МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра автоматизації та інформаційних систем

Навчальна дисципліна

«ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 7

Виконав
студент групи КН-23-1
Іщенко.Є.В
Перевірила
доцент кафедри АІС
Істоміна Н. М.

Кременчук 2025

Лабораторна робота № 7

Тема: Дослідження впливу пріоритету потоків

Мета: набути навичок задання потокам різних пріоритетів та доцільного використання цього інструменту.

Хід роботи:

Під час лабораторної роботи необхідно виконати такі дії:

1. Створіть консольний застосунок, який складається з трьох потоків. Кожен потік у циклі має виводити на екран числа від 0 до 9. Наведіть власний код і «прінтскрін» роботи програми.
2. Змініть код так, щоб потік 1 виводив числа від 0 до 9, потік 2 – від 10 до 19, потік 3 від 20 до 29. Наведіть власний код і «прінтскрін» роботи програми.
3. Змініть код так, щоб на екран усі числа від 0 до 29 виводилися послідовно. Наведіть власний код і «прінтскрін» роботи програми.
4. Створіть консольний застосунок, який складається з двох потоків. В основному потоці викликається другий потік, і робота основного потоку призупиняється на 150 мс, 200 мс. Водночас другий потік у циклі виводить числа від 0 до 10000000. Зробіть другий потік фоновим. Наведіть власний код. Для порівняння наведіть «прінтскріни» роботи програми для двох випадків: другий потік – основний, другий потік – фоновий.

Завдання 1:

```
using System;
using System.Threading;

class Program
{
    // ThreadLocal для забезпечення унікального Random для кожного потоку
    private static ThreadLocal<Random> random = new ThreadLocal<Random>(() => new Random());

    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.Default;

        // Створення трьох потоків
        Thread thread1 = new Thread(PrintNumbers);
        Thread thread2 = new Thread(PrintNumbers);
        Thread thread3 = new Thread(PrintNumbers);

        // Запуск потоків
        thread1.Start("Потік 1");
        thread2.Start("Потік 2");
        thread3.Start("Потік 3");

        // Очікування завершення
        thread1.Join();
        thread2.Join();
        thread3.Join();

        Console.WriteLine("Всі потоки завершили роботу.");
    }

    static void PrintNumbers(object threadName)
    {
        for (int i = 0; i < 10; i++)
        {
            // Генерація випадкового числа від 0 до 9
            int randomNumber = random.Value.Next(0, 10);
            Console.WriteLine($"{threadName}: {randomNumber}");
            Thread.Sleep(50);
        }
    }
}
```

Рисунок 1.1 – Код

На рисунку 1.2 – наведена робота консольного застосунку.

```
Потік 3: 3
Потік 1: 1
Потік 2: 2
Потік 3: 1
Потік 2: 0
Потік 1: 7
Потік 3: 8
Потік 2: 1
Потік 1: 9
Потік 3: 0
Потік 2: 1
Потік 1: 8
Потік 3: 3
Потік 2: 8
Потік 1: 8
Потік 3: 4
Потік 2: 5
Потік 1: 0
Потік 3: 6
Потік 1: 5
Потік 2: 7
Потік 3: 3
Потік 1: 4
Потік 2: 2
Потік 3: 3
Потік 1: 7
Потік 2: 9
Потік 3: 3
Потік 1: 1
Потік 2: 5
Всі потоки завершили роботу.
```

Рисунок 1.2 – Робота потоків

Завдання 2:

```
using System;
using System.Threading;

class Program
{
    // ThreadLocal для забезпечення унікального Random для кожного потоку
    private static ThreadLocal<Random> random = new ThreadLocal<Random>(() => new Random());

    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.Default;

        // Створення трьох потоків
        Thread thread1 = new Thread(PrintRange);
        Thread thread2 = new Thread(PrintRange);
        Thread thread3 = new Thread(PrintRange);

        // Запуск потоків із явно створеними кортежами (object)
        thread1.Start(Tuple.Create(0, 9, "Потік 1"));
        thread2.Start(Tuple.Create(10, 19, "Потік 2"));
        thread3.Start(Tuple.Create(20, 29, "Потік 3"));

        // Очікування завершення
        thread1.Join();
        thread2.Join();
        thread3.Join();

        Console.WriteLine("Всі потоки завершили роботу.");
    }

    static void PrintRange(object param)
    {
        // Явне розпакування параметра
        var tuple = (Tuple<int, int, string>)param;
        int start = tuple.Item1;
        int end = tuple.Item2;
        string threadName = tuple.Item3;

        for (int i = 0; i < 10; i++)
        {
            // Генерація випадкового числа в заданому діапазоні
            int randomNumber = random.Value.Next(start, end + 1);
            Console.WriteLine($"{threadName}: {randomNumber}");
            Thread.Sleep(50);
        }
    }
}
```

Рисунок 1.3 – Код 2

На рисунку 1.4 – наведена робота зміненого коду.

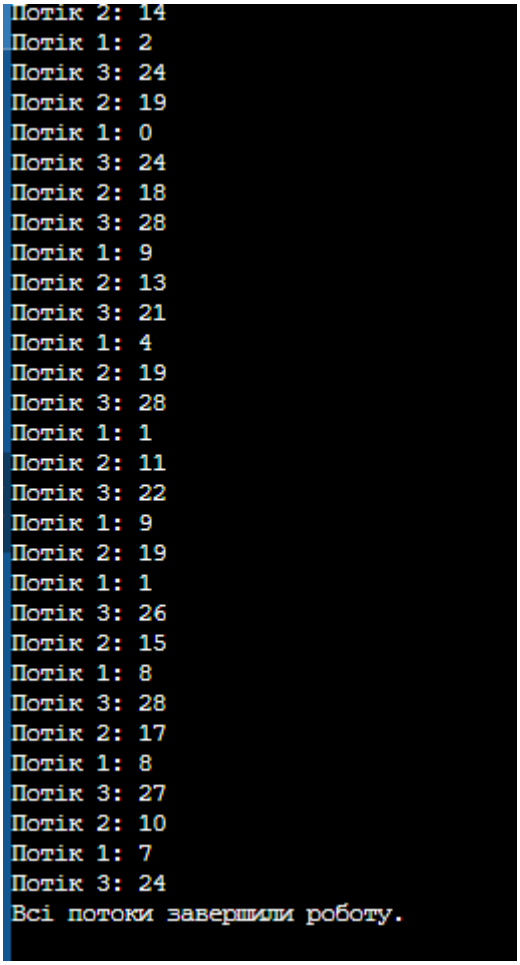

Рисунок 1.4 – Змінений вивід чисел для потоків

Завдання 3:

```
using System;
using System.Threading;

class Program
{
    // Лічильник, який визначає поточне число
    static int counter = 0;
    static object locker = new object();

    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.Default;

        // Створення трьох потоків
        Thread thread1 = new Thread(PrintSequential);
        Thread thread2 = new Thread(PrintSequential);
        Thread thread3 = new Thread(PrintSequential);

        // Запуск потоків
        thread1.Start("Потік 1");
        thread2.Start("Потік 2");
        thread3.Start("Потік 3");

        // Очікування завершення
        thread1.Join();
        thread2.Join();
        thread3.Join();

        Console.WriteLine("Всі потоки завершили роботу.");
    }

    static void PrintSequential(object threadName)
    {
        while (true)
        {
            lock (locker) // Блокування доступу до лічильника
            {
                if (counter > 29) break; // Завершення при досягненні 30

                // Вивід числа у суворій послідовності
                Console.WriteLine($"{threadName}: {counter}");
                counter++;
            }
            // Коротка затримка для імітації реальної роботи
            Thread.Sleep(50);
        }
    }
}
```

Рисунок 1.5 – код 3

На рисунку 1.6 – наведена робота зміненого коду.

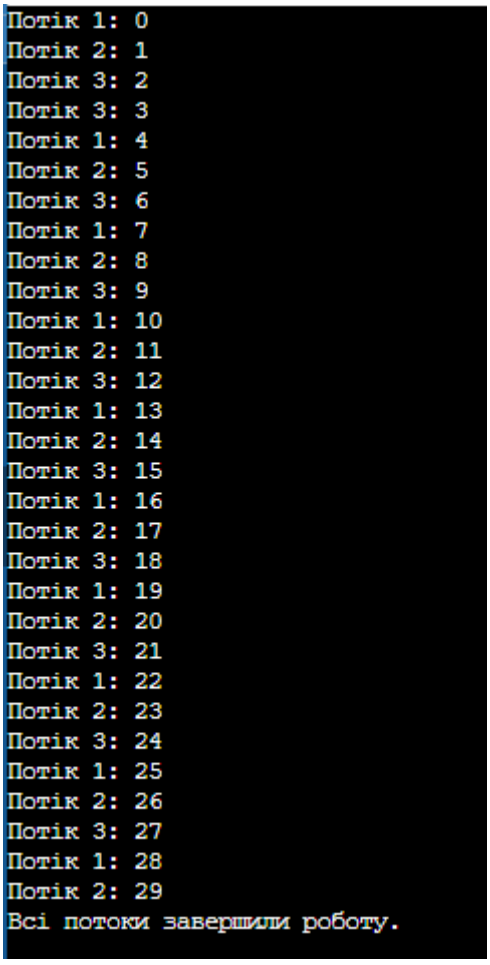

Рисунок 1.6 – Змінений вивід чисел для потоків на послідовний

Завдання 4:

```
using System;
using System.Threading;
using System.Diagnostics;

class Program
{
    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.Default;

        do
        {
            Console.WriteLine("\nВведіть режим роботи (1 - фоновий, 2 - основний):");
            string input = Console.ReadLine();
            bool isBackground = input == "1";

            Thread workerThread = new Thread(CountNumbers);
            workerThread.IsBackground = isBackground; // Встановлення режиму (фоновий або основний)
            Console.WriteLine($"\nЗапуск робітничого потоку... (Фоновий: {workerThread.IsBackground})");

            // Вимірювання часу виконання
            Stopwatch stopwatch = Stopwatch.StartNew();
            workerThread.Start();

            while (workerThread.IsAlive) // Працює, поки робітничий потік не завершився
            {
                Console.WriteLine("Основний потік працює...");
                Thread.Sleep(150); // 150 мс
                Console.WriteLine("Основний потік працює...");
                Thread.Sleep(200); // 200 мс
            }

            stopwatch.Stop();
            Console.WriteLine($"\nОсновний потік завершено.");
            Console.WriteLine($"Час виконання: {stopwatch.ElapsedMilliseconds} мс");

            Console.WriteLine("\nБажаєте повторити роботу? (Так - y, Ні - n):");
        }
        while (Console.ReadLine() == "y");

        Console.WriteLine("Програма завершена.");
    }

    static void CountNumbers()
    {
        Stopwatch workerStopwatch = Stopwatch.StartNew();

        for (int i = 0; i <= 10000000; i++)
        {
            if (i % 1000000 == 0) // Щоб не засмічувати консоль
                Console.WriteLine($"Робітничий потік: {i}");
        }

        workerStopwatch.Stop();
        Console.WriteLine($"Робітничий потік завершено. Час виконання: {workerStopwatch.ElapsedMilliseconds} мс");
    }
}
```

Рисунок 1.7 – код 4

На рисунку 1.8 – наведена робота застосунку з двома потоками.

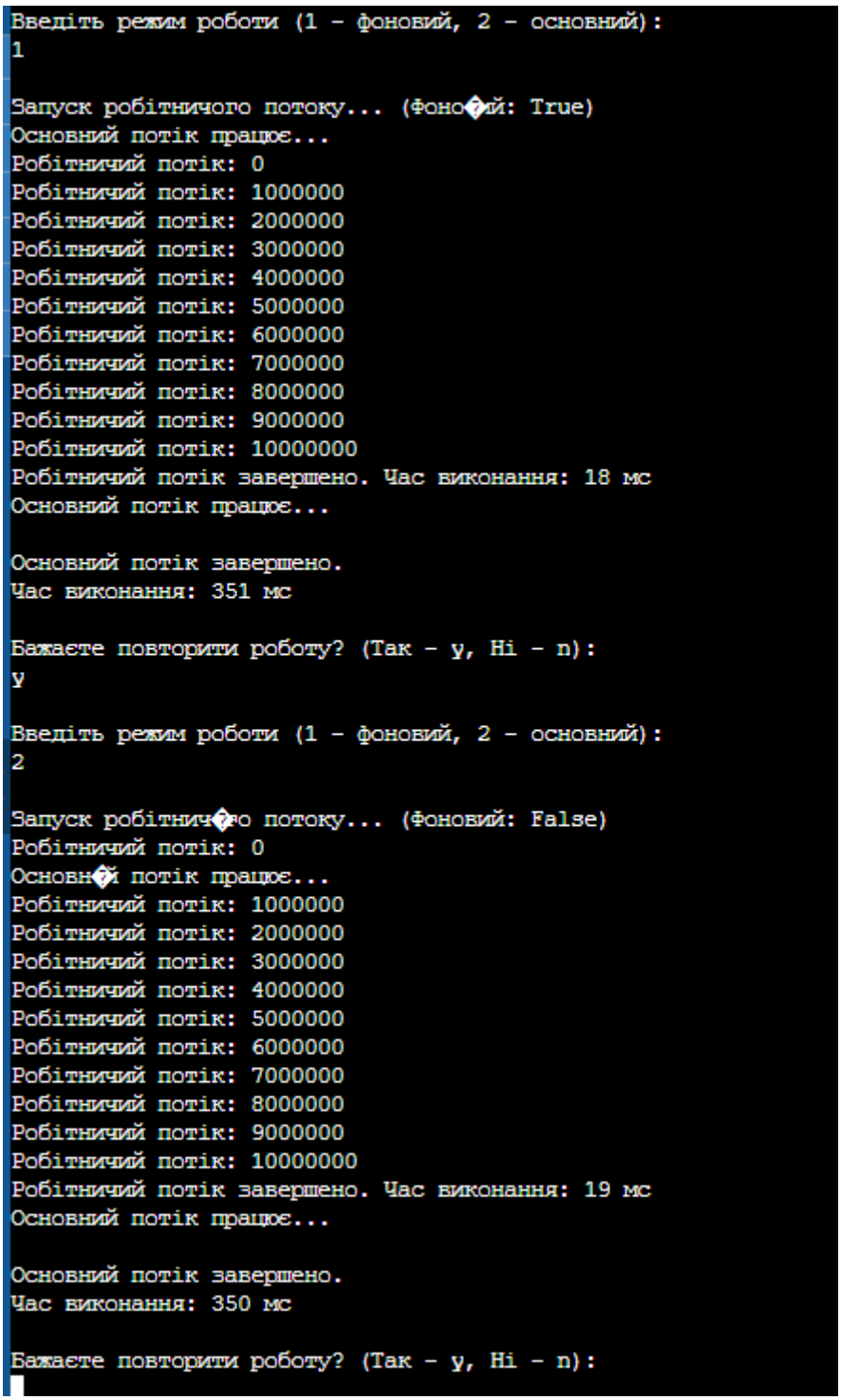

Рисунок 1.8 – Робота консольного застосунку з двома потоками

Висновки:

На цій лабораторній роботі ми досліджували вплив пріоритету потоків, набули навичок задання потокам різних пріоритетів та доцільного використання цього інструменту. Створили чотири консольних застосунки та провели перевірку роботи потоків.