Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №4**

З дисципліни «ПРО»

Виконав: Перевірив:

Студент групи ІО-21 доц. Корочкін О.В.

Коноз А.О.

Дата здачі\_\_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом\_\_\_\_\_

Київ 2014

**Завдання**

1.23 2.13 3.17

1. F1: E:=A+B+C+D(MA\*MZ).

2. F2: MZ:=Min(MA)\*MB+Max(MT)\*(MX\*MZ).

3. F3: d:=Min(A\*Trans(MB\*MM)+B\*Sort(MC).

**Лістинг коду**

using System;

using System.Threading;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace PROLab1

{

/\*\*

\* Виконав: Коноз Андрій

\* Група: ІО-21

\* Дата: 04.10.2014

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

\* F2: MZ:=Min(MA)\*MB+Max(MT)\*(MX\*MZ)

\* F3: d:=Min(A\*Trans(MB\*MM)+B\*Sort(MC))

\*/

class Lab4

{

private static int n;

static void Main(string[] args)

{

Console.WriteLine("Main thread started");

Console.WriteLine("Enter n: ");

n = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("n = " + n);

Thread tf1 = new Thread(MF1);

Thread tf2 = new Thread(MF2);

Thread tf3 = new Thread(MF3);

tf1.Name = "TF1";

tf1.Priority = ThreadPriority.Highest;

tf2.Name = "TF2";

tf2.Priority = ThreadPriority.BelowNormal;

tf3.Name = "TF3";

tf3.Priority = ThreadPriority.Lowest;

tf1.Start();

tf2.Start();

tf3.Start();

tf1.Join();

tf2.Join();

tf3.Join();

Console.WriteLine("Main thread finished");

}

/\*\*

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

\*/

public static void MF1()

{

double[] vectA;

double[] vectB;

double[] vectC;

double[] vectD;

double[,] matrA;

double[,] matrZ;

double[] vectRes;

Console.WriteLine(Thread.CurrentThread.Name + " started");

vectA = Tools.readVect(n, "Enter vector A for F1");

vectB = Tools.readVect(n, "Enter vector B for F1");

vectC = Tools.readVect(n, "Enter vector C for F1");

vectD = Tools.readVect(n, "Enter vector D for F1");

matrA = Tools.readMatr(n, "Enter matrix A for F1");

matrZ = Tools.readMatr(n, "Enter matrix Z for F1");

Thread.Sleep(1000);

vectRes = Tools.addVect(Tools.addVect(vectA, vectB), Tools.addVect(vectC, Tools.multVectMatr(vectD, Tools.multMatr(matrA, matrZ))));

Tools.showVect(vectRes, "Result of F1");

Console.WriteLine(Thread.CurrentThread.Name + " finished");

}

/\*

\* F2: MZ:=Min(MA)\*MB+Max(MT)\*(MX\*MZ)

\*/

public static void MF2()

{

double[,] matrA;

double[,] matrB;

double[,] matrM;

double[,] matrX;

double[,] matrZ;

double[,] matrRes;

Console.WriteLine(Thread.CurrentThread.Name + " started");

matrA = Tools.readMatr(n, "Enter matrix A for F2");

matrB = Tools.readMatr(n, "Enter matrix B for F2");

matrM = Tools.readMatr(n, "Enter matrix M for F2");

matrX = Tools.readMatr(n, "Enter matrix X for F2");

matrZ = Tools.readMatr(n, "Enter matrix Z for F2");

Thread.Sleep(500);

matrRes = Tools.addMatr(Tools.multNumbMatr(Tools.findMinMatr(matrA), matrB), Tools.multNumbMatr(Tools.findMaxMatr(matrM), Tools.multMatr(matrX, matrZ)));

Tools.showMatr(matrRes, "Result of F2");

Console.WriteLine(Thread.CurrentThread.Name + " finished");

}

/\*

\* F3: d:=Min(A\*Trans(MB\*MM)+B\*Sort(MC))

\*/

public static void MF3()

{

double[] vectA;

double[] vectB;

double[,] matrB;

double[,] matrM;

double[,] matrC;

double result;

Console.WriteLine(Thread.CurrentThread.Name + " started");

vectA = Tools.readVect(n, "Enter vector A for F3");

vectB = Tools.readVect(n, "Enter vector B for F3");

matrB = Tools.readMatr(n, "Enter matrix B for F3");

matrM = Tools.readMatr(n, "Enter matrix M for F3");

matrC = Tools.readMatr(n, "Enter matrix C for F3");

Thread.Sleep(100);

result = Tools.findMinVect(Tools.addVect(Tools.multVectMatr(vectA, Tools.transpMatr(Tools.multMatr(matrB, matrM))), Tools.multVectMatr(vectB, Tools.sortMatr(matrC))));

Console.WriteLine("Result of F3: " + result);

Console.WriteLine(Thread.CurrentThread.Name + " finished");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace PROLab1

{

class Tools

{

/\*\*

\* Додає два вектори

\* @param vectA вектор

\* @param vectB вектор

\* @return вектор суми

\*/

public static double[] addVect(double[] vectA, double[] vectB)

{

for (int i = 0; i < vectB.GetLength(0); i++)

{

vectA[i] += vectB[i];

}

return vectA;

}

/\*\*

\* Перемножає дві матриці

\* @param matrA матриця

\* @param matrB матриця

\* @return повертає добуток двох матриць

\*/

public static double[,] multMatr(double[,] matrA, double[,] matrB)

{

double temp = 0;

double[,] resultMatr = new double[matrA.GetLength(0), matrA.GetLength(1)];

for (int i = 0; i < matrA.GetLength(0); i++)

{

for (int j = 0; j < matrA.GetLength(1); j++)

{

temp = 0;

for (int k = 0; k < matrB.GetLength(0); k++)

{

temp += matrA[i, k] \* matrB[k, j];

}

resultMatr[i, j] = temp;

}

}

return resultMatr;

}

/\*\*

\* Транспонує матрицю

\* @param matrA матриця для транспонування

\* @return транспоновану матрицю

\*/

public static double[,] transpMatr(double[,] matrA)

{

double[,] resultMatr = new double[matrA.GetLength(1), matrA.GetLength(0)];

for (int i = 0; i < resultMatr.GetLength(0); i++)

{

for (int j = 0; j < resultMatr.GetLength(1); j++)

{

resultMatr[i, j] = matrA[j, i];

}

}

return resultMatr;

}

/\*\*

\* Знаходить максимальний елемент в матриці

\* @param matrA матриця для пошуку

\* @return максимальний елемент матриці

\*/

public static double findMaxMatr(double[,] matrA)

{

double max = matrA[0, 0];

for (int i = 0; i < matrA.GetLength(0); i++)

{

for (int j = 0; j < matrA.GetLength(1); j++)

{

if (max < matrA[i, j])

{

max = matrA[i, j];

}

}

}

return max;

}

/\*\*

\* Знаходить мінімальний елемент матриці

\* @param matrA матриця для пошуку

\* @return мінімальний елемент

\*/

public static double findMinMatr(double[,] matrA)

{

double min = matrA[0, 0];

for (int i = 0; i < matrA.GetLength(0); i++)

{

for (int j = 0; j < matrA.GetLength(1); j++)

{

if (min > matrA[i, j])

{

min = matrA[i, j];

}

}

}

return min;

}

/\*\*

\* Знаходить мінімальний елемент в векторі

\* @param vectA вектор для пошуку

\* @return мінімальний елемент

\*/

public static double findMinVect(double[] vectA)

{

double min = vectA[0];

for (int i = 0; i < vectA.GetLength(0); i++)

{

if (min > vectA[i])

{

min = vectA[i];

}

}

return min;

}

/\*\*

\* Додає дві матриці

\* @param matrA матриця

\* @param matrB матриця

\* @return суму матриць

\*/

public static double[,] addMatr(double[,] matrA, double[,] matrB)

{

for (int i = 0; i < matrB.GetLength(0); i++)

{

for (int j = 0; j < matrB.GetLength(1); j++)

{

matrA[i, j] += matrB[i, j];

}

}

return matrA;

}

/\*\*

\* Перемножає вектор на матрицю

\* @param vectA вектор

\* @param matrB матрицю

\* @return добуток вектора і матриці

\*/

public static double[] multVectMatr(double[] vectA, double[,] matrB)

{

double temp = 0;

double[] resultVect = new double[vectA.GetLength(0)];

for (int i = 0; i < matrB.GetLength(0); i++)

{

temp = 0;

for (int j = 0; j < resultVect.GetLength(0); j++)

{

temp += vectA[j] \* matrB[j, i];

}

resultVect[i] = temp;

}

return resultVect;

}

/\*\*

\* Сортує вектор за зростанням

\* @param vectA вектор для сортування

\* @return відсортований вектор

\*/

public static double[] sortVect(double[] vectA)

{

double temp = 0;

for (int i = 0; i < vectA.GetLength(0); i++)

{

for (int j = 0; j < (i - 1); j++)

{

if (vectA[j] > vectA[j + 1])

{

temp = vectA[j];

vectA[j] = vectA[j + 1];

vectA[j + 1] = temp;

}

}

}

return vectA;

}

/\*\*

\* Сортує матрицю за спаданням

\* @param matrA матриця для сортування

\* @return відсортовану матрицю

\*/

public static double[,] sortMatr(double[,] matrA)

{

double temp = 0;

for (int k = 0; k < matrA.GetLength(0); k++)

{

for (int i = 0; i < matrA.GetLength(0); i++)

{

for (int j = 0; j < (i - 1); j++)

{

if (matrA[k, j] < matrA[k, j + 1])

{

temp = matrA[k, j];

matrA[k, j] = matrA[k, j + 1];

matrA[k, j + 1] = temp;

}

}

}

}

return matrA;

}

/\*\*

\* Множить матрицю на число

\* @param a число

\* @param matrB матриця

\* @return результат множення

\*/

public static double[,] multNumbMatr(double a, double[,] matrB)

{

for (int i = 0; i < matrB.GetLength(0); i++)

{

for (int j = 0; j < matrB.GetLength(1); j++)

{

matrB[i, j] = matrB[i, j] \* a;

}

}

return matrB;

}

/\*\*

\* Зчитує матрицю

\* @param n кількість елементів

\* @param message повідомлення для відображення

\* @return зчитану матрицю

\*/

public static double[,] readMatr(int n, String message)

{

double[,] matr = new double[n, n];

Console.WriteLine(message);

for (int i = 0; i < matr.GetLength(0); i++)

{

for (int j = 0; j < matr.GetLength(1); j++)

{

matr[i, j] = 1;

}

}

return matr;

}

/\*\*

\* Відображає матрицю

\* @param matr матриця для відображення

\* @param message повідомлення для відображення

\*/

public static void showMatr(double[,] matr, String message)

{

if (matr.GetLength(0) < 10)

{

Console.WriteLine(message);

for (int i = 0; i < matr.GetLength(0); i++)

{

for (int j = 0; j < matr.GetLength(1); j++)

{

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

}

Console.WriteLine();

}

Console.WriteLine();

}

}

/\*\*

\* Зчитує вектор

\* @param n кількість елементів вектора

\* @param message повідомлення для відображення

\* @return зчитаний вектор

\*/

public static double[] readVect(int n, String message)

{

double[] vect = new double[n];

Console.WriteLine(message);

for (int i = 0; i < vect.GetLength(0); i++)

{

vect[i] = 1;

}

return vect;

}

/\*\*

\* Відображає вектор

\* @param vect вектор для відображення

\* @param message повідомлення для відображення

\*/

public static void showVect(double[] vect, String message)

{

if (vect.GetLength(0) < 10)

{

Console.WriteLine(message);

for (int i = 0; i < vect.GetLength(0); i++)

{

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

}

Console.WriteLine();

}

}

}

}