**16 Работа с файлами**

Задание 1. Компонентами файла g являются натуральные числа. Найти количество четных чисел среди компонент.

Листинг программы:

string filePath = @"./f.txt"; int evenNumbersCount = 0;

string fileContent = File.ReadAllText(filePath);

string[] components = fileContent.Split(' ');

foreach (string component in components)

{ if (int.TryParse(component, out int number)) {

if (number % 2 == 0) {evenNumbersCount++; }}}

Console.WriteLine($"Количество четных чисел: {evenNumbersCount}");

Таблица 16.1 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1 2 3 4 5 6 7 8 9 | Количество четных чисел: 4 |

Источник: собственная разработка

Анализ результатов:



Рисунок 16.1 – Результат работы программы

Источник: собственная разработка

Задание 2. С помощью метода CreateDirectory() создайте папку “New\_folder”.

Листинг программы:

Directory.CreateDirectory(@".\New\_folder\");

Таблица 16.2 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Создана папка New\_folder |

Источник: собственная разработка

Анализ результатов:



Рисунок 16.2 – Результат работы программы

Источник: собственная разработка

Задание 3. Создать текстовый файл, в который записать 5 строк различной длины.Для данного файла выполнить следующие действия: вывести весь файл на экран; подсчитать количество строк; подсчитать количество символов в каждой строке; удалить последнюю строку из файла, результат записать в новый файл; вывести на экран строки с s1 по s2; найти длину самой длинной строки и вывести ее на экран; вывести на экран все строки начинающиеся с заданной буквы;

переписать его строки в другой файл, порядок строк во втором файле должен быть обратным по отношению к порядку строк в заданном файле.

Листинг программы:

string fileName = "file.txt";

using (StreamWriter writer = new StreamWriter(fileName))

{ writer.WriteLine("Волшебная строка");writer.WriteLine("Обычная строка");

writer.WriteLine("Самая интересная строка");

writer.WriteLine("Спокойная строка");

writer.WriteLine("Интригующая строка");}

Console.WriteLine("Содержимое файла:");

using (StreamReader reader = new StreamReader(fileName))

{Console.WriteLine(reader.ReadToEnd());} Console.WriteLine();

int linesCount = File.ReadAllLines(fileName).Length;

Console.WriteLine($"Количество строк в файле: {linesCount}");

Console.WriteLine("Количество символов в каждой строке:");

using (StreamReader reader = new StreamReader(fileName))

{string line; while ((line = reader.ReadLine()) != null)

{Console.WriteLine($"{line} - {line.Length} символов");}}

Console.WriteLine(); string newFileName = "new\_file.txt";

List<string> lines = File.ReadAllLines(fileName).ToList();

lines.RemoveAt(lines.Count - 1); File.WriteAllLines(newFileName, lines);

Console.WriteLine($"Удалена последняя строка из файла {fileName}, результат записан в файл {newFileName}"); Console.WriteLine();

int s1 = 1; int s2 = 2; Console.WriteLine($"Строки с {s1} по {s2}:");

lines = File.ReadAllLines(fileName).ToList();

for (int i = s1 - 1; i < s2 && i < lines.Count; i++) {Console.WriteLine(lines[i]);} Console.WriteLine();

int maxLineLength = File.ReadAllLines(fileName).Max(line => line.Length);

Console.WriteLine($"Длина самой длинной строки: {maxLineLength}");

string maxLengthLine = File.ReadAllLines(fileName).FirstOrDefault(line => line.Length == maxLineLength); if (maxLengthLine != null)

{Console.WriteLine($"Самая длинная строка: {maxLengthLine}");}

Console.WriteLine(); Console.WriteLine("Введите букву для поиска строк: ");

char letter = Convert.ToChar(Console.Read());

Console.WriteLine($"Строки, начинающиеся на букву '{letter}':");

lines = File.ReadAllLines(fileName).ToList(); foreach (string line in lines)

{if (line.StartsWith(letter.ToString(), StringComparison.OrdinalIgnoreCase))

{Console.WriteLine(line);}} Console.WriteLine();

string reverseFilePath = @"./reverse.txt"; List<string> reversedLines = File.ReadAllLines(fileName).Reverse().ToList(); File.WriteAllLines(reverseFilePath, reversedLines);

Console.WriteLine($"Строки из файла записаны в файл {reverseFilePath} в обратном порядке.");

Таблица 16.3 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| и | Содержимое файла:  Волшебная строка  Обычная строка  Самая интересная строка  Спокойная строка  Интригующая строка  Количество строк в файле: 5  Количество символов в каждой строке:  Волшебная строка - 16 символов  Обычная строка - 14 символов  Самая интересная строка - 23 символов  Спокойная строка - 16 символов  Интригующая строка - 18 символов  Удалена последняя строка из файла file.txt, результат записан в файл new\_file.txt  Строки с 1 по 2:  Волшебная строка  Обычная строка  Длина самой длинной строки: 23  Самая длинная строка: Самая интересная строка  Введите букву для поиска строк: и  Строки, начинающиеся на букву 'и':  Интригующая строка |

Источник: собственная разработка

Анализ результатов:

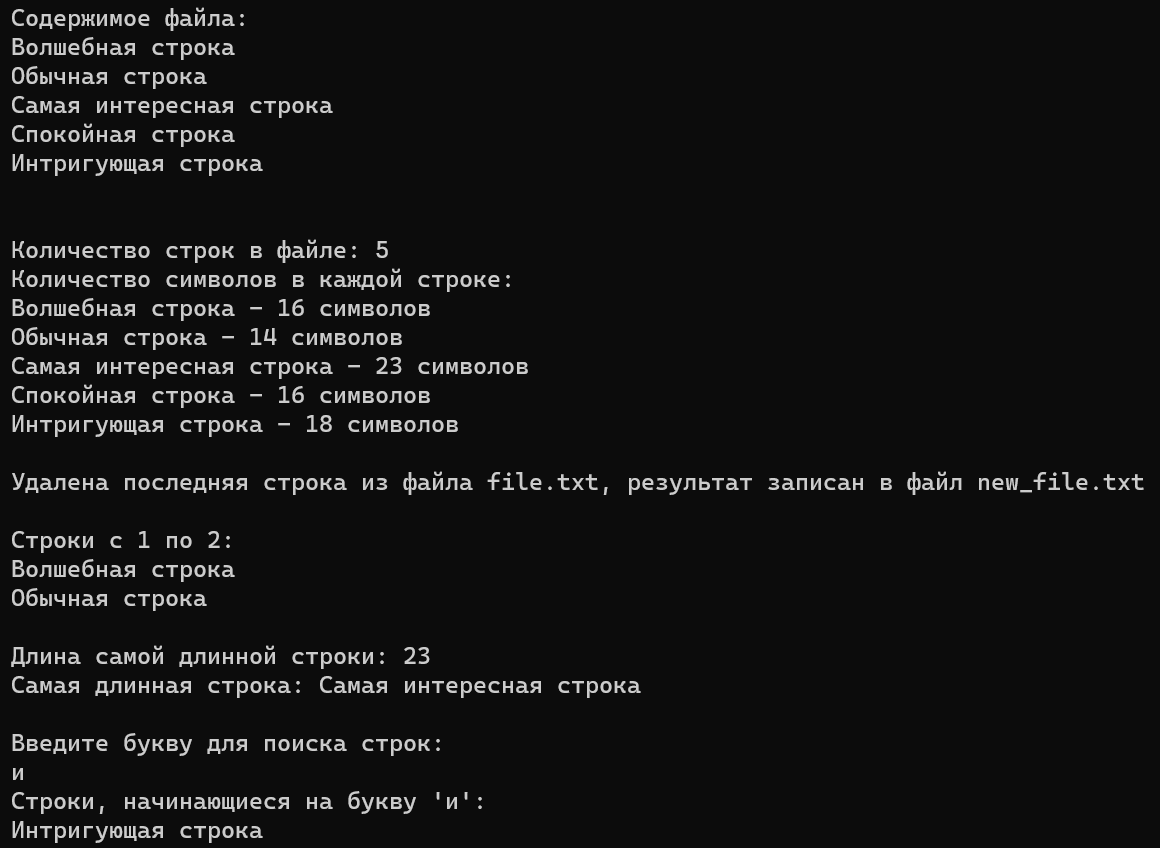


Рисунок 16.3 – Результат работы программы

Источник: собственная разработка

Задание 4. Имеется текстовый файл. Все четные строки этого файла записать во второй файл, а нечетные – в третий файл. Порядок следования строк сохраняется.

Листинг программы:

string inputFileName = "first.txt"; string evenFileName = "second.txt";

string oddFileName = "third.txt";

string[] allLines = File.ReadAllLines(inputFileName);

List<string> evenLines = new List<string>();

List<string> oddLines = new List<string>();

for (int i = 0; i < allLines.Length; i++){if (i % 2 == 0){

evenLines.Add(allLines[i]);} else {oddLines.Add(allLines[i]);}}

File.WriteAllLines(evenFileName, evenLines);

File.WriteAllLines(oddFileName, oddLines);

Таблица 16.4 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Первая нечётная строка  Вторая чётная строка  Третья нечётная строка  Четвёртая чётная строка | Содержимое файла Second.txt:  Первая нечётная строка  Третья нечётная строка  Содержимое файла Third.txt:  Вторая чётная строка  Четвёртая чётная строка |

Источник: собственная разработка

Анализ результатов:

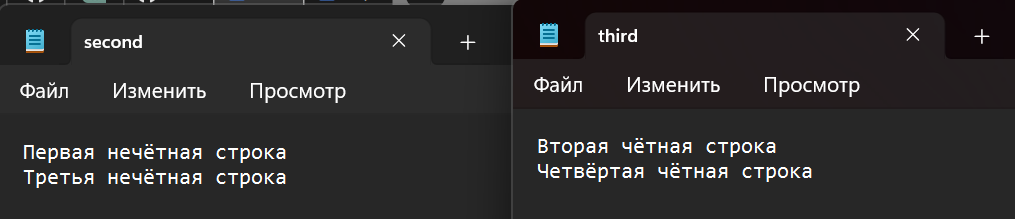


Рисунок 16.4 – Результат работы программы

Источник: собственная разработка