НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

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

(повна назва кафедри, циклової комісії)

**РОЗРАХУНКОВО - ГРАФІЧНА РОБОТА**

з дисципліни «Паралельне програмування»

(назва дисципліни)

на тему: «Розробка програмного забезпечення

для паралельних комп’ютерних систем»

Студента (ки) 3 курсу ІП-54 групи спеціальності 121 «Програмна інженерія»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Макаренко А.О.\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник доцент Корочкін О.В.

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_

Київ – 2018 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

Напрям підготовки 121 «Програмна інженерія»

(шифр і назва)

***З А В Д А Н Н Я***

НА РГР СТУДЕНТУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Макаренко Антону Олександровичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, імя, 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 21 травня 2018 р.

3. Вхідні дані до роботи

- математична задача: *a = mах(MB\*MC+ ММ)*

- структури ПКС CП та ПКС ЛП

- мови і бібліотеки програмування: *C#, Ada*

- засоби організації взаємодії процесів: ПКС СП – Семафори, мютекси, події, критичні секції, ПКС ЛП – механізм рандеву мови *Ada*

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

- розробка і тестування програми ПРГ1 для ПКС СП

- 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.

6. Дата видачі завдання \_\_12.04.2018\_\_

***КАЛЕНДАРНИЙ ПЛАН***

|  |  |  |
| --- | --- | --- |
| №  з/п | Назва етапів виконання РГР | Строк виконання етапів РГР |
| 1 | Виконання Розділу 1 | 01.05.2018 |
| 2 | Виконання Розділу 2 | 10.05.2018 |
| 5 | Оформлення РГР | 20.05.2018 |
| 6 | Перевірка РГР викладачем | 21.05.2018 |

|  |  |  |
| --- | --- | --- |
| **Студент** | **\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_Макаренко А.О.\_\_** |
|  | ( підпис ) | (прізвище та ініціали) |
| **Керівник роботи** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_Корочкін О.В.\_\_\_\_** |
|  | ( підпис ) | (прізвище та ініціали) |

ЗМІСТ

[ВСТУП 5](#_Toc513578690)

[РОЗДІЛ 1. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП 7](#_Toc513578691)

[1.1. Розробка паралельного математичного алгоритму 7](#_Toc513578692)

[1.2. Розробка алгоритмів процесів 8](#_Toc513578693)

[1.3. Розробка схеми взаємодії процесів 10](#_Toc513578694)

[1.4. Розробка програми ПРГ1 11](#_Toc513578695)

[1.5. Тестування програми ПРГ1 12](#_Toc513578696)

[1.6. Висновки до розділу 1 15](#_Toc513578697)

[РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП 17](#_Toc513578698)

[2.1. Розробка паралельного математичного алгоритму 17](#_Toc513578699)

[2.2. Розробка алгоритмів процесів 18](#_Toc513578700)

[2.3. Розробка схеми взаємодії процесів 20](#_Toc513578701)

[2.4. Розробка програми ПРГ2 20](#_Toc513578702)

[2.5. Тестування програми ПРГ2 21](#_Toc513578703)

[2.6. Висновки до розділу 2 24](#_Toc513578704)

[ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ДО РОБОТИ 26](#_Toc513578705)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 27](#_Toc513578706)

[ДОДАТКИ 28](#_Toc513578707)

[Додаток А 28](#_Toc513578708)

[Додаток Б 29](#_Toc513578709)

[Додаток В 30](#_Toc513578710)

[Додаток Г 32](#_Toc513578711)

[Додаток Д 37](#_Toc513578712)

[Додаток Е 38](#_Toc513578713)

[Додаток Ж 39](#_Toc513578714)

[Додаток И 41](#_Toc513578715)

# ВСТУП

Більшість комп’ютерних систем побудовані на принципі багатоядерності і паралельне програмування швидко стає технологією, яку повинен освоїти і вміти застосовувати будь-який професійний розробник ПЗ. Більшість сучасних мов програмування і бібліотек мають інструменти для розробки програм для паралельного виконання задач.

Традиційні програми пишуться для послідовних обчислень. Для них придумуються алгоритми як посілдовності інструкцій, які виконуються одним процесором комп’ютера. У кожен момент часу може виконуватись тільки одна інструкція, після завершення її виконання починається виконання наступної.

В паралельному програмуванні задачу розбивають на підзадачі, кожна з яких може бути вирішена незалежно. Для розв’язання цих задач використовуються декілька обчислювальних елементів і задачі виконуються паралельно.

В даній роботі будуть розроблені алгоритми обчислення заданого математичного виразу в паралельній комп’ютерній системі зі спільною та локальною пам’яттю.

В розділі 1 розробляється програма ПРГ1 для паралельної комп’ютерної системи зі спільною пам’яттю з чотирма процесорами.

В розділі 2 розробляється програма ПРГ2 для паралельної комп’ютерної системи з локальною пам’яттю з чотирма процесорами. Програма має кільцеву структуру.

Для програм ПРГ1 і ПРГ2 проводиться тестування для визначення коефіцієнтів прискорення та ефективності.

# РОЗДІЛ 1. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП

В даному розділі розробляється програма ПРГ1 для ПКС СП для задаі на мові програмування *C#.* Засоби взаємодії процесів: Семафори, мютекси, події, критичні секції.

Структура паралельної комп’ютерної системи зі спільною пам’яттю (ПКС СП) представлена на рис 1.1.

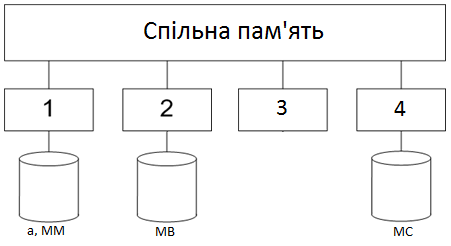


Рис.1.1. Структура ПКС зі спільною пам’яттю

## 1.1. Розробка паралельного математичного алгоритму

Паралельний математичний алгоритм подано у табл. 1.1.

Таблиця 1.1. Паралельний алгоритм задачі

|  |  |
| --- | --- |
| № кроку | Дія |
| 1 |  |

де:

* *N* – розмірність матриць;
* *P* – кількість процесорів;
* ;
* *MBH* — *H* рядкiв матриці *MB*;
* *MMH* — *H* рядкiв матрицi *MM*;
* *a* – результат – доступ до *a* організований через критичну секцію

Спiльнi ресурси: *MC*.

.

Для оцінки необхідного часу обчислень використаємо **теорему Мунро-Петерсона**, яка для комп’ютерної системи з необмеженим числом процесорів формулюється так: якщо виконується обчислення скалярної величини, яка потребує m бінарних операцій, то необхідний час обчислень *tp*:

1. При виконанні операції множення матриць для отримання кожного елементу результуючої матриці, необхідно виконати операцій множення та операцій додавання. Операція додавання двух матриць потребує одного такту. Операція знаходження максимального значення потребує одного такту

Отже

## 1.2. Розробка алгоритмів процесів

Алгоритм задачі

1. Введення
2. Сигнал задачам про введення ()
3. Очікування сигналу про введення даних від задач ()
4. Копіювати
5. Обчислення
6. Очікування сигналу про кінець обчислення від задач ()
7. Виведення результату

Алгоритм задачі

1. Введення
2. Сигнал задачам про введення ()
3. Очікування сигналу про введення даних від задач ()
4. Копіювати
5. Обчислення
6. Сигнал задачі про закінчення обчислення ()

Алгоритм задачі

1. Очікування сигналу про введення даних від задач ()
2. Копіювати
3. Обчислення
4. Сигнал задачі про закінчення обчислення ()

Алгоритм задачі

1. Введення
2. Сигнал задачам про введення ()
3. Очікування сигналу про введення даних від задач ()
4. Копіювати
5. Обчислення
6. Сигнал задачі про закінчення обчислення ()

## 1.3. Розробка схеми взаємодії процесів

На основі алгоритму процесів було розроблено структурну схему взаємодії процесів.

Задача синхронізації вирішується за допомогою семафорів:

* *inputMM* – для сигналізування про введення матриці
* *inputMB* – для сигналізування про введення матриці
* *inputMC* – для сигналізування про введення матриці
* *a2* – для сигналізування про закінчення обчислення задачі
* *a3* – для сигналізування про закінчення обчислення задачі
* *a4* – для сигналізування про закінчення обчислення задачі

Задача взаємного виключення вирішується за допомогою критичних секцій:

* Доступ до матриці за допомогою критичної секції в методі-аксесорі властивості
* Збереження максимального значення (результату) за допомогою критичної секції в методі-мутаторі властивості А

Схема взаємодії процесів наведена у додатку А.

## 1.4. Розробка програми ПРГ1

Програма ПРГ1 розроблена на мові програмування *C#*. Для взаємодії процесів використовуються семафори.

Програма складається з файлів *Program.cs, Data.cs* та *Utils.cs*

*Program.cs* містить в собі метод Main – головний метод програми

*Utils.cs* містить утилітні методи (генерація матриці одиниць)

*Data.cs* містить в собі клас *Data* в якому є:

* Властивість *MC* зв’язана з полем *mc*, має метод аксесор реалізований як критична секція, та метод мутатор.
* Властивість *A* зв’язана з полем *a*, має метод мутатор реалізований як критична секція, та метод аксесор.
* Об'єкти блокування *aLocker* та *mcLocker* для критичних секцій
* Семафори.
* Методи задач *Task1, Task2, Task3, Task4.*

Алгоритми роботи основної програми та процесів наведено у додатках Б, В.

Лістинг розробленої програми наведено у додатку Г.

## 1.5. Тестування програми ПРГ1

Тестування проводиться для оцінки коефіцієнтів прискорення і коефіцієнтів ефективності для розробленої програми.

Для виміру часу використовується об’єкт класу *Stopwatch* мови програмування *C#*. Для початку виміру викликається метод *Start*, для закінчення *Stop,* для отримання часу – властивість *ElapsedMilliseconds.*

Коефіцієнт прискорення показує скорочення часу виконання паралельної програми в паралельній системі з процесорами в порівнянні з часом виконання послідовної програми в однопроцесорній системі :

Коефіцієнт ефективності застосування комп’ютерної системи показує ступінь використання процесорів системи:

Результати тестування і проведених досліджень ефективності розробленої програми наведено в табл.1.2 – 1.4.

Таблиця 1.2. Час виконання програми для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
| 1 |  |  |  |
| 900 | 5,833 | 4,640 | 3,415 | 3,088 |
| 1800 | 85,288 | 73,415 | 53,727 | 42,690 |
| 2400 | 235,150 | 202,666 | 145,964 | 123,621 |

На основі даних із табл. 1.2 виконано розрахунок значень коефіцієнтів прискорення, які наведені в табл. 1.3.

Таблиця 1.3. Значення коефіцієнтів прискорення для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  |  |  |  |
| 900 | 1,000 | 1,257 | 1,708 | 1,889 |
| 1800 | 1,000 | 1,162 | 1,587 | 1,998 |
| 2400 | 1,000 | 1,160 | 1,611 | 1,902 |

На основі даних із табл. 1.3 виконано розрахунок значень коефіцієнтів ефективності, які наведені в табл. 1.4.

Таблиця 1.4. Значення коефіцієнтів ефективності для ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  |  |  |  |
| 900 | 100,000 | 62,856 | 56,935 | 47,223 |
| 1800 | 100,000 | 58,086 | 52,914 | 49,946 |
| 2400 | 100,000 | 58,014 | 53,700 | 47,555 |

За даними табл. 1.2 – 1.4 побудовано графіки зміни часу виконання обчислень та коефіцієнтів прискорення і ефективності в залежності від і (рис. 1.2 – 1.4).

Рис. 1.2. Графік залежності часу виконання програми ПРГ1 від кількості процесорів

Рис. 1.3. Графік залежності коефіцієнту прискорення від кількості процесорів

Рис. 1.4. Графік залежності коефіцієнту ефективності від кількості процесорів

## 1.6. Висновки до розділу 1

1. Реалізовано програму ПРГ1 для ПКС з СП використовуючи мову програмування *C#* з використанням семафорів і критичних секцій.
2. Коефіціент прискорення і коефіціент ефективності залежать від розмірності матриці і кількості процесорів. Коефіціент лежить в межах від до . Коефіціент – від до :

* Максимальне значення досягається при .
* Мінімальне значення досягається при
* Максимальне значення досягається при .
* Мінімальне значення досягається при 0

# РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП

В даному розділі розробляється програма ПРГ2 для ПКС ЛП для задаі на мові програмування *Ada.* Засоби взаємодії процесів: механізм рандеву.

Структура паралельної комп’ютерної системи з локальною пам’яттю (ПКС ЛП) представлена на рис 2.1.

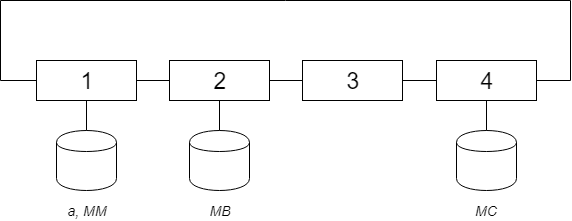


Рис.2.1. Структура ПКС з локальною пам’яттю (кільцева)

## 2.1. Розробка паралельного математичного алгоритму

Паралельний математичний алгоритм подано у табл. 2.1.

Таблиця 2.1. Паралельний алгоритм задачі

|  |  |
| --- | --- |
| № кроку | Дія |
| 1 |  |
| 2 |  |

де:

* *N* – розмірність матриць;
* *P* – кількість процесорів;
* ;
* *MBH* — *H* рядкiв матриці *MB*;
* *MMH* — *H* рядкiв матрицi *MM*;
* *ai –* результат обчислень отриманий з *i-*ї задачі
* *a* – результат.

.

## 2.2. Розробка алгоритмів процесів

Алгоритм задачі

1. Введення
2. Отримати *MCH* від задачі *Т4*
3. Надіслати *MMH* і *MC*до задачі *Т2*
4. Обчислення
5. Отримати *a2* від задачі *Т2*і обчислити *a1,2*як максимальне з *a2*і*a2*
6. Отримати *a3,4* від задачі *Т4*і обчислити *a*як максимальне з *a1,2*і*a3,4*
7. Виведення *a*

Алгоритм задачі

1. Введення
2. Отримати *MMH* і *MC*від задачі *Т2*
3. Надіслати *MMH*, *MC* і *MBH*до задачі *Т3*
4. Обчислення
5. Надіслати *a2*до задачі *Т1*

Алгоритм задачі

1. Отримати *MMH*, *MC* і *MBH*від задачі *Т2*
2. Надіслати *MMH* і *MBH*до задачі *Т4*
3. Обчислення
4. Надіслати *a3*до задачі *Т4*

Алгоритм задачі

1. Введення
2. Надіслати *MC*до задачі *Т1*
3. Отримати *MMH* і *MBH*від задачі *Т3*
4. Обчислення
5. Отримати *a3* від задачі *Т3*і обчислити *a3*,*4*як максимальне з *a3*і*a4*
6. Надіслати *a3,4*до задачі *Т1*

## 2.3. Розробка схеми взаємодії процесів

Розроблена схема взаємодії процесів кільцевої структури ПКС ЛП наведена у додатку Д.

## 2.4. Розробка програми ПРГ2

Програма розроблена на мові *Ada* з використанням механізму рандеву. Складається з фалів *data.adb data.ads main.adb*

Файл *main.ads* містить головну функцію програми яка викликає функцію *Func* і вимірює час її виконання.

Файл *data.adb* містить:

* Функція *MBxMCplusMMmaxPart* яка виконує обчислення
* Функція *Func* яка складається з задач *T1, T2,T3 ,T4*
* Задача *T1* має входи:
  + *acceptMC* – для передачі в задачу матриці *MC*
  + *maxT2* – для передачі в задачу *a2 -* результату обчислення задачі *T2*
  + *maxT4T3* для передачі в задачу *a3,4 -* результату обчислення задач *T3* i *T4*
* Задача *T2* має входи:
  + *acceptMMandMCoutMB* – для передачі в задачу матриць *MC* і *MM,*  з задачі *T­­­1­* та повернення в задачу *T­­­1­* матриці *MB*
* Задача *T3* має входи:
  + *acceptMMandMCandMB* – для передачі в задачу матриць *MC, MB* і *MM.*
* Задача *T4* має входи:
  + *acceptMMandMB* – для передачі в задачу матриць *MB* і *MM.*
  + *maxT3* – для передачі в задачу *a3 -* результату обчислення задачі *T3*

## 2.5. Тестування програми ПРГ2

Тестування проводиться для оцінки коефіцієнтів прискорення і коефіцієнтів ефективності для розробленої програми.

Для виміру часу використовується метод *Clock* мови програмування *Ada*, що міститься в пакеті *Ada.Calendar*, який повертає поточний час. Отримавши поточний час до і після обчислень, тривалість обчислень можна визначити як різницю цих значень.

Коефіцієнт прискорення показує скорочення часу виконання паралельної програми в паралельній системі з процесорами в порівнянні з часом виконання послідовної програми в однопроцесорній системі:

Коефіцієнт ефективності застосування комп’ютерної системи показує ступінь використання процесорів системи:

Результати тестування і проведених досліджень ефективності розробленої програми наведено в табл.2.2 – 2.4.

Таблиця 2.2. Час виконання програми для ПРГ2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
| 1 |  |  |  |
| 900 | 7,615 | 5,675 | 4,229 | 3,867 |
| 1800 | 67,869 | 50,620 | 36,629 | 29,324 |
| 2400 | 163,387 | 119,165 | 90,341 | 80,414 |

На основі даних із табл. 2.2 виконано розрахунок значень коефіцієнтів прискорення, які наведені в табл. 2.3.

Таблиця 2.3. Значення коефіцієнтів прискорення для ПРГ2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  |  |  |  |
| 900 | 1,000 | 1,342 | 1,801 | 1,969 |
| 1800 | 1,000 | 1,341 | 1,853 | 2,314 |
| 2400 | 1,000 | 1,371 | 1,809 | 2,032 |

На основі даних із табл. 2.3 виконано розрахунок значень коефіцієнтів ефективності, які наведені в табл. 2.4.

Таблиця 2.4. Значення коефіцієнтів ефективності для ПРГ2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  |  |  |  |
| 900 | 100,000 | 67,093 | 60,022 | 49,231 |
| 1800 | 100,000 | 67,038 | 61,763 | 57,861 |
| 2400 | 100,000 | 68,555 | 60,285 | 50,796 |

За даними табл. 2.2 – 2.4 побудовано графіки зміни часу виконання обчислень та коефіцієнтів прискорення і ефективності в залежності від і (рис. 2.2 – 2.4).

Рис. 2.2. Графік залежності часу виконання програми ПРГ1 від кількості процесорів

Рис. 2.3. Графік залежності коефіцієнту прискорення від кількості процесорів

Рис. 2.4. Графік залежності коефіцієнту ефективності від кількості процесорів

## 2.6. Висновки до розділу 2

1. Реалізовано програму ПРГ1 для ПКС з СП використовуючи мову програмування *C#* з використанням семафорів і критичних секцій.
2. Коефіціент прискорення і коефіціент ефективності залежать від розмірності матриці і кількості процесорів. Коефіціент лежить в межах від до . Коефіціент – від до :

* Максимальне значення досягається при .
* Мінімальне значення досягається при
* Максимальне значення досягається при .
* Мінімальне значення досягається при 0

# ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ДО РОБОТИ

1. Виконано проектування, розробку та тестування програм ПРГ1 та ПРГ2. ПРГ1 розроблена на мові програмування C# за допомогою семафорів та критичних секцій. ПРГ1 розроблена на мові програмування Ada за допомогою механізму рандеву.
2. Переваги ПКС з СП:

* Простіший алгоритм.
* Економія пам’яті
* Менша кількість повідомлень

1. Переваги ПКС з ЛП:

* Захищеність даних

1. Різниця в виміряному часі виконання більше зумовлена різницею між технологіями мов Ada і C#. Оскільки C# виконується за допомогою CLR, він повільніший. Також для коректних вимірів програму написану на C# потрібно «прогріваи» - виконати перед вимірами методи обчислення, для того щоб JIT-компілятор зміг обробити код цих методів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Рихтер Дж. CLR via C#. – Пер. с англ. – М.: Изд. Дом «Питер», 2013. – 896 с

Жуков І. А. Паралельні та розподілені обчислення: Навч.посіб. / І. А. Жуков, О. В. Корочкін. – Київ : «Корнійчук», 2005. – 226 с.

# ДОДАТКИ

# Додаток А

*Змн.*.

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*1*

*ОТ.6.050103 ІП-54*

*Розроб.*

*Макаренко А.О*

*Перевір.*

*Корочкін О.В.*

*Н. Контр.*

*Затверд.*

*Корочкін О.В.*

*ПРГ1. Схема взаємодії процесів*

*Літ.*

*Аркушів*

*1*

*НТУУ «КПІ ім. І.Сікорського», ФІОТ*

*ІП-32*



# Додаток Б

*Змн.*.

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*1*

*ОТ.6.050103 ІП-54*

*Розроб.*

*Макаренко А.О*

*Перевір.*

*Корочкін О.В.*

*Н. Контр.*

*Затверд.*

*Корочкін О.В.*

*ПРГ1. Алгоритм роботи основної програми*

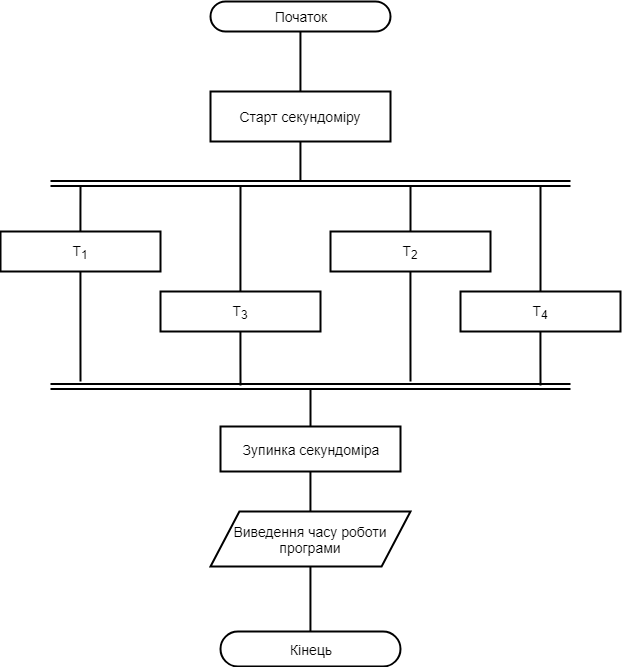
*Літ.*

*Аркушів*

*1*

*НТУУ «КПІ ім. І.Сікорського», ФІОТ*

*ІП-32*



# Додаток В C:\Projects\PP\sem2\rgr\FirstAlgorithm.png

*Змн.*.

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*1*

*ОТ.6.050103 ІП-54*

*Розроб.*

*Макаренко А.О*

*Перевір.*

*Корочкін О.В.*

*Н. Контр.*

*Затверд.*

*Корочкін О.В.*

*ПРГ1. Алгоритм роботи задач T1,T2,T3,T4*

*Літ.*

*Аркушів*

*2*

*НТУУ «КПІ ім. І.Сікорського», ФІОТ*

*ІП-32*

# Додаток Г

Лістинг ПРГ1

Файл **Data.cs**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Text;

using System.Threading;

namespace RGR\_first

{

// a = mах(MB\*MC+ ММ)

public class Data

{

private int size;

private int[][] MB, mc, MM;

private int a = int.MinValue;

private object aLocker = new object();

private object mcLocker = new object();

private int[][] MC

{

get {

lock (mcLocker) {

return mc;

}

}

set {

mc = value;

}

}

private int A

{

set {

lock (aLocker) {

if (value > a)

a = value;

}

}

get {

return a;

}

}

private Semaphore inputMM = new Semaphore(0, 3),

inputMB = new Semaphore(0, 3),

inputMC = new Semaphore(0, 3),

a2 = new Semaphore(0, 1),

a3 = new Semaphore(0, 1),

a4 = new Semaphore(0, 1);

public Data(int size)

{

this.size = size;

}

public void Task1()

{

Console.WriteLine("Task1 started");

MM = Utils.GetMatrixOfOne(size);

inputMM.Release(3);

inputMC.WaitOne();

inputMB.WaitOne();

var MC1 = MC;

A = MBxMCplusMMmaxPart(MB, MC1, MM, 0, MB.Length / 4);

a2.WaitOne();

a3.WaitOne();

a4.WaitOne();

Console.WriteLine("Task1 finished");

Console.WriteLine($"a = {A}");

}

public void Task2()

{

Console.WriteLine("Task2 started");

MB = Utils.GetMatrixOfOne(size);

inputMB.Release(3);

inputMC.WaitOne();

inputMM.WaitOne();

var MC2 = MC;

A = MBxMCplusMMmaxPart(MB, MC2, MM, MB.Length / 4, MB.Length / 2);

a2.Release();

Console.WriteLine("Task2 finished");

}

public void Task3()

{

Console.WriteLine("Task3 started");

inputMB.WaitOne();

inputMC.WaitOne();

inputMM.WaitOne();

var MC3 = MC;

A = MBxMCplusMMmaxPart(MB, MC3, MM, MB.Length / 2, 3 \* MB.Length / 4);

a3.Release();

Console.WriteLine("Task3 finished");

}

public void Task4()

{

Console.WriteLine("Task4 started");

MC = Utils.GetMatrixOfOne(size);

inputMC.Release(3);

inputMB.WaitOne();

inputMM.WaitOne();

var MC4 = MC;

A = MBxMCplusMMmaxPart(MB, MC4, MM, 3 \* MB.Length / 4, MB.Length);

a4.Release();

Console.WriteLine("Task4 finished");

}

public int MBxMCplusMMmaxPart(int[][] MB, int[][] MC, int[][] MM, int from, int to)

{

int max = int.MinValue;

int t1;

for (int k = 0; k < MB.Length; k++) {

for (var i = from; i < to; i++) {

int t = 0;

for (int j = 0; j < MB.Length; j++)

t += MB[i][j] \* MC[j][k];

t1 = MM[i][k] + t;

if (t1 > max)

max = t1;

}

}

return max;

}

}

}

Файл **Utils.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace RGR\_first

{

static class Utils

{

public static int[] GetVectorOfOne(int size)

{

int[] result = new int[size];

for (int i = 0; i < size; i++) {

result[i] = 1;

}

return result;

}

public static int[][] GetMatrixOfOne(int size)

{

int[][] result = new int[size][];

for (int i = 0; i < size; i++) {

result[i] = GetVectorOfOne(size);

}

return result;

}

}

}

Файл **Program.cs**

using System;

using System.Diagnostics;

using System.Threading;

namespace RGR\_first

{

class Program

{

// a = mах(MB\*MC+ ММ)

static void Main(string[] args)

{

Console.WriteLine("a = mах(MB\*MC+ ММ)");

Data data = new Data(900);

var t1 = new Thread(data.Task1);

var t2 = new Thread(data.Task2);

var t3 = new Thread(data.Task3);

var t4 = new Thread(data.Task4);

var stopwatch = new Stopwatch();

stopwatch.Start();

t1.Start();

t2.Start();

t3.Start();

t4.Start();

t1.Join();

stopwatch.Stop();

Console.WriteLine($"Elapsed time: {stopwatch.ElapsedMilliseconds} ms " +

$"({stopwatch.Elapsed.Minutes}:{stopwatch.Elapsed.Seconds}.{stopwatch.Elapsed.Milliseconds})");

Console.ReadLine();

}

}

}

# Додаток Д

*Змн.*.

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*1*

*ОТ.6.050103 ІП-54*

*Розроб.*

*Макаренко А.О*

*Перевір.*

*Корочкін О.В.*

*Н. Контр.*

*Затверд.*

*Корочкін О.В.*

*ПРГ2. Схема взаємодії процесів*

*Літ.*

*Аркушів*

*1*

*НТУУ «КПІ ім. І.Сікорського», ФІОТ*

*ІП-32*



# Додаток ЕC:\Users\demo6\Downloads\Алгоритм ПКС с общей памятью Main.png

*Змн.*.

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*1*

*ОТ.6.050103 ІП-54*

*Розроб.*

*Макаренко А.О*

*Перевір.*

*Корочкін О.В.*

*Н. Контр.*

*Затверд.*

*Корочкін О.В.*

*ПРГ2. Алгоритм роботи основної програми*

*Літ.*

*Аркушів*

*1*

*НТУУ «КПІ ім. І.Сікорського», ФІОТ*

*ІП-32*

# Додаток Ж C:\Projects\PP\sem2\rgr\SecondAlgorithm.png

*Змн.*.

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*1*

*ОТ.6.050103 ІП-54*

*Розроб.*

*Макаренко А.О*

*Перевір.*

*Корочкін О.В.*

*Н. Контр.*

*Затверд.*

*Корочкін О.В.*

*ПРГ2. Алгоритм роботи задач T1,T2,T3,T4*

*Літ.*

*Аркушів*

*2*

*НТУУ «КПІ ім. І.Сікорського», ФІОТ*

*ІП-32*

# Додаток И

Лістинг ПРГ2

GNAT GPL 2017 (20170515-63)

Copyright 1992-2017, Free Software Foundation, Inc.

Compiling: data.ads

Source file time stamp: 2018-05-12 23:22:52

Compiled at: 2018-05-20 03:55:05

1. with Ada.Synchronous\_Task\_Control;

2. use Ada.Synchronous\_Task\_Control;

3.

4. generic

5. n: Integer;

6. package Data is

7.

8. type Vector is array(0..n) of Integer;

9. type Matrix is array(0..n) of Vector;

10.

11. procedure Func;

12.

13. end Data;

13 lines: No errors

GNAT GPL 2017 (20170515-63)

Copyright 1992-2017, Free Software Foundation, Inc.

Compiling: data.adb

Source file time stamp: 2018-05-19 10:09:36

Compiled at: 2018-05-20 03:55:02

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

2. with Ada.Integer\_Text\_IO;

3. with Ada.Synchronous\_Task\_Control;

4. use Ada.Integer\_Text\_IO;

5. use Ada.Synchronous\_Task\_Control;

6.

7. -- a = max(MB\*MC + MM)

8.

9. package body Data is

10.

11. function MBxMCplusMMmaxPart(MB, MC, MM: Matrix; start, finish: Integer) return Integer is

12. temp: Integer;

13. max: Integer;

14. begin

15. max:= Integer'First;

16. loop1:

17. for i in start .. finish loop

18.

19. loop2:

20. for j in 0 .. MB'Length - 1 loop

21. temp:=0;

22. loop3:

23. for k in 0 .. MB'Length - 1 loop

24. temp:=temp + MB(i)(k) \* MC(k)(j);

25. end loop loop3;

26. temp:= temp + MM(i)(j);

27. if temp > max then

28. max := temp;

29. end if;

30. end loop loop2;

31. end loop loop1;

32. return max;

33. end MBxMCplusMMmaxPart;

34.

35. function getVectorOfOne return Vector is

36. result: Vector;

37. begin

38. for i in 0..N loop

39. result(i) := 1;

40. end loop;

41. return result;

42. end getVectorOfOne;

43.

44. function getMatrixOfOne return Matrix is

45. result: Matrix;

46. begin

47. for i in 0..N loop

48. result(i) := getVectorOfOne;

49. end loop;

50. return result;

51. end getMatrixOfOne;

52.

53.

54. -- a = max(MB\*MC + MM)

55. procedure Func is

56. task T1 is

57. entry acceptMC (m: in Matrix);

58. entry maxT2 (m: in Integer);

59. entry maxT4T3 (m: in Integer);

60. pragma Storage\_Size (200\_000\_000);

61. pragma Task\_Name ("Task1");

62. end T1;

63.

64. task T2 is

65. entry acceptMMandMCoutMB (m1, m2: in Matrix; m3: out Matrix);

66. pragma Storage\_Size (200\_000\_000);

67. pragma Task\_Name ("Task2");

68. end T2;

69.

70. task T3 is

71. entry acceptMMandMCandMB (m1, m2, m3: in Matrix);

72. pragma Storage\_Size (200\_000\_000);

73. pragma Task\_Name ("Task3");

74. end T3;

75.

76. task T4 is

77. entry acceptMMandMB (m1, m2: in Matrix);

78. entry maxT3 (m: in Integer);

79. pragma Storage\_Size (200\_000\_000);

80. pragma Task\_Name ("Task4");

81. end T4;

82.

83. task body T1 is

84. MB: Matrix;

85. MC: Matrix;

86. MM: Matrix;

87. max: Integer;

88. begin

89. Put\_Line("t1 started");

90.

91. MM:=getMatrixOfOne;

92.

93. accept acceptMC (m: in Matrix) do

94. MC:=m;

95. end acceptMC;

96.

97. T2.acceptMMandMCoutMB(MM, MC, MB);

98.

99. max:=MBxMCplusMMmaxPart(MB, MC, MM, 0, MB'length/4-1);

100.

101. accept maxT2 (m: in Integer) do

102. if m > max then

103. max := m;

104. end if;

105. end maxT2;

106.

107. accept maxT4T3 (m: in Integer) do

108. if m > max then

109. max := m;

110. end if;

111. end maxT4T3;

112. Put\_Line("t1 finished");

113. Put\_Line("a = " & Integer'Image(max) );

114. end T1;

115.

116. task body T2 is

117. MB: Matrix;

118. MC: Matrix;

119. MM: Matrix;

120. max: Integer;

121. begin

122. Put\_Line("t2 started");

123.

124. MB:=getMatrixOfOne;

125.

126. accept acceptMMandMCoutMB (m1, m2: in Matrix; m3: out Matrix) do

127. MC:=m1;

128. MM:=m2;

129. m3:=MB;

130. end acceptMMandMCoutMB;

131.

132. T3.acceptMMandMCandMB(MB, MC, MM);

133.

134. max:=MBxMCplusMMmaxPart(MB, MC, MM, MB'length/4, MB'length/2-1);

135.

136. T1.maxT2(max);

137.

138. Put\_Line("t2 finished");

139. end T2;

140.

141. task body T3 is

142. MB: Matrix;

143. MC: Matrix;

144. MM: Matrix;

145. max: Integer;

146. begin

147. Put\_Line("t3 started");

148.

149. accept acceptMMandMCandMB (m1, m2, m3: in Matrix) do

150. MB:=m1;

151. MC:=m2;

152. MM:=m3;

153. end acceptMMandMCandMB;

154.

155. T4.acceptMMandMB(MM, MB);

156.

157. max:=MBxMCplusMMmaxPart(MB, MC, MM, MB'length/2, 3\*MB'length/4-1);

158.

159. T4.maxT3(max);

160.

161. Put\_Line("t3 finished");

162.

163. end T3;

164.

165. task body T4 is

166. MB: Matrix;

167. MC: Matrix;

168. MM: Matrix;

169. max: Integer;

170. begin

171. Put\_Line("t4 started");

172.

173. MC:= getMatrixOfOne;

174. T1.acceptMC(MC);

175.

176. accept acceptMMandMB (m1, m2: in Matrix) do

177. MB:=m1;

178. MM:=m2;

179. end acceptMMandMB;

180.

181. max:=MBxMCplusMMmaxPart(MB, MC, MM, 3\*MB'length/4, MB'length-1);

182.

183. accept maxT3 (m: in Integer) do

184. if m > max then

185. max := m;

186. end if;

187. end maxT3;

188.

189. T1.maxT4T3(max);

190.

191. Put\_Line("t4 finished");

192. end T4;

193.

194.

195. begin

196. Put("");

197. end Func;

198. end Data;

Compiling: data.ads

Source file time stamp: 2018-05-12 23:22:52

Compiled at: 2018-05-20 03:55:02

1. with Ada.Synchronous\_Task\_Control;

2. use Ada.Synchronous\_Task\_Control;

3.

4. generic

5. n: Integer;

6. package Data is

7.

8. type Vector is array(0..n) of Integer;

9. type Matrix is array(0..n) of Vector;

10.

11. procedure Func;

12.

13. end Data;

198 lines: No errors

GNAT GPL 2017 (20170515-63)

Copyright 1992-2017, Free Software Foundation, Inc.

Compiling: main.adb

Source file time stamp: 2018-05-19 10:09:12

Compiled at: 2018-05-20 03:54:16

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

2. with Data;

3. with Ada.Integer\_Text\_IO;

4. with Ada.Synchronous\_Task\_Control;

5. use Ada.Integer\_Text\_IO;

6. use Ada.Synchronous\_Task\_Control;

7. with Ada.Calendar; use Ada.Calendar;

8. -- a = max(MB\*MC + MM)

9.

10. procedure Main is

11. package data1 is new data(2400);

12. use data1;

13. startTime: Time;

14. endTime: Time;

15. begin

16. Put\_Line("started");

17. Put\_Line("a = max(MB\*MC + MM)");

18. startTime := Clock;

19. Func;

20. endTime := Clock;

21. Put("Elapsed: ");

22. Put(Duration'Image(endTime - startTime));

23. end Main;

23 lines: No errors