Kpi-best

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І.СІКОРСЬКОГО»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №1**

з дисципліни «Паралельні та розподілені обчислення»

на тему: «Процеси в мові Ada. Задачі»

Виконав:

студент 3-го курсу

факультету ІОТ

групи ІП-53

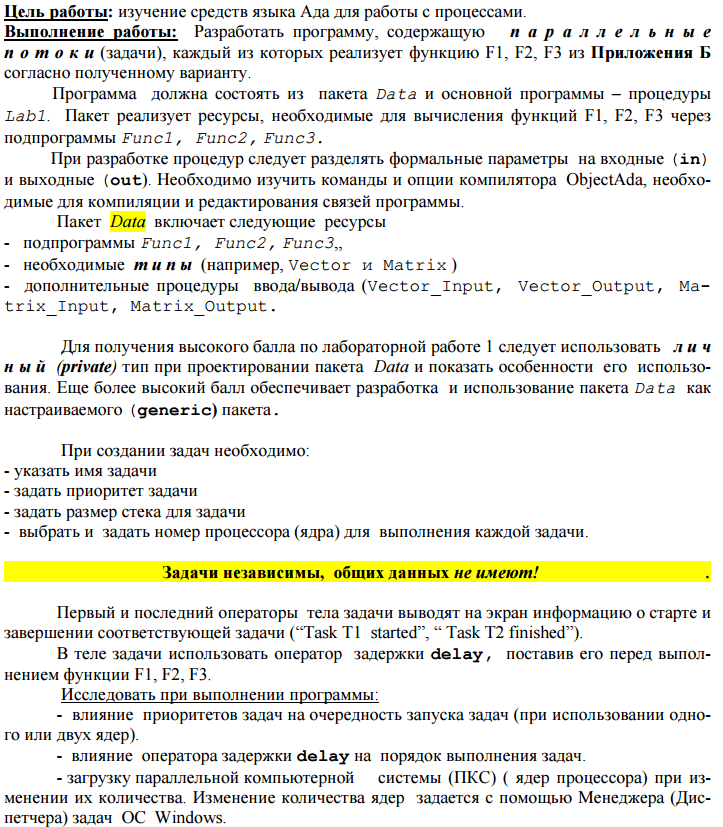
Сулима О.

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:**

**Lab1.adb**

with Data;

with Ada.Text\_IO; use Ada.Text\_IO;

with System.Multiprocessors; use System.Multiprocessors;

with Ada.Integer\_Text\_IO; use Ada.Integer\_Text\_IO;

procedure Lab1 is

N : Integer := 5;

package Lab1Data is new Data(N);

use Lab1Data;

vectorA, vectorB, resultVector, vectorT, vectorO, vectorP : Vector;

matrixA, matrixB, matrixP, matrixR, matrixS, matrixF, matrixG, matrixH, matrixK: Matrix;

integerRes2, integerRes3: Integer;

CPU0: CPU\_Range := 0;

CPU1: CPU\_Range := 1;

CPU2: CPU\_Range := 2;

procedure tasks is

task T1 is

pragma Priority(1);

pragma Storage\_Size(100000);

pragma CPU(CPU0);

end;

task body T1 is

vectorA: Vector;

matrixA, matrixE: Matrix;

begin

Put\_Line("T1 started");

randomVector(vectorA);

randomVector(vectorB);

randomVector(vectorT);

randomMatrix(matrixA);

randomMatrix(matrixE);

F1(vectorA, vectorB, vectorT, matrixA, matrixE, resultVector);

Put\_Line("T1 finished");

end T1;

task T2;

task body T2 is

matrixH, matrixK, matrixL, matrixF: Matrix;

begin

Put\_Line("T2 started");

fillMatrix(matrixH);

fillMatrix(matrixK);

fillMatrix(matrixL);

F2(matrixH, matrixK, integerRes2);

Put\_Line("T2 finished");

end T2;

task T3 is

pragma Priority(5);

pragma Storage\_Size(100000);

end;

task body T3 is

vectorP, vectorT: Vector;

matrixR, matrixS: Matrix;

integerRes: Integer;

begin

Put\_Line("T3 started");

randomVector(vectorP);

fillMatrix(matrixR);

fillMatrix(matrixS);

F3(matrixR, matrixS, vectorP, integerRes);

Put\_Line("T3 finished");

end T3;

begin

null;

end tasks;

begin

Put\_Line("Lab 1 started");

tasks;

Put\_Line("---");

Put\_Line("Task 1 results");

putVector(resultVector);

Put\_Line("---");

Put\_Line("---");

Put\_Line("Task 2 results");

Put(integerRes2);

Put\_Line("");

Put\_Line("---");

Put\_Line("---");

Put\_Line("Task 3 results");

Put(integerRes3);

Put\_Line("");

Put\_Line("---");

Get(N);

end Lab1;

**data..ads**

generic

N : in Integer;

package Data is

type Vector is array(1..N) of Integer;

type Matrix is array(1..N) of Vector;

procedure F1(vectorA, vectorB, vectorC: in Vector; matrixA, matrixE: in Matrix; vectorD: out Vector);

procedure F2(matrixH, matrixK: in Matrix; integerRes: out Integer);

procedure F3(matrixR, matrixS: in Matrix; vectorR: in Vector; integerRes: out Integer);

procedure putVector(vect: in Vector);

procedure putMatrix(matr: in Matrix);

procedure randomVector(vect: out Vector);

procedure randomMatrix(matr: out Matrix);

procedure fillVector(vect: out Vector);

procedure fillMatrix(matr: out Matrix);

end Data;

**data.ads**

generic

N : in Integer;

package Data is

type Vector is array(1..N) of Integer;

type Matrix is array(1..N) of Vector;

procedure F1(vectorA, vectorB, vectorC: in Vector; matrixA, matrixE: in Matrix; vectorD: out Vector);

procedure F2(matrixH, matrixK: in Matrix; integerRes: out Integer);

procedure F3(matrixR, matrixS: in Matrix; vectorR: in Vector; integerRes: out Integer);

procedure putVector(vect: in Vector);

procedure putMatrix(matr: in Matrix);

procedure randomVector(vect: out Vector);

procedure randomMatrix(matr: out Matrix);

procedure fillVector(vect: out Vector);

procedure fillMatrix(matr: out Matrix);

end Data;

**data.adb**

with Ada.Text\_IO, Ada.Integer\_Text\_IO;

use Ada.Text\_IO, Ada.Integer\_Text\_IO;

with Ada.Numerics.Discrete\_Random;

package body data is

type Rand\_Value is range 1..20;

package Rand\_Int is new Ada.Numerics.Discrete\_Random(Rand\_Value);

use Rand\_Int;

gen: Rand\_Int.Generator;

procedure putVector(vect: in Vector) is

begin

for i in 1..N loop

Put(Integer'Image(vect(i)));

end loop;

Put\_Line("");

end putVector;

procedure putMatrix(matr: in Matrix) is

begin

for i in 1..N loop

putVector(matr(i));

end loop;

end putMatrix;

procedure randomVector(vect: out Vector) is

begin

for i in 1..N loop

Rand\_Int.Reset(gen);

vect(i) := INTEGER(Rand\_Int.Random(gen));

end loop;

end randomVector;

procedure randomMatrix(matr: out Matrix) is

begin

for i in 1..N loop

randomVector(matr(i));

end loop;

end randomMatrix;

procedure fillVector(vect: out Vector) is

begin

for i in 1..N loop

vect(i) := 1;

end loop;

end fillVector;

procedure fillMatrix(matr: out Matrix) is

begin

for i in 1..N loop

fillVector(matr(i));

end loop;

end fillMatrix;

function sortVector(vect : Vector) return Vector is

m : Integer;

resultVector: Vector;

begin

for i in 1..N loop

resultVector(i) := vect(i);

end loop;

for i in reverse 1..N loop

for j in 1..(i-1) loop

if resultVector(j) > resultVector(j+1) then

m := resultVector(j);

resultVector(j) := resultVector(j+1);

resultVector(j+1) := m;

end if;

end loop;

end loop;

return resultVector;

end SortVector;

function "\*"(matrA, matrB: in Matrix) return Matrix is

sum : Integer;

resultMatr : Matrix;

begin

for i in 1..N loop

for j in 1..N loop

sum := 0;

for k in 1..N loop

sum := sum + matrA(i)(k) \* matrB(k)(j);

end loop;

resultMatr(i)(j) := sum;

end loop;

end loop;

return resultMatr;

end "\*";

function "\*"(int:in Integer; vect: in Vector) return Vector is

resultVector: Vector;

begin

for i in 1..N loop

resultVector(i) := vect(i) \* int;

end loop;

return resultVector;

end "\*";

function "\*"(int: in Integer; matr: in Matrix) return Matrix is

resultMatrix: Matrix;

begin

for i in 1..N loop

for j in 1..N loop

resultMatrix(i)(j) := int \* matr(i)(j);

end loop;

end loop;

return resultMatrix;

end "\*";

function "\*"(vectorA, vectorB: in Vector) return Vector is

resultVector : Vector;

begin

for i in 1..N loop

resultVector(i) := vectorA(i) \* vectorB(i);

end loop;

return resultVector;

end "\*";

function "\*"(vect: in Vector; matr: in Matrix) return Vector is

resultVect: Vector;

sum : Integer;

begin

for i in 1..N loop

sum := 0;

for j in 1..N loop

sum := sum + matr(i)(j) \* vect(i);

end loop;

resultVect(i) := sum;

end loop;

return resultVect;

end "\*";

function "+"(vectorA, vectorB: in Vector) return Vector is

resultVector: Vector;

begin

for i in 1..N loop

resultVector(i) := vectorA(i) + vectorB(i);

end loop;

return resultVector;

end "+";

function sort(MA:Matrix) return Matrix is

MT:Matrix;

begin

for i in 1..n loop

MT(i):=sortVector(MA(i));

end loop;

return MT;

end sort;

function searchMax(A : Matrix) return Integer is

Var : Integer := A(1)(1);

begin

for i in 1..N loop

for j in 1..N loop

if A(i)(j) > Var then

Var := A(i)(j);

end if;

end loop;

end loop;

return Var;

end searchMax;

function searchMin(A : Matrix) return Integer is

Var : Integer := A(1)(1);

begin

for i in 1..N loop

for j in 1..N loop

if A(i)(j) < Var then

Var := A(i)(j);

end if;

end loop;

end loop;

return Var;

end searchMin;

procedure F1(vectorA, vectorB, vectorC: in Vector; matrixA, matrixE: in Matrix; vectorD: out Vector) is

begin

vectorD := (vectorA + vectorB) + (vectorC \* (vectorB \* (matrixA \* matrixE)));

end F1;

procedure F2(matrixH, matrixK: in Matrix; integerRes: out Integer) is

begin

integerRes := searchMin(matrixH \* matrixK);

end F2;

procedure F3(matrixR, matrixS: in Matrix; vectorR: in Vector; integerRes: out Integer) is

begin

integerRes := searchMax(((sort(matrixR) \* (matrixR \* matrixS))));

end F3;

begin

Rand\_Int.Reset(gen);

end data;

**Результати виконання програми:**

