МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №5

з дисципліни «Паралельні та розподілені обчислення
Паралельне програмування-2»

Тема: «Ада. Захищені модулі»

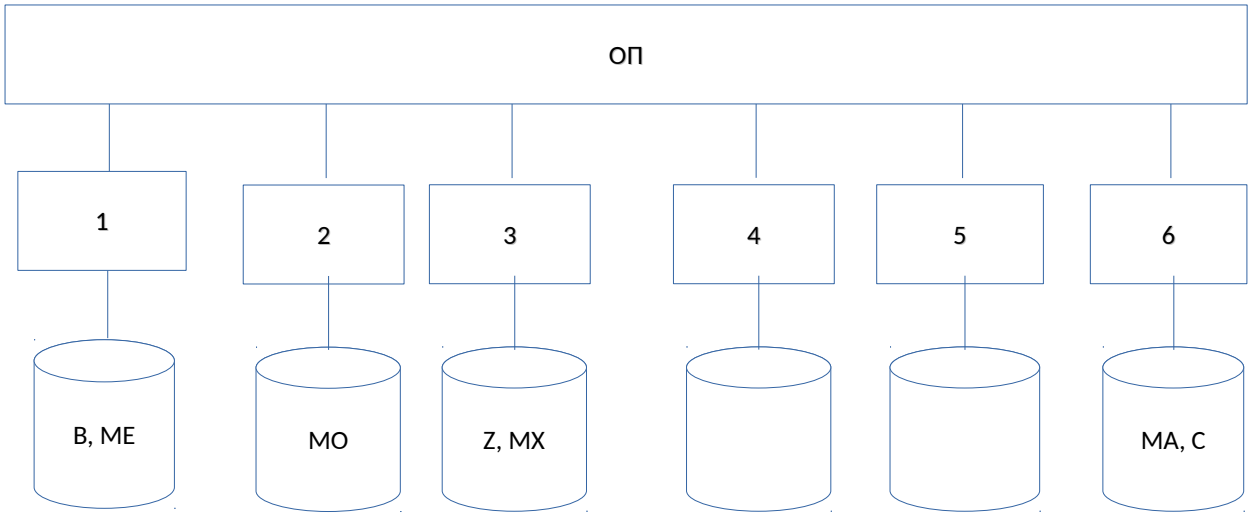
Виконав:
студент 3-го курсу
групи ІП-42
з номер заліковки 4206
Дзюба Влад

Київ – 2016

Завдання

Мета роботи: розробка програми для ПКС зі СП
Мова програмування: Ада
Засоби організації взаємодії процесів: монітори (захищені модулі мови Ада).

Варіант



$$MA = min(Z) \cdot MO + (B \cdot C)(MX \cdot ME)$$

Математичний паралельний алгоритм:

1. $z_i = min(Z_h), i \in [0..P-1]$
2. $z = min(z, z_i), i \in [0..P-1]$
3. $d_i = B_h \cdot C_h, i \in [0..P-1]$
4. $d = \sum_{0}^{P-1} d_i$
5. $MA_h = z \cdot MO_h + d \cdot (MX \cdot ME_h)$

CP: z, d, MX.

Алгоритм для кожного процесу:

T_1:		T_2:	
1. Ввести B.		1. Ввести MO.	
2. Сигнал про завершення вводу B.	S_1	2. Сигнал про завершення вводу MO.	S_1
3. Ввести ME.			
4. Сигнал про завершення вводу ME.	S_2	3. Чекати сигнал про завершення вводу Z з T_3 .	$W_{3,1}$
5. Чекати сигнал про завершення вводу Z з T_3 .	$W_{3,1}$	4. Обрахувати $z_2=min(Z_h)$	
6. Обрахувати $z_1=min(Z_h)$		5. Обрахувати $z=min(z,z_2)$	$КУ\,1$
7. Обрахувати $z=min(z,z_1)$	$КУ\,1$	6. Чекати сигнал про завершення вводу B з T_1 .	$W_{1,2}$
8. Чекати сигнал про завершення вводу C з T_6 .	$W_{6,2}$	7. Чекати сигнал про завершення вводу C з T_6 .	$W_{6,3}$
9. Обрахувати $d_1=B_h\cdot C_h$		8. Обрахувати $d_2=B_h\cdot C_h$	
10. Обрахувати $d=d+d_1$	$КУ\,2$	9. Обрахувати $d=d+d_1$	$КУ\,2$
11. Сигнал про завершення обрахунку z та d до всіх потоків.	S_3	10. Сигнал про завершення обрахунку z та d до всіх потоків.	S_2
12. Чекати сигнал про завершення обрахунку z та d з усіх потоків.	W_3	11. Чекати сигнал про завершення обрахунку z та d з усіх потоків.	W_4
13. Чекати сигнал про завершення вводу MO з T_2 .	$W_{2,4}$	12. Чекати сигнал про завершення вводу ME з T_1 .	$W_{1,5}$
14. Чекати сигнал про завершення вводу MX з T_3 .	$W_{3,5}$	13. Чекати сигнал про завершення вводу MX з T_3 .	$W_{3,6}$
15. Копіювати $z_1=z$	$КУ\,2$	14. Копіювати $z_2=z$	$КУ\,2$
16. Копіювати $d_1=d$	$КУ\,3$	15. Копіювати $d_2=d$	$КУ\,3$
17. Копіювати $MX_1=MX$	$КУ\,4$	16. Копіювати $MX_2=MX$	$КУ\,4$
18. Обрахувати $MA_h=z_1\cdot MO_h+d_1\cdot(MX_1\cdot ME_h)$.		17. Обрахувати $MA_h=z_2\cdot MO_h+d_2\cdot(MX_2\cdot ME_h)$.	
19. Сигнал про завершення обрахунку MA_h .	S_4	18. Сигнал про завершення обрахунку MA_h .	S_3

T_3:		T_4:	
1. Ввести Z.		1. Чекати сигнал про завершення вводу Z з T_3 .	$W_{3,1}$
2. Сигнал про завершення вводу Z.	S_1	2. Обрахувати $z_4 = min(Z_h)$	
3. Ввести MX.		3. Обрахувати $z = min(z, z_4)$	$КУ\,1$
4. Сигнал про завершення вводу MX.	S_2	4. Чекати сигнал про завершення вводу B з T_1 .	$W_{1,2}$
5. Обрахувати $z_3 = min(Z_h)$		5. Чекати сигнал про завершення вводу C з T_6 .	$W_{6,3}$
6. Обрахувати $z = min(z, z_3)$	$КУ\,1$	6. Обрахувати $d_4 = B_h \cdot C_h$	
7. Чекати сигнал про завершення вводу B з T_1 .	$W_{1,1}$	7. Обрахувати $d = d + d_4$	$КУ\,2$
8. Чекати сигнал про завершення вводу C з T_6 .	$W_{6,2}$	8. Сигнал про завершення обрахунку z та d до всіх потоків.	S_1
9. Обрахувати $d_3 = B_h \cdot C_h$		9. Чекати сигнал про завершення обрахунку z та d з усіх потоків.	W_4
10. Обрахувати $d = d + d_3$	$КУ\,2$	10. Чекати сигнал про завершення вводу ME з T_1 .	$W_{1,5}$
11. Сигнал про завершення обрахунку z та d до всіх потоків.	S_3	11. Чекати сигнал про завершення вводу MO з T_2 .	$W_{2,6}$
12. Чекати сигнал про завершення обрахунку z та d з усіх потоків.	W_3	12. Чекати сигнал про завершення вводу MX з T_3 .	$W_{3,7}$
13. Чекати сигнал про завершення вводу MO з T_2 .	$W_{2,4}$	13. Копіювати $z_4 = z$	$КУ\,2$
14. Чекати сигнал про завершення вводу ME з T_1 .	$W_{1,5}$	14. Копіювати $d_4 = d$	$КУ\,3$
15. Копіювати $z_3 = z$		15. Копіювати $MX_4 = MX$	$КУ\,4$
16. Копіювати $d_3 = d$	$КУ\,2$	16. Обрахувати $MA_h = z_4 \cdot MO_h + d_4 \cdot (MX_4 \cdot ME_h)$.	
17. Копіювати $MX_3 = MX$	$КУ\,3$	17. Сигнал про завершення обрахунку MA_h .	S_2
18. Обрахувати $MA_h = z_3 \cdot MO_h + d_3 \cdot (MX_3 \cdot ME_h)$.	$КУ\,4$		
19. Сигнал про завершення обрахунку MA_h .	S_4		

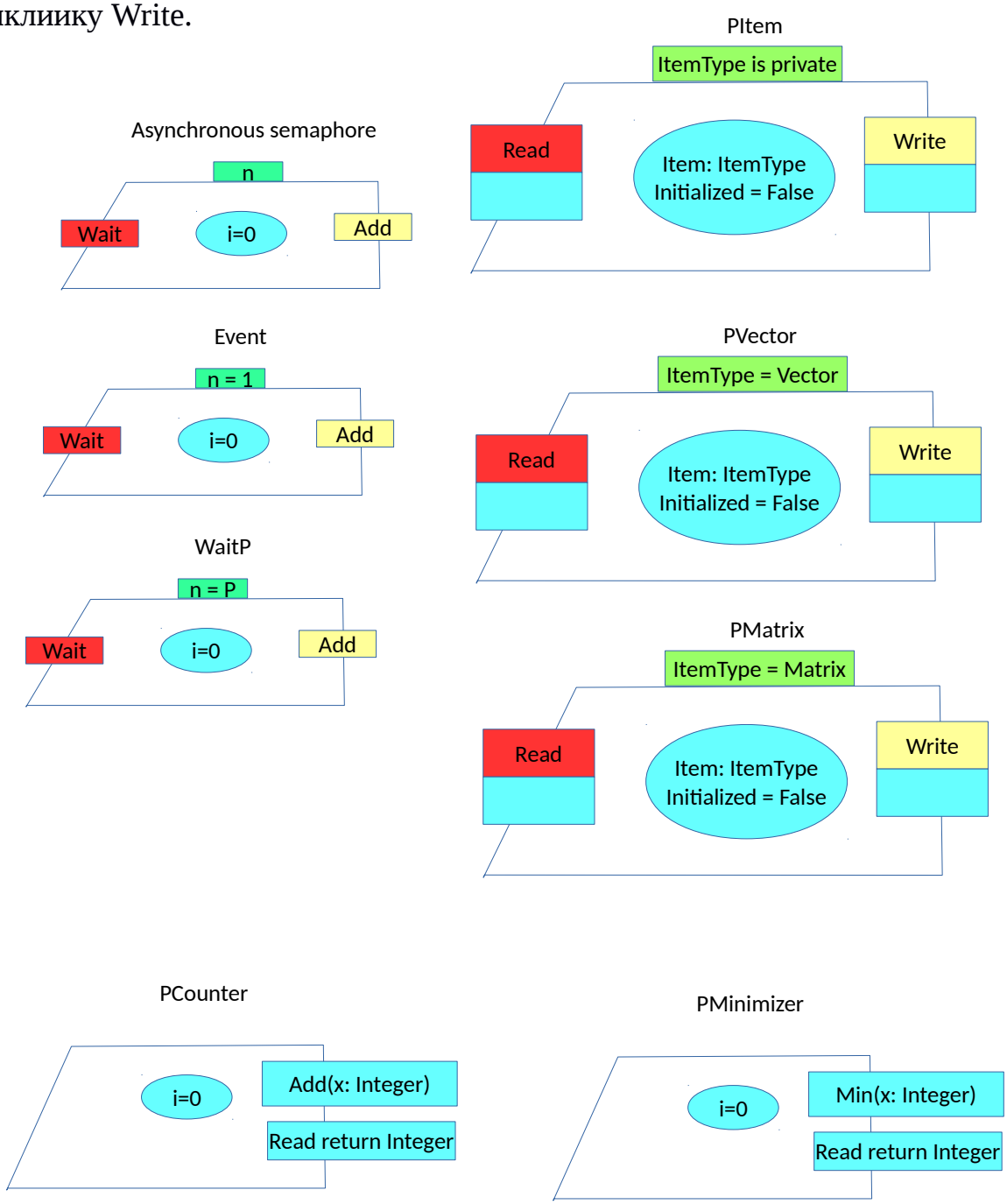
T_5:		T_6:	
1. Чекати сигнал про завершення вводу Z з T_3 .	$W_{3,1}$	1. Ввести C.	
2. Обрахувати $z_5 = min(Z_h)$		2. Сигнал про завершення вводу C.	S_1
3. Обрахувати $z = min(z, z_5)$	$КУ\,1$	3. Чекати сигнал про завершення вводу Z з T_3 .	$W_{3,1}$
4. Чекати сигнал про завершення вводу B з T_1 .	$W_{1,2}$	4. Обрахувати $z_6 = min(Z_h)$	
5. Чекати сигнал про завершення вводу C з T_6 .	$W_{6,3}$	5. Обрахувати $z = min(z, z_6)$	$КУ\,1$
6. Обрахувати $d_5 = B_h \cdot C_h$		6. Чекати сигнал про завершення вводу B з T_1 .	$W_{1,2}$
7. Обрахувати $d = d + d_5$		7. Обрахувати $d_6 = B_h \cdot C_h$	
8. Сигнал про завершення обрахунку z та d до всіх потоків.	$КУ\,2$	8. Обрахувати $d = d + d_6$	$КУ\,2$
9. Чекати сигнал про завершення обрахунку z та d з усіх потоків.	S_1	9. Сигнал про завершення обрахунку z та d до всіх потоків.	S_2
10. Чекати сигнал про завершення вводу ME з T_1 .	W_4	10. Чекати сигнал про завершення обрахунку z та d з усіх потоків.	W_3
11. Чекати сигнал про завершення вводу MO з T_2 .	$W_{1,5}$	11. Чекати сигнал про завершення вводу ME з T_1 .	$W_{1,4}$
12. Чекати сигнал про завершення вводу MX з T_3 .	$W_{2,6}$	12. Чекати сигнал про завершення вводу MO з T_2 .	$W_{2,5}$
13. Копіювати $z_5 = z$	$W_{3,7}$	13. Чекати сигнал про завершення вводу MX з T_3 .	$W_{3,6}$
14. Копіювати $d_5 = d$	$КУ\,2$	14. Копіювати $z_6 = z$	$КУ\,2$
15. Копіювати $MX_5 = MX$	$КУ\,3$	15. Копіювати $d_6 = d$	$КУ\,3$
16. Обрахувати $MA_h = z_5 \cdot MO_h + d_5 \cdot (MX_5 \cdot ME_h)$.	$КУ\,4$	16. Копіювати $MX_6 = MX$	$КУ\,4$
17. Сигнал про завершення обрахунку MA_h .	S_2	17. Обрахувати $MA_h = z_6 \cdot MO_h + d_6 \cdot (MX_6 \cdot ME_h)$.	
		18. Чекати сигнал про завершення обрахунку MA_h .	W_7
		19. Вивести MA	

Структурна схема взаємодії процесів

У програмі використовуються захищені типи Event та WaitP, які є реалізаціями налаштовуємого захищеного типу AsynchronousSemaphore. Змінними типу Event є ZIsRead, BIsRead, CIsRead, MEIsRead, MXIsRead, MOIsRead, які використовуються для синхронізації вводу данних з різних потоків. Змінним типу WaitN є zdCalculationEnd, MAIsCalculated, які використовуються для синхронізацію після обчислинь z та d, та MA відповідно.

Захищенні типи PCounter та PMaximizer використовуються для знаходження суми та максимума відповідно. Змінною PCounter є zInt, а змінною PMaximizer — d.

Для доступу до інших спільних ресурсів використовується PVector та PMatrix. Вони є реалізаціями настроюмого захищеного типу PItem. Їх особливістю є те, що операції Write, Read є також сигналом та чеканням сигнал. Таким чином Read контролюється флагом initialized, який ставиться в True тільки після виклиику Write.



Лістінг програми:

main.adb

```
--
-- Dzyuba Vlad, IP-42
--
with Ada.Text_IO; use Ada.Text_IO;
procedure main is
  N: constant Integer := 12;
  P: constant Integer := 6;
  H: constant Integer := N / P;
  subtype Index is Integer range 1..N;
  type Vector is array (Index) of Integer;
  type Matrix is array (Index) of Vector;

  generic
    type ItemType is private;
  package PItemPkg is
    protected type PItem is
      entry Read(outItem: in out ItemType);
      procedure Write(inItem: ItemType);
    private
      initialized: Boolean := False;
    end;
  end;

  package body PItemPkg is
    item: ItemType;

    protected body PItem is
      entry Read(outItem: in out ItemType) when initialized is
      begin
        outItem := item;
      end;

      procedure Write(inItem: ItemType) is
      begin
        item := inItem;
        initialized := True;
      end;
    end;
  end;

  protected type PCounter is
    procedure Add(x: Integer);
    function Read return Integer;
    private
      i: Integer := 0;
  end;

  protected body PCounter is
    procedure Add(x: Integer) is
    begin
      i := i + x;
```

```
    end;

    function Read return Integer is
    begin
      return i;
    end;
  end;

  protected type PMinimizer is
    procedure Min(x: Integer);
    function Read return Integer;
    private
      i: Integer := 0;
  end;

  protected body PMinimizer is
    procedure Min(x: Integer) is
    begin
      if x > i then
        i := x;
      end if;
    end;

    function Read return Integer is
    begin
      return i;
    end;
  end;

  package PVectorPkg is new PItemPkg(Vector);
  subtype PVector is PVectorPkg.PItem;
  package PMatrixPkg is new PItemPkg(Matrix);
  subtype PMatrix is PMatrixPkg.PItem;

  zInt: PMinimizer;
  d: PCounter;
  B, C, Z: Vector;
  MX: PMatrix;
  ME, MA, MO: Matrix;

generic
    n: Integer;
  package AsyncSemaphorePkg is
    protected type AsyncSemaphore is
      procedure Add;
      entry Wait;
      private
        i: Integer := 0;
    end;
  end;

  package body AsyncSemaphorePkg is
    protected body AsyncSemaphore is
      procedure Add is
```

```
    begin
      i := i + 1;
    end;

    entry Wait when i >= n is
    begin
      null;
    end;
  end;
end;

package EventPkg is new AsyncSemaphorePkg(1);
subtype Event is EventPkg.AsyncSemaphore;

package WaitPPkg is new AsyncSemaphorePkg(P);
subtype WaitP is WaitPPkg.AsyncSemaphore;

generic
  index: Integer;
package CalculationPkg is
  task type Calculation;
end;

ZIsRead, BIsRead, CIsRead, MEIsRead, MXIsRead, MOIsRead: Event;
zdCalculationEnd, MAIsCalculated: WaitP;

package body CalculationPkg is
  task body Calculation is
    xe, zIntI, dI: Integer;
    MXi: Matrix;
  begin
    ZIsRead.Wait;
    zIntI := Integer'Last;
    for i in index*H+1..(index + 1)*H loop
      if Z(i) < zIntI then
        zIntI := Z(i);
      end if;
    end loop;
    zInt.Min(zIntI);
    BIsRead.Wait;
    CIsRead.Wait;
    dI := 0;
    for i in index*H+1..(index + 1)*H loop
      dI := dI + B(i) * C(i);
    end loop;
    d.Add(dI);
    zdCalculationEnd.Add;
    zdCalculationEnd.Wait;
    zIntI := zInt.Read;
    dI := d.Read;
    MX.Read(MXi);
    MEIsRead.Wait;
    MOIsRead.Wait;
    for i in index*H+1..(index + 1)*H loop
```

```
      for j in 1..N loop
        xe := 0;
        for k in 1..N loop
          xe := xe + MXi(j)(k) * ME(k)(i);
        end loop;
        MA(i)(j) := zIntI * MO(j)(i) + dI * xe;
      end loop;
    end loop;
    MAIsCalculated.Add;
  end Calculation;
end;

function initVector return Vector is
  res: Vector;
begin
  for i in Index loop
    res(i) := 1;
  end loop;
  return res;
end;

function initMatrix return Matrix is
  mat: Matrix;
begin
  for i in Index loop
    mat(i) := initVector;
  end loop;
  return mat;
end;

task Read1;
task body Read1 is
  package CalculationIPkg is new CalculationPkg(0);
  calc: CalculationIPkg.Calculation;
begin
  Put_Line("Thread 1 has started");
  B := initVector;
  BIsRead.Add;
  ME := initMatrix;
  MEIsRead.Add;
end;

task Read2;
task body Read2 is
  package CalculationIPkg is new CalculationPkg(1);
  calc: CalculationIPkg.Calculation;
begin
  Put_Line("Thread 2 has started");
  MO := initMatrix;
  MOIsRead.Add;
end;

task Read3;
task body Read3 is
```

```
    package CalculationIPkg is new CalculationPkg(2);
    calc: CalculationIPkg.Calculation;
  begin
    Put_Line("Thread 3 has started");
    Z := initVector;
    ZIsRead.Add;
    MX.Write(initMatrix);
    MXIsRead.Add;
  end;

  task Read4;
  task body Read4 is
    package CalculationIPkg is new CalculationPkg(3);
    calc: CalculationIPkg.Calculation;
  begin
    Put_Line("Thread 4 has started");
  end;

  task Read5;
  task body Read5 is
    package CalculationIPkg is new CalculationPkg(4);
    calc: CalculationIPkg.Calculation;
  begin
    Put_Line("Thread 5 has started");
  end;

  task Read6;
  task body Read6 is
    package CalculationIPkg is new CalculationPkg(5);
    calc: CalculationIPkg.Calculation;
  begin
    Put_Line("Thread 6 has started");
    C := initVector;
    CIsRead.Add;
    MAisCalculated.Wait;
    if N < 20 then
      for i in 1..N loop
        for j in 1..N loop
          Put(Integer'Image(MA(i)(j)));
          Put(" ");
        end loop;
        Put_Line("");
      end loop;
    end if;
  end;
begin
  null;
end;
```