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

# ЛАБОРАТОРНА РОБОТА №1

# З ДИСЦИПЛІНИ “ОСНОВИ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ ”

# НА ТЕМУ: “ПОТОКИ В МОВІ АДА. ЗАДАЧІ”

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

**Перевірив:**
Доцент
Корочкін О. В.

м. Київ – 2020 р.

## Мета

Вивчення засобів мови Ада для роботи із процесами.

## Завдання

Розробити програму яка містить ***паралельні потоки*** (задачі), кожна із яких реалізовує функцію F1, F2, F3 із **Додатку Б** згідно отриманому варіанту.

Програма повинна складатися із пакету *Data* і основної програми — процедури *Lab1*. Пакет реалізовує ресурси, необхідні для обчислення функцій *F1, F2, F3* через підпрограми *Func1, Func2, Func3*.

При створенні задач необхідно:

- вказати ім'я задачі;
- встановити пріоритет задачі;
- задати розмір стека задачі;
- обрати і задати номер процесу (ядра) для виконання кожної задачі.

В тілі задачі задіяти оператор задержки **delay** при виконанні функцій F1, F2, F3 з невеликим часом затримки.

Дослідити при виконанні програми:

- вплив пріоритетів задач на чергу запуску задач;
- вплив оператора затримки **delay** на порядок виконання задач;
- завантаження центрального багатоядерного процесора паралельної комп'ютерної системи (ПКС). Зміна кількості ядер задається за допомогою Менеджера (Диспетчера) задач ОС Windows.

## Варіант завдання

| | | ***F1*** | ***F2*** | ***F3*** | |
|---|---|---|---|---|---|
| 25 | Шендріков Євгеній Олександрович | 1.21 | 2.27 | 3.30 | |

1.21 D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)

2.27 MF = (MG*MH)*TRANS(MK)

3.30 S = (MO*MP)*V+t*MR*(O+P)

# Лістинг програми

**lab1.adb**

```
-------------------------------------------------------
--|                    Labwork #1                     |
-------------------------------------------------------
--|   Author   |    Jack (Yevhenii) Shendrikov       |
--|   Group    |                IO-82                |
--|   Variant  |                 #25                 |
--|    Date    |             06.09.2020              |
-------------------------------------------------------
--| 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)        |
-------------------------------------------------------

with Data;
with Ada.Integer_Text_IO, Ada.Text_IO, Ada.Characters.Latin_1;
use Ada.Integer_Text_IO, Ada.Text_IO, Ada.Characters.Latin_1;
with System.Multiprocessors; use System.Multiprocessors;

procedure Lab1 is
   N, val: Integer;
   -- val -> a variable that determines what kind of input data will be sent
for processing in each task for each function (Func1..3), for example, the
input data can be random, all-ones or entered from the keyboard, it is selected
by the user in the body of the main program .

   procedure Tasks is
      package My_Data is new Data(N);
      use My_Data;

      CPU_0: CPU_Range := 0;
      CPU_1: CPU_Range := 1;
      CPU_2: CPU_Range := 2;


      task T1 is
         pragma Task_Name("T1");
         pragma Priority(4);
         pragma Storage_Size(500000000);
         pragma CPU (CPU_0);
      end T1;

      task T2 is
         pragma Task_Name("T2");
         pragma Priority(3);
         pragma Storage_Size(500000000);
         pragma CPU (CPU_1);
      end T2;

      task T3 is
         pragma Task_Name("T3");
         pragma Priority(7);
```

```
   pragma Storage_Size(500000000);
   pragma CPU (CPU_2);
end T3;


task body T1 is
   A,B,C,D: Vector;
   MA,ME: Matrix;
begin
   Put_Line("Task T1 started");
   delay 1.0;
   Input_Val_F1(A,B,C,MA,ME,val);
   D := Func1(A,B,C,MA,ME);
   delay 2.0;

   if N <= 10 then
      Put("T1 | ");
      Vector_Output(D, "D");
      New_Line;
   end if;

   Put_Line("Task T1 finished");
   New_Line;
end T1;

task body T2 is
   MG,MH,MK,MF: Matrix;
begin
   Put_Line("Task T2 started");
   delay 4.0;
   Input_Val_F2(MG,MH,MK,val);
   MF := Func2(MG,MH,MK);
   delay 5.0;

   if N <= 10 then
      Put_Line("T2 | ");
      Matrix_Output(MF, "MF");
      New_Line;
   end if;

   Put_Line("Task T2 finished");
   New_Line;
end T2;

task body T3 is
   t: Integer;
   V,O,P,S: Vector;
   MO,MP,MR: Matrix;
begin
   Put_Line("Task T3 started");
   delay(7.0);
   Input_Val_F3(t,V,O,P,MO,MP,MR,val);
   S := Func3(t,V,O,P,MO,MP,MR);
   delay(5.0);
```

```
         if N <= 10 then
            Put("T3 | ");
            Vector_Output(S, "S");
            New_Line;
         end if;

         Put_Line("Task T3 finished");
         New_Line;
      end T3;


   begin
      Put_Line("Calculations started");
      New_Line;
   end Tasks;

begin
   Put_Line("-----------------------------------------------------" & CR & LF
          & "| Function 1 | D = SORT(A)+SORT(B)+SORT(C)*(MA*ME) |" & CR & LF
          & "| Function 2 |      MF = (MG*MH)*TRANS(MK)         |" & CR & LF
          & "| Function 3 |     S = (MO*MP)*V+t*MR*(O+P)        |" & CR & LF
          & "-----------------------------------------------------" & CR & LF);

   Put_Line("!!! Note that if the value of N > 10 -> the result will not be
displayed !!!" & CR & LF
          & "!!! If you enter N <= 0 - execution will be terminated !!!"  & CR
& LF
          & "!!! Also note that the header may also be distorted if exe is
running in GNAT Studio !!!" & CR & LF);
   Put("Enter N: ");
   Get(N);

   if N <= 0 then
      raise PROGRAM_ERROR with "Restart the program and enter N > 0.";
   end if;

   New_Line;
   Put_Line("Select the method according to which the initial data will be
generated:");
   Put_Line("   [1] - Create All-Ones Matrices And Vectors.");
   Put_Line("   [2] - Create Matrices And Vectors With Random Values.");
   Put_Line("   [3] - Enter All Values From The Keyboard (The program works,
but due to threads the input can be a little distorted).");
   Put("Your choice [1] / [2] / [3]: ");
   Get(val);
   New_Line;

   if N > 10 and val = 3 then
      raise PROGRAM_ERROR with "If you want to enter a value from the keyboard
- enter N <= 10.";
   end if;

   Tasks;
end Lab1;
```

**data.adb**

```
with Ada.Numerics.Discrete_Random;
with Ada.Integer_Text_IO, Ada.Text_IO;
use Ada.Integer_Text_IO, Ada.Text_IO;

package body Data is

    -- Create All-Ones Matrix And Vector
   procedure All_Ones_Matrix (MA: out Matrix) is
   begin
      for i in 1..N loop
         for j in 1..N loop
            MA(i,j) := 1;
         end loop;
      end loop;
   end All_Ones_Matrix;

   procedure All_Ones_Vector (A: out Vector) is
   begin
      for i in 1..N loop
         A(i) := 1;
      end loop;
   end All_Ones_Vector;


   -- Function That Generate Random Numbers From -10..10
   function Rand_Gen return Integer is
      subtype randRange is Integer range -10..10;
      package Rand_Int is new Ada.Numerics.Discrete_Random(randRange);
      use Rand_Int;
      gen: Rand_Int.Generator;
   begin
      Rand_Int.Reset(gen);
      return Rand_Int.Random(gen);
   end Rand_Gen;

   -- Generate And Fill Random Matrix And Vector
   procedure Random_Matrix (MA: out Matrix) is
   begin
      for i in 1..N loop
         for j in 1..N loop
            MA(i,j) := Rand_Gen;
         end loop;
      end loop;
   end Random_Matrix;

   procedure Random_Vector (A: out Vector) is
   begin
      for i in 1..N loop
         A(i) := Rand_Gen;
      end loop;
   end Random_Vector;
```

```
   -- Print Matrix And Vector Into Console
   procedure Matrix_Output (MA: in Matrix; str: in String) is
   begin
      Put_Line("Matrix " & str & ":");
      for i in 1..N loop
         for j in 1..N loop
            Put(MA(i,j));
         end loop;
         New_Line;
      end loop;
      New_Line;
   end Matrix_Output;

   procedure Vector_Output (A: in Vector; str: in String) is
   begin
      Put("Vector " & str & ":");
      for i in 1..N loop
         Put(A(i));
      end loop;
      New_Line;
   end Vector_Output;

   procedure Num_Output (a: in Integer; str: in String) is
   begin
      Put("Number " & str & ":");
      Put(a);
      New_Line;
   end Num_Output;

    --Fill And Print Matrices And Vectors Into Console
   procedure Matrix_Input (MA: out Matrix; str: in String) is
   begin
      Put_Line("Enter the" & Positive'Image(N * N) & " elements of the Matrix "
& str & ":");
      for i in 1..N loop
         for j in 1..N loop
            Put(str & "(" & Integer'Image(i) & "." & Integer'Image(j) & " ) =
");
            Get(MA(i,j));
         end loop;
         New_Line;
      end loop;
   end Matrix_Input;

   procedure Vector_Input (A: out Vector; str: in String) is
   begin
      Put_Line("Enter the" & Positive'Image(N) & " elements of the Vector " &
str & ":");
      for i in 1..N loop
         Put(str & "(" & Integer'Image(i) & " ) = ");
         Get(A(i));
      end loop;
   end Vector_Input;
```

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


-- Calculates Sum Of 2 Vectors
function Sum_Vectors (A, B: in Vector) return Vector is
   result: Vector;
begin
   for i in 1..N loop
      result(i) := A(i) + B(i);
   end loop;
   return result;
end Sum_Vectors;


-- Transposing Matrix
function Matrix_Transposition(MA: in Matrix) return Matrix is
   temp: Integer;
   MZ: Matrix;
begin
   for i in 1..N loop
      for j in 1..N loop
         temp := MA(j,i);
         MZ(j,i) := MA(i,j);
         MZ(i,j) := temp;
      end loop;
   end loop;
   return MZ;
end Matrix_Transposition;


-- Multiply 2 Matrices
function Matrix_Multiplication (MA, MB: in Matrix) return Matrix is
   product : Integer;
   result: Matrix;
begin
   for i in 1..N loop
      for j in 1..N loop
         product := 0;
         for k in 1..N loop
            product := product + MA(i,k) * MB(k,j);
         end loop;
```

```
            result(i,j) := product;
         end loop;
      end loop;
      return result;
   end Matrix_Multiplication;


   -- Multipy Matrix And Vector
   function Matrix_Vector_Multiplication (MA: in Matrix; A: in Vector) return
Vector is
      product: Integer;
      result: Vector;
   begin
      for i in 1..N loop
         product := 0;
         for j in 1..N loop
            product := product + A(j) * MA(j,i);
         end loop;
         result(i) := product;
      end loop;
      return result;
   end Matrix_Vector_Multiplication;


   -- Multipy Integer And Matrix
   function Matrix_Integer_Multiplication (MA:in Matrix; a: in Integer) return
Matrix is
      MZ: Matrix;
   begin
      for i in 1..N loop
         for j in 1..N loop
            MZ(i,j) := a * MA(i,j);
         end loop;
      end loop;
      return MZ;
   end Matrix_Integer_Multiplication;


   -- Generate Initial Values
   procedure Input_Val_F1 (A,B,C: out Vector; MA,ME: out Matrix; val: Integer)
is
   begin
      case val is
         when 2 =>
               Random_Vector(A);
               Random_Vector(B);
               Random_Vector(C);
               Random_Matrix(MA);
               Random_Matrix(ME);

               if N <= 10 then
                  Put_Line("Entered values T1:");
                   Vector_Output(A, "A");
                   Vector_Output(B, "B");
```

```
                Vector_Output(C, "C");
                Matrix_Output(MA, "MA");
               Matrix_Output(ME, "ME");
            end if;

      when 3 =>
            Vector_Input(A, "A");
            Vector_Input(B, "B");
            Vector_Input(C, "C");
            Matrix_Input(MA, "MA");
            Matrix_Input(ME, "ME");

            Put_Line("Entered values T1:");
            Vector_Output(A, "A");
            Vector_Output(B, "B");
            Vector_Output(C, "C");
            Matrix_Output(MA, "MA");
            Matrix_Output(ME, "ME");

      when others =>
            All_Ones_Vector(A);
            All_Ones_Vector(B);
            All_Ones_Vector(C);
            All_Ones_Matrix(MA);
            All_Ones_Matrix(ME);
   end case;
end Input_Val_F1;

procedure Input_Val_F2 (MG,MH,MK: out Matrix; val: Integer) is
begin
   case val is
      when 2 =>
         Random_Matrix(MG);
         Random_Matrix(MH);
         Random_Matrix(MK);

         if N <= 10 then
            Put_Line("Entered values T2:");
            Matrix_Output(MG, "MG");
            Matrix_Output(MH, "MH");
            Matrix_Output(MK, "MK");
         end if;

      when 3 =>
            Matrix_Input(MG, "MG");
            Matrix_Input(MH, "MH");
            Matrix_Input(MK, "MK");

            Put_Line("Entered values T2:");
            Matrix_Output(MG, "MG");
            Matrix_Output(MH, "MH");
            Matrix_Output(MK, "MK");
```

```
      when others =>
               All_Ones_Matrix(MG);
               All_Ones_Matrix(MH);
               All_Ones_Matrix(MK);
      end case;
   end Input_Val_F2;

   procedure Input_Val_F3 (t: out Integer; V,O,P: out Vector; MO,MP,MR: out
Matrix; val: Integer) is
      begin
      case val is
         when 2 =>
            t := Rand_Gen;
            Random_Vector(V);
            Random_Vector(O);
            Random_Vector(P);
            Random_Matrix(MO);
            Random_Matrix(MP);
            Random_Matrix(MR);

            if N <= 10 then
               Put_Line("Entered values T3:");
               Num_Output(t, "t");
               Vector_Output(V, "V");
               Vector_Output(O, "O");
               Vector_Output(P, "P");
               Matrix_Output(MO, "MO");
               Matrix_Output(MP, "MP");
               Matrix_Output(MR, "MR");
            end if;

         when 3 =>
               Put("t = ");
               Get(t);
               Vector_Input(V, "V");
               Vector_Input(O, "O");
               Vector_Input(P, "P");
               Matrix_Input(MO, "MO");
               Matrix_Input(MP, "MP");
               Matrix_Input(MR, "MR");

               Put_Line("Entered values T3:");
               Num_Output(t, "t");
               Vector_Output(V, "V");
               Vector_Output(O, "O");
               Vector_Output(P, "P");
               Matrix_Output(MO, "MO");
               Matrix_Output(MP, "MP");
               Matrix_Output(MR, "MR");

         when others =>
               t := 1;
               All_Ones_Vector(V);
               All_Ones_Vector(O);
```

```
                All_Ones_Vector(P);
                All_Ones_Matrix(MO);
                All_Ones_Matrix(MP);
                All_Ones_Matrix(MR);
        end case;
    end Input_Val_F3;


    --Function 1 (SORT(A)+SORT(B)+SORT(C)*(MA*ME))
    function Func1 (A,B,C: out Vector; MA,ME: in Matrix) return Vector is
        MB: Matrix;
        R, O, Q: Vector;
    begin
        Sort_Vector(A);
        Sort_Vector(B);
        Sort_Vector(C);

        R := Sum_Vectors(A, B);
        MB := Matrix_Multiplication(MA, ME);
        O := Matrix_Vector_Multiplication(MB, C);

        Q := Sum_Vectors(R, O);
        return Q;
    end Func1;

    --Function 2 ((MG*MH)*TRANS(MK))
    function Func2 (MG,MH, MK: in Matrix) return Matrix is
        MA, MB, MC: Matrix;
    begin
        MA := Matrix_Multiplication(MG,MH);
        MB := Matrix_Transposition(MK);

        MC := Matrix_Multiplication(MA, MB);
        return MC;
    end Func2;

    --Function 3 ((MO*MP)*V+t*MR*(O+P))
    function Func3 (t: in Integer; V, O, P: in Vector; MO, MP, MR: in Matrix)
return Vector is
        A,B,C,D: Vector;
        MM,MZ: Matrix;
    begin
        MM := Matrix_Multiplication(MO, MP);
        A := Matrix_Vector_Multiplication(MM, V);

        B := Sum_Vectors(O,P);
        MZ := Matrix_Integer_Multiplication(MR, t);
        C := Matrix_Vector_Multiplication(MZ, B);

        D := Sum_Vectors(A, C);
        return D;
    end Func3;
end data;
```

**data.ads**

```
generic
   N: Integer;
package Data is
   type Vector is private;
   type Matrix is private;

   --Procedures, that fills random values into Vectors and Matrixes
   procedure Random_Vector (A: out Vector);
   procedure Random_Matrix (MA: out Matrix);

   --Procedures, that fills Matrixes and Vectors with 1:
   procedure All_Ones_Vector(A: out Vector);
   procedure All_Ones_Matrix(MA: out Matrix);

   --Procedures, that shows values of Vectors and Matrixes in the screen
   procedure Vector_Output (A: in Vector; str: in String);
   procedure Matrix_Output (MA: in Matrix; str: in String);

    --Procedures, that fills a values into Vectors and Matrixes
   procedure Vector_Input (A: out Vector; str: in String);
   procedure Matrix_Input (MA: out Matrix; str: in String);



   -- Procedures, that generates initial values for each Function
   -- Depending on the option chosen by the user (val)
   procedure Input_Val_F1 (A,B,C: out Vector; MA,ME: out Matrix; val: Integer);
   procedure Input_Val_F2 (MG,MH,MK: out Matrix; val: Integer);
   procedure Input_Val_F3 (t: out Integer; V,O,P: out Vector; MO,MP,MR: out
Matrix; val: Integer);

   --Function 1 (SORT(A)+SORT(B)+SORT(C)*(MA*ME))
   function Func1 (A,B,C: out Vector; MA,ME: in Matrix) return Vector;

   --Function 2 ((MG*MH)*TRANS(MK))
   function Func2 (MG,MH,MK: in Matrix) return Matrix;
```

```
   --Function 3 ((MO*MP)*V+t*MR*(O+P))
   function Func3 (t: in Integer; V, O, P: in Vector; MO, MP, MR: in Matrix)
return Vector;

   private
   type Vector is array (1..N) of Integer;
   type Matrix is array (1..N, 1..N) of Integer;

end Data;
```

## Аналіз проблем

Проблемою при тестуванні програмної частини стало введення початкових даних з клавіатури, при якому програма поводила себе доволі незвично.

Загально відомо, що кожна задача здійснює введення даних, обчислення своєї функції, виведення даних. Введення даних можливе з файлу, з клавіатури, генерацією випадкових чисел або заповненням усіх значень одиничками.

В процесі виконання лабораторної роботи було реалізовано останні 3 варіанти введення даних (клавіатура; випадкові значення; усі значення як одинички). Який саме вид даних піде на вхід обирає користувач (даний функціонал прописаний в тілі основної програми).

Продовжуючи аналіз проблеми візьмемо наступну ситуацію: припустимо, що N=3, користувач обрав введення з клавіатури, і завданням є для T1 ввести всі елементи як 1, для T2 - як 2, для T3 - як 3. Якщо з цією проблемою вводу з клавіатури нічого не зробити, то при вводі значень, якщо 3 потоки одночасно запускаються, буде незрозуміло, що куди вводити, буде боротьба за спільні ресурси і все закінчиться хаосом, так само і з виводом.

Описана проблема як для введення, так і для виведення результату є загальною проблемою паралельного програмування.

Можливе рішення даного питання полягає у введенні даних не з клавіатури, а програмно, за допомогою процедур, які будуть викликані у тілі тої чи іншої задачі. Наприклад, процедури, яка в T1 буде встановлювати всі елементи для певної матриці як 1 (ще одна процедура так само для векторів), а в T2 інша процедура буде встановлювати всі елементи як 2 і т.д. При цьому, якщо брати велике N (2000, 3000), то в коді, краще за все, поставити умову, щоб при таких (великих > 10,15) N в консоль нічого не виводило, крім того, ясна річ, що задача запустилась і завершилась.

При N < 10,15 в консоль будуть виводитись дані, але, щоб покращити ввід/вивід, у тілі задач слід вказувати оператор *delay* (один або декілька разів), під час виконання цього оператора задача блокується і керування передається іншій задачі, яка знаходиться у стані готовності. При таких невеликих N і програмному вводу/виводу, а також при правильному підборі *delay* – процеси не мають заважати один одному, і проблем з введенням та виведенням результатів в консоль можна уникнути, спроба чого була виконана у моїй програмі.

Також необхідно завжди вказувати пріоритет, бо саме він визначає привілеї задачі на володіння системними ресурсами (наприклад, процесором). Саме пріоритети працюють при формуванні черг, у яких знаходяться задачі.

Повертаючись до проблеми вводу з клавіатури, у програмі була зроблена спроба спростити ввід даних для користувача, таким чином, що в консолі вказувалось в який самий вектор (скаляр, матрицю) і на яку позицію користувач вводить дані. Виглядає це приблизно так (при N=3):

```
Enter the 3 elements of the Vector A:
A( 1 ) = 1
A( 2 ) = 1
A( 3 ) = 1

Enter the 3 elements of the Vector B:
B( 1 ) = 1
B( 2 ) = 1
B( 3 ) = Enter the 9 elements of the Matrix MG:
MG( 1. 1 ) =
```

Звісно, як можна помітити, це не вирішує основної проблеми, оскільки для рішення проблем паралельного програмування потрібні спеціальні програмні засоби, які ми будемо вивчати (семафори, мютекси, монітори та інше), та все ж це трохи спрощує для користувача ввід даних з клавіатури.

Наостанок, зазначу, що виключні ситуації (типу вводу N < 0 чи коли користувач ставить велике N і обирає введення з клавітуари) були оброблені – при них в тілі основної програми піднімається помилка і програма перестає працювати. Приклад з коду:

```
if N <= 0 then
   raise PROGRAM_ERROR with "Restart the program and enter N > 0.";
end if;
```

# 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 !!!
!!! Also note that the header may also be distorted if exe is running in GNAT
Studio !!!

Enter N: 2

Select the method according to which the initial data will be generated:
   [1] - Create All-Ones Matrices And Vectors.
   [2] - Create Matrices And Vectors With Random Values.
   [3] - Enter All Values From The Keyboard (The program works, but due to
streams the input can be a little distorted).
Your choice [1] / [2] / [3]: 2

Task T1 started
Task T3 started
Task T2 started
Calculations started

Entered values T1:
Vector A:         -8          8
Vector B:          5         -3
Vector C:          0         -7
Matrix MA:
          7         -9
         -9         -5

Matrix ME:
          4          0
         -1          1

T1 | Vector D:       -270         76
Task T1 finished

Entered values T2:
Matrix MG:
         -6          4
         -7         -5

Matrix MH:
         10          1
          2         -1

Matrix MK:
          5         -5
          2          0
```

```
Entered values T3:
Number t:          -3
Vector V:          -6           1
Vector O:           5          -2
Vector P:          -5           9
Matrix MO:
           2           1
          -2           6
Matrix MP:
           2          10
          10          -4
Matrix MR:
          -6          -4
          -9          -7

T2 |
Matrix MF:
        -210        -104
        -390        -160
Task T2 finished

T3 | Vector S:         161           7
Task T3 finished

[2020-09-08 14:48:54] process terminated successfully, elapsed time: 14.53s
```

## Перевірка результатів прикладу

У якості вхідних даних були взяті саме рандомізовані значення для скаляру, векторів та матриць, оскільки на них можна перевірити такі процедури як сортування та транспонування матриць, чого б не було видно, якщо б ми взяли всі значення як одиниці.

*Перевірка T1 з обчисленням F1*

Умова: $D = SORT(A) + SORT(B) + SORT(C) * (MA * ME)$

$Sort(A) \rightarrow (-8 \quad 8)$

$Sort(B) \rightarrow (-3 \quad 5)$

$Sort(C) \rightarrow (-7 \quad 0)$

$$MA \cdot ME = \begin{pmatrix} 7 & -9 \\ -9 & -5 \end{pmatrix} \cdot \begin{pmatrix} 4 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 37 & -9 \\ -31 & -5 \end{pmatrix}$$

$$SORT(C) \cdot (MA \cdot ME) = (-7 \quad 0) \cdot \begin{pmatrix} 37 & -9 \\ -31 & -5 \end{pmatrix} = (-259 \quad 63)$$

$$D = (-8 \quad 8) + (-3 \quad 5) + (-259 \quad 63) = (-270 \quad 76)$$

*Перевірка T2 з обчисленням F2*

Умова: $MF = (MG * MH) * TRANS(MK)$

$$MG \cdot MH = \begin{pmatrix} -6 & 4 \\ -7 & -5 \end{pmatrix} \cdot \begin{pmatrix} 10 & 1 \\ 2 & -1 \end{pmatrix} = \begin{pmatrix} -52 & -10 \\ -80 & -2 \end{pmatrix}$$

$$TRANS(MK) \rightarrow \begin{pmatrix} 5 & 2 \\ -5 & 0 \end{pmatrix}$$

$$MF = \begin{pmatrix} -52 & -10 \\ -80 & -2 \end{pmatrix} \cdot \begin{pmatrix} 5 & 2 \\ -5 & 0 \end{pmatrix} = \begin{pmatrix} -210 & -104 \\ -390 & -160 \end{pmatrix}$$

*Перевірка T3 з обчисленням F3*

Умова: $S = (MO * MP) * V + t * MR * (O + P)$

$$MO \cdot MP = \begin{pmatrix} 2 & 1 \\ -2 & 6 \end{pmatrix} \cdot \begin{pmatrix} 2 & 10 \\ 10 & -4 \end{pmatrix} = \begin{pmatrix} 14 & 16 \\ 56 & -44 \end{pmatrix}$$

$$O + P = (5 \quad -2) + (-5 \quad 9) = (0 \quad 7)$$

$$t \cdot MR = -3 \cdot \begin{pmatrix} -6 & -4 \\ -9 & -7 \end{pmatrix} = \begin{pmatrix} 18 & 12 \\ 27 & 21 \end{pmatrix}$$

$$V \cdot (MO \cdot MP) = (-6 \quad 1) \cdot \begin{pmatrix} 14 & 16 \\ 56 & -44 \end{pmatrix} = (-28 \quad -140)$$

$$(O + P) \cdot t \cdot MR = (0 \quad 7) \cdot \begin{pmatrix} 18 & 12 \\ 27 & 21 \end{pmatrix} = (189 \quad 147)$$

$$S = (-28 \quad -140) + (189 \quad 147) = (161 \quad 7)$$

Результати програми повністю збігаються з теоретичними даними. Програма працює вірно.

## Висновок

В даній лабораторній роботі було розглянуто принцип побудови паралельних програм на основі мови Ада. Було з'ясовано, що в якості досягнення паралельності при виконанні програми Ада надає можливість використовувати тип задачі (task), що забезпечує створення декількох потоків окремо, які будуть оброблювати ту чи іншу задану інформацію одночасно.

За принципом мови Ада, при оголошенні потоку, і переходу до функції, де вони були оголошені, програма автоматично запускає додаткові потоки на виконання, враховуючи основний потік процедури.

Було зроблено аналіз проблем, які виникли під час виконання, а також запропоновано шляхи їх рішення, перевірено працездатність програми і програмним шляхом оброблено декілька виключних ситуацій. Кінцева мета роботи досягнута.