

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 ІІІ 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:

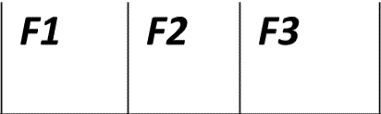
* вказати ім’я задачі;
* встановити пріоритет задачі;
* задати розмір стека задачі;
* обрати і задати номер процесу (ядра) для виконання кожної задачі.

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**



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 Т1 ввести всі елементи як 1, для Т2 - як 2, для Т3 - 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*

=

*Перевірка T2 з обчисленням F2*

*Перевірка T3 з обчисленням F3*

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.