Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорчького“
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Основи паралельного програмування

Лабораторна робота №1
“ПОТОКИ В МОВІ АДА. ЗАДАЧІ”

Виконала:
студентка групи ІВ-81
Дьяченко Т. С.
Перевірив:
Корочкін О.В.

Київ
2020 р.

ЗАВДАННЯ:

	ПІБ	F1	F2	F2
12	Дьяченко Тетяна Сергіївна	1.28	2.16	3.21

F1: E = MAX(A)*(X+B*(MA*MD)+C)
F2: ML = SORT(TRANS(MF)*MK)
F3: S = SORT(O*MO)*(MS*MT)

ВИКОНАННЯ РОБОТИ:

lib1.adb — основний файл програми, що містить створення паралельних потоків.
data.adb — файл, що містить тіло пакета data. В ньому прописані функції та процедури, що необхідні для обчислення функцій F1, F2, F3 та підпрограми, що безпосередньо обчислюють ці функції з використанням цих ресурсів.
data.ads — файл, що містить специфікацію пакету data.

ВИСНОВКИ:

В ході виконання завдання лабораторної роботи було створену паралельну програму на мові Ada. При вводі даних виникла проблема: усі потоки одночасно зверталися до консолі та клавіатури, тому неможливо було визначити які дані в які змінні потрапили. Цю проблему було вирішено за допомогою трьох об’єктів типу Suspension_object з пакету Ada.Synchronous_Task_Control. Кожен об’єкт відповідає за доступ відповідної задачі (потоку) до ресурсу консолі (прилад вводу/виводу). Спочатку доступ надається першому потоку, який, після завершення вводу своїх даних надає доступ другому, який, після зчитування своїх даних передає третьому. Цикл повторюється для почергового виводу результатів. Як альтернативне рішення було написано функції для автоматичного заповнення векторів та матриць, що можуть бути викликані в коді замість функцій, що читають з клавіатури.

ДОДАТОК. Лістінг програми:

lib1.adb

```
--------------------------------
-- Lab 1 Var 12
-- F1: E = MAX(A)*(X+B*(MA*MD)+C)
-- F2: ML = SORT(TRANS(MF)*MK)
-- F3: S = SORT(O*MO)*(MS*MT)
-- Diachenko Tetiana IV-81
-- Date 10.09.2020
```

```
--------------------------------

with Data, Text_IO, Ada.Integer_Text_IO, System.Multiprocessors,
Ada.Synchronous_Task_Control;
use Text_IO, Ada.Integer_Text_IO, System.Multiprocessors,
Ada.Synchronous_Task_Control;

procedure Lab1 is

	n: Integer := 3;
	package data_1 is new data(n);
	use data_1;

	SO1: Suspension_object;
	SO2: Suspension_object;
	SO3: Suspension_object;

	task T1 is
		pragma Priority(1);
		pragma Storage_Size(1000000);
		pragma CPU(1);
	end;
	task body T1 is
		A,B,C,X,E: Vector;
		f: Integer := 1;
		MA,MD: Matrix;
	begin
		Put_Line("T1 started.");
		delay(0.2);
		New_Line;
		Put_Line("-- F1: E = MAX(A)*(X+B*(MA*MD)+C) --");
		Put_Line("Input elements of vector A one-by-one:");
		Vector_Input(A);
		Put_Line("Input elements of vector B one-by-one:");
		Vector_Input(B);
		Put_Line("Input elements of vector C one-by-one:");
		Vector_Input(C);
		Put_Line("Input elements of vector X one-by-one:");
		Vector_Input(X);
		Put_Line("Input elements of matrix MA one-by-one:");
		Matrix_Input(MA);
		Put_Line("Input elements of matrix MD one-by-one:");
		Matrix_Input(MD);
		New_Line;
		New_Line;
```

```
        Set_True(SO2);

        E := Func1(A,B,C,X,MA,MD);

        if n < 7 then
                Suspend_Until_True(SO1);
                Set_False(SO1);
                Put_Line("-- F1: E = MAX(A)*(X+B*(MA*MD)+C) --");
                Vector_Print(E);
                New_Line;
                New_Line;
                Set_True(SO2);
        end if;

        Put_Line("T1 finished.");
end T1;


task T2 is
        pragma Priority(5);
        pragma Storage_Size(1000000);
        pragma CPU(2);
end;
task body T2 is
        f: Integer := 1;
        MF,MK,ML: Matrix;
begin
        Put_Line("T2 started.");
        Suspend_Until_True(SO2);
        Set_False(SO2);
        New_Line;
        Put_Line("-- F2: ML = SORT(TRANS(MF)*MK) --");
        Put_Line("Input elements of matrix MF one-by-one:");
        Matrix_Input(MF);
        Put_Line("Input elements of matrix MK one-by-one:");
        Matrix_Input(MK);
        New_Line;
        New_Line;
        Set_True(SO3);


        ML := Func2(MF,MK);

        if n < 7 then
                Suspend_Until_True(SO2);
```

```
                Set_False(SO2);
                Put_Line("-- F2: ML = SORT(TRANS(MF)*MK) --");
                Matrix_Print(ML);
                New_Line;
                New_Line;
                Set_True(SO3);
            end if;

            Put_Line("T2 finished.");
        end T2;


task T3 is
        pragma Priority(3);
            pragma Storage_Size(1000000);
            pragma CPU(3);
        end;
        task body T3 is
            f: Integer := 1;
            O,S: Vector;
            MO,MS,MT: Matrix;
        begin
            Put_Line("T3 started.");
            Suspend_Until_True(SO3);
            Set_False(SO3);
            New_Line;
            Put_Line("-- F3: S = SORT(O*MO)*(MS*MT) --");
            Put_Line("Input elements of vector O one-by-one:");
            Vector_Input(O);
            Put_Line("Input elements of matrix MO one-by-one:");
            Matrix_Input(MO);
            Put_Line("Input elements of matrix MS one-by-one:");
            Matrix_Input(MS);
            Put_Line("Input elements of matrix MT one-by-one:");
            Matrix_Input(MT);
            Set_True(SO1);

            S := Func3(O,MO,MS,MT);

            if n < 7 then
                Suspend_Until_True(SO3);
                Set_False(SO3);
                Put_Line("-- F3: S = SORT(O*MO)*(MS*MT) --");
                Vector_Print(S);
                New_Line;
```

```
                New_Line;
                Set_True(SO1);
            end if;

            Put_Line("T3 finished.");
        end T3;

begin
        Put_Line("Lab 1 started.");
end Lab1;
```

data.adb

```
--------------Package Data, body-------------
with Text_IO, Ada.Integer_Text_IO;
use Text_IO, Ada.Integer_Text_IO;

package body Data is

-----Read Vector from Input
procedure Vector_Input(A: out Vector) is
begin
        for i in 1..n loop
            Get(A(i));
        end loop;
        New_Line;
end Vector_Input;

-----Read Matrix from Input
procedure Matrix_Input(MA: out Matrix) is
begin
        for i in 1..n loop
            for j in 1..n loop
                Get(MA(i)(j));
            end loop;
            New_Line;
        end loop;
end Matrix_Input;


-----Fill Vector with Number
procedure Vector_Fill(A: out Vector; b: Integer) is
begin
        for i in 1..n loop
            A(i) := b;
```

```
        end loop;
end Vector_Fill;


-----Fill Matrix with Number
procedure Matrix_Fill(MA: out Matrix; a: Integer) is
begin
        for i in 1..n loop
                for j in 1..n loop
                        MA(i)(j) := a;
                end loop;
        end loop;
end Matrix_Fill;


-----Set Vector Element
procedure Vector_Set_Element(A: out Vector; i,b: Integer) is
begin
        A(i) := b;
end Vector_Set_Element;


-----Set Matrix Element
procedure Matrix_Set_Element(MA: out Matrix; i,j,a: Integer) is
begin
        MA(i)(j) := a;
end Matrix_Set_Element;


-----Multiply Scalar and Vector
function Mul_Vector_Scalar(A: in Vector; b: Integer) return Vector is
        R: Vector;
begin
        for i in 1..n loop
                R(i) := A(i)*b;
        end loop;
        return R;
end Mul_Vector_Scalar;


-----Multiply Vector and Matrix
function Mul_Vector_Matrix(A: in Vector; MA: in Matrix) return Vector is
        R: Vector;
        s: Integer;
begin
```

```
        for i in 1..n loop
                s := 0;
                for j in 1..n loop
                        s := s + A(j)*MA(j)(i);
                end loop;
                R(i) := s;
        end loop;
        return R;
end Mul_Vector_Matrix;


-----Multiply Matrix and Matrix
function Mul_Matrix_Matrix(MA,MB: in Matrix) return Matrix is
        MR: Matrix;
        s: Integer;
begin
        for i in 1..n loop
                for j in 1..n loop
                        s := 0;
                        for k in 1..n loop
                                s := s + MA(i)(k)*MB(k)(j);
                        end loop;
                        MR(i)(j) := s;
                end loop;
        end loop;
        return MR;
end Mul_Matrix_Matrix;


-----Add Two Vectors
function Add_Vectors(A,B: in Vector) return Vector is
        R: Vector;
begin
        for i in 1..n loop
                R(i) := A(i) + B(i);
        end loop;
        return R;
end Add_Vectors;


-----Transpose Matrix
procedure Transpose_Matrix(MA: in out Matrix) is
        t: Integer;
begin
        for i in 1..n loop
```

```
            for j in i..n loop
                t := MA(i)(j);
                MA(i)(j) := MA(j)(i);
                MA(j)(i) := t;
            end loop;
        end loop;
end Transpose_Matrix;


-----Sort Vector
procedure Sort_Vector(A: in out Vector) is
        t: Integer;
begin
        for i in 1..n loop
            for j in i..n loop
                if A(i) > A(j) then
                    t := A(i);
                    A(i) := A(j);
                    A(j) := t;
                end if;
            end loop;
        end loop;
end Sort_Vector;


-----Sort Matrix
procedure Sort_Matrix(MA: in out Matrix) is
        t: Integer;
begin
        for i in 1..n loop
            for j in 1..n loop
                for k in j..n loop
                    if MA(i)(j) > MA(i)(k) then
                        t := MA(i)(j);
                        MA(i)(j) := MA(i)(k);
                        MA(i)(k) := t;
                    end if;
                end loop;
            end loop;
        end loop;
end Sort_Matrix;


-----Find MAX in Vector
function Vector_Max(A: in Vector) return Integer is
```

```
        r: Integer;
begin
        r := A'First;
        for i in 1..n loop
                if r < A(i) then
                        r := A(i);
                end if;
        end loop;
        return r;
end Vector_Max;


-----Print Vector on Screen
procedure Vector_Print(A: in Vector) is
begin
        for i in 1..n loop
                Put(A(i));
                Put(" ");
        end loop;
end Vector_Print;


-----Print Matrix on Screen
procedure Matrix_Print(MA: in Matrix) is
begin
        for i in 1..n loop
                for j in 1..n loop
                        Put(MA(i)(j));
                        Put(" ");
                end loop;
                New_Line;
        end loop;
end Matrix_Print;


-----Calculate F1: E = MAX(A)*(X+B*(MA*MD)+C)
function Func1(A,B,C,X: in Vector; MA,MD: in Matrix) return Vector is
        MR: Matrix;
        E,D: Vector;
        n: Integer;
begin
        MR := Mul_Matrix_Matrix(MA, MD);
        D := Mul_Vector_Matrix(B, MR);
        E := Add_Vectors(X, D);
        D := Add_Vectors(E, C);
```

```
        n := Vector_Max(A);
        E := Mul_Vector_Scalar(D, n);
        return E;
end Func1;


-----Calculate F2: ML = SORT(TRANS(MF)*MK)
function Func2(MF,MK: in out Matrix) return Matrix is
        ML: Matrix;
begin
        Transpose_Matrix(MF);
        ML := Mul_Matrix_Matrix(MF, MK);
        Sort_Matrix(ML);
        return ML;
end Func2;


-----Calculate F3: S = SORT(O*MO)*(MS*MT)
function Func3(O: in Vector; MO,MS,MT: in Matrix) return Vector is
        S, R: Vector;
        MR: Matrix;
begin
        R := Mul_Vector_Matrix(O, MO);
        MR := Mul_Matrix_Matrix(MS, MT);
        Sort_Vector(R);
        S := Mul_Vector_Matrix(R, MR);
        return S;
end Func3;

end Data;
```

data.ads

```
generic
        n: Integer;

package Data is

        -------Vector and Matrix types declaration
        type Vector is private;
        type Matrix is private;


        -----Read Vector from Input
        procedure Vector_Input(A: out Vector);
```

```
-----Read Matrix from Input
procedure Matrix_Input(MA: out Matrix);

-----Fill Vector with Number
procedure Vector_Fill(A: out Vector; b: Integer);

-----Fill Matrix with Number
procedure Matrix_Fill(MA: out Matrix; a: Integer);

-----Set Vector Element
procedure Vector_Set_Element(A: out Vector; i,b: Integer);

-----Set Matrix Element
procedure Matrix_Set_Element(MA: out Matrix; i,j,a: Integer);

-----Multiply Scalar and Vector
function Mul_Vector_Scalar(A: in Vector; b: Integer) return Vector;

-----Multiply Vector and Matrix
function Mul_Vector_Matrix(A: in Vector; MA: in Matrix) return Vector;

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

-----Add Two Vectors
function Add_Vectors(A,B: in Vector) return Vector;

-----Transpose Matrix
procedure Transpose_Matrix(MA: in out Matrix);

-----Sort Vector
procedure Sort_Vector(A: in out Vector);

-----Sort Matrix
procedure Sort_Matrix(MA: in out Matrix);

-----Find MAX in Vector
function Vector_Max(A: in Vector) return Integer;

-----Print Vector on Screen
procedure Vector_Print(A: in Vector);

-----Print Matrix on Screen
procedure Matrix_Print(MA: in Matrix);
```

```
        -----Calculate F1: E = MAX(A)*(X+B*(MA*MD)+C)
        function Func1(A,B,C,X: in Vector; MA,MD: in Matrix) return Vector;

        -----Calculate F2: ML = SORT(TRANS(MF)*MK)
        function Func2(MF,MK: in out Matrix) return Matrix;

        -----Calculate F3: S = SORT(O*MO)*(MS*MT)
        function Func3(O: in Vector; MO,MS,MT: in Matrix) return Vector;


        ------Definition of Vector and Matrix types
        private
        type Vector is array (1..n) of Integer;
        type Matrix is array (1..n) of Vector;

end Data;
```