```
GNAT GPL 2015 (20150428-49)
Copyright 1992-2015, Free Software Foundation, Inc.

Compiling: lab1.adb
Source file time stamp: 2016-02-25 07:48:24
Compiled at: 2016-02-25 09:48:33

     1. ------------------------------------------------------------
     2. ------------------PARALLEL PROGRAMMING--------------------
     3. ------------------------LAB #1------------------------------
     4. ----       ----------ADA. SEMAPHORES----------       -----
     5. ----              MA = MB*MC + MO*ME*a              -----
     6. -----------------CREATED ON 24.02.2016--------------------
     7. ----------------BY OLEG PEDORENKO, IP-31------------------
     8. ------------------------------------------------------------
     9.
    10. with Ada.Text_IO; use Ada.Text_IO;
    11. with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
    12. with Ada.Synchronous_Task_Control; use Ada.Synchronous_Task_Control;
    13. with Ada.Float_Text_IO;     use Ada.Float_Text_IO;
    14. with Data;
    15.
    16. procedure Lab1 is
    17.     -- Константы
    18.     N: Integer := 8;
    19.     P: Integer := 2;
    20.     H: Integer := N/P;
    21.
    22.     --Типы
    23.     package Data_S is new Data(N, 100); use Data_S;
    24.
    25.     --Переменные
    26.     MA, MB, MC, MO, ME: Matrix;
    27.     MX, MY: Matrix;
    28.     a: Float;
    29.
    30.     --Semaphores
    31.     Sem1, Sem2, Sem3, Sem4: Suspension_Object;
    32.
    33.     procedure Start_Tasks is
    34.
    35.         task T1 is
    36.             pragma Storage_Size(1000000000);
    37.         end T1;
    38.
    39.         task body T1 is
    40.             Sum1: Float;
    41.         begin
    42.             Put_Line("Task 1 started");
    43.             --Enter MB, MC, MO, ME, a           V
    44.             Put_Line("MA =   "); Input(MA);
    45.             Put_Line("MB =   "); Input(MB);
    46.             Put_Line("MC =   "); Input(MC);
    47.             Put_Line("MO =   "); Input(MO);
    48.             Put_Line("ME =   "); Input(ME);
    49.             Input(MX);
    50.             a := 1.0;
    51.             --Signal S2,1---------------------------
    52.             Set_True(Sem1);
    53.
    54.             --Calculate MX = MB*MC
    55.             for I in 1..N loop
```



```
    56.                 for J in 1..N loop
    57.                     Sum1 := 0.0;
    58.                     for K in 1..N loop
    59.                         Sum1 := Sum1 + (MB(I)(K) * MC(K)(J)); -- MB * MC
    60.                     end loop;
    61.                     MX(I)(J) := Sum1;
    62.                 end loop;
    63.             end loop;
    64.             --Signal S2,2
    65.             Set_True(Sem2);
    66.             --Wait W2,1---------------------------
    67.             Suspend_Until_True(Sem3);
    68.             --Calculate MAh = MXh + MYh
    69.             for I in 1..H loop
    70.                 for J in 1..N loop
    71.                     MA(I)(J) := MX(I)(J) + MY(I)(J);
    72.                 end loop;
    73.             end loop;
    74.             --Wait W2,2
    75.             Suspend_Until_True(Sem4);
    76.             --Output of MA
    77.             Put_Line("MA =   "); Output(MA);
    78.
    79.             Put_Line("Task 1 Finished");
    80.             --
    81.         end T1;
    82.
    83.         task T2 is
    84.             pragma Storage_Size(1000000000);
    85.         end T2;
    86.
    87.         task body T2 is
    88.             Sum1: Float;
    89.         begin
    90.             Put_Line("Task 2 started");
    91.             --Wait W1,1---------------------------
    92.             Suspend_Until_True(Sem1);
    93.             --MY = MO*ME*a
    94.             for I in 1..N loop
    95.                 for J in 1..N loop
    96.                     Sum1 := 0.0;
    97.                     for K in 1..N loop
    98.                         Sum1 := Sum1 + (MO(I)(K) * ME(K)(J)); -- MO * ME
    99.                     end loop;
   100.                     MY(I)(J) := Sum1 * a;
   101.                 end loop;
   102.             end loop;
   103.             --Signal S1,1---------------------------
   104.             Set_True(Sem3);
   105.             --Wait W1,2
   106.             Suspend_Until_True(Sem2);
   107.             --Calculate MAh = MXh + MYh
   108.             for I in (H+1)..N loop
   109.                 for J in 1..N loop
   110.                     MA(I)(J) := MX(I)(J) + MY(I)(J);
   111.                 end loop;
   112.             end loop;
   113.             --Signal S1,2
   114.             Set_True(Sem4);
   115.             Put_Line("Task 2 finished");
   116.         end T2;
   117.
```

```
   118.     begin
   119.         null;
   120.     end Start_Tasks;
   121.
   122.
   123. begin
   124.     Start_Tasks;
   125. end Lab1;

 125 lines: No errors

 Compiling: data.ads
Source file time stamp: 2016-02-25 05:42:12
Compiled at: 2016-02-25 09:48:34

     1. generic
     2.     Size: Integer;
     3.     Random_Max: Integer;
     4. package Data is
     5.     subtype Range_T is Integer range 1 .. Size;
     6.     type Vector is array(Range_T) of Float;
     7.     type Matrix is array(Range_T) of Vector;
     8.
     9.     procedure Input(A: out Vector);
    10.     procedure Input(A: out Matrix);
    11.
    12.     procedure Output(A: in Vector);
    13.     procedure Output(A: in Matrix);
    14.     procedure Output(A: in Float);
    15.
    16. private
    17.
    18.
    19. end Data;

 110 lines: No errors

Compiling: data.adb
Source file time stamp: 2016-02-25 07:25:28
Compiled at: 2016-02-25 09:48:34

     1. with Ada.Text_IO;           use Ada.Text_IO;
     2. with Ada.Integer_Text_IO;   use Ada.Integer_Text_IO;
     3. with Ada.Float_Text_IO;     use Ada.Float_Text_IO;
     4. with Ada.Containers.Generic_Constrained_Array_Sort;
     5. with Ada.Numerics.Float_Random;
     6. with Ada.Numerics.Discrete_Random;
     7.
     8. package body Data is
     9.
    10.     maxOutputSize: Integer := 8;
    11.
    12.     procedure Get_Random(A: out Float) is
    13.         use Ada.Numerics.Float_Random;
    14.         subtype R is Integer range 1 .. Random_Max;
    15.         package Rand_Int is new Ada.Numerics.Discrete_Random (R);
    16.         G1: Generator;
    17.         G2: Rand_Int.Generator;
    18.     begin
    19.         Reset(G1); Rand_Int.Reset(G2);
    20.         A := Random(G1) * Float(Rand_Int.Random(G2));
    21.     end Get_Random;
```



```
    22.
    23.     function Cmp(Left: Float; Right: Float) return Boolean;
    24.     procedure Desc_Sort is
    25.         new Ada.Containers.
    26.             Generic_Constrained_Array_Sort(Range_T, Float, Vector, Cmp);
    27.
    28.     procedure Vector_Input(A: out Vector);
    29.     procedure Vector_Output(A: in Vector);
    30.     procedure Matrix_Input(A: out Matrix);
    31.     procedure Matrix_Output(A: in Matrix);
    32.
    33.     procedure Vector_Input(A: out Vector) is
    34.     begin
    35.         if Size <= maxOutputSize then
    36.             for I in Range_T loop
    37.                 --Get(A(I));
    38.                 A(I) := 1.0;
    39.             end loop;
    40.         else
    41.             for I in Range_T loop
    42.                 --Get_Random(A(I));
    43.                 A(I) := 1.0;
    44.             end loop;
    45.         end if;
    46.     end Vector_Input;
    47.
    48.     procedure Vector_Output(A: in Vector) is
    49.     begin
    50.         for I in Range_T loop
    51.             Output(A(I));
    52.         end loop;
    53.         New_Line;
    54.     end Vector_Output;
    55.
    56.     procedure Matrix_Input(A: out Matrix) is
    57.     begin
    58.         for I in Range_T loop
    59.             Vector_Input(A(I));
    60.         end loop;
    61.     end Matrix_Input;
    62.
    63.     procedure Matrix_Output(A: in Matrix) is
    64.     begin
    65.         for I in Range_T loop
    66.             Vector_Output(A(I));
    67.         end loop;
    68.     end Matrix_Output;
    69.
    70.     procedure Input(A: out Vector) is
    71.     begin
    72.         Vector_Input(A);
    73.         if Size <= maxOutputSize then
    74.             Vector_Output(A);
    75.         end if;
    76.     end Input;
    77.
    78.     procedure Input(A: out Matrix) is
    79.     begin
    80.         Matrix_Input(A);
    81.         if Size <= maxOutputSize then
    82.             Matrix_Output(A);
    83.         end if;
```

```
    84.     end Input;
    85.
    86.     procedure Output(A: in Vector) is
    87.     begin
    88.         if Size <= maxOutputSize then
    89.             Vector_Output(A);
    90.         end if;
    91.     end Output;
    92.
    93.     procedure Output(A: in Matrix) is
    94.     begin
    95.         if Size <= maxOutputSize then
    96.             Matrix_Output(A);
    97.         end if;
    98.     end Output;
    99.
   100.     procedure Output(A: in Float) is
   101.     begin
   102.         Put(A, 5, 2, 0);
   103.     end Output;
   104.
   105.     function Cmp(Left: Float; Right: Float) return Boolean is
   106.     begin
   107.         return (Left > Right);
   108.     end Cmp;
   109.
   110. end Data;
```