Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни
«Програмне забезпечення високопродуктивних комп’ютерних систем»

на тему

«Семафори, мютекси, події, критичні секції, бар’єри, атомік змінні (типи)»

Виконав
Студент групи ІМ-13
Котенко Ярослав Олегович

Перевірив
доц. Корочкін О.В.

Київ 2024

Завдання

Розробити паралельну програму для обчислення в паралельної системі (ПКС СП) функції :

Варіант: 8

X= p*max(C*(MA*MD))*R+ e*B

Введення – виведення даних			
1	2	3	4
X,e	C, MA	R,MD	B,p

Мова програмування: C#

Засоби організації взаємодії процесів : семафори, мютекси, події, критичні секції, атомік змінні (типи), бар'єри

Виконання лабораторної роботи

Етап 1. Побудова паралельного математичного алгоритму.

X= p*max(C*(MA*MD))*R+ e*B

1) $a_i = max(C * (MA * MD_H))$ CP: C, MA
2) $a = max(a, a_i)$ CP: a
3) $X_H = p * a * R_H + e * B_H$ CP: p, a, e – копії

N - розмірність вектора/матриці.

P - кількість потоків, які виконують обчислення.

H = N / P

Етап 2. Розробка алгоритмів потоків.

Задача Т1

Точки синхронізації

1. Введення X,e
2. **Сигнал** задачам Т2,Т3,Т4 про введення **– S2-1. S3-1, S4-1**
3. **Чекати** на введення даних в задачах Т2,Т3,Т4 **-- W2-1, W3-1,W4-1**
4. Обчислення 1: $a_1 = max(C * (MA * MD_H))$
5. Обчислення 2: $a = max(a, a_1)$ – КД1
6. **Сигнал** задачам Т2,Т3,Т4 про завершення обчислення 2 **– S2-2. S3-2, S4-2**
7. **Чекати** на завершення обчислення 2 в задачах Т2,Т3,Т4 **-- W2-2, W3-2,W4-2**
8. **Копія** p1 = p – КД2
9. **Копія** a1 = a – КД3
10. **Копія** e1 = e – КД4
11. Обчислення 3: $X_H = p * a * R_H + e * B_H$
12. **Сигнал** задачі Т4 про завершення обчислення 3 **– S4-3**

Задача Т2

1. Введення C, MA
2. **Сигнал** задачам Т1,Т3,Т4 про введення **– S1-1. S3-1, S4-1**
3. **Чекати** на введення даних в задачах Т1,Т3,Т4 **-- W1-1, W3-1,W4-1**
4. Обчислення 1: $a_2 = max(C * (MA * MD_H))$
5. Обчислення 2: $a = max(a, a_2)$ – КД1
6. **Сигнал** задачам Т1,Т3,Т4 про завершення обчислення 2 **– S1-2. S3-2, S4-2**
7. **Чекати** на завершення обчислення 2 в задачах Т1,Т3,Т4 **-- W1-2, W3-2,W4-2**
8. **Копія** p2 = p – КД2
9. **Копія** a2 = a – КД3
10. **Копія** e2 = e – КД4
11. Обчислення 3: $X_H = p * a * R_H + e * B_H$
12. **Сигнал** задачі Т4 про завершення обчислення 3 **– S4-3**

Задача Т3

1. Введення R, MD

2. **Сигнал** задачам T1,T2,T4 про введення **– S1-1. S2-1, S4-1**

3. **Чекати** на введення даних в задачах T1,T2,T4 **-- W1-1, W2-1,W4-1**

4. Обчислення 1: $a_3 = max(C * (MA * MD_H))$

5. Обчислення 2: $a = max(a, a_3)$ – КД1

6. **Сигнал** задачам T1,T2,T4 про завершення обчислення 2 **– S1-2. S2-2, S4-2**

7. **Чекати** на завершення обчислення 2 в задачах T1,T2,T4 **-- W1-2, W2-2,W4-2**

8. **Копія** p3 = p – КД2

9. **Копія** a3 = a – КД3

10. **Копія** e3 = e – КД4

11. Обчислення 3: $X_H = p * a * R_H + e * B_H$

12. **Сигнал** задачі T4 про завершення обчислення 3 **– S4-3**

Задача T4

1. Введення B, p

2. **Сигнал** задачам T1,T2,T3 про введення **– S1-1. S2-1, S3-1**

3. **Чекати** на введення даних в задачах T1,T2,T3 **-- W1-1, W2-1,W3-1**

4. Обчислення 1: $a_4 = max(C * (MA * MD_H))$

5. Обчислення 2: $a = max(a, a_4)$ – КД1

6. **Сигнал** задачам T1,T2,T3 про завершення обчислення 2 **– S1-2. S2-2, S3-2**

7. **Чекати** на завершення обчислення 2 в задачах T1,T2,T3 **-- W1-2, W2-2,W3-2**

8. **Копія** p4 = p – КД2

9. **Копія** a4 = a – КД3

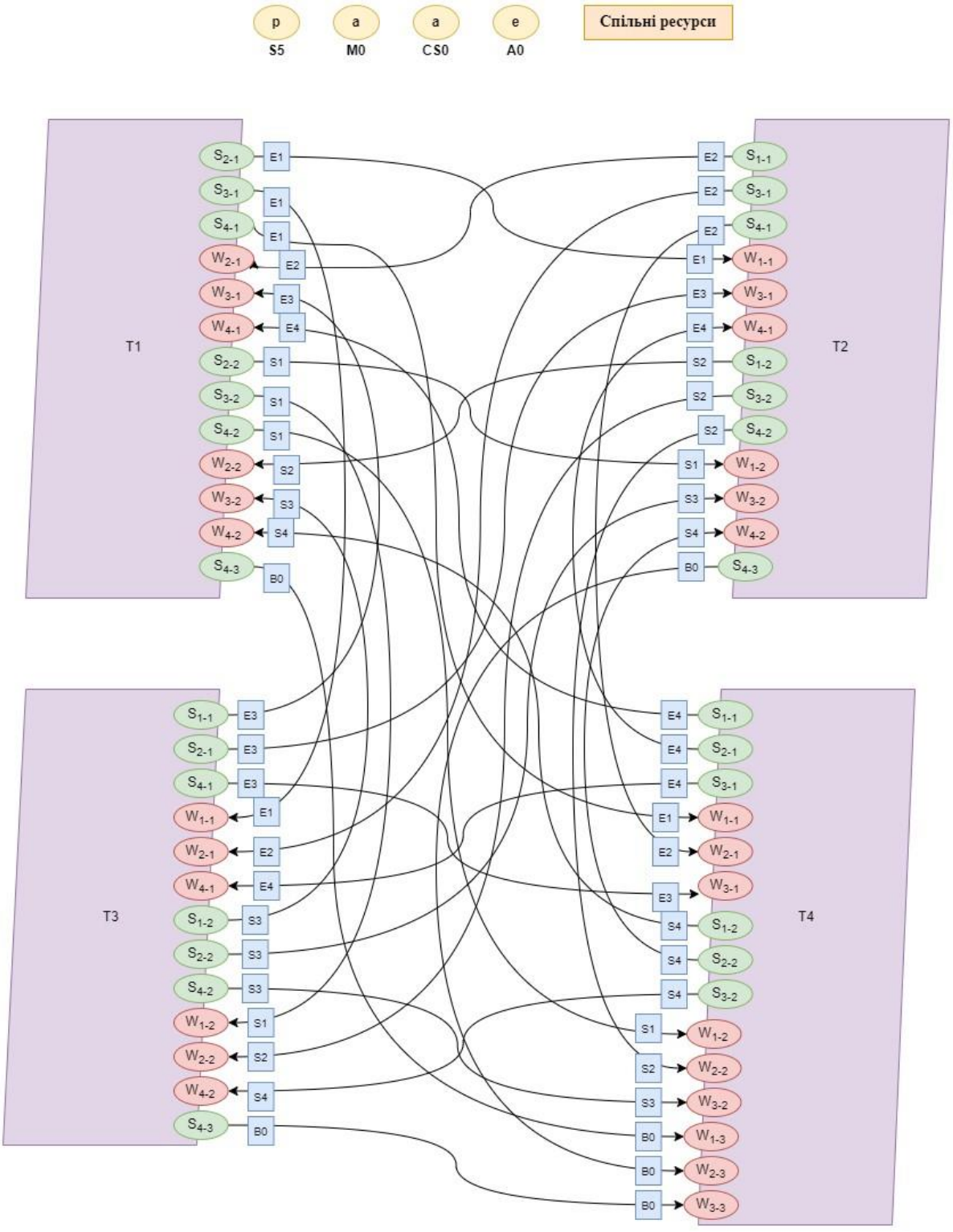
10. **Копія** e4 = e – КД4

11. Обчислення 3: $X_H = p * a * R_H + e * B_H$

12. **Чекати** завершення обчислення 3 в задачах T1,T2,T3 **-- W1-3, W2-3,W3-3**

13. Виведення результату X

Етап 3. Розробка схеми взаємодії задач



Кожна задача має чотири критичні ділянки (КД), і кожна з них обробляється за допомогою різних засобів: КД1 використовує м'ютекс, КД2 - семафор, КД3 - критичну секцію, КД4 – атомарну операцію. Також для синхронізації були використані події.

Події:

E1, E2, E3, E4 – призначені для синхронізації введення даних.

Семафори:

S1, S2, S3, S4 – призначені для синхронізації по обчисленню **a**.

S5 – призначений для захисту спільного ресурсу **p** (КД2).

Бар’єр:

B0 – призначений для синхронізації по завершенню обчислення **X**.

M'ютекс:

M0 – призначений для захисту спільного ресурсу **a** (КД1).

Критична секція:

CS0 – призначена для захисту спільного ресурсу **a** (КД3).

Атомарна операція:

A0 – призначена для захисту спільного ресурсу **e** (КД4).

Етап 4. Розроблення програми.

Програма складається з наступних компонентів:
основний клас **Program**, який є вхідною точкою програми;
клас **Data**, що містить усі загальні змінні та статичні методи для роботи з векторами та матрицями;
а також класи **T1**, **T2**, **T3**, **T4**, які відповідають відповідним потокам у програмі.

Лістинг програми

Program.cs

```
// Програмне забезпечення високопродуктивних комп'ютерних систем
// Лабораторна робота 3
// Варіант 8
// X= p*max(C*(MA*MD))*R+ e*B
// Котенко Ярослав Олегович
// Група ІМ-13
// 12.04.2024

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

namespace lab3
{
```

```
    internal class Program
    {
        public static void Main(string[] args)
        {
            int P = 4;
            int N = 1500;
            Data data = new Data(P, N);
            Stopwatch sw = new Stopwatch();
            sw.Start();

            var t1 = new Thread(() => T1.run(data));
            var t2 = new Thread(() => T2.run(data));
            var t3 = new Thread(() => T3.run(data));
            var t4 = new Thread(() => T4.run(data));
            t1.Start();
            t2.Start();
            t3.Start();
            t4.Start();
            t1.Join();
            t2.Join();
            t3.Join();
            t4.Join();
            sw.Stop();
            Console.WriteLine("Execution time: " + sw.Elapsed);
            Console.ReadKey();
        }
    }
}
```

Data.cs

```
using System;

namespace lab3
{
    public class Data
    {
        public int N;
        public int P;
        public int H;
        public Data(int p, int n)
        {
            N = n;
            P = p;
            H = n / p;
        }
        public int[,] MA;
        public int[,] MD;
        public int[,] MC;
        public int[] C;
        public int[] R;
        public int[] B;
        public int[] K;
        public int[] X;

        public long e = Int32.MaxValue;
        public int p;
        public int a;

        public EventWaitHandle E1 = new EventWaitHandle(false, EventResetMode.ManualReset);
        public EventWaitHandle E2 = new EventWaitHandle(false, EventResetMode.ManualReset);
        public EventWaitHandle E3 = new EventWaitHandle(false, EventResetMode.ManualReset);
        public EventWaitHandle E4 = new EventWaitHandle(false, EventResetMode.ManualReset);

        public Semaphore S1 = new Semaphore(0, 3);
```

```
        public Semaphore S2 = new Semaphore(0, 3);
        public Semaphore S3 = new Semaphore(0, 3);
        public Semaphore S4 = new Semaphore(0, 3);
        public Semaphore S5 = new Semaphore(1, 1);
        public Barrier B0 = new Barrier(4);

        public Mutex M0 = new Mutex();
        public object CS0 = new object();

        // Метод для заповнення скаляру одиницею
        public static int fillScalar()
        {
            return 1;
        }

        // Метод для заповнення вектору одиницями
        public static int[] fillVector(int N)
        {
            int[] result = new int[N];
            for (int i = 0; i < N; i++)
            {
                result[i] = 1;
            }
            return result;
        }

        // Метод для заповнення матриці одиницями
        public static int[,] fillMatrix(int N)
        {
            int[,] result = new int[N, N];
            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < N; j++)
                {
                    result[i, j] = 1;
                }
            }
            return result;
        }

        // Метод для обчислення 1
        public static int calculateStepOne(Data data, int start, int end)
        {
            int[,] MC = MultiplyMatrixByMatrix(data.MA, data.MD, start, data);
            int[] A = MultiplyVectorByMatrix(data.C, MC);

            int ai = findMaxInVector(A);
            return ai;
        }

        // Метод для обчислення 3
        public static int[] calculateStepThree(int[] R_H, int[] B_H, int p, int a, int e)
        {
            return AddVectors(ScalarMultiplyByVector(p, ScalarMultiplyByVector(a, R_H)), ScalarMultiplyByVector(e,
B_H));
        }

        // Метод для множення вектора на матрицю
        private static int[] MultiplyVectorByMatrix(int[] vector, int[,] matrix)
        {
            int vectorLength = vector.Length;
            int matrixColumns = matrix.GetLength(1);

            int[] result = new int[matrixColumns];

            for (int i = 0; i < matrixColumns; i++)
```

```
    {
        int sum = 0;
        for (int j = 0; j < vectorLength; j++)
        {
            sum += vector[j] * matrix[j, i];
        }
        result[i] = sum;
    }

    return result;
}

// Метод для множення двох матриць
private static int[,] MultiplyMatrixByMatrix(int[,] firstMatrix, int[,] secondMatrix, int start, Data data)
{
    int[,] result = new int[data.N, data.H];
    for (int i = 0; i < data.N; i++)
    {
        for (int j = start; j < start + data.H; j++)
        {
            int sum = 0;
            for (int k = 0; k < data.N; k++)
            {
                sum += firstMatrix[i, k] * secondMatrix[k, j];
            }
            result[i, j - start] = sum;
        }
    }
    return result;
}

// Метод для знаходження максимуму у векторі
private static int findMaxInVector(int[] vector)
{
    int max = vector[0];
    for (int i = 0; i < vector.Length; i++)
    {
        if (vector[i] > max)
        {
            max = vector[i];
        }
    }
    return max;
}

// Метод для визначення максимуму серед двох скалярів
public static int max(int fNum, int sNum)
{
    if (fNum > sNum) return fNum;
    return sNum;
}

// Метод для отримання частини вектора
public static int[] GetSubvector(int[] sourceVector, int startPos, int endPos)
{
    int size = endPos - startPos;
    int[] subvector = new int[size];

    for (int i = startPos; i < endPos; i++)
    {
        subvector[i - startPos] = sourceVector[i];
    }

    return subvector;
}
```

```
        // Метод для множення скаляру на вектор
        public static int[] ScalarMultiplyByVector(int scalar, int[] vector)
        {
            int[] result = new int[vector.Length];
            for (int i = 0; i < vector.Length; i++)
            {
                result[i] = scalar * vector[i];
            }
            return result;
        }

        // Метод для додавання двох векторів
        public static int[] AddVectors(int[] vectorA, int[] vectorB)
        {
            int[] result = new int[vectorA.Length];
            for (int i = 0; i < vectorA.Length; i++)
            {
                result[i] = vectorA[i] + vectorB[i];
            }
            return result;
        }

        // Метод для об'єднання фінального результату X
        public static void InsertIntoFullX(Data data, int[] sub_X, int startPos, int endPos)
        {
            for (int i = startPos, j = 0; i < endPos; i++, j++)
            {
                data.X[i] = sub_X[j];
            }
        }

        // Виведення фінального результату
        public static void PrintX(Data data)
        {
            Console.WriteLine("X : [ " + string.Join(" ", data.X) + " ]");
        }
    }
}
```

T1.cs

```
using System;

namespace lab3
{
    public class T1
    {
        private static int p1;
        private static int a1;
        private static int e1;
        public static void run(Data data)
        {
            Console.WriteLine("T1 started");
            // Введення e, X
            data.e = Data.fillScalar();
            data.X = new int[data.N];
            data.X = Data.fillVector(data.N);

            // Сигнал про введення e, X задачам T2, T3, T4
            data.E1.Set();

            // Чекати завершення введення даних у потоках T2, T3, T4
            data.E2.WaitOne();
            data.E3.WaitOne();
            data.E4.WaitOne();
```

```
            // Обчислення 1: a1 = max(C * (MA * MDH))
            a1 = Data.calculateStepOne(data, 0, data.H);

            // КД 1. Mutex. Обчислення 2: a = max(a, a1)
            data.M0.WaitOne();
            data.a = Data.max(data.a, a1);
            data.M0.ReleaseMutex();

            // Сигнал про обчислення "a" потокам T2,T3,T4
            data.S1.Release(3);

            // Чекати завершення обчислення "a" у потоках T2,T3,T4
            data.S2.WaitOne();
            data.S3.WaitOne();
            data.S4.WaitOne();

            // КД2. Semaphore. Копія p1 = p
            data.S5.WaitOne();
            p1 = data.p;
            data.S5.Release();

            // КД3. Критична секція. Копія a1 = a
            lock (data.CS0)
            {
                a1 = data.a;
            }

            // КД4. Атомарна операція. Копія e1 = e
            e1 = (int)Interlocked.Read(ref data.e);

            // Обчислення 3: 3) Xн = p1 * a1 * Rн + e1 * Bн
            int[] R_H = Data.GetSubvector(data.R, 0, data.H);
            int[] B_H = Data.GetSubvector(data.B, 0, data.H);

            int[] X_H = Data.calculateStepThree(R_H, B_H, p1, a1, e1);

            // Об'єднання фінального результату
            Data.InsertIntoFullX(data, X_H, 0, data.H);

            // Сигнал про завершення обчислення Xн потоку T4
            data.B0.SignalAndWait();
            Console.WriteLine("T1 finished");

        }
    }
}
```

T2.cs

```
using System;

namespace lab3
{
    public class T2
    {
        private static int p2;
        private static int a2;
        private static int e2;
        public static void run(Data data)
        {
            Console.WriteLine("T2 started");
            // Введення C, MA
            data.C = new int[data.N];
            data.MA = new int[data.N, data.N];
```

```
            data.C = Data.fillVector(data.N);
            data.MA = Data.fillMatrix(data.N);

            // Сигнал про введення C, MA задачам T1, T3, T4
            data.E2.Set();

            // Чекати завершення введення даних у потоках T1, T3, T4
            data.E1.WaitOne();
            data.E3.WaitOne();
            data.E4.WaitOne();

            // Обчислення 1: a2 = max(C * (MA * MDH))
            a2 = Data.calculateStepOne(data, data.H, data.H * 2);

            // КД 1. Mutex. Обчислення 2: a = max(a, a2)
            data.M0.WaitOne();
            data.a = Data.max(data.a, a2);
            data.M0.ReleaseMutex();

            // Сигнал про обчислення "a" потокам T1,T3,T4
            data.S2.Release(3);

            // Чекати завершення обчислення "a" у потоках T1,T3,T4
            data.S1.WaitOne();
            data.S3.WaitOne();
            data.S4.WaitOne();

            // КД2. Semaphore. Копія p2 = p
            data.S5.WaitOne();
            p2 = data.p;
            data.S5.Release();

            // КД3. Критична секція. Копія a2 = a
            lock (data.CS0)
            {
                a2 = data.a;
            }

            // КД4. Атомарна операція. Копія e2 = e
            e2 = (int)Interlocked.Read(ref data.e);

            // Обчислення 3: 3) Xн = p2 * a2 * Rн + e2 * Bн
            int[] R_H = Data.GetSubvector(data.R, data.H, data.H * 2);
            int[] B_H = Data.GetSubvector(data.B, data.H, data.H * 2);

            int[] X_H = Data.calculateStepThree(R_H, B_H, p2, a2, e2);

            // Об'єднання фінального результату
            Data.InsertIntoFullX(data, X_H, data.H, data.H * 2);

            // Сигнал про завершення обчислення Xн потоку T4
            data.B0.SignalAndWait();
            Console.WriteLine("T2 finished");
        }
    }
}
```

T3.cs

```
using System;

namespace lab3
{
    public class T3
    {
```

```
        private static int p3;
        private static int a3;
        private static int e3;
        public static void run(Data data)
        {
            Console.WriteLine("T3 started");
            // Введення R,MD
            data.R = new int[data.N];
            data.MD = new int[data.N, data.N];
            data.R = Data.fillVector(data.N);
            data.MD = Data.fillMatrix(data.N);

            // Сигнал про введення R,MD задачам T1, T2, T4
            data.E3.Set();

            // Чекати завершення введення даних у потоках T1, T2, T4
            data.E1.WaitOne();
            data.E2.WaitOne();
            data.E4.WaitOne();

            // Обчислення 1: a3 = max(C * (MA * MDH))
            a3 = Data.calculateStepOne(data, data.H * 2, data.H * 3);

            // КД 1. Mutex. Обчислення 2: a = max(a, a3)
            data.M0.WaitOne();
            data.a = Data.max(data.a, a3);
            data.M0.ReleaseMutex();

            // Сигнал про обчислення "a" потокам T1,T2,T4
            data.S3.Release(3);

            // Чекати завершення обчислення "a" у потоках T1,T2,T4
            data.S1.WaitOne();
            data.S2.WaitOne();
            data.S4.WaitOne();

            // КД2. Semaphore. Копія p3 = p
            data.S5.WaitOne();
            p3 = data.p;
            data.S5.Release();

            // КД3. Критична секція. Копія a3 = a
            lock (data.CS0)
            {
                a3 = data.a;
            }

            // КД4. Атомарна операція. Копія e3 = e
            e3 = (int)Interlocked.Read(ref data.e);

            // Обчислення 3: 3) Xн = p3 * a3 * Rн + e3 * Bн
            int[] R_H = Data.GetSubvector(data.R, data.H * 2, data.H * 3);
            int[] B_H = Data.GetSubvector(data.B, data.H * 2, data.H * 3);

            int[] X_H = Data.calculateStepThree(R_H, B_H, p3, a3, e3);

            // Об'єднання фінального результату
            Data.InsertIntoFullX(data, X_H, data.H * 2, data.H * 3);

            // Сигнал про завершення обчислення Xн потоку T4
            data.B0.SignalAndWait();
            Console.WriteLine("T3 finished");
        }

    }
}
```

T4.cs

```
using System;

namespace lab3
{
   public class T4
   {
      private static int p4;
      private static int a4;
      private static int e4;
      public static void run(Data data)
      {
         Console.WriteLine("T4 started");
         // Введення B,p
         data.p = Data.fillScalar();
         data.B = new int[data.N];
         data.B = Data.fillVector(data.N);

         // Сигнал про введення B, p задачам T1, T2, T3
         data.E4.Set();

         // Чекати завершення введення даних у потоках T1, T2, T3
         data.E1.WaitOne();
         data.E2.WaitOne();
         data.E3.WaitOne();

         // Обчислення 1: a4 = max(C * (MA * MDH))
         a4 = Data.calculateStepOne(data, data.H, data.H * 2);

         // КД 1. Mutex. Обчислення 2: a = max(a, a4)
         data.M0.WaitOne();
         data.a = Data.max(data.a, a4);
         data.M0.ReleaseMutex();

         // Сигнал про обчислення "a" потокам T1,T2,T3
         data.S4.Release(3);

         // Чекати завершення обчислення "a" у потоках T1,T2,T3
         data.S1.WaitOne();
         data.S2.WaitOne();
         data.S3.WaitOne();

         // КД2. Semaphore. Копія p4 = p
         data.S5.WaitOne();
         p4 = data.p;
         data.S5.Release();

         // КД3. Критична секція. Копія a4 = a
         lock (data.CS0)
         {
            a4 = data.a;
         }

         // КД4. Атомарна операція. Копія e4 = e
         e4 = (int)Interlocked.Read(ref data.e);

         // Обчислення 3: 3) Xн = p4 * a4 * Rн + e4 * Bн
         int[] R_H = Data.GetSubvector(data.R, data.H * 3, data.N);
         int[] B_H = Data.GetSubvector(data.B, data.H * 3, data.N);

         int[] X_H = Data.calculateStepThree(R_H, B_H, p4, a4, e4);

         // Об'єднання фінального результату
```

```
            Data.InsertIntoFullX(data, X_H, data.H * 3, data.N);

            // Чекати завершення обчислення Xн в потоках T1,T2,T3
            data.B0.SignalAndWait();

            // Виведення результату
            Data.PrintX(data);
            Console.WriteLine("T4 finished");
        }
    }
}
```

Результат виконання програми для N = 16.

```
T1 started
T2 started
T3 started
T4 started
T2 finished
T1 finished
T3 finished
X : [ 257 257 257 257 257 257 257 257 257 257 257 257 257 257 257 257 ]
T4 finished
Execution time: 00:00:00.0677274
```

Тестування:

Опис комп'ютера:

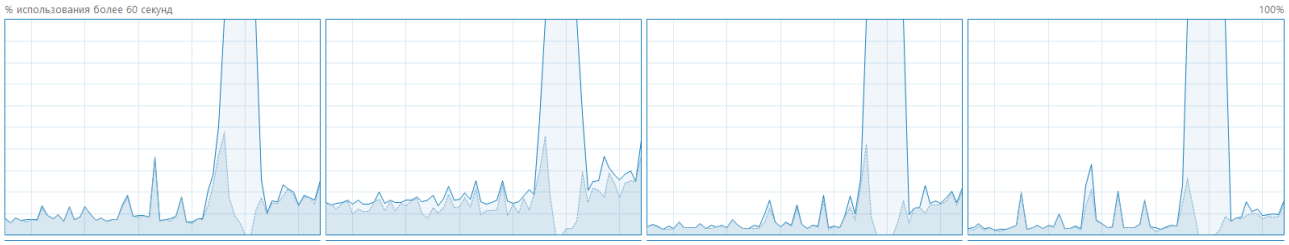
AMD Ryzen 7 4700U with Radeon Graphics

Базовая скорость:	2,00 ГГц
Сокетов:	1
Ядра:	8
Логических процессоров:	8

Для N = 1500

Час виконання на 1 ядрі: 20,77 с .

Час виконання на 4 ядрах: 7,36 с.



Коефіцієнт прискорення дорівнює:

$$K_y = \frac{T_1}{T_4} = \frac{20,77}{7,36} = 2,82$$

Висновок

1. Лабораторна робота була виконана з використанням мови програмування C#. Для створення багатопотокової програми був використаний клас **Thread**. У процесі розробки використовувалися різні засоби для роботи з потоками та синхронізації, зокрема: клас **Semaphore** для управління семафорами, клас **EventWaitHandle** для управління подіями, метод **Interlocked.Read()** для роботи з атомік-змінною, клас **Barrier** для роботи з бар'єром, клас **Mutex** для роботи з мютексами, та конструкція **lock** для обробки критичних секцій.

2. У процесі розробки було створено паралельний математичний алгоритм, який включає в себе множення вектора на матрицю, множення матриць, множення скаляра на вектор, знаходження максимуму вектора, а також додавання векторів. Також, було визначено спільні ресурси (СР): скаляри p, a, e, вектор C та матриця MA.

3. Був розроблений алгоритми для потоків, де кожен потік виконує паралельне обчислення своєї частини задачі. Також було визначено точки синхронізації та критичні ділянки КД1-КД4.

4. Була розроблена структурна схема взаємодії потоків, в якій використані наступні засоби організації:
a. Події використовуються для синхронізації введення даних.
b. Семафори використовуються для надсилання та приймання сигналу про завершення обчислення **a** та контролю копіювання спільного ресурсу **p**.
c. Бар'єр застосовано для синхронізації виведення фінального результату.
d. М'ютекс використовується для захисту спільного ресурсу **a** (КД1).
e. Критична секція застосована для безпечного копіювання спільного ресурсу **a** (КД3).

f. Атомарна операція застосована для безпечного копіювання спільного ресурсу **e**.

5. Було проведене тестування, результати якого підтвердили ефективність багатопотокової програми. При значенні N = 1500 коефіцієнт прискорення склав 2,82.