НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА АВТОМАТИЗАЦІЇ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА УПРАВЛІННЯ

Лабораторна робота №2
з дісципліни «Паралельне програмування»

Виконав:
студент 3 курсу
ФІОТ гр. ІП-31
Кахерський О.І.

Перевірив:
Корочкін О. В.

Київ – 2015 р.

Цель работы: изучение средств языка Ада для работы с процессами.
Выполнение работы: Разработать программу, содержащую п а р а л л е л ь н ы е задачи, каждая из которых реализует функцию F1, F2, F3 из лабораторной работы номер 1. Использовать пакет Data из лабораторной работы 1 !

Задачи независимы, общих данных не имеют!

При создании задач необходимо:
- указать имя задачи
- задать приоритет задачи
- задать размер стека для задачи
- выбрать и задать номер процессора (ядра) для выполнения каждой задачи.

Первый и последний операторы тела задачи выводят на экран информацию о старте и завершении соответствующей задачи (“Task T1 started”, “ Task T2 finished”).
В теле задачи использовать оператор задержки delay, поставив его перед выполнением функции F1, F2, F3.

Исследовать при выполнении программы:
- влияние приоритетов задач на очередность запуска задач (при использовании одного или двух ядер).
- влияние оператора задержки delay на порядок выполнения задач.

5

- загрузку параллельной компьютерной системы (ПКС) (ядер процессора) при изменении их количества. Изменение количества ядер задается с помощью Менеджера (Диспетчера) задач ОС Windows.

Завдання

1.1	2.1	3.10

F1: A = SORT(B) * (MB * MC)
F2: MF = MK + ML * (MN * MM)
F3: Q = SORT(X + Z) * (MT * MS)

Лістинг коду

GNAT 4.6.4
Copyright 1992-2010, Free Software Foundation, Inc.

Compiling: /Data/Документы/FICT/ПРО/lab2/lab2.adb (source file time stamp: 2015-09-21 18:45:26)

```
 1. with Ada.Text_IO, Data, System.Multiprocessors;
                            |
   >>> warning: "System.Multiprocessors" is an Ada 2012 unit

 2. use Ada.Text_IO, System.Multiprocessors;
 3. ------------------------------------------------------------------
 4. --                                                              --
 5. --            Паралельні і розподілені обчислення              --
 6. --                  Лабораторна робота №2.                      --
 7. --                                                              --
 8. --  Файл: lab1.ada                                              --
 9. --  Завдання:                                                   --
10. --        F1: A = SORT(B) * (MB * MC)                          --
11. --        F2: MF = MK + ML * (MN * MM)                         --
12. --        F3: Q = SORT(X + Z) * (MT * MS)                      --
13. --                                                             --
14. --  Автор: Кахерський Олег, група ІП-31                         --
15. --  Дата: 12.09.2015                                            --
16. --                                                             --
17. ------------------------------------------------------------------
18. procedure Lab2 is
19.
20.   CPU_1: CPU_Range:=0;
21.   CPU_2: CPU_Range:=1;
22.   package Data3 is new Data(3);
23.   use Data3;
24.   --------------------------
25.   --                Tasks spec              --
26.   --------------------------
27.
28.   --task for F1
29.   task T1 is
30.            pragma Task_Name("Task1");
31.            pragma Priority(6);
32.            pragma Storage_Size(1000);
33.            pragma CPU (CPU_1);
34.   end T1;
35.
36.   --task for F1
37.   task T2 is
38.            pragma Task_Name("Task2");
39.            pragma Priority(5);
40.            pragma Storage_Size(1000);
41.            pragma CPU (CPU_1);
```

```
42. end T2;
43.
44. --task for F1
45. task T3 is
46.         pragma Task_Name("Task3");
47.         pragma Priority(4);
48.         pragma Storage_Size(1000);
49.         pragma CPU (CPU_1);
50. end T3;
51.
52. ---------------------------
53. --                Task bodies               --
54. ---------------------------
55. task body T1 is
56.         A,B:Vector;
57.         MB,MC:Matrix;
58. begin
59.         New_Line;
60.         Put ("Task1 is started");
61.
62.         Put ("Input vector B: ");
63.         Input_Vector (B);
64.         Output_Vector (B);
65.         New_Line;
66.
67.         Put ("Input matrix MB: ");
68.         Input_Matrix (MB);
69.         Output_Matrix (MB);
70.         New_Line;
71.
72.         Put ("Input matrix MC: ");
73.         Input_Matrix (MC);
74.         Output_Matrix (MC);
75.         New_Line;
76.
77.         --calculation
78.         F1 (B, MB, MC, A);
79.
80.         --out
81.         Put ("Function F1, vector A: ");
82.         Output_Vector (A);
83.         New_Line;
84.         Put ("Task T1 is finished");
85. end T1;
86.
87. task body T2 is
88.         MK,ML,MN,MM,MF:Matrix;
89. begin
90. --input
91. --delay(10.7);
92.         New_Line;
93.         Put ("Task2 is started");
```

```
94.
95.             Put ("Input matrix MK: ");
96.             Input_Matrix (MK);
97.      Output_Matrix (MK);
98.      New_Line;
99.
100.     Put ("Input matrix ML: ");
101.     Input_Matrix (ML);
102.     Output_Matrix (ML);
103.     New_Line;
104.
105.
106.     Put ("Input matrix MN: ");
107.     Input_Matrix (MN);
108.     Output_Matrix (MN);
109.     New_Line;
110.
111.
112.     Put ("Input matrix MM: ");
113.     Input_Matrix (MM);
114.     Output_Matrix (MM);
115.     New_Line;
116.
117.
118.
119.    --calc
120.    F2 (MK, ML, MN, MM, MF);
121.
122.    --out
123.    Put ("Function F2, matrix MF: ");
124.    Output_Matrix (MF);
125.    New_Line;
126.    Put ("Task2 is finished");
127.  end t2;
128.
129.  task body T3 is
130.             X,Z,Q : Vector;
131.             MT,MS: Matrix;
132.  begin
133.
134.  --delay(40.0);
135.             New_Line;
136.             Put ("Task3 is started");
137.    -- Input
138.    Put ("Input vector X: ");
139.    Input_Vector (X);
140.    Output_Vector (X);
141.    New_Line;
142.
143.    Put ("Input vector Z: ");
144.    Input_Vector (Z);
145.    Output_Vector (Z);
```

```
146.    New_Line;
147.
148.    Put ("Input matrix MT: ");
149.    Input_Matrix (MT);
150.    Output_Matrix (MT);
151.    New_Line;
152.
153.    Put ("Input matrix MS: ");
154.    Input_Matrix (MS);
155.    Output_Matrix (MS);
156.    New_Line;
157.
158.
159.    --calc
160.    F3 (X, Z, MT, MS, Q);
161.    --Output
162.    Put ("Function F3, vector Q: ");
163.    Output_Vector (Q);
164.    New_Line;
165.
166.    Put("Task3 is finished");
167.    end T3;
168. begin
169.  null;
170. end Lab2;
```

170 lines: No errors, 1 warning

GNAT 4.6.4

Compiling: /home/oleg/Документы/FICT/ППО/лр1/data.adb (source file time stamp: 2015-09-12 12:09:00)

```
 1. ------------------------------------------------------------------
 2. --  File: data.adb                                              --
 3. --  Author: Kakherskyi Oleh, group IP-31                        --
 4. --  Date: 12.09.2014                                            --
 5. ------------------------------------------------------------------
 6. with Ada.Text_IO, Ada.Integer_Text_IO, Ada.Numerics.Discrete_Random;
 7. use Ada.Text_IO, Ada.Integer_Text_IO;
 8.
 9. package body Data is
10.
11.    ------------------
12.    -- Input_Vector --
13.    ------------------
14.
15.    procedure Input_Vector (V : out Vector) is
16.    begin
17.       for I in Index loop
18.          V(I):=1;
```

```
   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
         MA(I)(J):=1;
      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;

```

```
71.  function "+" --Vector_Vector_Add
72.    (Left  : Vector;
73.     Right : Vector) return Vector;
74.
75.  function "+" --Matrix_Matrix_Add
76.    (Left: Matrix;
77.     Right: Matrix) return Matrix;
78.
79.
80.  procedure Sort (V : in Vector; A : out Vector);
81.
82.  ---------------------------------------
83.  -- F1: E = A + B + C + D * (MA * MZ) --
84.  ---------------------------------------
85.
86.  procedure F1
87.     (VB            : in Vector;
88.      MB, MC        : in Matrix;
89.      VA            : out Vector)
90.     is
91.     Temp : Vector;
92.     begin
93.        SORT(VB, Temp);
94.        VA := Temp * (MB * MC);
95.     end F1;
96.
97.  -----------------------------------
98.  -- F2: MF = MK + ML * (MN * MM)   --
99.  -----------------------------------
100.
101.  procedure F2
102.     (MK, ML, MN, MM : in Matrix;
103.      MF             : out Matrix)
104.   is
105.   begin
106.     MF := MK + ML * (MN * MM);
107.  end F2;
108.
109.  -----------------------------------------
110.  -- F3: Q = SORT(X + Z) * (MT * MS)      --
111.  -----------------------------------------
112.
113.  procedure F3
114.     (VX, VZ     : in Vector;
115.      MT, MS     : in Matrix;
116.      VQ         : out Vector) is
117.   Temp : Vector;
118.   begin
119.     Sort(VX+VZ, Temp);
120.     VQ := Temp * (MT * MS);
121.  end F3;
122.
```

```
123.    function "*"--Matrix_Matrix_Multiply
124.      (Left  :  Matrix;
125.       Right :  Matrix) return Matrix
126.    is
127.   MR : Matrix;
128.         begin
129.           for i in Index loop
130.             for J in Index loop
131.               MR(I)(J) := 0;
132.               for K in Index loop
133.                 MR(I)(J) := MR(I)(J) + Left(I)(K) * Right(K)(J);
134.               end loop;
135.             end loop;
136.       end loop;
137.       return MR;
138.    end  "*";
139.
140.
141.    function "*"--Vector_Matrix_Multiply
142.      (Left  : Vector;
143.       Right : Matrix) return Vector
144.    is
145.       R : Vector;
146.    begin
147.       for J in Index loop
148.         R(j) := 0;
149.         begin
150.           for K in Index loop
151.             R(J) := R(J) + Left(K) * Right(K )(J);
152.           end loop;
153.         end;
154.       end loop;
155.       return R;
156.    end "*";
157.
158.    function "+"--Vector_Vector_Add
159.      (Left  : Vector;
160.       Right : Vector) return Vector
161.    is
162.       R : Vector;
163.    begin
164.       for J in Index loop
165.         R (J) :=  Left (J) + Right (J);
166.       end loop;
167.       return R;
168.    end "+";
169.
170.    function "+" --Matrix_Matrix_Add
171.      (Left: Matrix;
172.       Right: Matrix) return Matrix
173.    is
174.       R: Matrix;
```

```
175.    begin
176.       for I in Index loop
177.          for J in Index loop
178.             R(I)(J) := Left(I)(J) + Right(I)(J);
179.          end loop;
180.       end loop;
181.       return R;
182.    end "+";
183.
184.    procedure Sort (V : in Vector; A : out Vector) is
185.       Min  : Positive;
186.       Temp : Integer;
187.    begin
188.       A := V;
189.       for I in A'First..A'Last - 1 loop
190.          Min := I;
191.          for J in I + 1..A'Last loop
192.             if A (Min) > A (J) then
193.                Min := J;
194.             end if;
195.          end loop;
196.          if Min /= I then
197.             Temp    := A (I);
198.             A (I)   := A (Min);
199.             A (Min) := Temp;
200.          end if;
201.       end loop;
202.    end Sort;
203.
204. end Data;
```

Compiling: /home/oleg/Документы/FICT/ПРО/лр1/data.ads (source file time stamp: 2015-09-12 12:47:50)

```
1. ------------------------------------------------------------------
2. --  File: data.ads                                           --
3. --                                                  --
4. --  Author: Oleh Kakherskyi, group IP-31                     --
5. --  Date: 12.09.2012                                    --
6. ------------------------------------------------------------------
7.
8. generic
9.    N: in Natural; -- dimension of Vector and Matrix(N * N)
10. package Data is
11.
12.    type Vector is private;
13.    type Matrix is private;
14.
15.    ---------------------------------------
16.    -- F1: A = SORT(B) * (MB * MC)       --
17.    ---------------------------------------
18.
```

```
19.   procedure F1
20.      (VB            : in Vector;
21.       MB, MC        : in Matrix;
22.       VA            : out Vector);
23.
24.   ------------------------------------
25.   -- F2: MF = MK + ML * (MN * MM)   --
26.   ------------------------------------
27.
28.   procedure F2
29.      (MK, ML, MN, MM : in Matrix;
30.       MF             : out Matrix);
31.
32.   ----------------------------------------
33.   -- F3: Q = SORT(X + Z) * (MT * MS)      --
34.   ----------------------------------------
35.
36.   procedure F3
37.      (VX, VZ     : in Vector;
38.       MT, MS     : in Matrix;
39.       VQ         : out Vector);
40.
41.   ------------------
42.   -- Input_Vector --
43.   ------------------
44.
45.   procedure Input_Vector(V : out Vector);
46.
47.   ------------------
48.   -- Input_Matrix --
49.   ------------------
50.
51.   procedure Input_Matrix(MA : out Matrix);
52.
53.   -------------------
54.   -- Output_Vector --
55.   -------------------
56.
57.   procedure Output_Vector(V : in Vector);
58.
59.   -------------------
60.   -- Output_Matrix --
61.   -------------------
62.
63.   procedure Output_Matrix(MA : in Matrix);
64.   private
65.
66.      subtype Index is Integer range 1..N;
67.      type Vector is array (Index) of Integer;
68.      type Matrix is array (Index) of Vector;
69.
70. end Data;
```

204 lines: No errors