НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

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

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЕКТ**

з дисципліни «Паралельне програмування»

(назва дисципліни)

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

Студента 3 курсу \_\_ІП-42\_\_ групи

напряму підготовки 050103 «Програмна інженерія»

\_ Кафтанатія Богдана Сергійовича \_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник доцент Корочкін О.В.

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

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

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

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

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали

Київ- 2017 р.

Національний технічний університет України

“Київський політехнічний інститут”

Факультет (інститут) інформатики та обчислювальної техніки

( повна назва )

Кафедра обчислювальної техніки

( повна назва )

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 6.050103 «Програмна інженерія»

# (шифр і назва)

## ***З А В Д А Н Н Я***

### НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

\_\_\_\_\_\_\_\_Кафтанатію Богдану Сергійовичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(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 11 травня 2017 р.

3. Вхідні дані до роботи

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

5. Перелік графічного матеріалу

7. Дата видачі завдання \_\_\_\_\_\_\_\_хх\_\_\_\_\_

#### **КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| №  з/п | Назва етапів виконання КР | Строк виконання етапів КР |
| 1 | Виконання розділу 1 | 13.03.2017 |
| 2 | Виконання розділу 2 | 23.03.2016 |
| 3 | Виконання розділу 3 | 23.04.2016 |
| 4 | Оформлення КР | 8.05.2016 |
| 5 | Перевірка КР викладачем | 11.05.2016 |
| 6 | Захист КР | 18.05.2016 |

**Студент \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

( підпис ) (прізвище та ініціали)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

( підпис ) (прізвище та ініціали)

ЗМІСТ

ВСТУП…………………………………………………………………………… 3

РОЗДІЛ 1. АНАЛІЗ ЗАСОБІВ РОБОТИ З ПРОЦЕСАМИ В МОВІ АДА…... 5

* 1. Засоби програмування процесів…………………………………….
  2. Засоби организації взаємодії процесів через спільні змінні……….
  3. Засоби организації взаємодії процесів через посилання повідомлень
  4. Висновки до розділу 1………………………………………………….

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС ОП……………………

* 1. Розробка паралельного математичного алгоритму……………………
  2. Розробка алгоритмів процесів………………………………………….
  3. Розробка схеми взаємодії процесів…………………………………….
  4. Розробка програми ПРГ1……………………………………………….
  5. Тестування програми ПРГ1 …………………………………………….
  6. Висновки до розділу 2 …………………………………………………..

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП ……………………

* 1. Розробка паралельного математичного алгоритму………………….
  2. Розробка алгоритмів процесів…………………………………………
  3. Розробка схеми взаємодії процесів…………………………………..
  4. Розробка програми ПРГ2………………………………………………
  5. Тестування програми ПРГ2……………………………………………
  6. Висновки до розділу 3…………………………………………………

ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ДО РОБОТИ………………………37

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………… 39

ДОДАТКИ……………………………………………………............................. 41

**ВСТУП**

**РОЗДІЛ 1. ПОРІВНЯННЯ РЕАЛІЗАЦІЇ МЕХАНІЗМУ МОНІТОРІВ СЕКЦІЇ В МОВАХ І БІБЛІОТЕКАХ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ**

Листів

Лист

1

Літера

6.050102

ІП-42

**Розробка програмного забезпечення для паралельних комп’ютерних систем**

НТУУ КПІ 17 4211 - 000 ПЗ

Зм.

Виконав

Керівник

Конс.

Н. контр.

Зав. каф.

№ докум.

Кафтанатій Б.С.

Корочкін О.В.

Корочкін О.В.

Стіренко С.Г

Лит

Підпис

Дата

1

* 1. **Загальні поняття**

Ідея монітору, яку запропонував Б. Хансен і розвинув Ч. Хоар, грунтується на об’єднанні змінних, що описують спільний ресурс, і дій, які визначають засобі доступу до спільного ресурсу [12][36]. Монітор – програмний модуль, що містить (ховає) змінні та надає процедури для роботи з ними, причому доступ до змінних можливий тільки через процедури монітора.

Монітор – засіб розподілу ресурсів і взаємодії процесів. Це призначення монітора реалізується за допомогою властивостей, якими наділені процедури монітора. Характерна особливість процедур монітора – взаємне виключення ними одне одного. У будь-який момент часу може виконуватись тільки одна процедура монітора. З спроби виклику іншим процесом процедури, що виконується, або іншої процедури монітора цей процес блокується і розміщується в черзі блокованих процесів доти, поки активний процес не закінчить виконання процедури монітора. Тобто в моніторі не може «знаходитись» більше одного процесу. Така властивість процедур монітора забезпечує взаємне виключення процесів, які працюють з монітором.

У моніторі декларують локальні змінні (спільні змінні), які захищені монітором, і процедури монітора. Значення локальних змінних можуть бути встановлені під час створення монітора. Далі значення цих змінних можуть бути прочитані або зміненні процесами тільки за допомогою процедур, визначених в моніторі.

Загальна структура монітора:

monitor Ім’я монітора;

- - Опис локальних даних

- - Опис процедур для доступу по даних

begin

- - Ініціалізація локальних даних

end Ім’я монітора;

Властивості процедур монітора забезпечують вирішення завдання взаємного виключення за доступу до спільних ресурсів, об’явленими в моніторі. При цьому монітор формує чергу процесів, які викликали процедури монітора і є блокованими через зайнятість монітора (тобто спільного ресурсу).

Приклад монітора:

monitor Склад;

Товар: Data;

procedure На\_Склад (T: in Data);

procedure Зі\_Складу (T: out Data);

begin

Товар:= 0.0ж

end Склад;

* 1. **Монітори в мові Ада**

Концепцію моніторів у стандарті мови Ада реалізовано у вигляді спеціальних програмних модулів – захищених модулів (protected units) [16],[51],[65]. Їх призначення – розширення можливості мови для програмування паралельних процесів, зокрема, для вирішення проблеми доступу до спільних ресурсів і синхронізації процесів. Крім того, захищені модулі забезпечують підтримку різних парадигм систем реального часу, для розроблення яких мову Ада використовують в першу чергу.

Спільні дані і операції над ними (захищені операції) об’єднуються в захищеному модулі, аналогічно тому, як це робиться в інших модулях мови Ада – пакетах. Доступ до спільних ресурсів можливий тільки через захищені операції, які мають властивості, що дозволяють вирішити завдання взаємного виключення під час роботи зі спільними ресурсами.

Як і всі модулі в мові, захищені модулі складаються зі специфікації і тіла.

Специфікація захищеного модуля:

PROTECTED [TYPE] Ім’я\_Захищеного\_Модуля [Дискримінант]

- - Опис\_Захищених\_Операцій

[PRIVATE]

- - Опис\_Захищених\_Елементів

END Ім’я\_Захищеного\_Модуля;

Захищенні модулі – це:

* + *Захищені функції,*
  + *Захищені процедури,*
  + *Захищені входи.*

Захищенні функції забезпечують доступ тільки до читання захищених елементів. Але дозволяють робити це одночасно всім процесам автоматичним копіюванням елементів, які зчитуються. Це порушую головну властивість процедур монітора, яка дозволяє знаходитися в моніторі тільки одному процесу, але «порушення» дозоляє скоротити час доступу до захищених елементів і не має будь-яких наслідків, оскільки зміна даних заборонена і не виконується.

Захищені процедури забезпечують ексклюзивний доступ до захищених елементів через читання і запис.

Захищенні входи забезпечують ті самі функції, що й захищені процедури, додатково реалізуючи за допомогою бар’єрів ексклюзивний (умовний) доступ до тіла захищеного входу. Це дозволяє реалізувати за допомогою входів вирішення завдання умовної синхронізації.

Приватна частина специфікації обмежує видимість захищених елементів: операцій і об’єктів, що описані в ній. Спільні дані, доступ до яких контролюється захищеним модулем, описуються в приватній частині його специфікації.

Приклад 1.1 Специфікації захищеного модуля

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

--- Ада. Захищений модуль --

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

protected Контроль is

procedure Включення; -- захищені підпрограми

function Перемкнути (X: Float);

end Контроль.

protected type Сенсор is

entry Чекати; -- захищенні входи

entry Сигнал;

procedure Зміна\_Стану(x: in float);

function Замір\_Стану return float;

private

Прп: Boolean := False; -- Прапор

Стан: Float;

end

protected Блок232(Номер: in Positive) is

entry Параметри(X: out integer);

procedure Зміна\_Параметру(Y: in integer);

private

- - захищений елемент

Об’єкт: array(1 .. Номер) of integer;

end Блок232;

Тіло захищеного модуля реалізує захищені операції, які об’явлені і його специфікації, використовуючи для цього локальні ресурси, які можуть бути об’явлені в тілі модуля.

PROTECTED BODY Ім’я\_Захищеного\_Модуля IS

- - Локальні\_Описи

BEGIN

- - Реалізація захищених операцій і захищених елементів

END Ім’я\_Захищеного\_Модуля;

Захищені процедури і фнкції реалізуються в тілі захищеного модкля, як це робиться в тілі пакета. На відмінну від модулів-задач, реалізація захищеного входу в тілі захищеного модуля не пов’язана з оператором приймання accept, а виконується за допомогою тіла входу, в якому обов’язково використовується бар’єр.

Описання тіла захищеного входу:

ENTRY Ім’я\_Захищеного\_Входу WHEN Умова IS

BEGIN

. . . - - Послідовність\_Операторів

END Ім’я\_Захищеного\_Входу;

Конструкція When Умова – це бар’єр, де Умова – логічний вираз, який визначає відкритий або закритий вхід. Перевірка умови в бар’єрі виконується під час виклику захищеного входу. Якщо значення виразу Умова дорівнює true, то вхід відкритий і виконується тіло захищеного входу, інакше вхід є зачинений і виконання процесу, який викликав цей вхід, блокується до того часу, поки значення виразу Умова в бар’єрі не буде змінено в true іншою задачею за допомогою захищеної процедури або іншого захищеного входу.

В прикладі 1.2. наведено реалізацію тіла захищеного модуля Сенсор.

Приклад 1.2 Специфікації захищеного модуля

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

--- Ада. Тіло захищеного модуля ---

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

protected body Сенсор is

- - тіло захищеного входу з бар’єром

entry Чекати when Прп is

begin

Прп:= False;

end Чекати;

- - тіло захищеного входу з бар’єром

entry Сигнал when not Прп is

begin

Прп:= True;

end Сигнал;

procedure Зміна\_Стану(x : in float) is

begin

Стан := x;

end Зміна\_Стану;

function Замір\_Стану return float is

begin

return Стан;

end Замір\_Стану;

end Сенсор;

Структур захищеного модуля Сенсор показано на рис. 1.1. Захищену функцію Замір\_Стану() і процедуру Замір\_Стану() зображено справа, захищені входи Чекати() і Сигнал() – зліва. Захищені елементи (Стан, Прп) зображено всередині захищено модуля в овалі.

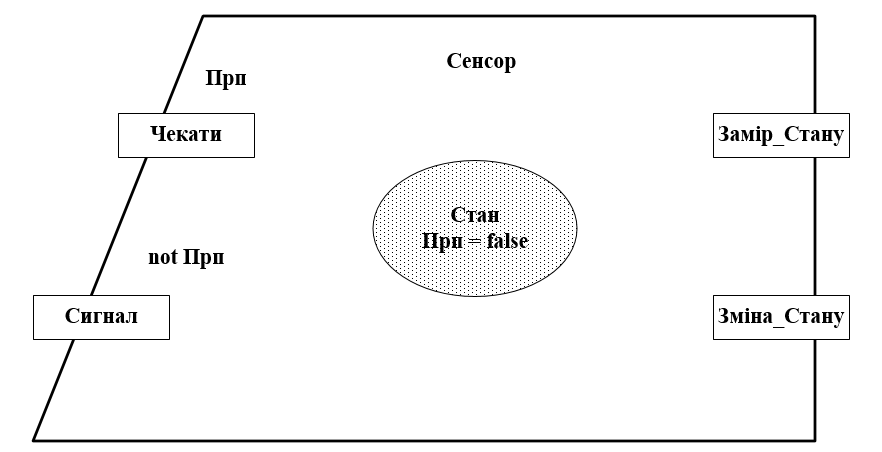


Рис. 1.1. Структура захищеного модуля

Виклик захищеної функції дозволяє процесу зчитувати дані із захищеного модуля. Кілька процесів можуть виконувати таке читання одночасно, викликаючи потрібні функції. Під час виконання читання в тілі захищеної функції заборонено зміну даних. Тіло захищеної функції може містити виклик іншої захищеної функції, але не виклик захищеної процедури.

Виклик захищеної процедури дозволяє процесу як читати, так і змінювати інформацію в захищеному модулі. На відмінну від захищеної функції під час виконання захищеної процедури дозволяється змінювати дані. Якщо кілька процесів виконують виклик захищених процедур, то тільки один з них отримує можливість роботи з викликаною процедурою. У тілі захищеної процедури дозволено виклик як захищеної функції, так і захищеної процедури.

Виклик закритого захищеного входу приводить до блокування процесу до того часу, поки вхід не стане відкритим, тобто умовний вираз в бар’єрі набуде значення True. Це може статися в разі виконання іншим процесом потрібної захищеної операції, пов’язаної зі змінними, використаними в бар’єрі.

Блокований процес розміщується в черзі, яка пов’язана із входом, а також за змінними, що використовуються в бар’єрі. Якщо вхід відкритий, то виконується тіло входу.

Приклад 1.3.

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

--- Ада. Захищений модуль у завданні взаємного виключення ---

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

protected Буфер is

procedure Додати(Вклад: out Positive);

procedure Видалити(Вклад: out Positive);

private

Лічильник: Integer:= 0;

end Буфер;

- - тіло захищеного модуля

protected body Буфер is

procedure Додати(Вклад: out Positive) is

begin

Лічильник := Лічильник + 1;

Вклад := Лічильник;

end Додати;

procedure Видалити(Вклад: out Positive) is

begin

Лічильник := Лічильник - 1;

Вклад := Лічильник;

end Видалити;

end Буфер;

Задачі додають або зменшують значення змінної Лічильник, викликаючи процедури Додати і Видалити захищеного модуля Буфер:

Буфер.Додати(Зарплата); Буфер.Видалити(Плата);

Захищений модуль Буфер гарантує синхронізований доступ задач до захищеної змінної Лічильник. Черги під час роботи із захищеним модулем не створюються, оскільки використовуються тільки захищені процедури, а не захищені входи.

У прикладі 1.4. захищений модуль Вклад виконує роль буфера, куди задачі Клієнт\_А і Клієнт\_В записують і звідки читають дані. Захищений модуль Вклад повинен забезпечувати взаємовиключний доступ задач до спільного ресурсу, яким є змінна Рахунок, а також синхронізацію процесів залежно від стану ресурсу.

Приклад 1.4.

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

--- Ада. Захищений модуль у завданні взаємного виключення та ---

--- синхронізації процесів ---

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

protected Вклад is

entry В\_Банк(M : in Гроші);

entry З\_Банку(M : out Гроші);

private

Рахунок: Гроші;

Прапор: Boolean := False;

end Вклад;

- - - - - - - - -

protected body Вклад is

entry В\_Банк(M : in Гроші) when Прапор = False is

begin

Рахунок := M;

Прапор := True;

end В\_Банк;

entry З\_Банку(M : out Гроші) when Прапор = true is

M := Рахунок;

Прапор := False;

end З\_Банку;

end Вклад;

- - - - - - - - -

task Клієнт\_А;

task body Клієнт\_А is

Дохід : Гроші;

begin

. . . .

- - виклик входу В\_Банк

Вклад.В\_Банк(Дохід);

. . . .

end Клієнт\_А;

- - - - - - - - -

task Клієнт\_В;

task body Клієнт\_В is

Сплата : Гроші;

begin

. . . .

- - виклик входу З\_Банку

Вклад.З\_Банку(Сплата);

. . . .

end Клієнт\_В;

* 1. **Монітори в мові Java**