Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Основи паралельного програмування

Лабораторна робота №6

**«Ада. Рандеву»**

Виконала:

студентка групи ІВ-71

Молчанова В.С.

Перевірив:

Корочкін О.В.

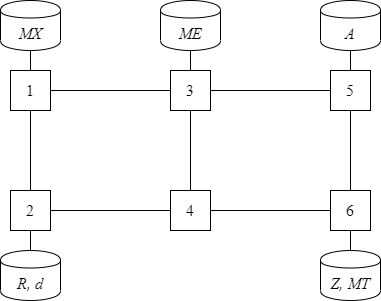
Київ

2020 р.

**Мета роботи**: розробка програми для ПКС з ЛП.

## Завдання

* *Р=6*
* *А = (R\*MT)\*ME + min(R)\*(Z\*MX) + sort(Z)\*d*



* Мова програмування: Ада
* Засоби: механізм рандеву.

## Виконання роботи

### 1. Паралельний алгоритм

Алгоритм обчислення виразу:

1. 
2. 
3. 
4. 
5. 
6. 
7. 

Спільні ресурси: .

### 2. Алгоритми потоків

|  |  |
| --- | --- |
|  | **Т1** |
| 1. | Ввести дані: *MX* |
| 2. | Отримати  від Т3 |
| 3. |  |
| 4. | Відправити до Т2 |
| 5. | Отримати  від Т2 |
| 6. | Відправити  до Т3 |
| 7. |  |
| 8. | Відправити  до Т2 |
| 9. | Отримати  від Т2 |
| 10. | Відправити  до Т3 |
| 11. | Обчислити |
| 12. | Відправити  до Т3 |

|  |  |
| --- | --- |
|  | **Т2** |
| 1. | Ввести дані: *R, d* |
| 2. | Отримати  від Т4 |
| 3. |  |
| 4. | Отримати  від Т1 |
| 5. |  |
| 6. | Отримати  від Т4 |
| 7. |  |
| 8. | Відправити  до Т1 |
| 9. | Відправити  до Т4 |
| 10. |  |
| 11. | Отримати  від Т1 |
| 12. |  |
| 13. | Отримати  від Т4 |
| 14. |  |
| 15. | Відправити  до Т1 та Т4 |
| 16. | Обчислити |
| 17. | Відправити  до Т4 |

|  |  |
| --- | --- |
|  | **Т3** |
| 1. | Ввести дані: *ME* |
| 2. | Отримати  від Т5 |
| 3. | Відправити  до Т1 |
| 4. |  |
| 5. | Відправити до Т4 |
| 6. | Отримати  від Т1 |
| 7. | Відправити  до Т5 |
| 8. |  |
| 9. | Відправити  до Т4 |
| 10. | Отримати  від Т1 |
| 11. | Відправити  до Т5 |
| 12. | Обчислити |
| 13. | Отримати  від Т1 |
| 14. | Відправити  до Т5 |

|  |  |
| --- | --- |
|  | **Т4** |
| 1. | Отримати  від Т6 |
| 2. | Відправити  до Т2 |
| 3. |  |
| 4. | Отримати  від Т3 |
| 5. |  |
| 6. | Отримати  від Т6 |
| 7. |  |
| 8. | Відправити  до Т2 |
| 9. | Отримати  від Т2 |
| 10. | Відправити  до Т6 |
| 11. |  |
| 12. | Отримати  від Т3 |
| 13. |  |
| 14. | Отримати  від Т6 |
| 15. |  |
| 16. | Відправити  до Т2 |
| 17. | Отримати  від Т2 |
| 18. | Відправити  до Т6 |
| 19. | Обчислити |
| 20. | Отримати  від Т2 |
| 21. | Відправити  до Т6 |

|  |  |
| --- | --- |
|  | **Т5** |
| 1. | Отримати  від Т6 |
| 2. | Відправити  до Т3 |
| 3. |  |
| 4. | Відправити до Т6 |
| 5. | Отримати  від Т3 |
| 6. |  |
| 7. | Відправити  до Т6 |
| 8. | Отримати  від Т3 |
| 9. | Обчислити |
| 10. | Отримати  від Т3 |
| 11. | Отримати  від Т6 |
| 12. | Вивести *А* |

|  |  |
| --- | --- |
|  | **Т6** |
| 1. | Ввести дані: *Z, MT* |
| 2. | Відправити  до Т5 |
| 3. | Відправити  до Т4 |
| 4. |  |
| 5. | Отримати  від Т5 |
| 6. |  |
| 7. | Відправити  до Т4 |
| 8. | Отримати  від Т4 |
| 9. |  |
| 10. | Отримати  від Т5 |
| 11. |  |
| 12. | Відправити  до Т4 |
| 13. | Отримати  від Т4 |
| 14. | Обчислити |
| 15. | Отримати  від Т4 |
| 16. | Відправити  до Т5 |

### 3. Структурна схема взаємодії потоків

У програмі використовуються захищені входи для передачі даних з одного потоку до іншого. Кожен з потоків має такі захищені входи:

* DataT*i* (на рис. 1 та 2 позначено як DT*i*) – приймає дані від потоку з номером *i.*
* SortResult (на рис. 2 позначено як SR) – надсилає вектор результату сортування *S*
* MinPartResult (на рис. 2 позначено як MPR) – надсилає частину вектора результату проміжного обчислення *Y* та локальний мінімум *m*
* MinResult (на рис. 3 позначено як MR) – надсилає повний вектор результату проміжного обчислення *Y* та мінімум *m.*
* Result (на рис. 3 позначено як R) – надсилає вектор результату обчислення *А*

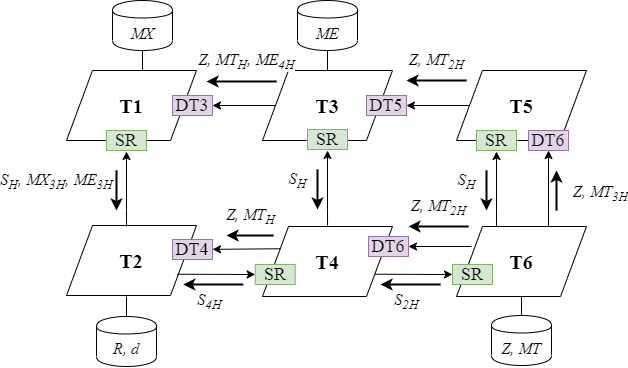


Рис. 1. Структурна схема взаємодії потоків під час підготовки до сортування та його виконання

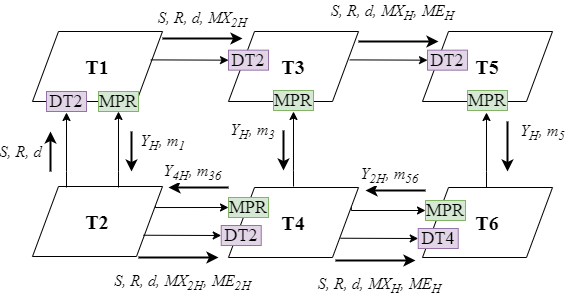


Рис. 2. Структурна схема взаємодії потоків під час обміну даними перед обчисленням виразу

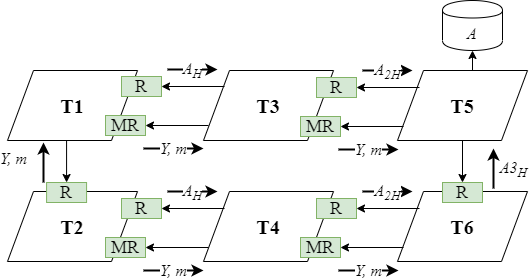


Рис. 3. Структурна схема взаємодії потоків під час передачі результатів після обчислення проміжних результатів та остаточного виразу

### 4. Лістинг програми

#### Файл main.adb

with Data; use Data;

with Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar, Ada.Float\_Text\_IO;

use Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar, Ada.Float\_Text\_IO;

procedure Main is

StartTime: Time;

ch : Character;

procedure RunTasks is

-----------------------------------------------------------

--------------------DESCRIPTION OF TASKS-------------------

-----------------------------------------------------------

--Note: Parameter name Z\_4H\_2\_3\_4 in task's entry means----

--------that it contains 4 parts of length H:--------------

--------for task itself and for tasks 2, 3 and 4-----------

-----------------------------------------------------------

task T1 is

entry DataT3(Z\_8H: in Vector; MT\_H, ME\_4H\_2\_4\_6: in Matrix);

entry SortResult(S\_H: out Vector; MX\_3H\_2\_4\_6, ME\_3H\_2\_4\_6: out Matrix);

entry DataT2(S\_8H, R\_8H: in Vector; d: in Integer);

entry MinPartResult(Y\_H: out Vector; m1: out Integer);

entry MinResult(Y\_8H: out Vector; m: out Integer);

entry Result(A\_H: out Vector);

end T1;

task T2 is

entry DataT4(Z\_8H: in Vector; MT\_H: in Matrix);

entry MinResult(Y\_8H: out Vector; m: out Integer);

entry Result(A\_H: out Vector);

end T2;

task T3 is

entry DataT5(Z\_8H: in Vector; MT\_2H\_1: in Matrix);

entry SortResult(S\_H: out Vector);

entry DataT1(S\_8H, R\_8H: in Vector; d: in Integer; MX\_2H\_5: in Matrix);

entry MinPartResult(Y\_H: out Vector; m3: out Integer);

entry MinResult(Y\_8H: out Vector; m: out Integer);

entry Result(A\_2H\_1: out Vector);

end T3;

task T4 is

entry DataT6(Z\_8H: in Vector; MT\_2H\_2: in Matrix);

entry SortResult(S\_4H\_3\_5\_6: out Vector);

entry DataT2(S\_8H, R\_8H: in Vector; d: in Integer; MX\_2H\_6, ME\_2H\_6: in Matrix);

entry MinPartResult(Y\_4H\_3\_5\_6: out Vector; m36: out Integer);

entry MinResult(Y\_8H: out Vector; m: out Integer);

entry Result(A\_2H\_2: out Vector);

end T4;

task T5 is

entry DataT6(Z\_8H: in Vector; MT\_3H\_1\_3: in Matrix);

entry SortResult(S\_H: out Vector);

entry DataT3(S\_8H, R\_8H: in Vector; d: in Integer; MX\_H, ME\_H: in Matrix);

entry MinPartResult(Y\_H: out Vector; m5: out Integer);

end T5;

task T6 is

entry SortResult(S\_2H\_5: out Vector);

entry DataT4(S\_8H, R\_8H: in Vector; d: in Integer; MX\_H, ME\_H: in Matrix);

entry MinPartResult(Y\_2H\_5: out Vector; m56: out Integer);

entry Result(A\_3H\_2\_4: out Vector);

end T6;

task body T1 is

Z, S, R, A, Y: Vector := new Vector\_t;

MT, ME, MX: Matrix := new Matrix\_t;

taskNumber: Integer := 1;

first, last, d\_in, m1\_in, m\_in: Integer;

begin

Put\_Line("Task " & Integer'Image(taskNumber) & " started...");

first := GetFirst(taskNumber);

last := GetLast(taskNumber);

InputMatrix(MX);

accept DataT3 (Z\_8H : in Vector; MT\_H : in Matrix; ME\_4H\_2\_4\_6 : in Matrix) do

CopyVectorPart(Z\_8H, 1, N, Z);

CopyMatrixPart(MT\_H, first, last, MT);

CopyMatrixPart(ME\_4H\_2\_4\_6, first, GetLast(2), ME);

CopyMatrixPart(ME\_4H\_2\_4\_6, GetFirst(4), GetLast(4), ME);

CopyMatrixPart(ME\_4H\_2\_4\_6, GetFirst(6), GetLast(6), ME);

end DataT3;

CopyVectorPart(Z, first, last, S);

SortH(S, first, last);

accept SortResult (S\_H : out Vector; MX\_3H\_2\_4\_6 : out Matrix; ME\_3H\_2\_4\_6 : out Matrix) do

CopyVectorPart(S, first, last, S\_H);

CopyMatrixPart(MX, GetFirst(2), GetLast(2), MX\_3H\_2\_4\_6);

CopyMatrixPart(MX, GetFirst(4), GetLast(4), MX\_3H\_2\_4\_6);

CopyMatrixPart(MX, GetFirst(6), GetLast(6), MX\_3H\_2\_4\_6);

CopyMatrixPart(ME, GetFirst(2), GetLast(2), ME\_3H\_2\_4\_6);

CopyMatrixPart(ME, GetFirst(4), GetLast(4), ME\_3H\_2\_4\_6);

CopyMatrixPart(ME, GetFirst(6), GetLast(6), ME\_3H\_2\_4\_6);

end SortResult;

accept DataT2 (S\_8H : in Vector; R\_8H : in Vector; d : in Integer) do

CopyVectorPart(S\_8H, 1, N, S);

CopyVectorPart(R\_8H, 1, N, R);

d\_in := d;

end DataT2;

T3.DataT1(S\_8H => S,

R\_8H => R,

d => d\_in,

MX\_2H\_5 => MX);

CalcMinPart(R, MT, first, last, Y, m1\_in);

accept MinPartResult (Y\_H : out Vector; m1 : out Integer) do

CopyVectorPart(Y, first, last, Y\_H);

m1 := m1\_in;

end MinPartResult;

T2.MinResult(Y\_8H => Y,

m => m\_in);

accept MinResult (Y\_8H : out Vector; m : out Integer) do

CopyVectorPart(Y, 1, N, Y\_8H);

m := m\_in;

end MinResult;

CalcResult(Y, Z, S, ME, MX, d\_in, m\_in, first, last, A);

accept Result (A\_H : out Vector) do

CopyVectorPart(A, first, last, A\_H);

end Result;

Put\_Line("Task " & Integer'Image(taskNumber) & " finished");

end T1;

task body T2 is

Z, S, R, A, Y: Vector := new Vector\_t;

MT, ME, MX: Matrix := new Matrix\_t;

taskNumber: Integer := 2;

first, last, d\_in, m2\_in, m12\_in, m\_in: Integer;

begin

Put\_Line("Task " & Integer'Image(taskNumber) & " started...");

first := GetFirst(taskNumber);

last := GetLast(taskNumber);

InputVector(R);

d\_in := 1;

accept DataT4 (Z\_8H : in Vector; MT\_H : in Matrix) do

CopyVectorPart(Z\_8H, 1, N, Z);

CopyMatrixPart(MT\_H, first, last, MT);

end DataT4;

CopyVectorPart(Z, first, last, S);

SortH(S, first, last);

T1.SortResult(S\_H => S,

MX\_3H\_2\_4\_6 => MX,

ME\_3H\_2\_4\_6 => ME);

MergeSort(S, GetFirst(1), GetLast(1), last);

T4.SortResult(S\_4H\_3\_5\_6 => S);

MergeSort(S, 1, last, N);

T1.DataT2(S\_8H => S,

R\_8H => R,

d => d\_in);

T4.DataT2(S\_8H => S,

R\_8H => R,

d => d\_in,

MX\_2H\_6 => MX,

ME\_2H\_6 => ME);

CalcMinPart(R, MT, first, last, Y, m2\_in);

T1.MinPartResult(Y\_H => Y,

m1 => m12\_in);

if m2\_in < m12\_in then

m12\_in := m2\_in;

end if;

T4.MinPartResult(Y\_4H\_3\_5\_6 => Y,

m36 => m\_in);

if m12\_in < m\_in then

m\_in := m12\_in;

end if;

for t in 1..2 loop

accept MinResult (Y\_8H : out Vector; m : out Integer) do

CopyVectorPart(Y, 1, N, Y\_8H);

m := m\_in;

end MinResult;

end loop;

CalcResult(Y, Z, S, ME, MX, d\_in, m\_in, first, last, A);

accept Result (A\_H : out Vector) do

CopyVectorPart(A, first, last, A\_H);

end Result;

Put\_Line("Task " & Integer'Image(taskNumber) & " finished");

end T2;

task body T3 is

Z, S, R, A, Y: Vector := new Vector\_t;

MT, ME, MX: Matrix := new Matrix\_t;

taskNumber: Integer := 3;

first, last, d\_in, m3\_in, m\_in: Integer;

begin

Put\_Line("Task " & Integer'Image(taskNumber) & " started...");

first := GetFirst(taskNumber);

last := GetLast(taskNumber);

InputMatrix(ME);

accept DataT5 (Z\_8H : in Vector; MT\_2H\_1 : in Matrix) do

CopyVectorPart(Z\_8H, 1, N, Z);

CopyMatrixPart(MT\_2H\_1, first, last, MT);

CopyMatrixPart(MT\_2H\_1, GetFirst(1), GetLast(1), MT);

end DataT5;

T1.DataT3(Z\_8H => Z,

MT\_H => MT,

ME\_4H\_2\_4\_6 => ME);

CopyVectorPart(Z, first, last, S);

SortH(S, first, last);

accept SortResult (S\_H : out Vector) do

CopyVectorPart(S, first, last, S\_H);

end SortResult;

accept DataT1 (S\_8H : in Vector; R\_8H : in Vector; d : in Integer; MX\_2H\_5 : in Matrix) do

CopyVectorPart(S\_8H, 1, N, S);

CopyVectorPart(R\_8H, 1, N, R);

d\_in := d;

CopyMatrixPart(MX\_2H\_5, first, last, MX);

CopyMatrixPart(MX\_2H\_5, GetFirst(5), GetLast(5), MX);

end DataT1;

T5.DataT3(S\_8H => S,

R\_8H => R,

d => d\_in,

MX\_H => MX,

ME\_H => ME);

CalcMinPart(R, MT, first, last, Y, m3\_in);

accept MinPartResult (Y\_H : out Vector; m3 : out Integer) do

CopyVectorPart(Y, first, last, Y\_H);

m3 := m3\_in;

end MinPartResult;

T1.MinResult(Y\_8H => Y,

m => m\_in);

accept MinResult (Y\_8H : out Vector; m : out Integer) do

CopyVectorPart(Y, 1, N, Y\_8H);

m := m\_in;

end MinResult;

CalcResult(Y, Z, S, ME, MX, d\_in, m\_in, first, last, A);

T1.Result(A\_H => A);

accept Result (A\_2H\_1 : out Vector) do

CopyVectorPart(A, first, last, A\_2H\_1);

CopyVectorPart(A, GetFirst(1), GetLast(1), A\_2H\_1);

end Result;

Put\_Line("Task " & Integer'Image(taskNumber) & " finished");

end T3;

task body T4 is

Z, S, R, A, Y: Vector := new Vector\_t;

MT, ME, MX: Matrix := new Matrix\_t;

taskNumber: Integer := 4;

first, last, d\_in, m4\_in, m34\_in, m36\_in, m\_in: Integer;

begin

Put\_Line("Task " & Integer'Image(taskNumber) & " started...");

first := GetFirst(taskNumber);

last := GetLast(taskNumber);

accept DataT6 (Z\_8H : in Vector; MT\_2H\_2 : in Matrix) do

CopyVectorPart(Z\_8H, 1, N, Z);

CopyMatrixPart(MT\_2H\_2, first, last, MT);

CopyMatrixPart(MT\_2H\_2, GetFirst(2), GetLast(2), MT);

end DataT6;

T2.DataT4(Z\_8H => Z,

MT\_H => MT);

CopyVectorPart(Z, first, last, S);

SortH(S, first, last);

T3.SortResult(S\_H => S);

MergeSort(S, GetFirst(3), GetLast(3), last);

T6.SortResult(S\_2H\_5 => S);

MergeSort(S, GetFirst(3), GetLast(3), GetLast(6));

accept SortResult (S\_4H\_3\_5\_6 : out Vector) do

CopyVectorPart(S, GetFirst(3), GetLast(6), S\_4H\_3\_5\_6);

end SortResult;

accept DataT2 (S\_8H : in Vector; R\_8H : in Vector; d : in Integer; MX\_2H\_6 : in Matrix; ME\_2H\_6 : in Matrix) do

CopyVectorPart(S\_8H, 1, N, S);

CopyVectorPart(R\_8H, 1, N, R);

d\_in := d;

CopyMatrixPart(MX\_2H\_6, first, last, MX);

CopyMatrixPart(MX\_2H\_6, GetFirst(6), GetLast(6), MX);

CopyMatrixPart(ME\_2H\_6, first, last, ME);

CopyMatrixPart(ME\_2H\_6, GetFirst(6), GetLast(6), ME);

end DataT2;

T6.DataT4(S\_8H => S,

R\_8H => R,

d => d\_in,

MX\_H => MX,

ME\_H => ME);

CalcMinPart(R, MT, first, last, Y, m4\_in);

T3.MinPartResult(Y\_H => Y,

m3 => m34\_in);

if m4\_in < m34\_in then

m34\_in := m4\_in;

end if;

T6.MinPartResult(Y\_2H\_5 => Y,

m56 => m36\_in);

if m34\_in < m36\_in then

m36\_in := m34\_in;

end if;

accept MinPartResult (Y\_4H\_3\_5\_6 : out Vector; m36 : out Integer) do

CopyVectorPart(Y, GetFirst(3), GetLast(6), Y\_4H\_3\_5\_6);

m36 := m36\_in;

end MinPartResult;

T2.MinResult(Y\_8H => Y,

m => m\_in);

accept MinResult (Y\_8H : out Vector; m : out Integer) do

CopyVectorPart(Y, 1, N, Y\_8H);

m := m\_in;

end MinResult;

CalcResult(Y, Z, S, ME, MX, d\_in, m\_in, first, last, A);

T2.Result(A\_H => A);

accept Result (A\_2H\_2 : out Vector) do

CopyVectorPart(A, first, last, A\_2H\_2);

CopyVectorPart(A, GetFirst(2), GetLast(2), A\_2H\_2);

end Result;

Put\_Line("Task " & Integer'Image(taskNumber) & " finished");

end T4;

task body T5 is

Z, S, R, A, Y: Vector := new Vector\_t;

MT, ME, MX: Matrix := new Matrix\_t;

taskNumber: Integer := 5;

first, last, d\_in, m5\_in, m\_in: Integer;

begin

Put\_Line("Task " & Integer'Image(taskNumber) & " started...");

first := GetFirst(taskNumber);

last := GetLast(taskNumber);

accept DataT6 (Z\_8H : in Vector; MT\_3H\_1\_3 : in Matrix) do

CopyVectorPart(Z\_8H, 1, N, Z);

CopyMatrixPart(MT\_3H\_1\_3, first, last, MT);

CopyMatrixPart(MT\_3H\_1\_3, GetFirst(1), GetLast(1), MT);

CopyMatrixPart(MT\_3H\_1\_3, GetFirst(3), GetLast(3), MT);

end DataT6;

T3.DataT5(Z\_8H => Z,

MT\_2H\_1 => MT);

CopyVectorPart(Z, first, last, S);

SortH(S, first, last);

accept SortResult (S\_H : out Vector) do

CopyVectorPart(S, first, last, S\_H);

end SortResult;

accept DataT3 (S\_8H : in Vector; R\_8H : in Vector; d : in Integer; MX\_H : in Matrix; ME\_H : in Matrix) do

CopyVectorPart(S\_8H, 1, N, S);

CopyVectorPart(R\_8H, 1, N, R);

d\_in := d;

CopyMatrixPart(MX\_H, first, last, MX);

CopyMatrixPart(ME\_H, first, last, ME);

end DataT3;

CalcMinPart(R, MT, first, last, Y, m5\_in);

accept MinPartResult (Y\_H : out Vector; m5 : out Integer) do

CopyVectorPart(Y, first, last, Y\_H);

m5 := m5\_in;

end MinPartResult;

T3.MinResult(Y\_8H => Y,

m => m\_in);

CalcResult(Y, Z, S, ME, MX, d\_in, m\_in, first, last, A);

T3.Result(A\_2H\_1 => A);

T6.Result(A\_3H\_2\_4 => A);

if N <= 12 then

OutputVector(A);

end if;

Put\_Line("Task " & Integer'Image(taskNumber) & " finished");

end T5;

task body T6 is

Z, S, R, A, Y: Vector := new Vector\_t;

MT, ME, MX: Matrix := new Matrix\_t;

taskNumber: Integer := 6;

first, last, d\_in, m6\_in, m56\_in, m\_in: Integer;

begin

Put\_Line("Task " & Integer'Image(taskNumber) & " started...");

first := GetFirst(taskNumber);

last := GetLast(taskNumber);

InputVector(Z);

InputMatrix(MT);

T5.DataT6(Z\_8H => Z,

MT\_3H\_1\_3 => MT);

T4.DataT6(Z\_8H => Z,

MT\_2H\_2 => MT);

CopyVectorPart(Z, first, last, S);

SortH(S, first, last);

T5.SortResult(S\_H => S);

MergeSort(S, GetFirst(5), GetLast(5), last);

accept SortResult (S\_2H\_5 : out Vector) do

CopyVectorPart(S, GetFirst(5), last, S\_2H\_5);

end SortResult;

accept DataT4 (S\_8H : in Vector; R\_8H : in Vector; d : in Integer; MX\_H : in Matrix; ME\_H : in Matrix) do

CopyVectorPart(S\_8H, 1, N, S);

CopyVectorPart(R\_8H, 1, N, R);

d\_in := d;

CopyMatrixPart(MX\_H, first, last, MX);

CopyMatrixPart(ME\_H, first, last, ME);

end DataT4;

CalcMinPart(R, MT, first, last, Y, m6\_in);

T5.MinPartResult(Y\_H => Y,

m5 => m56\_in);

if m6\_in < m56\_in then

m56\_in := m6\_in;

end if;

accept MinPartResult (Y\_2H\_5 : out Vector; m56 : out Integer) do

CopyVectorPart(Y, GetFirst(5), last, Y\_2H\_5);

m56 := m56\_in;

end MinPartResult;

T4.MinResult(Y\_8H => Y,

m => m\_in);

CalcResult(Y, Z, S, ME, MX, d\_in, m\_in, first, last, A);

T4.Result(A\_2H\_2 => A);

accept Result (A\_3H\_2\_4 : out Vector) do

CopyVectorPart(A, first, last, A\_3H\_2\_4);

CopyVectorPart(A, GetFirst(2), GetLast(2), A\_3H\_2\_4);

CopyVectorPart(A, GetFirst(4), GetLast(4), A\_3H\_2\_4);

end Result;

Put\_Line("Task " & Integer'Image(taskNumber) & " finished");

end T6;

begin

null;

end RunTasks;

begin

Get(ch);

StartTime := Clock;

RunTasks;

end Main;

#### Файл data.ads

package Data is

N: Integer := 6;

P: Integer := 6;

H: Integer := N / P;

type Vector\_t is array(1..N) of Integer;

type Vector is access Vector\_t;

type Matrix\_t is array(1..N) of Vector;

type Matrix is access Matrix\_t;

procedure InputVector(V: out Vector);

procedure InputMatrix(M: out Matrix);

procedure OutputVector(V : in Vector);

procedure CopyVectorPart(source: in Vector; first, last: Integer; result: out Vector);

procedure CopyMatrixPart(source: in MAtrix; first, last: Integer; result: out Matrix);

procedure SortH(V : in out Vector; first, last: Integer);

procedure MergeSort(V: in out Vector; first, firstEnd, last: Integer);

procedure CalcMinPart(R: in Vector; MT: in Matrix; first, last: Integer; Y: out Vector; m: out Integer);

procedure CalcResult(Y, Z, S: in Vector; ME, MX: in Matrix; d, m, first, last: in Integer; A: out Vector);

function GetFirst(taskNumber: Integer) return Integer;

function GetLast(taskNumber: Integer) return Integer;

end Data;

#### Файл data.adb

with Ada.INTEGER\_TEXT\_IO; use Ada.INTEGER\_TEXT\_IO;

with Ada.Text\_IO; use Ada.Text\_IO;

package body Data is

procedure InputVector(V: out Vector) is

begin

for i in 1 .. N loop

V(i) := 1;

end loop;

end InputVector;

procedure InputMatrix(M: out Matrix) is

begin

for i in 1 .. N loop

M(i) := new Vector\_t;

for j in 1 .. N loop

M(i)(j) := 1;

end loop;

end loop;

end InputMatrix;

procedure CalcResult(Y, Z, S: in Vector; ME, MX: in Matrix; d, m, first, last: in Integer; A: out Vector) is

begin

for i in first..last loop

A(i) := 0;

for j in 1..N loop

A(i) := A(i) + Y(j)\*ME(j)(i) + m\*Z(j)\*MX(j)(i);

end loop;

A(i) := A(i) + S(i)\*d;

end loop;

end CalcResult;

procedure OutputVector(V : in Vector) is

begin

for i in 1..N loop

Ada.Integer\_Text\_IO.Put ( V(i) );

end loop;

Put\_Line("");

end OutputVector;

procedure CopyVectorPart(source: in Vector; first, last: Integer; result: out Vector) is

begin

if result = null then

result := new Vector\_t;

end if;

for i in first..last loop

result(i) := source(i);

end loop;

end;

procedure CopyMatrixPart(source: in Matrix; first, last: Integer; result: out Matrix) is

begin

if result = null then

result := new Matrix\_t;

end if;

for i in 1..N loop

CopyVectorPart(source(i), first, last, result(i));

end loop;

end;

procedure SortH(V : in out Vector; first, last: Integer) is

buf: Integer;

begin

for i in first..last loop

for j in i..last loop

if V(i) > V(j) then

buf := V(i);

V(i) := V(j);

V(j) := buf;

end if;

end loop;

end loop;

end SortH;

procedure MergeSort(V: in out Vector; first, firstEnd, last: Integer) is

res: Vector := new Vector\_t;

i1: Integer := first;

i2: Integer := firstEnd+1;

i: integer := 1;

begin

while (i1 <= firstEnd or i2 <= last) loop

if V(i1) < V(i2) then

res(i) := V(i1);

i1 := i1 + 1;

if (i1 > firstEnd) then

while (i2 <= last) loop

i := i+1;

res(i) := V(i2);

i2 := i2 + 1;

end loop;

end if;

else

res(i) := V(i2);

i2 := i2 + 1;

if (i2> last) then

while (i1 <= firstEnd) loop

i := i+1;

res(i) := V(i1);

i1 := i1 + 1;

end loop;

end if;

end if;

i := i+1;

end loop;

--Write result of sort to the input vector

i1 := 1;

for j in first..last loop

V(j) := res(i1);

i1 := i1+1;

end loop;

end MergeSort;

function GetFirst(taskNumber: Integer) return Integer is

begin

return (taskNumber - 1) \* H + 1;

end GetFirst;

function GetLast(taskNumber: Integer) return Integer is

begin

return taskNumber \* H;

end GetLast;

procedure CalcMinPart(R: in Vector; MT: in Matrix; first, last: Integer; Y: out Vector; m: out Integer) is

begin

m := Integer'Last;

for i in first..last loop

if R(i) < m then

m := R(i);

end if;

Y(i) := 0;

for j in 1..N loop

Y(i) := Y(i) + R(j)\*MT(j)(i);

end loop;

end loop;

end CalcMinPart;

end Data;