**16 «РАБОТА С ФАЙЛАМИ».**

Задание 1. Дан файл f, компонентами которого являются натуральные числа. Получить в файле g все компоненты файла f, делящиеся на 3 и не делящиеся на 7.

Листинг программы:

using System;

using System.IO;

class Program

{

static void Main()

{

string inputFile = "f.txt";

string outputFile = "g.txt";

try

{

using (StreamReader reader = new StreamReader(inputFile))

using (StreamWriter writer = new StreamWriter(outputFile))

{

string line;

while ((line = reader.ReadLine()) != null)

{

if (int.TryParse(line, out int number))

{

if (number % 3 == 0 && number % 7 != 0)

{

writer.WriteLine(number);

}

}

else

{

Console.WriteLine($"Ошибка: Не удалось преобразовать строку '{line}' в число.");

}

}

}

Console.WriteLine("Результат записан в файл g.txt");

}

catch (IOException ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 21  14  9  3  28  17  6  35  12 | 9  3  6  12 |

Анализ результатов:

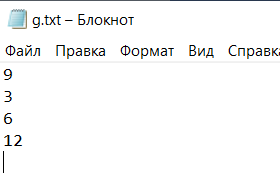


Рисунок 1.1 – Результат работы программы

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

Листинг программы:

using System;

using System.IO;

class Program

{

static void Main(string[] args)

{

string folderPath = "New\_folder";

try

{

Directory.CreateDirectory(folderPath);

Console.WriteLine("Папка успешно создана.");

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при создании папки: {ex.Message}");

}

}

}

Таблица 1.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Папка успешно создана. |

Анализ результатов:

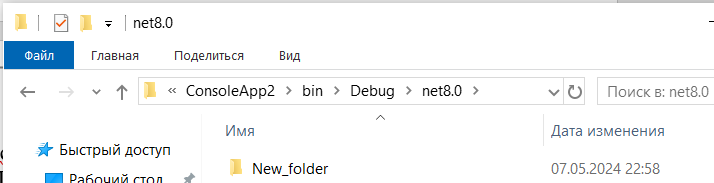


Рисунок 1.2 – Результат работы программы

Задание 3. Создать текстовый файл, в который записать 5 строк различной длины.

Для данного файла выполнить следующие действия:

a) вывести весь файл на экран;

b) подсчитать количество строк;

c) подсчитать количество символов в каждой строке;

d) удалить последнюю строку из файла, результат записать в новый

файл;

e) вывести на экран строки с s1 по s2;

f) найти длину самой длинной строки и вывести ее на экран;

g) вывести на экран все строки начинающиеся с заданной буквы;

h) переписать его строки в другой файл, порядок строк во втором

файле должен быть обратным по отношению к порядку строк в заданном

файле.

Листинг программы:

using System;

using System.IO;

using System.Linq;

class Program

{

static void Main(string[] args)

{

string fileName = "text\_file.txt";

// Создаем текстовый файл и записываем в него 5 строк различной длины

WriteTextToFile(fileName);

// a) Выводим весь файл на экран

Console.WriteLine("Содержимое файла:");

DisplayFileContent(fileName);

// b) Подсчитываем количество строк

int lineCount = CountLines(fileName);

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

// c) Подсчитываем количество символов в каждой строке

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

CountCharactersInEachLine(fileName);

// d) Удаляем последнюю строку из файла и записываем результат в новый файл

string newFileName = "new\_text\_file.txt";

RemoveLastLine(fileName, newFileName);

Console.WriteLine($"Последняя строка удалена из файла '{fileName}'. Результат записан в файл '{newFileName}'.");

// e) Выводим на экран строки с s1 по s2

int s1 = 2; // начиная со второй строки (индексация с нуля)

int s2 = 4; // заканчивая четвертой строкой

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

DisplayLinesInRange(fileName, s1, s2);

// f) Находим длину самой длинной строки и выводим ее на экран

int maxLength = FindLongestLine(fileName);

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

// g) Выводим на экран все строки, начинающиеся с заданной буквы

char startingLetter = 'T';

Console.WriteLine($"Строки, начинающиеся с буквы '{startingLetter}':");

DisplayLinesStartingWithLetter(fileName, startingLetter);

// h) Переписываем строки файла в другой файл в обратном порядке

string reversedFileName = "reversed\_text\_file.txt";

ReverseWriteToFile(fileName, reversedFileName);

Console.WriteLine($"Строки файла '{fileName}' переписаны в файл '{reversedFileName}' в обратном порядке.");

}

// Создание текстового файла и запись в него 5 строк различной длины

static void WriteTextToFile(string fileName)

{

using (StreamWriter writer = new StreamWriter(fileName))

{

writer.WriteLine("Short line");

writer.WriteLine("A bit longer line");

writer.WriteLine("This line is even longer");

writer.WriteLine("Another long line with some additional text");

writer.WriteLine("This is the longest line among all lines in this file");

}

}

// Вывод содержимого файла на экран

static void DisplayFileContent(string fileName)

{

using (StreamReader reader = new StreamReader(fileName))

{

Console.WriteLine(reader.ReadToEnd());

}

}

// Подсчет количества строк в файле

static int CountLines(string fileName)

{

return File.ReadAllLines(fileName).Length;

}

// Подсчет количества символов в каждой строке файла

static void CountCharactersInEachLine(string fileName)

{

string[] lines = File.ReadAllLines(fileName);

foreach (string line in lines)

{

Console.WriteLine($"Строка \"{line}\" содержит {line.Length} символов.");

}

}

// Удаление последней строки из файла и запись результата в новый файл

static void RemoveLastLine(string fileName, string newFileName)

{

string[] lines = File.ReadAllLines(fileName);

File.WriteAllLines(newFileName, lines.Take(lines.Length - 1));

}

// Вывод строк с s1 по s2 в файле

static void DisplayLinesInRange(string fileName, int s1, int s2)

{

string[] lines = File.ReadAllLines(fileName);

for (int i = s1; i <= s2; i++)

{

Console.WriteLine(lines[i]);

}

}

// Поиск длины самой длинной строки в файле

static int FindLongestLine(string fileName)

{

string[] lines = File.ReadAllLines(fileName);

int maxLength = lines.Max(line => line.Length);

return maxLength;

}

// Вывод строк файла, начинающихся с заданной буквы

static void DisplayLinesStartingWithLetter(string fileName, char letter)

{

string[] lines = File.ReadAllLines(fileName);

foreach (string line in lines)

{

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

{

Console.WriteLine(line);

}

}

}

// Переписывание строк файла в другой файл в обратном порядке

static void ReverseWriteToFile(string fileName, string reversedFileName)

{

string[] lines = File.ReadAllLines(fileName);

Array.Reverse(lines);

File.WriteAllLines(reversedFileName, lines);

}

}

Таблица 1.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Содержимое файла:  Short line  A bit longer line  This line is even longer  Another long line with some additional text  This is the longest line among all lines in this file  Количество строк в файле: 5  Количество символов в каждой строке:  Строка "Short line" содержит 10 символов.  Строка "A bit longer line" содержит 17 символов.  Строка "This line is even longer" содержит 24 символов.  Строка "Another long line with some additional text" содержит 43 символов.  Строка "This is the longest line among all lines in this file" содержит 53 символов.  Последняя строка удалена из файла 'text\_file.txt'. Результат записан в файл 'new\_text\_file.txt'.  Строки с 3 по 5:  This line is even longer  Another long line with some additional text  This is the longest line among all lines in this file  Длина самой длинной строки: 53  Строки, начинающиеся с буквы 'T':  This line is even longer  This is the longest line among all lines in this file  Строки файла 'text\_file.txt' переписаны в файл 'reversed\_text\_file.txt' в обратном порядке. |

Анализ результатов:

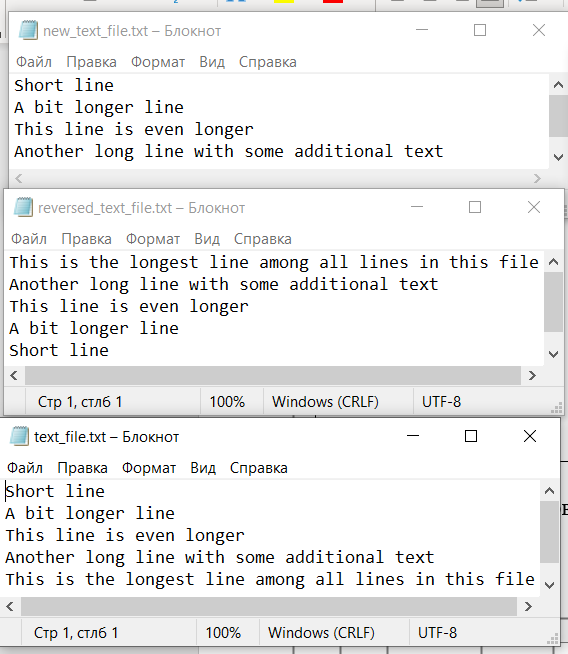


Рисунок 1.3 – Результат работы программы\

Задание 4. Имеется текстовый файл. Переписать в другой файл все его строки

с заменой в них символа 0 на символ 1 и наоборот.

Листинг программы:

using System;

using System.IO;

class Program

{

static void Main(string[] args)

{

string inputFileName = "input.txt";

string outputFileName = "output.txt";

ReverseSymbols(inputFileName, outputFileName);

Console.WriteLine($"Строки файла '{inputFileName}' переписаны с заменой символов и записаны в файл '{outputFileName}'.");

}

static void ReverseSymbols(string inputFileName, string outputFileName)

{

using (StreamReader reader = new StreamReader(inputFileName))

using (StreamWriter writer = new StreamWriter(outputFileName))

{

while (!reader.EndOfStream)

{

string line = reader.ReadLine();

string reversedLine = ReverseCharacters(line);

writer.WriteLine(reversedLine);

}

}

}

static string ReverseCharacters(string line)

{

char[] characters = line.ToCharArray();

for (int i = 0; i < characters.Length; i++)

{

if (characters[i] == '0')

characters[i] = '1';

else if (characters[i] == '1')

characters[i] = '0';

}

return new string(characters);

}

}

Таблица 1.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Строки файла 'input.txt' переписаны с заменой символов и записаны в файл 'output.txt'. |

Анализ результатов:

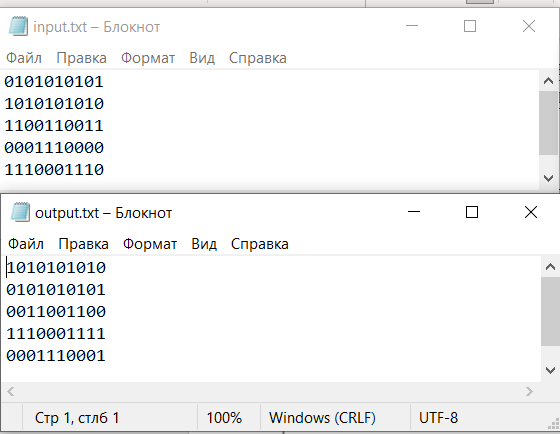


Рисунок 1.4 – Результат работы программы

Задание 5. Дана последовательность из n вещественных чисел. Необходимо:

 записать все числа в файл, затем вывести на экран все компоненты файла,

меньшие заданного числа.

 вывести на экран все положительные компоненты файла.

 вычислить среднее арифметическое компонентов файла, стоящих на

четных позициях.

Листинг программы:

using System;

using System.IO;

using System.Linq;

class Program

{

static void Main(string[] args)

{

double[] numbers = { 2.5, -1.3, 4.7, -0.8, 3.2, 1.6, -5.9, 7.1 };

string fileName = "numbers.txt";

WriteNumbersToFile(fileName, numbers);

double threshold = 2.0;

Console.WriteLine($"Числа файла меньше {threshold}:");

DisplayNumbersLessThan(fileName, threshold);

Console.WriteLine("Положительные числа файла:");

DisplayPositiveNumbers(fileName);

double average = CalculateEvenPositionAverage(fileName);

Console.WriteLine($"Среднее арифметическое чисел на четных позициях файла: {average}");

}

static void WriteNumbersToFile(string fileName, double[] numbers)

{

using (StreamWriter writer = new StreamWriter(fileName))

{

foreach (double number in numbers)

{

writer.WriteLine(number);

}

}

}

static void DisplayNumbersLessThan(string fileName, double threshold)

{

using (StreamReader reader = new StreamReader(fileName))

{

string line;

while ((line = reader.ReadLine()) != null)

{

double number = double.Parse(line);

if (number < threshold)

{

Console.WriteLine(number);

}

}

}

}

static void DisplayPositiveNumbers(string fileName)

{

using (StreamReader reader = new StreamReader(fileName))

{

string line;

while ((line = reader.ReadLine()) != null)

{

double number = double.Parse(line);

if (number > 0)

{

Console.WriteLine(number);

}

}

}

}

static double CalculateEvenPositionAverage(string fileName)

{

double sum = 0;

int count = 0;

using (StreamReader reader = new StreamReader(fileName))

{

string line;

int position = 0;

while ((line = reader.ReadLine()) != null)

{

if (position % 2 == 0)

{

double number = double.Parse(line);

sum += number;

count++;

}

position++;

}

}

if (count == 0)

{

return 0;

}

else

{

return sum / count;

}

}

}

Таблица 1.5 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Числа файла меньше 2:  -1,3  -0,8  1,6  -5,9  Положительные числа файла:  2,5  4,7  3,2  1,6  7,1  Среднее арифметическое чисел на четных позициях файла: 1,125 |

Анализ результатов:

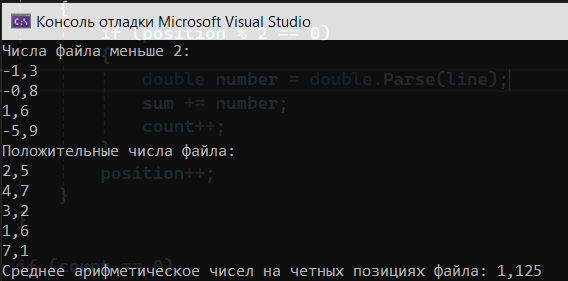


Рисунок 1.5 – Результат работы программы

Задание 6. Текстовый файл mark.txt содержит оценки учащихся класса за год в формате : Фамилия, оценки по 10 предметам, например:

Иванов 5 8 6 4 9 7 6 8 9 10

Петров 10 4 9 6 7 8 9 9 9 7

Сидоров 1 2 3 4 5 6 7 8 9 10 и т.д.

Файл Subjects.txt содержит названия предметов в том порядке, в котором

перечислены оценки, например:

1. Русский язык

2. Русская литература

3. Математика

4. Физика и.т.д до 10

Необходимо:

 вывести на экран фамилии отличников;

 определить среднюю оценку класса по каждому предмету;

 сформировать новый файл и записать туда фамилии учеников и их средний

бал за год, расположив фамилии по убыванию среднего балла.

Листинг программы:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

class Program

{

static void Main(string[] args)

{

string marksFileName = "mark.txt";

string subjectsFileName = "Subjects.txt";

List<Student> students = ReadStudentsFromFile(marksFileName);

List<string> subjects = ReadSubjectsFromFile(subjectsFileName);

Console.WriteLine("Отличники:");

DisplayExcellentStudents(students);

Dictionary<string, double> averageScores = CalculateAverageScores(students);

Console.WriteLine("\nСредняя оценка класса по каждому предмету:");

foreach (var kvp in averageScores)

{

Console.WriteLine($"{subjects[int.Parse(kvp.Key) - 1]}: {kvp.Value}");

}

string outputFileName = "average\_scores.txt";

WriteStudentsAverageToFile(students, outputFileName);

Console.WriteLine($"\nДанные о средних баллах студентов записаны в файл '{outputFileName}'.");

}

class Student

{

public string Name { get; set; }

public List<int> Marks { get; set; }

public double AverageScore => Marks.Average();

}

static List<Student> ReadStudentsFromFile(string fileName)

{

List<Student> students = new List<Student>();

using (StreamReader reader = new StreamReader(fileName))

{

string line;

while ((line = reader.ReadLine()) != null)

{

string[] parts = line.Split(' ');

Student student = new Student

{

Name = parts[0],

Marks = parts.Skip(1).Select(int.Parse).ToList()

};

students.Add(student);

}

}

return students;

}

static List<string> ReadSubjectsFromFile(string fileName)

{

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

using (StreamReader reader = new StreamReader(fileName))

{

string line;

while ((line = reader.ReadLine()) != null)

{

subjects.Add(line);

}

}

return subjects;

}

static void DisplayExcellentStudents(List<Student> students)

{

foreach (var student in students)

{

if (student.Marks.All(mark => mark >= 9))

{

Console.WriteLine(student.Name);

}

}

}

static Dictionary<string, double> CalculateAverageScores(List<Student> students)

{

Dictionary<string, List<int>> subjectScores = new Dictionary<string, List<int>>();

foreach (var student in students)

{

for (int i = 0; i < student.Marks.Count; i++)

{

string subjectIndex = (i + 1).ToString();

if (!subjectScores.ContainsKey(subjectIndex))

{

subjectScores[subjectIndex] = new List<int>();

}

subjectScores[subjectIndex].Add(student.Marks[i]);

}

}

Dictionary<string, double> averageScores = new Dictionary<string, double>();

foreach (var kvp in subjectScores)

{

averageScores[kvp.Key] = kvp.Value.Average();

}

return averageScores;

}

static void WriteStudentsAverageToFile(List<Student> students, string fileName)

{

using (StreamWriter writer = new StreamWriter(fileName))

{

foreach (var student in students.OrderByDescending(s => s.AverageScore))

{

writer.WriteLine($"{student.Name}: {student.AverageScore}");

}

}

}

}

Таблица 1.6 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Отличники:  Средняя оценка класса по каждому предмету:  Русский язык: 5,333333333333333  Литература: 4,666666666666667  Математика: 6  Физика: 4,666666666666667  Химия: 7  История: 7  География: 7,333333333333333  Биология: 8,333333333333334  Английский язык: 9  Информатика: 9  Данные о средних баллах студентов записаны в файл 'average\_scores.txt'. |

Анализ результатов:

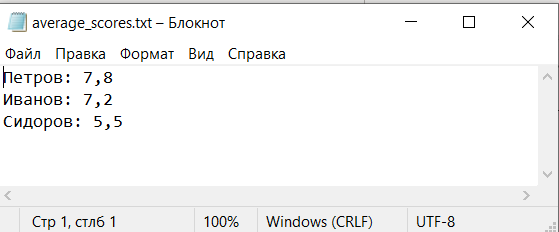


Рисунок 1.6 – Результат работы программы

Задание 7. Ввывести список всехфайлов на локальном диске(дасках).

Создать на диске D(С,V) каталог с именем Exmple\_36tp. Скапировать туда 3 разных файла с другого каталога. Поменять аттрибуты скопированых файлов на Скрытый, вместо них создать файлы ссылок.

Листинг программы:

using System;

using System.IO;

class Program

{

static void Main(string[] args)

{

string[] allFiles = Directory.GetFiles("C:\\");

string exampleDir = @"D:\Example\_36tp";

Directory.CreateDirectory(exampleDir);

string sourceDir = @"C:\SourceDirectory";

string[] filesToCopy = Directory.GetFiles(sourceDir);

foreach (string file in filesToCopy)

{

string fileName = Path.GetFileName(file);

string destFile = Path.Combine(exampleDir, fileName);

File.Copy(file, destFile, true);

}

foreach (string file in filesToCopy)

{

string fileName = Path.GetFileName(file);

string destFile = Path.Combine(exampleDir, fileName);

File.SetAttributes(destFile, FileAttributes.Hidden);

string linkFilePath = Path.Combine(exampleDir, "Link\_" + fileName);

CreateSymbolicLink(linkFilePath, destFile, SymbolicLink.File);

}

Console.WriteLine("Операции завершены успешно.");

Console.ReadLine();

}

[System.Runtime.InteropServices.DllImport("kernel32.dll")]

public static extern bool CreateSymbolicLink(string lpSymlinkFileName, string lpTargetFileName, SymbolicLink dwFlags);

public enum SymbolicLink

{

File = 0,

Directory = 1

}

}

Таблица 1.7 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Операции завершены успешно. |

Анализ результатов:

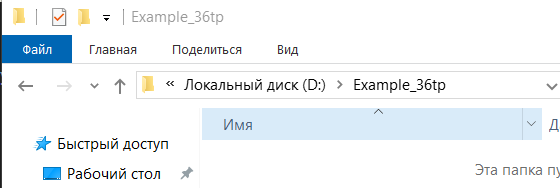


Рисунок 1.7 – Результат работы программы