Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

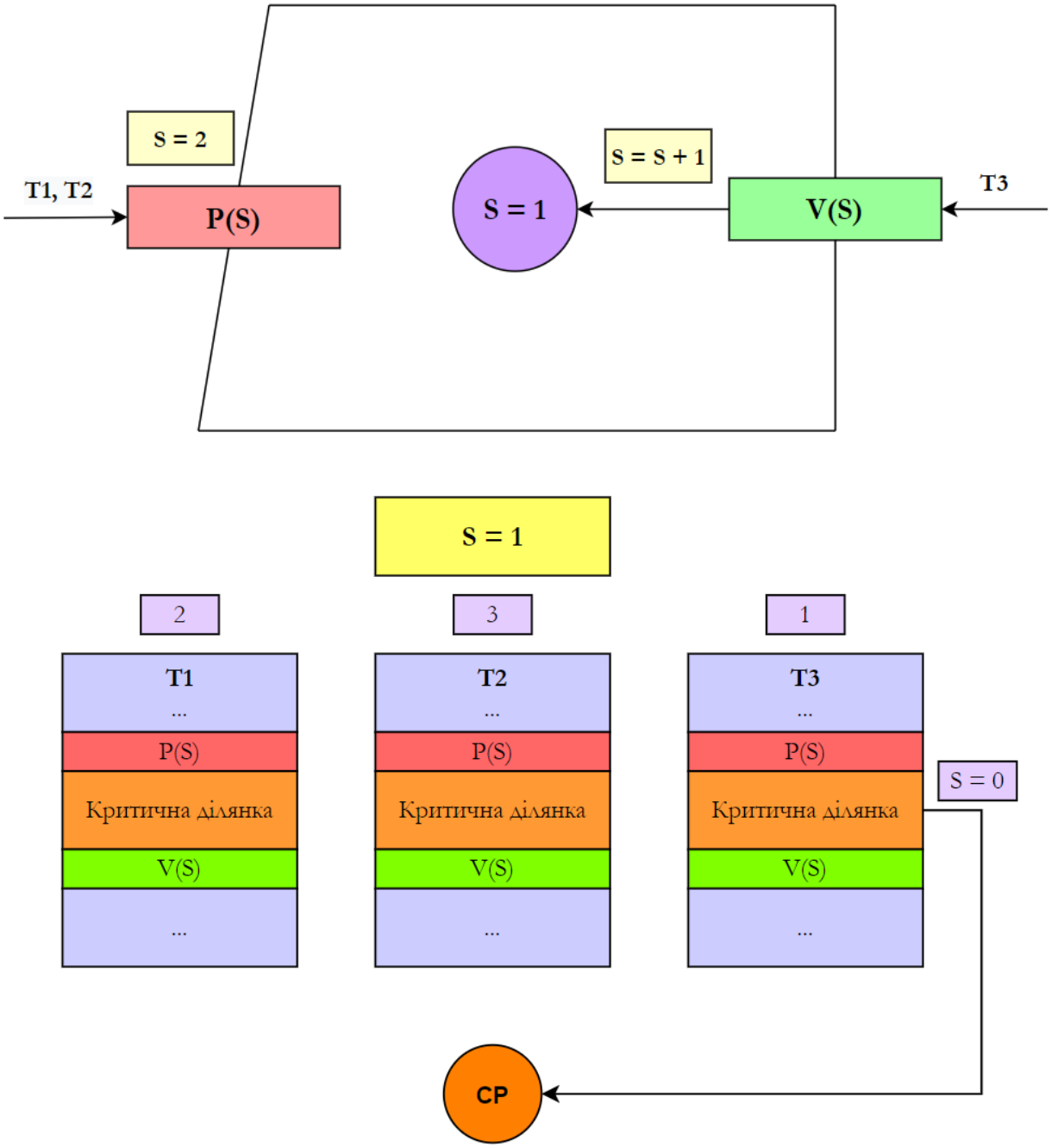
САМОСТІЙНА РОБОТА
З ДИСЦИПЛІНИ “ОСНОВИ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ ”
НА ТЕМУ: “СЕМАФОРИ В МОВІ ADA”

Виконав:
Студент III курсу ФІОТ
групи ІО-82
Шендріков Євгеній
Номер у списку - 25

Перевірив:
Доцент
Корочкін О. В.

м. Київ – 2020 р.

Структура програми



Маємо потоки T1, T2, T3, а також спільний ресурс (далі – СР). Далі ми створюємо множинний семафор і встановлюємо для нього значення, наприклад, (1, 2), тобто початкове значення 1 і максимальне значення 2. Перед критичною ділянкою в кожному потоці ми розміщуємо операцію P(S), а після критичної ділянки – V(S).

На початку виконання спільний ресурс вільний, ми встановили максимальне і початкове значення семафора як 2 та 1, тому перший потік, який звернеться до СР отримає можливість його використовувати і виконувати дії прописані в критичних ділянках (наприклад, введення, обчислення або виведення даних в консоль), інші потоки, які запізнилися, будуть блоковані і

чекатимуть доки СР звільниться, тобто коли якийсь з потоків, який працює зі спільним ресурсом, завершить його використання і звільнить СР.

В даній ситуації, першим до критичної ділянки дійде потік Т3, а Т1 та Т2 останнім звернуться до критичної ділянки. Виходячи з цього, Т3 першим звертається до СР, для цього він обов'язково виконує операцію P(S), яка перевіряє значення семафора, у нас це значення рівне 1, тобто не рівне 0, значення семафора зменшується на одиницю (S = 0), а потік заходить в свою критичну ділянку.

Тепер маємо ситуацію, що Т1 і Т2 працюють ще на початку програми, а Т3 знаходиться в своїй критичній ділянці і семафор на цей момент дорівнює 0. Т1 був наступний в черзі, тому він звертається до СР, виконує операцію P(S), операція бачить, що значення семафора 0 і блокує цей потік. Так само з Т2. Розблокування потоку Т1 відбудеться коли потік Т3 закінчить роботу в критичній ділянці і виконає операцію V(S), ця операція додає одиницю до семафора, операція P(S) цю одиницю побачить і розблокує Т1, після цього Т1 входить в критичну ділянку, значення семафору буде знову 0, а коли Т1 завершить своє виконання в критичній ділянці семафор отримає значення 1 і наступний потік, Т2, зможе використовувати СР.

Лістинг програми

semaphores.adb

```
1. package body Semaphores is
2.
3.    protected body SIMA is
4.
5.       entry P when Count > 0 is
6.          begin
7.          Count := Count - 1;
8.        end P;
9.
10.       entry V when Count < Max_Count is
11.         begin
12.         Count := Count + 1;
13.       end V;
14.
15.    end SIMA;
16.
17. end Semaphores;
```

semaphores.ads

```
1. package Semaphores is
2.
3.     protected type SIMA (Initial_Value: Natural; Max_Value: Natural) is
4.       entry P;
5.       entry V;
6.    private
7.       Count : Natural := Initial_Value;
8.       Max_Count : Natural := Max_Value;
9.    end SIMA;
10.
11. end Semaphores;
```

Main.adb

```
1. ------------------------------------------------------
2. --|                   Semaphores                     |
3. ------------------------------------------------------
4. --|   Author   |    Jack (Yevhenii) Shendrikov       |
5. --|   Group    |                IO-82                |
6. --|   Variant  |                 #25                 |
7. --|    Date    |             29.11.2020              |
8. ------------------------------------------------------
9. --| Function 1 | D = SORT(A)+SORT(B)+SORT(C)*(MA*ME) |
10. --| Function 2 |      MF = (MG*MH)*TRANS(MK)         |
11. --| Function 3 |     S = (MO*MP)*V+t*MR*(O+P)        |
12. ------------------------------------------------------
13.
14. with Data;
15. with Ada.Integer_Text_IO, Ada.Text_IO, Ada.Characters.Latin_1;
16. use Ada.Integer_Text_IO, Ada.Text_IO, Ada.Characters.Latin_1;
17. with System.Multiprocessors; use System.Multiprocessors;
18.
19. with Semaphores;
20. use Semaphores;
21.
22. procedure Main is
23.    N: Integer;
24.
25.    MY_SIMA : SIMA (1,2);
26.
27.    procedure Tasks is
28.       package My_Data is new Data(N);
29.       use My_Data;
30.
31.       CPU_0: CPU_Range := 0;
32.       CPU_1: CPU_Range := 1;
33.       CPU_2: CPU_Range := 2;
34.
35.
36.       task T1 is
37.          pragma Task_Name("T1");
38.          pragma Priority(4);
39.          pragma Storage_Size(500000000);
40.          pragma CPU (CPU_0);
41.       end T1;
42.
43.       task T2 is
44.          pragma Task_Name("T2");
45.          pragma Priority(3);
46.          pragma Storage_Size(500000000);
47.          pragma CPU (CPU_1);
48.       end T2;
49.
50.       task T3 is
51.          pragma Task_Name("T3");
52.          pragma Priority(7);
53.          pragma Storage_Size(500000000);
54.          pragma CPU (CPU_2);
55.       end T3;
56.
57.
58.       task body T1 is
59.          A,B,C,D: Vector;
60.          MA,ME: Matrix;
61.       begin
62.          Put_Line("Task T1 started");
63.          delay 0.5;
64.          Put_Line("T1 is waiting for a permit.");
65.
66.          -- Generate Input Values
```

```
67.          MY_SIMA.P; -- Acquire the semaphore
68.          New_Line; Put_Line("T1 gets a permit.");
69.          delay 1.0;
70.          Input_Val_F1(A,B,C,MA,ME);
71.          Put_Line("T1 releases the permit.");
72.          MY_SIMA.V; -- Release  the semaphore
73.          New_Line; Put_Line("T1 is waiting for a permit.");
74.
75.          -- Calculate The Result
76.          D := Func1(A,B,C,MA,ME);
77.          delay 1.0;
78.
79.          -- Output
80.          MY_SIMA.P; -- Acquire the semaphore
81.          Put_Line("T1 gets a permit.");
82.          Put("T1 | ");
83.          Vector_Output(D, "D");
84.          Put_Line("T1 releases the permit."); New_Line;
85.          MY_SIMA.V; -- Release  the semaphore
86.
87.          Put_Line("Task T1 finished"); New_Line;
88.       end T1;
89.
90.       task body T2 is
91.          MG,MH,MK,MF: Matrix;
92.       begin
93.          Put_Line("Task T2 started");
94.          delay 0.5;
95.          Put_Line("T2 is waiting for a permit.");
96.
97.          -- Generate Input Values
98.          MY_SIMA.P; -- Acquire the semaphore
99.          New_Line; Put_Line("T2 gets a permit.");
100.          delay 1.0;
101.          Input_Val_F2(MG,MH,MK);
102.          Put_Line("T2 releases the permit."); New_Line;
103.          MY_SIMA.V; -- Release  the semaphore
104.          New_Line; Put_Line("T2 is waiting for a permit.");
105.
106.          -- Calculate The Result
107.          MF := Func2(MG,MH,MK);
108.          delay 1.0;
109.
110.          -- Output
111.          MY_SIMA.P; -- Acquire the semaphore
112.          Put_Line("T2 gets a permit.");
113.          Put_Line("T2 | ");
114.          Matrix_Output(MF, "MF");
115.          Put_Line("T2 releases the permit.");
116.          MY_SIMA.V; -- Release  the semaphore
117.
118.          Put_Line("Task T2 finished"); New_Line;
119.       end T2;
120.
121.       task body T3 is
122.          t: Integer;
123.          V,O,P,S: Vector;
124.          MO,MP,MR: Matrix;
125.       begin
126.          Put_Line("Task T3 started");
127.          delay 0.5;
128.          Put_Line("T3 is waiting for a permit.");
129.
130.          -- Generate Input Values
131.          MY_SIMA.P; -- Acquire the semaphore
132.          New_Line; Put_Line("T3 gets a permit.");
133.          delay 1.0;
134.          Input_Val_F3(t,V,O,P,MO,MP,MR);
```

```
135.          Put_Line("T3 releases the permit.");
136.          MY_SIMA.V; -- Release  the semaphore
137.          New_Line; Put_Line("T3 is waiting for a permit.");
138.
139.          -- Calculate The Result
140.          S := Func3(t,V,O,P,MO,MP,MR);
141.          delay 1.0;
142.
143.          -- Output
144.          MY_SIMA.P; -- Acquire the semaphore
145.          Put_Line("T3 gets a permit.");
146.          Put("T3 | ");
147.          Vector_Output(S, "S");
148.          Put_Line("T3 releases the permit."); New_Line;
149.          MY_SIMA.V; -- Release  the semaphore
150.
151.          Put_Line("Task T3 finished"); New_Line;
152.       end T3;
153.
154.
155.    begin
156.       Put_Line("Calculations started");
157.       New_Line;
158.    end Tasks;
159.
160. begin
161.    Put_Line("Function 1: D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)" & CR & LF
162.           & "Function 2: MF = (MG*MH)*TRANS(MK)" & CR & LF
163.           & "Function 3: S = (MO*MP)*V+t*MR*(O+P)" & CR & LF);
164.
165.    Put_Line("!!! Note that if the value of N > 10 -> the result will not be displayed
!!!" & CR & LF
166.           & "!!! If you enter N <= 0 - execution will be terminated !!!"  & CR & LF);
167.    Put("Enter N: ");
168.    Get(N);
169.    New_Line;
170.
171.    Tasks;
172. end Main;
```

data.abs

```
1. generic
2.    N: Integer;
3. package Data is
4.    type Vector is private;
5.    type Matrix is private;
6.
7.    --Procedures, that fills random values into Vectors and Matrixes
8.    procedure Random_Vector (A: out Vector);
9.    procedure Random_Matrix (MA: out Matrix);
10.
11.    --Procedures, that fills Matrixes and Vectors with 1:
12.    procedure All_Ones_Vector(A: out Vector);
13.    procedure All_Ones_Matrix(MA: out Matrix);
14.
15.    --Procedures, that shows values of Vectors and Matrixes in the screen
16.    procedure Vector_Output (A: in Vector; str: in String);
17.    procedure Matrix_Output (MA: in Matrix; str: in String);
18.
19.     --Procedures, that fills a values into Vectors and Matrixes
20.    procedure Vector_Input (A: out Vector; str: in String);
21.    procedure Matrix_Input (MA: out Matrix; str: in String);
22.
23.
24.    -- Procedures, that generates initial values for each Function
25.    -- Depending on the option chosen by the user (val)
26.    procedure Input_Val_F1 (A,B,C: out Vector; MA,ME: out Matrix);
```

```
27.    procedure Input_Val_F2 (MG,MH,MK: out Matrix);
28.    procedure Input_Val_F3 (t: out Integer; V,O,P: out Vector; MO,MP,MR: out Matrix);
29.
30.    --Function 1 (SORT(A)+SORT(B)+SORT(C)*(MA*ME))
31.    function Func1 (A,B,C: out Vector; MA,ME: in Matrix) return Vector;
32.
33.    --Function 2 ((MG*MH)*TRANS(MK))
34.    function Func2 (MG,MH,MK: in Matrix) return Matrix;
35.
36.    --Function 3 ((MO*MP)*V+t*MR*(O+P))
37.    function Func3 (t: in Integer; V, O, P: in Vector; MO, MP, MR: in Matrix) return
Vector;
38.
39.    private
40.    type Vector is array (1..N) of Integer;
41.    type Matrix is array (1..N, 1..N) of Integer;
42.
43. end Data;
44. generic
45.    N: Integer;
46. package Data is
47.    type Vector is private;
48.    type Matrix is private;
49.
50.    --Procedures, that fills random values into Vectors and Matrixes
51.    procedure Random_Vector (A: out Vector);
52.    procedure Random_Matrix (MA: out Matrix);
53.
54.    --Procedures, that fills Matrixes and Vectors with 1:
55.    procedure All_Ones_Vector(A: out Vector);
56.    procedure All_Ones_Matrix(MA: out Matrix);
57.
58.    --Procedures, that shows values of Vectors and Matrixes in the screen
59.    procedure Vector_Output (A: in Vector; str: in String);
60.    procedure Matrix_Output (MA: in Matrix; str: in String);
61.
62.     --Procedures, that fills a values into Vectors and Matrixes
63.    procedure Vector_Input (A: out Vector; str: in String);
64.    procedure Matrix_Input (MA: out Matrix; str: in String);
65.
66.
67.    -- Procedures, that generates initial values for each Function
68.    -- Depending on the option chosen by the user (val)
69.    procedure Input_Val_F1 (A,B,C: out Vector; MA,ME: out Matrix);
70.    procedure Input_Val_F2 (MG,MH,MK: out Matrix);
71.    procedure Input_Val_F3 (t: out Integer; V,O,P: out Vector; MO,MP,MR: out Matrix);
72.
73.    --Function 1 (SORT(A)+SORT(B)+SORT(C)*(MA*ME))
74.    function Func1 (A,B,C: out Vector; MA,ME: in Matrix) return Vector;
75.
76.    --Function 2 ((MG*MH)*TRANS(MK))
77.    function Func2 (MG,MH,MK: in Matrix) return Matrix;
78.
79.    --Function 3 ((MO*MP)*V+t*MR*(O+P))
80.    function Func3 (t: in Integer; V, O, P: in Vector; MO, MP, MR: in Matrix) return
Vector;
81.
82.    private
83.    type Vector is array (1..N) of Integer;
84.    type Matrix is array (1..N, 1..N) of Integer;
85.
86. end Data;
```

data.adb

```
1. with Ada.Numerics.Discrete_Random;
2. with Ada.Integer_Text_IO, Ada.Text_IO;
3. use Ada.Integer_Text_IO, Ada.Text_IO;
4.
5. package body Data is
6.
7.     -- Create All-Ones Matrix And Vector
8.    procedure All_Ones_Matrix (MA: out Matrix) is
9.    begin
10.       for i in 1..N loop
11.          for j in 1..N loop
12.             MA(i,j) := 1;
13.          end loop;
14.       end loop;
15.    end All_Ones_Matrix;
16.
17.    procedure All_Ones_Vector (A: out Vector) is
18.    begin
19.       for i in 1..N loop
20.          A(i) := 1;
21.       end loop;
22.    end All_Ones_Vector;
23.
24.
25.    -- Function That Generate Random Numbers From -10..10
26.    function Rand_Gen return Integer is
27.       subtype randRange is Integer range -10..10;
28.       package Rand_Int is new Ada.Numerics.Discrete_Random(randRange);
29.       use Rand_Int;
30.       gen: Rand_Int.Generator;
31.    begin
32.       Rand_Int.Reset(gen);
33.       return Rand_Int.Random(gen);
34.    end Rand_Gen;
35.
36.
37.    -- Generate And Fill Random Matrix And Vector
38.    procedure Random_Matrix (MA: out Matrix) is
39.    begin
40.       for i in 1..N loop
41.          for j in 1..N loop
42.             MA(i,j) := Rand_Gen;
43.          end loop;
44.       end loop;
45.    end Random_Matrix;
46.
47.    procedure Random_Vector (A: out Vector) is
48.    begin
49.       for i in 1..N loop
50.          A(i) := Rand_Gen;
51.       end loop;
52.    end Random_Vector;
53.
54.
55.    -- Print Matrix And Vector Into Console
56.    procedure Matrix_Output (MA: in Matrix; str: in String) is
57.    begin
58.       Put_Line("Matrix " & str & ":");
59.       for i in 1..N loop
60.          for j in 1..N loop
61.             Put(MA(i,j));
62.          end loop;
63.          New_Line;
64.       end loop;
65.       New_Line;
66.    end Matrix_Output;
```

```
67.
68.    procedure Vector_Output (A: in Vector; str: in String) is
69.    begin
70.       Put("Vector " & str & ":");
71.       for i in 1..N loop
72.          Put(A(i));
73.       end loop;
74.       New_Line;
75.    end Vector_Output;
76.
77.    procedure Num_Output (a: in Integer; str: in String) is
78.    begin
79.       Put("Number " & str & ":");
80.       Put(a);
81.       New_Line;
82.    end Num_Output;
83.
84.
85.     --Fill And Print Matrices And Vectors Into Console
86.    procedure Matrix_Input (MA: out Matrix; str: in String) is
87.    begin
88.       Put_Line("Enter the" & Positive'Image(N * N) & " elements of the Matrix " & str &
":");
89.       for i in 1..N loop
90.          for j in 1..N loop
91.             Put(str & "(" & Integer'Image(i) & "." & Integer'Image(j) & " ) = ");
92.             Get(MA(i,j));
93.          end loop;
94.          New_Line;
95.       end loop;
96.    end Matrix_Input;
97.
98.    procedure Vector_Input (A: out Vector; str: in String) is
99.    begin
100.       Put_Line("Enter the" & Positive'Image(N) & " elements of the Vector " & str &
":");
101.       for i in 1..N loop
102.          Put(str & "(" & Integer'Image(i) & " ) = ");
103.          Get(A(i));
104.       end loop;
105.    end Vector_Input;
106.
107.
108.    -- Sort Vector
109.    procedure Sort_Vector(A: in out Vector) is
110.       temp: Integer;
111.    begin
112.       for i in 1..n loop
113.          for j in i..n loop
114.             if A(i)>A(j) then
115.                temp:=A(j);
116.                A(j):=A(i);
117.                A(i):=temp;
118.             end if;
119.          end loop;
120.       end loop;
121.    end Sort_Vector;
122.
123.
124.    -- Calculates Sum Of 2 Vectors
125.    function Sum_Vectors (A, B: in Vector) return Vector is
126.       result: Vector;
127.    begin
128.       for i in 1..N loop
129.          result(i) := A(i) + B(i);
130.       end loop;
131.       return result;
132.    end Sum_Vectors;
```

```
133.
134.
135.    -- Transposing Matrix
136.    function Matrix_Transposition(MA: in Matrix) return Matrix is
137.       temp: Integer;
138.       MZ: Matrix;
139.    begin
140.       for i in 1..N loop
141.          for j in 1..N loop
142.             temp := MA(j,i);
143.             MZ(j,i) := MA(i,j);
144.             MZ(i,j) := temp;
145.          end loop;
146.       end loop;
147.       return MZ;
148.    end Matrix_Transposition;
149.
150.
151.    -- Multiply 2 Matrices
152.    function Matrix_Multiplication (MA, MB: in Matrix) return Matrix is
153.       product : Integer;
154.       result: Matrix;
155.    begin
156.       for i in 1..N loop
157.          for j in 1..N loop
158.             product := 0;
159.             for k in 1..N loop
160.                product := product + MA(i,k) * MB(k,j);
161.             end loop;
162.             result(i,j) := product;
163.          end loop;
164.       end loop;
165.       return result;
166.    end Matrix_Multiplication;
167.
168.
169.    -- Multipy Matrix And Vector
170.    function Matrix_Vector_Multiplication (MA: in Matrix; A: in Vector) return Vector is
171.       product: Integer;
172.       result: Vector;
173.    begin
174.       for i in 1..N loop
175.          product := 0;
176.          for j in 1..N loop
177.             product := product + A(j) * MA(j,i);
178.          end loop;
179.          result(i) := product;
180.       end loop;
181.       return result;
182.    end Matrix_Vector_Multiplication;
183.
184.
185.    -- Multipy Integer And Matrix
186.    function Matrix_Integer_Multiplication (MA:in Matrix; a: in Integer) return Matrix
is
187.       MZ: Matrix;
188.    begin
189.       for i in 1..N loop
190.          for j in 1..N loop
191.             MZ(i,j) := a * MA(i,j);
192.          end loop;
193.       end loop;
194.       return MZ;
195.    end Matrix_Integer_Multiplication;
196.
197.
198.    -- Generate Initial Values
199.    procedure Input_Val_F1 (A,B,C: out Vector; MA,ME: out Matrix) is
```

```
200.    begin
201.       Random_Vector(A);
202.       Random_Vector(B);
203.       Random_Vector(C);
204.       Random_Matrix(MA);
205.       Random_Matrix(ME);
206.
207.       Put_Line("Entered values T1:");
208.       Vector_Output(A, "A");
209.       Vector_Output(B, "B");
210.       Vector_Output(C, "C");
211.       Matrix_Output(MA, "MA");
212.       Matrix_Output(ME, "ME");
213.    end Input_Val_F1;
214.
215.    procedure Input_Val_F2 (MG,MH,MK: out Matrix) is
216.    begin
217.       Random_Matrix(MG);
218.       Random_Matrix(MH);
219.       Random_Matrix(MK);
220.
221.       Put_Line("Entered values T2:");
222.       Matrix_Output(MG, "MG");
223.       Matrix_Output(MH, "MH");
224.       Matrix_Output(MK, "MK");
225.    end Input_Val_F2;
226.
227.    procedure Input_Val_F3 (t: out Integer; V,O,P: out Vector; MO,MP,MR: out Matrix) is
228.          begin
229.       Put("t = ");
230.       t := Rand_Gen;
231.       Random_Vector(V);
232.       Random_Vector(O);
233.       Random_Vector(P);
234.       Random_Matrix(MO);
235.       Random_Matrix(MP);
236.       Random_Matrix(MR);
237.
238.       Put_Line("Entered values T3:");
239.       Num_Output(t, "t");
240.       Vector_Output(V, "V");
241.       Vector_Output(O, "O");
242.       Vector_Output(P, "P");
243.       Matrix_Output(MO, "MO");
244.       Matrix_Output(MP, "MP");
245.       Matrix_Output(MR, "MR");
246.    end Input_Val_F3;
247.
248.
249.    --Function 1 (SORT(A)+SORT(B)+SORT(C)*(MA*ME))
250.    function Func1 (A,B,C: out Vector; MA,ME: in Matrix) return Vector is
251.    begin
252.       Sort_Vector(A);
253.       Sort_Vector(B);
254.       Sort_Vector(C);
255.
256.       return Sum_Vectors(Sum_Vectors(A, B),
Matrix_Vector_Multiplication(Matrix_Multiplication(MA, ME), C));
257.    end Func1;
258.
259.    --Function 2 ((MG*MH)*TRANS(MK))
260.    function Func2 (MG,MH, MK: in Matrix) return Matrix is
261.    begin
262.       return Matrix_Multiplication(Matrix_Multiplication(MG,MH),
Matrix_Transposition(MK));
263.    end Func2;
264.
265.    --Function 3 ((MO*MP)*V+t*MR*(O+P))
```

```
266.    function Func3 (t: in Integer; V, O, P: in Vector; MO, MP, MR: in Matrix) return
Vector is
267.    begin
268.       return Sum_Vectors(Matrix_Vector_Multiplication(Matrix_Multiplication(MO, MP),
V), Matrix_Vector_Multiplication(Matrix_Integer_Multiplication(MR, t), Sum_Vectors(O,P)));
269.    end Func3;
270. end data;
```

Приклад роботи програми

```
Function 1: D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)
Function 2: MF = (MG*MH)*TRANS(MK)
Function 3: S = (MO*MP)*V+t*MR*(O+P)

!!! Note that if the value of N > 10 -> the result will not be displayed !!!
!!! If you enter N <= 0 - execution will be terminated !!!

Enter N: 3

Task T1 started
Task T2 started
Task T3 started
Calculations started

T3 is waiting for a permit.

T3 gets a permit.
T1 is waiting for a permit.
T2 is waiting for a permit.

Entered values T3:
Number t:         -4
Vector V:         -2          1         -5
Vector O:          8        -10          5
Vector P:         -3         -8          7
Matrix MO:
          7        -10         -4
        -10         -7          6
          6          2          6

Matrix MP:
          0         -5          2
          4         -3          2
         -8        -10         -8

Matrix MR:
         -7         -9         -1
         -1          7        -10
          4         -2         -2

T3 releases the permit.

T3 is waiting for a permit.

T1 gets a permit.
Entered values T1:
Vector A:         -4          7         -4
Vector B:          8          5         -1
Vector C:         -9         10         -3
Matrix MA:
          1          7          0
         10         10         -2
          8         -3         -5
```

```
Matrix ME:
          3         -1         -3
          4          2          2
         -1         10          6

T1 releases the permit.

T1 is waiting for a permit.

T2 gets a permit.
Entered values T2:
Matrix MG:
         -3          6         -2
         -2          1          2
         -1         -5         -2

Matrix MH:
          9          0          2
         -8        -10          4
         -5         -9          2

Matrix MK:
          6         -7          1
          3         -9         -5
          2          7          9

T2 releases the permit.


T2 is waiting for a permit.

T3 gets a permit.
T3 | Vector S:         16       1201       -578
T3 releases the permit.

Task T3 finished

T1 gets a permit.
T1 | Vector D:       -330       -726       -618
T1 releases the permit.

Task T1 finished

T2 gets a permit.
T2 |
Matrix MF:
        -82        113       -298
        -16        124       -232
       -256       -359        324

T2 releases the permit.
Task T2 finished

All task finished
```