

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І.СІКОРСЬКОГО»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №3**

з дисципліни «Паралельні та розподілені обчислення»

на тему: «ПОТОКИ В МОВІ C#»

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:**

**Methods.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace LabPP3

{

class Methods

{

static object block = new object();

private int sizeOfArrays;

public Methods(int sizeOfArrays)

{

this.sizeOfArrays = sizeOfArrays;

}

public int[] FillingOfVector()

{

int[] vector = new int[sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

vector[i] = 1;

}

return vector;

}

public int[,] FillingOfMatrix()

{

int[,] matrix = new int[sizeOfArrays, sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

for (int j = 0; j < sizeOfArrays; j++)

{

matrix[i, j] = 1;

}

}

return matrix;

}

public void PutVector(int[] vector)

{

lock (block)

{

for (int i = 0; i < sizeOfArrays; i++)

{

Console.Write(vector[i] + " ");

}

Console.WriteLine();

}

}

public void PutMatrix(int[,] matrix)

{

lock (block)

{

for (int i = 0; i < sizeOfArrays; i++)

{

for (int j = 0; j < sizeOfArrays; j++)

{

Console.Write(matrix[i, j] + " ");

}

Console.WriteLine();

}

Console.WriteLine();

}

}

public int[] SumOfVector(int[] vector1, int[] vector2)

{

int[] resVector = new int[sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

resVector[i] = vector1[i] + vector2[i];

}

return resVector;

}

public int[] MultOfVector(int[] vector1, int[] vector2)

{

int[] resVector = new int[sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

resVector[i] = vector1[i] \* vector2[i];

}

return resVector;

}

public int[,] SumOfMatrix(int[,] matrix1, int[,] matrix2)

{

int[,] resMatrix = new int[sizeOfArrays, sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

for (int j = 0; j < sizeOfArrays; j++)

{

resMatrix[i, j] = matrix1[i, j] + matrix2[i, j];

}

}

return resMatrix;

}

public int[,] VectorToMatrixOnRow(int[] vector)

{

int[,] matrix = new int[sizeOfArrays, sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

matrix[0, i] = vector[i];

}

return matrix;

}

public int[,] VectorToMatrixOnColumn(int[] vector)

{

int[,] matrix = new int[sizeOfArrays, sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

matrix[i, 0] = vector[i];

}

return matrix;

}

public int[] MatrixToVector(int[,] matrix)

{

int[] vector = new int[sizeOfArrays];

int indexOfVector = 0;

for (int i = 0; i < sizeOfArrays; i++)

{

for (int j = 0; j < sizeOfArrays; j++)

{

if (matrix[i, j] != 0) vector[indexOfVector++] = matrix[i, j];

}

}

return vector;

}

public int[,] MultMatrix(int[,] matrix1, int[,] matrix2)

{

int[,] resultMatrix = new int[sizeOfArrays, sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

for (int j = 0; j < sizeOfArrays; j++)

{

for (int k = 0; k < sizeOfArrays; k++)

{

resultMatrix[i, j] += matrix1[i, k] \* matrix2[k, j];

}

}

}

return resultMatrix;

}

public int[,] TransMatrix(int[,] matrix)

{

int[, ] transMatr = new int[sizeOfArrays, sizeOfArrays];

for (int i = 0; i < sizeOfArrays; i++)

{

for (int j = 0; j < sizeOfArrays; j++)

{

transMatr[i, j] = matrix[j, i];

}

}

return transMatr;

}

public int MinOfMatrix(int[,] matrix)

{

int res = 9999;

for (int i = 0; i < sizeOfArrays; i++)

{

for (int j = 0; j < sizeOfArrays; j++)

{

if (matrix[i, j] < res)

{

res = matrix[i, j];

}

}

}

return res;

}

}

}

**Main.cs**

using System;

using System.Threading;

using LabPP3;

namespace ConsoleApplication

{

public class Program

{

static void Main(string[] args)

{

Function1 instance1 = new Function1();

Function2 instance2 = new Function2();

Function3 instance3 = new Function3();

ParameterizedThreadStart function1 = new ParameterizedThreadStart(instance1.StartOfFunction);

ParameterizedThreadStart function2 = new ParameterizedThreadStart(instance2.StartOfFunction);

ParameterizedThreadStart function3 = new ParameterizedThreadStart(instance3.StartOfFunction);

Thread thread1 = new Thread(function1);

Thread thread2 = new Thread(function2);

Thread thread3 = new Thread(function3);

thread1.Start(4);

thread2.Start(5);

thread3.Start(6);

}

}

}