**Лабораторная работа №5**

**Титов Павел, 04-1, 17 вариант**

**Задание 1 (номер варианта = номер по списку % 13)**

Напишите функцию, которая находит количество слов в файле, начинающихся и заканчивающихся одной и той же литерой.

namespace L5

{

    public class Titov\_Pavel\_05\_01

    {

        public string Filepath

        {

            get; set;

        }

        public Titov\_Pavel\_05\_01(string filepath)

        {

            Filepath = filepath;

        }

        public int Task1()

        {

            using (StreamReader sr = new StreamReader(Filepath))

            {

                int count = 0;

                string[] words = sr.ReadToEnd().Split(" .,!?:\n".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

                foreach (string word in words)

                {

                    if (word != "" && char.ToLower(word[0]) == char.ToLower(word[word.Length - 1]))

                    {

                        count++;

                    }

                }

                return count;

            }

        }

    }

}

**Задание 2 (номер варианта = номер по списку % 13)**

Напишите функцию, которая переформатирует файл f так, что все строки в нем будут выровнены симметрично по центру (ширина строки равна константе Width=120).

namespace L5

{

    public class Titov\_Pavel\_05\_02

    {

        public string Filepath

        {

            get; set;

        }

        public Titov\_Pavel\_05\_02(string filepath)

        {

            Filepath = filepath;

        }

        public void Task2()

        {

            const int width = 120;

            string[] words;

            using (StreamReader sr = new StreamReader(Filepath))

            {

                words = sr.ReadToEnd().Split("\n".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

            }

            for (var i = 0; i < words.Length; i++)

            {

                string word = words[i];

                int leftPadding = (width - word.Length) / 2;

                int rightPadding = width - word.Length - leftPadding;

                words[i] = new string(' ', leftPadding) + word + new string(' ', rightPadding);

            }

            using (StreamWriter writer = new StreamWriter(Filepath, false))

            {

                foreach (var word in words)

                {

                    writer.WriteLine("|" + word + "|");

                }

            }

        }

    }

}

**Задание 3**

В файле хранится информация (в соответствии с вариантом). Для обработки информации создать класс.

17. Информация в БД  
- название груза  
- дата отгрузки  
- номер вагона  
- стоимость перевозки  
а) Получить список номеров вагонов, использовавшихся в первом квартале заданного года.  
б) Найти среднюю стоимость перевозки угля.  
с) Упорядочить данные по дате отгрузки.

using FileHelpers;

namespace L5

{

    [DelimitedRecord(",")]

    [IgnoreFirst()]

    public class Item

    {

        public string? Name;

        public int VagonNumber;

        [FieldConverter(ConverterKind.Date, "dd-MM-yyyy")]

        public DateTime Date;

        public int Price;

    }

    public class Working

    {

        public string Filepath

        {

            get; set;

        }

        public Working(string filepath)

        {

            Filepath = filepath;

        }

        public List<int> GetVagonsNumber()

        {

            var engine = new FileHelperEngine<Item>();

            var records = engine.ReadFile(Filepath);

            List<int> vagonNumbers = new();

            foreach (var item in records)

            {

                if ((item.Date.Month + 2) / 3 == 1)

                    vagonNumbers.Add(item.VagonNumber);

            }

            return vagonNumbers;

        }

        public double AverageCostItem()

        {

            var engine = new FileHelperEngine<Item>();

            var records = engine.ReadFile(Filepath);

            int average = 0;

            foreach (var item in records)

            {

                average += item.Price;

            }

            if (records.Length == 0)

                throw new DivideByZeroException("Деление на ноль невозможно");

            return average / records.Length;

        }

        public void SortedDataByDate()

        {

            var engine = new FileHelperEngine<Item>();

            var records = engine.ReadFile(Filepath);

            records.OrderBy(i => i.Date);

            foreach (var item in records)

                Console.WriteLine($"{item.Name} {item.VagonNumber} {item.Date} {item.Price}");

        }

public int GetMinCost()

{

var engine = new FileHelperEngine<Item>();

var records = engine.ReadFile(Filepath);

return records.Min(item => item.Price);

}

public int GetMaxCost()

{

var engine = new FileHelperEngine<Item>();

var records = engine.ReadFile(Filepath);

return records.Max(item => item.Price);

}

public int GetSumPrice()

{

var engine = new FileHelperEngine<Item>();

var records = engine.ReadFile(Filepath);

return records.Sum(item => item.Price);

}

    }

}

**Program.cs**

namespace L5

{

class Program

{

public static void Main(string[] args)

{

Titov\_Pavel\_05\_01 ft = new("file.txt");

Console.WriteLine(ft.Task1());

Titov\_Pavel\_05\_02 sd = new("file2.txt");

sd.Task2();

Working wr = new("file.csv");

foreach (var item in wr.GetVagonsNumber())

Console.Write($"{item} ");

Console.WriteLine();

Console.WriteLine(wr.AverageCostItem());

wr.SortedDataByDate();

Console.WriteLine($"Минимальная стоимость угля: {wr.GetMinCost()}");

Console.WriteLine($"Максимальная стоимость угля: {wr.GetMaxCost()}");

Console.WriteLine($"Общая стоимость угля: {wr.GetSumPrice()}");

}

}

}

**File.csv**

name,vagonNumber,date,*price*

Valera,423,25-05-2020,*1234569*

Valera,123,26-06-2020,*12345699*

Valera,213,21-01-2020,*12345*

Valera,243,23-03-2020,*1234567*

Valera,173,22-02-2020,*123456*

Valera,122,24-04-2020,*1234568*

**File.txt**

abp basdfsdfasdfsj

heh kek lol

kdkljsadfha pop dkfhjsfh f k lj ashadpophehke ksdklfjhasdklfj hasdfkashdfkjsdlkfhaskjdfhlsk

**File2.txt (до)**

dsadasd assdasd

dsadasd assdasd

dsadasd assdasd

dsadasd assdasd

**File2.txt (после)**

| dsadasd assdasd |

| dsadasd assdasd |

| dsadasd assdasd |

| dsadasd assdasd |

Работа программы

