Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Паралельне програмування

Лабораторна робота №4
«Мова Java. Монітори»

Виконав:
студент групи ІО-21
Безщасний Р. Р.
Номер у списку групи: 2
Перевірив:
Корочкін Олександр Володимирович

Київ 2024р.

Лабораторна робота №4.2

Тема: «Мова Java. Монітори».

Мета: розробка програми для ПКС зі СП

Індивідуальне завдання:

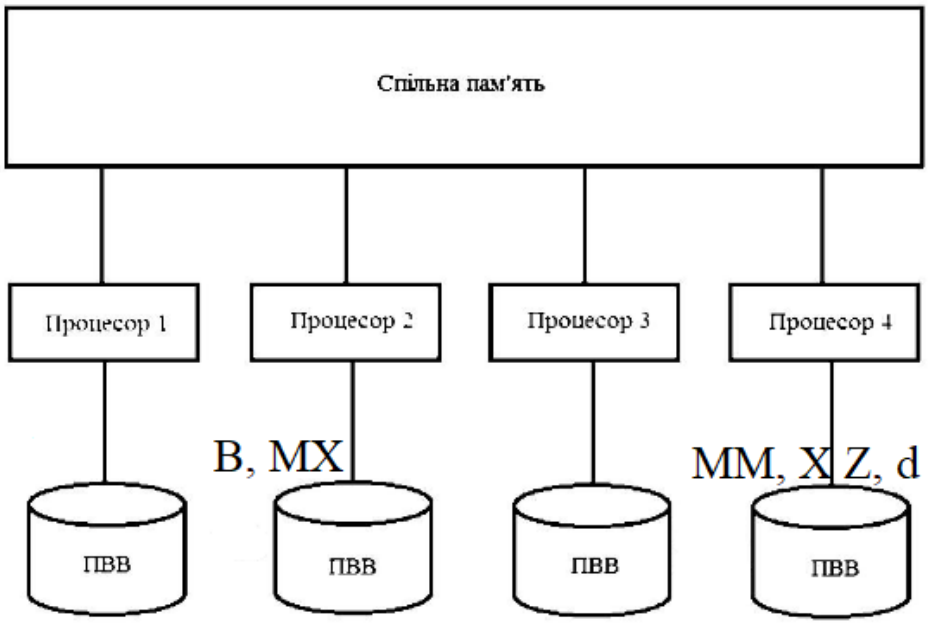
23	X = sort(d*B + Z*(MM*MX))* min(B)	-	B, MX	-	MM, X, Z, d

Етап перший: Побудова паралельного алгоритму

1) ai = min(Bh)
2) a = min(a, ai)
3) Fh =d*Bh +Z*(MM*MXh)
4) Lh = sort(Fh)
5) L2h = sort(Lh, Lh)
6) L = sort(L2h, L2h)
7) XH = Lh * a

CP: MM, Z, d, a, a

Структура паралельної комп'ютерної системи зі спільною пам'яттю:



Етап другий: Розроблення паралельних потоків

Задача T1

1	Очікування закінчення введення даних в T2, T4	W2,0 W0,4
2	Обчислення 1) a_1= min(B_H)	-
3	Обчислення 2) a= min(a, a_1)	КД1
4	Сигнал задачам T2, T3, T4 про обчислення 2)	S2,1 S3,1 S4,1
5	Сигнал про обчислення 2) у задачах T2, T3, T4	W2,1 W3,1 W4,1
6	Копія: d_1 = d	КД2

7	Копія: $a_1 = a$	КД3
8	Обчислення 3) $F_H = d_1 * B_H + Z_1 * (MM_1 * MX_H)$	-
9	Обчислення 4) $L_H = sort(F_H)$	-
10	Очікування закінчення обчислення 4) в задачі Т2	W2,2
11	Обчислення 5) $L_{2H} = sort(L_H, L_H)$	-
12	Очікування закінчення обчислення 5 в Т3	W3,2
13	Обчислення 6) $L = sort(L_{2H}, L_{2H})$	-
14	Сигнал Т2, Т3, Т4 задачам про обчислення 6)	S2,2 S3,2 S4,2
15	Обчислення 7) $X_H = L_H * a$	-
16	Сигнал 4 про завершення обчислення 7)	S4,6

Задача Т2

1	Введення: B, MX	-
2	Сигнал Т4 про введення даних	S2,0
3	Очікування закінчення введення даних в Т4	W4,0
4	Обчислення 1) $a_2 = min(B_H)$	-
5	Обчислення 2) $a = min(a, a_2)$	КД1
6	Сигнал задачам Т1, Т3, Т4 про обчислення 2)	S1,2 S3,2 S4,2
7	Сигнал про обчислення 2) у задачах Т1, Т3, Т4	W1,2, W3,2, W4,2
8	Копія: $a_2 = a$	КД2
9	Копія: $d_2 = d$	КД3
10	Обчислення 3) $F_H = d_2 * B_H + Z_2 * (MM_2 * MX_H)$	-
11	Обчислення 4) $L_H = sort(F_H)$	-
12	Сигнал задачі Т1 про завершення обчислення 4)	S1,3
13	Сигнал про обчислення 6) у Т1	W1,3
14	Обчислення 7) $X_H = L_H * a$	-
15	Сигнал 4 про завершення обчислення 7)	S4,6

Задача Т3

1	Очікування закінчення введення даних в Т2, Т4	W2,0 W4,0

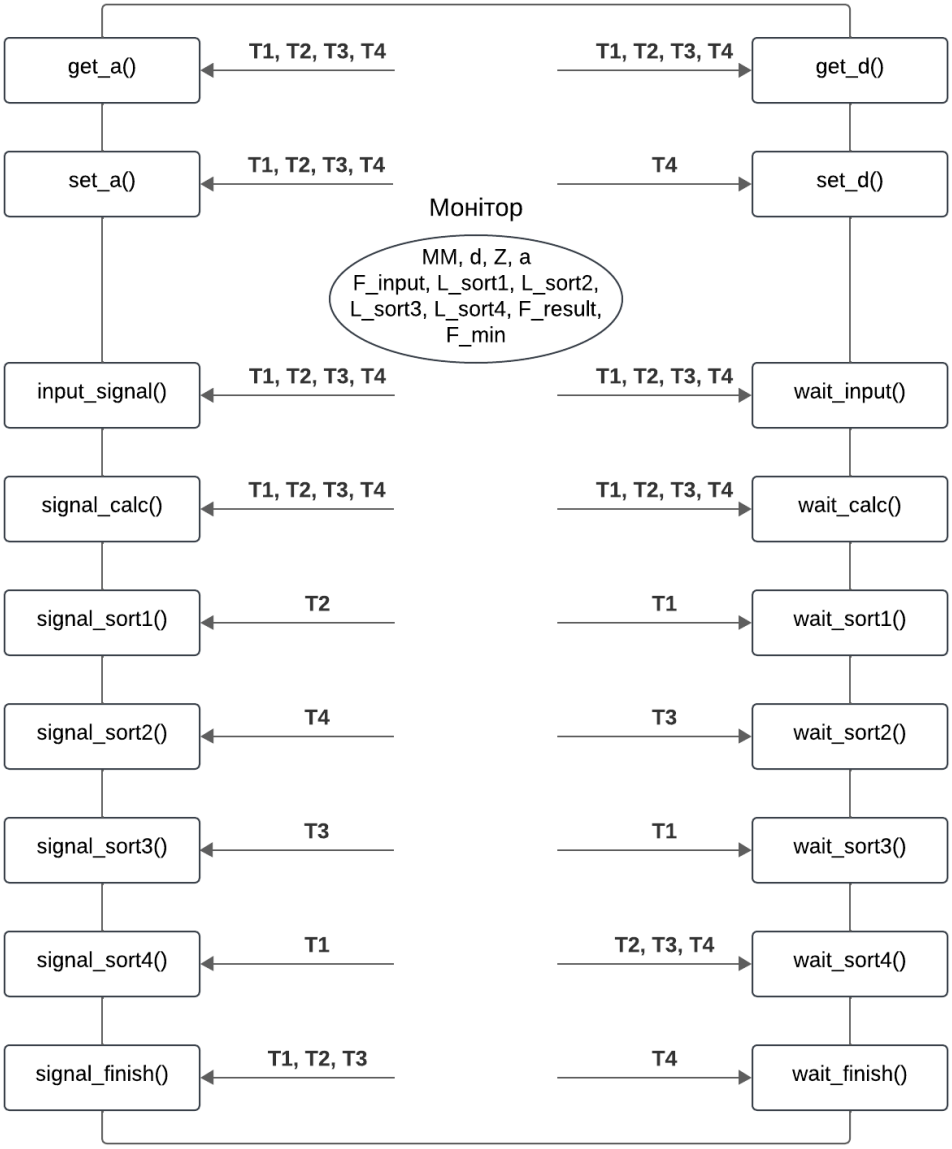
2	Обчислення 1) $a_3 = \min(B_H)$	-
3	Обчислення 2) $a = \min(a, a_3)$	КД1
4	Сигнал задачам Т1, Т2, Т4 про обчислення 2)	S1,3 S2,3 S4,3
5	Сигнал про обчислення 2) у задачах Т1, Т2, Т4	W1,3, W2,3, W4,3
6	Копія: $a_3 = a$	КД2
7	Копія: $d_3 = d$	КД3
8	Обчислення 3) $F_H = d_3 * B_H + Z_3 * (MM_3 * MX_H)$	-
9	Обчислення 4) $L_H = \text{sort}(F_H)$	-
10	Очікування закінчення обчислення 4) в задачі Т4	W4,4
11	Обчислення 5) $K_{2H} = \text{sort}(L_H, L_H)$	-
12	Сигнал задачі Т1 про обчислення 5)	S1,4
13	Сигнал про обчислення 6) у Т1	W1,4
14	Обчислення 7) $X_H = L_H * a$	-
15	Сигнал 4 про завершення обчислення 7)	S4,6

Задача Т4

1	Введення: Z, MM, d	-
2	Сигнал Т2 про введення даних	S4,0
3	Очікування закінчення введення даних в Т4	W2,0
3	Обчислення 1) $a_4 = \min(B_H)$	-
4	Обчислення 2) $a = \min(a, a_4)$	КД1
5	Сигнал задачам Т1, Т2, Т3 про обчислення 2)	S1,4 S2,4 S3,4
6	Сигнал про обчислення 2) у задачах Т1, Т2, Т3	W1,4 W2,4 W3,4
7	Копія: $a_4 = a$	КД2
8	Копія: $d_4 = d$	КД3
9	Обчислення 3) $F_H = d_4 * B_H + Z_4 * (MM_4 * MX_H)$	-
10	Обчислення 4) $L_H = \text{sort}(F_H)$	-
11	Сигнал задачі Т3 про обчислення 4)	S3,5
12	Сигнал про обчислення 6) у Т1	W1,5
13	Обчислення 7) $X_H = L_H * a$	-

14	Чекати сигнали T1, T2, T3 про обчислення 7)	W1,6, W2,6, W3,6
15	Виведення результату X	-

Етап третій:



Четвертий етап: Код програми

Lab3.java:

```
/*
Лабораторна робота №4 Варіант 23
X = sort(d * B + Z * (MM * MX)) * min(B)
T1:
T2: B, MX
T3:
T4: X, MM, d, Z
```

```
Безщасний Роман ІО-21
Дата 22.12.2024
*/

import java.util.Scanner;

public class Lab4 {
    public static void main(String[] args) {

        Monitor monitor = new Monitor();
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the matrices:");
        Data.N = scanner.nextInt();
        T1 T1 = new T1(monitor);
        T2 T2 = new T2(monitor);
        T3 T3 = new T3(monitor);
        T4 T4 = new T4(monitor);
        long startTime = System.currentTimeMillis();
        T1.start();
        T2.start();
        T3.start();
        T4.start();
        try {
            T1.join();
            T2.join();
            T3.join();
            T4.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        long endTime = System.currentTimeMillis();
        long duration = (endTime - startTime);
        System.out.println("Execution time in
milliseconds:"+duration);
    }
}
```

Monitor.java:

```
public class Monitor {
    private int d = 0;
    private int a = 0;
    private int F_input = 0;
    private int L_sort1 = 0;
    private int L_sort2 = 0;
    private int L_sort3 = 0;
    private int L_sort4 = 0;
    private int F_min = 0;
    private int F_result = 0;
    public synchronized int get_a(){ return a; }
    public synchronized int get_d(){ return d; }
    public synchronized void set_d(int d){ this.d = d; }
    public synchronized void set_a(int ai){ this.a = Math.max(this.a,
ai); }


    public synchronized void input_signal(){
        F_input += 1;
        if(F_input == 2){
            notifyAll();
        }
```

```
    }
    public synchronized void wait_input() {
        while(F_input != 2) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public synchronized void signal_sort1(){
        L_sort1 += 1;
        notifyAll();

    }
    public synchronized void wait_sort1() {
        while(L_sort1 != 1) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public synchronized void signal_sort2(){
        L_sort2 += 1;
        notifyAll();
    }
    public synchronized void wait_sort2() {
        while(L_sort2 != 1) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public synchronized void signal_sort3(){
        L_sort3 += 1;
        notifyAll();
    }
    public synchronized void wait_sort3() {
        while(L_sort3 != 1) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public synchronized void signal_sort4(){
        L_sort4 += 1;
        notifyAll();
    }
    public synchronized void wait_sort4() {
        while(L_sort4 != 1) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
```

```
        }
    }

    public synchronized void signal_calc(){
        F_min += 1;
        if(F_min == 4){ notifyAll();}
    }

    public synchronized void wait_calc() {
        while(F_min != 4) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public synchronized void signal_finish(){
        F_result += 1;
        if(F_result == 3){
            notify();
        }
    }
    public synchronized void wait_finish() {
        while(F_result != 3) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Data.java:

```
import java.util.Arrays;
import java.util.stream.IntStream;

public class Data {
    public static int N;
    public static int[] B;
    public static int[] Z;

    public static int[][] MM;
    public static int[][] MX;

    public static int[] Lh1;
    public static int[] Lh2;
    public static int[] Lh3;
    public static int[] Lh4;
    public static int[] Lh2_1;
    public static int[] Lh2_2;
    public static int[] X;

    //Функція для отримання частини вектора
    public static int[] fixArrayLength(int[] A, int start, int end) {
        int[] res = new int[end - start];
        IntStream.range(0, (end - start)).forEach(i -> res[i] =
A[i]);
```

```
        return res;
    }

    // Функція, що повертає певну частину матриці
    public static int[][] fixMatrixLength(int[][] matrix, int start,
int end) {
        int[][] res = new int[matrix.length][end - start];
        for (int i = 0; i < matrix.length; i++) {
            int ind = 0;
            for (int j = start; j < end; j++) {
                res[i][ind] = matrix[i][j];
                ind++;
            }
        }
        return res;
    }

    //Функція для множення матриць
    public static int[][] mulMatrix(int[][] vec1, int[][] vec2){
        int[][] result= new int[vec1.length][vec2[0].length];
        for (int i = 0; i < vec1.length; i++)
        {
            for (int j = 0; j < vec2[0].length; j++)
            {
                for (int k = 0; k < vec1[0].length; k++)
                {
                    result[i][j] += vec1[i][k] * vec2[k][j];
                }
            }
        }
        return result;
    }


    // Функція, що додає вектори
    public static int[] addArrays(int[] A, int[] B) {
        int[] res = new int[A.length];
        for (int i = 0; i < A.length; i++) {
            res[i] = A[i] + B[i];
        }
        return res;
    }

    // Функція, що множить вектор на матрицю
    static public int[] mulArrayMatrix(int[] A, int[][] matrix){
        int[] res = new int[matrix[0].length];
        for (int i = 0; i < matrix[0].length; i++) {
            for (int j = 0; j < A.length; j++) {
                res[i] += A[j] * matrix[j][i];
            }
        }
        return res;
    }

    // Функція, що вставляє вектор в частину іншого вектора
    public static void insertArrayRes(int[] A, int[] B, int start,
int end) {
        int ind = 0;
        for (int i = start; i < end; i++) {
            B[i] = A[ind++];
```

```
        }
    }

    // Функція для множення скаляру на вектор
    public static int[] mulScalArray(int scal, int[] A) {
        IntStream.range(0, A.length).forEach(i -> A[i] *= scal);
        return A;
    }

    //Функція, що перемножує вектори між собою
    public static int mulArrays(int[] A, int[] B) {
        int res = 0;
        for (int i = 0; i < A.length; i++) {
            res += A[i] * B[i];
        }
        return res;
    }

    //Функція для виведення вектора
    public static void printvec(int[] VH){
        for (int j : VH) {
            System.out.print(j + " ");
        }
        System.out.println();
    }
    //Функція для початовго сортування
    public static int[] Sort(int[] vec1) {
        Arrays.sort(vec1);
        return vec1;
    }
    //Функція для сортування злиттям
    public static int[] sortTwoVecs(int[] vector1, int[] vector2) {
        int[] result = new int[vector1.length + vector2.length];
        for (int i = 0; i < result.length; i++) {
            if (i < vector1.length) {
                result[i] = vector1[i];
            }
            else {
                result[i] = vector2[i-vector1.length];
            }
        }
        Arrays.sort(result);
        return result;
    }
}
```

T1.java:

```
public class T1 extends Thread {
    Monitor monitor;
    private final int start;
    private final int end;
    public T1(Monitor monitor) {

        start = 0;
        end = Data.N / 4;
        this.monitor = monitor;
    }

    @Override
    public void run(){
```

```
        System.out.println("Start Thread 1");

        // Очікування сигналів від T2, T4
        monitor.wait_input();

        // КД1
        int a1 = 1;
        monitor.set_a(a1);

        // Сигнал задачам T2, T3, T4 про обчислення 2)
        monitor.signal_calc();
        // Очікування сигналів T2, T3, T4 про обчислення
        monitor.wait_calc();

        int d1 = monitor.get_d(); // КД2
        a1 = monitor.get_a(); // КД3
        int[][] MX1 = Data.fixMatrixLength(Data.MX, start, end);
        // Обчислення 3)
        int[] F1 = Data.addArrays(Data.mulScalArray(d1,
Data.fixArrayLength(Data.B, start, end)),
                Data.mulArrayMatrix(Data.fixArrayLength(Data.Z,
start, end), Data.mulMatrix(Data.MM, MX1)));

        Data.Lh1 = Data.Sort(F1); // Обчислення 4)
        // Очікування закінчення обчислення 4) в задачі T2
        monitor.wait_sort1();
        // Обчислення 5) L2H = sort(LH, LH)
        Data.Lh2_1 = Data.sortTwoVecs(Data.Lh1, Data.Lh2);
        // Очікування закінчення обчислення 5 в T3
        monitor.wait_sort3();
        // Обчислення 6) L = sort(L2H, L2H)
        Data.X = Data.sortTwoVecs(Data.Lh2_1, Data.Lh2_2);
        // Сигнал T2, T3, T4 задачам про обчислення 6)
        monitor.signal_sort4();
        //Обчислення 7) XH = LH * a
        Data.insertArrayRes(Data.mulScalArray(a1, Data.X), Data.X,
start, end);
        // Сигнал 4 про завершення обчислення 7)
        monitor.signal_finish();

        System.out.println("Thread 1 has completed.");
    }
}
```

T2.java:

```
public class T2 extends Thread {
    private final Monitor monitor;
    private final int start;
    private final int end;
    public T2(Monitor monitor) {
        Data.B = new int[Data.N];
        Data.MX = new int[Data.N][Data.N];
        start = Data.N / 4;
        end = Data.N / 2;
        this.monitor = monitor;
    }
    @Override
    public void run() {
        System.out.println("Start Thread 2");
        for (int i = 0; i < Data.N; i++) { // Введення B, MX
```

```
            Data.B[i] = 1;
            for (int j = 0; j < Data.N; j++){
                Data.MX[i][j] = 1;
            }
        }
        // Сигнал T4 про введення даних
        monitor.input_signal();
        // Очікування закінчення введення даних в T4
        monitor.wait_input();
        // КД1
        int a2 = 1;
        monitor.set_a(a2);
        // Сигнал задачам T1, T3, T4 про обчислення 2)
        monitor.signal_calc();
        // Очікування сигналів T1, T3, T4 про обчислення
        monitor.wait_calc();

        a2 = monitor.get_a(); // КД2
        int d2 = monitor.get_d(); // КД3
        int[][] MX2 = Data.fixMatrixLength(Data.MX, start, end);
        // Обчислення 3)
        int[] F2 = Data.addArrays(Data.mulScalArray(d2,
Data.fixArrayLength(Data.B, start, end)),
                Data.mulArrayMatrix(Data.fixArrayLength(Data.Z,
start, end), Data.mulMatrix(Data.MM, MX2)));
        // Обчислення 4)
        Data.Lh2 = Data.Sort(F2);
        // Сигнал задачі T1 про завершення обчислення 4)
        monitor.signal_sort1();
        // Очікування на сигнал про обчислення 6) у T1
        monitor.wait_sort4();
        // Обчислення 7) XH = LH * a
        Data.insertArrayRes(Data.mulScalArray(a2, Data.X), Data.X,
start, end);
        // Сигнал 4 про завершення обчислення 7)
        monitor.signal_finish();

        System.out.println("Thread 2 has completed.");
    }
}
```

T3.java:

```
public class T3 extends Thread {
    private final Monitor monitor;
    private final int start;
    private final int end;
    public T3(Monitor monitor) {
        start = Data.N / 2;
        end = Data.N / 4 * 3;
        this.monitor = monitor;
    }
    @Override
    public void run() {
        System.out.println("Start Thread 3");
        // Очікування закінчення введення даних в T2, T4
        monitor.wait_input();
        // КД1
        int a3 = 1;
        monitor.set_a(a3);
        // Сигнал задачам T1, T2, T4 про обчислення 2)
```

```
        monitor.signal_calc();
        // Очікування сигналів T1, T2, T4 про обчислення
        monitor.wait_calc();

        a3 = monitor.get_a(); // КД 2
        int d3 = monitor.get_d(); // КД3
        int[][] MX3 = Data.fixMatrixLength(Data.MX, start, end);
        // Обчислення 3)
        int[] F3 = Data.addArrays(Data.mulScalArray(d3,
Data.fixArrayLength(Data.B, start, end)),
                Data.mulArrayMatrix(Data.fixArrayLength(Data.Z,
start, end), Data.mulMatrix(Data.MM, MX3)));
        // Обчислення 4)
        Data.Lh3 = Data.Sort(F3);
        // Очікування закінчення обчислення 4) в задачі T4
        monitor.wait_sort2();
        // Обчислення 5) K2H = sort(LH, LH)
        Data.Lh2_2 = Data.sortTwoVecs(Data.Lh3, Data.Lh4);
        // Сигнал задачі T1 про обчислення 5)
        monitor.signal_sort3();
        // Сигнал про обчислення 6) у T1
        monitor.wait_sort4();
        // Обчислення 7) XH = LH * a
        Data.insertArrayRes(Data.mulScalArray(a3, Data.X), Data.X,
start, end);
        // Сигнал 4 про завершення обчислення 7)
        monitor.signal_finish();

        System.out.println("Thread 3 has completed.");
    }

}
```

T4.java:

```
public class T4 extends Thread {
    private final Monitor monitor;
    private final int start;
    private final int end;

    public T4(Monitor monitor) {
        Data.MM = new int[Data.N][Data.N];
        Data.Z = new int[Data.N];
        Data.X = new int[Data.N];
        start = Data.N / 4 * 3;
        end = Data.N;
        this.monitor = monitor;
    }
    @Override
    public void run(){
        System.out.println("Start Thread 4");
        for (int i = 0; i < Data.N; i++) { // Введення Z, MM, d
            Data.X[i] = 1;
            Data.Z[i] = 1;
            for (int j = 0; j < Data.N; j++){
                Data.MM[i][j] = 1;
            }
        }

        monitor.set_d(1);
```

```
        // Сигнал T2 про введення даних
        monitor.input_signal();
        // Очікування закінчення введення даних в T4
        monitor.wait_input();
        // КД1
        int a4 = 1;
        monitor.set_a(a4);
        // Сигнал задачам T1, T2, T3 про обчислення 2)
        monitor.signal_calc();
        // Сигнал про обчислення 2) у задачах T1, T2, T3
        monitor.wait_calc();

        a4 = monitor.get_a(); // КД2
        int d4 = monitor.get_d(); // КД3
        int[][] MX4 = Data.fixMatrixLength(Data.MX, start, end);
        // Обчислення 3)
        int[] F4 = Data.addArrays(Data.mulScalArray(d4,
Data.fixArrayLength(Data.B, start, end)),
                Data.mulArrayMatrix(Data.fixArrayLength(Data.Z,
start, end), Data.mulMatrix(Data.MM, MX4)));
        // Обчислення 4)
        Data.Lh4 = Data.Sort(F4);
        // Сигнал задачі T3 про обчислення 4)
        monitor.signal_sort2();
        // Сигнал про обчислення 6) у T1
        monitor.wait_sort4();
        // Обчислення 7) XH = LH * a
        Data.insertArrayRes(Data.mulScalArray(a4, Data.X), Data.X,
start, end);
        // Чекати сигнали T1, T2, T3 про обчислення 7)
        monitor.wait_finish();
        // Виведення результату X
        if (Data.N == 4) {
            Data.printvec(Data.X);
        }

        System.out.println("Thread 4 has completed.");
    }
}
```

Результат при *N = 4 та 4 ядрах*:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-jav
Enter the size of the matrices:4
Start Thread 1
Start Thread 4
Start Thread 3
Start Thread 2
Thread 1 has completed.
Thread 2 has completed.
Thread 3 has completed.
5 5 5 5
Thread 4 has completed.
Execution time in milliseconds:13

Process finished with exit code 0
```

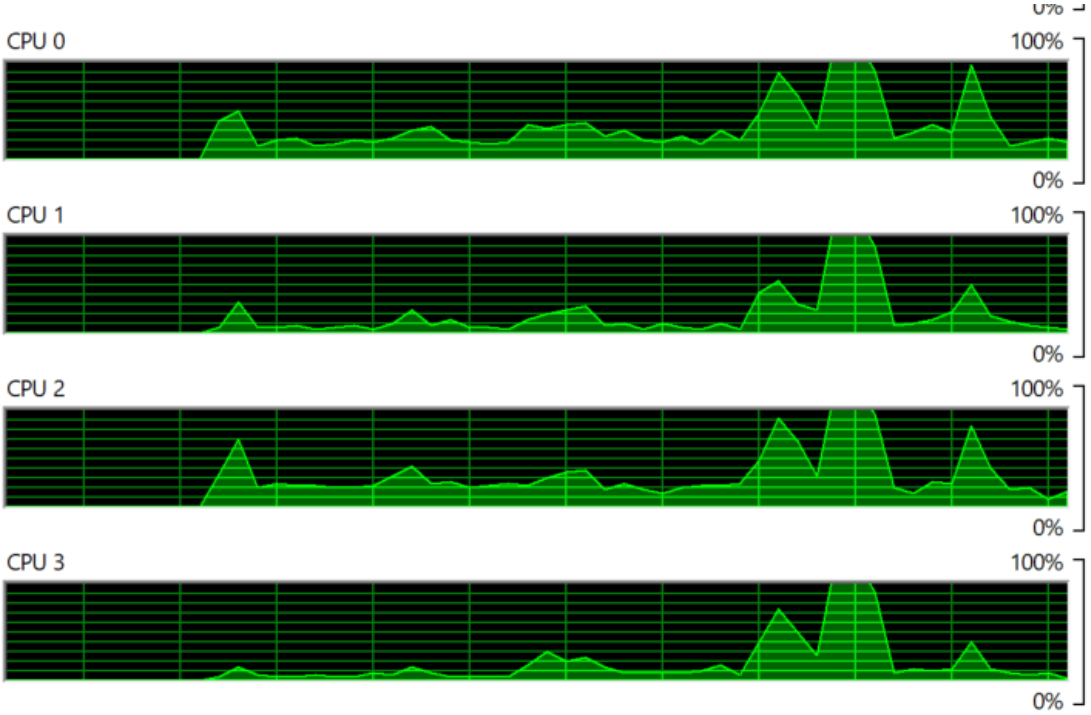
Підтвердження правильності розрахунків програми:

$$X = sort\left(1 * \begin{bmatrix}1\\1\\1\\1\end{bmatrix} + \begin{bmatrix}1\\1\\1\\1\end{bmatrix} * \left(\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix} * \begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix}\right)\right) *$$

$$\min\left(\begin{bmatrix}1\\1\\1\\1\end{bmatrix}\right) = \begin{bmatrix}5\\5\\5\\5\end{bmatrix}$$

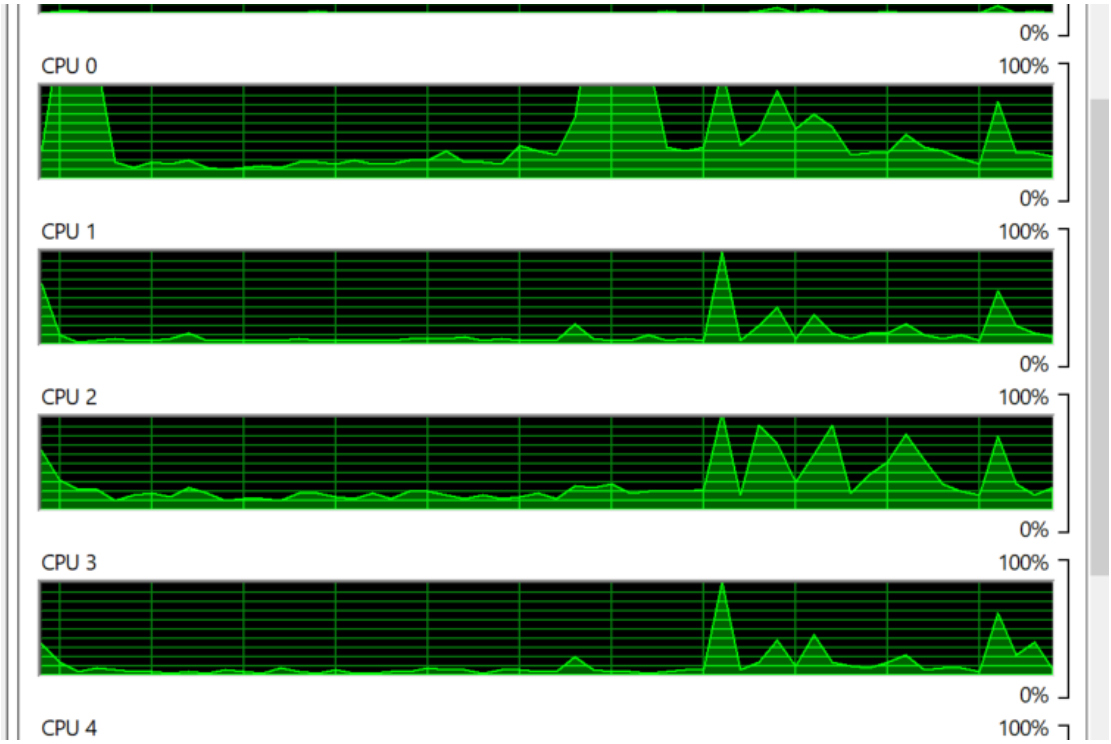
Результат при *N = 1000 та 4 ядрах*:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-
Enter the size of the matrices:1000
Start Thread 1
Start Thread 2
Start Thread 3
Start Thread 4
Thread 1 has completed.
Thread 2 has completed.
Thread 4 has completed.
Thread 3 has completed.
Execution time in milliseconds:1313

Process finished with exit code 0
```



Результат при *N = 1000 та 1 ядрі*:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent
Enter the size of the matrices:1000
Start Thread 1
Start Thread 2
Start Thread 3
Start Thread 4
Thread 2 has completed.
Thread 3 has completed.
Thread 4 has completed.
Thread 1 has completed.
Execution time in milliseconds:3846

Process finished with exit code 0
```



Коефіцієнт прискорення: 3846 / 1313 = 2.929

Висновок

Під час виконання лабораторної роботи я:

- Розробив паралельний алгоритм для математичної задачі згідно із заданим варіантом і визначив спільні змінні.
- Описав алгоритм роботи кожного потоку, виділив критичні секції та точки синхронізації.
- Створив структурну схему монітора, що забезпечує контроль взаємодії між задачами.
- Написав програму на Java, яка реалізує паралельне обчислення задачі за розробленим алгоритмом. Перевірив коректність результатів на тестовому масиві розміром 4.
- Провів тестування швидкодії програми на 1 і 4 ядрах для масивів розміром 1000, розрахувавши коефіцієнт прискорення, який становить 3.29.