НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кафедра обчислювальної техніки \_\_\_\_\_\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА**

з дисципліни «Паралельні та розподілені обчислення»

(назва дисципліни)

на тему: «Розробка програмного забезпечення для паралельних комп’ютерних систем»

Студента (ки) 3 курсу групи

спеціальності

123 «Комп’ютерна інженерія»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник доцент Корочкін О.В.

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_

Київ - 2021 рік

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) інформатики та обчислювальної техніки

( повна назва )

Кафедра обчислювальної техніки

( повна назва )

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 123 «Комп’ютерна інженерія»

*(шифр і назва)*

***З А В Д А Н Н Я***

НА РГР СТУДЕНТУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(прізвище, ім’я, по батькові)*

1. Тема роботи «Розробка програмного забезпечення для паралельних

комп’ютерних систем»

керівник роботи Корочкін Олександр Володимирович к.т.н.**,** доцент

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 10 травня 2021 р.

3. Вхідні дані до роботи

- математична задача A = (B\*C)\*(Z\*MM)+D\*(MX\*MR)

- структури ПКС ОП та ПКС ЛП

- мова (бібліотека) для ПРГ1: Java (монітори)

- мова (бібліотека) для ПРГ2: Ada (механізм рандеву)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- огляд засобів програмування для ПКС

- розробка і тестування програми ПРГ1 для ПКС ОП

- розробка і тестування програми ПРГ2 для ПКС ЛП

5. Перелік графічного матеріалу

- структурна схема ПКС СП

- структурна схема ПКС ЛП

- схеми алгоритмів процесів і головної програми для ПРГ1

- схеми алгоритмів процесів і головної програми для ПРГ2.

7. Дата видачі завдання \_\_\_\_\_\_\_\_\_4 04 2021\_\_\_\_\_\_\_\_\_\_\_.

ЗМІСТ

ВСТУП…………………………………………………………………………… 5

РОЗДІЛ 1. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП……………………6

1.1 Огляд засобів для програмування в ПКС СП………………………..6

1.2 Розробка паралельного математичного алгоритму……………………6

1.3 Розробка алгоритмів процесів………………………………………….7

1.4 Розробка схеми взаємодії процесів…………………………………….9

1.5 Розробка програми ПРГ1……………………………………………….12

1.6 Тестування програми ПРГ1 …………………………………………….12

1.7 Висновки до розділу 1 …………………………………………………..16

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП ……………………18

2.1 Огляд засобів для програмування в ПКС ЛП………………………..18

2.2 Розробка паралельного математичного алгоритму………………….18

2.3 Розробка алгоритмів процесів…………………………………………19

2.4 Розробка схеми взаємодії процесів…………………………………..21

2.5 Розробка програми ПРГ2………………………………………………25

2.6 Тестування програми ПРГ2……………………………………………25

2.7 Висновки до розділу 2…………………………………………………31

ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ …………….……………………… 32

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………… 33

ДОДАТКИ……………………………………………………............................. 34

Додаток А …………………………………………………………………34

Додаток Б …………………………………………………………………44

**ВСТУП**

В даній розрахунковій роботі розроблено алгоритм обчислення заданого математичного виразу в ПРГ1 та ПРГ2.

Порівнюється прискорення виконання обчислень та ефективність двох паралельних алгоритмів.

Перший розділ присвячені огляду засобів ПРГ1 та розробці програми для обчислення математичної задачі в паралельній комп’ютерній системі зі спільною пам’яттю. Програмне забезпечення для комп’ютерної системи із спільною пам’яттю розроблено на мові java з використанням моніторів. Проведено тестування отриманого програмного продукту і зроблено висновки по його ефективності.

Другий розділ присвячені огляду засобів ПРГ2 та розробці програми для обчислення математичної задачі в паралельній комп’ютерній системі з локальною пам’яттю. Програмне забезпечення для комп’ютерної системи із спільною пам’яттю розроблено на мові ada з використанням механізму рандеву. Проведено тестування отриманого програмного продукту і зроблено висновки по його ефективності.

Лістинги розроблених програм наведено у додатках.

**РОЗДІЛ 1. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП**

У даному розділі розглянуто розробку та дослідження програми ПРГ1 для ПКС з СП.

**1.1 Огляд засобів для програмування в ПКС СП**

Для синхронізації потоків в Java використовуються монітори, які є високорівневим механізмом, що дозволяє одночасно тільки одному потоку виконувати блок коду, захищений монітором.

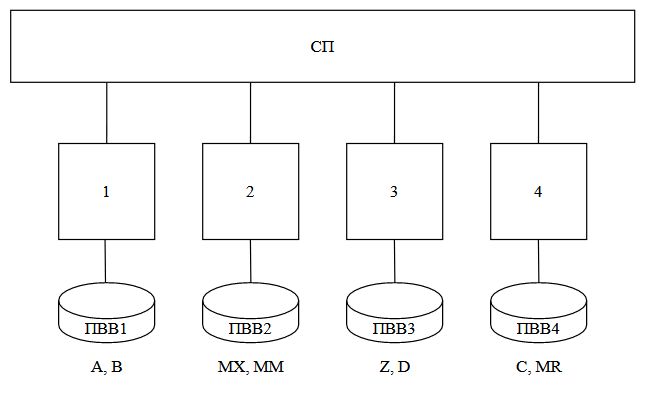
Синхронізація має кілька аспектів. Найбільш добре розуміється mutual exclusion - тільки один потік може володіти монітором, таким чином синхронізація на моніторі означає, що як тільки один потік входить в synchronized-блок, захищений монітором, ніякий інший потік не може увійти в блок, захищений цим монітором поки перший потік не вийде з synchronized-блоку. Синхронізація гарантує, що дані, записані в пам'ять до або всередині синхронізованого блоку, стають видимими для інших потоків, які синхронізуються на тому ж моніторі. Після виходу з синхронізованого блоку, звільняється (release) монітор, що має ефект скидання кеша в оперативну пам'ять, так що записи, зроблені нашим потоком, можуть бути видимими для інших потоків.

Читання-запис в поле є атомарною операцією, якщо поле оголошено volatile або захищене унікальним блокуванням, яке отримується перед будь-яким читанням-записом. Якщо всі спроби доступу до деякого полю захищені одним і тим самим блокуванням, то операції читання-запису цього поля є атомарними. Кожен потік, який звертається до volatile-полю, прочитає його значення перед тим як продовжити. volatile-поле негайно записується в пам'ять.

**1.2 Розробка паралельного математичного алгоритму**

Математичний вираз: A = (B\*C)\*(Z\*MM)+D\*(MX\*MR)

Мова програмування: Java (монітори)

Рисунок 1.1 - Структура ПКС СП

Паралельний алгоритм:

1. ai = (BH\*CH), i = 1..4.

2. a = a + ai, i = 1..4.

3. KH = (Z\*MMН)

4. AH = a\*K+DH\*(MX\*MRH)

Спільні ресурси: a, a, K, Z, MX.

**1.3 Розробка алгоритмів процесів**

Точки синхронізації,

Задача Т1 критичні ділянки

1. Сигнал Т2, T3, T4 про введення даних. S2,3,4-1

2. Очікування завершення обчислень в Т2, T3, T4. W2,3,4-1

3. Копіювання Z1 = Z, MX1 = MX КД

4. Обчислення а1 = (BH\*CH).

5. Обчислення а = a + a1. КД

6. Обчислення K = (Z1\*MMH) КД

7. Сигнал Т2, Т3, Т4 про завершення обчислення а та К. S2,3,4-2

8. Очікування завершення обчислень a та К в Т2, T3, T4. W2,3,4-2

9. Копіювання a1 = a, K1 = K КД

10. АH = а1\*K1+DH\*(MX1\*MRH)

11. Очікування сигналу про завершення обчислення А в Т2, Т3, Т4. W2,3,4-3

12. Вивід А.

Задача Т2

1. Сигнал Т1, T3, T4 про введення даних. S1,3,4-1

2. Очікування завершення обчислень в Т1, T3, T4. W1,3,4-1

3. Копіювання Z2 = Z, MX2 = MX КД

4. Обчислення а2 = (BH\*CH).

5. Обчислення а = a + a2. КД

6. Обчислення K = (Z2\*MMH) КД

7. Сигнал Т1, Т3, Т4 про завершення обчислення а та К. S1,3,4-2

8. Очікування завершення обчислень a та К в Т1, T3, T4. W1,3,4-2

9. Копіювання a2 = a, K2 = K КД

10. АH = а2\*K2+DH\*(MX2\*MRH)

11. Сигнал Т1 про завершення обчислення А. S1-3

Задача Т3

1. Сигнал Т1, T2, T4 про введення даних. S1,2,4-1

2. Очікування завершення обчислень в Т1, T2, T4. W1,2,4-1

3. Копіювання Z3 = Z, MX3 = MX КД

4. Обчислення а3 = (BH\*CH).

5. Обчислення а = a + a3. КД

6. Обчислення K = (Z3\*MMH) КД

7. Сигнал Т1, Т2, Т4 про завершення обчислення а та К. S1,2,4-2

8. Очікування завершення обчислень a та К в Т1, T2, T4. W1,2,4-2

9. Копіювання a3 = a, K3 = K КД

10. АH = а3\*K3+DH\*(MX3\*MRH)

11. Сигнал Т1 про завершення обчислення А. S1-3

Задача Т4

1. Сигнал Т1, T2, T3 про введення даних. S1,2,3-1

2. Очікування завершення обчислень в Т1, T2, T3. W1,2,3-1

3. Копіювання Z4 = Z, MX4 = MX КД

4. Обчислення а4 = (BH\*CH).

5. Обчислення а = a + a4. КД

6. Обчислення K = (Z4\*MMH) КД

7. Сигнал Т1, Т2, Т3 про завершення обчислення а та К. S1,2,3-2

8. Очікування завершення обчислень a та К в Т1, T2, T3. W1,2,3-2

9. Копіювання a4 = a, K4 = K КД

10. АH = а4\*K4+DH\*(MX4\*MRH)

11. Сигнал Т1 про завершення обчислення А. S1-3

**1.4 Розробка схеми взаємодії процесів**

На основі алгоритму процесів розроблена структурна схема монітору (рис. 1.3), що реалізована з допомогою захищеного модуля мови програмування Java.

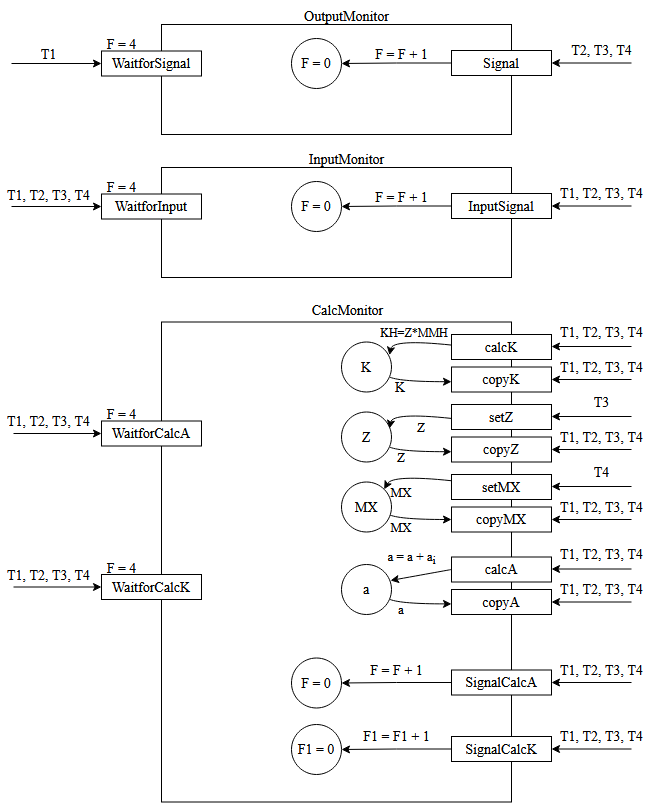
Для взаємодії процесів використовується захищені модулі: CalcMonitor, InputMonitor та OutputMonitor.

CalcMonitor відповідає за вирішення задачі взаємного виключення:

* copyZ - Копіювання спільного ресурсу Z потоками Т1, Т2, Т3, Т4.
* setZ - Введення спільного ресурсу Z потоком Т3.
* copyMX - Копіювання спільного ресурсу МX потоками Т1, Т2, Т3, Т4.
* setMX - Введення спільного ресурсу MX потоком Т4.
* copyK - Копіювання спільного ресурсу K потоками Т1, Т2, Т3, Т4.
* calcK - обчислення KH.
* copyA - Копіювання спільного ресурсу а потоками Т1, Т2, Т3, Т4.
* calcA - обчислення а = a + ai, i = 1..4 потоками Т1, Т2, Т3, Т4.
* SignalCalcA - сигнал про завершення обчислення а в потоках Т1, Т2, Т3, Т4.
* WaitforCalcA - очікування сигналів про завершення обчислення а в потоках Т1, Т2, Т3, Т4.
* WaitforCalcK - очікування сигналів про завершення обчислення KH в потоках Т1, Т2, Т3, Т4.
* SignalCalcK - сигнал про завершення обчислення KH в потоках Т1, Т2, Т3, Т4.

InputMonitor та OutputMonitor - за синхронізацію задач:

* WaitforSignal - очікування сигналів про завершення обчислення A в потоках Т1, Т3, Т4.
* Signal - сигнал про завершення обчислення A в потоках Т1, Т3, Т4.
* WaitforInput - очікування сигналів про завершення вводу в потоках Т1, Т2, Т3, Т4.
* InputSignal - сигнал про завершення вводу в потоках Т1, Т2, Т3, Т4.

Рисунок 1.2 – Структурна схема монітора для вирішення задачі синхронізації для ПРГ1

**1.5 Розробка програми ПРГ1**

Програма ПРГ1 згідно технічного завдання розроблена на мові програмування Java.

Для взаємодії процесів використовується концепція моніторів.

Програма складається з головного класу Main, який складається з:

* Визначення констант N, P, H.
* Процедури введення/виведення матриці.
* Специфікації та реалізації моніторів InputMonitor та OutputMonitor.
* Calculations – клас для операцій над матрицями та векторами

**1.6 Тестування програми ПРГ1**

Для тестування використовувалась паралельна обчислювальна система з наступними апаратними характеристиками:

* процесор: Intel Core i5-7200U, 4 ядра;
* оперативна пам’ять: DDR3 8 Гб;
* Операційна система: Microsoft Windows 10;

Для вимірювання часу виконання використовувалася функція currentTimeMillis мови Java.

Таблиця 1.1. Час виконання програми для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N/P | P=1 | P=2 | P=3 | P=4 |
| 900 | 9533 | 6476 | 3799 | 2784 |
| 1800 | 32527 | 21797 | 12476 | 9341 |
| 2400 | 92131 | 59516 | 36415 | 25706 |

Підрахунок коефіцієнту прискорення (КП) виконується за формулою КП = Т1 / Tp

Підрахунок коефіцієнту ефективності (КЕ) відбувається за формулою КЕ = КY / Р \* 100%

Таблиця 1.2. Значення Кп для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N/P | P=1 | P=2 | P=3 | P=4 |
| 900 | 1 | 1,472 | 2,509 | 3,423 |
| 1800 | 1 | 1,492 | 2,607 | 3,482 |
| 2400 | 1 | 1,548 | 2,530 | 3,584 |

Таблиця 1.3. Значення Ке для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N/P | P=1 | P=2 | P=3 | P=4 |
| 900 | 100% | 73% | 83% | 86% |
| 1800 | 100% | 74% | 87% | 87% |
| 2400 | 100% | 77% | 84% | 90% |

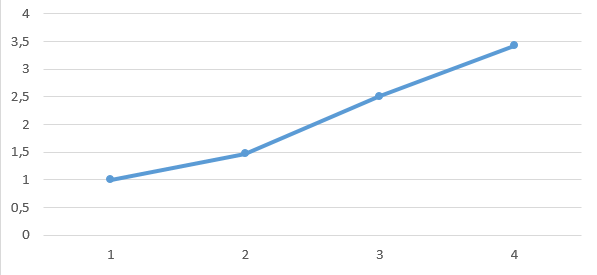


Рисунок 1.3 – Графік залежності коефіцієнту прискорення від кількості процесорів при N = 900

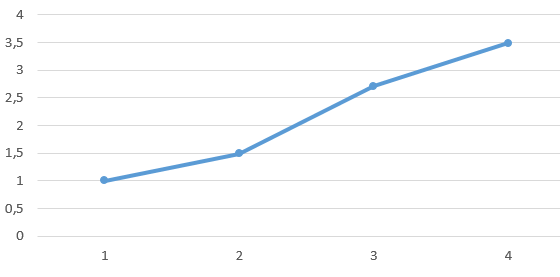


Рисунок 1.4 – Графік залежності коефіцієнту прискорення від кількості процесорів при N = 1800

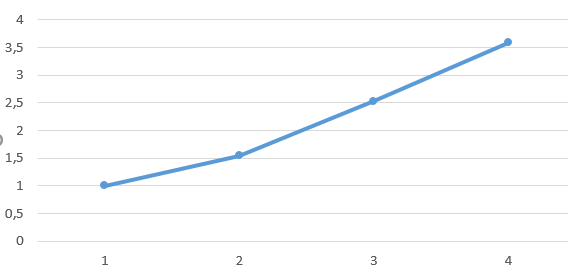


Рисунок 1.5 – Графік залежності коефіцієнту прискорення від кількості процесорів при N = 2400

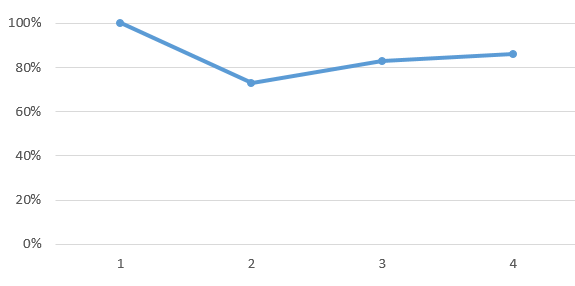
****

Рисунок 1.6 – Графік залежності коефіцієнту ефективності від кількості процесорів при N = 900

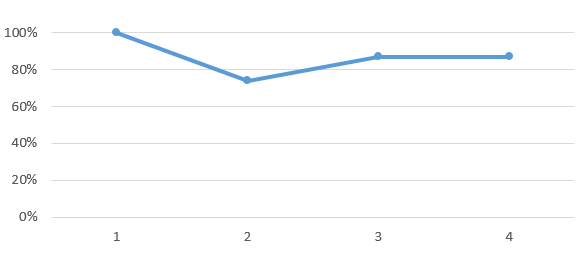
****

Рисунок 1.7 – Графік залежності коефіцієнту ефективності від кількості процесорів при N = 1800

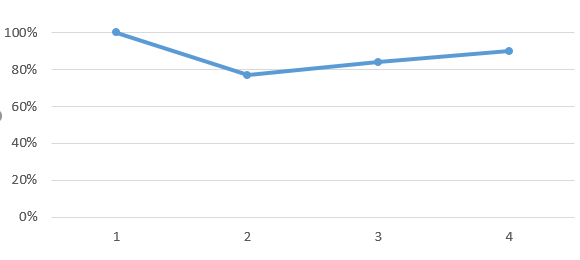
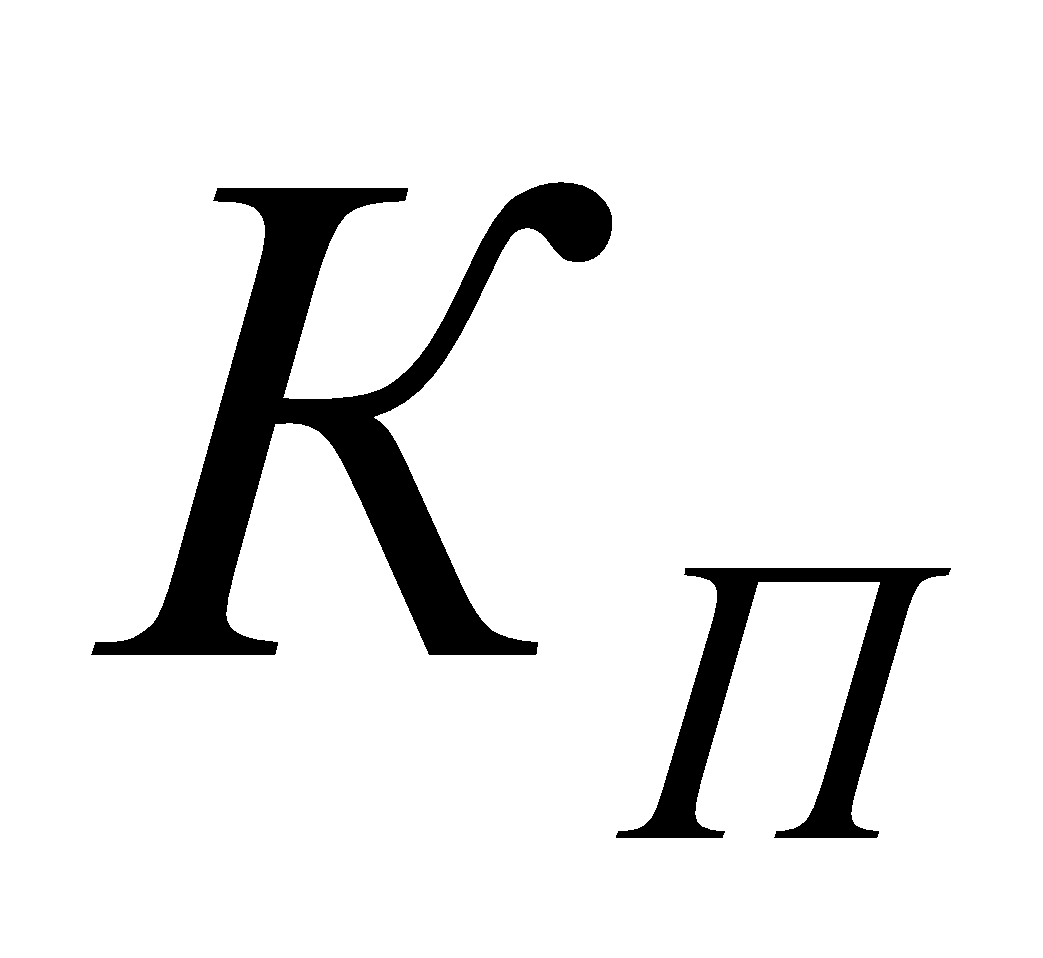
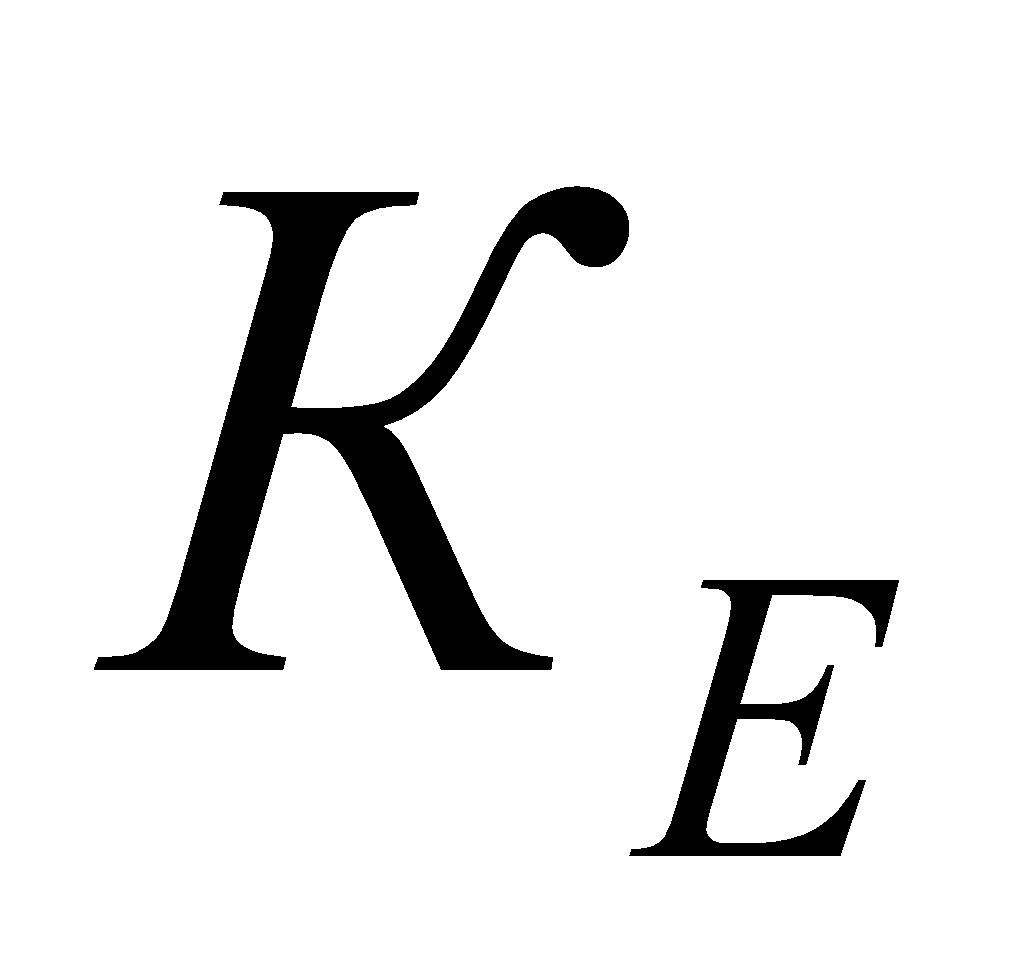
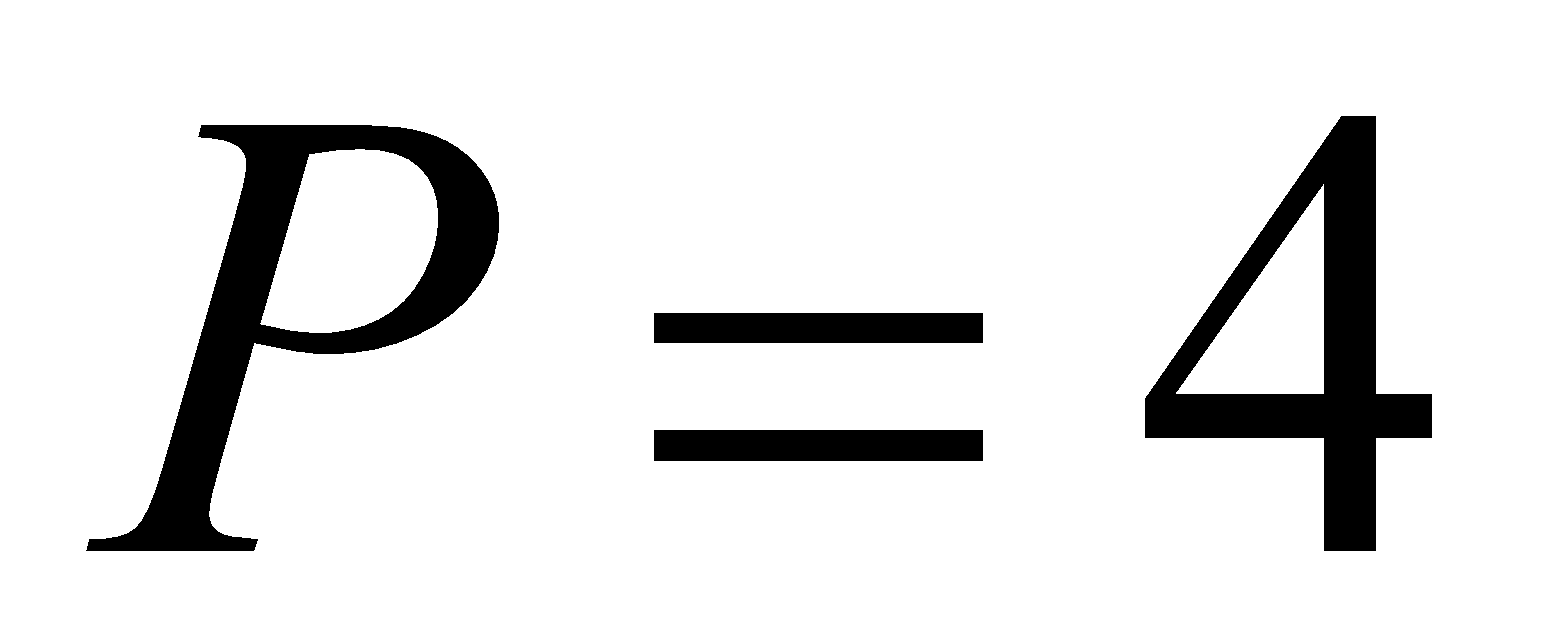
****

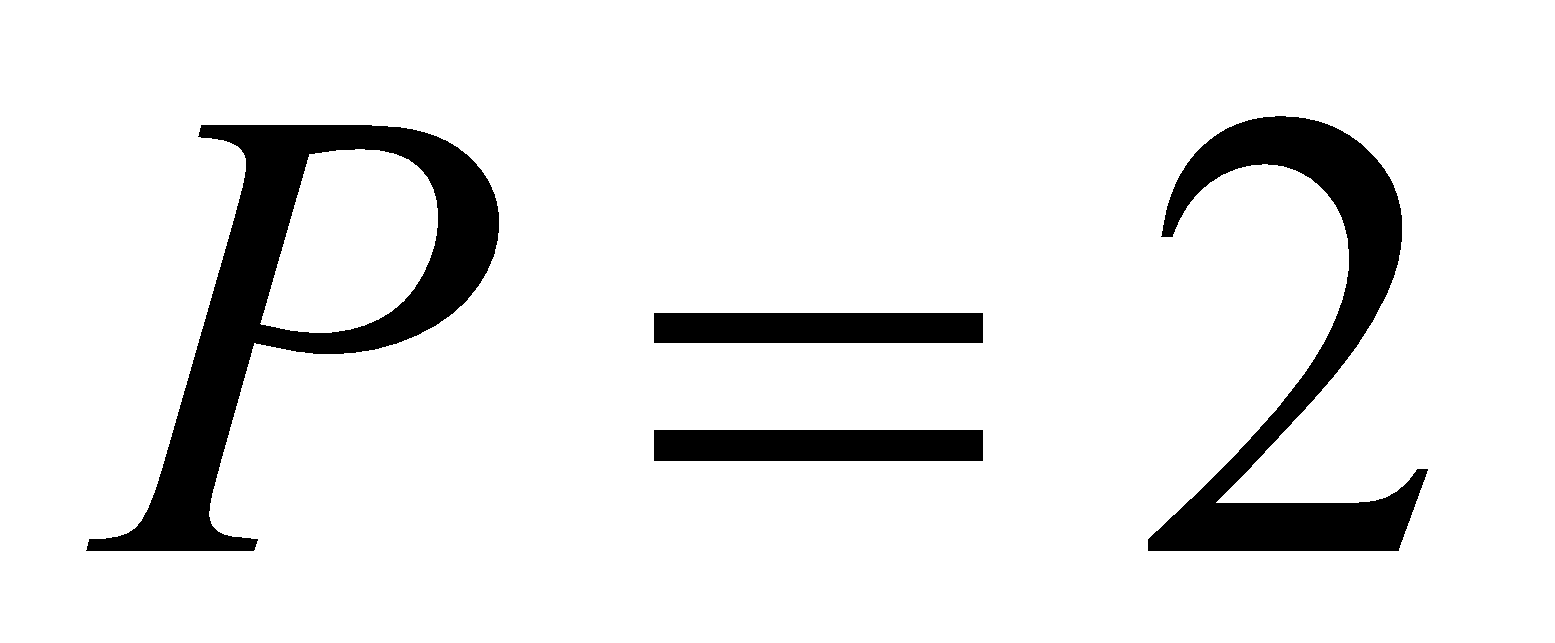
Рисунок 1.8 – Графік залежності коефіцієнту ефективності від кількості процесорів при N = 2400

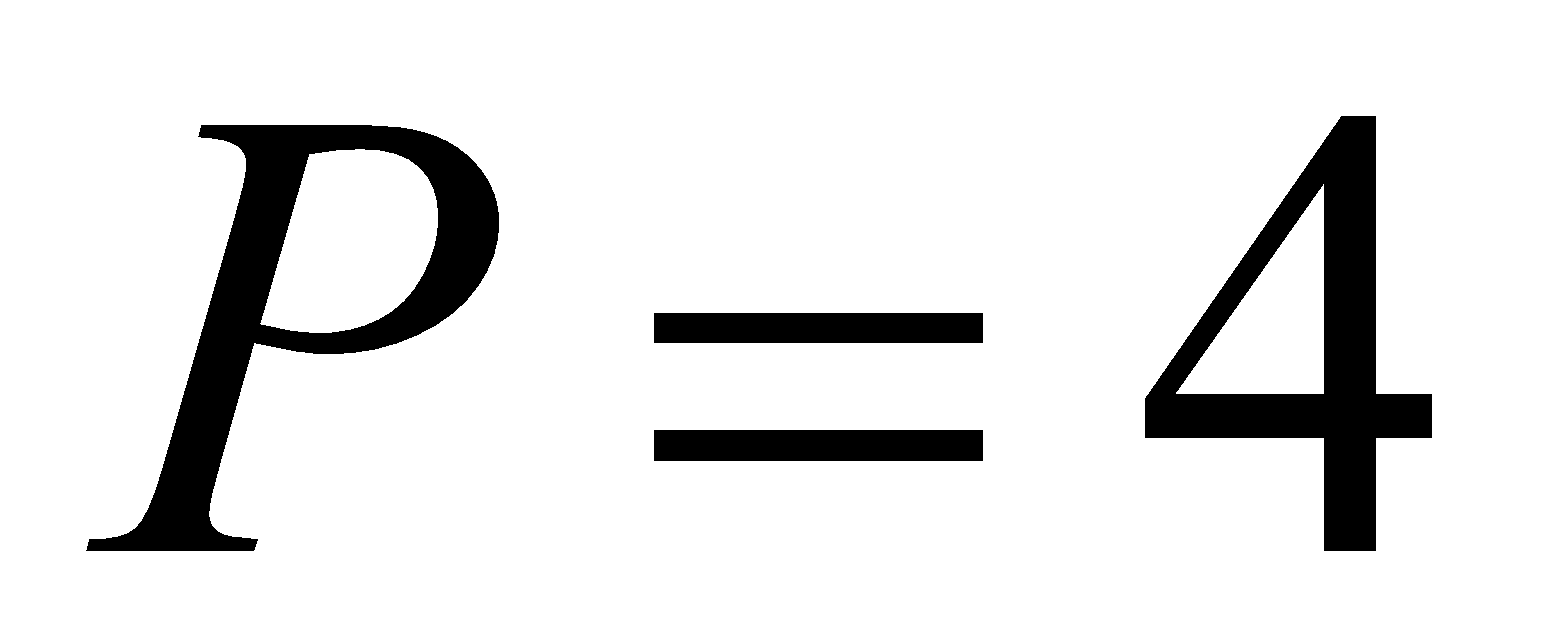
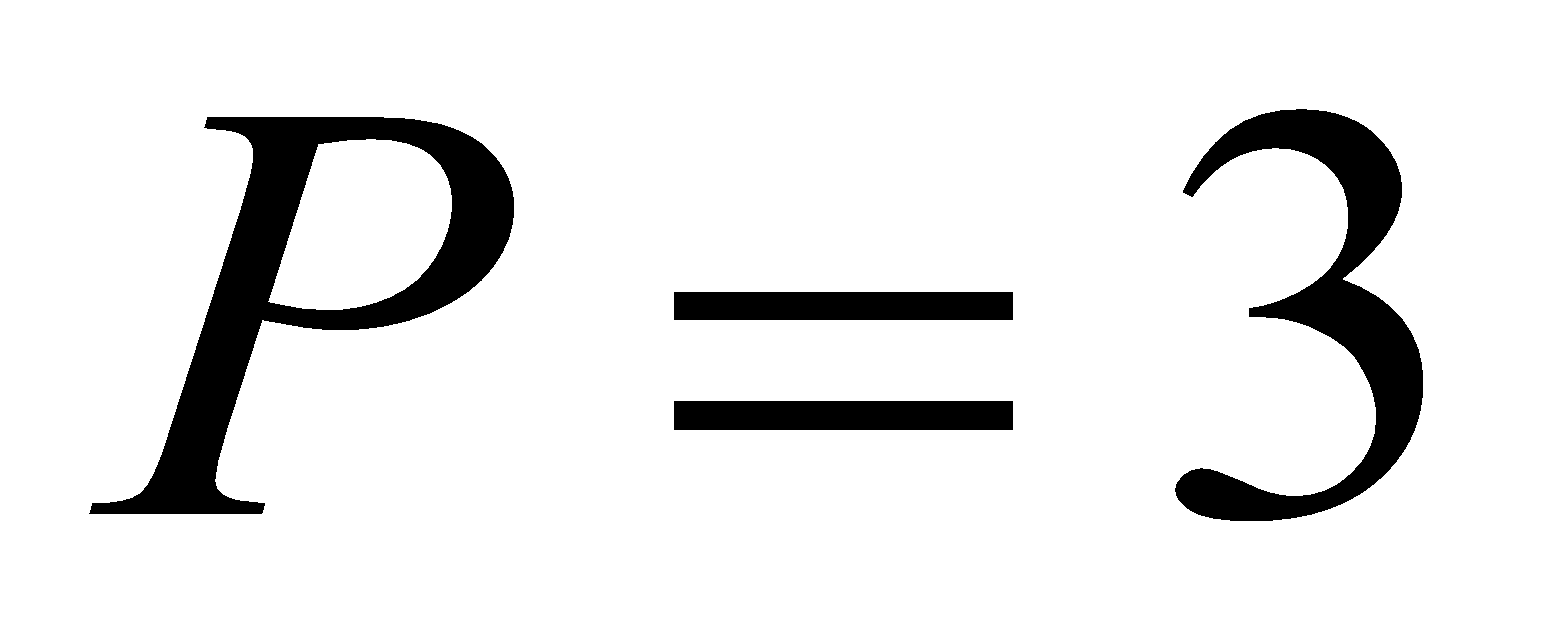
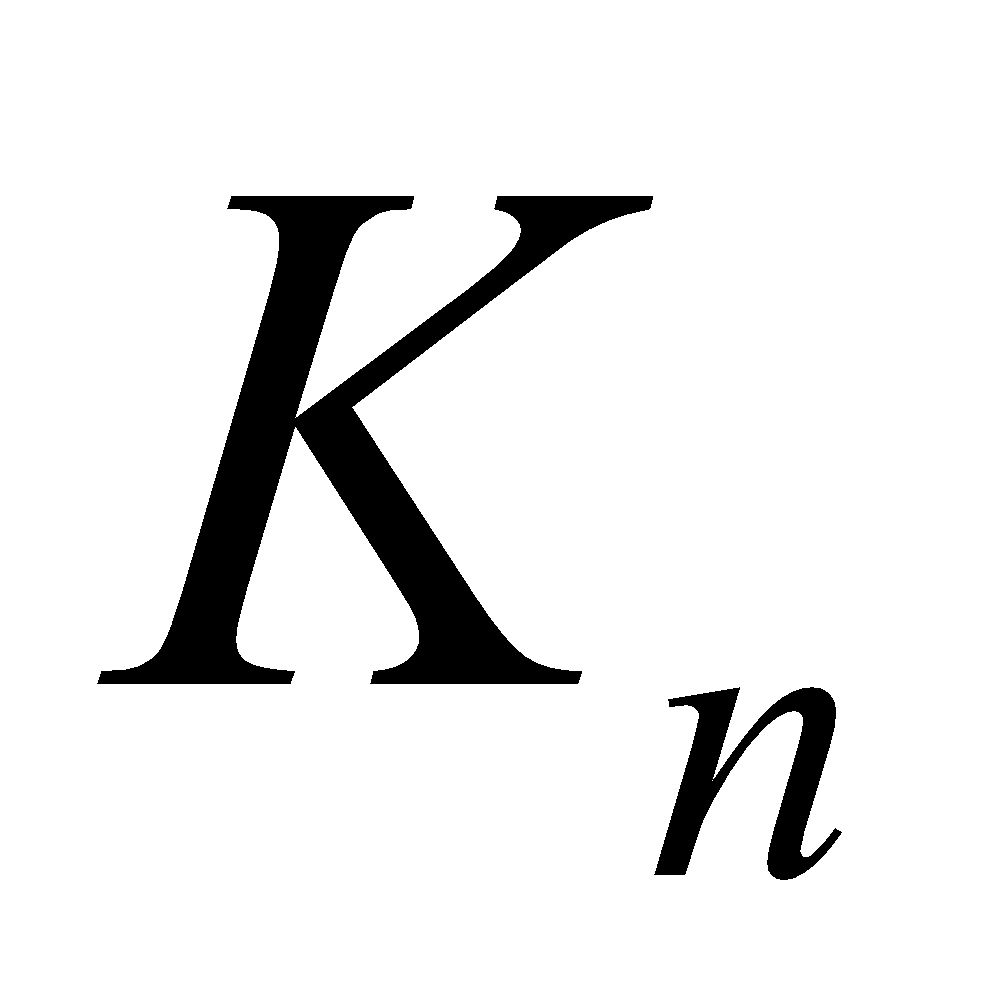
**1.7 Висновки до розділу 1**

Виконано розробку програми ПРГ1 для ПКС СП з використанням мови Java. Тестування програми ПРГ1 показало наступне:

1. використання багатоядерної ПКС та програми ПРГ1 забезпечує скорочення часу обчислення заданої математичної задачі. Значення  лежать в межах від 1,47 до 3,58. Значення  лежать в межах від 73% до 90% ;

2. Максимальне значення *Kп =* 3,58 забезпечує ПКС з та *N =* 2400

3. Мінімальне значення *Kп =* 1,47 виявлено у ПКС з  та *N =* 900

4. З ростом N для та    збільшується.

**РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП**

У даному розділі розглянуто розробку та дослідження програми ПРГ2 для ПКС з ЛП.

**2.1** **Огляд засобів для програмування в ПКС ЛП**

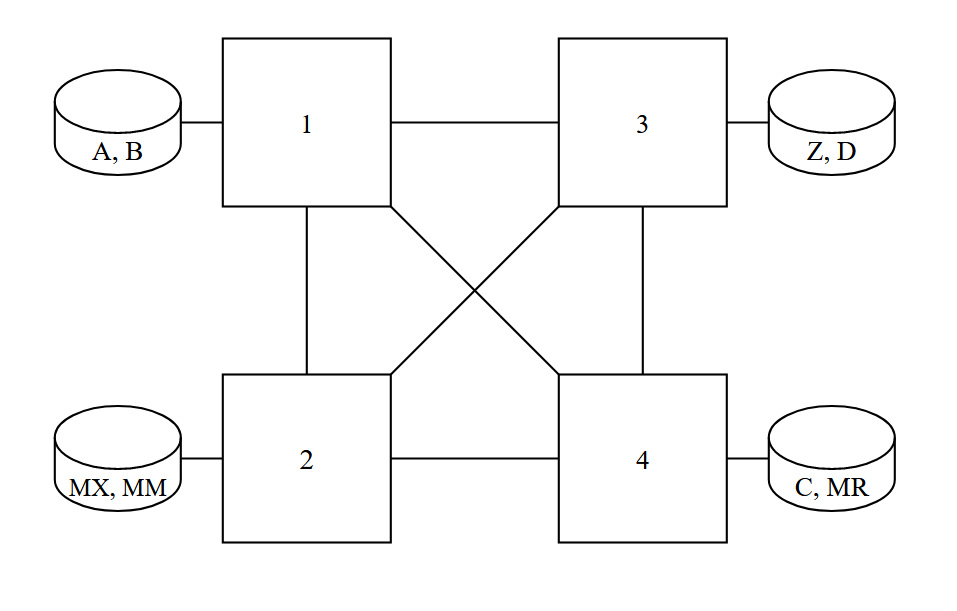
Вибір бібліотеки багато в чому залежить від задачі, яку необхідно вирішити. Наприклад бібліотека PVM надійніша оскільки перевірка виконується автоматично, в MPI їх треба виконувати самому, правда, якщо вони не потрібні заощаджується час, а механізм рандеву заснований на високорівневих концепціях рандеву, він надає засоби блокування і таймаутів, а також засоби для виконання вибіркового перестроювання черг клієнтів та аварійного завершення.

Основна ідея механізму рандеву досить проста. У специфікації завдання публікуються різні входи у завдання, в яких вона готова чекати звернення до неї від інших задач. Далі, в тілі задачі вказуються інструкції прийняття звернень до відповідних входів, зазначених у специфікації цієї задачі.

**2.2** **Розробка паралельного математичного алгоритму**

Математичний вираз: A = (B\*C)\*(Z\*MM)+D\*(MX\*MR)

Мова програмування: Ada (механізм рандеву)

Рисунок 2.1 - Структура ПКС ЛП

Паралельний алгоритм:

1. ai = (BH\*CH), i = 1..4.

2. a = a + ai, i = 1..4.

3. KH = (Z\*MMН)

4. AH = a\*K+DH\*(MX\*MRH)

**2.3** **Розробка алгоритмів процесів**

Задача Т1

1. Ввести B.

2.Передати ВН задачам Т2, Т3, Т4.

3. Отримати МХ, ММН від задачі Т2.

4. Отримати Z, DH від задачі Т3.

5. Отримати CH, MRH від задачі Т4.

6. Обчислити а1 = (BH\*CH).

7. Обчислити КН = (Z\*MMH).

8. Передати а1 задачі Т2.

9. Передати КН задачам Т2, Т3, Т4.

10. Отримати а від задачі Т3.

11. Отримати КН від задач Т2, Т3, Т4.

12. Обчислити АН = a\*K+DH\*(MX\*MRH).

13. Отримати АН від задач Т2, Т3, Т4.

14. Вивести А.

Задача Т2

1. Ввести MХ, ММ.

2.Передати MХ, ММН задачам Т2, Т3, Т4.

5. Отримати ВН від задачі Т1.

6. Отримати Z, DH від задачі Т3.

7. Отримати CH, MRH від задачі Т4.

8. Обчислити а2 = (BH\*CH).

9. Обчислити КН = (Z\*MMH).

10. Отримати а1 від задачі Т1.

11. Передати КН від задач Т1, Т3, Т4.

12. Отримати КН задачам Т1, Т3, Т4.

13. Обчислити а2 = а2 + а1.

14. Передати а2 задачі Т3.

15. Отримати а від задачі Т3.

16. Обчислити АН = a\*K+DH\*(MX\*MRH).

17. Передати АН задачі А1.

Задача Т3

1. Ввести Z, D.

2.Передати Z, DН задачам Т1 ,Т2, Т4.

5. Отримати ВН від задачі Т1.

6. Отримати МХ, ММН від задачі Т2.

7. Отримати CH, MRH від задачі Т4.

8. Обчислити а3 = (BH\*CH).

9. Обчислити КН = (Z\*MMH).

10. Передати а3 задачі Т4.

10. Передати КН задачам Т1, Т2, Т4.

11. Отримати КН від задач Т1, Т2, Т4

12. Отримати а2 від задачі Т2.

13. Отримати а4 від задачі Т4.

14. Обчислити а3 = а2 + а4.

15. Обчислити а = а + а3.

16. Передати а задачам Т1, Т2, Т4.

17. Обчислити АН = a\*K+DH\*(MX\*MRH).

18. Передати АН задачі А1.

Задача Т4

1. Ввести C, MR.

2.Передати CН, MRН. задачі Т1, Т2, Т3.

3. Отримати ВН від задачі Т1.

4. Отримати МХ, ММН від задачі Т2.

5. Отримати Z, DH від задачі Т3.

6. Обчислити а4 = (BH\*CH).

7. Обчислити КН = (Z\*MMH).

8. Передати КН задачам Т1, Т2, Т3.

9. Отримати КН від задач Т1, Т2, Т3.

10. Отримати а3 від задачі Т3.

11. Обчислити а4 = а4 + а3.

12. Передати а4 задачі Т3.

13. Отримати а від задачі Т3.

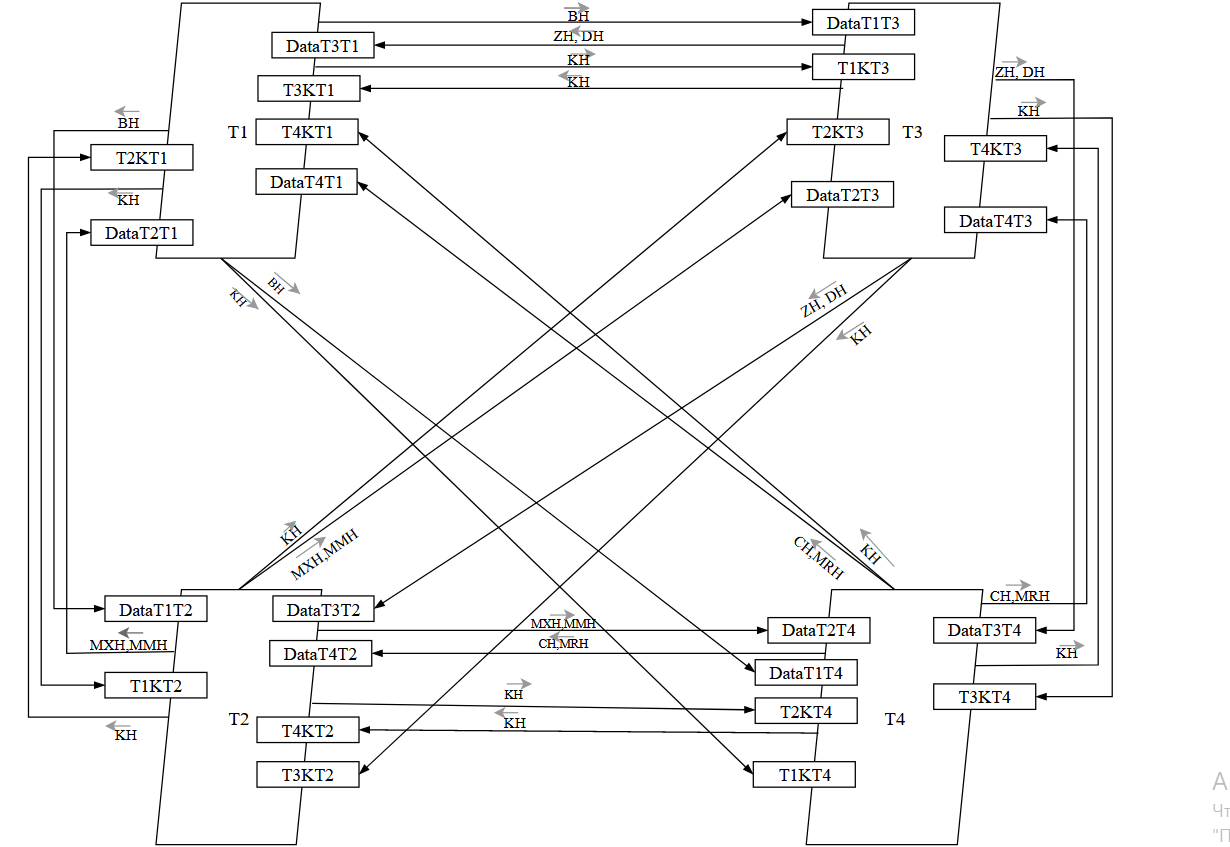
14. Обчислити АН = a\*K+DH\*(MX\*MRH).

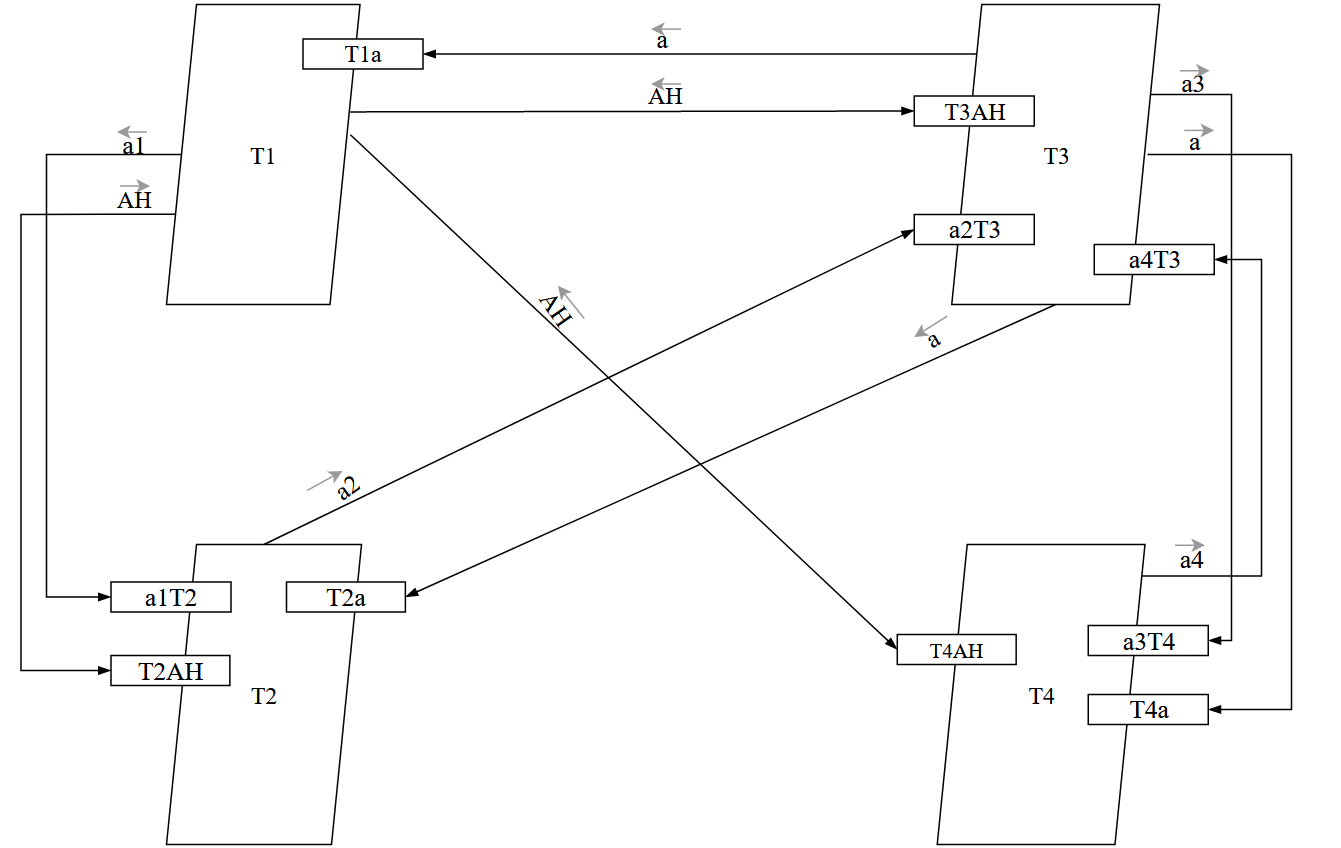
15. Передати АН задачі А1.

**2.4** **Розробка схеми взаємодії процесів**

На основі алгоритму для всіх задач розроблено структурну схему взаємодії процесів (рис. 2.2, 2.3). На структурній схемі взаємодії процесів показано:

* DataT1T2, DataT1T3, DataT1T4, DataT2T1, DataT2T3, DataT2T4, DataT3T1, DataT3T2, DataT3T4, DataT4T1, DataT4T2, DataT4T3 (на рис. 2.2) - для передачі даних між задачами.
* T1KT2, T1KT3, T1KT4, T2KT1, T2KT3, T2KT4, T3KT1, T3KT2, T3KT4, T4KT1, T4KT2, T4KT3 (на рис. 2.2) - для передачі проміжного результату часткового обчислення вектора К.
* а1Т2, а2Т3, а3Т4, а4Т3 (на рис. 2.3) - для передачі проміжного результату обчислення скалярного добутку векторів.
* Т1А, Т2А, Т4А (на рис. 2.3) - для передачі результату обчислення скалярного добутку векторів.
* Т2АН, Т3АН, Т4АН (на рис. 2.3) - для передачі частин остаточного виразу до задачі Т1.

Рисунок 2.2 – Структурна схема взаємодії потоків під час

Рисунок 2.3 – Структурна схема взаємодії потоків під час передачі результатів після обчислення проміжних результатів та остаточного виразу

**2.5 Розробка програми ПРГ2**

Програма ПРГ2 згідно технічного завдання розроблена на мові програмування Ada. Для взаємодії процесів використовується механізм рандеву.

Програма складається з модулю Main.adb, який містить процедуру main:

* Визначення констант N, P, H.
* Визначення типів Vector та Matrix.
* Процедур введення/виведення вектору та введення матриці.
* Специфікації та реалізації задач Т1, Т2, Т3, Т4

Лістниг розробленої програми наведено у додатку Б.

**2.6 Тестування програми ПРГ2**

Для тестування використовувалась паралельна обчислювальна система з наступними апаратними характеристиками:

* процесор: Intel Core i5-7200U, 4 ядра;
* оперативна пам’ять: DDR3 8 Гб;
* Операційна система: Microsoft Windows 10;

Для виміру часу використовується стандартна процедура *Clock* мови програмування Ada, що повертає поточний час системи. Різниця між часом системи після завершення обчислення і часом при запуску програми дає час виконання обчислення, що вимірюється в секундах.

Таблиця 2.1. Час виконання програми для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N/P | P=1 | P=2 | P=3 | P=4 |
| 900 | 14347 | 9159 | 5547 | 4015 |
| 1800 | 39837 | 24157 | 15037 | 10695 |
| 2400 | 102348 | 61168 | 37978 | 26236 |

Підрахунок коефіцієнту прискорення (КП) виконується за формулою КП = Т1 / Tp

Підрахунок коефіцієнту ефективності (КЕ) відбувається за формулою КЕ = КY / Р \* 100%

Таблиця 2.2. Значення Кп для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N/P | P=1 | P=2 | P=3 | P=4 |
| 900 | 1 | 1.566 | 2.58 | 3.57 |
| 1800 | 1 | 1.649 | 2.65 | 3.72 |
| 2400 | 1 | 1.673 | 2.694 | 3.901 |

Таблиця 2.3. Значення Ке для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N/P | P=1 | P=2 | P=3 | P=4 |
| 900 | 100% | 78.32% | 86.26% | 89.35% |
| 1800 | 100% | 82.45% | 88.37% | 93.14% |
| 2400 | 100% | 83.66% | 89.83% | 97.52% |

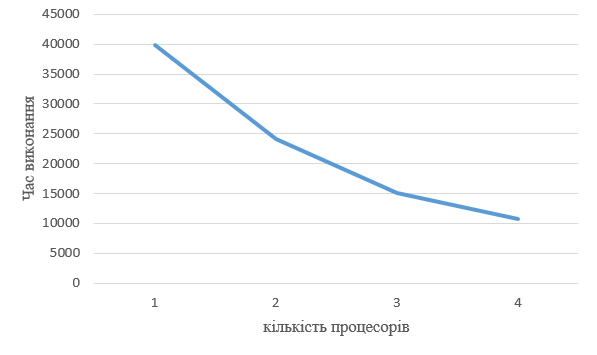
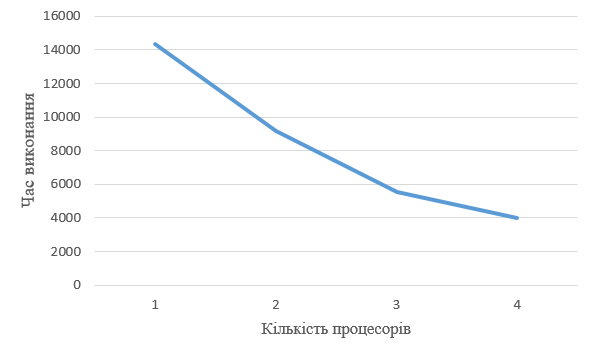
Рисунок 2.3 – Графік залежності часу виконання від кількості процесорів при N = 900

Рисунок 2.4 – Графік залежності часу виконання від кількості процесорів при N = 1800

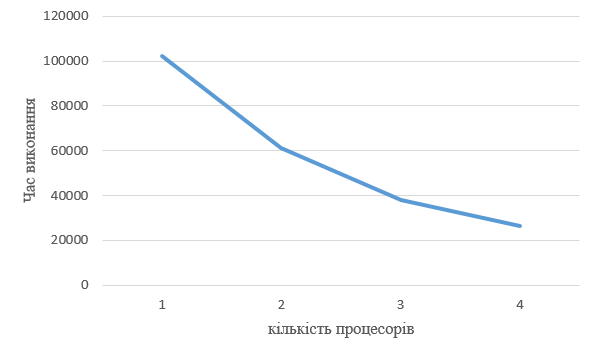
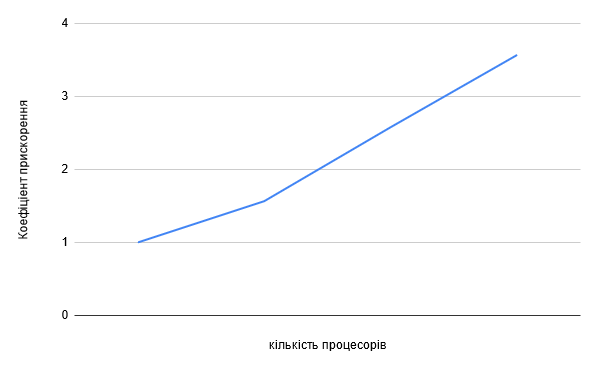
Рисунок 2.5 – Графік залежності часу виконання від кількості процесорів при N = 2400

Рисунок 2.6 – Графік залежності коефіцієнту прискорення від кількості процесорів при N = 900

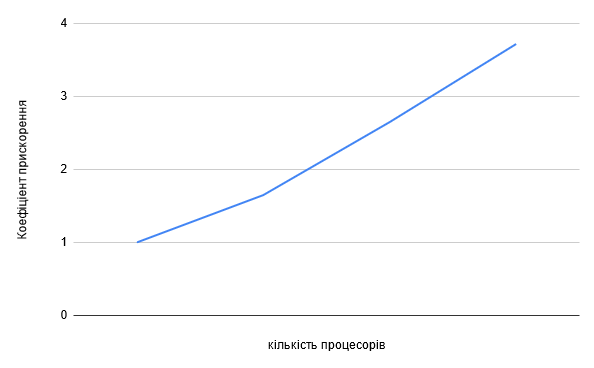


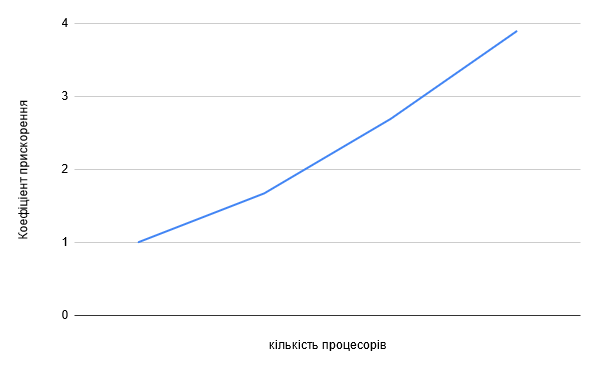
Рисунок 2.7 – Графік залежності коефіцієнту прискорення від кількості процесорів при N = 1800

Рисунок 2.8 – Графік залежності коефіцієнту прискорення від кількості процесорів при N = 2400

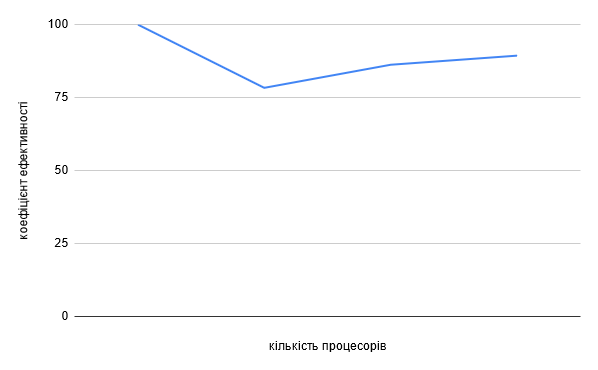
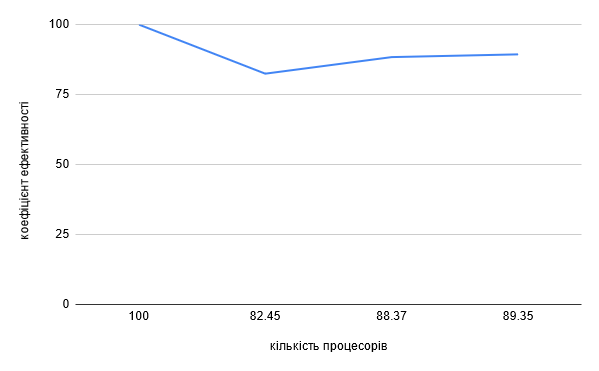
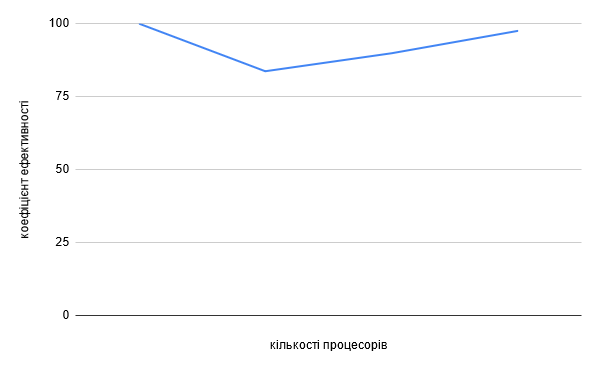
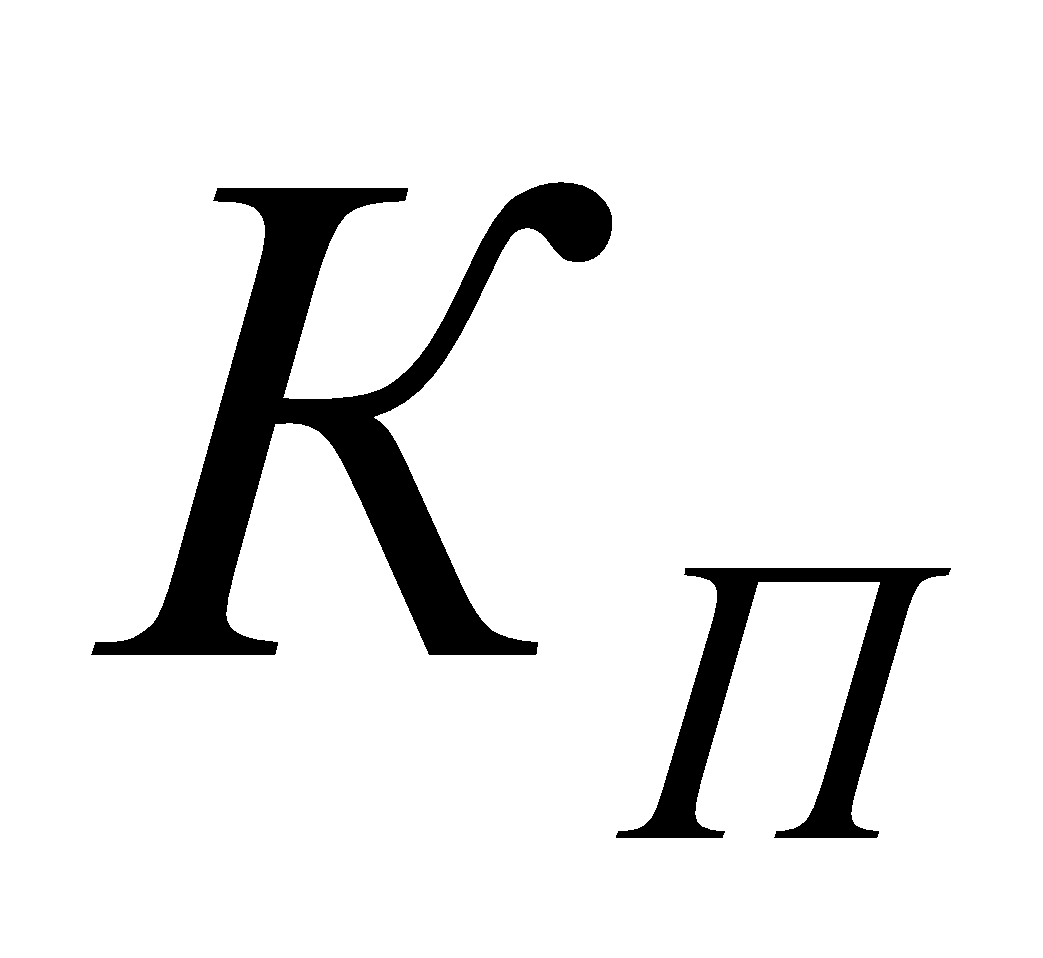
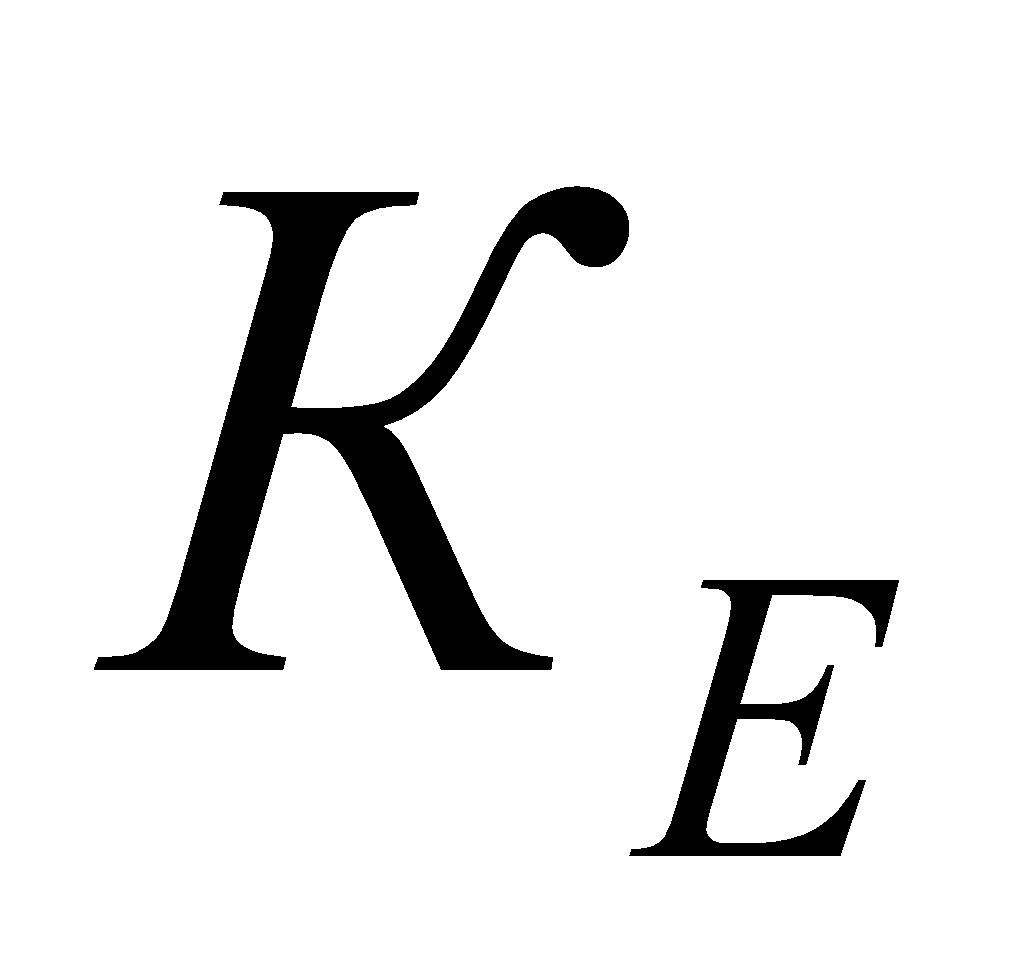
Рисунок 2.9 – Графік залежності коефіцієнту ефективності від кількості процесорів при N = 900

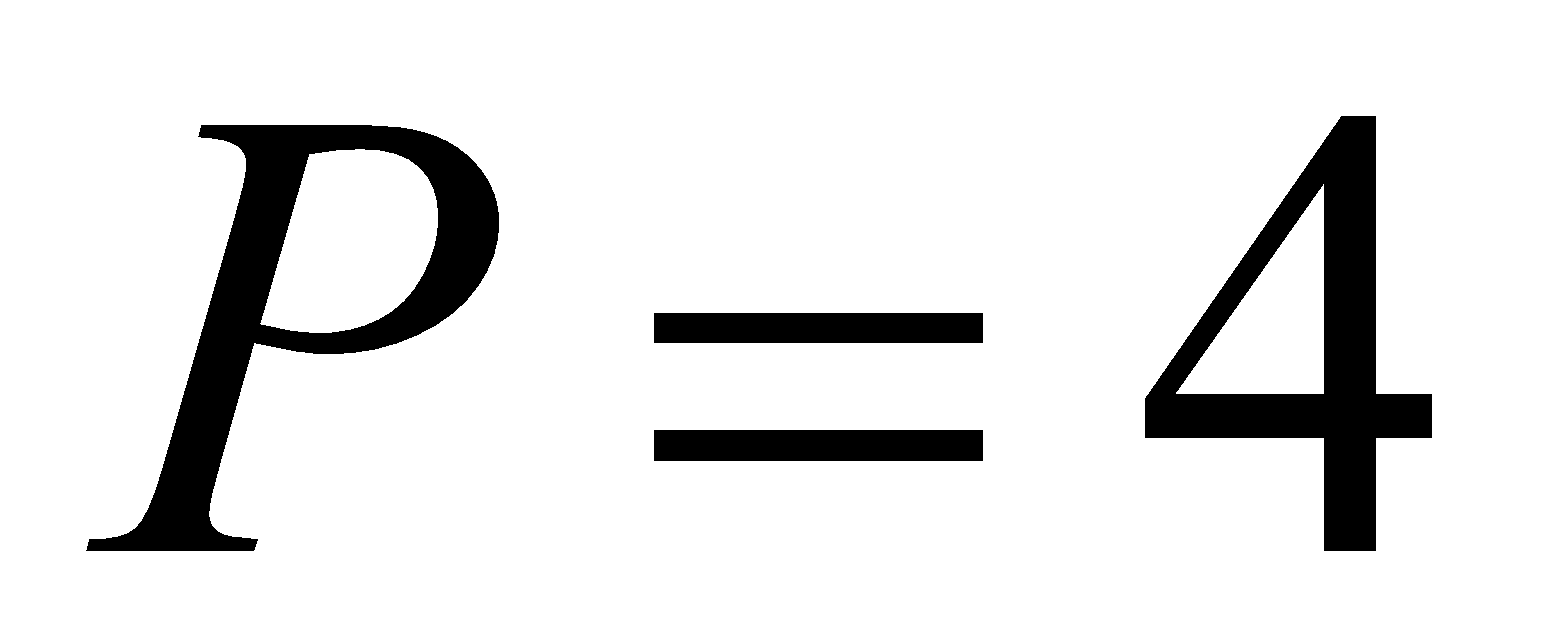
Рисунок 2.10 – Графік залежності коефіцієнту ефективності від кількості процесорів при N = 1800

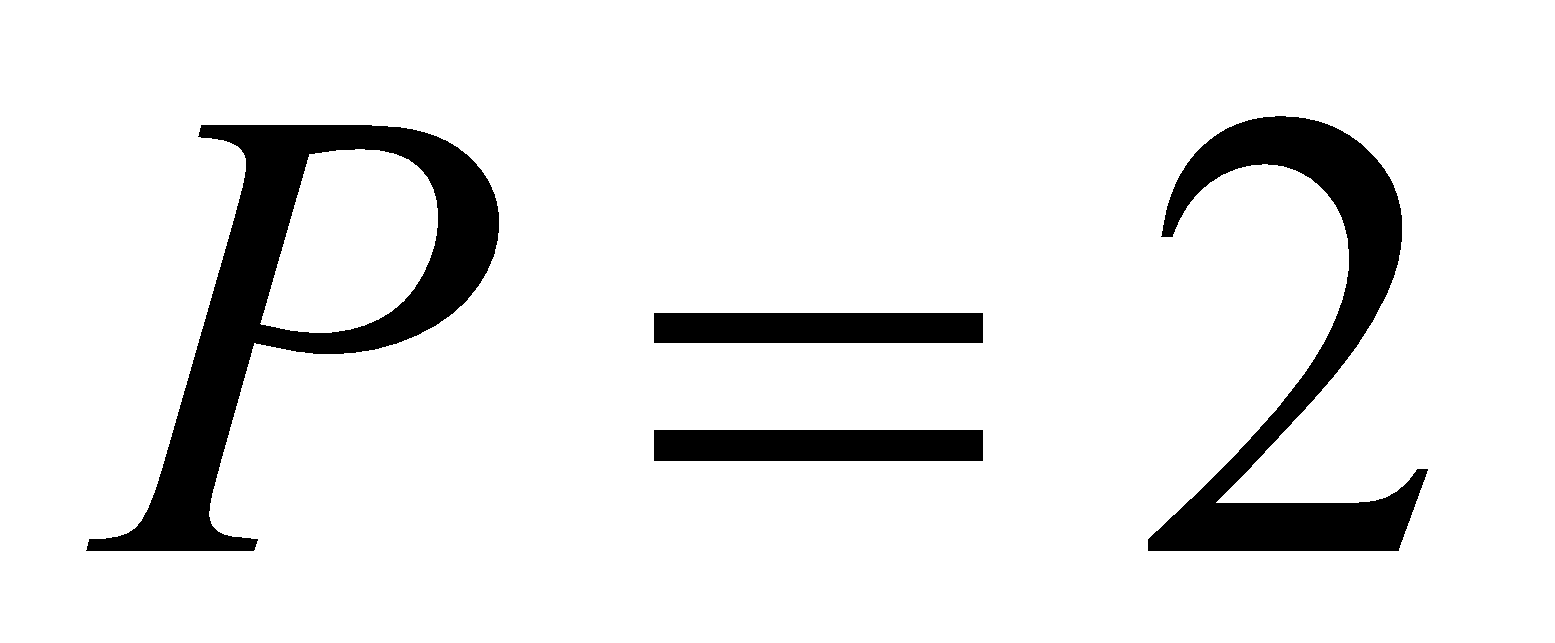
Рисунок 2.11 – Графік залежності коефіцієнту ефективності від кількості процесорів при N = 2400

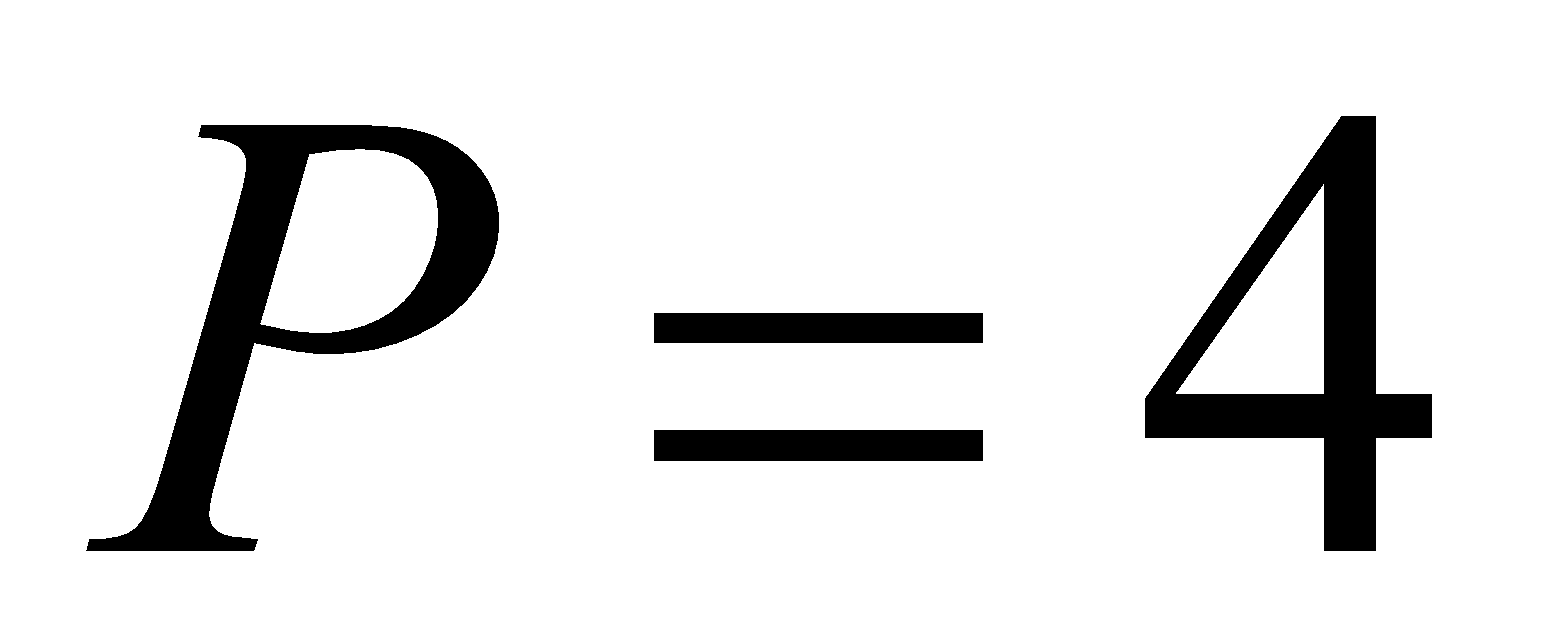
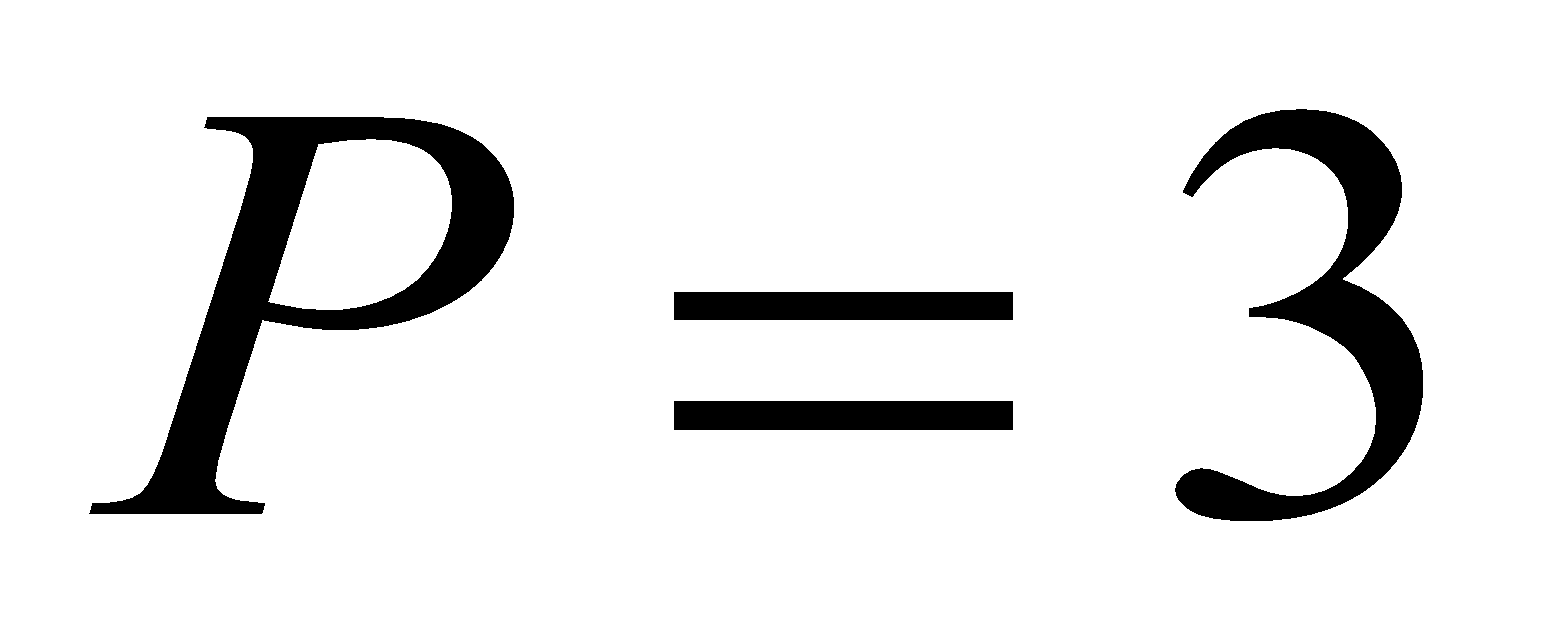
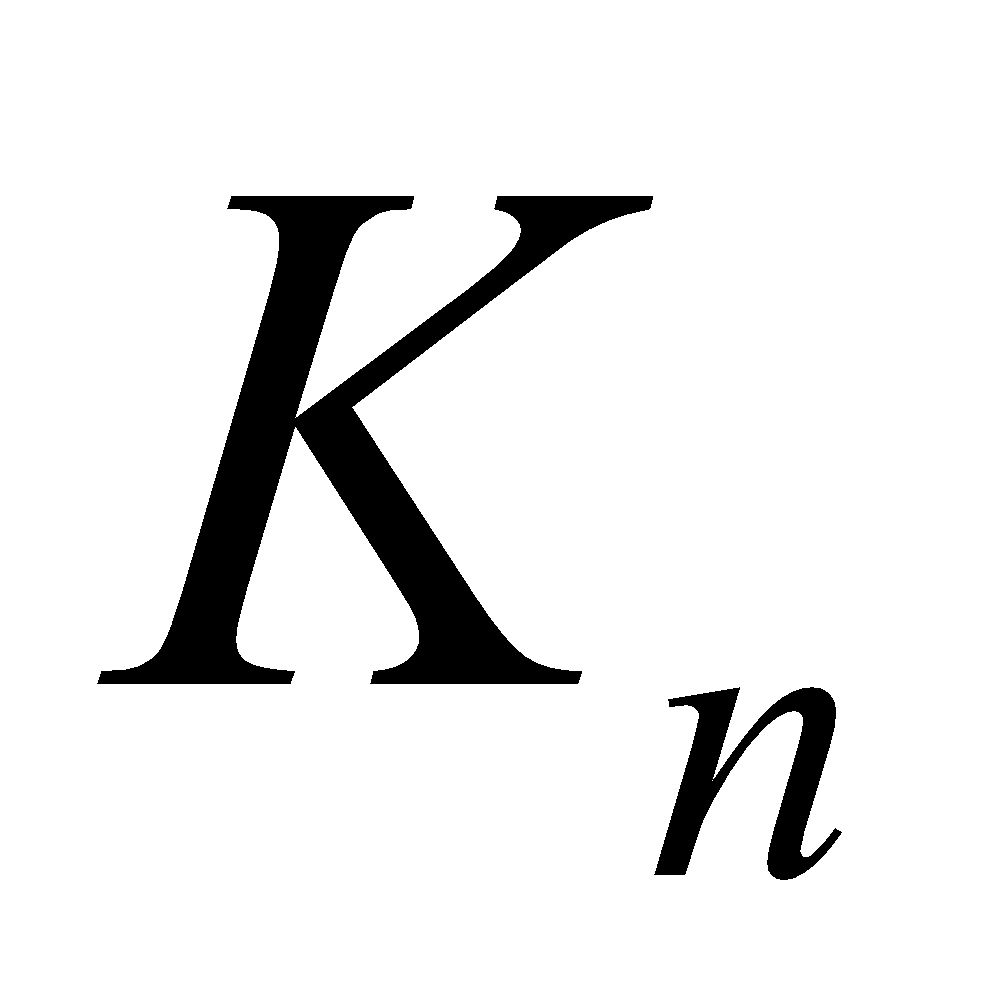
**2.7 Висновки до розділу 2**

Виконано розробку програми ПРГ2 для ПКС ЛП з використанням мови Ada. Тестування програми ПРГ2 показало наступне:

1. використання багатоядерної ПКС та програми ПРГ2 забезпечує скорочення часу обчислення заданої математичної задачі. Значення  лежать в межах від 1.566 до 3,9. Значення  лежать в межах від 78.32% до 97.52% ;

2. Максимальне значення *Kп =* 3,901 забезпечує ПКС з та *N =* 2400

3. Мінімальне значення *Kп =* 1,566 виявлено у ПКС з  та *N =* 900

4. З ростом N для та    збільшується.

**ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ**

1. В даній роботі було проведено аналіз принципів роботи паралельних програм та засобів роботи з процесами в ПРГ1 та ПРГ2. та мові Ada з використанням механізму рандеву.

2. Було розроблено програми на мові Java з використанням моніторів для ПКС зі спільною пам’яттю та Ada з використанням механізму рандеву для ПКС із локальною пам’яттю та проведено тестування цих програм.

3. Тестування ПРГ1 та ПРГ2 час виконання в ПРГ1 значно менший, ніж в ПРГ2 при задіяних 1-го, 2-х ядер та малих N. При великих N час виконання в ПРГ1 та ПРГ2 майже не відрізняється. Це пояснюється тим, що складна топологія системи збільшує час на передачу даних між процесорами і як наслідок впливає на загальний час виконання ПРГ2.

4. Коефіціент прискорення і коефіціент ефективності в ПРГ2 більший, ніж в ПРГ1.

5. Алгоритм роботи ПРГ1 значно легший, ніж у ПРГ2. У випадку системи з локальною пам‘яттю, велика кількість повідомлень різного розміру ускладнюють систему з пересилкою повідомлень, а також підвищують вірогідність помилки при реалізації.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення: Навч. Посібник [Текст]. – К.: Корнійчук, 2005. – 226 с. – ISBN 996-7599-36-1.

2. Ясько М.М. Навчальний посібник до вивчення курсу Паралельні та розподілені обчислення – РВВ ДНУ, 2010. – 76с.

3. Брайан Гетц, Тім Пейерлс, Джошуа Блох, Джозеф Боубер, Девід Холмс, Дуг Леа Java Concurrency in Practice – 2006 – ISBN 978-0321349606.

4. Девід Фланаган Java in a Nutshell, 5th Edition – 2005 – ISBN: 0596007736.

5. Джеймс Гослин. Особенности языка Java. 2004

6. Моріс Херліхи, Нір Шавіт The Art of Multiprocessor Programming – 2008 – ISBN 978-0123973375.

7. Багатоядерні та багатопроцесорні обчислювальні системи [Електронний ресурс]. – Режим доступу: https://stud.com.ua/50132/informatika/bagatoyaderni\_bagatoprotsesorni\_obchislyuvalni\_sistemi.

8. Корочкин А.В. Ада95.Введение в программирование. – Киев: “Світ”, 1998- 240 с

9. Е. Валях Последовательно-параллельные вычисления / Валях Е. – Пер. с англ. – Москва: Мир, 1985 – 456с.

**Додатки**

**Додаток А**

**Лістинг програми ПРГ1 для ПКС СП**

Main.java

1. /\*\*-- РГР Роздiл1 Java

2. \* -- A = (B\*C)\*(Z\*MM)+D\*(MX\*MR)

3. \* --

4. \* -- Date

5. \*/

6. public class Main{

7. public static int N = 8;

8. public static int P = 4;

9. public static int H = N/P;

10. static int a;

11. static int d;

12. static int[] Z = new int[N];

13. static int[] K = new int[N];

14. static int[][] MR = new int[N][N];

15. static int[][] MX = new int[N][N];

16. static int[] B = new int[N];

17. static int[] C = new int[N];

18. static int[] D = new int[N];

19. static int[][] MM = new int[N][N];

20. public static void main(String[] args) {

21. long startTime = System.currentTimeMillis();

22. System.out.println("Main started.");

23. InputMonitor inputMonitor = new InputMonitor();

24. CalcMonitor calcMonitor = new CalcMonitor();

25. OutputMonitor outputMonitor = new OutputMonitor();

26. T1 T1 = new T1(inputMonitor, calcMonitor, outputMonitor);

27. T2 T2 = new T2(inputMonitor, calcMonitor, outputMonitor);

28. T3 T3 = new T3(inputMonitor, calcMonitor, outputMonitor);

29. T4 T4 = new T4(inputMonitor, calcMonitor, outputMonitor);

30. T1.start();

31. T2.start();

32. T3.start();

33. T4.start();

34. try {

35. T1.join();

36. T2.join();

37. T3.join();

38. T4.join();

39. } catch (InterruptedException e) {

40. e.printStackTrace();

41. }

42. System.out.println("\nMain finished.\n\nTime: " + (System.currentTimeMillis() - startTime));

43. }

44. }

Calculations.java

45. import java.util.Arrays;

46. public class Calculations {

47. private static int H = Main.H;

48. public static void inputVector(int[] Vect, int val){

49. for (int i = 0; i < Vect.length; i++){

50. Vect[i] = val;

51. }

52. }

53. public static void inputMatrix(int[][] Matr, int val){

54. for (int i = 0; i < Matr.length; i++)

55. for (int j = 0; j < Matr[i].length; j++)

56. Matr[i][j] = val;

57. }

58. public static void outputVector(int[] Vect){

59. for (int[] aVect : Vect) {

60. System.out.print(Arrays.toString(aVect) + " ");

61. }

62. }

63. }

OutputMonitor.java

64. public class OutputMonitor {

65. private static int F = 0;

66. synchronized void Signal(){

67. F++;

68. if (F >= 4) notifyAll();

69. }

70. synchronized void WaitforSignal(){

71. try {

72. if (F < 3) wait();

73. } catch (Exception e){

74. e.printStackTrace();

75. }

76. }

77. }

InputMonitor.java

78. public class InputMonitor {

79. private static int F = 0;

80. synchronized void InputSignal(){

81. F++;

82. if (F >= 4) notifyAll();

83. }

84. synchronized void WaitforInput(){

85. try {

86. if (F < 3) wait();

87. } catch (Exception e){

88. e.printStackTrace();

89. }

90. }

91. }

CalcMonitor.java

92. public class CalcMonitor {

93. private static int F = 0;

94. private static int F1 = 0;

95. private int[] Z = Main.Z;

96. private int[] K = Main.Z;

97. private int[][] MX = Main.MX;

98. synchronized int[] copyK() {

99. return this.K;

100. }

101. synchronized int copyA(int q) {

102. return q;

103. }

104. synchronized int[] copyZ() { return this.Z; }

105. synchronized int[][] copyMX() { return this.MX; }

106. synchronized void SignalCalcA() {

107. F++;

108. if (F >= 4) notifyAll();

109. }

110. synchronized void SignalCalcK() {

111. F1++;

112. if (F1 >= 4) notifyAll();

113. }

114. synchronized void WaitforCalcA(){

115. try {

116. if (F < 4) wait();

117. } catch(Exception e){

118. e.printStackTrace();

119. }

120. }

121. synchronized void WaitforCalcK(){

122. try {

123. if (F1 < 4) wait();

124. } catch(Exception e){

125. e.printStackTrace();

126. }

127. }

128. }

T1.java

129. public class T1 extends Thread{

130. private InputMonitor inputMonitor;

131. private CalcMonitor calcMonitor;

132. private OutputMonitor outputMonitor;

133. T1(InputMonitor inputMonitor, CalcMonitor calcMonitor, OutputMonitor outputMonitor){

134. this.calcMonitor = calcMonitor;

135. this.inputMonitor = inputMonitor;

136. this.outputMonitor = outputMonitor;

137. }

138. public void run(){

139. int aRes = 0;

140. int start = 0, end = Main.H;

141. System.out.println("T1 started");

142. Calculations.inputVector(Main.D, 1);

143. Calculations.inputMatrix(Main.MR, 1);

144. // cигнал Т2, Т3, T4 про завершення введення даних

145. inputMonitor.InputSignal();

146. // oчікування завершення введення даних у інших потоках

147. inputMonitor.WaitforInput();

148. // копія MX, Z.

149. int[][] MX1 = calcMonitor.copyMX();

150. int[] Z1 = calcMonitor.copyZ();

151. int e = 0;

152. // обчислення a1 = (BH\*CH)

153. for (int i = start; i < end; i++) {

154. aRes += Main.B[i]\*Main.C[i]

155. }

156. // обчислення a

157. Main.a = aRes + a1;

158. // сигнал потокам про завершення обчислення а

159. calcMonitor.SignalCalcA();

160. // очікування сигналу від потоків про завершення обчислення а.

161. calcMonitor.WaitforCalcA();

162. // копія а

163. int a1 = calcMonitor.copyA(Main.a);

164. // обчислення K

165. for (int i = 0; i < Main.N; i++) {

166. for (int j = start; j < end; j++) {

167. Main.K[j] += Z1[j] \* Main.MM[i][j];

168. }

169. }

170. // Копія К.

171. int K1 = calcMonitor.copyK(Main.K);

172. // сигнал потокам про завершення обчислення K

173. calcMonitor.SignalCalcK();

174. // очікування сигналу від потоків про завершення обчислення K.

175. calcMonitor.WaitforCalcK();

176. // Обчислення AН = a1\*K + DH\*(MX1\*MRH)

177. int[][] MX\_R = new int[Main.N][Main.N];

178. for (int j = start; j < end; j++){

179. for (int i = 0; i < Main.N; i++){

180. MX\_R[i][j] = 0;

181. for (int k = 0; k < Main.N; k++){

182. MX\_R[i][j] += MX1[i][k] \* Main.MR[j][k];

183. }

184. }

185. }

186. int[] D\_MX = new int[Main.N];

187. for (int j = start; j < end; j++){

188. for (int i = 0; i < Main.N; i++){

189. D\_MX[i][j] = 0;

190. for (int k = 0; k < Main.N; k++){

191. D\_MX[j] += D1[j] \* MC\_M[j][k];

192. }

193. }

194. }

195. for (int j = start; j < end; j++){

196. for (int i = 0; i < Main.N; i++){

197. Main.A[j] = a1\*K1[j]+D\_MX[j][i];

198. }

199. }

200. // очікування сигналу від Т1, Т3, Т4 про завершення обчислення A.

201. outputMonitor.WaitforSignal();

202. // вивід A.

203. if (Main.N <= 10){

204. System.out.print("[ ");

205. Calculations.outputMatrix(Main.A);

206. System.out.println("]");

207. }

208. System.out.println("T1 finished.");

209. }

210. }

T2.java

211. public class T2 extends Thread{

212. private InputMonitor inputMonitor;

213. private CalcMonitor calcMonitor;

214. private OutputMonitor outputMonitor;

215. T2(InputMonitor inputMonitor, CalcMonitor calcMonitor, OutputMonitor outputMonitor){

216. this.inputMonitor = inputMonitor;

217. this.calcMonitor = calcMonitor;

218. this.outputMonitor = outputMonitor;

219. }

220. public void run(){

221. int arEs = 0;

222. int start = Lab2.H, end = Lab2.H\*2;

223. System.out.println("T2 started");

224. Calculations.inputVector(Main.D, 1);

225. Calculations.inputMatrix(Main.MR, 1);

226. // cигнал Т2, Т3, T4 про завершення введення даних

227. inputMonitor.InputSignal();

228. // oчікування завершення введення даних у інших потоках

229. inputMonitor.WaitforInput();

230. // копія MX, Z.

231. int[][] MX2 = calcMonitor.copyMX();

232. int[] Z2 = calcMonitor.copyZ();

233. int e = 1;

234. // обчислення a2 = (BH\*CH)

235. for (int i = start; i < end; i++) {

236. arEs += Main.B[i]\*Main.C[i]

237. }

238. // обчислення a

239. Main.a = arEs + a2;

240. // сигнал потокам про завершення обчислення а

241. calcMonitor.SignalCalcA();

242. // очікування сигналу від потоків про завершення обчислення а.

243. calcMonitor.WaitforCalcA();

244. // копія а

245. int a2 = calcMonitor.copyA(Main.a);

246. // обчислення K

247. for (int i = 0; i < Main.N; i++) {

248. for (int j = start; j < end; j++) {

249. Main.K[j] += Z2[j] \* Main.MM[i][j];

250. }

251. }

252. // Копія К.

253. int K2 = calcMonitor.copyK(Main.K);

254. // сигнал потокам про завершення обчислення K

255. calcMonitor.SignalCalcK();

256. // очікування сигналу від потоків про завершення обчислення K.

257. calcMonitor.WaitforCalcK();

258. // Обчислення AН = a2\*K + DH\*(MX2\*MRH)

259. int[][] MX\_R = new int[Main.N][Main.N];

260. for (int j = start; j < end; j++){

261. for (int i = 0; i < Main.N; i++){

262. MX\_R[i][j] = 0;

263. for (int k = 0; k < Main.N; k++){

264. MX\_R[i][j] += MX2[i][k] \* Main.MR[j][k];

265. }

266. }

267. }

268. int[] D\_MX = new int[Main.N];

269. for (int j = start; j < end; j++){

270. for (int i = 0; i < Main.N; i++){

271. D\_MX[i][j] = 0;

272. for (int k = 0; k < Main.N; k++){

273. D\_MX[j] += D2[j] \* MC\_M[j][k];

274. }

275. }

276. }

277. for (int j = start; j < end; j++){

278. for (int i = 0; i < Main.N; i++){

279. Main.A[j] = a2\*K2[j]+D\_MX[j][i];

280. }

281. }

282. // cигнал про завершення обчислення MR.

283. outputMonitor.Signal();

284. System.out.println("T2 finished.");

285. }

286. }

T3.java

287. public class T3 extends Thread{

288. private InputMonitor inputMonitor;

289. private CalcMonitor calcMonitor;

290. private OutputMonitor outputMonitor;

291. T3(InputMonitor inputMonitor, CalcMonitor calcMonitor, OutputMonitor outputMonitor){

292. this.calcMonitor = calcMonitor;

293. this.inputMonitor = inputMonitor;

294. this.outputMonitor = outputMonitor;

295. }

296. public void run(){

297. int areS = 0;

298. int start = Lab2.H\*2, end = Lab2.H\*3;

299. System.out.println("T3 started");

300. Calculations.inputVector(Main.D, 1);

301. Calculations.inputMatrix(Main.MR, 1);

302. // cигнал Т2, Т3, T4 про завершення введення даних

303. inputMonitor.InputSignal();

304. // oчікування завершення введення даних у інших потоках

305. inputMonitor.WaitforInput();

306. // копія MX, Z.

307. int[][] MX3 = calcMonitor.copyMX();

308. int[] Z3 = calcMonitor.copyZ();

309. int e = 3;

310. // обчислення a3 = (BH\*CH)

311. for (int i = start; i < end; i++) {

312. areS += Main.B[i]\*Main.C[i]

313. }

314. // обчислення a

315. Main.a = areS + a3;

316. // сигнал потокам про завершення обчислення а

317. calcMonitor.SignalCalcA();

318. // очікування сигналу від потоків про завершення обчислення а.

319. calcMonitor.WaitforCalcA();

320. // копія а

321. int a3 = calcMonitor.copyA(Main.a);

322. // обчислення K

323. for (int i = 0; i < Main.N; i++) {

324. for (int j = start; j < end; j++) {

325. Main.K[j] += Z3[j] \* Main.MM[i][j];

326. }

327. }

328. // Копія К.

329. int K3 = calcMonitor.copyK(Main.K);

330. // сигнал потокам про завершення обчислення K

331. calcMonitor.SignalCalcK();

332. // очікування сигналу від потоків про завершення обчислення K.

333. calcMonitor.WaitforCalcK();

334. // Обчислення AН = a3\*K + DH\*(MX3\*MRH)

335. int[][] MX\_R = new int[Main.N][Main.N];

336. for (int j = start; j < end; j++){

337. for (int i = 0; i < Main.N; i++){

338. MX\_R[i][j] = 0;

339. for (int k = 0; k < Main.N; k++){

340. MX\_R[i][j] += MX3[i][k] \* Main.MR[j][k];

341. }

342. }

343. }

344. int[] D\_MX = new int[Main.N];

345. for (int j = start; j < end; j++){

346. for (int i = 0; i < Main.N; i++){

347. D\_MX[i][j] = 0;

348. for (int k = 0; k < Main.N; k++){

349. D\_MX[j] += D3[j] \* MC\_M[j][k];

350. }

351. }

352. }

353. for (int j = start; j < end; j++){

354. for (int i = 0; i < Main.N; i++){

355. Main.A[j] = a3\*K3[j]+D\_MX[j][i];

356. }

357. }

358. // cигнал про завершення обчислення A.

359. outputMonitor.Signal();

360. System.out.println("T3 finished.");

361. }

362. }

T4.java

363. public class T4 extends Thread{

364. private InputMonitor inputMonitor;

365. private CalcMonitor calcMonitor;

366. private OutputMonitor outputMonitor;

367. T4(InputMonitor inputMonitor, CalcMonitor calcMonitor, OutputMonitor outputMonitor){

368. this.calcMonitor = calcMonitor;

369. this.inputMonitor = inputMonitor;

370. this.outputMonitor = outputMonitor;

371. }

372. public void run(){

373. int Ares = 0;

374. int start = Lab2.H\*3, end = Lab2.N;

375. System.out.println("T4 started");

376. Calculations.inputVector(Main.D, 1);

377. Calculations.inputMatrix(Main.MR, 1);

378. // cигнал Т2, Т3, T4 про завершення введення даних

379. inputMonitor.InputSignal();

380. // oчікування завершення введення даних у інших потоках

381. inputMonitor.WaitforInput();

382. // копія MX, Z.

383. int[][] MX4 = calcMonitor.copyMX();

384. int[] Z4 = calcMonitor.copyZ();

385. int e = 4;

386. // обчислення a4 = (BH\*CH)

387. for (int i = start; i < end; i++) {

388. Ares += Main.B[i]\*Main.C[i]

389. }

390. // обчислення a

391. Main.a = Ares + a4;

392. // сигнал потокам про завершення обчислення а

393. calcMonitor.SignalCalcA();

394. // очікування сигналу від потоків про завершення обчислення а.

395. calcMonitor.WaitforCalcA();

396. // копія а

397. int a4 = calcMonitor.copyA(Main.a);

398. // обчислення K

399. for (int i = 0; i < Main.N; i++) {

400. for (int j = start; j < end; j++) {

401. Main.K[j] += Z4[j] \* Main.MM[i][j];

402. }

403. }

404. // Копія К.

405. int K4 = calcMonitor.copyK(Main.K);

406. // сигнал потокам про завершення обчислення K

407. calcMonitor.SignalCalcK();

408. // очікування сигналу від потоків про завершення обчислення K.

409. calcMonitor.WaitforCalcK();

410. // Обчислення AН = a3\*K + DH\*(MX3\*MRH)

411. int[][] MX\_R = new int[Main.N][Main.N];

412. for (int j = start; j < end; j++){

413. for (int i = 0; i < Main.N; i++){

414. MX\_R[i][j] = 0;

415. for (int k = 0; k < Main.N; k++){

416. MX\_R[i][j] += MX4[i][k] \* Main.MR[j][k];

417. }

418. }

419. }

420. int[] D\_MX = new int[Main.N];

421. for (int j = start; j < end; j++){

422. for (int i = 0; i < Main.N; i++){

423. D\_MX[i][j] = 0;

424. for (int k = 0; k < Main.N; k++){

425. D\_MX[j] += D4[j] \* MC\_M[j][k];

426. }

427. }

428. }

429. for (int j = start; j < end; j++){

430. for (int i = 0; i < Main.N; i++){

431. Main.A[j] = a4\*K4[j]+D\_MX[j][i];

432. }

433. }

434. // cигнал про завершення обчислення A.

435. outputMonitor.Signal();

436. System.out.println("T4 finished.");

437. }

438. }

**Додаток Б**

**Лістинг програми ПРГ2 для ПКС ЛП**

1. -- RGR. Ada

2. -- A = (B\*C)\*(Z\*MM)+D\*(MX\*MR)

3. --

4. -- Date 5. with Ada.Integer\_Text\_IO, Ada.Text\_IO;

6. use Ada.Integer\_Text\_IO, Ada.Text\_IO;

7. procedure Lab1 is

8. N: Integer:=2400;

9. P: Integer:=4;

10. H: Integer:=N/P;

11. type Vector\_All is array(Integer range <>) of Integer;

12. subtype Vector is Vector\_All(1..N);

13. subtype VectorH is Vector\_All(1..1\*H);

14. subtype Vector2H is Vector\_All(1..2\*H);

15. subtype Vector3H is Vector\_All(1..3\*H);

16. type Matrix\_All is array(Integer range <>) of Vector;

17. subtype Matrix is Matrix\_All(1..N);

18. subtype MatrixH is Matrix\_All(1..1\*H);

19. subtype Matrix2H is Matrix\_All(1..2\*H);

20. subtype Matrix3H is Matrix\_All(1..3\*H);

21. -----Procedure for Input Vector-----

22. procedure VectorInput(V: out Vector) is

23. begin

24. for i in 1..N loop

25. V(i) := 1;

26. end loop;

27. end VectorInput;

28. -----Procedure for Input Matrix-----

29. procedure MatrixInput(M: out Matrix) is

30. begin

31. for i in 1..N loop

32. for j in 1..N loop

33. M(i)(j) := 1;

34. end loop;

35. end loop;

36. end MatrixInput;

37. -----Procedure for Output Vector-----

38. procedure VectorOutput(V: in Vector; str: in String) is

39. begin

40. Put("Vector " & str & ":");

41. for i in 1..N loop

42. put(V(i));

43. Put\_Line(" ");

44. end loop;

45. Put\_Line("");

46. end VectorOutput;

47. -----Procedure of calculation AH = a\*K+DH\*(MX\*MRH)-----

48. procedure Calculate(a : Integer; DH : VectorH; K : Vector; MX : Matrix; MRH : MatrixH; AH : out VectorH) is

49. sum1 : Integer;

50. begin

51. for i in 1..H loop

52. for j in 1..N loop

53. sum1 := 0;

54. for k in 1..N loop

55. sum1 := sum1 + MX(k)(j) \* MRH(i)(k);

56. end loop;

57. -----calculation AH-----

58. AH(i) := a \* K(j) + DH(i) \* sum1;

59. end loop;

60. end loop;

61. end Calculate;

62. procedure tasksProcedure is

63. ------------------------------Specification Task 1------------------------------

64. task T1 is

65. pragma Storage\_Size(3000000000);

66. entry DataT4T1(VH : VectorH; MH : MatrixH); -- CH MRH

67. entry DataT3T1(V : Vector; VH : VectorH);

68. entry DataT2T1(MX : Matrix; MMH : MatrixH);

69. entry T4KT1(V4H : Vector);

70. entry T3KT1(V3H : Vector3H);

71. entry T2KT1(V2H : Vector2H);

72. entry T1a(a : Integer);

73. entry T2AH(resAH : VectorH);

74. entry T3AH(res2AH : VectorH);

75. entry T4AH(res3AH : VectorH);

76. end T1;

77. ------------------------------Specification Task 2-----------------------

78. task T2 is

79. pragma Storage\_Size(3000000000);

80. entry DataT4T2(VH : VectorH; MH : MatrixH); -- CH MRH

81. entry DataT3T2(V : Vector; VH : VectorH);

82. entry DataT1T2(VH : VectorH);

83. entry T4KT2(V4H : Vector4H);

84. entry T3KT2(V3H : Vector3H);

85. entry T1KT2(VH : VectorH);

86. entry T2a(a : Integer);

87. entry a1T2(a1 : Integer);

88. end T2;

89. ------------------------------Specification Task 3---------------------

90. task T3 is

91. pragma Storage\_Size(3000000000);

92. entry DataT4T3(VH : VectorH; MH : MatrixH); -- CH MRH

93. entry DataT2T3(M : Matrix; MMH : MatrixH);

94. entry DataT1T3(V : Vector);

95. entry T4KT3(V4H : Vector4H);

96. entry T2KT3(V2H : Vector2H);

97. entry T1KT3(VH : VectorH);

98. entry a2T3(a2 : Integer);

99. entry a4T3(a4 : Integer);

100. end T3;

101. ------------------------------Specification Task 4----------------------

102. task T4 is

103. pragma Storage\_Size(3000000000);

104. entry DataT3T4(V : Vector; VH : VectorH); -- CH MRH

105. entry DataT2T4(M : Matrix; MMH : MatrixH);

106. entry DataT1T4(VH : VectorH);

107. entry T3KT4(V3H : Vector3H);

108. entry T2KT4(V2H : Vector2H);

109. entry T1KT4(VH : VectorH);

110. entry a3T4(a3 : Integer);

111. entry T4a(a : Integer);

112. end T4;

113. ------------------------------Task 1------------------------------

114. task body T1 is

115. a1, resa : Integer;

116. B, Z, A, resk: Vector;

117. MXH, MMH, MRH : MatrixH;

118. DH, BH, CH, KH, AH : VectorH; K2H : VactorH;

119. begin

120. Put\_Line("T1 started.");

121. VectorInput(B);

122. ----------------------passing BH to T2---------------------

123. T2.DataT1T2(B(H+1..2\*H));

124. ----------------------passing BH to T3---------------------

125. T3.DataT1T3(B(2\*H+1..3\*H));

126. ----------------------passing BH to T4---------------------

127. T4.DataT1T4(B(3\*H+1..N));

128. ----------------------get MXH, MMH ot T2-----------------------

129. accept DataT2T1 (M : Matrix; MMH : MatrixH) do

130. MX := M;

131. MMH := MMH;

132. end DataT2T1;

133. ----------------------get Z, DH ot T3-----------------------

134. accept DataT3T1 (V : in Vector; VH : in VectorH) do

135. Z := V;

136. DH := VH;

137. end DataT3T1;

138. ----------------------get CH, MRH ot T4-----------------------

139. accept DataT4T1 (VH : in VectorH; MH : in MatrixH) do

140. CH := VH;

141. MRH := MH;

142. end DataT4T1;

143. ----------------------calculation a1 = BH \* CH----------------------

144. a1 := 0;

145. for i in 1..H loop

146. a1 := a1 + BH(i) \* CH(i);

147. end loop;

148. ------------------passing a1 to T2----------------------------------

149. T2.a1T2(a1);

150. ----------------------------get a ot T3-----------------------------

151. accept T1a (a : in Integer) do

152. resa := a;

153. end T1a;

154. -----------------calculation KH = (Z\*MMH)------------------

155. for i in 1..H loop

156. for j in 1..N loop

157. KH := Z(j)\*MMH(i)(j);

158. end loop;

159. end loop;

160. -------------------passing KH to T2, T3, T4-------------------------

161. T2.T1KT2(KH);

162. T3.T1KT3(KH);

163. T4.T1KT4(KH);

164. ----------------------get KH ot T2----------------------------------

165. accept T2KT1 (V2H : in Vector2H) do

166. KH:=V2H;

167.

168. end T2KT1;

169. ----------------------get KH ot T3----------------------------------

170. accept T3KT1 (V3H : in Vector3H) do

171. K2H:=V3H;

172. KH := K2H;

173. end T3KT1;

174. ----------------------get KH ot T4----------------------------------

175. accept T4KT1 (V4H : in Vector\_All) do

176. K4H := V4H;

177. resk := KH;

178. end T4KT1;

179. ----------------------calculation AH = K + a\*EH\*x-------------------

180. Calculate(a => resa, DH => DH, MX => MX, MRH => MRH, K => resk, AH => AH);

181. ----------------------get AH ot T2---------------------------------

182. accept T2AH (resAH : in VectorH) do

183. A(1..H) := AH;

184. A(H+1..2\*H) := resAH;

185. end T2AH;

186. accept T3AH (res2AH : in VectorH) do

187. A(1..H) := AH;

188. A(2\*H+1..3\*H) := res2AH;

189. end T3AH;

190. accept T4AH (res3AH : in VectorH) do

191. A(1..H) := AH;

192. A(3\*H+1..N) := res3AH;

193. end T4AH;

194. ---------------------print A----------------------------------------

195. if (N<10) then

196. delay(0.5);

197. VectorOutput(A, "A");

198. end if;

199. Put\_Line("T1 finished.");

200. end T1;

201. ---------------------------------Task 2------------------------------------

202. task body T2 is

203. a2, resa : Integer;

204. DH, BH, CH, KH : VectorH;

205. K2H : Vector2H;

206. Z, resk : Vector;

207. MX, MM : Matrix;

208. MRH : MatrixH;

209. begin

210. Put\_Line("T2 started.");

211. ----------------------------Enter MX, MM----------------------------

212. MatrixInput(MX);

213. MatrixInput(MM);

214. ----------------------passing MX, MMH to T1---------------------

215. T1.DataT2T1(MX, MMH(2\*H+1..3\*H));

216. ----------------------passing MX, MMH to T3---------------------

217. T3.DataT2T3(MX, MMH(2\*H+1..3\*H));

218. ----------------------passing MX, MMH to T4---------------------

219. T4.DataT2T4(MX, MMH(2\*H+1..3\*H));

220. -----------------get B, Z, DH, CH, MRH ot T1, T3, T4-----------------

221. accept DataT1T2 (VH : in VectorH) do

222. BH := VH;

223. end DataT1T2;

224. accept DataT3T2 (V : in Vector; VH : in VectorH) do

225. Z := V;

226. DH := VH;

227. end DataT3T2;

228. accept DataT4T2 (VH : in VectorH; MH : in MatrixH) do

229. CH := VH;

230. MRH := MH;

231. end DataT4T2;

232. ----------------------calculation a2 = BH \* CH----------------------

233. a2 := 0;

234. for i in H+1..2\*H loop

235. a2 := a2 + BH(i) \* CH(i);

236. end loop;

237. -------------accept a1 ot T1 and calculation a2 = a2 + a1-----------

238. accept a1T2 (a1 : in Integer) do

239. a2 := a2 + a1;

240. end a1T2;

241. ------------------------passing a2 to T3----------------------------

242. T3.a2T3(a2);

243. ----------------------------get a ot T3-----------------------------

244. accept T2a (a : in Integer) do

245. resa := a;

246. end T2a;

247. -----------------calculation KH = (Z\*MMH)------------------

248. for i in H+1..2\*H loop

249. for j in 1..N loop

250. KH := Z(j)\*MMH(i)(j);

251. end loop;

252. end loop;

253. ----------------------passing KH to T4------------------------------

254. T1.T2KT1(K2H);

255. T3.T2KT3(K2H);

256. T4.T2KT4(K2H);

257. ----------------------get KH ot T2----------------------------------

258. accept T1KT2 (VH : in VectorH) do

259. KH:=VH;

260. KH := K2H;

261. end T1KT2;

262. accept T3KT2 (V3H : in Vector3H) do

263. KH := V3H;

264. resk := KH;

265. end T3KT2;

266. accept T4KT2 (V4H : in Vector) do

267. K2H := V4H;

268. resk := K2H;

269. end T4KT2;

270. -------------------calculation AH = a\*K+DH\*(MX\*MRH)------------------

271. Calculate(a => resa, DH => DH, MX => MX, MRH => MRH, K => K, AH => AH);

272. --------------------------passing AH to T1-------------------------

273. T1.T2AH(AH);

274. Put\_Line("T2 finished.");

275. end T2;

276. ----------------------Task 3-------------------------

277. task body T3 is

278. x, a3, resa : Integer;

279. D, Z, resk: Vector;

280. MM, MRH, MX : Matrix;

281. AH, KH, KH2 : VectorH;

282. K2H : Vector2H; K3H : Vector3H;

283. CH, BH, D3H : Vector3H;

284. begin

285. Put\_Line("T3 started.");

286. ----------------------------Enter MX, MM----------------------------

287. VectorInput(Z);

288. VectorInput(D);

289. ----------------------passing MX, MMH to T1---------------------

290. T1.DataT2T1(Z, DH(3\*H+1..N));

291. ----------------------passing MX, MMH to T3---------------------

292. T3.DataT2T3(MX, MMH(3\*H+1..N));

293. ----------------------passing MX, MMH to T4---------------------

294. T4.DataT2T4(MX, MMH(3\*H+1..N));

295. -----------------get B, Z, DH, CH, MRH ot T1, T3, T4-----------------

296. accept DataT1T3 (VH : in VectorH) do

297. BH := VH;

298. end DataT1T3;

299. accept DataT2T3 (V : in Vector; VH : in VectorH) do

300. Z := V;

301. DH := VH;

302. end DataT2T3;

303. accept DataT4T3 (VH : in VectorH; MH : in MatrixH) do

304. CH := VH;

305. MRH := MH;

306. end DataT4T3;

307. ----------------------calculation a2 = BH \* CH----------------------

308. a3 := 0;

309. for i in 3\*H+1..N loop

310. a3 := a3 + BH(i) \* CH(i);

311. end loop;

312. ------------------------passing a3 to T4----------------------------

313. T4.a3T4(a3);

314. -----------accept a2 ot T2 and calculation a3 = a3 + a2------------

315. accept a4T3 (a4 : in Integer) do

316. a3 := a2 + a4;

317. end a4T3;

318. resa := a3;

319. ----------------------pass a to T1, T2, T4--------------------------

320. T1.T1a(resa);

321. T2.T2a(resa);

322. T4.T4a(resa);

323. -----------------calculation KH = (Z\*MMH)------------------

324. for i in 2\*H+1..3\*H loop

325. for j in 1..N loop

326. KH := Z(j)\*MMH(j)(i);

327. end loop;

328. end loop;

329. ----------------- get H ot T1, T2, T3 -----------------

330. accept T1KT3 (VH : in VectorH) do

331. KH:=VH;

332. KH:=K3H;

333. end T1KT3;

334. accept T2KT3 (V2H : in Vector2H) do

335. K3H:= V2H;

336. end T2KT3;

337. accept T4KT3 (V4H : in Vector) do

338. K3H := V4H;

339. resk:=K3H;

340. end T4KT3;

341. -------------------calculation AH = a\*K+DH\*(MX\*MRH)------------------

342. Calculate(a => resa, DH => DH, MX => MX, MRH => MRH, K => K, AH => AH);

343. --------------------------passing A2H to T1-------------------------

344. T1.T3AH(AH);

345. Put\_Line("T3 finished.");

346. end T3;

347. ------------------------------Task 4------------------------------

348. task body T4 is

349. resa, a4: Integer;

350. E, B, C, D, resk : Vector;

351. ME : Matrix;

352. AH, KH, KH4 : VectorH;

353. K2H : Vector2H; K3H : Vector3H;

354. MMH : MatrixH;

355. begin

356. Put\_Line("T4 started.");

357. --------------------------Input C, MR--------------------------

358. VectorInput(C);

359. MatrixInput(MR);

360. ----------------passing CH, MRH to T1, T2, T3, T4------------------

361. T4.DataT2T4(CH, MRH(3\*H+1..N));

362. T1.DataT4T1(CH, MRH);

363. T2.DataT4T2(CH, MRH);

364. T3.DataT4T3(CH, MRH);

365. -----------------get B, Z, DH, CH, MRH ot T1, T3, T4-----------------

366. accept DataT1T4 (VH : in VectorH) do

367. BH := VH;

368. end DataT1T4;

369. accept DataT2T4 (M : in Matrix; MMH : in MatrixH) do

370. MX := M;

371. MMH := MH;

372. end DataT2T4;

373. accept DataT3T4 (V : in Vector; VH : VectorH) do

374. Z := V;

375. DH := VH;

376. end DataT3T4;

377. ----------------------calculation a4 = BH \* CH-----------------------

378. a4 := 0;

379. for i in 3\*H+1..N loop

380. a4 := a4 + BH(i) \* CH(i);

381. end loop;

382. ---------------------accept a3 ot T3-------------------------------

383. accept a3T4 (a3 : in Integer) do

384. a4 := a4 + a3;

385. end a3T4;

386. ------------------------passing a4 to T3-----------------------------

387. T3.a4T3(a);

388. accept T4a (a : in Integer) do

389. resa := a;

390. end T4a;

391. -----------------calculation KH = (Z\*MMH)------------------

392. for i in 3\*H+1..N loop

393. for j in 1..N loop

394. KH4 := Z(j)\*MMH(j)(i);

395. end loop;

396. end loop;

397. ----------------- pass K to T1, T2, T3 -----------------

398. T1.T4KT1(KH4);

399. T2.T4KT2(KH4);

400. T3.T4KT3(KH4);

401. ----------------- get KH ot T1, T2, T3 -----------------

402. accept T1KT4 (VH : in VectorH) do

403. KH:=VH;

404. end T1KT4;

405. accept T2KT4 (V2H : in Vector2H) do

406. KH2:= V2H;

407. end T2KT4;

408. accept T3KT4 (V3H : in Vector3H) do

409. KH4 := V3H;

410. resk:=KH4;

411. end T3KT4;

412. -------------------calculation AH = a\*K+DH\*(MX\*MRH)------------------

413. Calculate(a => resa, DH => DH, MX => MX, MRH => MRH, K => K, AH => AH);

414. --------------------------passing AH to T1-------------------------

415. T1.T3AH(AH);

416. Put\_Line("T4 finished.");

417. end T4;

418. begin

419. Put\_Line("");

420. end tasksProcedure;

421. begin

422. Put\_Line("Lab4 started.");

423. New\_Line;

424. tasksProcedure;

425. Put\_Line("Lab4 finished.");

426. New\_Line;

427. end Lab1;