Kpi-best

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І.СІКОРСЬКОГО»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №2**

з дисципліни «Паралельні та розподілені обчислення»

на тему: «ПОТОКИ В МОВІ JAVA»

Виконав:

студент 3-го курсу

факультету ІОТ

групи ІП-53

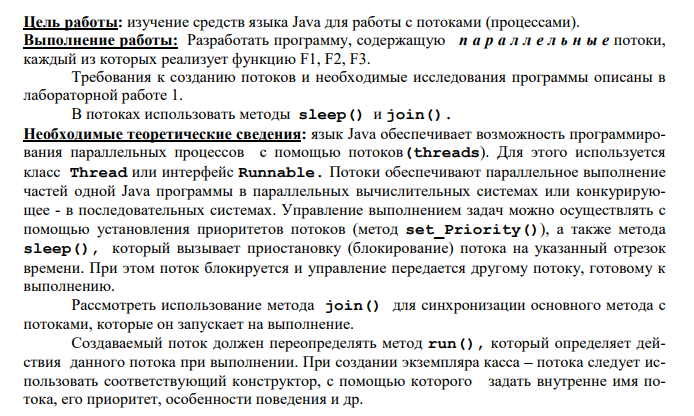
Сулима О.

Перевірив:

доц.

Долголенко О. М.

Київ 2017



**Варіант 4**

1.18 d = (A\*B) + (C\*(B\*(MA\*MD)).

2.18 h = MIN(MG\*ML) .

3.18 s = MAX(SORT(MS) + MA\*MB) .

**Лістинг:**

**FirstTask.java**

public class FirstTask extends Thread {

private int[] vectorA = new int[Work.N];

private int[] vectorB = new int[Work.N];

private int[] vectorC = new int[Work.N];

private int[][] matrixA = new int[Work.N][Work.N];

private int[][] matrixD = new int[Work.N][Work.N];

private int[] resultVector = new int[Work.N];

@Override

public void run() {

System.out.println("Start 1 thread...");

inputData();

int[] firstUnitVector = multVectors(vectorA, vectorB);

int[] secondUnitVector = multVectors(vectorC, multIntegerWithVector(vectorB, multMatrix(matrixA, matrixD)));

resultVector = addVectors(firstUnitVector, secondUnitVector);

displayResult(resultVector);

System.out.println("\nFinish 1 thread...\n");

}

private void displayResult(int[] resultVector) {

for (int aResultVector : resultVector) {

System.out.print(aResultVector + " ");

}

}

private int[] addVectors(int[] firstUnitVector, int[] secondUnitVector) {

int[] tmp = new int[firstUnitVector.length];

for (int i = 0; i < firstUnitVector.length; i++) {

tmp[i] = firstUnitVector[i] + secondUnitVector[i];

}

return tmp;

}

private void inputData() {

randomVector(vectorA);

randomVector(vectorB);

randomVector(vectorC);

randomMatrix(matrixA);

randomMatrix(matrixD);

}

private void randomVector(int[] vector) {

for (int i = 0; i < vector.length; i++) {

vector[i] = (int) (Math.random() \* 10) + 1;

}

}

private void randomMatrix(int[][] matrix) {

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix.length; j++) {

matrix[i][j] = (int) (Math.random() \* 10) + 1;

}

}

}

private int[] multVectors(int[] firstVector, int[] secondVector) {

int[] resultSum = new int[firstVector.length];

for (int i = 0; i < firstVector.length; i++) {

resultSum[i] = firstVector[i] \* secondVector[i];

}

return resultSum;

}

private int multMatrix(int[][] firstMatrix, int[][] secondMatrix) {

int resultSum = 1;

int tmpMult;

for (int i = 0; i < firstMatrix.length; i++) {

for (int j = 0; j < firstMatrix.length; j++) {

tmpMult = firstMatrix[i][j] \* secondMatrix[i][j];

resultSum += tmpMult;

}

}

return resultSum;

}

private int multInteger(int first, int second) {

return first \* second;

}

private int[] multIntegerWithVector(int[] vector, int value) {

for (int i = 0; i < vector.length; i++) {

vector[i] = vector[i] \* value;

}

return vector;

}

}

**SecondTask.java**

public class SecondTask extends Thread {

private int[][] matrixG = new int[Work.N][Work.N];

private int[][] matrixL = new int[Work.N][Work.N];

@Override

public void run() {

System.out.println("Start 2 thread...");

randomMatrix(matrixG);

randomMatrix(matrixL);

int result = searchMin(multMatrix(matrixG, matrixL));

System.out.println(result);

System.out.println("Finish 2 thread...\n");

}

private void randomMatrix(int[][] matrix) {

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix.length; j++) {

matrix[i][j] = (int) (Math.random() \* 100) + 1;

}

}

}

private int[][] multMatrix(int[][] firstMatrix, int[][] secondMatrix) {

int[][] resultMatrix = new int[Work.N][Work.N];

int tmpMult;

for (int i = 0; i < firstMatrix.length; i++) {

for (int j = 0; j < firstMatrix.length; j++) {

tmpMult = firstMatrix[i][j] \* secondMatrix[i][j];

resultMatrix[i][j] = tmpMult;

}

}

return resultMatrix;

}

private int searchMin(int[][] matrix) {

int value = matrix[0][0];

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix.length; j++) {

if (matrix[i][j] < value)

value = matrix[i][j];

}

}

return value;

}

}

**ThirdTask.java**

package lab2;

import java.security.SecureRandom;

import java.util.Arrays;

public class ThirdTask extends Thread {

private int[][] matrixS = new int[Work.N][Work.N];

private int[][] matrixA = new int[Work.N][Work.N];

private int[][] matrixB = new int[Work.N][Work.N];

@Override

public void run() {

System.out.println("Start 3 thread...");

inputData();

int[][] multTemp = multMatrix(matrixA, matrixA);

int[][] sortTemp = sortMatrix(matrixS);

int[][] tempMatrix = addingMatrix(sortTemp, multTemp);

int result = searchMax(tempMatrix);

System.out.println(result);

System.out.println("Finish 3 thread...");

}

private void inputData() {

randomMatrix(matrixS);

randomMatrix(matrixA);

randomMatrix(matrixB);

}

private void randomMatrix(int[][] matrix) {

SecureRandom secureRandom = new SecureRandom();

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix.length; j++) {

//matrix[i][j] = (int) (Math.random() \* 10) + 1;

matrix[i][j] = secureRandom.nextInt();

}

}

}

private int[][] sortMatrix(int[][] matrix) {

int[] flat = new int[Work.N \* Work.N];

int ctr = 0;

for (int row = 0; row < matrix.length; row++) {

for (int col = 0; col < matrix.length; col++) {

flat[ctr++] = matrix[row][col];

}

}

Arrays.sort(flat);

ctr = 0;

for (int row = 0; row < 2; row++) {

for (int col = 0; col < 4; col++) {

matrix[row][col] = flat[ctr++];

}

}

return matrix;

}

private int[][] multMatrix(int[][] firstMatrix, int[][] secondMatrix) {

int[][] resultMatrix = new int[Work.N][Work.N];

int tmpMult;

for (int i = 0; i < firstMatrix.length; i++) {

for (int j = 0; j < firstMatrix.length; j++) {

tmpMult = firstMatrix[i][j] \* secondMatrix[i][j];

resultMatrix[i][j] = tmpMult;

}

}

return resultMatrix;

}

private int[][] addingMatrix(int[][] firstMatrix, int[][] secondMatrix) {

int[][] resultMatrix = new int[Work.N][Work.N];

for (int i = 0; i < firstMatrix.length; i++) {

for (int j = 0; j < firstMatrix.length; j++) {

resultMatrix[i][j] = firstMatrix[i][j] + secondMatrix[i][j];

}

}

return resultMatrix;

}

private int searchMax(int[][] matrix) {

int max = 0;

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix.length; j++) {

if (matrix[i][j] > max)

max = matrix[i][j];

}

}

return max;

}

}

**Результати виконання програми:**

