**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №4

з дисципліни «Основи програмної інженерії» на тему:

«Робота з файлами»

**Виконав:**

Студент групи КНТ-122 О. А. Онищенко

**Прийняли:**

Викладач: О. І. Качан

Викладач: Т. І. Каплієнко

2023

Робота з файлами

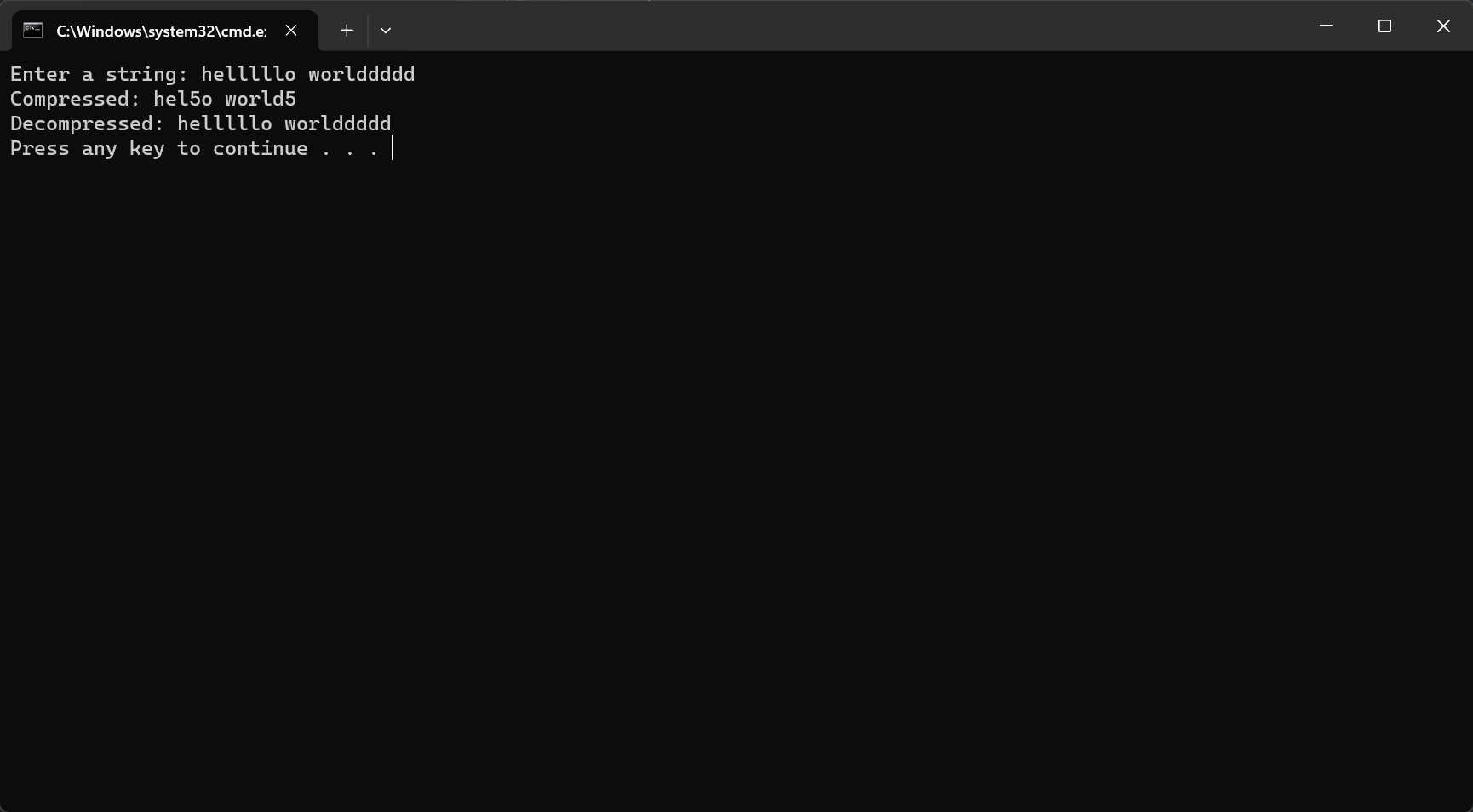
Мета роботи

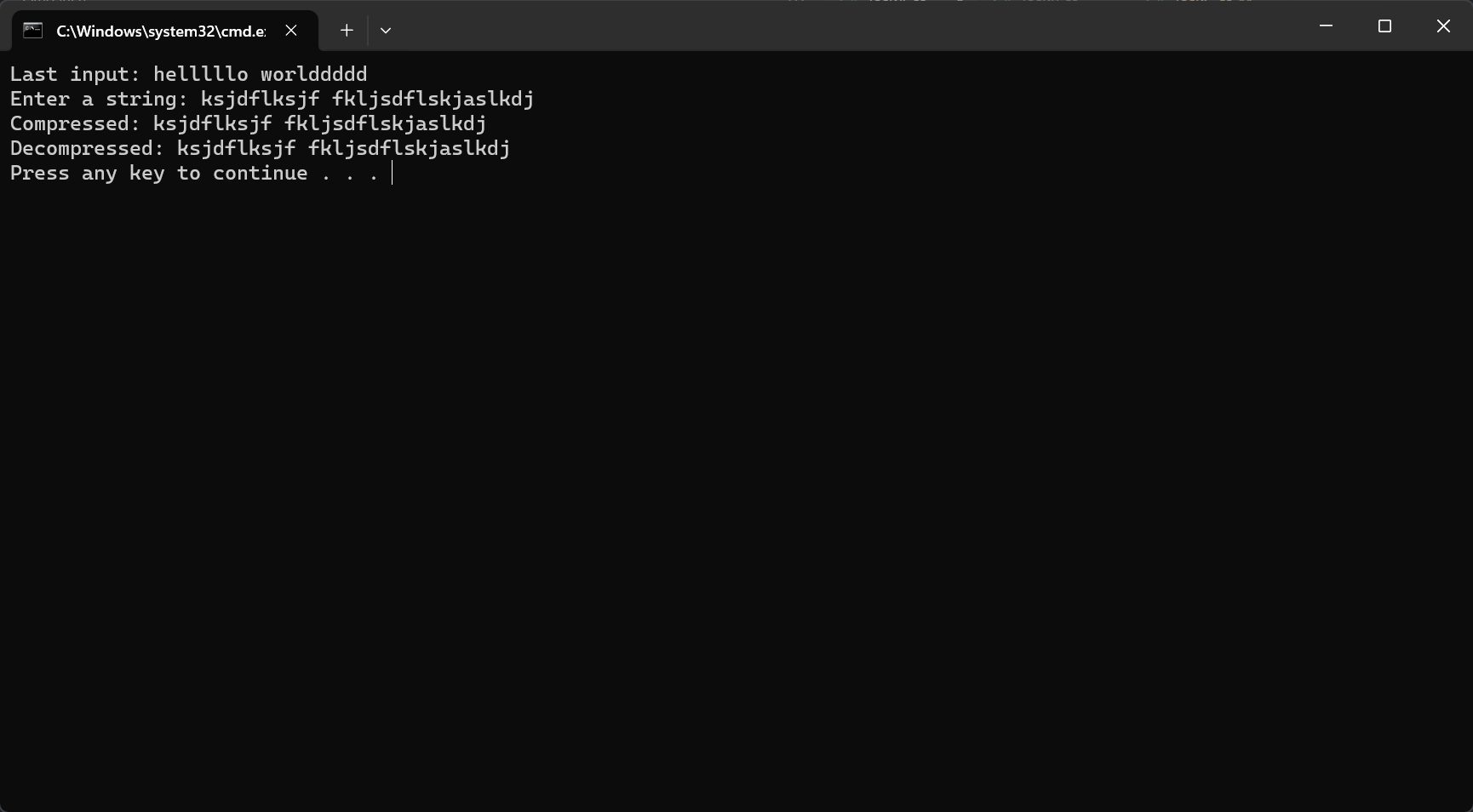
Вивчити основні принципи роботи з файлами у середовищі Visual Studio C#.

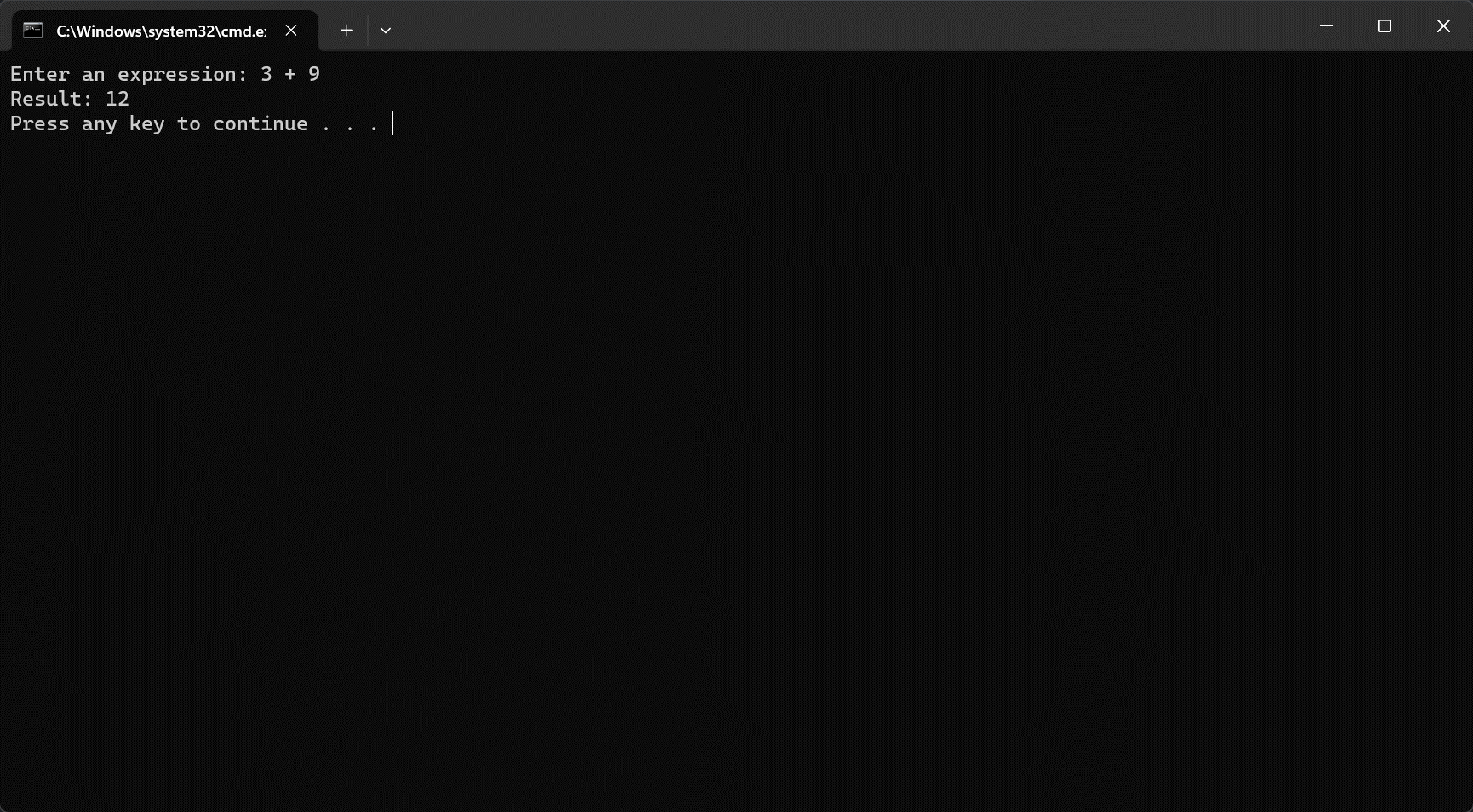
Завдання до роботи

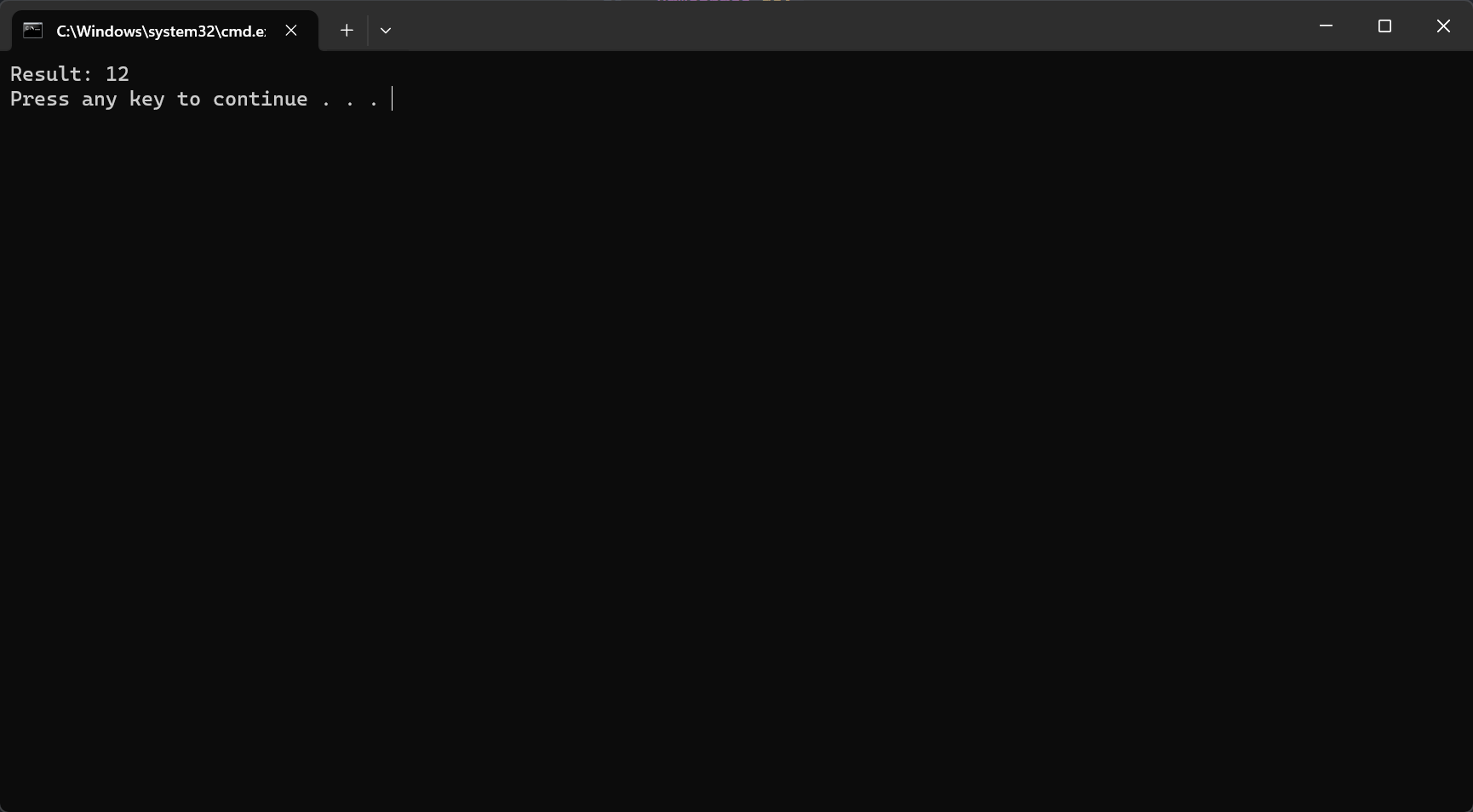
Додати зберігання та завантаження останнього вигляду програми у файлі. Порядок зберігання та завантаження інформації у файли реалізувати у наступний спосіб: зберігати кожну сесію роботи програми, нову сесію дописувати у кінець файлу. Під час роботи програми реалізувати завантаження збереженої сесії.

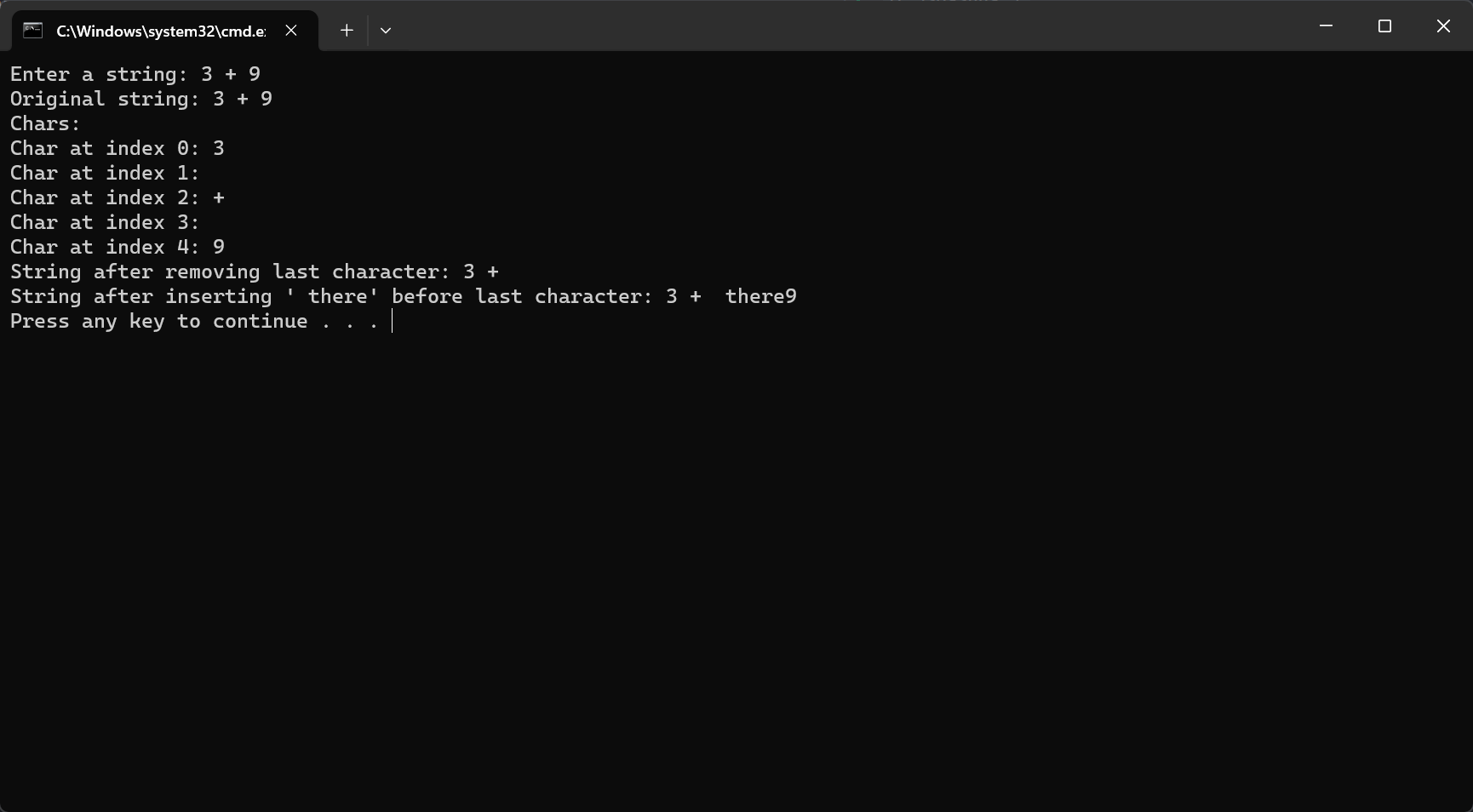
Результати виконання роботи

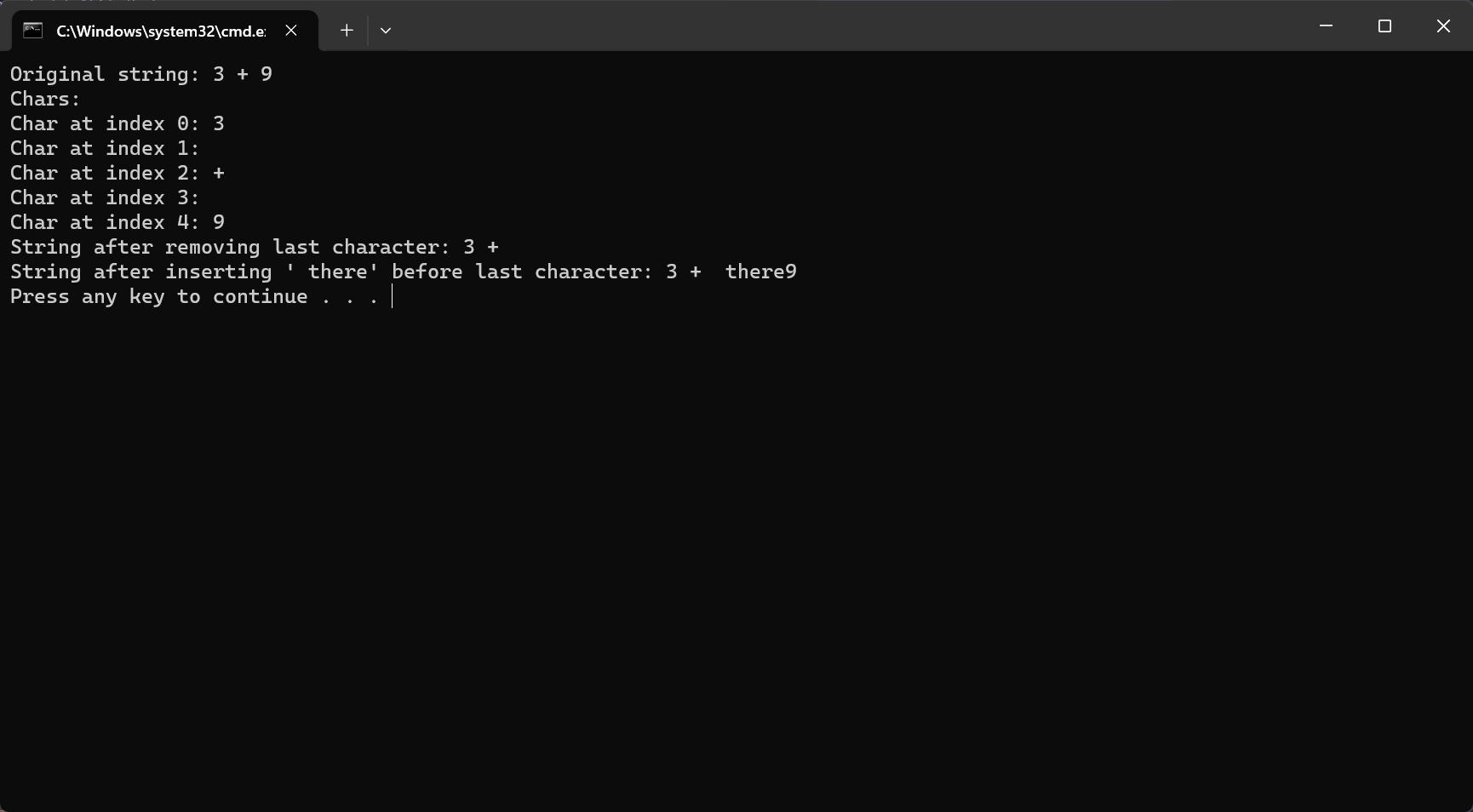












Код

*// Завдання 1*

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace dev

{

    internal class Program

    {

        static void Main(string[] *args*)

        {

            string loadedInput = LoadInput();

            if (loadedInput != null)

            {

                Console.Write("Last input: ");

                Console.WriteLine(loadedInput);

            }

            Console.Write("Enter a string: ");

            string input = Console.ReadLine();

            string compressed = Compress(input);

            Console.WriteLine($"Compressed: {compressed}");

            string decompressed = Decompress(compressed);

            Console.WriteLine($"Decompressed: {decompressed}");

            SaveInput(input);

        }

        private static string LoadInput()

        {

            string filename = "prevInput.txt";

            if (File.Exists(filename))

            {

                return File.ReadAllLines(filename)?.Last();

            }

            return null;

        }

        private static void SaveInput(string *input*)

        {

            File.AppendAllText("prevInput.txt", *input* + Environment.NewLine);

        }

        private static string Compress(string *input*)

        {

            StringBuilder sb = new StringBuilder();

            int count = 1;

            for (int i = 1; i < *input*.Length; i++)

            {

                if (*input*[i] == *input*[i - 1] && !Char.IsWhiteSpace(*input*[i]))

                {

                    count++;

                }

                else

                {

                    if (count > 1)

                    {

                        sb.Append(*input*[i - 1].ToString() + count);

                    }

                    else

                    {

                        sb.Append(*input*[i - 1]);

                    }

                    count = 1;

                }

            }

            if (count > 1)

            {

                sb.Append(*input*[*input*.Length - 1].ToString() + count);

            }

            else

            {

                sb.Append(*input*[*input*.Length - 1]);

            }

            return sb.ToString();

        }

        private static string Decompress(string *input*)

        {

            StringBuilder sb = new StringBuilder();

            int count = 0;

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

            {

                if (Char.IsDigit(*input*[i]))

                {

                    count = int.Parse(*input*[i].ToString());

                    while (count > 1)

                    {

                        sb.Append(*input*[i - 1]);

                        count--;

                    }

                }

                else

                {

                    sb.Append(*input*[i]);

                }

            }

            return sb.ToString();

        }

    }

}

*// Завдання 2*

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace dev

{

    internal class Program

    {

        private const string SessionFile = "session.txt";

        static void Main(string[] *args*)

        {

            string input;

            if (File.Exists(SessionFile))

            {

                input = File.ReadAllLines(SessionFile).Last();

            }

            else

            {

                Console.Write("Enter an expression: ");

                input = Console.ReadLine();

            }

            double result = Calculate(input);

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

            File.AppendAllLines(SessionFile, new[] { input });

        }

        private static double Calculate(string *input*)

        {

            string[] parts = *input*.Split(' ');

            double num1 = double.Parse(parts[0]);

            string operation = parts[1];

            double num2 = double.Parse(parts[2]);

            switch (operation)

            {

                case "+":

                case "plus":

                    return num1 + num2;

                case "-":

                case "minus":

                    return num1 - num2;

                case "\*":

                case "multiply":

                case "times":

                    return num1 \* num2;

                case "/":

                case "divide":

                case "by":

                    return num1 / num2;

                default:

                    throw new Exception("Invalid operation");

            }

        }

    }

}

*// Завдання 3*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace dev

{

    internal class Program

    {

        static void Main(string[] *args*)

        {

            string stateFilePath = "state.txt";

            string input = "";

            if (File.Exists(stateFilePath))

            {

                string[] previousState = File.ReadAllLines(stateFilePath);

                if (previousState.Length > 0)

                    input = previousState[previousState.Length - 1];

            }

            if (string.IsNullOrEmpty(input))

            {

                Console.Write("Enter a string: ");

                input = Console.ReadLine();

            }

            Console.WriteLine("Original string: " + input);

*// Chars*

            Console.WriteLine("Chars:");

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

            {

                Console.WriteLine("Char at index " + i + ": " + input[i]);

            }

*// Remove*

            string removed = input.Remove(input.Length - 1, 1);

            Console.WriteLine("String after removing last character: " + removed);

*// Insert*

            string inserted = input.Insert(input.Length - 1, " there");

            Console.WriteLine("String after inserting ' there' before last character: " + inserted);

            File.AppendAllText(stateFilePath, input + Environment.NewLine);

        }

    }

}

Висновки

Таким чином, ми вивчили основні принципи роботи з файлами у середовищі Visual Studio C#.

Контрольні питання

**Що таке файловий потік?**

Файловий потік - це послідовність байтів, яку можна використовувати для читання з файлу та запису до нього. Це клас, який використовується для читання та запису файлів у C#. Він є частиною простору імен System.IO.

**Які класи є основними для роботи з файлами та потоками у Visual Studio С#?**

Нижче наведено кілька найпоширеніших класів потоків у Visual Studio C#:

FileStream - для читання та запису у файл.

IsolatedStorageFileStream - для читання та запису до файлу в ізольованому сховищі.

MemoryStream - для читання і запису в пам'ять як резервне сховище.

BufferedStream - для підвищення продуктивності операцій читання та запису.

NetworkStream - для читання та запису через мережеві сокети.

PipeStream - для читання та запису через анонімні та іменовані канали.

**Поясніть призначення та використання об’єкту типуFilestream.**

Щоб маніпулювати файлами за допомогою FileStream, потрібно створити об'єкт класу FileStream. Цей об'єкт має чотири параметри: ім'я файлу, режим файлу, доступ до файлу та спільний доступ до файлу. Клас FileStream підтримує як синхронні, так і асинхронні операції запису та читання файлів. Ви можете використовувати його для читання, запису, відкриття і закриття файлів у файловій системі, а також для маніпулювання іншими дескрипторами операційної системи, пов'язаними з файлами, зокрема каналами, стандартним введенням і стандартним виведенням. Ви можете використовувати методи Read, Write, CopyTo і Flush для виконання синхронних операцій або методи ReadAsync, WriteAsync, CopyToAsync і FlushAsync для виконання асинхронних операцій. Використовуйте асинхронні методи для виконання ресурсоємних файлових операцій без блокування основного потоку.