Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# Лабораторна робота № 1

з дисципліни «Паралельні і розподілені обчислення»

Виконав студент групи ІО-01 *Редько Олександр*

## Завдання



**Лістинг:**

------------------------------------------------------------------

-- --

-- Паралельні і розподілені обчислення --

-- Лабораторна робота №1. Підпрограми і пакети --

-- --

-- Файл: lab1.ada --

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

-- F2: MD = (MA \* MB) \* TRANS(MC) --

-- F3: E = (MA \* MM) \* B + MB \* SORT(A) --

-- --

-- Автор: Редько Олександр, група IO-01 --

-- Дата: 09.09.2012 --

-- --

------------------------------------------------------------------

with Ada.Text\_IO, Data;

use Ada.Text\_IO;

procedure Lab1 is

package Data3 is new Data(3);

use Data3;

A, B, C, D, E, E1 : Vector;

MA, MB, MC, MD, MM, MZ : Matrix;

begin

----------------

-- Input data --

----------------

Put ("Input vector A: ");

Input\_Vector (A);

Output\_Vector (A);

New\_Line;

Put ("Input vector B: ");

Input\_Vector (B);

Output\_Vector (B);

New\_Line;

Put ("Input vector C: ");

Input\_Vector (C);

Output\_Vector (C);

New\_Line;

Put ("Input vector D: ");

Input\_Vector (D);

Output\_Vector (D);

New\_Line;

Put ("Input matrix MA: ");

Input\_Matrix (MA);

Output\_Matrix (MA);

New\_Line;

Put ("Input matrix MB: ");

Input\_Matrix (MB);

Output\_Matrix (MB);

New\_Line;

Put ("Input matrix MC: ");

Input\_Matrix (MC);

Output\_Matrix (MC);

New\_Line;

Put ("Input matrix MM: ");

Input\_Matrix (MM);

Output\_Matrix (MM);

New\_Line;

Put ("Input matrix MZ: ");

Input\_Matrix (MZ);

Output\_Matrix (MZ);

New\_Line;

-----------------

-- Calculation --

-----------------

F1 (A, B, C, D, MA, MZ, E1);

F2 (MA, MB, MC, MD);

E := E1;

F3 (A, B, MA, MB, MM, E);

-----------------

-- Output data --

-----------------

Put ("Function F1, vector E: ");

Output\_Vector (E1);

New\_Line;

Put ("Function F2, matrix MD: ");

Output\_Matrix (MD);

New\_Line;

Put ("Function F3, vector E: ");

Output\_Vector (E);

New\_Line;

end Lab1;

------------------------------------------------------------------

-- File: data.ads --

-- --

-- Author: Redko Alexander, group IO-01 --

-- Date: 09.09.2012 --

------------------------------------------------------------------

generic

N: in Natural; -- dimension of Vector and Matrix(N \* N)

package Data is

type Vector is private;

type Matrix is private;

---------------------------------------

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

---------------------------------------

procedure F1

(VA, VB, VC, VD : in Vector;

MA, MZ : in Matrix;

VE : out Vector);

------------------------------------

-- F2: MD = (MA \* MB) \* TRANS(MC) --

------------------------------------

procedure F2

(MA, MB, MC : in Matrix;

MD : out Matrix);

------------------------------------------

-- F3: E = (MA \* MM) \* B + MB \* SORT(A) --

------------------------------------------

procedure F3

(VA, VB : in Vector;

MA, MB, MM : in Matrix;

VE : out Vector);

------------------

-- Input\_Vector --

------------------

procedure Input\_Vector(V : out Vector);

------------------

-- Input\_Matrix --

------------------

procedure Input\_Matrix(MA : out Matrix);

-------------------

-- Output\_Vector --

-------------------

procedure Output\_Vector(V : in Vector);

-------------------

-- Output\_Matrix --

-------------------

procedure Output\_Matrix(MA : in Matrix);

private

subtype Index is Integer range 1..N;

type Vector is array (Index) of Integer;

type Matrix is array (Index) of Vector;

end Data;

------------------------------------------------------------------

-- File: data.adb --

-- Author: Redko Alexander, group IO-01 --

-- Date: 09.09.2012 --

------------------------------------------------------------------

with Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Numerics.Discrete\_Random;

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

package body Data is

------------------

-- Input\_Vector --

------------------

procedure Input\_Vector (V : out Vector) is

begin

for I in Index loop

Get(V(I));

end loop;

end Input\_Vector;

------------------

-- Input\_Matrix --

------------------

procedure Input\_Matrix (MA : out Matrix) is

begin

for I in Index loop

for J in Index loop

Get(MA(I)(J));

end loop;

end loop;

end Input\_Matrix;

-------------------

-- Output\_Vector --

-------------------

procedure Output\_Vector (V : in Vector) is

begin

New\_Line;

for I in Index loop

Put(Item => V(I), Width => 5);

end loop;

New\_Line;

end Output\_Vector;

-------------------

-- Output\_Matrix --

-------------------

procedure Output\_Matrix (MA : in Matrix) is

begin

New\_Line;

for I in Index loop

for J in Index loop

Put(Item => MA(i)(j), Width => 5);

end loop;

New\_line;

end loop;

end Output\_Matrix;

function "\*" --Matrix\_Matrix\_Multiply

(Left : Matrix;

Right : Matrix) return Matrix;

function "\*" --Vector\_Matrix\_Multiply

(Left : Vector;

Right : Matrix) return Vector;

function "+" --Vector\_Vector\_Add

(Left : Vector;

Right : Vector) return Vector;

procedure Transpose (A : in Matrix; R : out matrix);

procedure Sort (V : in Vector; A : out Vector);

---------------------------------------

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

---------------------------------------

procedure F1

(VA, VB, VC, VD : in Vector;

MA, MZ : in Matrix;

VE : out Vector) is

begin

VE := VA + VB + VC + VD \* (MA \* MZ);

end F1;

------------------------------------

-- F2: MD = (MA \* MB) \* TRANS(MC) --

------------------------------------

procedure F2

(MA, MB, MC : in Matrix;

MD : out Matrix) is

MCT : Matrix;

begin

Transpose(MC, MCT);

MD := (MA \* MB) \* MCT;

end F2;

------------------------------------------

-- F3: E = (MA \* MM) \* B + MB \* SORT(A) --

------------------------------------------

procedure F3

(VA, VB : in Vector;

MA, MB, MM : in Matrix;

VE : out Vector) is

VS : Vector;

begin

Sort(VE, VS);

VE := VB \* (MA \* MM) + VS \* MB;

end F3;

function "\*"--Matrix\_Matrix\_Multiply

(Left : Matrix;

Right : Matrix) return Matrix

is

MR : Matrix;

begin

for i in Index loop

for J in Index loop

MR(I)(J) := 0;

for K in Index loop

MR(I)(J) := MR(I)(J) + Left(I)(K) \* Right(K)(J);

end loop;

end loop;

end loop;

return MR;

end "\*";

function "\*"--Vector\_Matrix\_Multiply

(Left : Vector;

Right : Matrix) return Vector

is

R : Vector;

begin

for J in Index loop

R(j) := 0;

begin

for K in Index loop

R(J) := R(J) + Left(K) \* Right(K )(J);

end loop;

end;

end loop;

return R;

end "\*";

function "+"--Vector\_Vector\_Add

(Left : Vector;

Right : Vector) return Vector

is

R : Vector;

begin

for J in Index loop

R (J) := Left (J) + Right (J);

end loop;

return R;

end "+";

procedure Transpose (A : in Matrix; R : out Matrix) is

begin

for J in Index loop

for K in Index loop

R (J)(K) := A (K)(j);

end loop;

end loop;

end Transpose;

procedure Sort (V : in Vector; A : out Vector) is

Min : Positive;

Temp : Integer;

begin

A := V;

for I in A'First..A'Last - 1 loop

Min := I;

for J in I + 1..A'Last loop

if A (Min) > A (J) then

Min := J;

end if;

end loop;

if Min /= I then

Temp := A (I);

A (I) := A (Min);

A (Min) := Temp;

end if;

end loop;

end Sort;

end Data;