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 № 8

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

Кременчук 2025

Лабораторна робота № 8

Тема: Синхронізація потоків

Мета: набути навичок синхронізації паралельних потоків та організації спільного доступу до даних у C#.

Хід роботи:

1. Створити програмний код згідно з прикладом. У звіті наведіть власний код і «прінтскрін» роботи програми.
2. Змініть код так: кількість потоків – 10; нова загальна змінна – y; перші 5 потоків працюють зі змінною x, другі 5 – із y. Наведіть власний код і «прінтскрін» роботи програми.
3. Використайте «lock» для синхронізації потоків. Наведіть власний код і «прінтскрін» роботи програми.
4. Використайте монітор для синхронізації потоків. Наведіть власний код і «прінтскрін» роботи програми.
5. Використайте «AutoResetEvent» для синхронізації потоків. Наведіть власний код і «прінтскрін» роботи програми.

Завдання 1:

Створимо програмний код згідно з прикладом.

```
using System;
using System.Threading;

class Program
{
    static int x = 0;

    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.Default;
        for (int i = 0; i < 5; i++)
        {
            Thread myThread = new Thread(Count);
            myThread.Name = "Потік " + i.ToString();
            myThread.Start();
        }

        Console.ReadLine();
    }

    static void Count()
    {
        for (int i = 0; i < 5; i++)
        {
            x++;
            Console.WriteLine($"{Thread.CurrentThread.Name} = {x}");
            Thread.Sleep(100);
        }
    }
}
```

Рисунок 1.1 – Робота коду

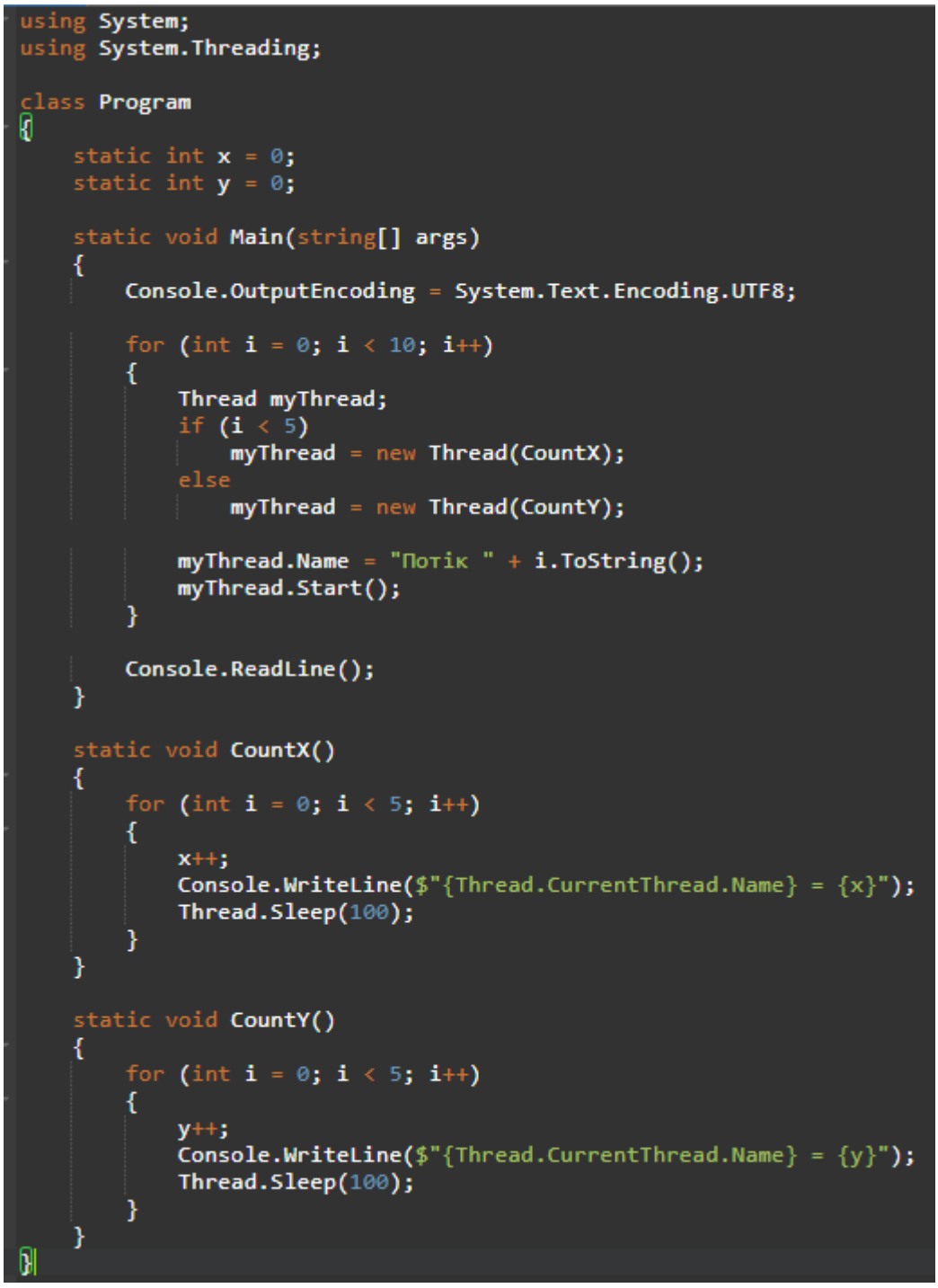
На рисунку 1.2 наведена робота програмного коду згідно з прикладом.

```
Потік 1 = 5
Потік 4 = 5
Потік 2 = 5
Потік 3 = 5
Потік 0 = 5
Потік 1 = 6
Потік 3 = 9
Потік 0 = 10
Потік 4 = 8
Потік 2 = 8
Потік 1 = 11
Потік 4 = 12
Потік 3 = 13
Потік 0 = 14
Потік 2 = 15
Потік 1 = 16
Потік 4 = 17
Потік 3 = 18
Потік 0 = 19
Потік 2 = 20
Потік 4 = 21
Потік 1 = 22
Потік 3 = 23
Потік 2 = 25
Потік 0 = 24
```

Рисунок 1.2 – Робота коду

Завдання 2:

Змініть код згідно із другим завданням.



```
using System;
using System.Threading;

class Program
{
    static int x = 0;
    static int y = 0;

    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;

        for (int i = 0; i < 10; i++)
        {
            Thread myThread;
            if (i < 5)
                myThread = new Thread(CountX);
            else
                myThread = new Thread(CountY);

            myThread.Name = "Потік " + i.ToString();
            myThread.Start();
        }

        Console.ReadLine();
    }

    static void CountX()
    {
        for (int i = 0; i < 5; i++)
        {
            x++;
            Console.WriteLine($"{Thread.CurrentThread.Name} = {x}");
            Thread.Sleep(100);
        }
    }

    static void CountY()
    {
        for (int i = 0; i < 5; i++)
        {
            y++;
            Console.WriteLine($"{Thread.CurrentThread.Name} = {y}");
            Thread.Sleep(100);
        }
    }
}
```

Рисунок 1.3 – Код 2

На рисунку 1.4 наведений змінений код першого завдання.

Рисунок 1.4 – Змінений код першого завдання

Завдання 3:

Використаємо «lock» для синхронізації потоків.

```
using System;
using System.Threading;

class Program
{
    static int x = 0;
    static int y = 0;
    static readonly object locker = new object();

    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        for (int i = 0; i < 10; i++)
        {
            Thread myThread;
            if (i < 5)
                myThread = new Thread(CountX);
            else
                myThread = new Thread(CountY);

            myThread.Name = "Потік " + i.ToString();
            myThread.Start();
        }

        Console.ReadLine();
    }

    static void CountX()
    {
        for (int i = 0; i < 5; i++)
        {
            lock (locker)
            {
                x++;
                Console.WriteLine($"{Thread.CurrentThread.Name} = {x}");
            }
            Thread.Sleep(100);
        }
    }

    static void CountY()
    {
        for (int i = 0; i < 5; i++)
        {
            lock (locker)
            {
                y++;
                Console.WriteLine($"{Thread.CurrentThread.Name} = {y}");
            }
            Thread.Sleep(100);
        }
    }
}
```

Рисунок 1.5 – Код 3

На рисунку 1.6 наведене використання «lock» для синхронізації потоків.

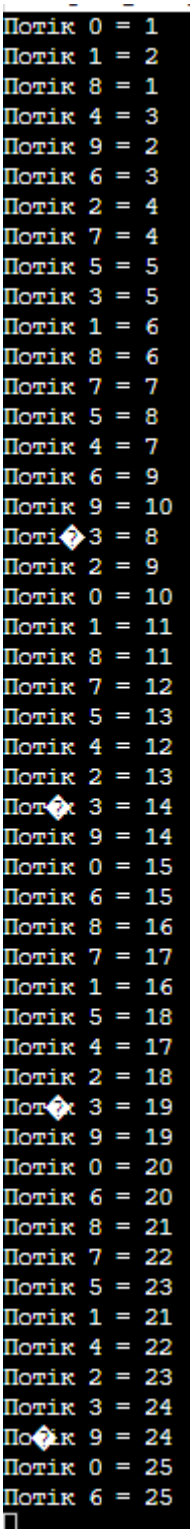


Рисунок 1.6 – Використання «lock» для синхронізації потоків

Завдання 4:

Використаємо монітор для синхронізації потоків

```
using System.Threading;

class Program
{
    static int x = 0;
    static int y = 0;
    static readonly object locker = new object();

    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        for (int i = 0; i < 10; i++)
        {
            Thread myThread;
            if (i < 5)
                myThread = new Thread(CountX);
            else
                myThread = new Thread(CountY);

            myThread.Name = "Потік " + i.ToString();
            myThread.Start();
        }

        Console.ReadLine();
    }

    static void CountX()
    {
        for (int i = 0; i < 5; i++)
        {
            Monitor.Enter(locker);
            try
            {
                x++;
                Console.WriteLine($"{Thread.CurrentThread.Name} = {x}");
            }
            finally
            {
                Monitor.Exit(locker);
            }
            Thread.Sleep(100);
        }
    }

    static void CountY()
    {
        for (int i = 0; i < 5; i++)
        {
            Monitor.Enter(locker);
            try
            {
                y++;
                Console.WriteLine($"{Thread.CurrentThread.Name} = {y}");
            }
            finally
            {
                Monitor.Exit(locker);
            }
            Thread.Sleep(100);
        }
    }
}
```

Рисунок 1.7 – Код 4

На рисунку 1.8 наведене використання монітору для синхронізації потоків.

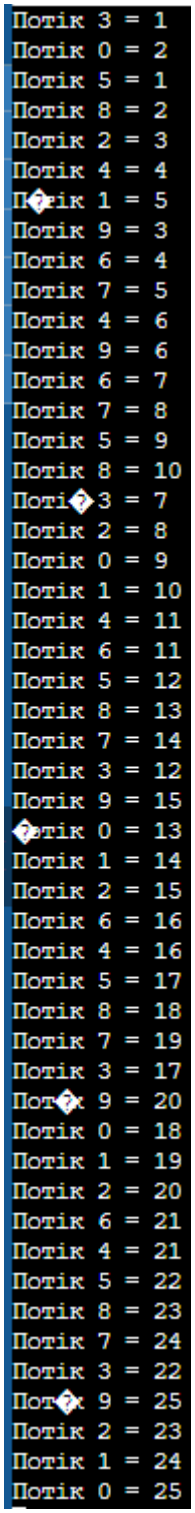


Рисунок 1.8 – Використання монітору для синхронізації потоків

Завдання 5:

Використаємо «AutoResetEvent» для синхронізації потоків

```
using System;
using System.Threading;

class Program
{
    static int x = 0;
    static int y = 0;
    static AutoResetEvent autoEventX = new AutoResetEvent(true);
    static AutoResetEvent autoEventY = new AutoResetEvent(false);

    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        for (int i = 0; i < 10; i++)
        {
            Thread myThread;
            if (i < 5)
                myThread = new Thread(CountX);
            else
                myThread = new Thread(CountY);

            myThread.Name = "Потік " + i.ToString();
            myThread.Start();
        }

        Console.ReadLine();
    }

    static void CountX()
    {
        for (int i = 0; i < 5; i++)
        {
            autoEventX.WaitOne(); // Очікування дозволу для X
            x++;
            Console.WriteLine($"{Thread.CurrentThread.Name} = {x}");
            autoEventY.Set(); // Дозволяємо потокам Y працювати
            Thread.Sleep(100);
        }
    }

    static void CountY()
    {
        for (int i = 0; i < 5; i++)
        {
            autoEventY.WaitOne(); // Очікування дозволу для Y
            y++;
            Console.WriteLine($"{Thread.CurrentThread.Name} = {y}");
            autoEventX.Set(); // Дозволяємо потокам X працювати
            Thread.Sleep(100);
        }
    }
}
```

Рисунок 1.9 – Код 5

На рисунку 1.10 наведене використання «AutoResetEvent» для синхронізації потоків.

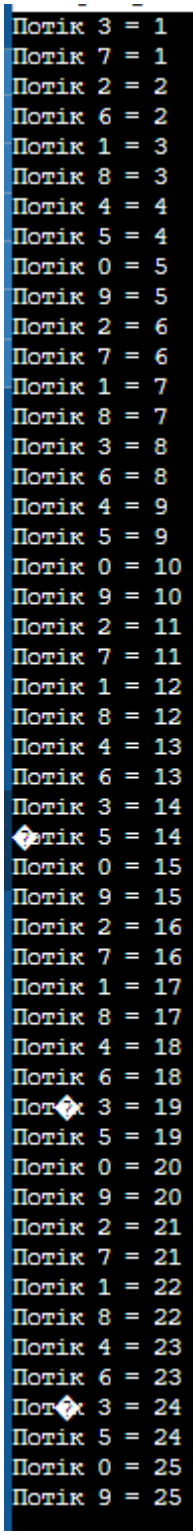


Рисунок 1.10 – Використання «AutoResetEvent» для синхронізації потоків

Висновки:

На цій лабораторній роботі ми синхронізовували потоки, набули навичок синхронізації паралельних потоків та організації спільного доступу

до даних у C#. Створили п’ять консольних застосунки та побачили синхронізацію потоків.