НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №2**

з дисципліни **«**Паралельні та розподілені обчислення**»**

Виконав:

студент 3 курсу гр. ІО-42

Кочетов Данило

№ ЗК 4213

Перевірив:

Долголенко О. М.

Київ 2016 р.

***Завдання:***

1.13; 2.13; 3.13

F1: C = A\*(MA\*ME) + B + D

F2: ML = MIN(MF)\*MG + MAX(MH) \* (MK\*MF)

F3: T = (MO\*MP)\*S + MR\*SORT(S)

***Лістинг програми:***

***// Lab2.java***

*public class Lab2 extends Thread {*

*public final int N = 1000;*

*public static void main(String[] args) {*

*(new Lab2()).start();*

*}*

*@Override*

*public void run() {*

*setName("Lab 2");*

*System.out.println("Lab 2 start\n");*

*F1 f1 = new F1("F1", Thread.MIN\_PRIORITY, N);*

*F2 f2 = new F2("F2", Thread.NORM\_PRIORITY, N);*

*F3 f3 = new F3("F3", Thread.MAX\_PRIORITY, N);*

*f1.start();*

*f2.start();*

*f3.start();*

*try {*

*f1.join();*

*f2.join();*

*f3.join();*

*//System.out.println(f1.getResult());*

*//System.out.println(f2.getResult());*

*//System.out.println(f3.getResult());*

*} catch (InterruptedException e) {*

*e.printStackTrace();*

*}*

*System.out.println("\nLab 2 end");*

*}*

*}*

***// F1.java***

*public class F1 extends Thread {*

*private Vector result;*

*public Vector getResult() {*

*return result;*

*}*

*private int N;*

*F1(String name, int priority, int N) {*

*setName(name);*

*setPriority(priority);*

*this.N = N;*

*}*

*@Override*

*public void run() {*

*try {*

*sleep(500);*

*System.out.println("Task 1 start");*

*Vector A = new Vector(N), B = new Vector(N), D = new Vector(N);*

*Matrix MA = new Matrix(N), ME = new Matrix(N);*

*result = MA.multiply(ME).multiply(A).sum(B).sum(D);*

*System.out.println("Task 1 end");*

*} catch (InterruptedException e) {*

*e.printStackTrace();*

*}*

*}*

*}*

***// F2.java***

*public class F2 extends Thread {*

*private Matrix result;*

*public Matrix getResult() {*

*return result;*

*}*

*private int N;*

*F2(String name, int priority, int N) {*

*setName(name);*

*setPriority(priority);*

*this.N = N;*

*}*

*@Override*

*public void run() {*

*try {*

*sleep(250);*

*System.out.println("Task 2 start");*

*Matrix MF = new Matrix(N), MG = new Matrix(N), MH = new Matrix(N), MK = new Matrix(N);*

*result = MG.multiply(MF.min()).sum(MK.multiply(MF).multiply(MH.max()));*

*System.out.println("Task 2 end");*

*} catch (InterruptedException e) {*

*e.printStackTrace();*

*}*

*}*

*}*

***// F3.java***

*public class F3 extends Thread {*

*private Vector result;*

*public Vector getResult() {*

*return result;*

*}*

*private int N;*

*F3(String name, int priority, int N) {*

*setName(name);*

*setPriority(priority);*

*this.N = N;*

*}*

*@Override*

*public void run() {*

*try {*

*sleep(100);*

*System.out.println("Task 3 start");*

*Vector S = new Vector(N);*

*Matrix MO = new Matrix(N), MP = new Matrix(N), MR = new Matrix(N);*

*result = MO.multiply(MP).multiply(S).sum(MR.multiply(S.sort()));*

*System.out.println("Task 3 end");*

*} catch (InterruptedException e) {*

*e.printStackTrace();*

*}*

*}*

*}*

***// Vector.java***

*import java.util.Random;*

*public class Vector {*

*private long[] grid;*

*Vector(int N) {*

*grid = new long[N];*

*Random r = new Random();*

*for (int i = 0; i < N; ++i)*

*grid[i] = r.nextInt(20);*

*}*

*Vector(long[] grid) {*

*this.grid = grid;*

*}*

*public int getSize() {*

*return grid.length;*

*}*

*public long get(int i) {*

*return grid[i];*

*}*

*public Vector sum(Vector v) {*

*int N = getSize();*

*long[] newGrid = new long[N];*

*for (int i = 0; i < N; ++i)*

*newGrid[i] = grid[i] + v.get(i);*

*return new Vector(newGrid);*

*}*

*public Vector sort() {*

*int N = getSize();*

*long[] newGrid = grid.clone();*

*for (int i = 0; i < N; ++i) {*

*for (int k = 0; k < N - i - 1; ++k) {*

*if (newGrid[k] > newGrid[k + 1]) {*

*long t = newGrid[k];*

*newGrid[k] = newGrid[k + 1];*

*newGrid[k + 1] = t;*

*}*

*}*

*}*

*return new Vector(newGrid);*

*}*

*@Override*

*public String toString() {*

*String res = "";*

*int N = getSize();*

*for (int i = 0; i < N; ++i)*

*res += grid[i] + " ";*

*return res;*

*}*

*}*

***// Matrix.java***

*import java.util.Random;*

*public class Matrix {*

*Matrix(int N) {*

*Random r = new Random();*

*grid = new long[N][N];*

*for (int i = 0; i < N; ++i)*

*for (int k = 0; k < N; ++k)*

*grid[i][k] = r.nextInt(20);*

*}*

*Matrix(long[][] grid) {*

*this.grid = grid.clone();*

*}*

*public long get(int i, int k) {*

*return grid[i][k];*

*}*

*private long[][] grid;*

*public int getSize() {*

*return grid[0].length;*

*}*

*public Matrix multiply(Matrix m) {*

*int N = getSize();*

*long[][] newGrid = new long[N][N];*

*for (int i = 0; i < N; ++i) {*

*for (int k = 0; k < N; ++k) {*

*newGrid[i][k] = 0;*

*for (int j = 0; j < N; ++j) {*

*newGrid[i][k] += grid[i][j] \* m.get(j, k);*

*}*

*}*

*}*

*return new Matrix(newGrid);*

*}*

*public Vector multiply(Vector v) {*

*int N = getSize();*

*long[] newGrid = new long[N];*

*for (int i = 0; i < N; ++i) {*

*newGrid[i] = 0;*

*for (int k = 0; k < N; ++k) {*

*newGrid[i] += v.get(k) \* grid[i][k];*

*}*

*}*

*return new Vector(newGrid);*

*}*

*public Matrix multiply(long a) {*

*int N = getSize();*

*long[][] newGrid = new long[N][N];*

*for (int i = 0; i < N; ++i) {*

*for (int k = 0; k <N; ++k) {*

*newGrid[i][k] = grid[i][k] \* a;*

*}*

*}*

*return new Matrix(newGrid);*

*}*

*public Matrix sum(Matrix m) {*

*int N = getSize();*

*long[][] newGrid = new long[N][N];*

*for (int i = 0; i < N; ++i) {*

*for (int k = 0; k < N; ++k) {*

*newGrid[i][k] = grid[i][k] + m.get(i, k);*

*}*

*}*

*return new Matrix(newGrid);*

*}*

*public long min() {*

*long res = grid[0][0];*

*int N = getSize();*

*for (int i = 0; i < N; ++i) {*

*for (int k = 0; k < N; ++k) {*

*if (res < grid[i][k])*

*res = grid[i][k];*

*}*

*}*

*return res;*

*}*

*public long max() {*

*long res = grid[0][0];*

*int N = getSize();*

*for (int i = 0; i < N; ++i) {*

*for (int k = 0; k < N; ++k) {*

*if (res > grid[i][k])*

*res = grid[i][k];*

*}*

*}*

*return res;*

*}*

*@Override*

*public String toString() {*

*String res = "";*

*int N = getSize();*

*for (int i = 0; i < N; ++i) {*

*for (int k = 0; k < N; ++k) {*

*res += grid[i][k] + "\t";*

*}*

*res += "\n";*

*}*

*return res;*

*}*

*}*