ЛАБОРАТОРНАЯ РАБОТА №6

Курс «Алгоритмизация и программирование»

Тема: Работа с файлами.

Цель: Научиться программно работать с бинарными и текстовыми файлами в файловой системе ОС Windows.

Ход работы

Вариант 8

1. Дополнить код лабораторной работы №5, организовав хранение данных предметной области в бинарном (четные варианты) или текстовом (нечетные варианты) файле lab.dat. Данные должны считываться из файла при запуске программы и записываться в файл при закрытии программы.

Вариант 8

Общественный транспорт			
Вид транспорта	№ маршрута	Протяженность маршрута (км)	Время в дороге (мин)
Тр	12	27.55	75
Т-с	17	13.6	57
А	12а	57.3	117
Перечисляемый тип: Тр - трамвай, Тс - троллейбус, А - автобус			

```
// Метод для записи данных в файл
Ссылок: 1
static void SaveData(Transport[] transports, int transportCount, LogEntry[] logs, int logCount)
{
    using (BinaryWriter writer = new BinaryWriter(File.Open("lab.dat", FileMode.Create)))
    {
        // Записываем количество элементов в массиве
        writer.Write(transportCount);
        writer.Write(logCount);

        // Записываем данные о транспорте
        for (int i = 0; i < transportCount; i++)
        {
            writer.Write(transports[i].Type);
            writer.Write(transports[i].RouteNumber);
            writer.Write(transports[i].Length);
            writer.Write(transports[i].Time);
        }

        // Записываем данные о логах
        for (int i = 0; i < logCount; i++)
        {
            writer.Write(logs[i].Action);
            writer.Write(logs[i].Details);
            writer.Write(logs[i].Timestamp.ToString());
        }
    }
}
```

```
// Метод для чтения данных из файла
Ссылок: 1
static void LoadData(ref Transport[] transports, ref int transportCount, ref LogEntry[] logs, ref int logCount)
{
    if (File.Exists("lab.dat"))
    {
        using (BinaryReader reader = new BinaryReader(File.Open("lab.dat", FileMode.Open)))
        {
            transportCount = reader.ReadInt32();
            logCount = reader.ReadInt32();

            // Загружаем данные о транспорте
            transports = new Transport[transportCount];
            for (int i = 0; i < transportCount; i++)
            {
                transports[i].Type = reader.ReadString();
                transports[i].RouteNumber = reader.ReadString();
                transports[i].Length = reader.ReadDouble();
                transports[i].Time = reader.ReadInt32();
            }

            // Загружаем данные о логах
            logs = new LogEntry[logCount];
            for (int i = 0; i < logCount; i++)
            {
                logs[i].Action = reader.ReadString();
                logs[i].Details = reader.ReadString();
                logs[i].Timestamp = DateTime.Parse(reader.ReadString());
            }
        }
    }
}
```

```
switch (choice)
{
    case 1:
        ViewTable(transports, transportCount);
        break;
    case 2:
        AddRecord(ref transports, ref transportCount, ref logs, ref logCount);
        break;
    case 3:
        DeleteRecord(ref transports, ref transportCount, ref logs, ref logCount);
        break;
    case 4:
        UpdateRecord(transports, transportCount, ref logs, ref logCount);
        break;
    case 5:
        SearchRecords(transports, transportCount);
        break;
    case 6:
        ViewLog(logs, logCount, maxIdleTime);
        break;
    case 7:
        Console.WriteLine("Завершение работы программы.");
        // Сохранение данных в файл перед завершением
        SaveData(transports, transportCount, logs, logCount);
        return;
    default:
        Console.WriteLine("Неверный ввод. Пожалуйста, выберите пункт из списка.");
        break;
}
```

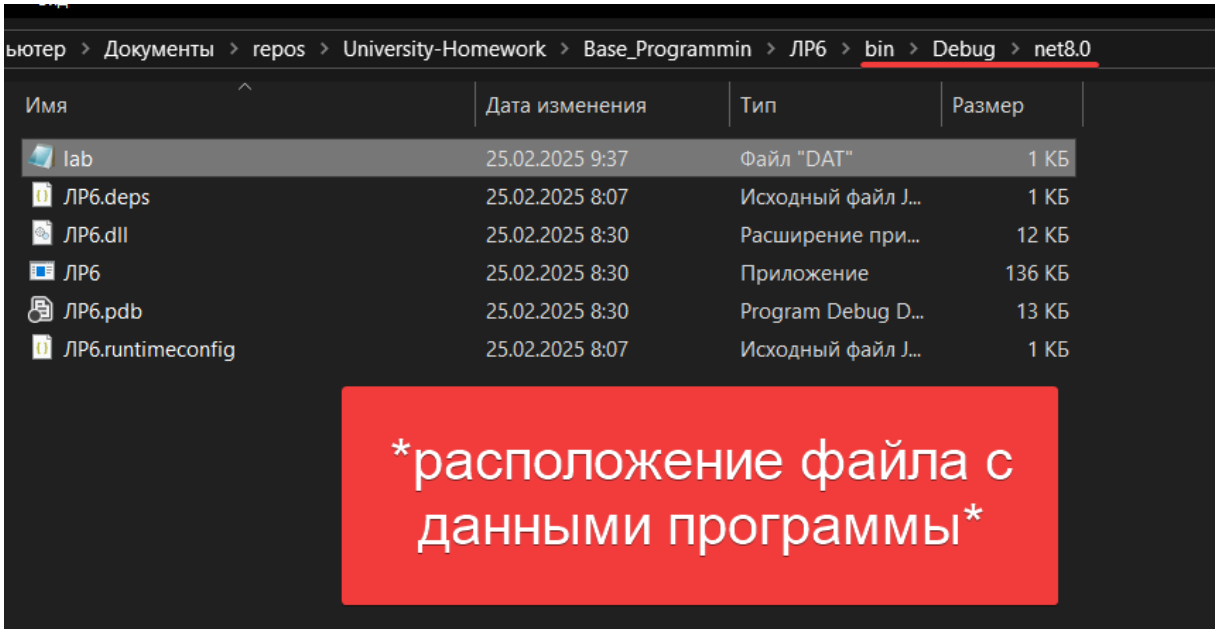
```
static void Main()
{
    Transport[] transports = new Transport[50];
    LogEntry[] logs = new LogEntry[50];
    int transportCount = 0;
    int logCount = 0;
    DateTime lastActionTime = DateTime.Now; // Время последнего действия
    TimeSpan maxIdleTime = TimeSpan.Zero; // Самое долгое время бездействия пользователя

    // Чтение данных из файла при запуске программы
    LoadData(ref transports, ref transportCount, ref logs, ref logCount);

    while (true)
    {
```

```
Список транспорта
-------------------------------------------------------------------------------
Тип транспорта  Маршрут         Дистанция (км)              Длительность (мин)
-------------------------------------------------------------------------------
Тр              31а             10                          23
А               56              2                           10
М               35              12                          30
-------------------------------------------------------------------------------

Доступные действия:
1 - Показать таблицу
2 - Добавить новую запись
3 - Удалить запись
4 - Обновить запись
5 - Найти записи
6 - Показать лог действий
7 - Завершить работу
Введите номер действия: 7
Завершение работы программы.
```

Добавили несколько записей и завершили работу программы

```
Доступные действия:
1 - Показать таблицу
2 - Добавить новую запись
3 - Удалить запись
4 - Обновить запись
5 - Найти записи
6 - Показать лог действий
7 - Завершить работу
Введите номер действия: 1

Список транспорта
-------------------------------------------------------------------------------
Тип транспорта  Маршрут         Дистанция (км)              Длительность (мин)
-------------------------------------------------------------------------------
Тр              31а             10                          23
А               56              2                           10
М               35              12                          30
-------------------------------------------------------------------------------

Доступные действия:
1 - Показать таблицу
2 - Добавить новую запись
3 - Удалить запись
4 - Обновить запись
5 - Найти записи
```

Запустили программу снова, записи успешно загрузились



2. Написать программу для работы с бинарными файлами, в соответствии с вариантом (приложение А).

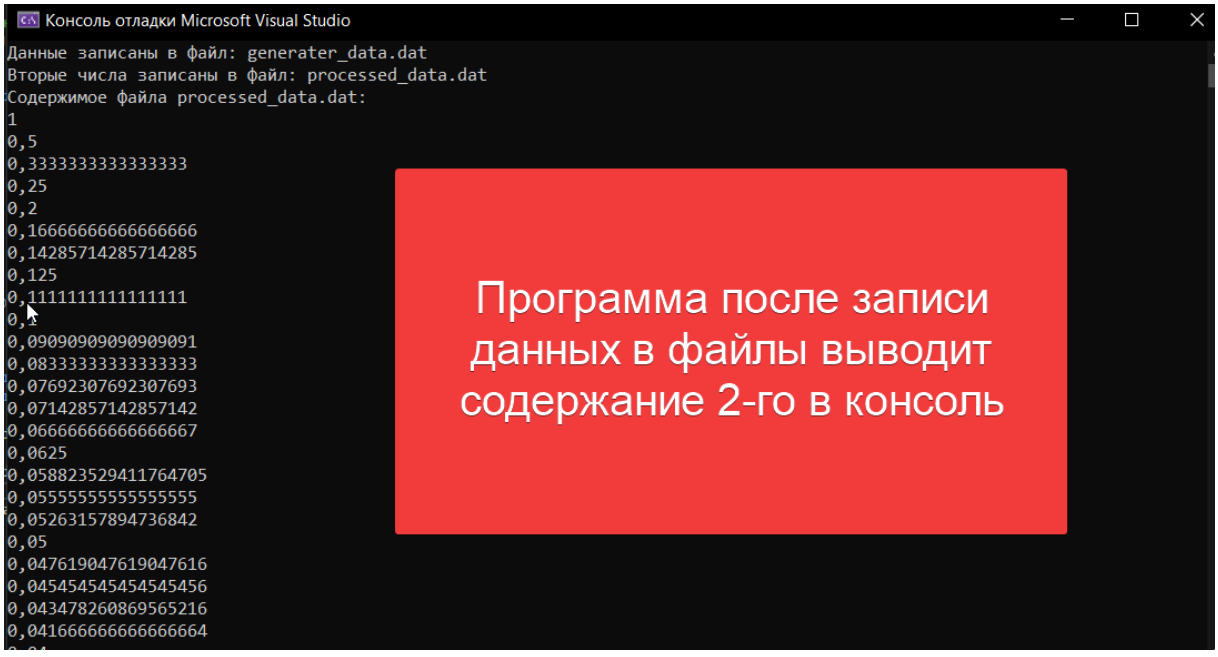
Вариант 8.
Программно записать в бинарный файл набор пар «N – 1/N» для N от 1 до 100. Написать функцию, которая считывает из этого файла все вторые числа из каждой пары и записывает во второй файл.

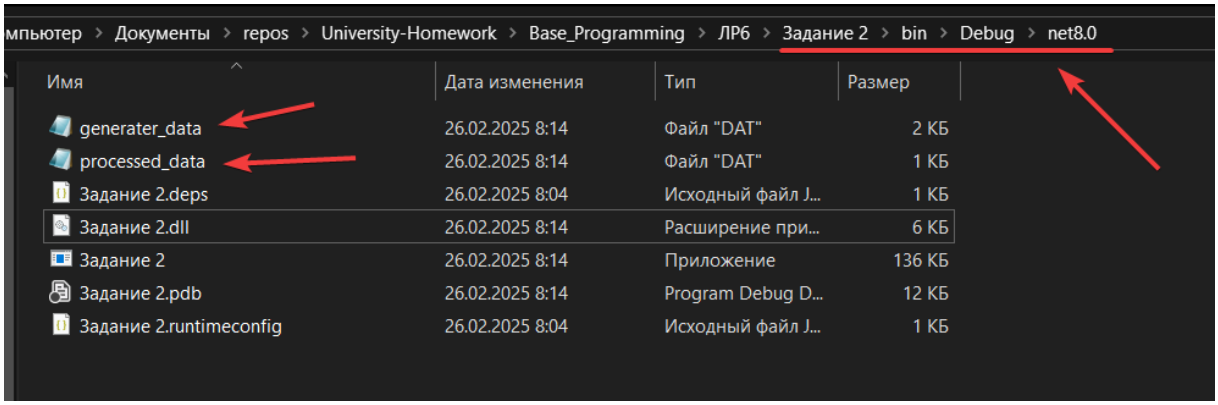
```
using System;
using System.IO;

Ссылок: 0
class Program
{
    // Функция для записи пар чисел
    Ссылок: 1
    static void WritePairsToFile(string filePath)
    {
        using (BinaryWriter writer = new BinaryWriter(File.Open(filePath, FileMode.Create)))
        {
            for (int N = 1; N <= 100; N++)
            {
                writer.Write(N);            // Записываем число N
                writer.Write(1.0 / N);      // Записываем число 1/N
            }
        }
        Console.WriteLine("Данные записаны в файл: " + filePath);
    }
```

```
// Функция для записи всех вторых чисел в новый файл
Ссылок: 1
static void WriteSecondNumbersToFile(string inputFilePath, string outputFilePath)
{
    using (BinaryReader reader = new BinaryReader(File.Open(inputFilePath, FileMode.Open)))
    using (BinaryWriter writer = new BinaryWriter(File.Open(outputFilePath, FileMode.Create)))
    {
        while (reader.BaseStream.Position < reader.BaseStream.Length)
        {
            int N = reader.ReadInt32();         // Считываем N
            double secondNumber = reader.ReadDouble();  // Считываем 1/N

            // Записываем только второе число
            writer.Write(secondNumber);
        }
    }
    Console.WriteLine("Вторые числа записаны в файл: " + outputFilePath);
}
```

```csharp
// Функция для чтения и вывода данных из файла
Ссылок: 1
static void ReadAndDisplayFile(string filePath)
{
    using (BinaryReader reader = new BinaryReader(File.Open(filePath, FileMode.Open)))
    {
        Console.WriteLine($"Содержимое файла {filePath}:");
        while (reader.BaseStream.Position < reader.BaseStream.Length)
        {
            double number = reader.ReadDouble();
            Console.WriteLine(number);
        }
    }
}
```

```csharp
static void Main()
{
    // Пути к файлам
    string inputFilePath = "generater_data.dat";
    string outputFilePath = "processed_data.dat";

    WritePairsToFile(inputFilePath);

    WriteSecondNumbersToFile(inputFilePath, outputFilePath);

    ReadAndDisplayFile(outputFilePath);
}
```



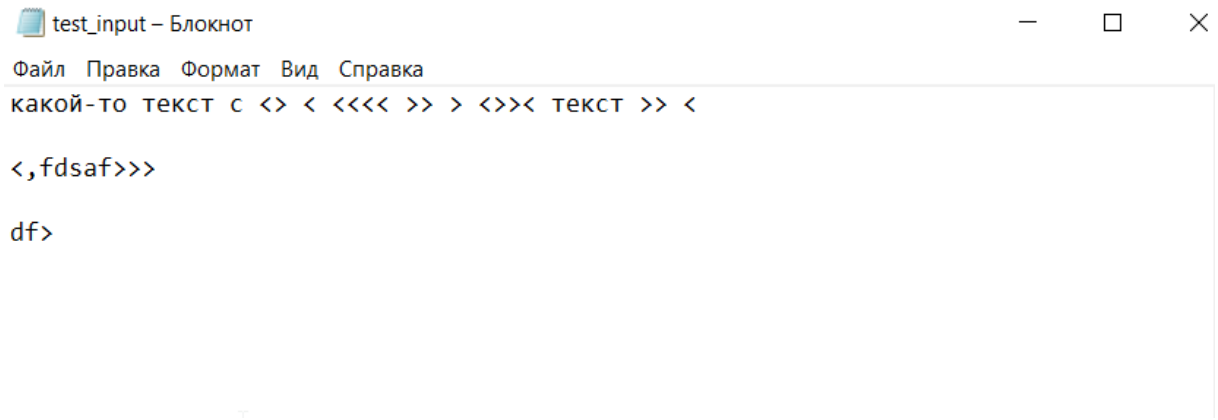


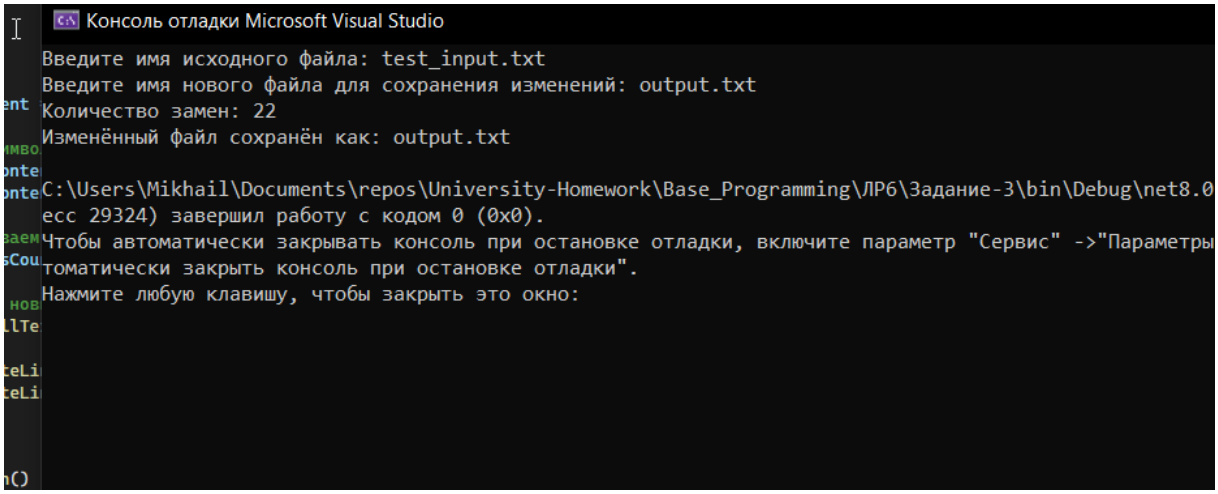
3. Написать программу для работы с текстовыми файлами, в соответствии с вариантом (приложение Б).

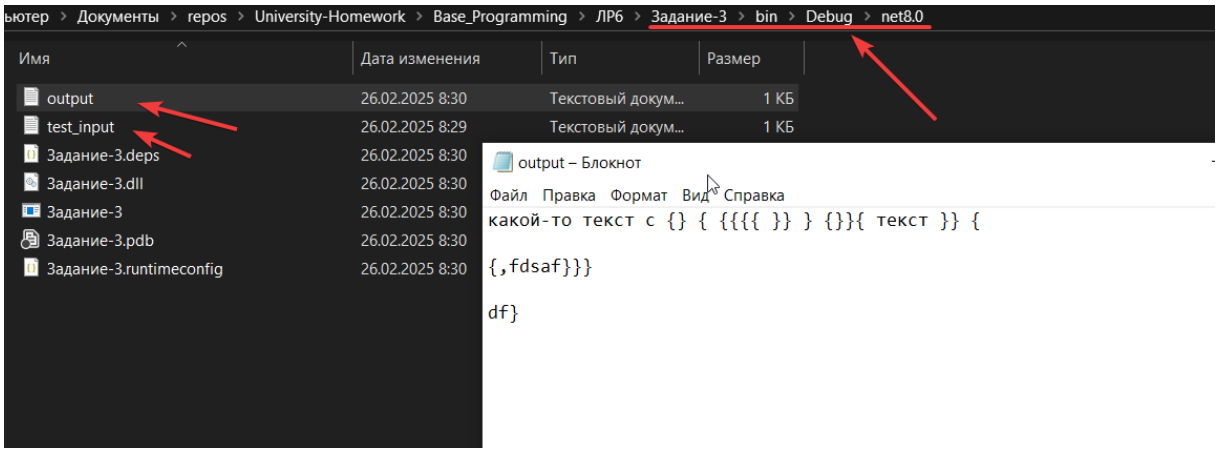
Вариант 8.
Считать текстовый файл, сформировать новый файл, в котором заменить все символы «<» и «>» на символы «{» и «}». Вывести на консоль количество замен.

```
using System;
using System.IO;

Ссылок: 0
class Program
{
    // Функция для замены символов в файле
    Ссылок: 1
    static void ReplaceSymbolsInFile(string inputFilePath, string outputFilePath)
    {
        int replacementsCount = 0;

        // Существует ли исходный файл
        if (!File.Exists(inputFilePath))
        {
            Console.WriteLine($"Ошибка: файл {inputFilePath} не найден.");
            return;
        }


        string content = File.ReadAllText(inputFilePath);

        // Замена символов
        content = content.Replace("<", "{");
        content = content.Replace(">", "}");

        // Подсчитываем количество замен
        replacementsCount = (content.Split('{').Length - 1) + (content.Split('}').Length - 1);

        // Запись в новый файл
        File.WriteAllText(outputFilePath, content);

        Console.WriteLine($"Количество замен: {replacementsCount}");
        Console.WriteLine($"Изменённый файл сохранён как: {outputFilePath}");
    }
```

```
static void Main()
{
    Console.Write("Введите имя исходного файла: ");
    string inputFilePath = Console.ReadLine();

    Console.Write("Введите имя нового файла для сохранения изменений: ");
    string outputFilePath = Console.ReadLine();

    // Замена символов в файле
    ReplaceSymbolsInFile(inputFilePath, outputFilePath);
}
```





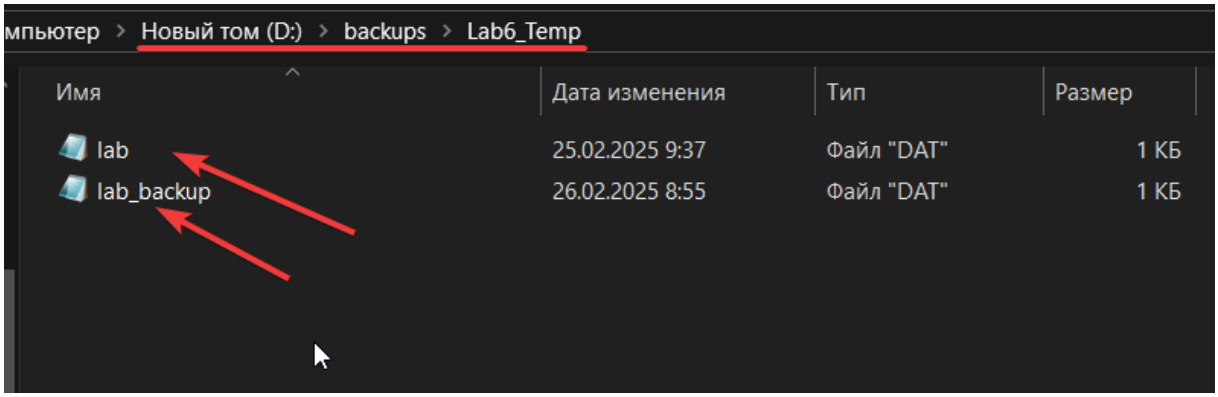


4. Написать программу, которая создает на одном из разделов жесткого диска директорию Lab6_Temp, автоматически копирует в эту директорию Ваш файл lab.dat из задания 1 и создает в ней копию этого файла lab_backup.dat путем побайтового копирования. Вывести на консоль информацию о файле lab.dat: размер, время последнего изменения, время последнего доступа.

```
using System;
using System.IO;

Ссылок: 0
class Program
{
    Ссылок: 0
    static void Main()
    {
        string sourceFilePath = @"C:\Users\Mikhail\Documents\repos\University-Homework\Base_Programming\ЛР6\Задание-1\bin\Debug\net8.0\lab.dat";

        string destinationDirectory = @"D:\backups\Lab6_Temp";

        string destinationFilePath = Path.Combine(destinationDirectory, "lab.dat");

        string backupFilePath = Path.Combine(destinationDirectory, "lab_backup.dat");
```

```
try
{
    // Создаем директорию, если она не существует
    if (!Directory.Exists(destinationDirectory))
    {
        Directory.CreateDirectory(destinationDirectory);
        Console.WriteLine($"Директория {destinationDirectory} была создана.");
    }

    // Проверяем, существует ли исходный файл
    if (File.Exists(sourceFilePath))
    {
        // Копируем файл lab.dat в новую директорию
        File.Copy(sourceFilePath, destinationFilePath, true);
        Console.WriteLine($"Файл {sourceFilePath} скопирован в {destinationFilePath}");

        // Создаем резервную копию lab.dat побайтово
        using (FileStream sourceStream = new FileStream(sourceFilePath, FileMode.Open, FileAccess.Read))
        using (FileStream backupStream = new FileStream(backupFilePath, FileMode.Create, FileAccess.Write))
        {
            int byteRead;
            while ((byteRead = sourceStream.ReadByte()) != -1)
            {
                backupStream.WriteByte((byte)byteRead);
            }
        }
        Console.WriteLine($"Резервная копия файла создана в {backupFilePath}");

        // Записываем информацию о файле lab.dat
        FileInfo fileInfo = new FileInfo(sourceFilePath);

        Console.WriteLine("\nИнформация о файле lab.dat:");
        Console.WriteLine($"Размер: {fileInfo.Length} байт");
        Console.WriteLine($"Время последнего изменения: {fileInfo.LastWriteTime}");
        Console.WriteLine($"Время последнего доступа: {fileInfo.LastAccessTime}");
    }
    else
    {
        Console.WriteLine($"Файл {sourceFilePath} не найден.");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"Ошибка: {ex.Message}");
}
```

```
Консоль отладки Microsoft Visual Studio
Файл C:\Users\Mikhail\Documents\repos\University-Homework\Base_Programming\ЛР6\Задание-1\bin\Debug\net8.0
\lab.dat скопирован в D:\backups\Lab6_Temp\lab.dat
Резервная копия файла создана в D:\backups\Lab6_Temp\lab_backup.dat

Информация о файле lab.dat:
Размер: 216 байт
Время последнего изменения: 25.02.2025 9:37:18
Время последнего доступа: 26.02.2025 8:55:58

C:\Users\Mikhail\Documents\repos\University-Homework\Base_Programming\ЛР6\Задание-4\bin\Debug\net8.0\Зада
ние-4.exe (процесс 16432) завершил работу с кодом 0 (0x0).
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"
Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```



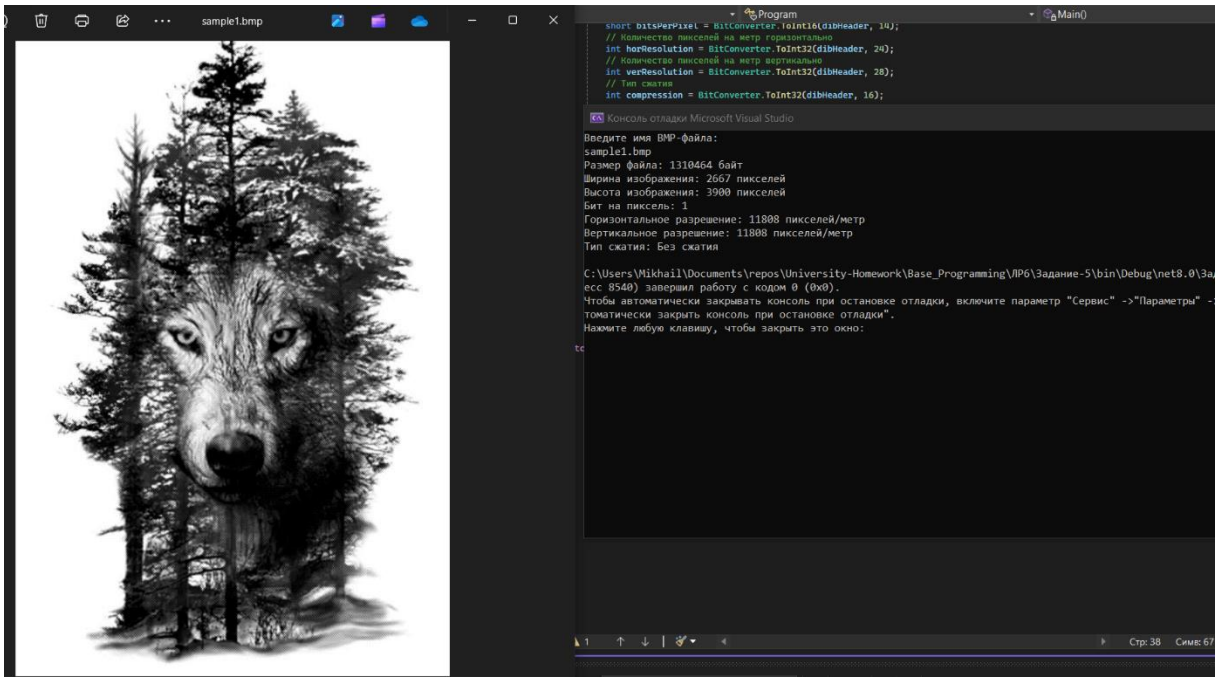
5. Написать программу, которая позволяет ввести имя bmp-файла, считать его заголовки и вывести на консоль информацию о размере файла, ширине и высоте в пикселях, количестве бит на пиксель, разрешении горизонтальном и вертикальном (количестве пикселей на метр), типе сжатия (без сжатия / 4бит RLE / 8бит RLE). Подготовьте несколько файлов изображений и проверьте на них Вашу программу. Структура bmp-файла приведена в приложении В.

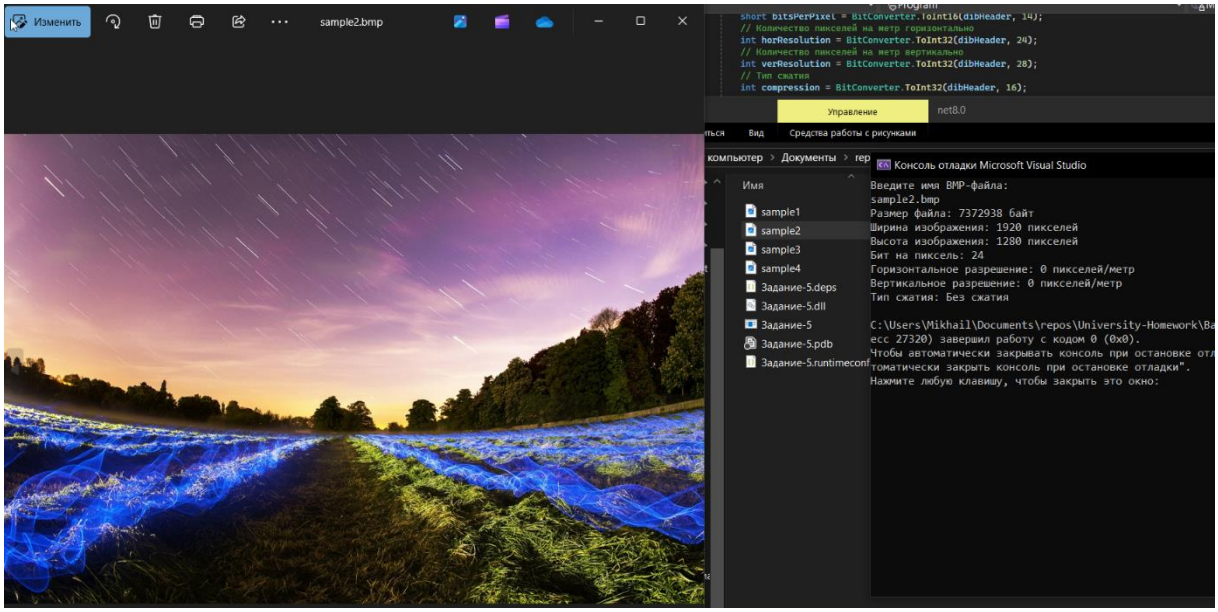
```
using System;
using System.IO;
using System.Text;

Ссылок: 0
class Program
{
    Ссылок: 0
    static void Main()
    {
        Console.WriteLine("Введите имя BMP-файла:");
        string fileName = Console.ReadLine();

        if (!File.Exists(fileName))
        {
            Console.WriteLine("Файл не найден!");
            return;
        }
```

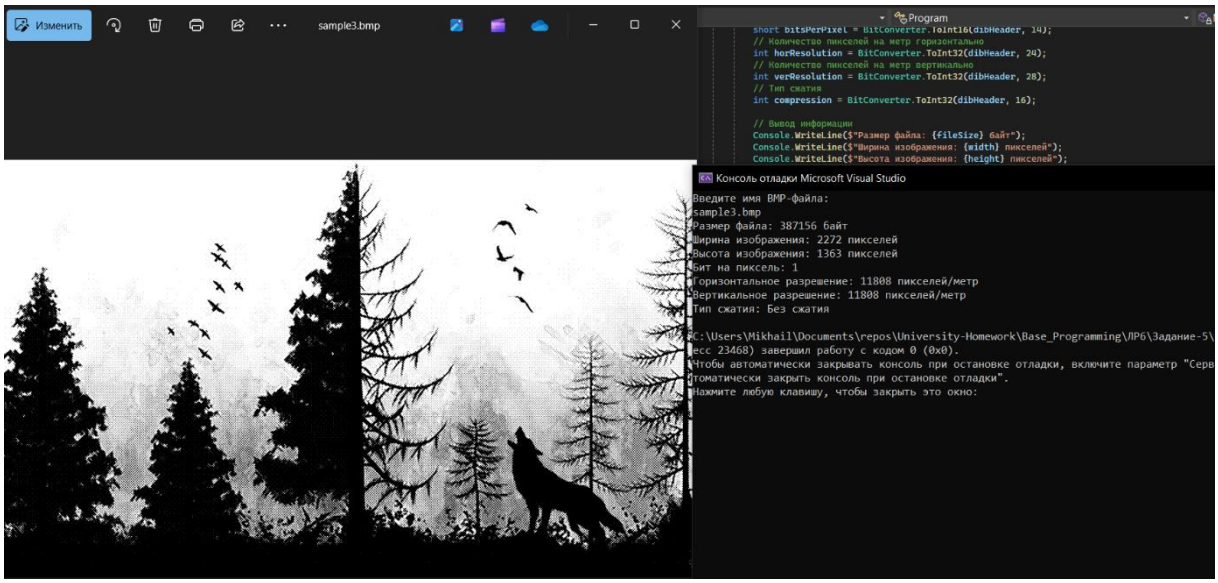
```
try
{
    using (FileStream fs = new FileStream(fileName, FileMode.Open, FileAccess.Read))
    {
        // Чтение заголовка файла
        byte[] fileHeader = new byte[14];
        fs.Read(fileHeader, 0, 14);

        // Проверка сигнатуры BMP
        if (fileHeader[0] != 'B' || fileHeader[1] != 'M')
        {
            Console.WriteLine("Это не BMP файл.");
            return;
        }
```

```
            // Чтение заголовка изображения
            byte[] dibHeader = new byte[40];
            fs.Read(dibHeader, 0, 40);
            int fileSize = BitConverter.ToInt32(fileHeader, 2);
            // Смещение к пиксельным данным
            int offset = BitConverter.ToInt32(fileHeader, 10);
            int width = BitConverter.ToInt32(dibHeader, 4);
            int height = BitConverter.ToInt32(dibHeader, 8);
            short bitsPerPixel = BitConverter.ToInt16(dibHeader, 14);
            // Количество пикселей на метр горизонтально
            int horResolution = BitConverter.ToInt32(dibHeader, 24);
            // Количество пикселей на метр вертикально
            int verResolution = BitConverter.ToInt32(dibHeader, 28);
            // Тип сжатия
            int compression = BitConverter.ToInt32(dibHeader, 16);
```

```
            // Вывод информации
            Console.WriteLine($"Размер файла: {fileSize} байт");
            Console.WriteLine($"Ширина изображения: {width} пикселей");
            Console.WriteLine($"Высота изображения: {height} пикселей");
            Console.WriteLine($"Бит на пиксель: {bitsPerPixel}");

            Console.WriteLine($"Горизонтальное разрешение: {horResolution} пикселей/метр");
            Console.WriteLine($"Вертикальное разрешение: {verResolution} пикселей/метр");

            // Определение типа сжатия
            string compressionType = compression switch
            {
                0 => "Без сжатия",
                1 => "RLE 4-бит",
                2 => "RLE 8-бит",
                _ => "Неизвестный тип сжатия"
            };
            Console.WriteLine($"Тип сжатия: {compressionType}");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Ошибка при чтении файла: " + ex.Message);
    }
  }
}
```

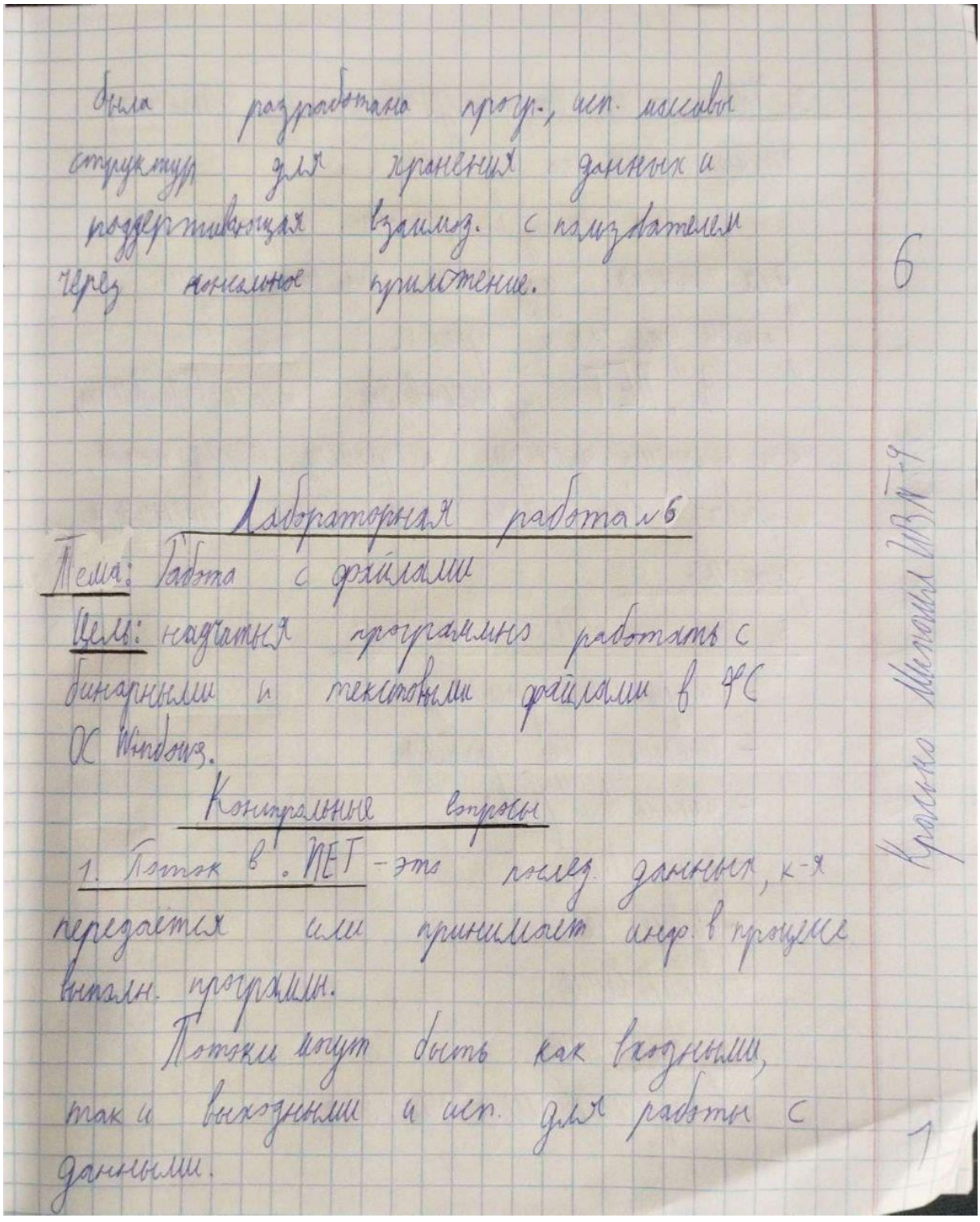






Контрольные вопросы на следующей странице

Контрольные вопросы



была разработана прогр., исп. массивы структур для хранения данных и поддерживающая взаимод. с пользователем через консольное приложение.

6

Лабораторная работа №6

Тема: Работа с файлами

Цель: научиться программно работать с бинарными и текстовыми файлами в ОС Windows.

Контрольные вопросы

1. Поток в .NET - это послед. данных, к-я передается или принимается инф. в процессе выполн. программы.

Потоки могут быть как входными, так и выходными и исп. для работы с данными.

Краснов Михаил ИВТ-9

7

2. Файл – это структурный набор данных, хранящийся на диске.

Потоки исп. для работы с файлами: для чтения данных из файла или записи данных в файл.

В .NET существует класс FileStream, к-й предост. доступ к файлу через поток, позволяя выполнять операции чтения и записи.

3. Осн. файловые операции:

- открытие файла
- чтение из файла
- запись в файл
- закрытие
- удаление
- копирование
- перемещение
- получение метаданных

Кулешов Николай ИВТ-4

2

4. Для работы с текстовыми файлами в .NET есть исп. классы из пространства имён System.IO:
- StreamReader
- StreamWriter.

Пример:

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

5. Для работы с бинарными файлами в .NET исп. классы BinaryReader и BinaryWriter, к-е позволяет читать и записывать данные в к-м формате.

```
using (BinaryReader reader = new BinaryReader
(File.Open(" test.txt")
{
    int value = reader.ReadInt32();
}
```

Краснов Школьный ИВТ-4

6. Прямой доступ к файлу означает возможность читать или записывать данные в произвольном порядке, не следуя последовательности.

Это достигается с исп. RandomAccessFile или аналогич. механизмов в .NET, таких как класс FileStream с указанием позиции в файле через метод Seek.

7 В .NET для работы с ФС можно исп. методы из классов File и Directory();

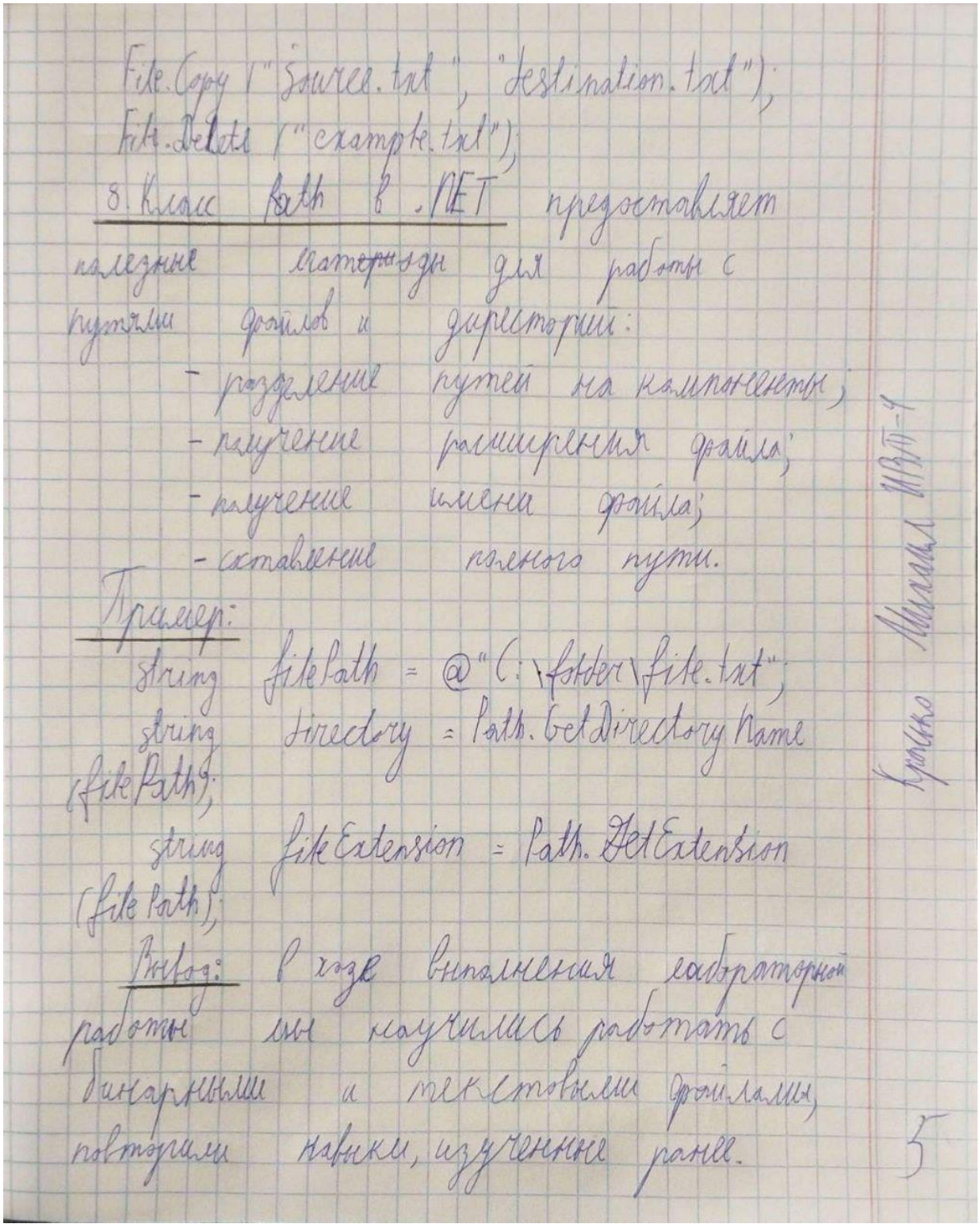
- Создание файла/директории: File.Create(), Directory.CreateDirectory();
- File.Delete(), Directory.Delete();
- File.Delete(), Directory.Delete();
- File.Copy();
- File.Move().

Пример:

File.Create("example.txt");

Кравец Милослав ПИЭ-4

9


File.Copy ("source.txt", "destination.txt");
File.Delete ("example.txt");

8. Класс Path в .NET предоставляет полезные методы для работы с путями файлов и директорий:

- разделение путей на компоненты;
- получение расширения файла;
- получение имени файла;
- составление полного пути.

Пример:

string filePath = @"C:\folder\file.txt";
string directory = Path.GetDirectoryName (filePath);
string fileExtension = Path.GetExtension (filePath);

Вывод: в ходе выполнения лабораторной работы мы научились работать с бинарными и текстовыми файлами, повторили навыки, изученные ранее.

Кунько Михаил ПИЭ-1

5

Вывод: в ходе выполнения лабораторной работы мы научились работать с бинарными и текстовыми файлами, повторили навыки по работе с Windows Forms, изученные ранее.