НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №4**

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

Виконав:

студент 3 курсу гр. ІО-31

Долинний Олександр

№ ЗК 3110

Перевірив:

Корочкін О. В.

Київ 2015 р.

***Завдання:***

1.1 3.21 3.11

F1: A = sort(B) \* (MB \* MC)

F2: W = sort(R \* MT) \* (MX \* MS)

F3: R = sort(S + T) \* trans(MS \* MR)

***Лістинг програми:***

using System;

using System.Threading;

/\*

Parallel and distributed computing.

Lab 4: Threads in C#

Dolinniy Alexandr

IO-31

23.10.2015

Func1: A = sort(B)\*(MB\*MC)

Func2: W = sort(R\*MT)\*(MX\*MS)

Func3: R = sort(S+T)\*trans(MS\*MR)

\*/

namespace lab4

{

class MainClass

{

public static void Main (string[] args)

{

Console.WriteLine ("Lab 4 started");

int n = 4;

Data data = new Data(n);

ThreadFunctions t = new ThreadFunctions (data);

Thread t1 = new Thread(t.Task1, 1000);

Thread t2 = new Thread(t.Task2, 1000);

Thread t3 = new Thread(t.Task3, 1000);

t1.Name = "Thread 1";

t1.Priority = ThreadPriority.BelowNormal;

t2.Name = "Thread 2";

t2.Priority = ThreadPriority.AboveNormal;

t3.Name = "Thread 3";

t3.Priority = ThreadPriority.Highest;

t1.Start();

t2.Start();

t3.Start();

t1.Join ();

t2.Join ();

t3.Join ();

Console.WriteLine ("Lab 4 finished");

;

}

}

}

using System;

/\*

Parallel and distributed computing.

Lab 4: Threads in C#

Dolinniy Alexandr

IO-31

23.10.2015

Func1: A = sort(B)\*(MB\*MC)

Func2: W = sort(R\*MT)\*(MX\*MS)

Func3: R = sort(S+T)\*trans(MS\*MR)

\*/

namespace lab4

{

public class Data

{

private int n;

public int getN()

{

return n;

}

public Data(int n)

{

this.n = n;

}

public int[] vectorInput() {

int[] vector = new int[n];

for (int i = 0; i < n; i++) {

vector[i] = Console.Read();

}

return vector;

}

public void vectorFill(int value, int[] vector)

{

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

{

vector[i] = value;

}

return;

}

public void matrixFill(int value, int[,] matrix)

{

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

{

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

{

matrix[i, j] = value;

}

}

return;

}

public int[,] matrixInput() {

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

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

matrix[i,j] = Console.Read();

}

}

return matrix;

}

public void vectorOutput(int[] vector)

{

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

{

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

}

Console.WriteLine("");

}

public void matrixOutput(int[,] matrix)

{

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

{

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

{

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

}

Console.WriteLine();

}

}

public int[] func1(int[] b, int[,] mb, int[,] mc)

{

return vectorMatrixMult(b, matrixMult(mb, mc));

}

public int[] func2(int[] r, int[,] mt, int[,] mx, int[,] ms)

{

return vectorMatrixMult(vectorSort(vectorMatrixMult(r, mt)), matrixMult(mx, ms));

}

public int[] func3(int[] s, int[] t, int[,] ms, int[,] mr)

{

return vectorMatrixMult(vectorAdd(s, t), matrixTrans(matrixMult(ms, mr)));

}

private int[] vectorAdd(int[] a, int[] b)

{

int[] c = new int[n];

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

{

c[i] = a[i] + b[i];

}

return c;

}

private int[,] matrixTrans(int[,] ma)

{

int buf;

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

{

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

{

buf = ma[i,j];

ma[i,j] = ma[j,i];

ma[j,i] = buf;

}

}

return ma;

}

private int[] vectorMatrixMult(int[] a, int[,] ma)

{

int[] c = new int[n];

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

{

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

{

c[i] += a[j] \* ma[j,i];

}

}

return c;

}

private int[,] matrixMult(int[,] ma, int[,] mb) {

int[,] c = new int[n,n];

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

for (int k = 0; k < n; k++) {

c[i,j] += ma[i,k] \* mb[k,j];

}

}

}

return c;

}

private int[] vectorSort(int[] a)

{

int[] c = new int[n];

Array.Copy(a, c, n);

Array.Sort(c);

return c;

}

}

}

using System;

using System.Threading;

namespace lab4

{

public class ThreadFunctions

{

Data d;

public ThreadFunctions(Data d)

{

this.d = d;

}

public void Task1()

{

int n;

int e;

int[] a, b;

int[,] mb, mc;

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

n = d.getN ();

a = new int[n];

b = new int[n];

d.vectorFill(1, b);

mb = new int[n,n];

mc = new int[n,n];

d.matrixFill(1, mb);

d.matrixFill(1, mc);

Thread.Sleep(3000);

a = d.func1(b, mb, mc);

Console.WriteLine("F1:");

d.vectorOutput(a);

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

}

public void Task2()

{

int n;

int[] w, r;

int[,] mt, mx, ms;

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

n = d.getN();

r = new int[n];

d.vectorFill(1, r);

mt = new int[n,n];

d.matrixFill (1, mt);

ms = new int[n, n];

d.matrixFill(1, ms);

mx = new int[n,n];

d.matrixFill (1, mx);

Thread.Sleep(2000);

w = d.func2(r, mt, mx, ms);

Console.WriteLine("F2:");

d.vectorOutput(w);

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

}

public void Task3()

{

int n;

int[] r,s,t;

int[,] mr, ms;

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

n = d.getN();

s = new int[n];

d.vectorFill(1, s);

t = new int[n];

d.vectorFill(1, t);

mr = new int[n,n];

d.matrixFill (1, mr);

ms = new int[n,n];

d.matrixFill (1, ms);

Thread.Sleep(1000);

r = d.func3(s, t, ms, mr);

Console.WriteLine("F3:");

d.vectorOutput(r);

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

}

}

}