|  |  |
| --- | --- |
| КГЭУ | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования** **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»** (ФГБОУ ВО «КГЭУ») |

**Институт цифровых технологий и экономики**

**Кафедра «Цифровые системы и модели»**

# **Дисциплина: Объектно-ориентированное программирование**

**Отчет о лабораторной работе № 1**

**Статические методы**

**Исполнитель:** Тазеев Р.Р.

**Группа:** ТРП-1-23

**Проверяющий:** Халидов А.А.

**Должность:** доцент кафедры ЦСМ

**КАЗАНЬ – 2024**

**Введение**

Лабораторная работа по объектно-ориентированному программированию на языке C# посвящена разработке и тестированию методов для обработки символьных данных и строк. В процессе работы студенты реализуют методы для выполнения различных операций, таких как арифметические вычисления с символами, преобразование текста в символьные массивы, удаление пробелов и работа с строками. Цель работы — освоение методов обработки данных и применение объектно-ориентированных подходов для решения задач программирования.

**Задание №1.**

Создать класс со статическими методами обработки символьных данных. Выполнить тестирование разработанных методов.

1. Метод принимает символы двух цифр и символ операции (+ - \* /), возвращает результат указанной операции над численными значениями символов.  
Код:

 public static int Proc(char d1, char d2, char op)

        {

            if (!char.IsDigit(d1) || !char.IsDigit(d2))

                throw new ArgumentException("Оба символа должны быть цифрами.");

            int n1 = d1 - '0';

            int n2 = d2 - '0';

            return op switch

            {

                '+' => n1 + n2,

                '-' => n1 - n2,

                '\*' => n1 \* n2,

                '/' => n2 != 0 ? n1 / n2 : throw new DivideByZeroException(),

                \_ => throw new ArgumentException("Неверная операция."),

            };

        }

Вывод:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

2. Метод выводит для каждого символа десятичный и шестнадцатеричный код:

Символы: Пробел, Точка, Запятая, Точка с запятой, Двоеточие.

Русские буквы: Ё, ё.

Код:

public static void PrintCharCodes()

        {

            char[] symbols = { ' ', '.', ',', ';', ':', 'Ё', 'ё' };

            foreach (char c in symbols)

            {

                int decimalCode = (int)c;

                string hexCode = decimalCode.ToString("X");

                Console.WriteLine($"Символ: {c}, Десятичный код: {decimalCode}, Шестнадцатеричный код: {hexCode}");

            }

        }

Вывод:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

3. Метод выводит для каждого символа десятичный и шестнадцатеричный код.

Цифры: 0 ... 9.

Латинские буквы: A ... Z; а ... z.

Русские буквы: А ... Я; а ... я.Код:

public static void PrintAllCharCodes()

        {

            // Массив символов

            char[] symbols = new char[62 + 66]; // 10 цифр + 26 латинских (A-Z) + 26 латинских (a-z) + 32 русских (А-Я) + 32 русских (а-я)

            int index = 0;

            // Добавляем цифры 0-9

            for (char c = '0'; c <= '9'; c++)

                symbols[index++] = c;

            // Добавляем латинские буквы A-Z

            for (char c = 'A'; c <= 'Z'; c++)

                symbols[index++] = c;

            // Добавляем латинские буквы a-z

            for (char c = 'a'; c <= 'z'; c++)

                symbols[index++] = c;

            // Добавляем русские буквы А-Я

            for (char c = 'А'; c <= 'Я'; c++)

                symbols[index++] = c;

            // Добавляем русские буквы а-я

            for (char c = 'а'; c <= 'я'; c++)

                symbols[index++] = c;

            // Выводим десятичные и шестнадцатеричные коды

            foreach (char c in symbols)

            {

                int decimalCode = (int)c;

                string hexCode = decimalCode.ToString("X");

                Console.WriteLine($"Символ: {c}, Десятичный код: {decimalCode}, Шестнадцатеричный код: {hexCode}");

            }

        }

Вывод:

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Изображение выглядит как снимок экрана, шаблон, ткань, искусство

Автоматически созданное описание

Изображение выглядит как снимок экрана, шаблон, Симметрия, ткань

Автоматически созданное описание

4. Метод преобразования произвольного текста в символьный массив.

ПРИМЕЧАНИЕ

В произвольном тексте должны быть всевозможные символы, например,

string s = @" s45g, df:\n = уам. Nи8т\tм Пт

Чdb; <>ум5ы). :Ядиь, vf3ь:9м( +

Djб6x2b 7; [] Ймд. Sfw 22я.";

Код:

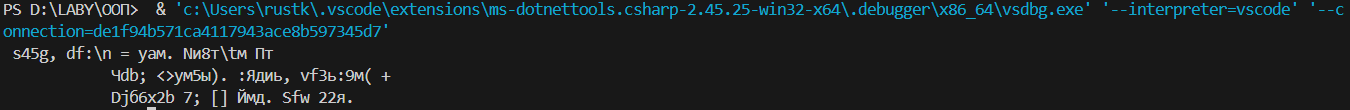
public static char[] ConvertToCharArray(string text)

        {

            return text.ToCharArray();

        }

Вывод:



5. Метод преобразования символьного массива в произвольный текст

Код:

public static string ConvertToText(char[] charArray)

        {

            return new string(charArray);

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

6. Метод для вывода массива символов.

Код:

public static void PrintCharArray(char[] charArray)

        {

            foreach (char c in charArray)

            {

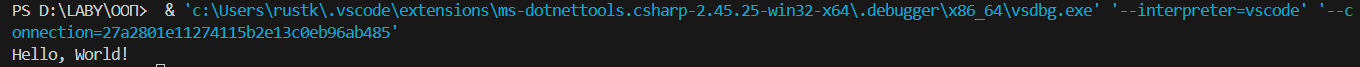
                Console.Write(c);

            }

            Console.WriteLine(); // Для переноса строки после вывода массива

        }

Вывод:



7. Метод для вывода кодов символов массива.

Код:

public static void PrintInputCharCodes(char[] charArray)

        {

            foreach (char c in charArray)

            {

                int decimalCode = (int)c;

                string hexCode = decimalCode.ToString("X");

                Console.WriteLine($"Символ: {c}, Десятичный код: {decimalCode}, Шестнадцатеричный код: {hexCode}");

            }

        }

Вывод: Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

8. Метод для определения в символьном массиве белых пробелов и вывода кодов их символов.

Код:

 public static void PrintWhitespaceCharCodes(char[] charArray)

        {

            foreach (char c in charArray)

            {

                if (char.IsWhiteSpace(c))

                {

                    int decimalCode = (int)c;

                    string hexCode = decimalCode.ToString("X");

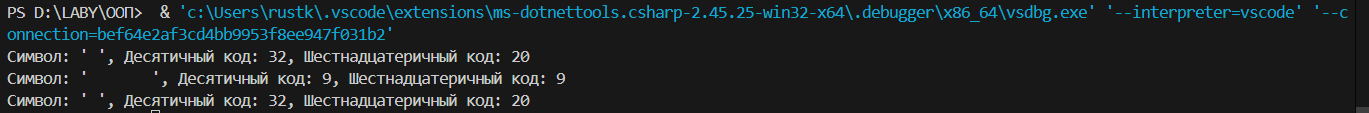
                    Console.WriteLine($"Символ: '{c}', Десятичный код: {decimalCode}, Шестнадцатеричный код: {hexCode}");

                }

            }

        }

Вывод:



9. Метод, который определяет в символьном массиве:

Количество букв. Количество цифр

Код:

public static void CountLettersAndDigits(char[] charArray, out int letterCount, out int digitCount)

        {

            letterCount = 0;

            digitCount = 0;

            foreach (char c in charArray)

            {

                if (char.IsLetter(c))

                {

                    letterCount++;

                }

                else if (char.IsDigit(c))

                {

                    digitCount++;

                }

            }

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

10. Метод для поиска заданного символа в символьном массиве

Код:

public static int FindChar(char[] charArray, char target)

        {

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

            {

                if (charArray[i] == target)

                {

                    return i; // Возвращаем индекс, если символ найден

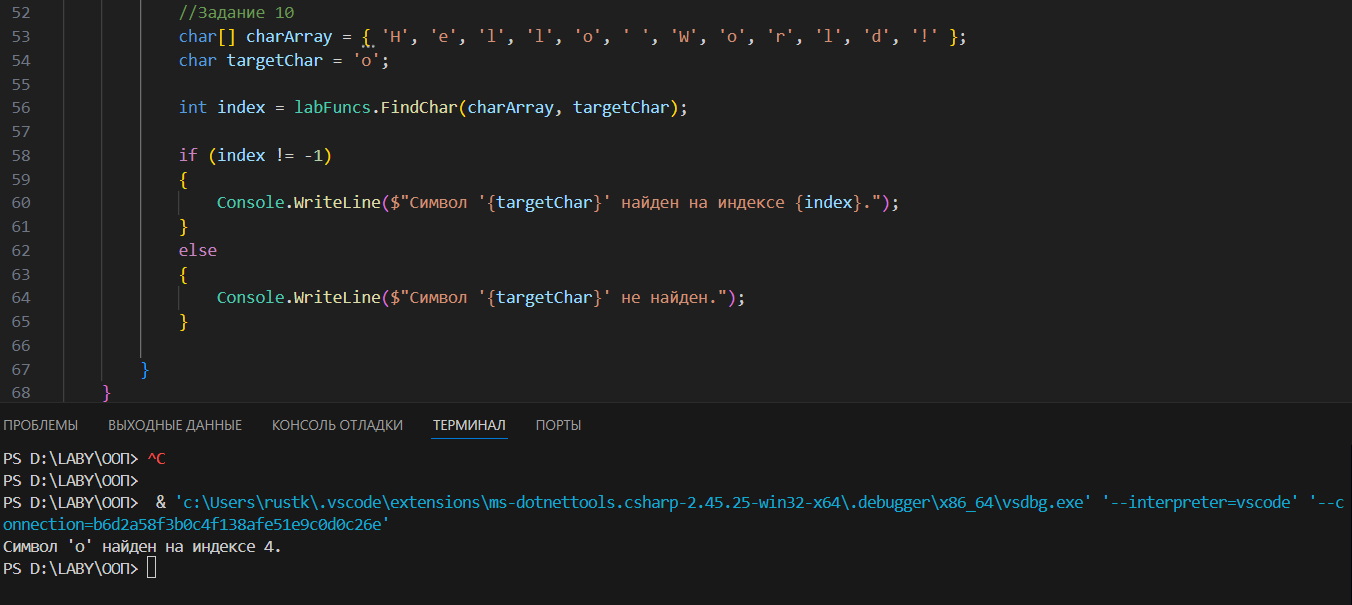
                }

            }

            return -1; // Возвращаем -1, если символ не найден

        }

Вывод:



11. Метод для определения количества повторений указанного символа в символьном массиве.

Код:

 public static int CountCharOccurrences(char[] charArray, char target)

        {

            int count = 0;

            foreach (char c in charArray)

            {

                if (c == target)

                {

                    count++;

                }

            }

            return count;

        }

Вывод: Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

12. Метод принимает два символа («первый» и «второй») и в символьном массиве заменяет все символы «первый» на символ «второй».

Код:

public static void ReplaceChar(char[] charArray, char first, char second)

        {

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

            {

                if (charArray[i] == first)

                {

                    charArray[i] = second;

                }

            }

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

13. Метод принимает символ и из исходного символьного массива формирует новый символьный массив, в котором удалены все символы, совпадающие с заданным

Код:

public static char[] RemoveChar(char[] charArray, char target)

        {

            int newSize = 0;

            foreach (char c in charArray)

            {

                if (c != target)

                {

                    newSize++;

                }

            }

            char[] newArray = new char[newSize];

            int index = 0;

            foreach (char c in charArray)

            {

                if (c != target)

                {

                    newArray[index++] = c;

                }

            }

            return newArray;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

14. Из исходного символьного массива метод формирует новый символьный массив, в котором все подряд идущие пробелы заменены на один пробел.

Код:

 public static char[] CompressSpaces(char[] charArray)

        {

            int newSize = 0;

            bool inSpaceSequence = false;

            foreach (char c in charArray)

            {

                if (char.IsWhiteSpace(c))

                {

                    if (!inSpaceSequence)

                    {

                        newSize++;

                        inSpaceSequence = true;

                    }

                }

                else

                {

                    newSize++;

                    inSpaceSequence = false;

                }

            }

            char[] newArray = new char[newSize];

            int index = 0;

            inSpaceSequence = false;

            foreach (char c in charArray)

            {

                if (char.IsWhiteSpace(c))

                {

                    if (!inSpaceSequence)

                    {

                        newArray[index++] = ' ';

                        inSpaceSequence = true;

                    }

                }

                else

                {

                    newArray[index++] = c;

                    inSpaceSequence = false;

                }

            }

            return newArray;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

15. Из исходного символьного массива метод формирует новый символьный массив, в котором все элементы расположены в обратной последовательности.

Код:

public static char[] ReverseArray(char[] charArray)

        {

            char[] reversedArray = new char[charArray.Length];

            int index = 0;

            // Заполняем новый массив в обратном порядке

            for (int i = charArray.Length - 1; i >= 0; i--)

            {

                reversedArray[index++] = charArray[i];

            }

            return reversedArray;

        }

Вывод: Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

16. Метод определяет читаются ли элементы символьного массива одинаково как слева направо, так и наоборот.

Код:

public static bool IsPalindrome(char[] charArray)

        {

            int left = 0;

            int right = charArray.Length - 1;

            while (left < right)

            {

                if (charArray[left] != charArray[right])

                {

                    return false;

                }

                left++;

                right--;

            }

            return true;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

17. Метод, который определяет в символьном массиве (как произвольном тексте):

Количество слов.

Количество предложений.

Код:

public static (int wordCount, int sentenceCount) CountWordsAndSentences(char[] charArray)

        {

            int wordCount = 0;

            int sentenceCount = 0;

            bool inWord = false;

            foreach (char c in charArray)

            {

                if (char.IsLetterOrDigit(c))

                {

                    if (!inWord)

                    {

                        wordCount++;

                        inWord = true;

                    }

                }

                else

                {

                    if (c == '.' || c == '?' || c == '!')

                    {

                        sentenceCount++;

                    }

                    inWord = false;

                }

            }

            if (inWord)

            {

                wordCount++;

            }

            return (wordCount, sentenceCount);

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

18. Метод, который преобразует символьный массив (как произвольный текст) так, чтобы предложения начинались с заглавной буквы, а все остальные буквы предложения становились строчными.

Код:

public static char[] CapitalizeSentences(char[] charArray)

        {

            bool capitalizeNext = true;

            char[] resultArray = new char[charArray.Length];

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

            {

                char c = charArray[i];

                if (char.IsLetter(c))

                {

                    if (capitalizeNext)

                    {

                        resultArray[i] = char.ToUpper(c);

                        capitalizeNext = false;

                    }

                    else

                    {

                        resultArray[i] = char.ToLower(c);

                    }

                }

                else

                {

                    resultArray[i] = c;

                    if (c == '.' || c == '?' || c == '!')

                    {

                        capitalizeNext = true;

                    }

                }

            }

            return resultArray;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

19. Метод, который определяет в символьном массиве (как произвольном тексте) и возвращает: Все разделители. Все знаки препинания.

Код:

public static (char[] separators, char[] punctuation) ExtractSeparatorsAndPunctuation(char[] charArray)

        {

            HashSet<char> separatorsSet = new HashSet<char>();

            HashSet<char> punctuationSet = new HashSet<char>();

            foreach (char c in charArray)

            {

                if (char.IsWhiteSpace(c))

                {

                    separatorsSet.Add(c);

                }

                else if (c == '.' || c == ',' || c == ';' || c == ':' || c == '?' || c == '!')

                {

                    punctuationSet.Add(c);

                }

            }

            return (separatorsSet.ToArray(), punctuationSet.ToArray());

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

**ЗАДАНИЕ № 2**

Создать класс со статическими методами обработки строк. Выполнить тестирование разработанных методов.

1. Метод для удаления пробелов в строке. *Этострокадлятеста*

Код:

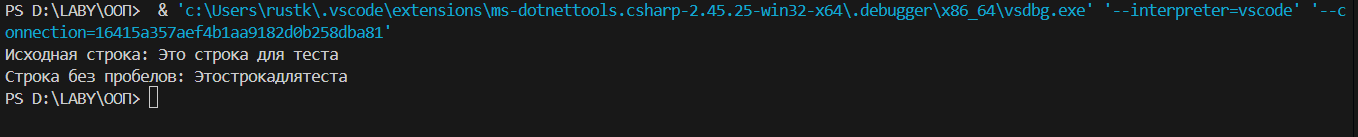
public static string RemoveSpaces(string input)

        {

            return input.Replace(" ", "");

        }

Вывод:



2. Метод для замены пробелов дефисами в строке. *Это-строка-для-теста*

Код:

public static string ReplaceSpacesWithHyphens(string input)

        {

            return input.Replace(" ", "-");

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

3. Метод возвращает строку, записанную справа налево. *атсет ялд акортс отЭ*

Код:

public static string ReverseString(string input)

        {

            char[] charArray = input.ToCharArray();

            Array.Reverse(charArray);

            return new string(charArray);

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

4. Метод для подсчета суммарного числа букв 'а' и букв 'т' в строке. \

*k'а' = 2 k'т' = 4*Код:

 public static int CountAAndT(string input)

        {

            int countA = 0;

            int countT = 0;

            foreach (char c in input)

            {

                if (c == 'а' || c == 'А')

                {

                    countA++;

                }

                else if (c == 'т' || c == 'Т')

                {

                    countT++;

                }

            }

            return countA + countT;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

5. Метод для подсчета кол-ва слов в предложении (между словами только один пробел). Кол-во слов = 4

Код:

public static int CountWords(string input)

        {

            if (string.IsNullOrWhiteSpace(input))

            {

                return 0;

            }

            string[] words = input.Split(' ', StringSplitOptions.RemoveEmptyEntries);

            return words.Length;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

6. Метод для формирования массива слов из слов, имеющихся в предложении. Это строка для теста

Код:

 public static string[] ExtractWords(string input)

        {

            if (string.IsNullOrWhiteSpace(input))

            {

                return Array.Empty<string>();

            }

            return input.Split(' ', StringSplitOptions.RemoveEmptyEntries);

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

7. Метод для вывода элементов (слов) строкового массива. *Это строка для теста*

Код:

public static void PrintStringArray(string[] array)

        {

            foreach (string element in array)

            {

                Console.WriteLine(element);

            }

        }

Вывод: Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание8. Метод для формирования строки в виде обратного порядка слов в предложении (сформированное предложение начинается с заглавной буквы, все остальные буквы – строчные, в конце предложения – точка). Теста для строка это.

Код:

public static string ReverseWordsInSentence(string input)

        {

            string[] words = input.Split(' ', StringSplitOptions.RemoveEmptyEntries);

            Array.Reverse(words);

            string reversedSentence = string.Join(" ", words);

            reversedSentence = char.ToUpper(reversedSentence[0]) + reversedSentence.Substring(1).ToLower() + ".";

            return reversedSentence;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

9. Задан текст, содержащий несколько строк. Метод для определения количества символов в каждой строке.

Код:

            string[] lines = input.Split(new[] { '\n' }, StringSplitOptions.None);

            int[] charCounts = new int[lines.Length];

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

            {

                charCounts[i] = lines[i].Length;

            }

            return charCounts;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

10. Задан текст, содержащий несколько строк. Метод для удаления из него: указанной (в виде номера) строки и первой по порядку следования строки, в конце которой стоит вопросительный знак. В обоих случаях результат записать в другой текст.

Код:

 public static string RemoveLineAndQuestionLine(string input, int lineToRemove)

        {

            string[] lines = input.Split(new[] { '\n' }, StringSplitOptions.None);

            var result = new List<string>();

            bool questionMarkLineRemoved = false;

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

            {

                if (i == lineToRemove - 1)

                {

                    continue;

                }

                if (!questionMarkLineRemoved && lines[i].Trim().EndsWith("?"))

                {

                    questionMarkLineRemoved = true;

                    continue;

                }

                result.Add(lines[i]);

            }

            return string.Join("\n", result);

        }

Вывод: Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

11. Дан текст, каждая строка которого представляет собой арифметическое выражение. Метод, который для каждой строки проверяет сбалансированность открывающих и закрывающих скобок, выводит на экран порядковый номер и содержимое строк, в которых скобки не сбалансированы.

Код:

public static void CheckBracketBalance(string input)

        {

            string[] lines = input.Split(new[] { '\n' }, StringSplitOptions.None);

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

            {

                if (!labFuncs.AreBracketsBalanced(lines[i]))

                {

                    Console.WriteLine($"Номер строки: {i + 1}");

                    Console.WriteLine($"Несбалансированная строка: {lines[i]}");

                }

            }

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

12. Имеется текст. Метод, который выполняет следующие действия: а) Возвращает длину самой длинной строки. б) Возвращает номер самой длинной строки. Если таких строк несколько, то находит номер одной из них. в) Возвращает самую длинную строку. Если таких строк несколько, то возвращает первую из них.

Код:

public static int GetLongestLineLength(string input)

        {

            string[] lines = input.Split(new[] { '\n' }, StringSplitOptions.None);

            int maxLength = 0;

            foreach (string line in lines)

            {

                if (line.Length > maxLength)

                {

                    maxLength = line.Length;

                }

            }

            return maxLength;

        }

        public static int GetLongestLineNumber(string input)

        {

            string[] lines = input.Split(new[] { '\n' }, StringSplitOptions.None);

            int maxLength = 0;

            int longestLineIndex = 0;

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

            {

                if (lines[i].Length > maxLength)

                {

                    maxLength = lines[i].Length;

                    longestLineIndex = i + 1;

                }

            }

            return longestLineIndex;

        }

        public static string GetLongestLine(string input)

        {

            string[] lines = input.Split(new[] { '\n' }, StringSplitOptions.None);

            string longestLine = "";

            int maxLength = 0;

            foreach (string line in lines)

            {

                if (line.Length > maxLength)

                {

                    maxLength = line.Length;

                    longestLine = line;

                }

            }

            return longestLine;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

13. Имеется текст. Метод, который переписывает его строки в перевернутом виде в другой текст. Порядок строк во втором тексте должен: а) совпадать с порядком строк в заданном тексте; б) быть обратным по отношению к порядку строк в заданном тексте.

Код:

public static string ReverseLinesInSameOrder(string input)

        {

            string[] lines = input.Split(new[] { '\n' }, StringSplitOptions.None);

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

            {

                lines[i] = ReverseString(lines[i]);

            }

            return string.Join("\n", lines);

        }

        public static string ReverseLinesInReverseOrder(string input)

        {

            string[] lines = input.Split(new[] { '\n' }, StringSplitOptions.None);

            Array.Reverse(lines);

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

            {

                lines[i] = ReverseString(lines[i]);

            }

            return string.Join("\n", lines);

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

14. Метод для нахождения элемента строкового массива с наибольшим количеством символов. Im = 1 Jm = 0

Код:

public static string FindLongestString(string[] strings)

        {

            if (strings == null || strings.Length == 0)

            {

                throw new ArgumentException("Массив строк не может быть пустым.");

            }

            string longestString = strings[0];

            foreach (string str in strings)

            {

                if (str.Length > longestString.Length)

                {

                    longestString = str;

                }

            }

            return longestString;

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

15. Метод для преобразования строкового массива в строку с заданием символа-разделителя между элементами массива в строке. один, два, три, четыре, пять, шесть, семь, восемь, девять (обращение к методу с ',') один; два; три; четыре; пять; шесть; семь; восемь; девять (обращение к методу с ';')

Код:

public static string JoinWithDelimiter(string[] strings, char delimiter)

        {

            if (strings == null || strings.Length == 0)

            {

                return string.Empty;

            }

            return string.Join(delimiter, strings);

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

16. Метод для преобразования строки в одномерный строковый массив с указанием символа разделителя между подстроками, которые станут элементами строкового массива. Метод для вывода одномерного строкового массива. (обращение к методу с ',' затем вывод) один два три четыре пять шесть семь восемь девять (обращение к методу с ';' затем вывод) один два три четыре пять шесть семь восемь девять

Код:

public static string[] SplitIntoArray(string input, char delimiter)

        {

            if (string.IsNullOrEmpty(input))

            {

                return new string[0];

            }

            return input.Split(delimiter);

        }

        public static void PrintArray(string[] array)

        {

            if (array == null || array.Length == 0)

            {

                Console.WriteLine("Массив пуст.");

                return;

            }

            foreach (string item in array)

            {

                Console.WriteLine(item);

            }

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

17. Метод для преобразования одномерного строкового массива в строку с заданием символа-разделителя между элементами массива в строке. один, два, три, четыре, пять, шесть, семь, восемь, девять (обращение к методу с ',') один; два; три; четыре; пять; шесть; семь; восемь; девять (обращение к методу с ';')

Код:

public static string JoinArrayWithDelimiter(string[] array, char delimiter)

        {

            if (array == null || array.Length == 0)

            {

                return string.Empty;

            }

            return string.Join(delimiter, array);

        }

Вывод:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Ссылка на файлы лабораторной работы: <https://github.com/ironsast/OOP_LABS>

**Вывод:**

В рамках лабораторной работы были разработаны методы для обработки символьных данных и строк на языке C#. В частности, реализованы методы для выполнения арифметических операций над символами, вывода их кодов, преобразования текста в символьные массивы и обратно, а также для замены символов и удаления пробелов.

Кроме того, разработаны методы для работы со строками, включая удаление пробелов, замены их на дефисы, переворот строки, подсчёт букв и формирование массива слов. Особое внимание было уделено обработке строки с заданным разделителем и нахождению самой длинной строки в массиве.

Все методы были протестированы на примерах, что подтвердило их правильность и соответствие поставленным задачам. Эта работа продемонстрировала возможности языка C# для эффективной обработки данных и текста.