Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №2

Ада. Задачі

Виконав:

студент групи ІП-32

Ковальчук О. М.

Перевірив:

Корочкін О. В.

Київ - 2015

Лабораторна робота №2. АДА. Задачі

Мета роботи: вивчення засобів мови Ада для роботи з задачами

Мова програмування: Ада

Завдання: розробити програму мовою Ада, яка містить паралельні задачі, кожна з яких виконує обчислення трьох функцій F1, F2, F3 із лабораторної роботи №1. Програма повинна використовувати пакет Data із попередньої лабораторної роботи.

Функції:

F1: C = A - B * (MA * MD)

F2: o = Min(MK * MM)

F3: T = (MS * MZ) * (W + X)

Лістинг програми

```
GNAT 4.5.4
Copyright 1992-2010, Free Software Foundation, Inc.


Compiling: lab2.adb (source file time stamp: 2015-10-01 05:51:43)

     1. -------------------------------------
     2. --      Parallel programming      --
     3. --              Lab 2             --
     4. --                                --
     5. -- Func1: C = A - B * (MA * MD)   --
     6. -- Func2: o = Min(MK * MM)        --
     7. -- Func3: T = (MS * MZ) * (W + X) --
     8. --                                --
     9. -- @author Olexandr Kovalchuk     --
    10. -- @group IP 32                   --
    11. --                                --
    12. -- @date 2015-09-27               --
    13. -------------------------------------
    14. with Data;
    15. with Ada.Text_IO; use Ada.Text_IO;
    16. with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
    17.
    18. procedure Lab2 is
    19.   SIZE: constant Integer := 1000;
    20.   package DataBase is new Data(SIZE => SIZE); use DataBase;
    21.
    22.   procedure RunTasks is
    23.
    24.     task T1 is
    25.       pragma Storage_Size(32*SIZE*(SIZE + 3));
    26.       pragma Priority(7);
    27.       pragma CPU(1);
                     |
        >>> warning: unrecognized pragma "Cpu"

    28.     end T1;
    29.     task body T1 is
    30.       MA, MD: Matrix;
    31.       A, B, C: Vector;
    32.     begin
```

```
33.       Put_Line("Task T1 started");
34.       GetMatrix(MA);
35.       GetMatrix(MD);
36.       GetVector(A);
37.       GetVector(B);
38.       delay 0.25;
39.       C := Func1(A, B, MA, MD);
40.       Put("C = A - B * (MA * MD) = ("); Put(C); Put_Line(")");
41.       Put_Line("Task T1 finished");
42.     end T1;
43.
44.     task T2 is
45.       pragma Storage_Size(32*SIZE*SIZE);
46.       pragma Priority(4);
47.       pragma CPU(2);
                |
    >>> warning: unrecognized pragma "Cpu"

48.     end T2;
49.     task body T2 is
50.       MK, MM: Matrix;
51.       o: Integer;
52.     begin
53.       Put_Line("Task T2 started");
54.       GetMatrix(MK);
55.       GetMatrix(MM);
56.       o := Func2(MK, MM);
57.       delay 0.000001;
58.       Put("o = Min(MK * MM) = "); Put(o, 2); New_Line;
59.       Put_Line("Task T2 finished");
60.     end T2;
61.
62.     task T3 is
63.       pragma Storage_Size(32*SIZE*(SIZE + 2));
64.       pragma Priority(6);
65.       pragma CPU(3);
                |
    >>> warning: unrecognized pragma "Cpu"

66.     end T3;
67.     task body T3 is
68.       MS, MZ: Matrix;
69.       W, X, T: Vector;
70.     begin
71.       Put_Line("Task T3 started");
72.       GetMatrix(MS);
73.       GetMatrix(MZ);
74.       GetVector(W);
75.       GetVector(X);
76.       delay 0.5;
77.       T := Func3(MS, MZ, W, X);
78.       Put("T = (MS * MZ) * (W + X) = ("); Put(T); Put_Line(")");
79.       Put_Line("Task T3 finished");
80.     end T3;
81.   begin
82.     null;
83.   end RunTasks;
84. begin
85.   Put_Line("Lab2 started");
86.   RunTasks;
87.   Put_Line("Lab2 finished");
88. end Lab2;
89.

89 lines: No errors, 3 warnings
```

```
GNAT 4.5.4
Copyright 1992-2010, Free Software Foundation, Inc.


Compiling: data.adb (source file time stamp: 2015-10-01 05:26:56)

     1. with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
     2. with Ada.Text_IO; use Ada.Text_IO;
     3.
     4. package body Data is
     5.
     6.   procedure Fill(filler: in Integer; mtrx: out Matrix) is
     7.   begin
     8.     for r in mtrx'range(1) loop
     9.       for c in mtrx'range(2) loop
    10.         mtrx(r, c) := filler;
    11.       end loop;
    12.     end loop;
    13.   end;
    14.
    15.   procedure Fill(filler: in Integer; vec: out Vector) is
    16.   begin
    17.     for i in vec'range(1) loop
    18.       vec(i) := filler;
    19.     end loop;
    20.   end;
    21.
    22.   function "*"(left, right : in Matrix) return Matrix is
    23.     temp : Matrix;
    24.   begin
    25.     Fill(0, temp);
    26.     for i in left'range(1) loop
    27.       for j in right'range(2) loop
    28.         for k in left'range(2) loop
    29.           temp(i, j) := temp(i, j) + left(i, k) * right(k, j);
    30.         end loop;
    31.       end loop;
    32.     end loop;
    33.     return temp;
    34.   end "*";
    35.
    36.   function "*"(left : in Vector; right : in Matrix) return Vector is
    37.     temp : Vector;
    38.     helper : Matrix;
    39.   begin
    40.     Fill(0, helper);
    41.     for i in left'range(1) loop
    42.       helper(1, i) := left(i);
    43.     end loop;
    44.     helper := helper * right;
    45.     for i in temp'range(1) loop
    46.       temp(i) := helper(1, i);
    47.     end loop;
    48.     return temp;
    49.   end "*";
    50.
    51.   function "*"(left : in Matrix; right : in Vector) return Vector is
    52.     temp : Vector;
    53.     helper : Matrix;
    54.   begin
    55.     Fill(0, helper);
    56.     for i in right'range(1) loop
    57.       helper(i, 1) := right(i);
    58.     end loop;
    59.     helper := left * helper;
```

```
    for i in temp'range(1) loop
      temp(i) := helper(i, 1);
    end loop;
    return temp;
  end "*";

  function "+"(left, right : in Vector) return Vector is
    temp: Vector;
  begin
    for i in temp'range(1) loop
      temp(i) := left(i) + right(i);
    end loop;
    return temp;
  end;

  function "-"(left, right : in Vector) return Vector is
    temp: Vector;
  begin
    for i in temp'range(1) loop
      temp(i) := left(i) - right(i);
    end loop;
    return temp;
  end;


  function Min(mtrx: in Matrix) return Integer is
    min : Integer;
  begin
    min := mtrx(1, 1);
    for row in mtrx'range(1) loop
      for col in mtrx'range(2) loop
        if min > mtrx(row, col) then
          min := mtrx(row, col);
        end if;
      end loop;
    end loop;
    return min;
  end;

  function Func1(A, B: in Vector; MA, MD: in Matrix) return Vector is
  begin
    return A - B * (MA * MD);
  end;

  function Func2(MK, MM: in Matrix) return Integer is
  begin
    return Min(MK * MM);
  end;

  function Func3(MS, MZ: in Matrix; W, X: in Vector) return Vector is
  begin
    return (MS * MZ) * (W + X);
  end;

  procedure GetMatrix(mtrx: out Matrix) is
  begin
    Fill(1, mtrx);
  end;

  procedure GetVector(vec: out Vector) is
  begin
    Fill(1, vec);
  end;

  procedure Put(value: in Matrix) is
  begin
```

```
126.     for row in value'range(1) loop
127.       for col in value'range(2) loop
128.         Put(value(row, col), 4);
129.         Put(',');
130.       end loop;
131.       New_Line;
132.     end loop;
133.   end;
134.
135.   procedure Put(value: in Vector) is
136.   begin
137.     for i in value'range(1) loop
138.       Put(value(i), 4);
139.       Put(',');
140.     end loop;
141.   end;
142.
143. end Data;
144.
```

```
Compiling: data.ads (source file time stamp: 2015-09-27 09:59:07)

 1. generic
 2.   SIZE : in Natural := 2;
 3. package Data is
 4.   type Vector is private;
 5.   type Matrix is private;
 6.
 7.   function Func1(A, B: in Vector; MA, MD: in Matrix) return Vector;
 8.   function Func2(MK, MM: in Matrix) return Integer;
 9.   function Func3(MS, MZ: in Matrix; W, X: in Vector) return Vector;
10.
11.   procedure GetMatrix(mtrx: out Matrix);
12.   procedure GetVector(vec: out Vector);
13.
14.   procedure Put(value: in Matrix);
15.   procedure Put(value: in Vector);
16.
17.   private
18.     type Vector is array (1 .. SIZE) of Integer;
19.     type Matrix is array (1 .. SIZE, 1 .. SIZE) of Integer;
20. end Data;
21.

144 lines: No errors
```