Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Основи паралельного програмування

Лабораторна робота №1

**«Потоки в мові Ada»**

Виконала:

студентка групи ІВ-71

Молчанова В.С.

Перевірив:

Долголенко О.М.

Київ

2019 р.

**Завдання:**

F1: c = MAX(MA\*MB)\*(A\*B)

F2: MF = MAX(MG)\*(MH\*MK)

F3: T= SORT(O + P)\*TRANS(MR\*MS)

**Лістинг програми:**

**main.adb**

----------------Main programm------------------------

--Parallel and distributed computing.

--Labwork 1. Ada. Subprograms and packages

--Varvara Molchanova

--IV-71

--04.10.2019

--F1: c = MAX(MA\*MB)\*(A\*B)

--F2: MF = MAX(MG)\*(MH\*MK)

--F3: T = SORT(O+P)\*TRANS(MR\*MS)

with data, semaphore, Text\_IO, Ada.Integer\_Text\_IO, System.Multiprocessors;

use semaphore, Text\_IO, Ada.Integer\_Text\_IO, System.Multiprocessors;

procedure Main is

N: Integer := 3;

package data1 is new data (N);

use data1;

S: CountingSemaphore(1, 1);

procedure tasks is

task T1 is

pragma Priority(1);

pragma Storage\_Size(100000);

pragma CPU(1);

end;

task body T1 is

A, B: Vector;

MA, MB: Matrix;

f1: Integer;

begin

Put\_Line("T1 started");

Fill\_With\_Ones(A);

Fill\_With\_Ones(B);

Fill\_With\_Ones(MA);

Fill\_With\_Ones(MB);

f1:=Func1(A, B, MA, MB);

S.Wait;

Put\_Line("---F1: c = MAX(MA\*MB)\*(A\*B) ---");

Put\_Line(To\_String(f1));

Put\_Line("T1 finished");

New\_Line;

New\_Line;

S.Signal;

end T1;

task T2 is

pragma Priority(3);

pragma Storage\_Size(100000);

pragma CPU(2);

end;

task body T2 is

MG, MH, MK, f2: Matrix;

begin

Put\_Line("T2 started");

Fill\_With\_Ones(MG);

Fill\_With\_Ones(MH);

Fill\_With\_Ones(MK);

f2:=Func2(MG, MH, MK);

S.Wait;

Put\_Line("---F2: MF = MAX(MG)\*(MH\*MK)---");

Matrix\_Output(f2);

Put\_Line("T2 finished");

New\_Line;

New\_Line;

S.Signal;

end T2;

task T3 is

pragma Priority(5);

pragma Storage\_Size(100000);

pragma CPU(3);

end;

task body T3 is

MR, MS : Matrix;

O, P: Vector;

f3: Vector;

begin

Put\_Line("T3 started");

Fill\_With\_Ones(O);

Fill\_With\_Ones(P);

Fill\_With\_Ones(MR);

Fill\_With\_Ones(MS);

f3:=Func3(O, P, MR, MS);

S.Wait;

Put\_Line("---F3: T = SORT(O+P)\*TRANS(MR\*MS)---");

Vector\_Output(f3);

Put\_Line("T3 finished");

New\_Line;

New\_Line;

S.Signal;

end T3;

begin

null;

end tasks;

Begin

tasks;

End Main;

**data.ads**

generic

N: Integer;

package data is

---Declaration of private types

type Vector is private;

type Matrix is private;

function To\_String(I : Integer) return String;

--Write vector on screen

procedure Vector\_Output(A: in Vector);

--Write matrix on screen

procedure Matrix\_Output (A: in Matrix);

--Calculate F1

function Func1 (A, B: in Vector; MA, MB : in Matrix) return Integer;

--Calculate F2

function Func2 (MG, MH, MK: in Matrix) return Matrix;

--Calculate F3

function Func3 (O, P: in Vector; MR, MS : in Matrix) return Vector;

--Filling matrix with ones

procedure Fill\_With\_Ones(MA: in out Matrix);

--Filling vector with ones

procedure Fill\_With\_Ones (A: in out Vector);

procedure Fill\_By\_Number(A: in out Matrix);

procedure Fill\_By\_Number(A: in out Vector);

--Determination private types and methods

private

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

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

--Scalar multiplication of vectors

function Multiply(A, B: in Vector) return Integer;

--Multiplication of matrices

function Multiply(MA, MB: in Matrix) return Matrix;

--Multiplication of matrix and integer

function Multiply(MA: in Matrix; k: in Integer) return Matrix;

function Multiply(k: in Integer; MA: in Matrix) return Matrix;

--Multiplication of vector and matrix

function Multiply(A: in Vector; MB: in Matrix) return Vector;

function Multiply(MA: in Matrix; B: in Vector) return Matrix;

--Multiplication of Vector and Integer

function Multiply (A: in Vector; k: in Integer) return Vector;

function Multiply (k: in Integer; A: in Vector ) return Vector;

--Sum of vectors

function Add(A, B: in Vector) return Vector;

--Sum of matrixes

function Add(MA, MB: in Matrix) return Matrix;

--Sorting of vector

procedure Sort(A: in out Vector);

--Sorting of matrix

procedure Sort(MA: in out Matrix);

--Transposition of Matrix

function Transpose(A: in Matrix) return Matrix;

function Max(V: in Vector) return Integer;

function Max(M: in Matrix) return Integer;

end data;

**data.adb**

-----------Package Data, body-----------

with Text\_IO, Ada.Integer\_Text\_IO, Ada.Strings.Unbounded, Ada.Strings.Fixed;

use Text\_IO, Ada.Integer\_Text\_IO, Ada.Strings.Unbounded, Ada.Strings.Fixed;

package body data is

function To\_String(I : Integer) return String is

begin

return Trim(Integer'Image(I), Ada.Strings.Left);

end To\_String;

--Write vector on screen

procedure Vector\_Output(A: in Vector) is

S: Ada.Strings.Unbounded.Unbounded\_String;

begin

for i in 1..N loop

S:=S & To\_Unbounded\_String(To\_String(A(i))) & To\_Unbounded\_String(" ");

end loop;

Put\_Line(Ada.Strings.Unbounded.To\_String(S));

end Vector\_Output;

--Write matrix on screen

procedure Matrix\_Output (A: in Matrix) is

begin

for i in 1..N loop

for j in 1..N loop

Put(A(i)(j));

Put(" ");

end loop;

Put\_Line(" ");

end loop;

end Matrix\_Output;

--Filling matrix with ones

procedure Fill\_With\_Ones(MA: in out Matrix) is

begin

for i in 1..N loop

Fill\_With\_Ones(MA(i));

end loop;

end Fill\_With\_Ones;

--Filling vector with ones

procedure Fill\_With\_Ones (A: in out vector) is

begin

for i in 1..N loop

A(i) := 1;

end loop;

end Fill\_With\_Ones;

procedure Fill\_By\_Number(A: in out Matrix) is

begin

for i in 1..N loop

for j in 1..N loop

A(i)(j) := i \* N + j;

end loop;

end loop;

end Fill\_By\_Number;

procedure Fill\_By\_Number(A: in out Vector) is

begin

for i in 1..N loop

A(i) := i;

end loop;

end Fill\_By\_Number;

--Scalar multiplication of vectors

function Multiply(A, B: in Vector) return Integer is

result: Integer;

begin

result := 0;

for i in 1..N loop

result := result + A(i)\*B(i);

end loop;

return result;

end Multiply;

--Multiplication of matrices

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

result, MB\_Transposed: Matrix;

begin

MB\_Transposed := Transpose(MB);

for i in 1..N loop

for j in 1..N loop

result(i)(j) := Multiply(MA(i), MB\_Transposed(j));

end loop;

end loop;

return result;

end Multiply;

--Multiplication of matrix and integer

function Multiply(MA: in Matrix; k: in Integer) return Matrix is

MP: Matrix;

begin

for i in 1..N loop

MP(i) := Multiply(MA(i), k);

end loop;

return MP;

end Multiply;

function Multiply(k: in Integer; MA: in Matrix) return Matrix is

begin

return Multiply(MA, k);

end Multiply;

--Multiplication of vector and matrix

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

P: Vector;

S: Integer;

begin

for i in 1..N loop

s := 0;

for j in 1..N loop

S := s + A(i)\*MB(j)(i);

end loop;

P(i) := S;

end loop;

return P;

end Multiply;

function Multiply(MA: in Matrix; B: in Vector) return Matrix is

MP: Matrix;

S: Integer;

begin

for i in 1..N loop

s := 0;

for j in 1..N loop

S := s + MA(i)(j)\*B(j);

MP(i)(j) := S;

end loop;

end loop;

return MP;

end Multiply;

--Multiplication of Vector and Integer

function Multiply (A: in Vector; k: in Integer) return Vector is

B: Vector;

begin

for i in 1..N loop

B(i) := A(i)\*k;

end loop;

return B;

end Multiply;

function Multiply (k: in Integer; A: in Vector) return Vector is

begin

return Multiply(A, k);

end Multiply;

--Sum of vectors

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

S: Vector;

begin

for i in 1..N loop

S(i) := A(i)+B(i);

end loop;

return S;

end Add;

--Sum of matrixes

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

MS: Matrix;

begin

for i in 1..N loop

MS(i) := Add(MA(i), MB(i));

end loop;

return MS;

end Add;

--Sorting of vector

procedure Sort(A: in out Vector) is

S: Integer;

begin

for i in 1..N loop

for j in i..N loop

if A(i)>A(j) then

S:=A(j);

A(j):=A(i);

A(i):=S;

end if;

end loop;

end loop;

end Sort;

--Sorting of matrix

procedure Sort(MA: in out Matrix) is

begin

for i in 1..N loop

Sort(MA(i));

end loop;

end Sort;

--Transposition of Matrix

function Transpose(A: in Matrix) return Matrix is

result: Matrix;

begin

for i in 1..N loop

for j in i..N loop

result(j)(i) := A(i)(j);

result(i)(j) := A(j)(i);

end loop;

end loop;

return result;

end Transpose;

function Max(V: in Vector) return Integer is

maximum: Integer;

begin

maximum := V(1);

for i in 2..N loop

if V(i) > maximum then

maximum := V(i);

end if;

end loop;

return maximum;

end Max;

function Max(M: in Matrix) return Integer is

maximum: Integer;

begin

maximum := Max(M(1));

for i in 2..N loop

if Max(M(i)) > maximum then

maximum := Max(M(i));

end if;

end loop;

return maximum;

end Max;

--Calculate F1

function Func1(A, B: in Vector; MA, MB : in Matrix) return Integer is

MC: Matrix;

c, d: Integer;

begin

MC := Multiply(MA, MB);

c := Max(MC);

d := Multiply(A, B);

return c \* d;

end Func1;

--Calculate F2

function Func2(MG, MH, MK: in Matrix) return Matrix is

MD, MN: Matrix;

k: Integer;

begin

k := Max(MG);

MD:=Multiply(MH,MK);

MN:=Multiply(MD, k);

return MN;

end Func2;

--Calculate F3

function Func3 (O, P: in Vector; MR, MS : in Matrix) return Vector is

MD:Matrix;

R, S:Vector;

begin

R := Add(O, P);

Sort(R);

MD:=Transpose(Multiply(MR,MS));

S:=Multiply(R,MD);

return S;

end Func3;

end data;

**Результат роботи:**

T1 started

T2 started

T3 started

---F1: c = MAX(MA\*MB)\*(A\*B) ---

9

T1 finished

---F2: MF = MAX(MG)\*(MH\*MK)---

3 3 3

3 3 3

3 3 3

T2 finished

---F3: T = SORT(O+P)\*TRANS(MR\*MS)---

18 18 18

T3 finished