Лабораторна робота №1

Процеси та потоки

Хід роботи:

Завдання 1:

Необхідно написати дві програми (**три**), які будуть мати спільні дані та одночасно до них звертатися.

Існує кілька механізмів реалізації спільного доступу до даних різних процесів.

Скористаємося одним з них, найбільш зручним - проектуванням файлу в пам'ять.

Одна програма буде сортувати дані у файлі, а інша відображати вміст цього файлу. Працювати обидва процеси будуть одночасно. **Третя програма** буде створювати (або заповнювати по новому) масив випадкових чисел.

Створіть файл data.dat. У ньому мають бути записані числа, згенеровані випадковим чином. Кількість чисел - 20-30 штук. Діапазон значень: від 10 до 100. (Це саме числа, а не символьні рядки зберігають ASCII коди цифр !!!)

Для початку створимо програму яка буде створювати файл data.dat та заповнювати його цифрами, всю роботу я виконував на мові C#:

```
using System;
using System.IO;

class Program
{
    static void Main()
    {
        Random rand = new Random();
        int[] numbers = new int[rand.Next(20, 31)]; // Генеруємо випадкову кількість
чисел від 20 до 30

        for (int i = 0; i < numbers.Length; i++)
        {
            numbers[i] = rand.Next(10, 101); // Генеруємо випадкове число від 10 до
100
        }

        // Записуємо числа у файл data.dat
        using (BinaryWriter writer = new BinaryWriter(File.Open("data.dat",
FileMode.Create)))
        {
            foreach (int number in numbers)
            {
                writer.Write(number);
            }
        }

        Console.WriteLine("Числа були успішно згенеровані та записані у файл
data.dat.");
```

```
    }
}
```

Програма №1. "Сортування даних" (консольна)

Беремо за основу програму "Hello windows"
Включаємо обробку події натискання клавіші, і відстежуємо в ньому натискання пробілу. Якщо користувач натиснув пробіл, значить починаємо сортування даних.
Виконуємо проектування файлу в пам'ять. Використовуємо для цього створений файл data.dat. В результаті отримаємо доступ до даних як до звичайного одновимірного масиву.
Виконуємо сортування масиву, будь-яким з методів сортування. Вставте 1-но секундну затримку для кожної ітерації сортування масиву, це дозволить потім наочніше побачити процес сортування.
По закінченню сортування, програма виводить у вікно, рядок «Робота завершена».

Також для легкості подальшої роботи я використав одразу mutex

Та MemoryMappedFile для запису даних пам'ять комп’ютера для подальшого використання в другій програмі.

```
using System;
using System.IO;
using System.IO.MemoryMappedFiles;
using System.Linq;
using System.Text;
using System.Threading;

class Program
{
    static Mutex mutex = new Mutex();

    static void Main()
    {
        Console.WriteLine("Натисніть пробіл, щоб почати сортування даних...");

        while (Console.ReadKey().Key != ConsoleKey.Spacebar) { }

        int[] numbers;

        // Проектуємо файл в пам'ять
        try
        {
            using (BinaryReader reader = new
BinaryReader(File.Open("C:\\Users\\Fleks\\source\\repos\\lab_1_1\\lab_1_1\\data.dat",
FileMode.Open)))
            {
                numbers = new int[reader.BaseStream.Length / sizeof(int)];

                for (int i = 0; i < numbers.Length; i++)
                {
                    numbers[i] = reader.ReadInt32();
                }
            }
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"Помилка при читанні файлу: {ex.Message}");
            return;
        }

        // Сортуємо масив
        try
        {
            mutex.WaitOne();

            using (MemoryMappedFile mmf =
MemoryMappedFile.CreateNew("sort_iterations", 10000))
            {
                for (int i = 0; i < numbers.Length - 1; i++)
                {
                    for (int j = 0; j < numbers.Length - i - 1; j++)
                    {
                        if (numbers[j] > numbers[j + 1])
                        {
                            // Обмін елементів
                            int temp = numbers[j];
                            numbers[j] = numbers[j + 1];
                            numbers[j + 1] = temp;
                        }
                    }

                    // Записуємо поточний стан масиву після кожної ітерації
                    using (MemoryMappedViewStream stream = mmf.CreateViewStream())
                    {
                        BinaryWriter writer = new BinaryWriter(stream);
                        writer.Write($"Iteration {i + 1}: {string.Join(", ",
numbers)}\n");
                    }

                    // Затримка на 1 секунду
                    Thread.Sleep(1000);
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка при сортуванні: {ex.Message}");
        }
        finally
        {
            mutex.ReleaseMutex();
        }

        Console.WriteLine("Робота завершена");
    }
}
```

Програма №2. «Виведення файлу даних у вікно» (віконна)

Виконуємо проектування файлу в пам'ять. Використовуємо для цього створений файл data.dat. В результаті отримаємо доступ до даних як до звичайного одновимірного масиву. Цей же файл проектує в пам'ять попередня програма.

Створюємо таймер на 0.5 секунди. При отриманні повідомлення від таймера, виконуємо висновок всього масиву в вікно. Передбачте коректний перевивід даних у вікно, без накладень. У вікно виводиться не числа з масиву, а рядки

одного і того ж символу, наприклад «*», в кількості, що дорівнює числу з масиву.

Запускаємо на виконання обидві програми одночасно. Коли друга програма запустилася і виконує висновок даних у вікно (виводить поки одну й ту ж саму картинку кожні пів секунди), натискаємо пробіл в першій програмі і вона починає сортувати масив. При цьому, так як вони дані беруть з одного і того ж файлу (обидві проектували його собі на згадку), то перша вносить зміни переставляючи дані при сортуванні, а друга виводить з себе у вікно і ми бачимо хід процесу сортування. Тимчасову затримку в першій програмі можна при потребі збільшити.

Ці дві програми демонструють можливість організації спільного доступу процесів до одних і тих самих даних. Так само демонструється механізм проектування файлу в пам'ять, як один з найкращих методів доступу до файлу.

Для цього завдання я обрав Win form для легкості роботи, та одразу брав данні з пам’яті.

```
using System;
using System.IO.MemoryMappedFiles;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using Timer = System.Windows.Forms.Timer;

namespace lab1_2
{
    public partial class Form1 : Form
    {
        static Mutex mutex = new Mutex();
        static Timer timer;
        static string previousIterationResult = "";

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            timer = new Timer();
            timer.Interval = 500; // 0.5 секунди
            timer.Tick += DisplayData;
            timer.Start();
        }

        private void DisplayData(object sender, EventArgs e)
        {
            // Виводимо дані в richTextBox1
            try
            {
```

```
                mutex.WaitOne();

                using (MemoryMappedFile mmf =
MemoryMappedFile.OpenExisting("sort_iterations"))
                {
                    using (MemoryMappedViewStream stream = mmf.CreateViewStream())
                    {
                        BinaryReader reader = new BinaryReader(stream);
                        string iterationResult = reader.ReadString();

                        // Додаємо текст до richTextBox1, якщо він відрізняється від
попереднього
                        if (iterationResult != previousIterationResult)
                        {
                            richTextBox1.AppendText(iterationResult +
Environment.NewLine);
                            previousIterationResult = iterationResult;
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                //richTextBox1.AppendText($"Помилка при виведенні даних: {ex.Message}"
+ Environment.NewLine);
            }
            finally
            {
                mutex.ReleaseMutex();
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {

        }

        private void Form1_KeyDown(object sender, KeyEventArgs e)
        {
        }
    }
}
```

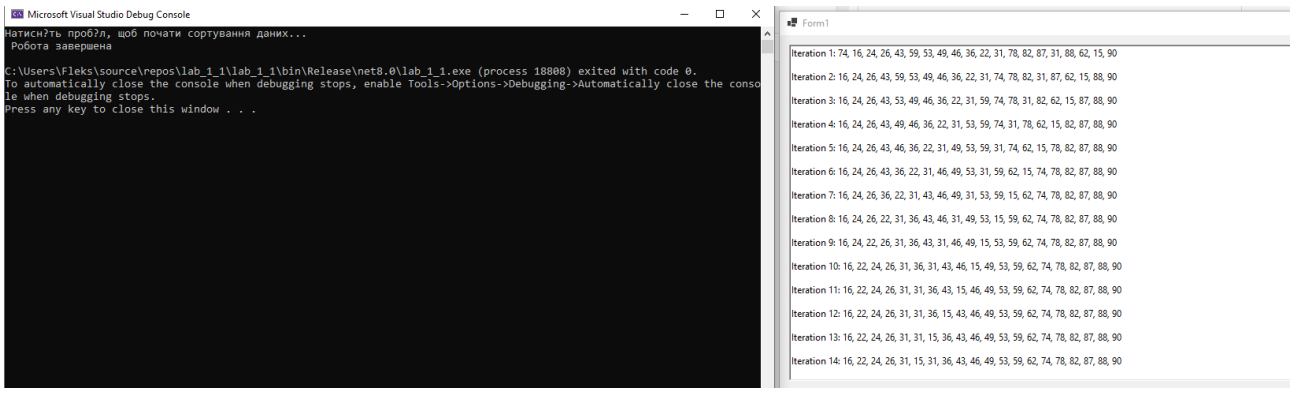

Рис 1 – демонстрація роботи програми

Додаткове завдання.

Написати четверту програму (консольну), яка буде одночасно працювати, та намагатися відсортувати той самий масив в іншому напрямку та іншим відомим методом сортування.

```
using System;
using System.IO;
using System.IO.MemoryMappedFiles;
using System.Linq;
using System.Text;
using System.Threading;

class Program2
{
    static Mutex mutex = new Mutex();

    static void Main()
    {
        Console.WriteLine("Натисніть пробіл, щоб почати сортування даних...");

        while (Console.ReadKey().Key != ConsoleKey.Spacebar) { }

        int[] numbers;

        // Проектуємо файл в пам'ять
        try
        {
            using (BinaryReader reader = new
BinaryReader(File.Open("C:\\Users\\Fleks\\source\\repos\\lab_1_1\\lab_1_1\\data.dat",
FileMode.Open)))
            {
                numbers = new int[reader.BaseStream.Length / sizeof(int)];

                for (int i = 0; i < numbers.Length; i++)
                {
                    numbers[i] = reader.ReadInt32();
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка при читанні файлу: {ex.Message}");
            return;
        }

        // Сортуємо масив
        try
        {
            mutex.WaitOne();

            using (MemoryMappedFile mmf =
MemoryMappedFile.CreateNew("sort_iterations", 10000))
            {
                for (int i = 1; i < numbers.Length; i++)
                {
                    int key = numbers[i];
                    int j = i - 1;

                    // Переміщуємо елементи масиву, які більше за ключ, на одну
позицію вперед
                    while (j >= 0 && numbers[j] < key)
                    {
                        numbers[j + 1] = numbers[j];
                        j = j - 1;
```

```
                    }
                    numbers[j + 1] = key;

                    // Записуємо поточний стан масиву після кожної ітерації
                    using (MemoryMappedViewStream stream = mmf.CreateViewStream())
                    {
                        BinaryWriter writer = new BinaryWriter(stream);
                        writer.Write($"Iteration {i + 1}: {string.Join(", ",
numbers)}\n");
                    }

                    // Затримка на 1 секунду
                    Thread.Sleep(1000);
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка при сортуванні: {ex.Message}");
        }
        finally
        {
            mutex.ReleaseMutex();
        }

        Console.WriteLine("Робота завершена");
    }
}
```