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

# Лабораторна робота №2

з дисципліни

Програмне забезпечення високопродуктивних комп'ютерних систем

ВИКОНАЛА:
Студентка групи ІМ-21
Рабійчук Дар’я Олександрівна
№ у списку(варіант) - 18

ПЕРЕВІРИВ:
доц. Корочкін О. В.

**Київ 2025**

## Завдання

1) Розробити паралельний алгоритм рішення математичної задачі; виявити спільні ресурси для потоків
2) Описати алгоритм кожного потоку (T1–T4) з визначенням критичних ділянок (КД) і точок синхронізації (Wij , Sij);
3) Розробити структурну схему взаємодії задач, де застосувати ВСІ вказані засоби взаємодії процесів
4) Розробити програму (обов'язкові "шапка" (header), коментарі), виконати налагодження програми; отримати правильні результати обчислень.
5) За допомогою Диспетчеру задач Windows проконтролювати завантаження ядер процесору.
6) Провести тестування програми для різних значень N (1000 і більше) і кількості процесорів (ядер) 1 та 4 з метою отримання часу виконання програми Час1 та Час4 визначення коефіцієнту прискорення Кп = Час1/Час4.

**Варіант 22**

W = max(C*MD)*C + E*(MA*MB)*d

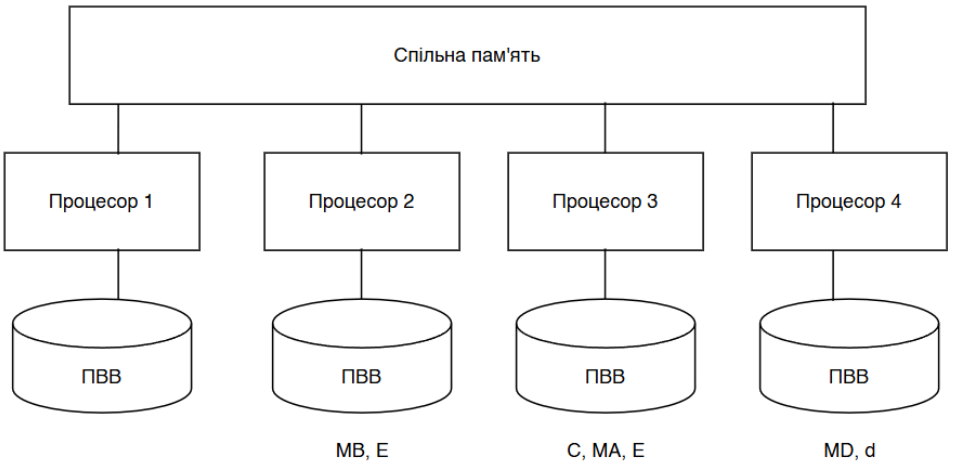
Введення-виведення даних:

1 –

2 MB, E

3 C, MA, W

4 MD, d




## Етап 1. Побудова паралельного математичного алгоритму

$$W = \max(C * MD) * C + E * (MA * MB) * d$$

1) ai = max( C*MDн) | CP: C 1 = 1..P

2) a = max(a, ai) | CP: a

3) Wн = a * C н+ E * (MA * MBн) * d | CP: a, E, MA, d

CP: C, a, E, MA, d

a потребує синхроніації запису в 2 кроці. ?????

a та d потребують копіювання перед кроком 3.

**Етап 2. Розробка алгоритмів потоків**

**T1:**

**1. Чекати** на введення даних у потоках T2, T3, T4 **W2,3,4-1**

**2.** Обчислення 1: $a_1 = \max(C * MD_н)$

**3.** Обчислення 2: $a = \max(a, a_1)$ **КД1**

4. **Сигнал** T2, T3, T4 про завершення a **S2,3,4-2**

5. **Чекати** на завершення обчислень a у потоках T2, T3, T4 **W2,3,4-2**

**6. Копія** $a_1 = a$ **КД2**

**7. Копія** $b_1 = b$ **КД3**

**8.** Обчислення $W_н = a_1 * C н+ E * (MA * MBн) * d_1$

**9. Сигнал** про завершення обчислень W потоку T3 **S3-3**

**T2:**

**1.** Введення MB, ME

**2. Сигнал** задачі T1, T3, T4 про введення **S1,3,4-1**

**3. Чекати** на введення даних у потоках T3, T4 **W1,3,4-1**

**4.** Обчислення 1: $a_2 = \max(C * MDн)$

**5.** Обчислення 2: $a = \max(a, a_2)$ **КД1**

6. **Сигнал** T1, T3, T4 про завершення a **S1,3,4-2**

7. **Чекати** на завершення обчислень a у потоках T1, T3, T4 **W1,3,4-2**

**8. Копія** $a_2 = a$ **КД2**

**9. Копія** $b_2 = b$ **КД3**

**10.** Обчислення $W_H = a_2 * C н + E * (MA * MBн) * d_2$

**11. Сигнал** про завершення обчислень W потоку T3 **S3-3**

**T3:**

**1.** Введення C, MA

**2. Сигнал** задачі T1, T2, T4 про введення **S1,2,4-1**

**3. Чекати** на введення даних у потоках T2, T4 **W1,2,4-1**

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

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

6. **Сигнал** T1, T2, T4 про завершення a **S1,2,4-2**

7. **Чекати** на завершення обчислень a у потоках T1, T2, T4 **W1,2,4-2**

**8. Копія** $a_3 = a$ **КД2**

**9. Копія** $b_3 = b$ **КД3**

**10.** Обчислення $W_H = a_3 * C н + E * (MA * MBн) * d_3$

**11. Чекати** на завершення обчислень W **W1,2,4-3**

**12.** Вивід W

**T4:**

**1.** Введення MD, d

**2. Сигнал** задачі T1, T2, T3 про введення **S1,2,3-1**

**3. Чекати** на введення даних у потоках T2, T3 **W1,2,3-1**

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

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

6. **Сигнал** T1, T2, T3 про завершення a **S1,2,3-2**

7. **Чекати** на завершення обчислень a у потоках T1, T2, T3 **W1,2,3-2**

**8. Копія** $a_4 = a$ **КД2**

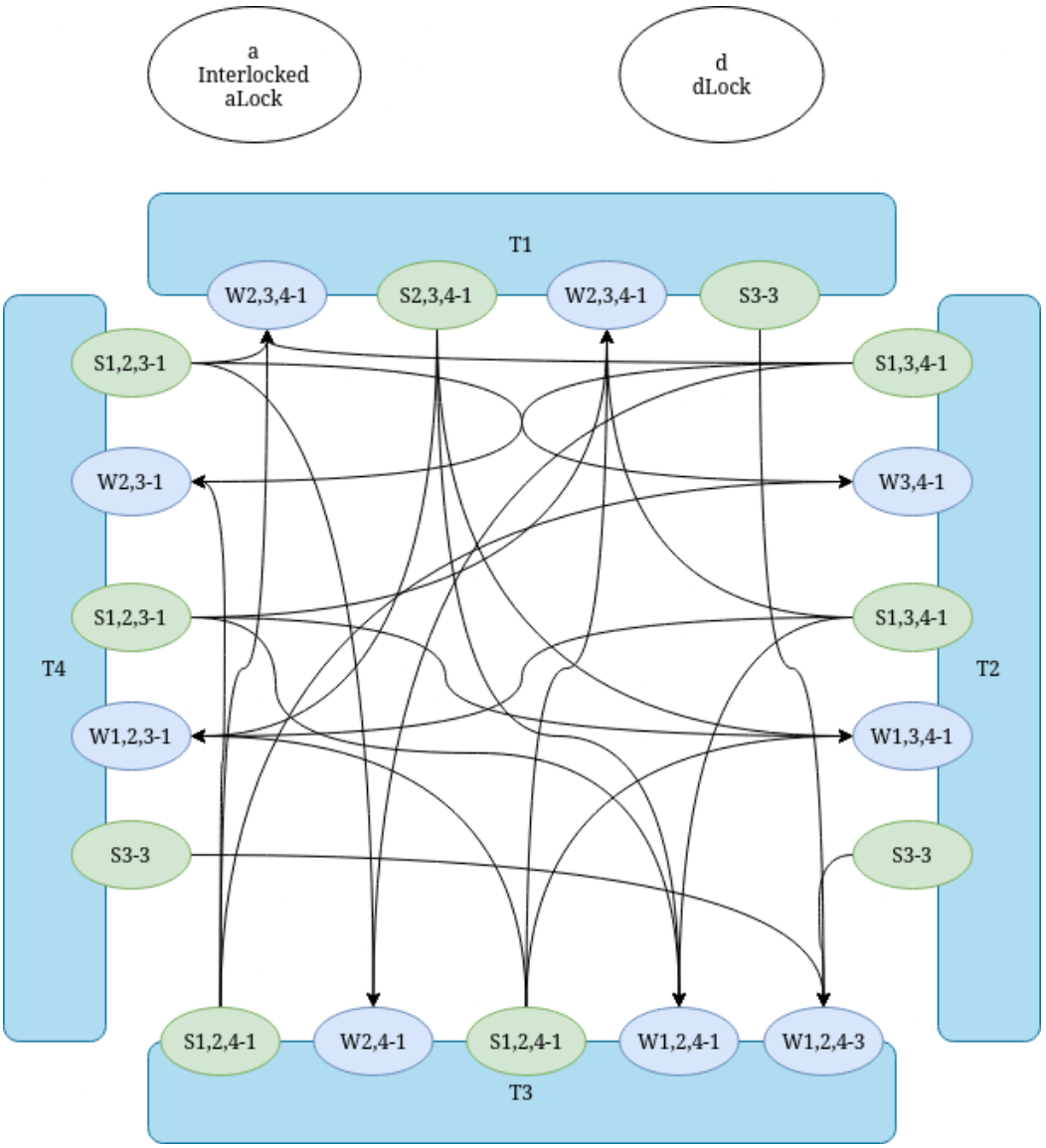
**9. Копія** $b_4 = b$ **КД3**

**10.** Обчислення $W_H = a_4$ * С н+ Е * (МА * МВн) * $d_4$

**11. Сигнал** про завершення обчислень W потоку Т3 **S3-3**

## Етап 3. Розробка схеми

**Опис засобів? Три КД?**




Для синхронізації на першому та 3 етапах будуть використані семафори, для 2 етапу буде застосовано бар'єр.

## Етап 4. Розробка програми

## Program.cs

```
// Програмне забезпечення високопродуктивних комп'ютерних систем

// Лабораторна робота No2: Семафори, Критичні секції, Атомік змінні, Бар'єри
// Варіант 22
// W = max(C * MD)*C + E * (MA * MB) * d
// Рабійчук Дар'я Олександрівна
// 15.04.2025

class Program
{
    const int N = 4; // розмірність
    const int P = 4; // кількість потоків

    static int[] C = new int[N];
    static int[] E = new int[N];
    static int[] W = new int[N];
    static int[,] MA = new int[N, N];
    static int[,] MB = new int[N, N];
    static int[,] MD = new int[N, N];

    static int d; // Змінна для d
    static int a; // Змінна для максимального значення (atomic via Interlocked)
    static object dLock = new object(); // Lock для копіювання d
    static object aLock = new object(); // Lock для копіювання a
    static SemaphoreSlim inputSem2 = new SemaphoreSlim(0, P); // Семафори для
першої синхронізації (введення)
    static SemaphoreSlim inputSem3 = new SemaphoreSlim(0, P);
    static SemaphoreSlim inputSem4 = new SemaphoreSlim(0, P);
    static Barrier barrierMax = new Barrier(P); // Бар'єр для другої синхронізації
(a)
    static SemaphoreSlim finalSem = new SemaphoreSlim(0, P); // Семафор для
третьої синхронізації (W)

    static void Main()
    {
        Thread t1 = new Thread(Thread1) { Name = "Thread1" };
        Thread t2 = new Thread(Thread2) { Name = "Thread2" };
        Thread t3 = new Thread(Thread3) { Name = "Thread3" };
        Thread t4 = new Thread(Thread4) { Name = "Thread4" };

        t1.Start();
        t2.Start();
        t3.Start();
        t4.Start();
    }

    static void Thread1()
    {
        Console.WriteLine("T1 is started");

        int start = 0;
        int end = N / P;

        // T1 Крок 1: Чекати на введення даних у потоках T2, T3, T4 (W2,3,4-1)
```

```
        inputSem2.Wait();
        inputSem3.Wait();
        inputSem4.Wait();

        // T1 Крок 2: Обчислення 1: a1 = max(C * MDH)
        int localMax = MaxC_MDn(start, end);

        // T1 Крок 3: Обчислення 2: a = max(a, a1) (КД1)
        Interlocked.CompareExchange(ref a, Math.Max(a, localMax), a);

        // T1 Крок 4: Сигнал T2, T3, T4 про завершення a (S2,3,4-2)
        barrierMax.SignalAndWait(); // Бар'єр для синхронізації a

        // T1 Крок 5: Чекати на завершення обчислень a у потоках T2, T3, T4
(W2,3,4-2)
        // Виконується автоматично через бар'єр вище

        // T1 Крок 6: Копія a1 = a (КД2)
        int a1;
        lock (aLock) { a1 = a; } // Захищене копіювання a

        // T1 Крок 7: Копія b1 = b (КД3)
        int d1;
        lock (dLock) { d1 = d; } // Захищене копіювання d

        // T1 Крок 8: Обчислення WH = a1 * Cн + E * (MA * MBн) * d1
        MultiplyCByA(a1, start, end);
        int[,] D = MultiplyMA_MB(start, end);
        MultiplyEDdAndAddToW(D, d1, start, end);

        // T1 Крок 9: Сигнал про завершення обчислень W потоку T3 (S3-3)
        finalSem.Release();

        Console.WriteLine("T1 is finished");
    }

    static void Thread2()
    {
        Console.WriteLine("T2 is started");

        int start = N / P;
        int end = N / P * 2;

        // T2 Крок 1: Введення MB, ME
        E = Data.StaticVectorInitialize(1, N);
        MB = Data.StaticMatrixInitialize(1, N, N);

        // T2 Крок 2: Сигнал задачі T1, T3, T4 про введення (S1,3,4-1)
        inputSem2.Release(3);

        // T2 Крок 3: Чекати на введення даних у потоках T3, T4 (W1,3,4-1)
        inputSem3.Wait();
        inputSem4.Wait();

        // T2 Крок 4: Обчислення 1: a2 = max(C * MDн)
        int localMax = MaxC_MDn(start, end);

        // T2 Крок 5: Обчислення 2: a = max(a, a2) (КД1)
        Interlocked.CompareExchange(ref a, Math.Max(a, localMax), a);
```

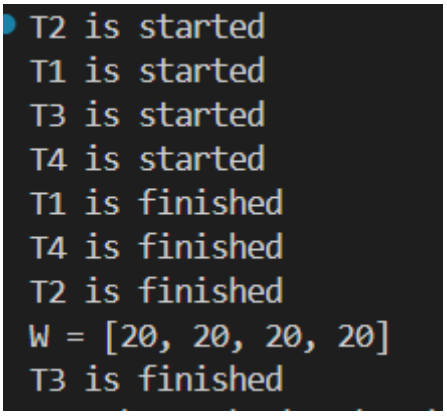
```
        // T2 Крок 6: Сигнал T1, T3, T4 про завершення a (S1,3,4-2)
        barrierMax.SignalAndWait(); // Бар'єр для синхронізації a

        // T2 Крок 7: Чекати на завершення обчислень a у потоках T1, T3, T4
(W1,3,4-2)
        // Виконується автоматично через бар'єр вище

        // T2 Крок 8: Копія a2 = a (КД2)
        int a2;
        lock (aLock) { a2 = a; }

        // T2 Крок 9: Копія b2 = b (КД3)
        int d2;
        lock (dLock) { d2 = d; } // Захищене копіювання d

        // T2 Крок 10: Обчислення WH = a2 * Сн + E * (MA * MBн) * d2
        MultiplyCByA(a2, start, end);
        int[,] D = MultiplyMA_MB(start, end);
        MultiplyEDdAndAddToW(D, d2, start, end);

        // T2 Крок 11: Сигнал про завершення обчислень W потоку T3 (S3-3)
        finalSem.Release();

        Console.WriteLine("T2 is finished");
    }

    static void Thread3()
    {
        Console.WriteLine("T3 is started");

        int start = N / P * 2;
        int end = N / P * 3;

        // T3 Крок 1: Введення C, MA
        C = Data.StaticVectorInitialize(1, N);
        MA = Data.StaticMatrixInitialize(1, N, N);

        // T3 Крок 2: Сигнал задачі T1, T2, T4 про введення (S1,2,4-1)
        inputSem3.Release(3);

        // T3 Крок 3: Чекати на введення даних у потоках T2, T4 (W1,2,4-1)
        inputSem2.Wait();
        inputSem4.Wait();

        // T3 Крок 4: Обчислення 1: a3 = max(C * MDн)
        int localMax = MaxC_MDn(start, end);

        // T3 Крок 5: Обчислення 2: a = max(a, a3) (КД1)
        Interlocked.CompareExchange(ref a, Math.Max(a, localMax), a);

        // T3 Крок 6: Сигнал T1, T2, T4 про завершення a (S1,2,4-2)
        barrierMax.SignalAndWait(); // Бар'єр для синхронізації a

        // T3 Крок 7: Чекати на завершення обчислень a у потоках T1, T2, T4
(W1,2,4-2)
        // Виконується автоматично через бар'єр вище

        // T3 Крок 8: Копія a3 = a (КД2)
```

```
        int a3;
        lock (aLock) { a3 = a; }

        // T3 Крок 9: Копія b3 = b (КД3)
        int d3;
        lock (dLock) { d3 = d; } // Захищене копіювання d

        // T3 Крок 10: Обчислення WH = a3 * Cн + E * (MA * MBн) * d3
        MultiplyCByA(a3, start, end);
        int[,] D = MultiplyMA_MB(start, end);
        MultiplyEDdAndAddToW(D, d3, start, end);

        // T3 Крок 11: Чекати на завершення обчислень W (W1,2,4-3)
        finalSem.Wait();
        finalSem.Wait();
        finalSem.Wait();

        // T3 Крок 12: Вивід W
        FinalAdd();
        Console.WriteLine("W = [" + string.Join(", ", W) + "]");

        Console.WriteLine("T3 is finished");
    }

    static void Thread4()
    {
        Console.WriteLine("T4 is started");

        int start = N / P * 3;
        int end = N;

        // T4 Крок 1: Введення MD, d
        d = 1; // Значення для d
        MD = Data.StaticMatrixInitialize(1, N, N);

        // T4 Крок 2: Сигнал задачі T1, T2, T3 про введення (S1,2,3-1)
        inputSem4.Release(4);

        // T4 Крок 3: Чекати на введення даних у потоках T2, T3 (W1,2,3-1)
        inputSem2.Wait();
        inputSem3.Wait();

        // T4 Крок 4: Обчислення 1: a4 = max(C * MDн)
        int localMax = MaxC_MDn(start, end);

        // T4 Крок 5: Обчислення 2: a = max(a, a4) (КД1)
        Interlocked.CompareExchange(ref a, Math.Max(a, localMax), a);

        // T4 Крок 6: Сигнал T1, T2, T3 про завершення a (S1,2,3-2)
        barrierMax.SignalAndWait(); // Бар'єр для синхронізації a

        // T4 Крок 7: Чекати на завершення обчислень a у потоках T1, T2, T3
(W1,2,3-2)
        // Виконується автоматично через бар'єр вище

        // T4 Крок 8: Копія a4 = a (КД2)
        int a4;
        lock (aLock) { a4 = a; } // Захищене копіювання a
```

```
        // T4 Крок 9: Копія b4 = b (КДЗ)
        int d4;
        lock (dLock) { d4 = d; } // Захищене копіювання d

        // T4 Крок 10: Обчислення WH = a4 * Сн + E * (MA * MBн) * d4
        MultiplyCByA(a4, start, end);
        int[,] D = MultiplyMA_MB(start, end);
        MultiplyEDdAndAddToW(D, d4, start, end);

        // T4 Крок 11: Сигнал про завершення обчислень W потоку T3 (S3-3)
        finalSem.Release();

        Console.WriteLine("T4 is finished");
    }

    static void Wait(SemaphoreSlim sem, int count) {
      for (int i = 0; i < count; i++) sem.Wait();
    }

    static int MaxC_MDn(int startCol, int endCol)
    {
        int max = int.MinValue;
        for (int i = startCol; i < endCol; i++)
        {
            int sum = 0;
            for (int j = 0; j < N; j++)
                sum += C[j] * MD[j, i];
            max = Math.Max(max, sum);
        }
        return max;
    }

    static void MultiplyCByA(int aValue, int start, int end)
    {
        for (int i = start; i < end; i++)
            C[i] *= aValue;
    }

    static int[,] MultiplyMA_MB(int startCol, int endCol)
    {
        int[,] result = new int[N, endCol - startCol];
        for (int i = 0; i < N; i++)
            for (int j = startCol; j < endCol; j++)
                for (int k = 0; k < N; k++)
                    result[i, j - startCol] += MA[i, k] * MB[k, j];
        return result;
    }

    static void MultiplyEDdAndAddToW(int[,] D, int dValue, int start, int end)
    {
        for (int i = start; i < end; i++)
        {
            int sum = 0;
            for (int j = 0; j < N; j++)
                sum += E[j] * D[j, i - start];
            W[i] = sum * dValue;
        }
    }
```

```
    static void FinalAdd()
    {
        for (int i = 0; i < N; i++)
            W[i] += C[i];
    }
}
```

**Вивід програми при N=4**




**Етап 5  Тестування**

На тестовій системі встановлений процесор AMD Ryzen 5 4500U, який має 6 фізичних ядер і підтримує технологію SMT. Це дозволяє отримати 12 логічних ядер для виконання обчислень.

Тестування відбувається при N=1000.

При навантаженні всіх 12 логічних ядер час виконання склав 7.6 с.

Після зменшенні доступних логічних ядер процесору до 1 час збільшився до 19.21 с.

**КП**: 19.21 / 7.6 = 2.52.

## Висновки

1) Розробка математичного алгоритму: На першому етапі було розроблено паралельний математичний алгоритм, що дозволяє розв’язувати завдання згідно з варіантом, і визначити спільні ресурси, точки синхронізації та доступу до них.

2) Розробка алгоритмів для потоків: На другому етапі були розроблені алгоритми для кожного потоку, що дозволило визначити точки синхронізації, зокрема посилання сигналів та очікування під час введення, обчислення та

виведення даних. Визначено критичні ділянки для запису та копіювання спільних ресурсів.

3) Засоби синхронізації: Використано семафори для синхронізації потоків після обчислення доданків, критичні секції для безпечного доступу до спільних ресурсів, атомарні змінні для захищеного копіювання спільних змінних і бар’єри для синхронізації після введення даних та обчислень.

4) Розробка багатопоточної програми: Програма була створена мовою C# для паралельних обчислень за допомогою багатопоточності. Кожен потік відповідає за виконання частини обчислень згідно з розробленим алгоритмом.

5) Тестування та результати: Під час тестування з різною кількістю логічних ядер було з'ясовано, що з усіма 12 логічними ядрами час виконання склав 7.6 с, тоді як при використанні лише одного ядра час збільшився до 19.21 с. Коефіцієнт прискорення склав 2.52, що свідчить про прискорення при збільшенні кількості ядер. Прискорення досить далеке від теоретичного 4, можливо через однопоточне виведення та різницю в складності обчислень для різних потоків.