НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №1

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:

```
---------------------------------------
data.ads
---------------------------------------

with Ada.Text_IO, Ada.Long_Integer_Text_IO, System.Multiprocessors, Ada.Numerics.Discrete_Random;
use Ada.Text_IO, Ada.Long_Integer_Text_IO, System.Multiprocessors;

generic N: Long_Integer;
package Data is
        type Vector is private;
        type Matrix is private;

        procedure Read_Vector(A: out Vector);
        procedure Read_Matrix(MA: out Matrix);
        procedure Fill_Vector_1(A: in out Vector);
        procedure Fill_Matrix_1(MA: in out Matrix);
        function Multiply_Matrices(MA, MB: in Matrix) return Matrix;
        function Multiply_Vector_Matrix(A: in Vector; MB: in Matrix) return Vector;
        function Multiply_Scalar_Matrix(A: in Long_Integer; MB: in Matrix) return Matrix;
        function Sum_Vectors(A, B: in Vector) return Vector;
        function Sum_Matrices(MA, MB: in Matrix) return Matrix;
        function Min_Vector(A: in Vector) return Long_Integer;
        function Min_Matrix(MA: in Matrix) return Long_Integer;
        function Max_Vector(A: in Vector) return Long_Integer;
        function Max_Matrix(MA: in Matrix) return Long_Integer;
        function Sort_Vector(A: in Vector) return Vector;
        procedure Print_Vector(A: in Vector);
        procedure Print_Matrix(MA: in Matrix);
        procedure Funcs(C: out Vector; A: in Vector; MA, ME: in Matrix; B, D: in Vector; ML: out Matrix; MF,
MG, MH, MK: in Matrix; T: out Vector; MO, MP: in Matrix; S: in Vector; MR: in Matrix);

private
        type Vector is array(1..N) of Long_Integer;
        type Matrix is array(1..N, 1..N) of Long_Integer;
end Data;

---------------------------------------
data.adb
---------------------------------------

package body Data is

        procedure Read_Vector(A: out Vector) is
        begin
                for i in 1..N loop
                        Get(A(i));
                end loop;
        end Read_Vector;

        procedure Read_Matrix(MA: out Matrix) is
        begin
                for i in 1..N loop
                        for k in 1..N loop
                                Get(MA(i, k));
                        end loop;
                end loop;
        end Read_Matrix;

        procedure Fill_Vector(A: in out Vector) is

                subtype r is range 1..20;
                package Random is new Ada.Numerics.Discrete_Random(r);
                use Random;
                G: Generator;
                D: Dice;

        begin
        Reset(G);
        for i in 1..N loop
                A(i) := Random(G);
        end loop;
end Fill_Vector;

procedure Fill_Matrix(MA: in out Matrix) is

        subtype r is range 1..20;
        package Random is new Ada.Numerics.Discrete_Random(r);
        use Random;
        G: Generator;
        D: Dice;

begin
        Reset(G);
        for i in 1..N loop
                for k in 1..N loop
                        MA(i, k) := Random(G);
                end loop;
        end loop;
end Fill_Matrix;

function Multiply_Matrices(MA, MB: in Matrix) return Matrix is

        res: Matrix;

begin
        for i in 1..N loop
                for k in 1..N loop
                        res(i, k) := 0;
                        for j in 1..N loop
                                res(i, k) := res(i, k) + MA(i, j) * MB(j, k);
                        end loop;
                end loop;
        end loop;

        return res;
end Multiply_Matrices;

function Multiply_Vector_Matrix(A: in Vector; MB: in Matrix) return Vector is

        res: Vector;

begin
        for i in 1..N loop
                res(i) := 0;
                for j in 1..N loop
                        res(i) := res(i) + MB(i, j) * A(j);
                end loop;
        end loop;

        return res;
end Multiply_Vector_Matrix;

function Multiply_Scalar_Matrix(A: in Long_Integer; MB: in Matrix) return Matrix is

        res: Matrix;

begin
        for i in 1..N loop
                for k in 1..N loop
                        res(i, k) := A * MB(i, k);
                end loop;
        end loop;

        return res;
end Multiply_Scalar_Matrix;

function Sum_Vectors(A, B: in Vector) return Vector is

        res: Vector;

begin
        for i in 1..N loop
                res(i) := A(i) + B(i);
        end loop;

        return res;
end Sum_Vectors;
```

```
function Sum_Matrices(MA, MB: in Matrix) return Matrix is

        res: Matrix;

begin
        for i in 1..N loop
                for k in 1..N loop
                        res(i, k) := MA(i, k) + MB(i, k);
                end loop;
        end loop;

        return res;
end Sum_Matrices;

function Min_Vector(A: in Vector) return Long_Integer is

res: Long_Integer;

begin
        res := A(1);
        for i in 2..N loop
                if res < A(i) then
                        res := A(i);
                end if;
        end loop;

        return res;
end Min_Vector;

function Min_Matrix(MA: in Matrix) return Long_Integer is

        res: Long_Integer;

begin
        res := MA(1, 1);
        for i in 1..N loop
                for k in 1..N loop
                        if res < MA(i, k) then
                                res := MA(i, k);
                        end if;
                end loop;
        end loop;

        return res;
end Min_Matrix;

function Max_Vector(A: in Vector) return Long_Integer is

res: Long_Integer;

begin
        res := A(1);
        for i in 2..N loop
                if res > A(i) then
                        res := A(i);
                end if;
        end loop;

        return res;
end Max_Vector;

function Max_Matrix(MA: in Matrix) return Long_Integer is

        res: Long_Integer;

begin
        res := MA(1, 1);
        for i in 1..N loop
                for k in 1..N loop
                        if res > MA(i, k) then
                                res := MA(i, k);
                        end if;
                end loop;
        end loop;

        return res;
end Max_Matrix;

function Sort_Vector(A: in Vector) return Vector is
```

```
                res: Vector;
                t: Long_Integer;

        begin
                res := A;
                for i in 1..N loop
                        for k in 1..N-i loop
                                if res(i) > res(i + 1) then
                                        t := res(i);
                                        res(i) := res(i + 1);
                                        res(i + 1) := t;
                                end if;
                        end loop;
                end loop;

                return res;
        end Sort_Vector;

        procedure Print_Vector(A: in Vector) is
        begin
                for i in 1..N loop
                        Put(A(i));
                end loop;
                New_Line;
        end Print_Vector;

        procedure Print_Matrix(MA: in Matrix) is
        begin
                for i in 1..N loop
                        for k in 1..N loop
                                Put(MA(i, k));
                        end loop;
                        New_Line;
                end loop;
        end Print_Matrix;

        procedure Funcs(C: out Vector; A: in Vector; MA, ME: in Matrix; B, D: in Vector; ML: out Matrix; MF,
MG, MH, MK: in Matrix; T: out Vector; MO, MP: in Matrix; S: in Vector; MR: in Matrix) is

                task Func1 with CPU=>1 is
                        pragma Priority(10);
                        pragma Storage_Size(300_000_000);
                end Func1;

                task Func2 with CPU=>4 is
                        pragma Priority(9);
                        pragma Storage_Size(300_000_000);
                end Func2;

                task Func3 with CPU=>3 is
                        pragma Priority(8);
                        pragma Storage_Size(300_000_000);
                end Func3;

                task body Func1 is
                begin
                        Put_Line("Task 1 begin");
                        C := Sum_Vectors(Multiply_Vector_Matrix(A, Multiply_Matrices(MA, ME)), Sum_Vectors(B,
D));
                        Put_Line("Task 1 end");
                end Func1;

                task body Func2 is
                begin
                        Put_Line("Task 2 begin");
                        ML := Sum_Matrices(Multiply_Scalar_Matrix(Min_Matrix(MF), MG),
Multiply_Scalar_Matrix(Max_Matrix(MH), Multiply_Matrices(MK, MF)));
                        Put_Line("Task 2 end");
                end Func2;

                task body Func3 is
                begin
                        Put_Line("Task 3 begin");
                        T := Sum_Vectors(Multiply_Vector_Matrix(S, Multiply_Matrices(MO, MP)),
Multiply_Vector_Matrix(Sort_Vector(S), MR));
                        Put_Line("Task 3 end");
                end Func3;

        begin
```

```
                null;
        end Funcs;


end Data;

----------------------------------------
Lab1.adb
----------------------------------------

with Data, Ada.Text_IO, Ada.Integer_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO;

procedure Lab1 is

        N: constant Long_Integer := 1000;

        package ConcreteData is new Data(N);
        use ConcreteData;

        A, B, C, D, T, S: Vector;
        MA, ME, ML, MF, MG, MH, MK, MO, MP, MR: Matrix;

begin
        Put_Line("Enter vector A:");
        Fill_Vector(A);

        Put_Line("Enter vector B:");
        Fill_Vector(B);

        Put_Line("Enter vector D:");
        Fill_Vector(D);

        Put_Line("Enter maxtrix MA:");
        Fill_Matrix(MA);

        Put_Line("Enter maxtrix ME:");
        Fill_Matrix(ME);

        Put_Line("Enter maxtrix MF:");
        Fill_Matrix(MF);

        Put_Line("Enter maxtrix MG:");
        Fill_Matrix(MG);

        Put_Line("Enter maxtrix MH:");
        Fill_Matrix(MH);

        Put_Line("Enter maxtrix MK:");
        Fill_Matrix(MK);

        Put_Line("Enter maxtrix MO:");
        Fill_Matrix(MO);

        Put_Line("Enter maxtrix MP:");
        Fill_Matrix(MP);

        Put_Line("Enter vector MS:");
        Fill_Vector(S);

        Put_Line("Enter maxtrix MR:");
        Fill_Matrix(MR);

        Funcs(C, A, MA, ME, B, D, ML, MF, MG, MH, MK, T, MO, MP, S, MR);

        Put_Line("F1 =");
        Print_Vector(C);
        Put_Line("F2 =");
        Print_Matrix(ML);
        Put_Line("F3 =");
        Print_Vector(T);
end Lab1;
```