ст. Гр КН-923с

Кузьмін Станіслав

**Звіт з лабораторної роботи №2**

Робота із рядками та файлове введення / виведення у мові C# .

**Мета роботи:** набути навички в написанні програм з використанням рядків та застосування файлових операцій.

Завдання 1 Використовуючи завдання власного варіанта до лабораторної роботи 12 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.

/\* Розробити функцію, яка виконує ту обробку символьного рядка, яка визначена у виданому індивідуальному завданні (див. п.6).

Вирівнювання по лівому краю\*/

using System;

class Program

{

static void Main()

{

Console.Write("Введіть рядок: ");

string input = Console.ReadLine();

Console.Write("Введіть довжину: ");

if (int.TryParse(Console.ReadLine(), out int length))

{

// Виклик функції

string result = PadLeftAlign(input, length);

Console.WriteLine($"Рядок вирівняний по лівому краю: '{result}'");

}

else

{

Console.WriteLine("Введено некоректну довжину.");

}

}

static string PadLeftAlign(string str, int length)

{

if (str.Length >= length)

{

return str;

}

int spacesToAdd = length - str.Length;

// Додавання пробілів

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

{

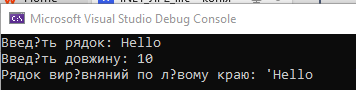
str += " ";

}

return str;

}

}



Завдання 2 Виконати минуле завдання за варіантом із відліком з кінця, тобто 30 -1-й, 29- 2-ий, 28-3-ій, і т.д.

//Рядок без повторень

using System;

class Program

{

static void Main()

{

Console.Write("Введіть рядок: ");

string input = Console.ReadLine();

string result = RemoveDuplicateCharacters(input);

Console.WriteLine($"Результат: '{result}'");

}

// Функція для заміни повторюваних символів на один

static string RemoveDuplicateCharacters(string str)

{

if (string.IsNullOrEmpty(str))

{

return str;

}

// новий рядок без повторюваних символів

string result = "";

char? lastChar = null;

foreach (char c in str)

{

if (c != lastChar)

{

result += c;

lastChar = c;

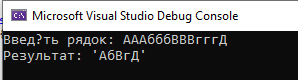
}

}

return result;

}

}



Завдання 3 Визначити кількість входжень у текстовий рядок буквосполучення "нн";

// Кількість сполучень нн

using System;

class Program

{

static void Main()

{

Console.Write("Введіть текстовий рядок: ");

string input = Console.ReadLine();

string pattern = "нн";

int count = CountSymbol(input, pattern);

Console.WriteLine($"Кількість входжень буквосполучення '{pattern}' у рядку: {count}");

}

// Функція

static int CountSymbol(string text, string pattern)

{

if (string.IsNullOrEmpty(text) || string.IsNullOrEmpty(pattern))

{

return 0;

}

int count = 0;

int patternLength = pattern.Length;

int textLength = text.Length;

for (int i = 0; i <= textLength - patternLength; i++)

{

// Substring початок, довжина =

if (text.Substring(i, patternLength) == pattern)

{

count++;

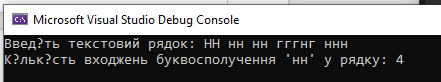
}

}

return count;

}

}



Завдання 4 Замінити в текстовому рядку кожну кому двома знаками питання;

//Замінити в текстовому рядку кожну кому двома знаками питання;

using System;

class Program

{

static void Main()

{

Console.Write("Введіть рядок: ");

string input = Console.ReadLine();

string result = ReplaceComma(input);

Console.WriteLine($"Результат: {result}");

}

static string ReplaceComma(string text)

{

int newLength = text.Length;

foreach (char c in text)

{

if (c == ',')

{

newLength++;

}

}

char[] resultArray = new char[newLength];

int index = 0;

foreach (char c in text)

{

if (c == ',')

{

resultArray[index] = '?';

index++;

resultArray[index] = '?';

index++;

}

else

{

resultArray[index] = c;

index++;

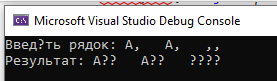
}

}

return new string(resultArray);

}

}



Завдання 5 Студенти шифрують свої записки, записуючи всі слова навпаки. Скласти алгоритм і програму шифрування і розшифровування

//Шифратор

using System;

class Program

{

static void Main()

{

Console.Write("Введіть текст для шифрування:");

string input = Console.ReadLine();

string encryptedText = Encrypt(input);

Console.WriteLine("Зашифрований текст:");

Console.WriteLine(encryptedText);

string decryptedText = Decrypt(encryptedText);

Console.WriteLine("Розшифрований текст:");

Console.WriteLine(decryptedText);

}

static string Encrypt(string text)

{

// Розділити текст на слова

// Split

string[] words = text.Split(' ');

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

{

words[i] = ReverseWord(words[i]);

}

return string.Join(" ", words);

}

static string Decrypt(string text)

{

string[] words = text.Split(' ');

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

{

words[i] = ReverseWord(words[i]);

}

return string.Join(" ", words);

}

static string ReverseWord(string word)

{

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

int index = 1;

foreach (char c in word)

{

charArray[word.Length - index] = c;

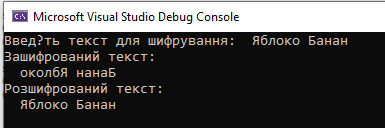
index++;

}

return new string(charArray);

}

}



Завдання 6 а) підраховує кількість великих літер у тексті; б) виводить на екран слова, що мають найменшу кількість літер; в) видаляє всі слова, що починаються з малої літери.

/\*) підраховує кількість великих літер у тексті;

б) виводить на екран слова, що мають найменшу кількість літер;

в) видаляє всі слова, що починаються з малої літери.\*/

using System;

using System.Text;

class Program

{

static void Main()

{

Console.OutputEncoding = UTF8Encoding.UTF8;

Console.Write("Введіть текст для шифрування:");

string input = Console.ReadLine();

int uppercaseCount = CountUpperLetters(input);

Console.WriteLine($"Кількість великих літер: {uppercaseCount}");

SmallNumber(input);

Console.WriteLine("Текст після видалення слів з маленькою літерою:");

string result = DeleteSmallLetter(input);

Console.WriteLine(result);

}

static int CountUpperLetters(string text)

{

int count = 0;

foreach (char c in text)

{

if (c >= 'А' && c <= 'Я')

{

count++;

}

}

return count;

}

static void SmallNumber(string text)

{

string[] words = text.Split(' ');

int minLength = int.MaxValue;

foreach (string word in words)

{

if (word.Length < minLength)

{

minLength = word.Length;

}

}

Console.WriteLine("Слова з найменшою довжиною:");

foreach (string word in words)

{

if (word.Length == minLength)

{

Console.WriteLine(word);

}

}

}

static string DeleteSmallLetter(string text)

{

string[] words = text.Split(' ');

string result = "";

foreach (string word in words)

{

if (word.Length > 0 && !(word[0] >= 'а' && word[0] <= 'я'))

{

result += word + " ";

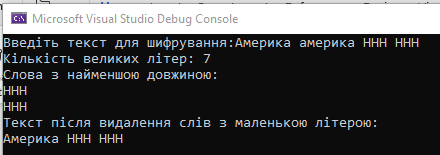
}

}

return result;

}

}



Завдання 7 Модифікувати перші три завдання цієї роботи таким чином, щоб введення початкового рядка чи рядків та виведення результату обробки цього рядка чи рядків були реалізовані за допомогою технологій використання текстових файлів у мові C#.

Для завдання 1

using System;

using System.IO;

class Program

{

static void Main()

{

// Вказуємо шляхи до вхідного, вихідного файлів і файлу з довжиною

string inputFilePath = "input.txt";

string outputFilePath = "output.txt";

string lengthFilePath = "length.txt";

// Зчитуємо вміст вхідного файлу і видаляємо пробіли на початку та в кінці

string input = File.ReadAllText(inputFilePath).Trim();

// Перевіряємо, чи існує файл з довжиною

if (File.Exists(lengthFilePath))

{

// Зчитуємо вміст файлу з довжиною і видаляємо пробіли

string lengthContent = File.ReadAllText(lengthFilePath).Trim();

// Спробуємо перетворити вміст на ціле число

if (int.TryParse(lengthContent, out int length))

{

// Викликаємо метод для вирівнювання рядка по лівому краю з заданою довжиною

string result = PadLeftAlign(input, length);

// Записуємо результат у вихідний файл

File.WriteAllText(outputFilePath, $"Рядок вирівняний по лівому краю: '{result}'");

}

else

{

// Якщо перетворення не вдалося, записуємо повідомлення про некоректну довжину

File.WriteAllText(outputFilePath, "Введено некоректну довжину.");

}

}

else

{

// Якщо файл з довжиною не знайдено, записуємо відповідне повідомлення

File.WriteAllText(outputFilePath, "Файл з довжиною не знайдено.");

}

}

// Метод для вирівнювання рядка по лівому краю з вказаною довжиною

static string PadLeftAlign(string str, int length)

{

// Якщо рядок вже довший або дорівнює вказаній довжині, повертаємо його як є

if (str.Length >= length)

{

return str;

}

// Обчислюємо кількість пробілів, які потрібно додати

int spacesToAdd = length - str.Length;

// Додаємо пробіли до рядка

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

{

str += " ";

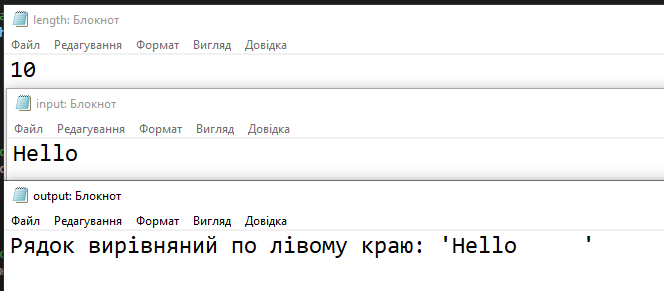
}

// Повертаємо вирівняний рядок

return str;

}

}



Для Завдання 2

using System;

using System.IO;

using System.Text;

class Program

{

static void Main()

{

// Шляхи до вхідного та вихідного файлів

string inputFilePath = "input.txt";

string outputFilePath = "output.txt";

// Зчитування вмісту вхідного файлу і видалення пробілів на початку та в кінці

string input = File.ReadAllText(inputFilePath).Trim();

// Видалення дублікованих символів з рядка

string result = RemoveDuplicateCharacters(input);

// Запис результату в вихідний файл

File.WriteAllText(outputFilePath, $"Результат: '{result}'");

}

// Метод для видалення дублікованих символів з рядка

static string RemoveDuplicateCharacters(string str)

{

// Перевіряємо, чи рядок порожній або null

if (string.IsNullOrEmpty(str))

{

return str;

}

// Використовуємо StringBuilder для створення результату

StringBuilder result = new StringBuilder();

char? lastChar = null; // Змінна для збереження останнього символу

// Проходимо по кожному символу рядка

foreach (char c in str)

{

// Додаємо символ до результату, якщо він не є дублікатом попереднього

if (c != lastChar)

{

result.Append(c);

lastChar = c; // Оновлюємо останній символ

}

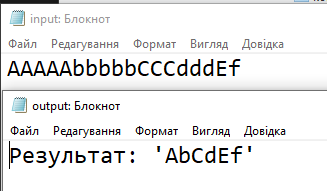
}

// Повертаємо рядок без дублікованих символів

return result.ToString();

}

}



Для завдання 3

using System;

using System.IO;

class Program

{

static void Main()

{

// Шляхи до вхідного, вихідного файлів та файлу з шаблоном

string inputFilePath = "input.txt";

string outputFilePath = "output.txt";

string patternFilePath = "pattern.txt";

// Зчитуємо вміст вхідного файлу

string input = File.ReadAllText(inputFilePath);

// Зчитуємо шаблон з файлу

string pattern = File.ReadAllText(patternFilePath);

// Підраховуємо кількість входжень шаблону у рядку

int count = CountSymbol(input, pattern);

// Записуємо результат у вихідний файл

File.WriteAllText(outputFilePath, $"Кількість входжень буквосполучення '{pattern}' у рядку: {count}");

}

// Метод для підрахунку кількості входжень шаблону у тексті

static int CountSymbol(string text, string pattern)

{

// Перевіряємо, чи рядок або шаблон порожні або null

if (string.IsNullOrEmpty(text) || string.IsNullOrEmpty(pattern))

{

return 0;

}

int count = 0; // Лічильник входжень шаблону

int patternLength = pattern.Length; // Довжина шаблону

int textLength = text.Length; // Довжина тексту

// Проходимо по всьому тексту, щоб знайти всі входження шаблону

for (int i = 0; i <= textLength - patternLength; i++)

{

// Перевіряємо, чи підрядок з тексту дорівнює шаблону

if (text.Substring(i, patternLength) == pattern)

{

count++; // Збільшуємо лічильник

}

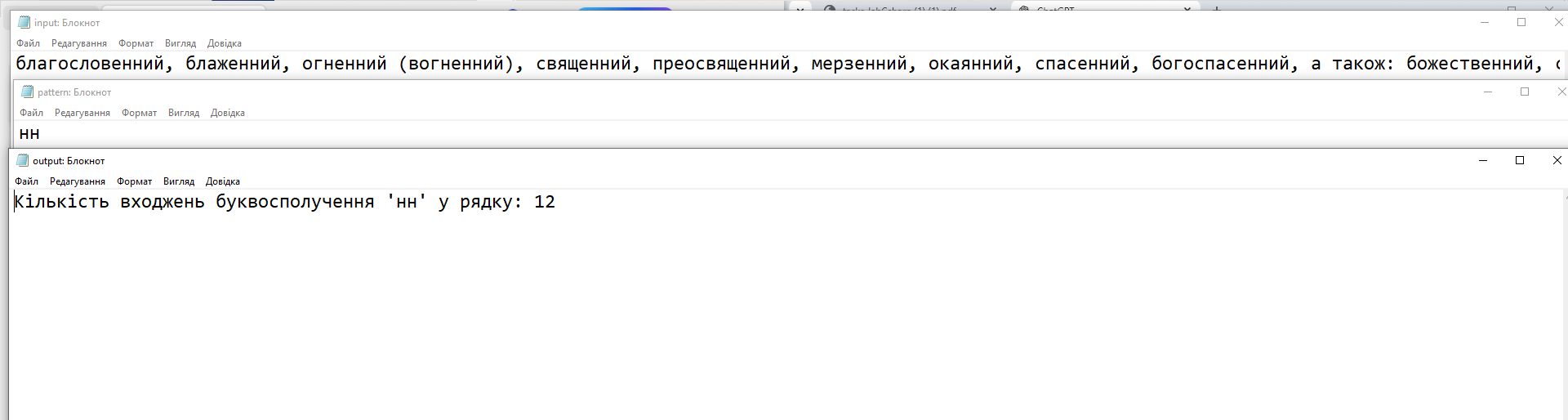
}

// Повертаємо загальну кількість входжень

return count;

}

}



Завдання 8\* Модифікувати четверте і п’яте завдання цієї роботи таким чином, щоб виведення результату обробки рядків були реалізовані за допомогою технологій використання двійкових файлів у мові C# із можливістю перевірки коректності результатів шляхом виведення контенту двійкового файлу на екран консолі.

Для завдання 4

using System;

using System.IO;

class Program

{

static void Main()

{

// Шлях до бінарного файлу

string filePath = "output.bin";

// Зчитування рядка з консолі

Console.Write("Введіть рядок: ");

string input = Console.ReadLine();

// Заміна ком в рядку на два символи '?'

string result = ReplaceComma(input);

// Запис результату у бінарний файл

using (BinaryWriter writer = new BinaryWriter(File.Open(filePath, FileMode.Create)))

{

writer.Write(result); // Записуємо рядок у файл

}

// Читання результату з бінарного файлу

using (BinaryReader reader = new BinaryReader(File.Open(filePath, FileMode.Open)))

{

string readResult = reader.ReadString(); // Читаємо рядок з файлу

Console.WriteLine($"Результат з файлу: {readResult}"); // Виводимо результат на консоль

}

}

// Метод для заміни ком у рядку на два символи '?'

static string ReplaceComma(string text)

{

// Обчислюємо нову довжину рядка, враховуючи подвоєння ком

int newLength = text.Length;

foreach (char c in text)

{

if (c == ',')

{

newLength++; // Збільшуємо довжину на одиницю для кожної коми

}

}

// Створюємо масив символів для результату з новою довжиною

char[] resultArray = new char[newLength];

int index = 0;

// Проходимо по кожному символу вихідного рядка

foreach (char c in text)

{

if (c == ',')

{

// Замінюємо кому на два символи '?'

resultArray[index] = '?';

index++;

resultArray[index] = '?';

index++;

}

else

{

// Копіюємо інші символи без змін

resultArray[index] = c;

index++;

}

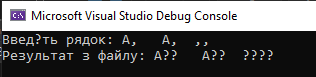
}

// Повертаємо новий рядок з заміненими комами

return new string(resultArray);

}

}



Для завдання 5

using System;

using System.IO;

class Program

{

static void Main()

{

// Шляхи до бінарних файлів для зберігання зашифрованого і дешифрованого тексту

string filePath1 = "Encrypt.bin";

string filePath2 = "Decrypt.bin";

// Зчитування тексту з консолі для шифрування

Console.Write("Введіть текст для шифрування:");

string input = Console.ReadLine();

// Шифрування тексту

string encryptedText = Encrypt(input);

// Запис зашифрованого тексту в бінарний файл

using (BinaryWriter writer = new BinaryWriter(File.Open(filePath1, FileMode.Create)))

{

writer.Write(encryptedText);

}

// Читання зашифрованого тексту з бінарного файлу для перевірки

using (BinaryReader reader = new BinaryReader(File.Open(filePath1, FileMode.Open)))

{

string readResult = reader.ReadString();

Console.WriteLine($"Зашифрований результат з файлу: {readResult}");

}

// Дешифрування тексту

string decryptedText = Decrypt(encryptedText);

// Запис дешифрованого тексту в інший бінарний файл

using (BinaryWriter writer = new BinaryWriter(File.Open(filePath2, FileMode.Create)))

{

writer.Write(decryptedText);

}

// Читання дешифрованого тексту з бінарного файлу для перевірки

using (BinaryReader reader = new BinaryReader(File.Open(filePath2, FileMode.Open)))

{

string readResult = reader.ReadString();

Console.WriteLine($"Дешифрований результат з файлу: {readResult}");

}

}

// Метод для шифрування тексту (зворотній порядок слів)

static string Encrypt(string text)

{

// Розділяємо текст на слова за пробілами

string[] words = text.Split(' ');

// Реверсуємо кожне слово

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

{

words[i] = ReverseWord(words[i]);

}

// Об'єднуємо слова назад в рядок з пробілами

return string.Join(" ", words);

}

// Метод для дешифрування тексту (зворотній порядок слів, так само як у шифруванні)

static string Decrypt(string text)

{

// Розділяємо текст на слова за пробілами

string[] words = text.Split(' ');

// Реверсуємо кожне слово

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

{

words[i] = ReverseWord(words[i]);

}

// Об'єднуємо слова назад в рядок з пробілами

return string.Join(" ", words);

}

// Метод для реверсування символів у слові

static string ReverseWord(string word)

{

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

int index = 1;

// Проходимо по кожному символу в слові

foreach (char c in word)

{

// Розміщуємо символи в зворотному порядку

charArray[word.Length - index] = c;

index++;

}

// Повертаємо нове слово у зворотному порядку

return new string(charArray);

}

}

