НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №3**

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:***

***// Lab3.cs***

using System;

using System.Threading;

namespace lab3

{

class Lab3

{

private static readonly int N = 500;

static void Main(string[] args)

{

Console.WriteLine("Lab 3 start");

Console.WriteLine();

Thread f1 = new Thread((new F1(N)).run);

Thread f2 = new Thread((new F2(N)).run);

Thread f3 = new Thread((new F3(N)).run);

f1.Name = "F1";

f2.Name = "F2";

f3.Name = "F3";

f1.Priority = ThreadPriority.Lowest;

f2.Priority = ThreadPriority.Normal;

f3.Priority = ThreadPriority.Highest;

f1.Start();

f2.Start();

f3.Start();

f1.Join();

f2.Join();

f3.Join();

Console.WriteLine();

Console.WriteLine("Lab 3 end");

Console.Write("Press any key...");

Console.ReadKey();

}

}

}

***// F1.cs***

using System;

namespace lab3

{

public class F1

{

private Vector result;

public Vector getResult()

{

return result;

}

private int N;

public F1(int N)

{

this.N = N;

}

public void run()

{

Console.WriteLine("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);

Console.WriteLine("Task 1 end");

}

}

}

***// F2.cs***

using System;

namespace lab3

{

public class F2

{

private Matrix result;

public Matrix getResult()

{

return result;

}

private int N;

public F2(int N)

{

this.N = N;

}

public void run()

{

Console.WriteLine("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()));

Console.WriteLine("Task 2 end");

}

}

}

***// F3.cs***

using System;

namespace lab3

{

public class F3

{

private Vector result;

public Vector getResult()

{

return result;

}

private int N;

public F3(int N)

{

this.N = N;

}

public void run()

{

Console.WriteLine("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()));

Console.WriteLine("Task 3 end");

}

}

}

***// Vector.cs***

using System;

namespace lab3

{

public class Vector {

private long[] grid;

public Vector(int N) {

grid = new long[N];

Random r = new Random();

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

grid[i] = r.Next(20);

}

public 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 = (long[]) 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);

}

public String toString() {

String res = "";

int N = getSize();

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

res += grid[i] + " ";

return res;

}

}

}

***// Matrix.cs***

using System;

namespace lab3

{

public class Matrix

{

public 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.Next(20);

}

public Matrix(long[,] grid)

{

this.grid = (long[,]) grid.Clone();

}

public long get(int i, int k)

{

return grid[i, k];

}

private long[,] grid;

public int getSize()

{

return grid.GetLength(0);

}

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;

}

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;

}

}

}