**Виконав: Мельник Б. В.; КН-922в**

**Лабораторна робота №3**

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

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

**Індивідуальні завдання**

**Варіант №12**

**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#.

***Завдання 12 лабораторної роботи:*** Розробити функцію, яка виконує ту обробку символьного рядка, яка визначена у виданому індивідуальному завданні (див. п.6). При реалізації функції забороняється користуватися функціями бібліотек мови ***C***.

Примітки:

* У більшості завдань необхідно передбачити роботу функції при некоректному завданні її параметрів.
* У тих завданнях, де застосований термін “слово”, слід вживати будь-яку послідовність символів, обмежену якимось числом пропусків і початком/закінченням рядка.

***Індивідуальне завадння:***

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання, виконуваного функцією** |
| 12 | Доводить довжину рядка до заданої, вставляючи додаткові пропуски між словами |

***Текст програми:***

using System; // Підключення простору імен System

class Task\_1 // Оголошення класу Task\_1

{

static void Main() // Головний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

Console.WriteLine("Введіть рядок:"); // Виведення запиту на введення рядка

string input = Console.ReadLine(); // Зчитування введеного рядка

Console.WriteLine("Введіть бажану довжину рядка:"); // Виведення запиту на введення бажаної довжини рядка

int desiredLength = int.Parse(Console.ReadLine()); // Зчитування та перетворення введеної довжини в ціле число

string result = JustifyString(input, desiredLength); // Виклик методу JustifyString для вирівнювання рядка

Console.WriteLine($"Результат: >>{result}<<"); // Виведення результату

Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

}

static string JustifyString(string input, int desiredLength) // Метод для вирівнювання рядка

{

if (string.IsNullOrEmpty(input) || desiredLength <= input.Length) // Перевірка, чи рядок порожній або бажана довжина менша або дорівнює довжині рядка

{

return input; // Повернення початкового рядка, якщо умова виконується

}

string[] words = input.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries); // Розбиття рядка на слова

int totalSpaces = desiredLength - input.Length + (words.Length - 1); // Обчислення загальної кількості пробілів

int spaceBetweenWords = totalSpaces / (words.Length - 1); // Обчислення кількості пробілів між словами

int extraSpaces = totalSpaces % (words.Length - 1); // Обчислення залишкових пробілів

string result = words[0]; // Ініціалізація результату першим словом

for (int i = 1; i < words.Length; i++) // Цикл для додавання пробілів між словами

{

int spacesToAdd = spaceBetweenWords + (i <= extraSpaces ? 1 : 0); // Обчислення кількості пробілів для додавання

result += new string(' ', spacesToAdd) + words[i]; // Додавання пробілів та наступного слова до результату

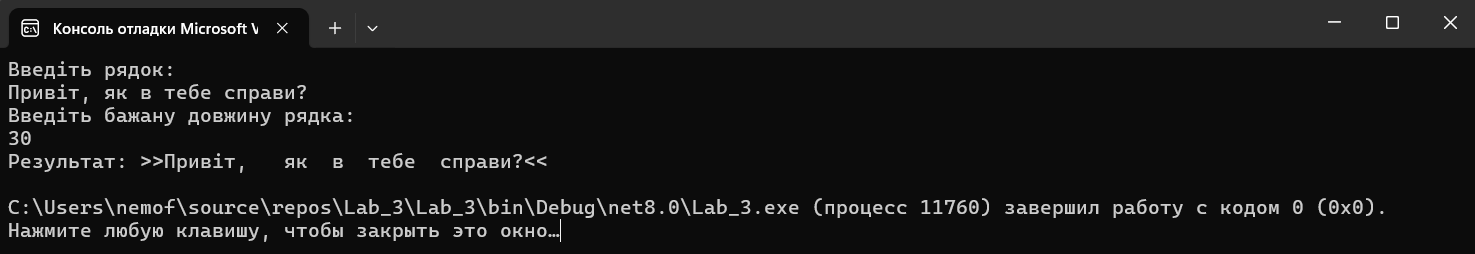
}

return result; // Повернення вирівняного рядка

}

}

***Результат роботи програми:***



**Алгоритм побудови коду**

* **Підключення бібліотек**: Програма використовує простір імен System, що дозволяє працювати з базовими функціями C#.
* **Встановлення кодування**: У методі Main() встановлюється кодування для вводу та виводу символів UTF-8 для правильного відображення та обробки тексту.
* **Введення даних**: Програма очікує введення рядка та бажаної довжини рядка, яку користувач хоче отримати після вирівнювання.
* **Основна логіка вирівнювання**:
  + Створюється метод JustifyString, що приймає вхідний рядок та бажану довжину.
  + Рядок розбивається на окремі слова.
  + Обчислюється загальна кількість пробілів, яку потрібно додати між словами, та розподіляються залишкові пробіли.
  + У циклі між кожним словом додаються пробіли до досягнення бажаної довжини рядка.
* **Виведення результату**: Програма виводить отриманий вирівняний рядок на екран.

**Функціональність**

* Програма вирівнює рядок до вказаної користувачем довжини шляхом додавання пробілів між словами.
* Якщо довжина рядка вже більше або дорівнює бажаній довжині, повертається початковий рядок без змін.
* Підтримується правильне вирівнювання з рівномірним розподілом пробілів між словами.

**Висновок**

Ця програма на мові C# реалізує функціонал вирівнювання рядків за допомогою додавання пробілів між словами, щоб досягти заданої довжини. Це приклад простої, але корисної задачі форматування тексту, що може бути застосована у текстових редакторах або інших системах, де важливо отримати текст із певними параметрами.

**Завдання 2**

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

***Індивідуальне завдання:***

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання, виконуваного функцією** |
| 19 | Видаляє з рядка задану кількість слів, починаючи з слова із заданим номером |

**Текст програми:**

using System; // Підключення простору імен System

class Task\_2 // Оголошення класу Task\_2

{

static void Main() // Головний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

string input; // Оголошення змінної для вводу рядка

int start, count; // Оголошення змінних для номера слова та кількості слів

while (true) // Безкінечний цикл

{

Console.Write("Введіть рядок > "); // Вивід запиту на ввід рядка

input = Console.ReadLine(); // Зчитування вводу користувача

if (input == "\*\*\*") break; // Перевірка на умову завершення циклу

Console.Write("Введіть номер слова та кількість слів для видалення > "); // Запит на ввід параметрів

if (!int.TryParse(Console.ReadLine(), out start) || !int.TryParse(Console.ReadLine(), out count)) // Перевірка коректності вводу параметрів

{

Console.WriteLine("Некоректні параметри."); // Вивід повідомлення про помилку

continue; // Повернення до початку циклу

}

string result = RemoveWords(input, start, count); // Виклик методу для видалення слів

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

break; // Додавання умови для виходу з циклу після виведення результату

}

// Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми (закоментовано)

}

static string RemoveWords(string input, int start, int count) // Метод для видалення слів з рядка

{

if (start < 1 || count < 0) return "Помилка в параметрах."; // Перевірка коректності параметрів

string[] words = input.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries); // Розбиття рядка на слова

if (start > words.Length) return "Помилка в параметрах."; // Перевірка коректності параметра start

int end = Math.Min(start + count - 1, words.Length); // Обчислення кінцевого індексу для видалення

string[] resultWords = new string[words.Length - (end - start + 1)]; // Створення нового масиву для результату

Array.Copy(words, 0, resultWords, 0, start - 1); // Копіювання слів до початку видалення

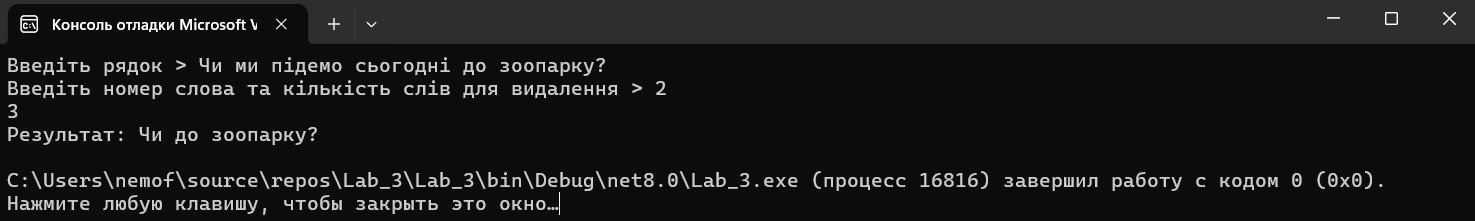
Array.Copy(words, end, resultWords, start - 1, words.Length - end); // Копіювання слів після видалення

return string.Join(" ", resultWords); // Об'єднання слів у рядок та повернення результату

}

}

***Результат роботи програми:***



**Алгоритм побудови коду:**

1. **Читання Введеного Рядка:**
   * Програма запитує користувача ввести рядок та зберігає його.
   * Якщо користувач вводить спеціальний символ "\*\*\*", програма завершує виконання.
2. **Задання Параметрів:**
   * Програма запитує два числа: номер слова, з якого починається видалення, та кількість слів для видалення.
   * Якщо введені некоректні дані (наприклад, не числа), виводиться повідомлення про помилку, і програма починає цикл знову.
3. **Метод RemoveWords:**
   * Розбиває рядок на масив слів.
   * Перевіряє коректність параметрів (номер слова не менший за 1 і кількість слів не від'ємна).
   * Якщо параметри коректні, обчислює індекс початку та кінця слів для видалення.
   * Створює новий масив, копіюючи слова до і після діапазону видалених слів.
   * Повертає результат — новий рядок без вилучених слів.

**Функціональність Коду:**

1. Програма працює з рядками і виконує обробку введених даних користувача.
2. Основна функція полягає у видаленні заданого діапазону слів із рядка за заданими параметрами.
3. Передбачено перевірки на коректність вводу, що забезпечує стабільну роботу програми навіть при помилках користувача.

**Висновок:**

Код вирішує поставлену задачу, дозволяючи користувачеві динамічно маніпулювати текстом через видалення певної кількості слів з конкретної позиції. Завдяки перевіркам на помилки та універсальності методу RemoveWords, програма є стійкою та функціональною для широкого спектру введених даних.

**Завдання 3**

Виконати завдання згідно із номером вашого варіанту:

**12.** Визначити позиції входження у текстовий рядок букви 'Ф'.

***Текст програми:***

using System; // Підключення простору імен System

using System.Collections.Generic; // Підключення простору імен для роботи зі списками

class Task\_3 // Оголошення класу Task\_3

{

static void Main() // Головний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

Console.WriteLine("Введіть текстовий рядок:"); // Виведення запиту на введення тексту

string text = Console.ReadLine(); // Зчитування введеного тексту

char target = 'Ф'; // Визначення цільового символу

List<int> positions = new List<int>(); // Створення списку для зберігання позицій

for (int i = 0; i < text.Length; i++) // Цикл для проходження по всьому тексту

{

if (text[i] == target) // Перевірка, чи символ на поточній позиції дорівнює цільовому

{

positions.Add(i); // Додавання позиції до списку

}

}

Console.WriteLine("Позиції входження букви 'Ф':"); // Виведення заголовку результатів

foreach (int pos in positions) // Цикл для виведення всіх знайдених позицій

{

Console.WriteLine(pos); // Виведення поточної позиції

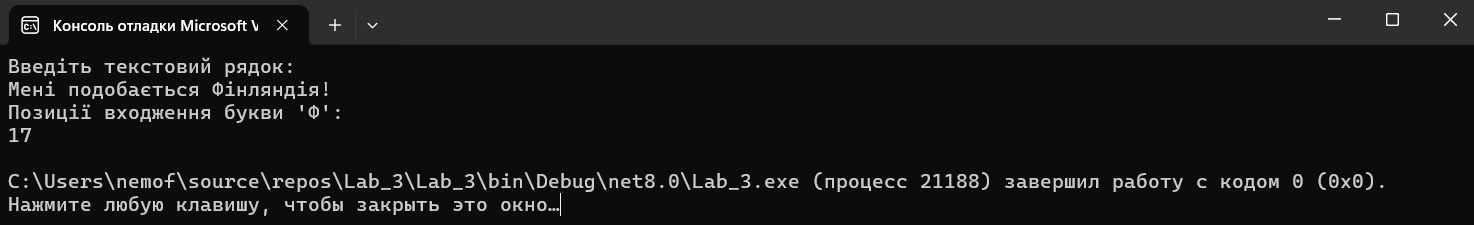
}

Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

}

}

***Результат роботи програми:***



**Алгоритм побудови коду:**

1. **Підключення необхідних просторів імен**: Код починається з підключення простору імен System, який надає базові класи для введення-виведення, та System.Collections.Generic для використання списків.
2. **Оголошення класу та головного методу**: Основний клас програми Task\_3 містить метод Main, де виконується основна логіка.
3. **Встановлення кодування**: Для коректної роботи з текстом, що містить символи кирилиці, встановлюється кодування UTF-8 для введення та виведення.
4. **Зчитування тексту**: Програма виводить запит користувачу для введення текстового рядка та зчитує цей текст за допомогою методу Console.ReadLine().
5. **Оголошення змінних**:
   * Змінна text зберігає введений рядок.
   * Змінна target містить символ, який потрібно знайти — в даному випадку це буква 'Ф'.
   * Використовується список positions для зберігання позицій входження букви 'Ф' у рядок.
6. **Цикл для пошуку позицій**:
   * Створюється цикл for, що проходить по кожному символу введеного тексту.
   * Якщо символ на поточній позиції дорівнює букві 'Ф', ця позиція зберігається у список positions.
7. **Виведення результатів**:
   * Програма виводить всі знайдені позиції за допомогою циклу foreach.
8. **Завершення програми**:
   * Метод Console.ReadKey() використовується для затримки завершення програми до натискання клавіші.

**Функціональність програми:**

* Програма дозволяє користувачеві ввести текстовий рядок.
* Вона знаходить всі позиції входження букви 'Ф' у цьому рядку.
* Результати виводяться на екран у вигляді списку позицій, де кожен елемент відображає індекс відповідної букви.

**Висновок:**

Дана програма є простою реалізацією пошуку символа в рядку. Вона успішно виконує завдання визначення позицій букви 'Ф' та виводить їх. Алгоритм базується на послідовному проходженні через рядок та перевірці кожного символу.

**Завдання 4**

Виконати завдання згідно із номером вашого варіанту:

**12.** У заданому текстовому рядку подвоїти всі букви л, н, с, о.

***Текст програми:***

using System; // Підключення простору імен System

class Task\_4 // Оголошення класу Task\_4

{

static void Main() // Головний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

Console.WriteLine("Введіть текстовий рядок:"); // Виведення запиту на введення тексту

string input = Console.ReadLine(); // Зчитування введеного тексту

string result = DoubleLetters(input); // Виклик методу DoubleLetters для обробки тексту

Console.WriteLine("Результат: " + result); // Виведення результату

Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

}

static string DoubleLetters(string input) // Метод для подвоєння певних літер у рядку

{

string lettersToDouble = "лнсо"; // Літери, які потрібно подвоїти

string result = ""; // Змінна для збереження результату

foreach (char c in input) // Прохід по кожному символу вхідного рядка

{

result += c; // Додавання символу до результату

if (lettersToDouble.Contains(c)) // Перевірка, чи символ є в списку літер для подвоєння

{

result += c; // Додавання символу ще раз, якщо він є в списку

}

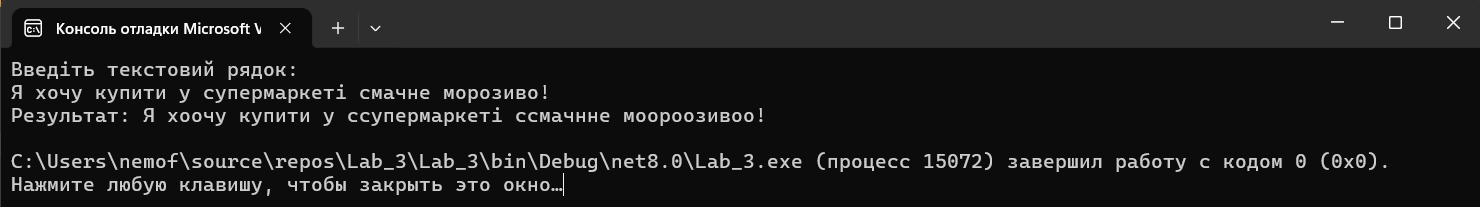
}

return result; // Повернення результату

}

}

***Результат роботи програми:***



**Алгоритм побудови коду:**

1. **Читання введеного рядка**:
   * Спочатку програма запитує у користувача текстовий рядок. Для цього використовується метод Console.ReadLine(), що дозволяє зчитати введений текст.
2. **Обробка тексту**:
   * Метод DoubleLetters приймає вхідний рядок та обробляє його. У цьому методі проходиться по кожному символу рядка.
   * Якщо символ входить до списку літер для подвоєння (літери "л", "н", "с", "о"), він дублюється.
3. **Формування результату**:
   * Результат зберігається у змінну result, в яку додаються всі символи рядка, причому ті, що відповідають умовам, додаються двічі.
4. **Виведення результату**:
   * Після обробки тексту отриманий результат виводиться на екран за допомогою Console.WriteLine().

**Функціональність коду:**

* Код коректно обробляє введений рядок, подвоюючи тільки потрібні літери.
* Використовується цикл foreach для проходу по символах рядка.
* Перевірка літер для подвоєння здійснюється за допомогою методу Contains(), що належить до рядків.
* Код підтримує введення та виведення тексту з українськими символами завдяки встановленню кодування UTF-8.

**Висновок:**

Дана програма дозволяє легко подвоювати певні літери у введеному рядку. Вона демонструє простий алгоритм для обробки рядків та використання основних методів для маніпуляцій з текстом у C#.

**Завдання 5**

Виконати завдання згідно із номером вашого варіанту:

**12.** У словах з закінченням ий замінити його закінченням енький.

***Текст програми:***

using System; // Підключення базової бібліотеки System

using System.Text.RegularExpressions; // Підключення бібліотеки для роботи з регулярними виразами

class Task\_5 // Оголошення класу Task\_5

{

static void Main() // Основний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

Console.WriteLine("Введіть рядок:"); // Виведення запиту на введення рядка

string input = Console.ReadLine(); // Зчитування введеного рядка

string pattern = @"ий\b"; // Оголошення шаблону для пошуку (слова, що закінчуються на "ий")

string replacement = "енький"; // Оголошення рядка для заміни

string result = Regex.Replace(input, pattern, replacement); // Заміна знайдених шаблонів у введеному рядку

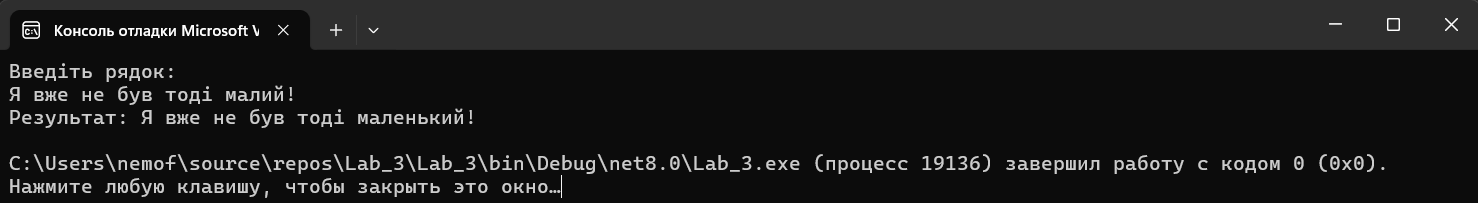
Console.WriteLine("Результат: " + result); // Виведення результату

Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

}

}

***Результат роботи програми:***



**Алгоритм побудови коду:**

1. **Підключення бібліотек**:
   * Підключаються бібліотеки System для базових операцій і System.Text.RegularExpressions для роботи з регулярними виразами, які дозволяють знайти і замінити потрібні частини тексту.
2. **Основний метод програми**:
   * Main — це метод, який виконується при запуску програми.
   * Налаштовується кодування UTF-8 для коректної обробки тексту українською мовою.
3. **Введення рядка**:
   * Програма виводить запит на введення рядка, який містить слова з закінченням "ий".
   * Введений текст зберігається у змінну input.
4. **Оголошення шаблону**:
   * Визначається регулярний вираз @"ий\b", який шукає слова, що закінчуються на "ий". Символ \b означає межу слова, щоб переконатися, що "ий" знаходиться в кінці слова.
5. **Заміна**:
   * Використовується метод Regex.Replace, який замінює всі знайдені збіги шаблону на слово "енький".
6. **Виведення результату**:
   * Програма виводить оброблений рядок з усіма замінами.
7. **Завершення програми**:
   * Програма чекає натискання клавіші перед завершенням для того, щоб користувач встиг переглянути результат.

**Функціональність:**

Програма дозволяє користувачу вводити рядки тексту, в яких всі слова із закінченням "ий" будуть автоматично замінені на "енький". Це може бути корисно для автоматизованої обробки текстів, коли потрібно змінити закінчення слів відповідно до певних правил.

**Висновок:**

Представлена програма на C# ефективно виконує пошук і заміну закінчень слів за допомогою регулярних виразів. Завдяки використанню шаблонів, код можна легко адаптувати для інших завдань зі зміни тексту.

**Завдання 6\***

З клавіатури вводиться текстовий рядок. Розробити програму, яка реалізує вказані дії:

|  |  |
| --- | --- |
| **№**  **варіанту** | **Маніпуляції над рядком** |
| 12 | а) кількість слів, які містять однакову кількість голосних і приголосних літер;  б) виводить на екран найдовше слово;  в) видаляє з тексту всі слова-паліндроми. |

***Текст програми:***

using System; // Підключення простору імен System

using System.Linq; // Підключення простору імен LINQ для роботи з колекціями

using System.Text.RegularExpressions; // Підключення простору імен для роботи з регулярними виразами

class Task\_6

{

static void Main()

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

Console.WriteLine("Введіть текстовий рядок:"); // Виведення запиту на введення рядка

string input = Console.ReadLine(); // Зчитування введеного рядка

// Завдання а)

var result = CountWordsWithEqualVowelsAndConsonants(input); // Підрахунок слів з однаковою кількістю голосних і приголосних

Console.WriteLine($"Кількість слів з однаковою кількістю голосних і приголосних літер: {result.count} ({string.Join(", ", result.words)})"); // Виведення результату

// Завдання б)

string longestWord = FindLongestWord(input); // Пошук найдовшого слова

Console.WriteLine($"Найдовше слово: {longestWord}"); // Виведення найдовшого слова

// Завдання в)

string resultWithoutPalindromes = RemovePalindromes(input); // Видалення слів-паліндромів

Console.WriteLine($"Текст без слів-паліндромів: {resultWithoutPalindromes}"); // Виведення результату без паліндромів

Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

}

static (int count, List<string> words) CountWordsWithEqualVowelsAndConsonants(string input)

{

string[] words = Regex.Split(input, @"\W+"); // Розбиття рядка на слова за допомогою регулярного виразу

int count = 0; // Лічильник слів

List<string> equalWords = new List<string>(); // Список слів з однаковою кількістю голосних і приголосних

foreach (var word in words) // Перебір кожного слова

{

if (string.IsNullOrEmpty(word)) continue; // Пропуск порожніх слів

int vowels = word.Count(c => "аеєиіїоуюяАЕЄИІЇОУЮЯ".Contains(c)); // Підрахунок голосних літер

int consonants = word.Count(c => "бвгґджзйклмнпрстфхцчшщБВГҐДЖЗЙКЛМНПРСТФХЦЧШЩ".Contains(c)); // Підрахунок приголосних літер

if (vowels == consonants) // Перевірка, чи рівна кількість голосних і приголосних

{

count++; // Збільшення лічильника

equalWords.Add(word); // Додавання слова до списку

}

}

return (count, equalWords); // Повернення кількості слів та списку слів

}

static string FindLongestWord(string input)

{

string[] words = Regex.Split(input, @"\W+"); // Розбиття рядка на слова за допомогою регулярного виразу

return words.OrderByDescending(w => w.Length).FirstOrDefault(); // Пошук найдовшого слова

}

static string RemovePalindromes(string input)

{

string[] words = Regex.Split(input, @"(\W+)"); // Розбиття рядка на слова та розділові знаки

var resultWords = words.Where(word => !IsPalindrome(word) || Regex.IsMatch(word, @"\W")); // Видалення паліндромів, залишаючи розділові знаки

return string.Join("", resultWords); // Об'єднання слів у рядок

}

static bool IsPalindrome(string word)

{

if (word.Length < 2) return false; // Короткі слова не є паліндромами

string cleanedWord = new string(word.Where(char.IsLetter).ToArray()).ToLower(); // Видалення не літер та перетворення в нижній регістр

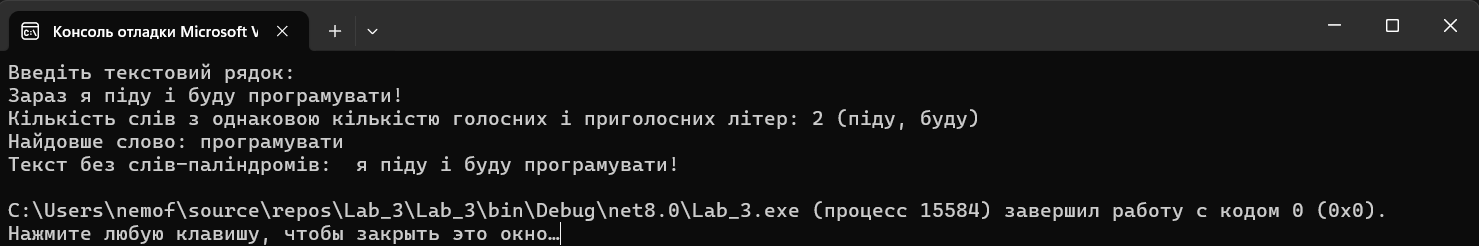
string reversed = new string(cleanedWord.Reverse().ToArray()); // Реверс слова

return cleanedWord.Equals(reversed, StringComparison.OrdinalIgnoreCase); // Перевірка, чи є слово паліндромом

}

}

***Результат роботи програми:***



**Алгоритм коду**

1. **Введення текстового рядка**: Користувач вводить рядок тексту з клавіатури, який використовується для виконання подальших дій.
2. **Реалізація підрахунку слів з однаковою кількістю голосних і приголосних літер**:
   * Текст розбивається на окремі слова за допомогою регулярного виразу.
   * Для кожного слова підраховуються голосні і приголосні літери.
   * Якщо кількість голосних дорівнює кількості приголосних, слово додається до списку таких слів, а лічильник збільшується.
3. **Пошук найдовшого слова**:
   * Текст розбивається на слова, після чого здійснюється пошук слова з найбільшою довжиною.
4. **Видалення слів-паліндромів**:
   * Кожне слово перевіряється на те, чи є воно паліндромом. Якщо слово є паліндромом, воно видаляється з тексту.

**Використані методи**

* **CountWordsWithEqualVowelsAndConsonants**: Підраховує слова, які мають однакову кількість голосних і приголосних.
* **FindLongestWord**: Повертає найдовше слово в тексті.
* **RemovePalindromes**: Видаляє слова-паліндроми з тексту.
* **IsPalindrome**: Перевіряє, чи є слово паліндромом.

**Функціональність програми**

Програма дозволяє:

1. Ввести текстовий рядок з клавіатури.
2. Підрахувати кількість слів, які мають однакову кількість голосних і приголосних літер.
3. Знайти найдовше слово в тексті.
4. Видалити всі паліндроми зі вхідного тексту.

**Висновок**

Програма успішно виконує всі поставлені завдання: обчислює кількість слів з рівною кількістю голосних і приголосних, знаходить найдовше слово і видаляє слова-паліндроми. Завдяки використанню регулярних виразів і LINQ, програма працює ефективно та зручно в користуванні.

**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#.

***Текст модифікованої програми 1 Завдання:***

using System; // Підключення простору імен System

using System.IO; // Підключення простору імен для роботи з файлами

class Task\_7\_1 // Оголошення класу Task\_7\_1

{

static void Main() // Головний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

string inputFilePath = "input.txt"; // Шлях до вхідного файлу

string outputFilePath = "output.txt"; // Шлях до вихідного файлу

if (!File.Exists(inputFilePath)) // Перевірка існування вхідного файлу

{

Console.WriteLine("Файл input.txt не знайдено."); // Виведення повідомлення про відсутність файлу

return; // Завершення виконання програми

}

string input = File.ReadAllText(inputFilePath); // Читання вмісту вхідного файлу

Console.WriteLine("Введіть бажану довжину рядка:"); // Запит на введення бажаної довжини рядка

int desiredLength = int.Parse(Console.ReadLine()); // Зчитування та перетворення введеного значення в ціле число

string result = JustifyString(input, desiredLength); // Вирівнювання рядка до бажаної довжини

File.WriteAllText(outputFilePath, result); // Запис результату у вихідний файл

Console.WriteLine($"Результат записано в файл {outputFilePath}"); // Виведення повідомлення про успішний запис

Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

}

static string JustifyString(string input, int desiredLength) // Метод для вирівнювання рядка

{

if (string.IsNullOrEmpty(input) || desiredLength <= input.Length) // Перевірка на порожній рядок або недостатню довжину

{

return input; // Повернення вхідного рядка без змін

}

string[] words = input.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries); // Розбиття рядка на слова

int totalSpaces = desiredLength - input.Length + (words.Length - 1); // Обчислення загальної кількості пробілів

int spaceBetweenWords = totalSpaces / (words.Length - 1); // Обчислення кількості пробілів між словами

int extraSpaces = totalSpaces % (words.Length - 1); // Обчислення додаткових пробілів

string result = words[0]; // Початкове значення результату - перше слово

for (int i = 1; i < words.Length; i++) // Цикл для додавання пробілів між словами

{

int spacesToAdd = spaceBetweenWords + (i <= extraSpaces ? 1 : 0); // Обчислення кількості пробілів для додавання

result += new string(' ', spacesToAdd) + words[i]; // Додавання пробілів та наступного слова до результату

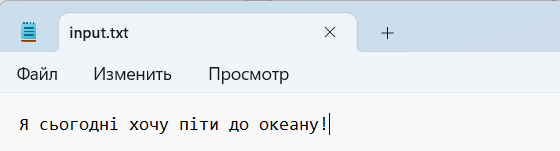
}

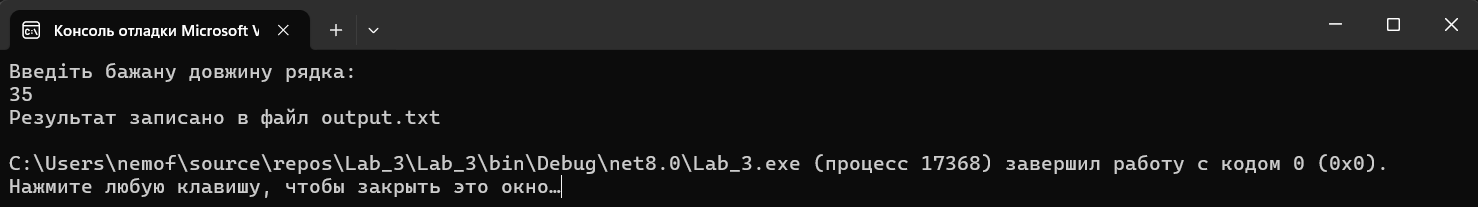
return result; // Повернення вирівняного рядка

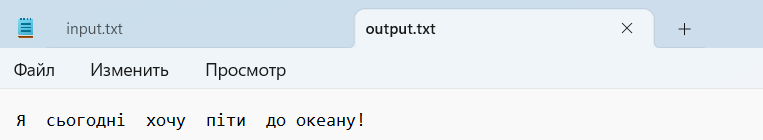
}

}

***Результат роботи програми:***







**Алгоритм побудови коду:**

1. **Підключення бібліотек:**
   * Підключаємо стандартні простори імен System та System.IO для роботи з текстовими файлами.
2. **Основна програма (Main):**
   * Встановлюємо кодування для коректного відображення тексту на кирилиці.
   * Оголошуємо шляхи до вхідного (input.txt) та вихідного файлу (output.txt).
   * Перевіряємо наявність вхідного файлу. Якщо файл не знайдено, виводимо відповідне повідомлення і завершуємо роботу програми.
   * Читаємо вхідний текст з файлу input.txt.
   * Користувачу пропонується ввести бажану довжину рядка, яку потрібно отримати після вирівнювання.
   * Викликається функція вирівнювання рядка, результат якої записується у файл output.txt.
   * Виводимо повідомлення про успішний запис результату в файл.
3. **Функція вирівнювання рядка (JustifyString):**
   * Перевіряється, чи не є вхідний рядок порожнім, а також чи задана довжина не менша за поточну довжину рядка.
   * Розбиваємо вхідний рядок на слова, видаляючи зайві пробіли.
   * Обчислюємо кількість пробілів, які потрібно додати між словами, щоб досягти бажаної довжини рядка.
   * Додаємо необхідну кількість пробілів між словами, рівномірно розподіляючи їх.
   * Повертаємо вирівняний рядок.

**Функціональність:**

1. **Введення/Виведення:** Програма працює з текстовими файлами: зчитує вміст з файлу input.txt, вирівнює рядок до заданої довжини, та записує результат у файл output.txt.
2. **Обробка рядка:** Програма вирівнює рядок за кількістю пробілів між словами так, щоб він займав рівно задану довжину.
3. **Користувацький ввід:** Користувач вводить бажану довжину рядка з консолі, яка використовується для вирівнювання.

**Висновок:**

Ця програма дозволяє ефективно працювати з текстовими файлами, обробляючи текст у них і вирівнюючи рядки за заданою довжиною. Такий підхід корисний у випадках, коли необхідно форматувати текст з різних джерел і виводити його у відформатованому вигляді в інші файли.

***Текст модифікованої програми 2 Завдання:***

using System; // Підключення простору імен System

using System.IO; // Підключення простору імен для роботи з файлами

class Task\_7\_2 // Оголошення класу Task\_7\_2

{

static void Main() // Головний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

string inputFilePath = "input.txt"; // Шлях до вхідного файлу

string outputFilePath = "output.txt"; // Шлях до вихідного файлу

if (!File.Exists(inputFilePath)) // Перевірка наявності вхідного файлу

{

Console.WriteLine("Файл input.txt не знайдено."); // Виведення повідомлення про відсутність файлу

return; // Завершення виконання програми

}

string input = File.ReadAllText(inputFilePath); // Читання вмісту вхідного файлу

Console.Write("Введіть номер слова та кількість слів для видалення: "); // Запит на введення даних

int start = int.Parse(Console.ReadLine()); // Зчитування та перетворення введеного номера слова

int count = int.Parse(Console.ReadLine()); // Зчитування та перетворення кількості слів для видалення

string result = RemoveWords(input, start, count); // Виклик методу для видалення слів

File.WriteAllText(outputFilePath, result); // Запис результату у вихідний файл

Console.WriteLine($"Результат записано в файл {outputFilePath}"); // Виведення повідомлення про успішний запис

Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

}

static string RemoveWords(string input, int start, int count) // Метод для видалення слів

{

if (start < 1 || count < 0) // Перевірка коректності параметрів

{

return "Помилка в параметрах."; // Повернення повідомлення про помилку

}

string[] words = input.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries); // Розбиття тексту на слова

if (start > words.Length) // Перевірка, чи не виходить номер слова за межі масиву

{

return "Помилка в параметрах."; // Повернення повідомлення про помилку

}

int end = Math.Min(start + count - 1, words.Length); // Обчислення кінцевого індексу для видалення

string[] resultWords = new string[words.Length - (end - start + 1)]; // Створення нового масиву для результату

Array.Copy(words, 0, resultWords, 0, start - 1); // Копіювання слів до початку видалення

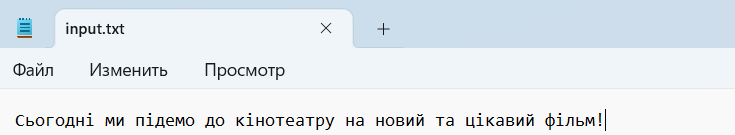
Array.Copy(words, end, resultWords, start - 1, words.Length - end); // Копіювання слів після видалення

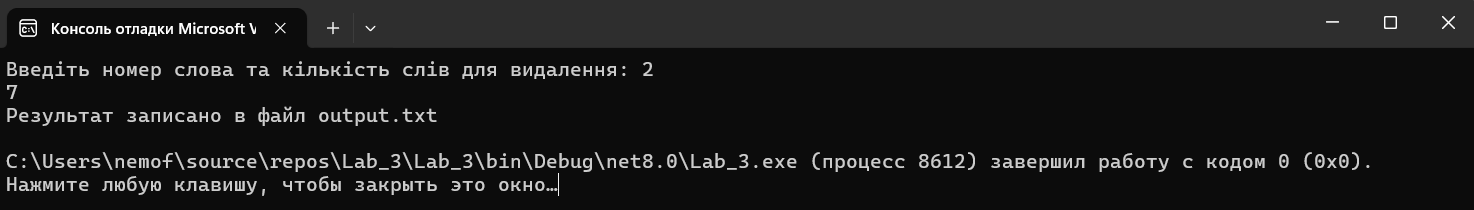
return string.Join(" ", resultWords); // Об'єднання слів у рядок та повернення результату

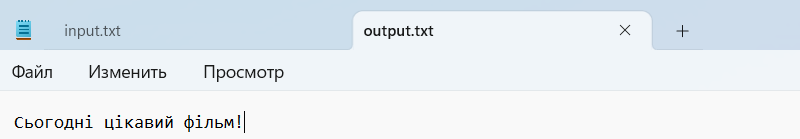
}

}

***Результат роботи програми:***







**Алгоритм побудови коду**

1. **Підключення просторів імен**:
   * System — для основних функцій програми.
   * System.IO — для роботи з файлами.
2. **Оголошення шляху до файлів**:
   * Шлях до вхідного файлу (input.txt) та вихідного файлу (output.txt).
3. **Читання вхідного файлу**:
   * Перевіряється, чи існує файл input.txt.
   * Якщо файл існує, його вміст зчитується за допомогою методу File.ReadAllText.
4. **Введення параметрів користувачем**:
   * Вводиться номер слова для початку видалення та кількість слів, які потрібно видалити.
5. **Видалення слів**:
   * Метод RemoveWords приймає рядок, номер слова та кількість слів для видалення.
   * Текст розбивається на масив слів.
   * Видаляються слова в заданому діапазоні.
6. **Запис результату у вихідний файл**:
   * Після обробки результат записується у файл output.txt.

**Функціональність**

* Програма дозволяє користувачам вводити дані про кількість слів для видалення та початкову позицію через консоль.
* Вхідні дані (початковий текст) зчитуються з файлу input.txt, а результат зберігається у файл output.txt.
* Метод RemoveWords перевіряє коректність введених даних, зокрема:
  + Чи вірно вказаний номер стартового слова.
  + Чи кількість слів для видалення відповідає розміру тексту.

**Висновок**

Програма успішно реалізує обробку текстових файлів для видалення слів з рядка на основі заданих параметрів. Це дає змогу автоматизувати процес обробки тексту з використанням зовнішніх файлів, що робить програму зручною для багаторазового використання без необхідності ручного введення рядків.

***Текст модифікованої програми 3 Завдання:***

using System; // Підключення простору імен System

using System.IO; // Підключення простору імен для роботи з файлами

using System.Collections.Generic; // Підключення простору імен для використання колекцій

class Task\_7\_3

{

static void Main()

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

string inputFilePath = "input.txt"; // Шлях до вхідного файлу

string outputFilePath = "output.txt"; // Шлях до вихідного файлу

if (!File.Exists(inputFilePath)) // Перевірка наявності вхідного файлу

{

Console.WriteLine("Файл input.txt не знайдено."); // Виведення повідомлення про відсутність файлу

return; // Завершення виконання програми

}

string text = File.ReadAllText(inputFilePath); // Читання всього тексту з вхідного файлу

char target = 'Ф'; // Символ, який шукаємо у тексті

List<int> positions = new List<int>(); // Список для збереження позицій символу

for (int i = 0; i < text.Length; i++) // Прохід по всьому тексту

{

if (text[i] == target) // Перевірка, чи символ у тексті відповідає цільовому символу

{

positions.Add(i); // Додавання позиції символу до списку

}

}

using (StreamWriter writer = new StreamWriter(outputFilePath)) // Відкриття вихідного файлу для запису

{

writer.WriteLine("Позиції входження букви 'Ф':"); // Запис заголовку у файл

foreach (int pos in positions) // Прохід по всіх знайдених позиціях

{

writer.WriteLine(pos); // Запис позиції у файл

}

}

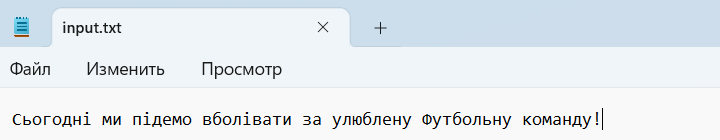
Console.WriteLine($"Результат записано в файл {outputFilePath}"); // Виведення повідомлення про успішний запис

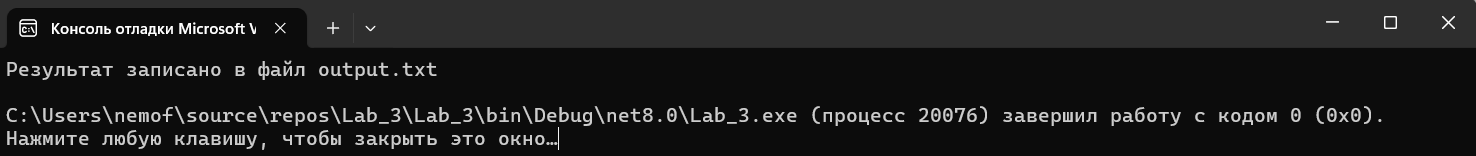
Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

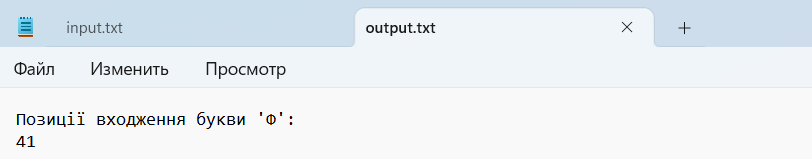
}

}

***Результат роботи програми:***







**Алгоритм побудови коду:**

1. **Підключення просторів імен:**
   * Простір System використовується для основних операцій.
   * Простір System.IO дозволяє працювати з файлами (читання та запис).
   * Простір System.Collections.Generic потрібен для використання списків (List).
2. **Основні етапи програми:**
   * **Налаштування кодування:** Встановлення правильного кодування для коректної роботи з українськими символами.
   * **Перевірка наявності файлу:** Якщо вхідний файл відсутній, програма повідомляє про це і завершується.
   * **Читання файлу:** Програма зчитує текст із вхідного файлу input.txt.
   * **Пошук символу:** Використовується цикл для пошуку позицій символу 'Ф' у рядку та збереження їх у список.
   * **Запис результату:** Позиції символу записуються у вихідний файл output.txt.
   * **Виведення повідомлення:** Користувач отримує повідомлення про успішний запис результатів у файл.
3. **Використання текстових файлів:**
   * Функція File.ReadAllText зчитує весь текст із вхідного файлу.
   * Функція StreamWriter дозволяє записувати дані в новий файл або перезаписувати наявний.

**Функціональність:**

Програма виконує пошук символу 'Ф' у тексті, який зберігається в текстовому файлі, і зберігає всі позиції цього символу в іншому файлі. Програма автоматизує процес обробки тексту, забезпечуючи введення та виведення даних через файли. Це дозволяє легко працювати з великими обсягами даних або використовувати результати в інших програмах.

**Висновок:**

Дана програма демонструє, як можна використовувати текстові файли для введення та виведення даних у C#. Вона дає можливість автоматизувати обробку тексту, забезпечуючи точність і зручність роботи з файлами.

**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.

***Текст модифікованої програми 4 Завдання:***

using System; // Підключення простору імен System

using System.IO; // Підключення простору імен System.IO

class Task\_8\_1 // Оголошення класу Task\_8\_1

{

static void Main() // Головний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

Console.WriteLine("Введіть текстовий рядок:"); // Виведення запиту на введення тексту

string input = Console.ReadLine(); // Зчитування введеного тексту

string result = DoubleLetters(input); // Виклик методу DoubleLetters для обробки тексту

// Запис результату у двійковий файл

using (BinaryWriter writer = new BinaryWriter(File.Open("result4.bin", FileMode.Create))) // Створення двійкового файлу для запису

{

writer.Write(result); // Запис результату у файл

}

// Читання результату з двійкового файлу

using (BinaryReader reader = new BinaryReader(File.Open("result4.bin", FileMode.Open))) // Відкриття двійкового файлу для читання

{

string fileContent = reader.ReadString(); // Зчитування вмісту файлу

Console.WriteLine("Результат з файлу: " + fileContent); // Виведення результату на екран

}

Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

}

static string DoubleLetters(string input) // Метод для подвоєння певних літер у тексті

{

string lettersToDouble = "лнсо"; // Літери, які потрібно подвоїти

string result = ""; // Рядок для збереження результату

foreach (char c in input) // Прохід по кожному символу вхідного рядка

{

result += c; // Додавання символу до результату

if (lettersToDouble.Contains(c)) // Перевірка, чи символ є у списку літер для подвоєння

{

result += c; // Додавання символу ще раз, якщо він є у списку

}

}

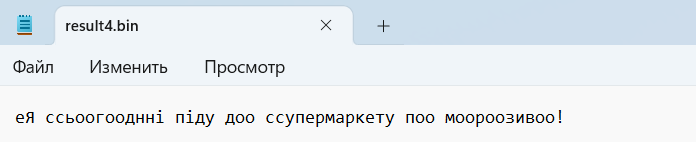
return result; // Повернення результату

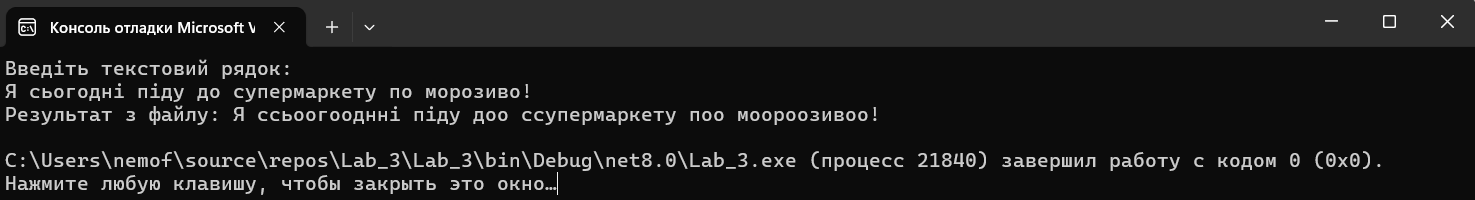
}

}

***Результат роботи програми:***







**Алгоритм побудови коду:**

1. **Зчитування рядка:** Програма запитує текстовий рядок у користувача.
2. **Обробка тексту:** Викликається метод DoubleLetters(), який подвоює символи "л", "н", "с", "о" у тексті.
3. **Запис у двійковий файл:** Оброблений результат записується у файл формату .bin за допомогою класу BinaryWriter.
4. **Читання з файлу:** Програма зчитує записаний результат з файлу за допомогою класу BinaryReader.
5. **Перевірка результату:** Виводиться результат з файлу на консоль для візуальної перевірки.

**Опис функціональності:**

* **Метод DoubleLetters(string input)**: Приймає на вхід рядок і повертає новий рядок, у якому символи "л", "н", "с", "о" подвоюються. Програма проходить через кожен символ вхідного рядка і додає його до результату. Якщо символ належить до списку, він додається вдруге.
* **Запис та читання двійкових файлів**: Файли використовуються для зберігання оброблених даних у двійковому форматі, що дозволяє ефективніше працювати з даними і уникати проблем із кодуванням текстових файлів.

**Висновок:**

Програма успішно реалізує обробку рядка та запис результату у двійковий файл. Читання і виведення результату з файлу дозволяє впевнитись у коректності обробки даних.

***Текст модифікованої програми 5 Завдання:***

using System; // Підключення простору імен System

using System.IO; // Підключення простору імен для роботи з файлами

using System.Text.RegularExpressions; // Підключення простору імен для роботи з регулярними виразами

class Task\_8\_2 // Оголошення класу Task\_8\_2

{

static void Main() // Головний метод програми

{

Console.InputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для вводу

Console.OutputEncoding = System.Text.Encoding.UTF8; // Встановлення кодування для виводу

Console.WriteLine("Введіть рядок:"); // Виведення запиту на введення рядка

string input = Console.ReadLine(); // Зчитування введеного рядка

string pattern = @"ий\b"; // Оголошення шаблону для пошуку

string replacement = "енький"; // Оголошення рядка для заміни

string result = Regex.Replace(input, pattern, replacement); // Заміна шаблону в введеному рядку

// Запис результату у двійковий файл

using (BinaryWriter writer = new BinaryWriter(File.Open("result5.bin", FileMode.Create))) // Відкриття файлу для запису

{

writer.Write(result); // Запис результату у файл

}

// Читання результату з двійкового файлу

using (BinaryReader reader = new BinaryReader(File.Open("result5.bin", FileMode.Open))) // Відкриття файлу для читання

{

string fileContent = reader.ReadString(); // Зчитування вмісту файлу

Console.WriteLine("Результат з файлу: " + fileContent); // Виведення результату на екран

}

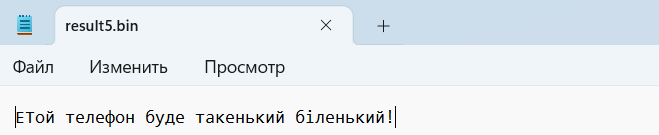
Console.ReadKey(); // Очікування натискання клавіші перед завершенням програми

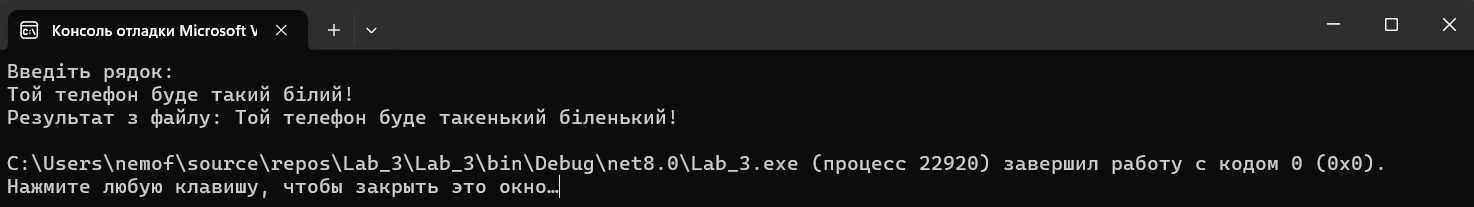
}

}

***Результат роботи програми:***







**Алгоритм побудови коду**

1. **Імпорт бібліотек:**
   * Підключення бібліотек для роботи з файлами (System.IO) та регулярними виразами (System.Text.RegularExpressions).
2. **Введення рядка:**
   * Запит на введення рядка користувачем.
   * Зчитування введеного рядка.
3. **Обробка рядка:**
   * Оголошення регулярного виразу для пошуку та рядка для заміни.
   * Заміна всіх входжень шаблону в рядку за допомогою Regex.Replace.
4. **Запис у двійковий файл:**
   * Відкриття файлу для запису у двійковому форматі.
   * Запис обробленого рядка у файл.
5. **Читання з двійкового файлу:**
   * Відкриття файлу для читання.
   * Зчитування вмісту файлу.
   * Виведення результату на екран консолі.
6. **Завершення програми:**
   * Очікування натискання клавіші для завершення програми.

**Функціональність**

* **Введення Рядка:** Користувач вводить рядок, який буде оброблений.
* **Обробка Рядка:** З використанням регулярних виразів виконується заміна всіх слів, що закінчуються на "ий", на "енький".
* **Запис у Файл:** Результат обробки записується у двійковий файл result5.bin.
* **Читання з Файлу:** Зчитується вміст двійкового файлу та виводиться на екран для перевірки коректності обробки.

**Висновок**

Модифікація завдання за допомогою двійкових файлів дозволяє не тільки зберігати результати обробки у компактному форматі, але і перевіряти їх коректність шляхом виведення вмісту файлу на екран. Використання двійкових файлів забезпечує ефективний спосіб роботи з даними, який може бути корисним для зберігання та обробки великого обсягу інформації в майбутніх проектах.

**Загальний Висновок**

У ході виконання лабораторної роботи №3 було розглянуто та практично застосовано різноманітні методи обробки рядків та операції введення/виведення у мові C#. Були реалізовані завдання, пов'язані з маніпуляціями над рядками, включаючи вирівнювання, видалення слів, пошук та заміну символів, а також роботу з текстовими та двійковими файлами.

**Фрагменти коду C#, що ілюструють деякі з цих операцій:**

* **Робота з рядками:**
  + Розбиття рядка на слова:

*string text = "Це приклад рядка для розбиття.";*

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

* + Пошук підрядка:

*string text = "Знайдіть слово в цьому тексті.";*

*int index = text.IndexOf("слово");*

* **Файлове введення/виведення:**
  + Читання з текстового файлу:

*string filePath = "шлях/до/файлу.txt";*

*string text = File.ReadAllText(filePath);*

* + Запис у текстовий файл:

*string filePath = "шлях/до/файлу.txt";*

*string textToWrite = "Новий текст для запису.";*

*File.WriteAllText(filePath, textToWrite);*

Під час виконання роботи були закріплені навички використання регулярних виразів для ефективного пошуку та заміни шаблонів у тексті. Також було продемонстровано, як застосовувати різні методи роботи з рядками, такі як розбиття на слова, підрахунок символів, пошук найдовшого слова та інші.

Важливим аспектом роботи було використання файлового введення/виведення. Було реалізовано читання та запис даних у текстові та двійкові файли, що дозволяє зберігати та обробляти великі обсяги текстової інформації.

У результаті виконання лабораторної роботи було досягнуто поставленої мети: набуто практичних навичок у написанні програм з використанням рядків та застосування файлових операцій у мові C#. Це дозволить ефективно вирішувати різноманітні завдання обробки текстової інформації у майбутніх проектах.