**Лабораторна робота №3. Робота із рядками та файлове введення / виведення у мові C# .**

**Виконавець: Лазоренко М.Д.**

**Група виконавця: КН-922в**

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

**Індивідуальні завдання**

**Варіант №9**

**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#.

*Лабораторна робота 12:*

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання, виконуваного функцією** |
| 9 | Визначає в рядку номер позиції слова із заданим номером |

***Текст програми:***

using System;

class task01

{

// Функція для визначення позиції слова в рядку

static int FindWordPosition(string input, int wordNumber)

{

if (wordNumber <= 0)

{

Console.WriteLine("Некоректний номер слова.");

return -1;

}

int currentWord = 0; // Лічильник слів

int position = -1; // Позиція потрібного слова

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

{

// Якщо знайдено початок слова

if ((i == 0 || input[i - 1] == ' ') && input[i] != ' ')

{

currentWord++;

if (currentWord == wordNumber)

{

position = i;

break;

}

}

}

if (position == -1)

{

Console.WriteLine("Слово з таким номером не знайдено.");

}

return position;

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

Console.WriteLine("Введіть рядок:");

string input = Console.ReadLine();

Console.WriteLine("Введіть номер слова:");

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

int position = FindWordPosition(input, wordNumber);

if (position != -1)

{

Console.WriteLine($"Позиція {wordNumber}-го слова: {position}");

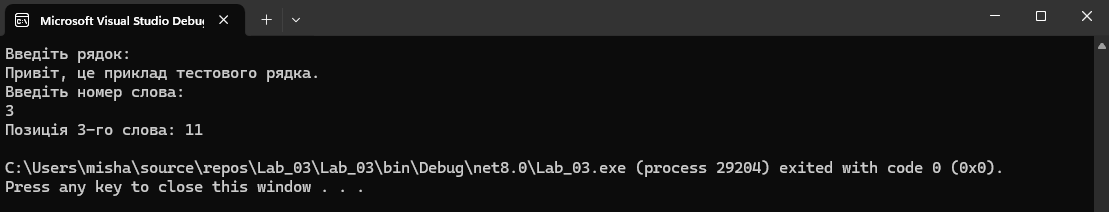
}

Console.ReadKey();

}

}

***Результат роботи програми:***



Ця програма на C# призначена для знаходження позиції слова в рядку за заданим номером. Вона дозволяє користувачу ввести текстовий рядок і вказати, яке слово він хоче знайти, після чого програма повертає позицію цього слова в рядку.

**Основні етапи виконання програми:**

1. **Введення рядка**: Користувач вводить текстовий рядок, у якому будуть виконуватись пошуки.
2. **Введення номера слова**: Користувач вводить номер слова, позицію якого він хоче знайти. Нумерація слів починається з 1.
3. **Функція FindWordPosition**:
   * Програма перевіряє, чи введений номер слова є коректним (більше нуля).
   * Виконується перебір символів рядка, щоб визначити позицію початку кожного слова.
   * При знаходженні потрібного слова функція повертає його позицію в рядку. Якщо слово не знайдено, виводиться відповідне повідомлення.
4. **Виведення результату**: Якщо слово знайдено, програма виводить позицію слова у форматі, що інформує користувача.

**Основні функції програми:**

* **Перевірка коректності вводу**: Програма перевіряє, чи введений номер слова є дійсним.
* **Підрахунок слів**: Вона здатна підрахувати позиції слів, навіть якщо рядок містить пробіли.
* **Виведення результату**: Інформативні повідомлення для користувача про позицію слова або про те, що слово не знайдено.

**Завдання 2**

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

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання, виконуваного функцією** |
| 22 | Переводить рядок, що містить десяткове подання цілого числа, в рядок, що містить його шістнадцяткове подання |

***Текст програми:***

using System;

class task02

{

// Функція для переведення десяткового рядка в шістнадцятковий формат

static string DecimalToHexadecimal(string decimalString)

{

try

{

// Перетворюємо рядок на ціле число

int decimalNumber = int.Parse(decimalString);

// Перетворюємо число на шістнадцяткове представлення

string hexString = decimalNumber.ToString("X");

return hexString;

}

catch (FormatException)

{

return "Помилка: Введений рядок не є коректним числом.";

}

catch (OverflowException)

{

return "Помилка: Число занадто велике для обробки.";

}

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

Console.WriteLine("Введіть десяткове число:");

string input = Console.ReadLine();

// Викликаємо функцію для перетворення

string hexResult = DecimalToHexadecimal(input);

// Виводимо результат

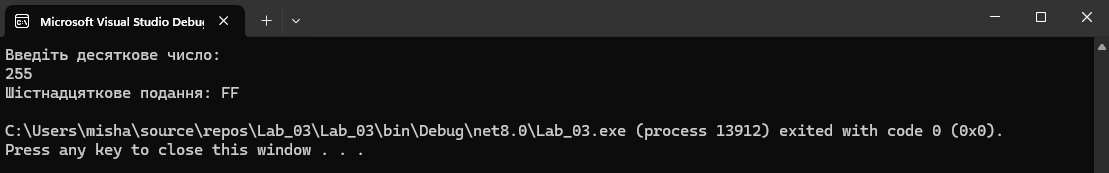
Console.WriteLine($"Шістнадцяткове подання: {hexResult}");

Console.ReadKey();

}

}

***Результат роботи програми:***



Ця програма на C# призначена для перетворення десяткового числа, введеного користувачем, в його шістнадцяткове представлення. Вона реалізує просту функцію для обробки введеного рядка та повертає результат у вигляді шістнадцяткового числа.

**Основні етапи виконання програми:**

1. **Введення десяткового числа**: Програма запитує у користувача введення десяткового числа у вигляді рядка.
2. **Функція DecimalToHexadecimal**:
   * Програма намагається перетворити введений рядок у ціле число за допомогою int.Parse.
   * Якщо перетворення успішне, число конвертується у шістнадцятковий формат за допомогою методу ToString("X").
   * У разі помилок, таких як некоректний формат рядка або переповнення, програма обробляє винятки і повертає відповідні повідомлення про помилку.
3. **Виведення результату**: Після перетворення програма виводить результат у шістнадцятковому форматі, або повідомлення про помилку, якщо таке сталося.

**Основні функції програми:**

* **Перетворення формату**: Програма здатна конвертувати десяткові числа у шістнадцяткові формати.
* **Обробка помилок**: Включає обробку можливих помилок під час перетворення, забезпечуючи зручність використання.
* **Виведення результатів**: Інформує користувача про результати перетворення.

**Завдання 3**

У текстовому рядку підрахувати кількість букв 'м'.

***Текст програми:***

using System;

class task03

{

// Функція для підрахунку кількості літер 'м'

static int CountLetterM(string input)

{

int count = 0;

// Перебираємо кожен символ у рядку

foreach (char c in input)

{

// Перевіряємо, чи є символ буквою 'м' (враховуємо як малу, так і велику літеру)

if (c == 'м' || c == 'М')

{

count++;

}

}

return count;

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

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

string input = Console.ReadLine();

// Викликаємо функцію для підрахунку

int count = CountLetterM(input);

// Виводимо результат

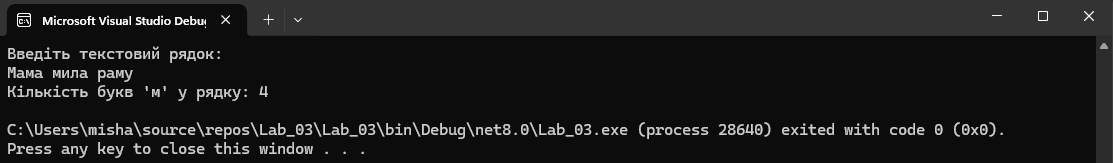
Console.WriteLine($"Кількість букв 'м' у рядку: {count}");

Console.ReadKey();

}

}

***Результат роботи програми:***



Ця програма на C# призначена для підрахунку кількості літер 'м' (як малих, так і великих) у текстовому рядку, введеному користувачем. Програма реалізує просту функцію для обробки введеного рядка та повертає кількість знайдених букв.

**Основні етапи виконання програми:**

1. **Введення текстового рядка**: Програма запитує у користувача введення текстового рядка.
2. **Функція CountLetterM**:
   * Ініціалізується змінна count для підрахунку кількості літер 'м'.
   * Програма перебирає кожен символ введеного рядка за допомогою циклу foreach.
   * Для кожного символу перевіряється, чи є він літерою 'м' (як мале, так і велике 'М').
   * Якщо символ є літерою 'м', лічильник count збільшується на одиницю.
3. **Виведення результату**: Після завершення перебору символів програма виводить результат у вигляді кількості букв 'м', знайдених у рядку.

**Основні функції програми:**

* **Підрахунок символів**: Програма здатна підраховувати кількість конкретних символів у тексті.
* **Відображення результатів**: Інформує користувача про результати підрахунку.

**Завдання 4**

В текстовому рядку замінити лапки двома комами.

***Текст програми:***

using System;

class task04

{

// Функція для заміни лапок на дві коми

static string ReplaceQuotesWithCommas(string input)

{

// Замінюємо прямі лапки '"' на дві коми

string result = input.Replace("\"", ",,");

return result;

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

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

string input = Console.ReadLine();

// Викликаємо функцію для заміни лапок

string modifiedInput = ReplaceQuotesWithCommas(input);

// Виводимо результат

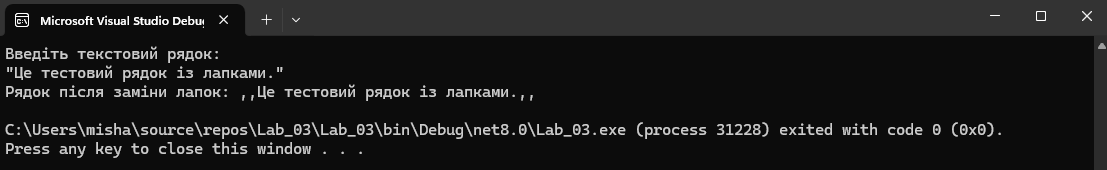
Console.WriteLine($"Рядок після заміни лапок: {modifiedInput}");

Console.ReadKey();

}

}

***Результат роботи програми:***



Ця програма на C# призначена для заміни всіх прямих лапок у введеному текстовому рядку на дві коми. Вона реалізує просту функцію, яка здійснює цю заміну, і виводить результат на екран.

**Основні етапи виконання програми:**

1. **Введення текстового рядка**: Програма запитує у користувача введення текстового рядка, в якому можуть міститися прямі лапки (").
2. **Функція ReplaceQuotesWithCommas**:
   * Приймає вхідний рядок input як параметр.
   * Використовує метод Replace для заміни всіх входжень прямої лапки на дві коми (,,).
   * Повертає модифікований рядок.
3. **Виведення результату**: Після виконання функції програма виводить на екран модифікований рядок, у якому всі лапки замінено на коми.

**Основні функції програми:**

* **Заміну символів**: Програма демонструє, як замінювати один символ на інший у рядку, що може бути корисним для обробки текстових даних.
* **Відображення результатів**: Інформує користувача про результати обробки введеного рядка.

**Завдання 5**

Знайти кількість слів, що починаються на букву С.

***Текст програми:***

using System;

class task05

{

// Функція для підрахунку кількості слів, що починаються на букву 'С' або 'с'

static int CountWordsStartingWithS(string input)

{

int count = 0;

// Розбиваємо рядок на слова за пробілами

string[] words = input.Split(new[] { ' ', '\t', '\n' }, StringSplitOptions.RemoveEmptyEntries);

// Перевіряємо кожне слово

foreach (string word in words)

{

if (word.Length > 0 && (word[0] == 'С' || word[0] == 'с'))

{

count++;

}

}

return count;

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

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

string input = Console.ReadLine();

// Викликаємо функцію для підрахунку слів, що починаються на 'С' або 'с'

int count = CountWordsStartingWithS(input);

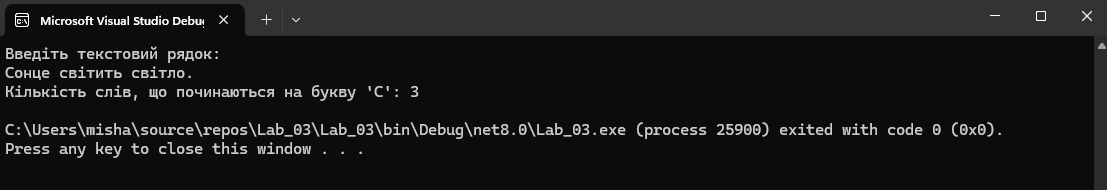
// Виводимо результат

Console.WriteLine($"Кількість слів, що починаються на букву 'С': {count}");

Console.ReadKey();

}

}

***Результат роботи програми:*** ******

Ця програма на C# призначена для підрахунку кількості слів у введеному текстовому рядку, які починаються з букви 'С' або 'с'. Вона реалізує функцію, яка обробляє текстовий рядок, розбиваючи його на слова і підраховуючи відповідні елементи.

**Основні етапи виконання програми:**

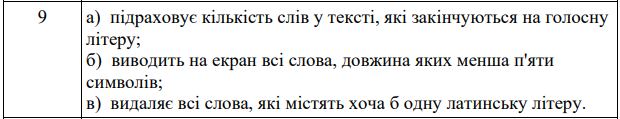
1. **Введення текстового рядка**: Програма запитує у користувача введення текстового рядка, у якому можуть міститися слова, що починаються на букву 'С' або 'с'.
2. **Функція CountWordsStartingWithS**:
   * Приймає вхідний рядок input як параметр.
   * Використовує метод Split для розбиття рядка на масив слів, відділених пробілами, табуляцією та новими рядками.
   * Ініціалізує лічильник count для підрахунку слів, що починаються з 'С' або 'с'.
   * Перебирає кожне слово в масиві:
     + Перевіряє, чи перша буква слова є 'С' або 'с', і, у разі відповідності, збільшує лічильник count.
3. **Виведення результату**: Після виконання функції програма виводить на екран кількість слів, що починаються на букву 'С'.

**Основні функції програми:**

* **Підрахунок слів**: Програма демонструє, як розбивати рядки на слова і підраховувати їх за певною ознакою, що може бути корисним при обробці текстових даних.
* **Відображення результатів**: Інформує користувача про результати обробки введеного рядка.

**Завдання 6**

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

******

***Текст програми:***

using System;

using System.Linq;

using System.Text.RegularExpressions;

class task06

{

// Функція для перевірки, чи закінчується слово на голосну

static bool EndsWithVowel(string word)

{

char[] vowels = { 'а', 'е', 'є', 'и', 'і', 'ї', 'о', 'у', 'ю', 'я', 'А', 'Е', 'Є', 'И', 'І', 'Ї', 'О', 'У', 'Ю', 'Я' };

return vowels.Contains(word.Last());

}

// Функція для перевірки, чи містить слово латинські літери

static bool ContainsLatinLetters(string word)

{

return word.Any(c => (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z'));

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

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

string input = Console.ReadLine();

// Розбиваємо текст на слова за пробілами

string[] words = input.Split(new[] { ' ', '\t', '\n' }, StringSplitOptions.RemoveEmptyEntries);

// а) Підраховуємо кількість слів, що закінчуються на голосну літеру

int vowelEndingCount = words.Count(EndsWithVowel);

Console.WriteLine($"Кількість слів, що закінчуються на голосну літеру: {vowelEndingCount}");

// б) Виводимо всі слова, довжина яких менша за 5 символів

Console.WriteLine("Слова довжиною менше 5 символів:");

foreach (string word in words.Where(w => w.Length < 5))

{

Console.WriteLine(word);

}

// в) Видаляємо всі слова, що містять латинські літери

string[] filteredWords = words.Where(w => !ContainsLatinLetters(w)).ToArray();

string result = string.Join(" ", filteredWords);

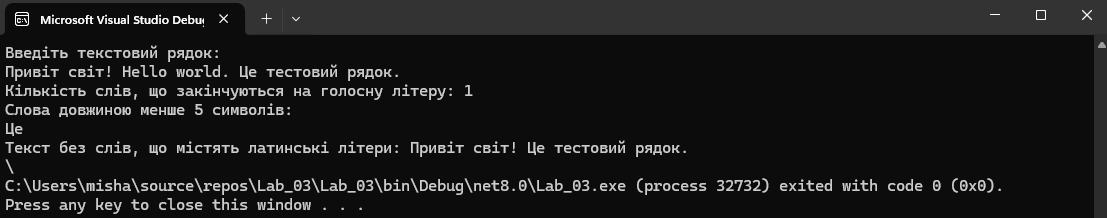
Console.WriteLine($"Текст без слів, що містять латинські літери: {result}");

Console.ReadKey();

}

}

***Результат роботи програми:***



Ця програма на C# виконує аналіз введеного текстового рядка, розбиваючи його на слова і обробляючи їх відповідно до кількох критеріїв. Вона реалізує три основні функції, які дозволяють підраховувати слова, що закінчуються на голосні, виводити короткі слова, а також фільтрувати слова з латинськими літерами.

**Основні етапи виконання програми:**

1. **Введення текстового рядка**: Програма запитує у користувача введення текстового рядка.
2. **Розбиття рядка на слова**: Введений рядок розбивається на масив слів, використовуючи метод Split, з урахуванням пробілів, табуляцій та нових рядків.
3. **Обробка слів**:
   * **Підрахунок слів, що закінчуються на голосну літеру**:
     + Використовується функція EndsWithVowel, яка перевіряє, чи остання буква слова є голосною. Голосні літери української мови визначені в масиві vowels.
     + Підрахунок слів, що закінчуються на голосну, здійснюється за допомогою методу Count на масиві слів.
   * **Виведення коротких слів**:
     + Програма виводить всі слова, довжина яких менша за 5 символів, використовуючи метод Where для фільтрації.
   * **Фільтрація слів, що містять латинські літери**:
     + Використовується функція ContainsLatinLetters, яка перевіряє наявність латинських літер у слові.
     + Всі слова, що містять латинські літери, видаляються з масиву, а результуючий текст знову об'єднується в рядок за допомогою string.Join.
4. **Виведення результатів**:
   * Кількість слів, що закінчуються на голосну.
   * Перелік слів довжиною менше 5 символів.
   * Текст без слів, що містять латинські літери.

**Основні функції програми:**

* **Підрахунок голосних закінчень**: Дозволяє швидко оцінити, скільки слів у тексті закінчуються на голосну.
* **Фільтрація коротких слів**: Допомагає виявити короткі слова, що може бути корисно для стилістичного аналізу.
* **Фільтрація латинських слів**: Забезпечує можливість обробки текстів, що містять латинські символи, дозволяючи користувачу отримати чистий український текст.

**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#.

***Task07\_1:***

using System;

using System.IO;

class task07\_1

{

// Функція для визначення позиції слова в рядку

static int FindWordPosition(string input, int wordNumber)

{

if (wordNumber <= 0)

{

return -1;

}

int currentWord = 0;

int position = -1;

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

{

if ((i == 0 || input[i - 1] == ' ') && input[i] != ' ')

{

currentWord++;

if (currentWord == wordNumber)

{

position = i;

break;

}

}

}

return position;

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

// Читаємо вхідний рядок з файлу

string input = File.ReadAllText("input.txt");

// Читаємо номер слова з файлу

int wordNumber = int.Parse(File.ReadAllText("word\_number.txt"));

// Знаходимо позицію слова

int position = FindWordPosition(input, wordNumber);

// Записуємо результат у файл

string result = (position != -1) ? $"Позиція {wordNumber}-го слова: {position}" : "Слово з таким номером не знайдено.";

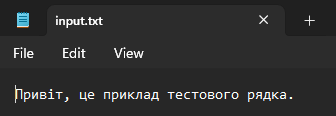
File.WriteAllText("output.txt", result);

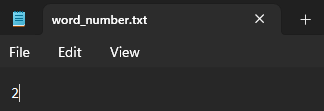
Console.ReadKey();

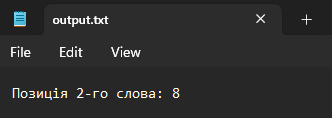
}

}

***Результат роботи програми:***

******

****

****

Ця програма на C# призначена для знаходження позиції конкретного слова в текстовому рядку, зчитуючи вхідні дані з файлів. Вона виконує кілька ключових завдань: зчитування тексту та номера слова з файлів, визначення позиції вказаного слова та запис результату в інший файл.

**Основні етапи виконання програми:**

1. **Зчитування вхідних даних**:
   * Програма читає текстовий рядок з файлу input.txt, використовуючи метод File.ReadAllText.
   * Номер слова, позицію якого потрібно знайти, зчитується з файлу word\_number.txt і перетворюється на ціле число.
2. **Визначення позиції слова**:
   * Використовується функція FindWordPosition, яка приймає два параметри: рядок input і номер слова wordNumber.
   * Програма перевіряє, чи є wordNumber дійсним (позитивним) значенням. Якщо ні, повертається -1.
   * У циклі перебираються символи рядка. Кожного разу, коли виявляється початок нового слова (умова: символ перед ним - пробіл або це перший символ рядка), лічильник слів (currentWord) збільшується. Якщо номер поточного слова дорівнює запитаному, зберігається його позиція.
   * Позиція слова повертається функцією.
3. **Запис результату**:
   * Програма формує результат на основі отриманої позиції: якщо позиція не дорівнює -1, виводиться позиція слова у форматі Позиція {wordNumber}-го слова: {position}. Якщо слово не знайдено, повертається повідомлення про це.
   * Результат записується у файл output.txt за допомогою File.WriteAllText.
4. **Завершення програми**:
   * Програма чекає натискання клавіші для завершення.

***Task07\_2:***

using System;

using System.IO;

class task07\_2

{

// Функція для переведення десяткового рядка в шістнадцятковий формат

static string DecimalToHexadecimal(string decimalString)

{

try

{

int decimalNumber = int.Parse(decimalString);

string hexString = decimalNumber.ToString("X");

return hexString;

}

catch (FormatException)

{

return "Помилка: Введений рядок не є коректним числом.";

}

catch (OverflowException)

{

return "Помилка: Число занадто велике для обробки.";

}

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

// Читаємо десяткове число з файлу

string input = File.ReadAllText("input.txt");

// Перетворюємо на шістнадцяткове

string hexResult = DecimalToHexadecimal(input);

// Записуємо результат у файл

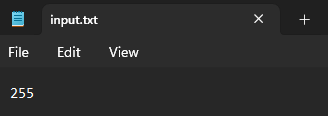
File.WriteAllText("output.txt", $"Шістнадцяткове подання: {hexResult}");

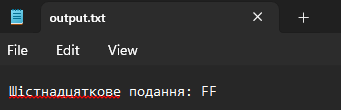
Console.ReadKey();

}

}

***Результат роботи програми:***

****

****

Ця програма на C# призначена для перетворення десяткового числа, яке зчитується з файлу, у його шістнадцяткове представлення. Вона реалізує обробку помилок, щоб обробити некоректні або занадто великі числа, а також записує результат у інший файл.

**Основні етапи виконання програми:**

1. **Зчитування вхідних даних**:
   * Програма зчитує рядок з файлу input.txt, який має містити десяткове число, використовуючи метод File.ReadAllText.
2. **Перетворення десяткового числа у шістнадцяткове**:
   * Використовується функція DecimalToHexadecimal, яка приймає рядок decimalString як параметр.
   * Програма намагається перетворити цей рядок у ціле число (int) за допомогою int.Parse.
   * Якщо перетворення успішне, число конвертується у шістнадцятковий формат за допомогою методу ToString("X"), який повертає рядок у шістнадцятковому форматі.
   * Якщо виникає помилка, наприклад, рядок не є коректним числом (FormatException) або число занадто велике для обробки (OverflowException), програма повертає відповідне повідомлення про помилку.
3. **Запис результату**:
   * Результат перетворення (шістнадцяткове представлення або повідомлення про помилку) записується у файл output.txt.
   * Якщо виникла помилка, то буде записано текст помилки.
4. **Завершення програми**:
   * Програма чекає натискання клавіші для завершення.

***Task07\_3:***

using System;

using System.IO;

class task07\_3

{

// Функція для підрахунку кількості літер 'м'

static int CountLetterM(string input)

{

int count = 0;

// Перебираємо кожен символ у рядку

foreach (char c in input)

{

// Перевіряємо, чи є символ буквою 'м' (враховуємо як малу, так і велику літеру)

if (c == 'м' || c == 'М')

{

count++;

}

}

return count;

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

// Читаємо вхідний текст з файлу

string input = File.ReadAllText("input.txt");

// Викликаємо функцію для підрахунку

int count = CountLetterM(input);

// Записуємо результат у файл

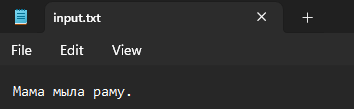
File.WriteAllText("output.txt", $"Кількість букв 'м' у рядку: {count}");

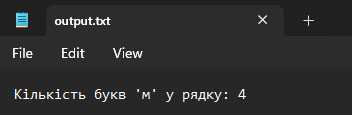
Console.ReadKey();

}

}

***Результат роботи програми:***

****

****

Ця програма на C# призначена для підрахунку кількості літер 'м' (як малих, так і великих) у текстовому рядку, зчитаному з файлу. Результат підрахунку записується у інший файл.

**Основні етапи виконання програми:**

1. **Зчитування вхідних даних**:
   * Програма зчитує текстовий рядок з файлу input.txt за допомогою методу File.ReadAllText.
2. **Підрахунок літер 'м'**:
   * Використовується функція CountLetterM, яка приймає рядок input як параметр.
   * Програма перебирає кожен символ у рядку за допомогою циклу foreach.
   * Якщо символ є літерою 'м' або 'М', лічильник count збільшується на одиницю.
3. **Запис результату**:
   * Після підрахунку програма записує результат у файл output.txt.
4. **Завершення програми**:
   * Програма чекає натискання клавіші для завершення.

**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.

***Task08\_1:***

using System;

using System.IO;

class task08\_1

{

// Функція для заміни лапок на дві коми

static string ReplaceQuotesWithCommas(string input)

{

// Замінюємо прямі лапки '"' на дві коми

string result = input.Replace("\"", ",,");

return result;

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

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

string input = Console.ReadLine();

// Викликаємо функцію для заміни лапок

string modifiedInput = ReplaceQuotesWithCommas(input);

// Записуємо результат у двійковий файл

using (FileStream fs = new FileStream("output.dat", FileMode.Create))

using (BinaryWriter writer = new BinaryWriter(fs))

{

writer.Write(modifiedInput);

}

// Читаємо вміст двійкового файлу і виводимо на екран

using (FileStream fs = new FileStream("output.dat", FileMode.Open))

using (BinaryReader reader = new BinaryReader(fs))

{

string fileContent = reader.ReadString();

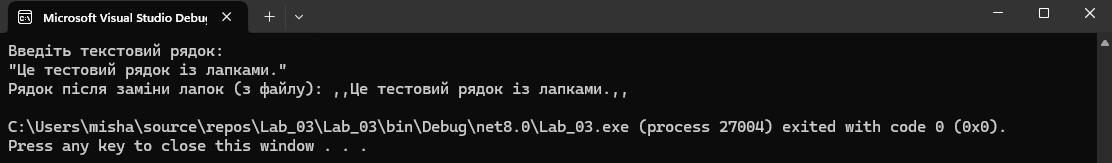
Console.WriteLine($"Рядок після заміни лапок (з файлу): {fileContent}");

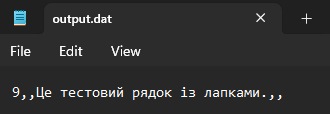
}

Console.ReadKey();

}

}

***Результат роботи програми:***



Ця програма на C# призначена для заміни всіх лапок у введеному текстовому рядку на дві коми та збереження модифікованого рядка у двійковому файлі. Після запису програма також читає вміст цього файлу та виводить його на екран.

**Основні етапи виконання програми:**

1. **Зчитування вхідних даних**:
   * Програма запитує у користувача введення текстового рядка через консоль.
2. **Заміна лапок**:
   * Використовується функція ReplaceQuotesWithCommas, яка приймає рядок input як параметр.
   * Внутрішньо метод Replace замінює всі прямі лапки (") на дві коми (,).
3. **Запис результату у двійковий файл**:
   * Програма відкриває файл output.dat у режимі створення (при цьому, якщо файл уже існує, він буде перезаписаний).
   * Використовується BinaryWriter для запису модифікованого рядка в двійковий файл.
4. **Читання вмісту з двійкового файлу**:
   * Програма відкриває файл output.dat у режимі читання.
   * Використовується BinaryReader для зчитування рядка з двійкового файлу.
5. **Виведення результату на екран**:
   * Програма виводить на екран модифікований рядок, зчитаний з файлу.
6. **Завершення програми**:
   * Програма чекає натискання клавіші для завершення.

***Task08\_2:***

using System;

using System.IO;

class task08\_2

{

// Функція для підрахунку кількості слів, що починаються на букву 'С' або 'с'

static int CountWordsStartingWithS(string input)

{

int count = 0;

// Розбиваємо рядок на слова за пробілами

string[] words = input.Split(new[] { ' ', '\t', '\n' }, StringSplitOptions.RemoveEmptyEntries);

// Перевіряємо кожне слово

foreach (string word in words)

{

if (word.Length > 0 && (word[0] == 'С' || word[0] == 'с'))

{

count++;

}

}

return count;

}

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

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

string input = Console.ReadLine();

// Викликаємо функцію для підрахунку слів, що починаються на 'С' або 'с'

int count = CountWordsStartingWithS(input);

// Записуємо результат у двійковий файл

using (FileStream fs = new FileStream("output.dat", FileMode.Create))

using (BinaryWriter writer = new BinaryWriter(fs))

{

writer.Write($"Кількість слів, що починаються на букву 'С': {count}");

}

// Читаємо вміст двійкового файлу і виводимо на екран

using (FileStream fs = new FileStream("output.dat", FileMode.Open))

using (BinaryReader reader = new BinaryReader(fs))

{

string fileContent = reader.ReadString();

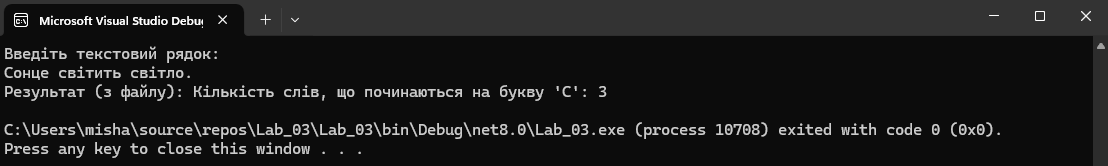
Console.WriteLine($"Результат (з файлу): {fileContent}");

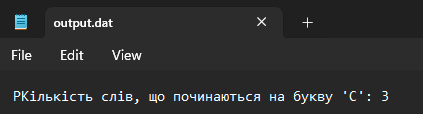
}

Console.ReadKey();

}

}

***Результат роботи програми:*** ******



Ця програма на C# призначена для підрахунку кількості слів, що починаються на букву 'С' або 'с', в введеному текстовому рядку. Результат записується у двійковий файл, а потім читається з цього файлу та виводиться на екран.

**Основні етапи виконання програми:**

1. **Зчитування вхідних даних**:
   * Програма запитує у користувача введення текстового рядка через консоль.
2. **Підрахунок слів, що починаються на 'С' або 'с'**:
   * Використовується функція CountWordsStartingWithS, яка приймає рядок input як параметр.
   * Рядок розбивається на слова за допомогою методу Split, який використовує пробіли, табуляції та нові рядки як роздільники.
   * Кожне слово перевіряється, чи починається з 'С' або 'с'. Якщо так, лічильник count збільшується.
3. **Запис результату у двійковий файл**:
   * Програма відкриває файл output.dat у режимі створення (при цьому, якщо файл уже існує, він буде перезаписаний).
   * Використовується BinaryWriter для запису рядка, який містить кількість слів, що починаються на 'С', у двійковий файл.
4. **Читання вмісту з двійкового файлу**:
   * Програма відкриває файл output.dat у режимі читання.
   * Використовується BinaryReader для зчитування рядка з двійкового файлу.
5. **Виведення результату на екран**:
   * Програма виводить на екран результат, зчитаний з файлу.
6. **Завершення програми**:
   * Програма чекає натискання клавіші для завершення.

**Висновок**

У процесі виконання лабораторної роботи №3 було досягнуто основної мети – набуто практичних навичок у роботі з рядками та файловими операціями на мові програмування C#. Було розроблено програми, які ефективно маніпулюють текстовими даними, виконують читання та запис інформації у файли. Також вивчено методи обробки рядків, що дозволяє виконувати пошук та зміну текстових даних, а також обробку помилок при роботі з файлами. Отримані знання стануть корисними для подальшого вдосконалення в програмуванні, особливо в роботі з текстовою інформацією та файлами у C#.