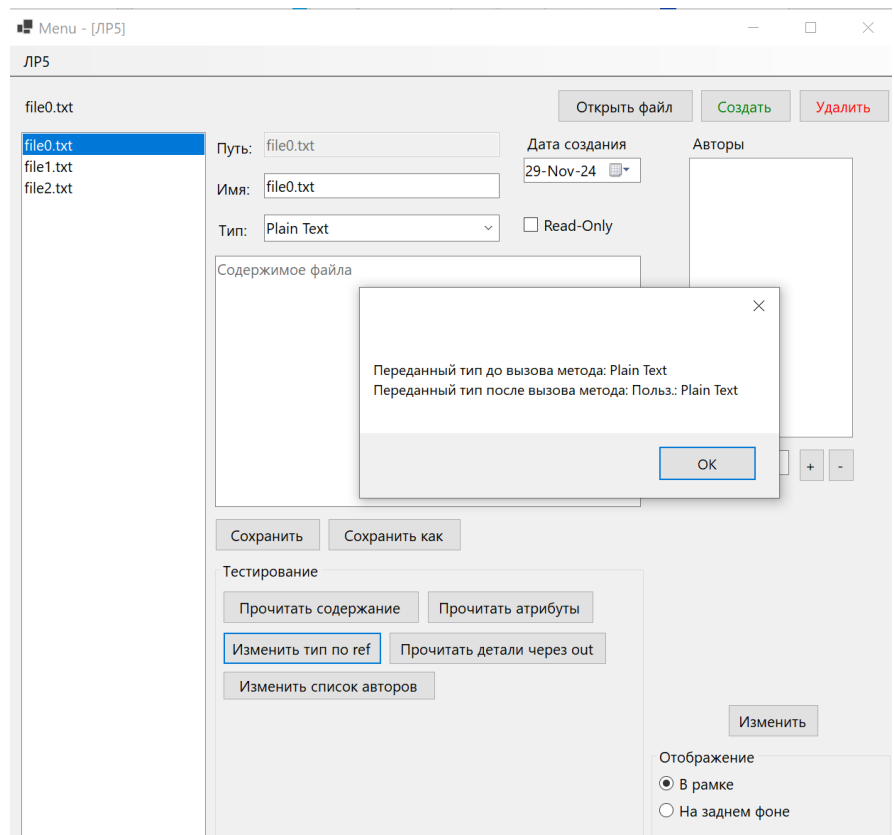


Рисунок 9

На рис. 10 показано окно тестирования метода с двумя выходными параметрами out.

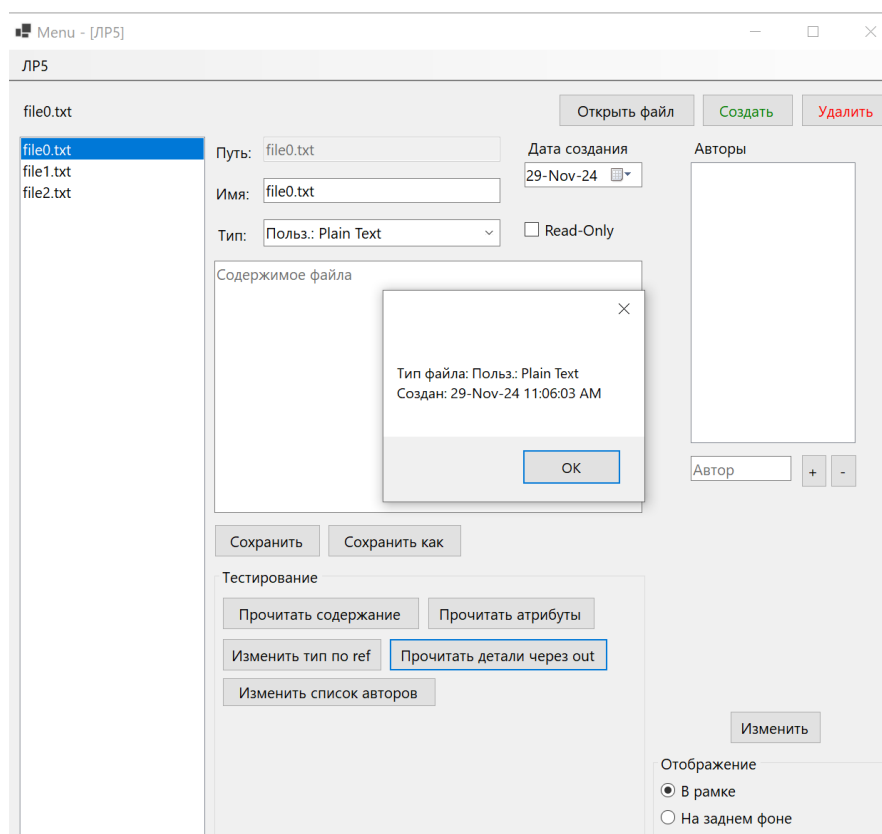


Рисунок 10

На рис. 11 изображено окно тестирования метода, изменяющего передаваемый в него список.

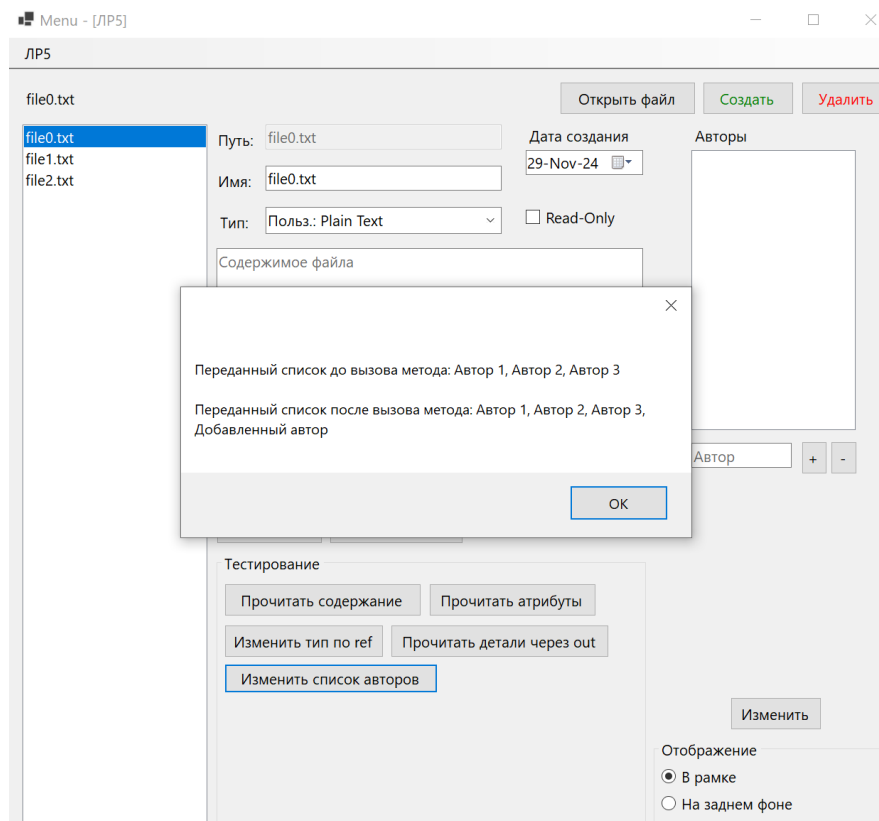


Рисунок 11

ВЫВОДЫ

- Работа с текстовыми файлами с помощью `StreamReader` и `StreamWriter` позволяет легко реализовывать операции чтения и записи данных в файлы, а также предоставляет гибкость при обработке содержимого файлов.
- Работа с массивами как ссылочными типами продемонстрировала важность правильного понимания передачи параметров в методы. Изменения, сделанные в массиве, переданном в метод, сохраняются, что может быть полезно в ряде задач, но требует внимательного подхода, чтобы избежать нежелательных изменений данных.
- Передача параметров по ссылке (`ref`) и использование выходных параметров (`out`) обеспечивают дополнительную гибкость при разработке методов, позволяя сохранять изменения в параметрах или возвращать несколько значений из метода. Это особенно полезно, когда требуется возвращать не одно, а несколько значений.
- Использование необязательных параметров упрощает вызов методов и делает код более лаконичным, позволяя вызывать методы с минимально необходимыми аргументами или задавать стандартные значения для некоторых параметров.