1. **Работа с файлами**

Задание 1. Компонентами файла f являются действительные числа. Найти сумму наибольшего и наименьшего значений компонент.

Листинг программы:

namespace Task\_1

{

class Program

{

static void Main(string[] args)

{

string path = "..\\..\\..\\f.txt";

FileStream file = new FileStream(@path, FileMode.Open);

StreamReader reader = new StreamReader(file);

string s;

int n = 0;

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

{

n++;

}

reader.Close();

Console.WriteLine($"Сумма наибольшего и наименьшего значений компонент: " +

$"{SumMaxMinFromArray(FillArrayFromFile(n, path))}");

}

public static double SumMaxMinFromArray(double[] array)

{

return array.Max() + array.Min();

}

public static double[] FillArrayFromFile(int n, string path)

{

FileStream file1 = new FileStream(@path, FileMode.Open);

StreamReader reader1 = new StreamReader(file1);

double[] array = new double[n];

for (int i = 0; i < n; i++)

{

array[i] = double.Parse(reader1.ReadLine());

}

return array;

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Путь к файлу | 2337,65655 |

Анализ результатов:



Рисунок 1.1 – Результат работы программы

Источник: собственная разработка

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

Листинг программы:

namespace Task\_2

{

class Program

{

public static void Main(string[] args)

{

Console.Write("Введите название папки: ");

string nameFolder = Console.ReadLine();

string path = "..\\..\\..\\";

Directory.CreateDirectory(path + nameFolder);

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| NewFolder | В папке программы Task\_2 создастся папка NewFolder |

Анализ результатов:



Рисунок 2.1 – 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:

namespace Task\_3

{

class Program

{

public static void Main(string[] args)

{

try

{

string path = "..\\..\\..\\Text.txt";

Console.WriteLine("Содержимое файла: ");

OutputFile(path);

Console.WriteLine("Кол-во строк в файле: ");

Console.WriteLine(FileRowCount(path));

Console.WriteLine("Кол-во символов в каждой строке файла: ");

CharCountRowInFile(path);

Console.WriteLine("Файл с удаленной последней строкой: ");

OutputFile(DeleteLastLine(path));

Console.Write("Введите с какой строки вывести сообщение: ");

int s1 = int.Parse(Console.ReadLine());

Console.Write("Введите по какую строку вывести сообщение: ");

int s2 = int.Parse(Console.ReadLine());

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

OutText(s1, s2, path);

Console.WriteLine("Самая длинная строка в файле: ");

OutLongRow(path);

Console.Write("Введите букву: ");

char alf = char.Parse(Console.ReadLine());

Console.WriteLine("Все строки начинающиеся с заданной буквы: ");

OutRowStartAlf(alf, path);

Console.WriteLine("Реверсия строк: ");

OutputFile(ReversStringInFile(path));

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static string ReversStringInFile(string path)

{

string newPath = "..\\..\\..\\reversText.txt";

string[] textFile = File.ReadAllLines(path);

StreamWriter reversFile = new StreamWriter(new FileStream(newPath, FileMode.Create, FileAccess.Write));

for (int i = textFile.Length; i > 0; i--)

reversFile.WriteLine(textFile[i - 1]);

reversFile.Close();

return newPath;

}

public static void OutRowStartAlf(char alf, string path)

{

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

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

{

string line = lines[i];

if (line[0] == alf)

{

Console.WriteLine(lines[i]);

}

}

}

public static void OutLongRow(string path)

{

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

int max = 0;

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

{

if (lines[i].Length > lines[max].Length)

{

max = i;

}

}

Console.WriteLine(lines[max]);

}

public static void OutText(int s1, int s2, string path)

{

FileStream file = new FileStream(@path, FileMode.Open);

StreamReader reader = new StreamReader(file);

string line = "";

int i = 1;

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

{

if (i >= s1 && i <= s2)

Console.WriteLine(line);

i++;

}

reader.Close();

}

public static string DeleteLastLine(string path)

{

string newPath = "..\\..\\..\\newText.txt";

var lines = File.ReadAllLines(path);

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

return newPath;

}

public static void CharCountRowInFile(string path)

{

FileStream file = new FileStream(@path, FileMode.Open);

StreamReader reader = new StreamReader(file);

string line = "";

int i = 1;

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

{

Console.WriteLine($"Строка({i}): {line.Length}");

i++;

}

reader.Close();

}

public static int FileRowCount(string path)

{

return File.ReadAllLines(path).Length;

}

public static void OutputFile(string path)

{

FileStream file = new FileStream(@path, FileMode.Open);

StreamReader reader = new StreamReader(file);

string line = "";

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

{

Console.WriteLine(line);

}

reader.Close();

}

}

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 5 п | Операции с текстом файла |

Анализ результатов:

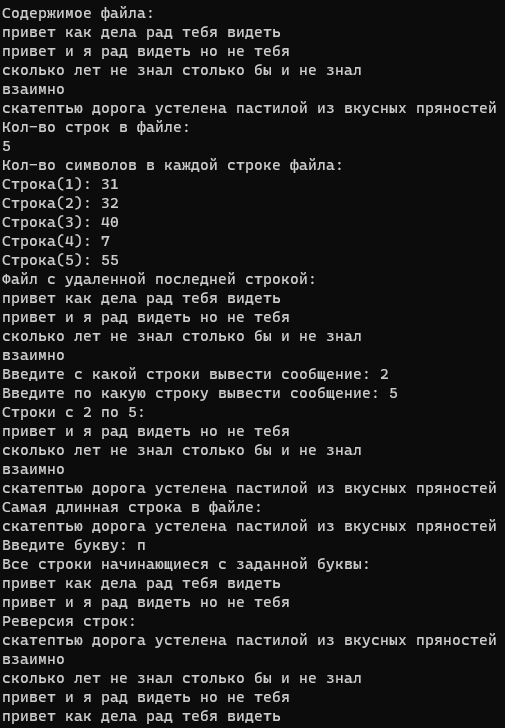


Рисунок 3.1 – Результат работы программы

Источник: собственная разработка

Задание 4. Имеются два текстовых файла с одинаковым числом строк. Выяснить совпадают ли их строки. Если нет, то получить номер первой строки, в которой эти файлы отличаются друг от друга.

Листинг программы:

namespace Task\_4

{

class Program

{

public static void Main(string[] args)

{

string pathOne = "..\\..\\..\\TextOne.txt";

string pathTwo = "..\\..\\..\\TextTwo.txt";

FileStream fileOne = new FileStream(@pathOne, FileMode.Open);

StreamReader readerOne = new StreamReader(fileOne);

FileStream fileTwo = new FileStream(@pathTwo, FileMode.Open);

StreamReader readerTwo = new StreamReader(fileTwo);

string one, two = null;

int i = 1;

while ((one = readerOne.ReadLine()) != null && (two = readerTwo.ReadLine()) != null)

{

if (one != two)

{

Console.WriteLine($"Строка № {i}");

break;

}

i++;

}

fileOne.Close();

fileTwo.Close();

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  |  |

Анализ результатов:



Рисунок 4.1 – Результат работы программы

Источник: собственная разработка

Задание 5. Пусть в файлах f1 и f2 целые числа упорядочены по возрастанию (файлы предварительно создайте программно). Создайте файл f3, объединив в нем компоненты файлов f1 и f2 и сохранив упорядоченность.

Листинг программы:

namespace Task\_5

{

class Program

{

public static void Main(string[] args)

{

string fileNaameOne = "f1.txt";

string fileNaameTwo = "f2.txt";

string fileNaameThree = "f3.txt";

Console.WriteLine($"Файл {fileNaameOne}");

string pathOne = CreateNewFile(fileNaameOne);

SetInFile(pathOne);

OutputFile(pathOne);

Console.WriteLine($"Файл {fileNaameTwo}");

string pathTwo = CreateNewFile(fileNaameTwo);

SetInFile(pathTwo);

OutputFile(pathTwo);

Console.WriteLine($"Файл {fileNaameThree}");

string pathThree = CreateNewFile(fileNaameThree);

Generalization(pathOne, pathTwo, pathThree);

SortedFile(pathThree);

OutputFile(pathThree);

}

public static void Generalization(string pathOne, string pathTwo, string pathWriter)

{

string[] file1 = File.ReadAllLines(pathOne);

string[] file2 = File.ReadAllLines(pathTwo);

StreamWriter writer = File.CreateText(pathWriter);

int lineNum = 0;

while (lineNum < file1.Length || lineNum < file2.Length)

{

if (lineNum < file1.Length)

writer.WriteLine(file1[lineNum]);

if (lineNum < file2.Length)

writer.WriteLine(file2[lineNum]);

lineNum++;

}

writer.Close();

}

public static string CreateNewFile(string fileName)

{

string path = @"..\..\..\..\" + fileName;

FileStream file = new FileStream(path, FileMode.Create, FileAccess.Write);

file.Close();

return path;

}

public static List<int> GenerationIntegerNumbers()

{

Random rnd = new Random();

List<int> array = new List<int>();

for (int i = 0; i < 10; i++)

{

array.Add(rnd.Next(100));

}

array.Sort();

return array;

}

public static void SetInFile(string path)

{

FileStream file = new FileStream(@path, FileMode.Open);

StreamWriter writer = new StreamWriter(file);

List<int> array = GenerationIntegerNumbers();

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

writer.WriteLine(array[i]);

writer.Close();

}

public static void OutputFile(string path)

{

FileStream file = new FileStream(@path, FileMode.Open);

StreamReader reader = new StreamReader(file);

string line = "";

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

{

Console.Write(line + " ");

}

Console.WriteLine();

reader.Close();

}

public static void SortedFile(string path)

{

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

List<int> list = lines.Select(int.Parse).ToList();

list.Sort();

StreamWriter writer = new StreamWriter(path);

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

writer.WriteLine(list[i]);

writer.Close();

}

}

}

Таблица 5.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  |  |

Анализ результатов:

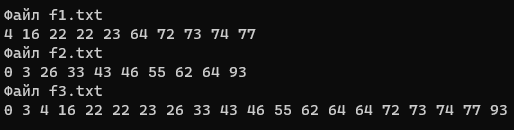


Рисунок 5.1 – Результат работы программы

Источник: собственная разработка

Задание 6. Пусть матрица A целых чисел размером 100х100 записана по строкам в текстовый файл.

Необходимо:

определить, является ли она единичной;

определить, есть ли в матрице нулевые строки;

определить, упорядочены ли значения m-го столбца по возрастанию (m вводится с клавиатуры);

запишите в новый файл m-ый столбец матрицы;

создайте новый файл, в который запишите все ненулевые строки матрицы с указанием их номера.

Листинг программы:

namespace Task\_6

{

class Program

{

public static int size = 100;

public static void Main(string[] args)

{

string fileNaameOne = "f1.txt";

string fileNaameTwo = "f2.txt";

string fileNaameThree = "f3.txt";

string path = CreateFileOne(fileNaameOne);

int[,] array = ReadFile(path);

Console.WriteLine(TheUnitMatrix(array) == 1 ? "Единичная" : TheUnitMatrix(array) == -1 ? "Не квадратная!" : "НЕ Единичная");

Console.WriteLine(TheZeroLine(array));

Console.Write("Введите номер столбца: ");

int row = int.Parse(Console.ReadLine());

Console.WriteLine($"Цифры в {row} строке: {OrderedNumbers(array, row)}");

string pathTwo = CreateFileTwo(fileNaameTwo, array, row);

string pathThree = CreateFileThree(fileNaameThree, array);

}

public static string CreateFileOne(string name)

{

Console.WriteLine($"Файл {name}");

string path = CreateNewFile(name);

SetInFile(path);

return path;

}

public static string CreateFileTwo(string name, int[,] array, int row)

{

Console.WriteLine($"Файл {name}");

string path = CreateNewFile(name);

FileStream file = new FileStream(@path, FileMode.Open);

StreamWriter writer = new StreamWriter(file);

for (int i = 0; i < array.GetLength(0); i++)

{

writer.Write(array[row, i] + " ");

}

writer.Close();

return path;

}

public static string CreateFileThree(string name, int[,] array)

{

Console.WriteLine($"Файл {name}");

string path = CreateNewFile(name);

FileStream file = new FileStream(@path, FileMode.Open);

StreamWriter writer = new StreamWriter(file);

for (int i = 0; i < array.GetLength(0); i++)

{

int b = 0;

for (int j = 0; j < array.GetLength(1); j++)

{

b += array[i, j];

}

if (b == 0)

{

for (int ii = 0; ii < array.GetLength(0); ii++)

{

writer.Write(array[i, ii] + " ");

}

}

}

writer.Close();

return path;

}

public static int[,] ReadFile(string path)

{

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

int[,] array = new int[size, size];

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

{

int j = 0;

for (int k = 0; k < lines[i].Length; k++)

{

if (lines[i][k] != ' ')

{

array[i, j] = int.Parse(Convert.ToString(lines[i][k]));

j++;

}

}

}

return array;

}

public static string OrderedNumbers(int[,] array, int row)

{

string text = "Упорядочены";

int ferst = array[row, 0];

for (int i = 0; i < array.GetLength(0); i++)

{

if (array[row, i] < ferst)

{

return "Не упорядочены";

}

ferst = array[row, i];

}

return text;

}

public static string TheZeroLine(int[,] array)

{

string text = "";

int b = 0;

for (int i = 0; i < array.GetLength(0); i++)

{

for (int j = 0; j < array.GetLength(1); j++)

{

b += array[i, j];

}

if (b == 0)

{

text = $"Нулевая строка {i}";

}

else

{

text = $"Нулевой строки нет";

}

}

return text;

}

public static int TheUnitMatrix(int[,] array)

{

if (array.GetLength(0) != array.GetLength(1)) return -1;

for (int i = 0; i < array.GetLength(0); i++)

{

for (int j = 0; j < array.GetLength(1); j++)

{

if (i == j && array[i, j] != 1) return 0;

if (i != j && array[i, j] != 0) return 0;

}

}

return 1;

}

public static string CreateNewFile(string fileName)

{

string path = @"..\..\..\..\" + fileName;

FileStream file = new FileStream(path, FileMode.Create, FileAccess.Write);

file.Close();

return path;

}

public static int[,] GenerationIntegerNumbers()

{

Random rnd = new Random();

int[,] array = new int[size, size];

for (int i = 0; i < array.GetLength(0); i++)

{

for (int j = 0; j < array.GetLength(1); j++)

{

array[i, j] = rnd.Next(2);

}

}

return array;

}

public static void SetInFile(string path)

{

FileStream file = new FileStream(@path, FileMode.Open);

StreamWriter writer = new StreamWriter(file);

int[,] array = GenerationIntegerNumbers();

for (int i = 0; i < array.GetLength(0); i++)

{

for (int j = 0; j < array.GetLength(1); j++)

{

writer.Write(array[i, j] + " ");

}

writer.WriteLine();

}

writer.Close();

}

}

}

Таблица 6.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 | Не упорядочены |

Анализ результатов:

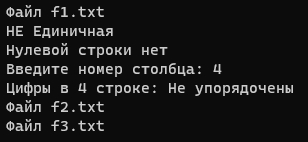


Рисунок 6.1 – Результат работы программы

Источник: собственная разработка

Задание 7. Вывести список всех файлов на локальном диске(дисках). Создать на диске D(С,V) каталог с именем Exmple\_30tp. Скопировать туда 3 разных файла с другого каталога. Поменять атрибуты скопированных файлов на Скрытый, вместо них создать файлы ссылок.

Листинг программы:

namespace Task\_7

{

class Program

{

public static void Main(string[] args)

{

string dir = @"D:";

Console.WriteLine($"Вывод каталогов с диска {dir}");

string[] dirs = GetDirs(dir);

OutDir(dirs);

string path = @"D:\Exmple\_30tp\";

CreateDir(path);

string pathCopy = @"D:\Автомобили\citroen\ситроен\";

CopyPDFFiles(pathCopy, path);

GetHidenAttributsFile(path);

string linkPath1 = @"D:\Exmple\_30tp\Citroen C5.pdf";

string linkPath2 = @"D:\Exmple\_30tp\citroen-c5-i (1).pdf";

LinkFile(path + "c5", linkPath1);

LinkFile(path + "nextC5", linkPath2);

Console.WriteLine($"Вывод каталогов с диска {dir}");

OutDir(GetDirs(dir));

}

public static string[] GetDirs(string dir)

{

return Directory.GetDirectories(dir);

}

public static void OutDir(string[] dirs)

{

Console.WriteLine(String.Join(Environment.NewLine, dirs));

}

public static void CreateDir(string path)

{

Directory.CreateDirectory(path);

}

public static void CopyPDFFiles(string path, string pathTo)

{

foreach (string file in Directory.GetFiles(path, "\*.pdf", SearchOption.AllDirectories))

{

File.Copy(path + Path.GetFileName(file), pathTo + Path.GetFileName(file));

}

}

public static void GetHidenAttributsFile(string path)

{

foreach (string file in Directory.GetFiles(path, "\*.pdf", SearchOption.AllDirectories))

{

File.SetAttributes(file, FileAttributes.Hidden);

}

}

public static void LinkFile(string path, string linkPath)

{

File.CreateSymbolicLink(path, linkPath);

}

}

}

Таблица 7.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  |  |

Анализ результатов:

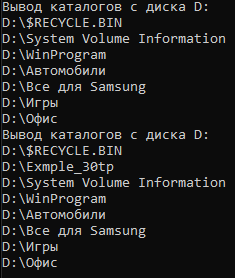


Рисунок 7.1 – Результат работы программы

Источник: собственная разработка