**16 РАБОТА С ФАЙЛАМИ**

Задание №1. Ввести числа в файл. Найти минимум и посчитать количество положительных элементов в файле.

Листинг программы:

using System.Text;

int[] mass = new int[25];

int countNeqative = 0;

Random rnd = new Random();

using (FileStream file = new FileStream(@"input.txt", FileMode.OpenOrCreate))

{

using (StreamWriter \_sw = new StreamWriter(file, Encoding.UTF8))

{

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

{

\_sw.WriteLine(rnd.Next(-100, 100));

}

}

}

using (StreamReader \_sr = new StreamReader(@"input.txt"))

{

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

{

mass[i] = int.Parse(\_sr.ReadLine());

}

}

int max = mass.Max();

for (int e = 0; e < mass.Length; e++)

{

if (mass[e] <= 0)

{

countNeqative++;

}

}

Console.WriteLine($"Максимум:{max}\nКоличество отрицательных цифр:{countNeqative}");

Таблица 16.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| input.txt | 94, 16 |

Анализ результатов представлен на рисунке 16.1.



Рисунок 16.1 – Результат работы программы

Источник: собственная разработка

Задание №2. С помощью метода CreateDirectory() создать папку “New\_folder”.

Листинг программы:

var directory = Directory.CreateDirectory(@"New\_folder");

Console.WriteLine("Ok");

Таблица 16.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | New\_Folder |

Анализ результатов представлен на рисунке 16.2.



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:

try

{

string path = @"file.txt";

using (StreamReader sr = new StreamReader(path))

{

PrintFile(sr);

}

string[] str = StringSplit(path);

Console.WriteLine(str.Length);

int[] arr = CharCount(str);

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

Console.WriteLine($"В {i} строке {arr[i]} символов");

DeleteLast(str);

Console.Write("Введите начальную строку: ");

int start = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите конечную строку: ");

int end = Convert.ToInt32(Console.ReadLine());

PrintFromRange(str, start, end);

StringMax(str);

Console.Write("Введите букву: ");

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

PrintFromLetter(str, letter);

PrintInFile(str);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

void PrintFile(StreamReader sr)

{

Console.WriteLine(sr.ReadToEnd());

}

string[] StringSplit(string path)

{

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

return str;

}

int[] CharCount(string[] str)

{

int[] arr = new int[str.Length];

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

arr[i] = str[i].Length;

return arr;

}

void DeleteLast(string[] str)

{

string[] st = new string[str.Length - 1];

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

st[i] = str[i];

File.WriteAllLines("out.txt", st);

}

void PrintFromRange(string[] str, int start, int end)

{

for (int i = start; i <= end; i++)

{

Console.WriteLine(str[i]);

}

}

void StringMax(string[] str)

{

Console.WriteLine(str.Select((x) => x.Length).Max());

}

void PrintFromLetter(string[] str, char letter)

{

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

if (str[i][0] == letter)

Console.WriteLine(str[i]);

}

void PrintInFile(string[] str)

{

Array.Reverse(str);

StreamWriter sw = new StreamWriter("out1.txt");

foreach (string st in str)

{

sw.WriteLine(st);

}

sw.Close();

}

Таблица 16.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| file.txt | dfsf, fsdfeqe, refwefs, dsadad, dfsfsdfs, out.txt, out1.txt |

Анализ результатов представлен на рисунке 16.3.

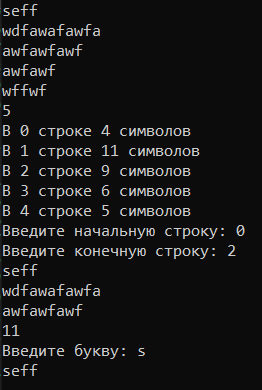


Рисунок 16.3 – Результат работы программы

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:

using System.Text;

var firstFilePath = @"firstD.txt";

var secondFilePath = @"secondD.txt";

if (!File.Exists(firstFilePath))

{

File.Create(firstFilePath);

}

if (!File.Exists(secondFilePath))

{

File.Create(secondFilePath);

}

var linesFromFile1 = File.ReadAllLines(firstFilePath);

var linesFromFile2 = File.ReadAllLines(secondFilePath);

if (linesFromFile1.Count() != linesFromFile2.Count())

{

Console.WriteLine("Файлы неодинаковые по длине");

}

else

{

Console.WriteLine("Файл одинаковые по длине");

var index = FindFirstIndex(linesFromFile1, linesFromFile2);

if (index == -1)

{

Console.WriteLine("Файлы одинаковые");

}

else

{

Console.WriteLine($"Файлы неодинаковые в строке {index}");

}

}

int FindFirstIndex(string[] linesFromFile1, string[] linesFromFile2)

{

var index = -1;

for (int i = 0; i < linesFromFile1.Count(); i++)

{

if (!linesFromFile1[i].Equals(linesFromFile2[i]))

{

return i + 1;

}

}

return index;

}

Таблица 16.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| firstD.txt  secondD.txt | Файл одинаковые по длине  Файлы неодинаковые в строке 2 |

Анализ результатов представлен на рисунке 16.4.



Рисунок 16.4 – Результат работы программы

Источник: собственная разработка

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.

вычсилить среднее арифметическое компонентов файла, стоящих на четных позициях.

Листинг программы:

class Program

{

static void ShowMenu()

{

Console.WriteLine();

Console.WriteLine("Меню:");

Console.WriteLine("0 - Выйти из программы");

Console.WriteLine("1 - Вывести на экран все компоненты файла,меньшие заданного числа.");

Console.WriteLine("2 - Вывести на экран все положительные компоненты файла");

Console.WriteLine("3 - Вычислить среднее арифметическое компонентов файла, стоящих\nна чётных позициях");

Console.WriteLine();

}

static void Task1(BinaryReader fromFile)

{

double userNum;

double item;

do

{

Console.Write("Введите опорное число: ");

try

{

userNum = double.Parse(Console.ReadLine());

break;

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Неверный формат данных.");

}

} while (true); // do 1

Console.WriteLine("Числа, меньшие заданного вами:");

try // 1

{

fromFile.BaseStream.Seek(0, SeekOrigin.Begin);

while (fromFile.BaseStream.Position < fromFile.BaseStream.Length)

{

item = fromFile.ReadDouble();

if (item < userNum) Console.WriteLine(item);

}

}

catch (IOException A)

{

Console.WriteLine("Внимание! Возникло исключение при чтении файла.\n" + A.Message);

}

}

static void Task2(BinaryReader fromFile)

{

double item;

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

try

{

fromFile.BaseStream.Seek(0, SeekOrigin.Begin);

while (fromFile.BaseStream.Position < fromFile.BaseStream.Length)

{

item = fromFile.ReadDouble();

if (item > 0.0) Console.WriteLine(item);

}

}

catch (IOException A)

{

Console.WriteLine("Внимание! Возникло исключение при чтении файла.\n" + A.Message);

}

// end try 1

} // Task2()

static void Task3(BinaryReader fromFile)

{

int i = 0;

double summ = 0.0;

double item; // Элемент файла

Console.WriteLine("Среднее арифметическое компонентов файла, стоящих на чётных позициях:");

try // 1

{

fromFile.BaseStream.Seek(0, SeekOrigin.Begin);

while (fromFile.BaseStream.Position < fromFile.BaseStream.Length)

{

item = fromFile.ReadDouble();

summ += item; // Суммируем

i++;

fromFile.BaseStream.Position += 8;

}

}

catch (IOException A)

{

Console.WriteLine("Внимание! Возникло исключение при чтении файла.\n" + A.Message);

}

Console.WriteLine("{0:F3}", summ / i);

}

static void Main(string[] args)

{

int i = 0;

int seqLength = 0;

double currNum;

double[] arr;

string fileName = "output.dat";

FileStream fs;

BinaryWriter dataOut;

BinaryReader dataIn;

Console.WriteLine("Ввод последовательности из n вещественных чисел:");

do

{

Console.Write("Укажите длину последовательности: ");

try

{

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

if (seqLength <= 0) throw new FormatException();

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Неверный формат данных.");

}

} while (seqLength <= 0); // do 1

arr = new double[seqLength];

do

{

Console.Write("[{0}]: ", i);

try

{

currNum = double.Parse(Console.ReadLine());

arr[i] = currNum;

i++;

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Неверный формат данных.");

}

} while (i < arr.Length); // do 2

Console.WriteLine("Полученная последовательность:");

for (i = 0; i < arr.Length; i++)

{

Console.WriteLine("{0}", arr[i]); // Здесь я вывожу не в строку, а в столбец. Мне решилось, что так нагляднее.

}

try

{

fs = new FileStream(fileName, FileMode.Create, FileAccess.Write);

dataOut = new BinaryWriter(fs);

}

catch (IOException A)

{

Console.WriteLine("Внимание! Возникло исключение при открытии файла {0}.", fileName);

Console.WriteLine(A.Message);

Console.WriteLine("Аварийное завершение программы.");

return;

}

try

{

for (i = 0; i < arr.Length; i++)

{

dataOut.Write(arr[i]);

}

Console.WriteLine("Файл записан.");

}

catch (IOException B)

{

Console.WriteLine("Внимание! Возникло исключение при записи файла.\n" + B.Message);

}

finally

{

dataOut.Close();

}

try

{

fs = new FileStream(fileName, FileMode.Open, FileAccess.Read);

dataIn = new BinaryReader(fs);

}

catch (IOException C)

{

Console.WriteLine("Внимание! Возникло исключение при открытии файла {0}.", fileName);

Console.WriteLine(C.Message);

Console.WriteLine("Аварийное завершение программы.");

return;

}

ShowMenu();

bool IsExit;

byte itemMenu;

try

{

do

{

IsExit = false;

try

{

Console.WriteLine("Введите номер пункта меню");

itemMenu = byte.Parse(Console.ReadLine());

switch (itemMenu)

{

case 0:

IsExit = true;

break;

case 1:

Task1(dataIn);

break;

case 2:

Task2(dataIn);

break;

case 3:

Task3(dataIn);

break;

default:

Console.WriteLine("Нет такого пункта меню.");

break;

}

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Неверный формат при вводе пункта меню.");

}

} while (!IsExit); // do 3

}

finally

{

dataIn.Close();

}

Console.WriteLine("Конец программы");

}

}

Таблица 16.5 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5, 6, 7, 8, 1, 2, 3 | out.dat, 1, 2, 5.333 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 16.5.

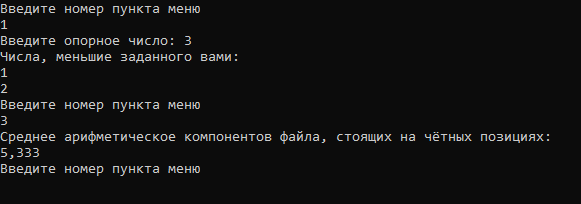


Рисунок 16.5 – Результат работы программы

Источник: собственная разработка

Задание №6. Дано предложение (вводится с клавиатуры). Слова в предложении разделяются одним или несколькими пробелами. В предложении могут встречаться знаки препинания и цифры.

Создать текстовый файл и записать в него все символы данного предложения, отличные от цифр. Вывести содержимое файла на экран.

Листинг программы:

string str = @"file.txt";

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

string[] Payments = File.ReadAllLines(str, Encoding.Default);

for (int x = 0; x < Payments.Length; x++)

{

Console.WriteLine($"Входная строка: {Payments[x]}");

var a = Payments[x].Split(new[] { ',', ' ', '-', '.' }).ToList();

char letter = 'l';

var res = Regex.Matches(Payments[x], "\\b" + letter + @"\w\*(-\w+)\*", RegexOptions.IgnoreCase).Cast<Match>().Select(i => i.Value).ToArray();

Console.WriteLine($"Слова, начинающиеся на символ {letter}:")

Console.WriteLine(string.Join("\r\n", res));

PaymentInf.Add(Payments[x]);

}

int length = 3;

Console.WriteLine($"Слова, имеющие длину {length} символа(ов):");

foreach (string R in PaymentInf)

{

string[] mas = R.Split(new Char[] { ' ', ',', '.' });

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

{

if (mas[i].Length == length)

{

Console.WriteLine(mas[i]);

}

}

}

Console.WriteLine($"Слова, начинающиеся и заканчивающиеся на одну и ту же букву:");

foreach (string R in PaymentInf)

{

string[] words = R.Split(' ', ',', '.', '-');

foreach (string word in words)

{

if (word.Length > 1)

{

if (word[0] == word[word.Length - 1])

{

Console.WriteLine(word);

}

}

}

}

Console.WriteLine($"Слова, начинающиеся на ту же букву, что и последнее слово:");

foreach (string R in PaymentInf)

{

string lastWord = R.Split(' ', ',', '.', '-').Last();

var matches = Regex.Matches(R, @"\b[\w]+\b", RegexOptions.Compiled);

var lastMatch = matches.Cast<Match>().Last();

string lastRemoved = R.Remove(lastMatch.Index, lastMatch.Length);

string[] words = lastRemoved.Split(' ', ',', '.', '-');

foreach (string word in words)

{

if (word.Length > 1)

{

if (word[0] == lastWord[0])

{

Console.WriteLine(word);

}

}

}

}

Таблица 16.6 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| file.txt | strings |

Источник: собственная разработка

Анализ результатов представлен на рисунке 16.6.

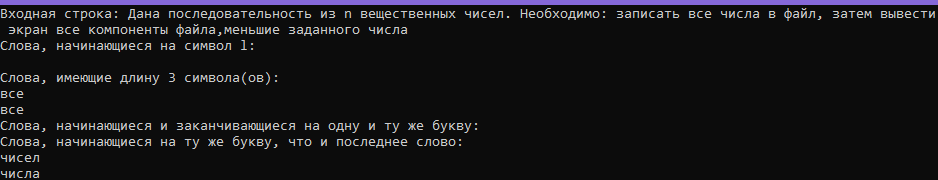


Рисунок 16.6 – Результат работы программы

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:

DirectoryInfo direct = new DirectoryInfo(@"G:\Колледж\Git\C-practice\16\16");

Console.WriteLine("No search pattern returns:");

foreach (var fi in direct.GetFiles())

{

Console.WriteLine(fi.Name);

}

Console.WriteLine();

string path = @"G:\Колледж\Git\C-practice\16\16";

try

{

if (Directory.Exists(path))

{

Console.WriteLine("That path exists already.");

var \_pathFile = Path.Combine(path, "file1.txt");

File.Copy(Path.Combine(@"G:\Колледж\Git\C-practice\16\16\z7\bin\Debug\net5.0", "file1.txt"), \_pathFile, true);

var \_pathFile2 = Path.Combine(path, "file2.txt");

File.Copy("file2.txt", \_pathFile2, true);

var \_pathFile3 = Path.Combine(path, "file3.txt");

File.Copy("file3.txt", \_pathFile3, true);

var \_file = new FileInfo(\_pathFile);

var \_file2 = new FileInfo(\_pathFile2);

var \_file3 = new FileInfo(\_pathFile3);

\_file.Attributes = FileAttributes.Hidden;

CreateShortcut("file1", path, \_pathFile);

\_file2.Attributes = FileAttributes.Hidden;

CreateShortcut("file2", path, \_pathFile2);

\_file3.Attributes = FileAttributes.Hidden;

CreateShortcut("file3", path, \_pathFile3);

return;

}

DirectoryInfo di1 = Directory.CreateDirectory(path);

Console.WriteLine("The directory was created successfully at {0}.", Directory.GetCreationTime(path));

var pathFile = Path.Combine(path, "file1.txt");

File.Copy("file1.txt", pathFile, true);

var pathFile2 = Path.Combine(path, "file2.txt");

File.Copy("file2.txt", pathFile2, true);

var pathFile3 = Path.Combine(path, "file3.txt");

File.Copy("file3.txt", pathFile3, true);

var file = new FileInfo(pathFile);

var file2 = new FileInfo(pathFile2);

var file3 = new FileInfo(pathFile3);

file.Attributes = FileAttributes.Hidden;

CreateShortcut("file1", path, pathFile);

file2.Attributes = FileAttributes.Hidden;

CreateShortcut("file2", path, pathFile2);

file3.Attributes = FileAttributes.Hidden;

CreateShortcut("file3", path, pathFile3);

}

catch (Exception e)

{

Console.WriteLine("The process failed: {0}", e.ToString());

}

finally { }

static void CreateShortcut(string shortcutName, string shortcutPath, string targetFileLocation)

{

string shortcutLocation = System.IO.Path.Combine(shortcutPath, shortcutName + ".lnk");

WshShell shell = new WshShell();

IWshShortcut shortcut = (IWshShortcut)shell.CreateShortcut(shortcutLocation);

shortcut.Description = "My shortcut description";

shortcut.IconLocation = @"c:\myicon.ico";

shortcut.TargetPath = targetFileLocation;

shortcut.Save();

}

Таблица 16.7 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| file1.txt, file2.txt, file3.txt | file1.lnk, file2.lnk, file3.lnk |

Источник: собственная разработка

Анализ результатов представлен на рисунке 16.7.

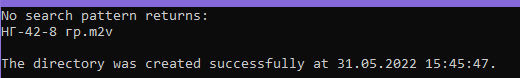


Рисунок 16.7 – Результат работы программы

Источник: собственная разработка