МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

**для самостійної роботи студентів
і виконання контрольної роботи №1 з дисципліни
«Технології розподілених систем та паралельних обчислень»**

для студентів 5-го курсу заочної форми навчання.
рівень підготовки “бакалавр”

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

для самостійної роботи студентів
і виконання контрольної роботи №1 з дисципліни
«Технології розподілених систем та паралельних обчислень»

для студентів заочної форми навчання.
рівень підготовки “бакалавр”

ЗАТВЕРДЖЕНО

на засіданні робочої групи
заочної та післядипломної освіти

Методичні вказівки для самостійної роботи і для виконання контрольної роботи №1 з курсу «Технології розподілених систем та паралельних обчислень» для студентів 5-го курсу заочної форми навчання. Рівень підготовки “бакалавр”. Вказівки призначені для надання допомоги при самостійному вивченні дисципліни. / Рольщиков В.Б. Одеса, ОДЕКУ, 2018, 40 с., укр. мова.

Укладач ст. викл. каф. інформаційних технологій Рольщиков В.Б.

ЗМІСТ

Ці методичні вказівки призначені для надання допомоги студентам 5-го курсу заочного факультету Одеського державного екологічного університету, які навчаються за спеціальністю «Комп'ютерні науки і інформатика», при самостійному вивченні розділів курсу «Технології розподілених систем та паралельних обчислень» і при виконанні контрольної роботи з цього курсу.

Опис навчальної дисципліни

<table>
<tr><td rowspan="2">Найменування показників</td><td rowspan="2">Галузь знань, напрям підготовки, освітньо-кваліфікаційний рівень</td><td>Характеристика навчальної дисципліни</td></tr>
<tr><td>заочна форма навчання</td></tr>
<tr><td rowspan="4">Кількість кредитів– 7,5</td><td>Галузь знань
_____
(шифр і назва)</td><td rowspan="3">Нормативна</td></tr>
<tr><td>Напрям підготовки
_____
(шифр і назва)</td></tr>
<tr><td rowspan="3">Спеціальність (професійне спрямування):
122 Комп'ютерні науки
_____
(шифр і назва)</td><td>Рік підготовки:</td></tr>
<tr><td rowspan="2">Змістових модулів – 6
Теоретичних – 3
Практичних – 3</td><td>5-й</td></tr>
<tr><td>Семестр</td></tr>
<tr><td rowspan="3">Індивідуальне науково-дослідне завдання:
_____
(назва)</td><td rowspan="4">Освітньо-кваліфікаційний рівень:
бакалавр</td><td></td></tr>
<tr><td>Лекції</td></tr>
<tr><td>8 год.</td></tr>
<tr><td rowspan="3">Загальна кількість годин – 195</td><td>Практичні, семінарські</td></tr>
<tr><td>– год.</td></tr>
<tr><td>Лабораторні</td></tr>
<tr><td>8 год.</td><td>Самостійна робота</td></tr>
<tr><td></td><td></td><td>179 год.</td></tr>
<tr><td></td><td></td><td>Вид контролю: іспит</td></tr>
</table>

Співвідношення кількості годин аудиторних занять до самостійної і індивідуальної роботи для заочної форми навчання становить: 0,09

1 Мета та завдання навчальної дисципліни

Мета: надати студентам основні відомості про архітектуру сучасних паралельних обчислювальних систем, принципи функціонування окремих складових систем та взаємозв'язки між складовими. Ознайомити студентів з методами розпаралелювання різних чисельних методів. Навчити студентів із засобами паралельного програмування як у локальних обчислювальних мережах, так й GRID-системах.

Предмет та практична значимість дисципліни: Паралельні обчислення є перспективною (і дуже привабливою) областю застосування обчислювальної техніки і являють собою складну науково-технічну область діяльності, методи й програми паралельного рішення завдань обробки даних відносяться до числа важливих кваліфікаційних характеристик сучасного фахівця із прикладної математики, інформатиці й обчислювальній техніці.

Завдання дисципліни: роз'яснення найбільш складних питань шляхом читання курсу лекцій, винесенням менш складних питань на вивчення під час самостійної роботи студентів, а, також, закріплення теоретичного матеріалу за допомогою і під час виконання циклу лабораторних робіт, котрі охоплюють найбільш важливі теми дисципліни.

За результатами вивчення навчальної дисципліни студент повинен

знати: архітектуру сучасних паралельних обчислювальних систем як мережевих, в тому числі й GRID, так й багатопроцесорних систем з пам'яттю загального використання, основні принципи розпаралелювання задач та паралельні алгоритми, що застосовуються для вирішення в складних задачах моделювання в науці і техніці;

вміти: засобами мови програмування mpC та засобами бібліотеки функцій MPI створювати паралельні програми різного призначення, тобто СЛАР, ЗДР, складати завдання для вирішення складних задач у GRID-системах.

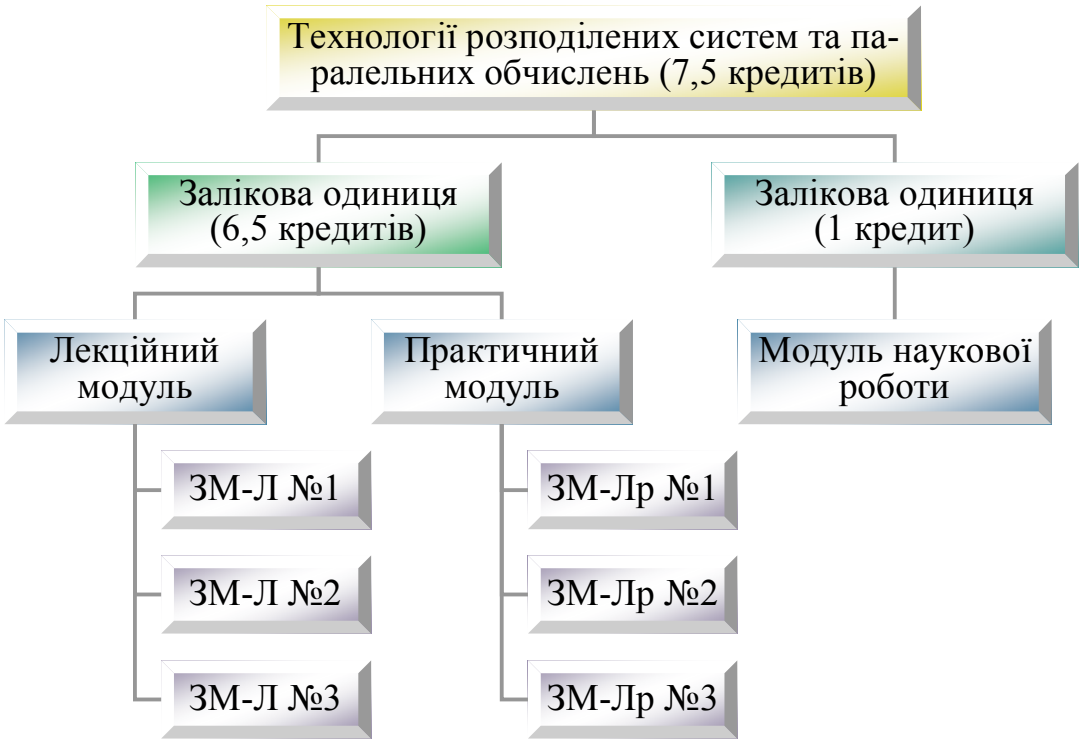
Дисципліна базується на наступних дисциплінах «Алгоритмічні мові та основи програмування», «Об'єктно-орієнтоване програмування», «Чисельні методи в інформатиці», «Операційні системи».

Дисципліна забезпечує вивчення: «Grid-технології та хмарні обчислення».

Компетенції щодо навчальної дисципліни:

- **знання** основ парадигми паралельного програмування, характеристик і властивостей різних систем призначених для виконання паралельних обчислень (архітектури суперкомп'ютерів, кластер них систем, GRID);
- **вміння** створювати алгоритми паралельного розв'язання різних проблем комп'ютерного моделювання фізичних та соціологічних систем, складання паралельних програм за допомогою бібліотек стандарту MPI і функцій системи розробки PVM.

2 Схема навчальної дисципліни



3 Програма навчальної дисципліни

Змістовий модуль 1. 3.13.02 Паралельні обчислювальні методи.

Тема 1. 3.13.02.01 Організація паралельних обчислень з використанням наявних технологій (PVM, MPI).

Тема 2. 3.13.02.02 Паралельні перетворення арифметичних виразів.

Тема 3. 3.13.02.03 Базові алгоритми паралельних обчислень

Тема 4. 3.13.02.04 Паралельні методи розв'язання СЛАР

Тема 5. 3.13.02.05 Паралельні методи розв'язання систем нелінійних рівнянь.

Тема 6. 3.13.02.06 Ефективність паралельних обчислювальних методів під час розв'язання нелінійної задачі Коші для ЗДР.

Тема 7. 3.13.02.07 Паралельні методи чисельного розв'язання жорстких ЗДР та їх реалізація в багатопроцесорних структурах.

Змістовий модуль 2. 3.13.03 Паралельне програмування.

Тема 1. 3.13.03.01 Побудова паралельних обчислювальних систем (конвеєрні, матричні, мультипроцесорні).

Тема 2. 3.13.03.02 Побудова кластерних систем.

Тема 3. 3.13.03.03 Способи передавання даних. Типи паралелізму.

Тема 4. 3.13.03.04 Комутація і синхронізація в розподілених системах.

Тема 5. 3.13.03.05 Програмування паралельних обчислень на неоднорідних мережах комп'ютерів на мові mpC.

Тема 6. 3.13.03.06 Засоби підтримки паралельних обчислень (PVM, MPI).

Тема 7. 3.13.03.07 Комунікаційні, колективні, глобальні обчислювальні операції над розподіленими даними.

Тема 8. 3.13.03.08 Моделі віддаленого виклику процедур (RPC) та віддаленого застосування методів (RMI).

Змістовий модуль 3. 3.13.01 Grid технології.

Тема 1. 3.13.01.01 Базові складові Grid і ресурси.

Тема 2. 3.13.01.01 Зв'язок Grid і веб-технологій. Програмне Grid-забезпечення (ПГЗ).

Тема 3. 3.13.01.02 Організація і управління розподіленням ресурсів (WSRF, GRAM, CONDOR).

Тема 4. 3.13.01.03 Grid і бази даних. Управління Grid-оточенням.

Тема 5. 3.13.01.04 Безпека файлової системи. Сертифікат відкритих ключів.

Тема 6. 3.13.01.05 Система підтримки функціонування: послуга протоколювання процесу виконання завдань.

Тема 7. 3.13.01.06 Grid-портал для доступу користувачів до ресурсів і прикладних програм Grid.

Тема 8. 3.13.01.07 Grid-застосування.

4 Програма лекційного (теоретичного) модуля

У табл. 4.1 наведений приблизний розподіл часу, який, згідно з робочою програмою дисципліни, відводиться на лекційний модуль і вивчення студентами змістовних модулів, розділів програми і тем курсу під час самостійної роботи. Розподіл може змінюватись відповідно до змін, які час від часу можуть виникати у робочій програмі. Зміни, якщо вони відбуваються, повинні доводитися до відома студентів викладачем під час установчих лекцій.

Таблиця 4.1 Приблизний розподіл часу лекційного модуля і СРС

Змістовні модулі	Розділи програми (шифр і назва)	Теми (шифр і назва)	К-сть ауд. годин	К-сть годин СРС	Форми завдань на СРС	Форми поточ. контр. СРС
1	2	3	4	5	6	7
ЗМ-Л1	3.13.02 Паралельні обчислювальні методи	3.13.02.01 Організація паралельних обчислень з використанням наявних технологій (PVM, MPI)		4	Вивчення розділів теоретичного матеріалу (ВРТМ)	
		3.13.02.02 Паралельні перетворення арифметичних виразів		6	ВРТМ	

1	2	3	4	5	6	7
		3.13.02.03 Базові алгоритми паралельних обчислень		6	ВРТМ	
		3.13.02.04 Паралельні методи розв'язання СЛАР	2	6	ВРТМ	
		3.13.02.05 Паралельні методи розв'язання систем нелінійних рівнянь		6	ВРТМ	
		3.13.02.06 Ефективність паралельних обчислювальних методів під час розв'язання нелінійної задачі Коші для ЗДР		6	ВРТМ	
		3.13.02.07 Паралельні методи чисельного розв'язання жорстких ЗДР та їх реалізація в багатопроцесорних структурах	4	6	ВРТМ	
ЗМ-Л2	3.13.03 Паралельне програмування	3.13.03.01 Побудова паралельних обчислювальних систем (конвеєрні, матричні, мультипроцесорні)		3	ВРТМ	
		3.13.03.02 Побудова кластерних систем		5	ВРТМ	

1	2	3	4	5	6	7
		3.13.03.03 Способи передавання даних. Типи паралелізму		5	ВРТМ	
		3.13.03.04 Комутація і синхронізація в розподілених системах		5	ВРТМ	
		3.13.03.05 Програмування паралельних обчислень на неоднорідних мережах комп'ютерів на мові mpC		5	ВРТМ	
		3.13.03.06 Засоби підтримки паралельних обчислень (PVM, MPI)	1	5	ВРТМ	
		3.13.03.07 Комунікаційні, колективні, глобальні обчислювальні операції над розподіленими даними		5	ВРТМ	
		3.13.03.08 Моделі віддаленого виклику процедур (RPC) та віддаленого застосування методів (RMI)		5	ВРТМ	
ЗМ-Л3	3.13.01 Grid техно-	3.13.01.01 Базові складові Grid і ресурси		3	ВРТМ	

1	2	3	4	5	6	7
	логії	3.13.01.01 Зв'язок Grid і веб-технологій. Програмне Grid-забезпечення (ПГЗ)		3	ВРТМ	
		3.13.01.02 Організація і управління розподіленням ресурсів (WSRF, GRAM, CONDOR)		3	ВРТМ	
		3.13.01.03 Grid і бази даних. Управління Grid-оточенням		5	ВРТМ	
		3.13.01.04 Безпека файлової системи. Сертифікат відкритих ключів		5	ВРТМ	
		3.13.01.05 Система підтримки функціонування: послуга протоколювання процесу виконання завдань		4	ВРТМ	
		3.13.01.06 Grid-портал для доступу користувачів до ресурсів і прикладних програм Grid		4	ВРТМ	
		3.13.01.07 Grid-застосування	1	4	ВРТМ	
Індивідуальне завдання				36		
Разом			8	145		

Знання, якими повинні оволодіти студенти

- за **першим** змістовним модулем: базових алгоритмів паралельних обчислень, паралельних методів розв'язання СЛАР, нелінійних рівнянь, чисельного розв'язання ЗДР;
- за **другим** змістовним модулем: архітектури паралельних обчислювальних систем (конвеєрних, матричних, мультипроцесорних), основ мові mpC, засобів підтримки паралельних обчислень PVM та MPI, моделей RPC та RMI;
- за **третім** змістовним модулем: базових складових Grid, програмного Grid-забезпечення, Grid-застосування.

Наявне навчально-методичне забезпечення змістовних модулів

- **перший**: Посібник «Застосування засобів інтерфейсу передачі повідомлень при програмуванні розподілених систем мовою Java – Одеса, 2018. – 209 с.», «Технології розподілених систем та паралельних обчислень : Конспект лекцій (змістовний модуль №1) / Одеса, ОДЕКУ, 2018. 40 с.», демонстраційні лекційні матеріали;
- **другій**: конспект лекцій «Технології розподілених систем та паралельних обчислень (змістовний модуль №2) – Одеса: ОДЕКУ 2016. – 155с.», демонстраційні лекційні матеріали;
- **третій**: демонстраційні лекційні матеріали.

5 Програма практичного модуля

За результатами виконання практичного модуля (див. табл. 5.1) студенти повинні оволодіти наступними вміннями.

Вміння за модулями:

- **перший** – вміти налагоджувати оточення операційного середовища для роботи з класами пакету MPJ, компіляція і виконання паралельної програми мовою Java з застосуванням технологій MPI.
- **другий** – вміти виконувати обмін повідомленнями типу точка/точка без виникнення тупикових ситуацій, вміння використовувати колективні методи класів пакету MPJ.

– **третій** – вміння розробляти власний паралельний алгоритм з використанням парадигми розпаралелювання даних, вміння вибирати і застосовувати ті чи інші методи з множини методів стандарту MPI згідно з семантикою програми, що розробляється.

Таблиця 5.1 Вміст і приблизний розподіл часу практичного модуля дисципліни

Змістовні модулі	Тема лабораторної роботи	К-ть аудтор. годин	К-ть годин СРС	Форми завд. на СРС	Форми поточ. конролю СРС
1	2	3	4	5	6
ЗМ-Лр1	Знайомство з бібліотекою класів MPJ. Налагодження операційного середовища. Компіляція і запуск програми				
ЗМ-Лр2	Застосування класів і методів пересилань точка-точка бібліотеки MPI	4	2	Підготовка до роботи (ПР)	Усне опитування (УО)
	Застосування класів і методів колективної взаємодії бібліотеки MPI	4	2	ПР	УО
ЗМ-Лр3	Створення програми із застосуванням за самостійним вибором студента функцій MPI для вирішення різних паралельних задач				
Разом		8	4		

Як і у разі з лекційним модулем вміст практичного модуля і відповідний розподіл часу може дещо змінюватись згідно з робочою програмою дисципліни. Про такі зміни студенти попереджаються під час проведення установочних занять.

У якості **методичного забезпечення** всіх модулів використовується навчальний посібник «Застосування засобів інтерфейсу передачі повідомлень при програмуванні розподілених систем мовою Java – Одеса, 2018. – 209 с.». Крім того застосовується **інтерактивна документація** з методів класів пакету MPJ, яка знаходиться за електронною адресою //192.168.70.202/library/Технологія розподілених систем та паралельних обчислень/MPJ_doc/index.htm.

Всі лабораторні роботи виконуються мовою Java **в комп'ютерних класах** кафедри інформаційних технологій із застосуванням **локальної мережі робочих станцій** (імітується структура обчислювального кластера) на базі ОС Windows и сервера під керування ОС Linux.

6 Організація самостійної роботи студента заочника

Згідно вимог Положення про організацію та контроль самостійної та індивідуальної роботи студентів ОДЕКУ від 28.11.2013 р. самостійна робота студента (СРС) під час вивчення навчальної дисципліни складається із декілька видів:

1) підготовка до лекцій (ПЛЗ);
2) підготовка до лабораторної роботи (ПЛр);
3) підготовка до колоквіуму (ПКл);
4) вивчення розділів теоретичного матеріалу (ВРТМ);
5) виконання контрольної роботи (КР);
6) перевірка контрольної роботи(ПКР);
7) самостійне виконання індивідуального завдання.

Види самостійної роботи та форми її контролю, які застосовуються при роботі зі студентами заочної форми навчання наведені у табл. 6.1. Крім того в таблиці наведений приблизний розподіл часу, що відводиться на СРС. Як вже говорилось раніш цей розподіл може дещо змінюватись, крім того від в дуже

великій степені залежить від індивідуальних особливостей кожного студента, але повний обсяг часу лімітується робочими планами і програмою дисципліни.

Таблиця 6.1 – Види самостійної роботи та форми її контролю

Змістовий модуль	Розділи роботи	Завдання	К-ть годин СРС	Контролюючи заходи	Термін проведення
1	2	3	4	5	6
ЗМ-Л1 ЗМ-Лр1	Паралельні обчислювальні методи	Вивчення розділів теоретичного матеріалу (ВРТМ)	40		
ЗМ-Л2 ЗМ-Лр2	Паралельне програмування	ВРТМ, ПЛр	42	УО	
ЗМ-Л3 ЗМ-Лр3	Grid технології	ВРТМ	31		
Індивідуальне завдання		Міжсесійна КР	36	ПКР	М/сес.
		Підготовка до іспиту	30	іспит	Сесія
			179		

7 Індивідуальні завдання

Для заочної форми навчання робочою навчальною програмою дисципліни передбачається виконання міжсесійної контрольної роботи.

Тематика міжсесійної контрольної роботи присвячена питанням створення паралельних програм з застосуванням технології і функцій стандарту інтерфейсу пересилання повідомлень MPI і передбачає обробку великого обсягу даних шляхом розпаралелювання даних. У якості бібліотеки функцій MPI використовується пакет класів MPJ, у якому ці функції реалізуються методами

відповідних класів мовою Java.

Нижче наведені деякі приклади індивідуальних завдань.

1. Використовуючи формулу трапецій та блочно-циклічну парадигму розпаралелювання даних за допомогою 7-і паралельних процесів обчислити інтеграл $\int_0^1 \frac{1}{1+x^2} dx$. Для підвищення точності обчислення, інтервал інтегрування розбити на два мільйона частин.
2. Використовуючи 15 паралельних процесів по технології master-slave, знайти добуток матриці розмірністю 10000×10000 дійсних чисел на вектор розмірністю 10000 також дійсних чисел. Розподілення матриці по процесах виконати блочно-циклічно.

Для виконання контрольної роботи №1 відводиться 36 годин за рахунок загального часу відведеного на самостійну роботу (по 12 годин на кожну тему).

8 Методи навчання

Навчання ведеться за допомогою читання лекцій, виконання контрольної роботи, проведення лабораторних робіт та за рахунок самостійної роботи студентів.

9 Організація поточного, семестрового та підсумкового контролю знань

9.1 Методи контролю

Поточний контроль знань та вмінь студентів виконується шляхом усного опитування на лабораторних роботах, оцінювання якості і терміну виконання індивідуального контрольного завдання. Індивідуальне контрольне завдання **повинно бути виконане і обов'язково зараховане не менш ніж за тиждень до початку екзаменаційної сесії**. Студенту видається план-графік виконання контрольної роботи, якого він повинен суворо дотримуватись. Виконані етапи контрольної роботи надсилаються студентом на електронну почтову адресу кафедри інформаційних технологій. Затримки зі здачею етапів і всього виконаного завдання, тобто порушення план-графіку істотно зніжує загальну оцінку за цей контрольний захід. Коли контрольне завдання здається безпосередньо перед проведенням іспитової контрольної роботи, оцінка за його виконання **не**

може перевищувати 60 відсотків, від максимально можливої, навіть при відмінному 100 відсотковому виконанні роботи.

Наприкінці залікової сесії проводиться тестова іспитова контрольна робота. Перевірка правильності виконання тестових завдань **виконується за допомогою спеціального програмного забезпечення на комп'ютерах** кафедри інформаційних технологій. Нижче наведені деякі приклади запитань з іспитової контрольної роботи:

1) У мережах міжз'єднань пропускна здатність часто обмежується …
2) Твердження, що, при послідовній частині програми відмінної від 0, ідеальне підвищення швидкості роботи суперкомп'ютера неможливо…
3) Маршрутизація, при якій для всіх пакетів, що направляються до тому самому кінцевого пункту, вибір маршруту однаковий, називається …
4) Коефіцієнт розгалуження мережі міжз'єднань визначає …
5) Чи може інтерфейс введення/виведення процесора містити свій власний під процесор?
6) Маршрутизація, при якій джерело визначає весь шлях пакета по мережі заздалегідь і виражає його списком з номерів портів, називається…
7) Системи з невеликими по розмірі компонентами, взаємодіючі по схемах з високою пропускною здатністю, називаються системами …
8) Програміст має досить велику свободу дій і гнучкість у роботі при …
9) Скорість обміну даними між процесами є важливою…
10) При здійсненні паралелізму даних …
11) Циклічний розподіл ітерацій передбачає …
12) У поточний час для розпаралелювання існує …
13) Блочно-циклічний розподіл ітерацій це – …
14) За методом координат простір ітерацій розбивається на …
15) Чи зобов'язана віртуальна топологія повторювати фізичну топологію цільового комп'ютера?

16) Чи може MPI-програма (стандарт 1.1) продовжувати роботу після аварійного завершення одного з процесів?
17) Метод Finalize слугує для …
18) Метод Init в програмі повинний виконуватись …
19) Як приймаючий процес може визначити довжину отриманого повідомлення?
20) Спеціальний метод Get_processor_name надає можливість …

9.2 Методика оцінювання студентів

Поточна та підсумкова оцінка рівня знань студентів здійснюється за модульною системою. Розподіл балів за змістовими модулями з теоретичної і практичної частини курсу, яку може набрати студент, наведений у табл. 9.1.

При проведенні міжсесійного контролю студент вважається атестованим, якщо він набрав не менше 50% від максимально можливої суми балів за модулями, що завершені у атестаційний період.

Таблиця 9.1 – Розподіл балів, які отримують студенти

Поточне тестування та самостійна робота студентів заочної форми навчання		Підсумковий тест (іспит)	Сума
Індивідуальне завдання	Лабораторні роботи		
100	100	100	300

Студент вважається таким, що **не виконав навчального плану** з дисципліни «Технології розподілених систем та паралельних обчислень» якщо:

- не виконано хоча б один практичний модуль (практичний модуль вважається виконаним, якщо отримана за нього кількість балів становить більше 50% від максимально можливої);
- набрано менше 50% балів від суми балів практичних модулів у навчальному році;
- набрано менше 50% балів від суми балів теоретичних модулів у навчальному році,

до тих пір доки не ліквідує ці заборгованості

Для студентів, які виконали навчальний план, формується інтегральна сума балів – сума балів одержаних з теоретичної та практичної частин курсу.

Студенти виконують **тестову іспитову контрольну роботу** з курсу.

Екзаменаційний білет містить 20 тестових запитань, які мають різні ваги приблизно в такому співвідношенні: 12 запитань з вагою 5; 6 запитань з вагою 4 і 2 запитання з вагою 8. Тобто вірні відповіді на всі запитання з першої групи забезпечують студентові 60% від повної суми балів, вірні відповіді на другу групу додають ще 24% і останні 2 питання доповнюють відповідь до 100 відсотків.

Іспит проводиться на комп'ютерному обладнанні кафедри ІТ. Час відповіді обмежений пів годиною.

Інтегральна оцінка з дисципліни виставляється згідно «Положення про проведення підсумкового контролю знань студентів в ОДЕКУ» та «Положення про організацію поточного та підсумкового контролю знань студентів заочної форми навчання в ОДЕКУ», які затверджені Методичною радою університету.

Для студентів заочної форми навчання загальна кількісна **інтегральна оцінка** з дисципліни згідно до «Положення про організацію поточного та підсумкового контролю знань студентів заочної форми навчання ОДЕКУ» від 1.03.2011р., накопичена підсумкова оцінка засвоєння студентом навчальної дисципліни розраховується за формулою:

$$ПО = 0,5ОПК + 0,25(ОЗЕ + ОМ),$$

де:

ОПК – кількісна оцінка (у відсотках від максимально можливої) заходу підсумкового контролю;

ОЗЕ – кількісна оцінка (у відсотках від максимально можливої) заходів контролю СРС під час проведення аудиторних занять;

ОМ – кількісна оцінка (у відсотках від максимально можливої) заходів контролю СРС у міжсесійний період.

Наприкінці навчального семестру студент отримує інтегральну оцінку з дисципліни за системою оцінювання, що використовуються в університеті, з обов'язковим переведенням оцінок до шкали ЄКТАС. Інтегральна оцінка виставляється викладачем у відомість та у залікову книжку студента у строки, які визначені розкладом екзаменаційної сесії. Дострокове виставлення оцінки можливе лише за письмовим дозволом декана факультету (начальника навчально-консультаційного центру).

При підсумковій атестації використовується шкала відповідності сумарної суми балів (%) підсумкової атестації з дисципліни у формі іспиту (табл. 9.2).

Таблиця 9.2 – Шкала оцінювання: національна та ECTS

Сума балів за всі види навчальної діяльності	Оцінка ECTS	Оцінка за національною шкалою
		для екзамену, курсового проекту (роботи), практики
1	2	3
90 – 100	**A**	відмінно
82 – 89,9	**B**	добре
74 – 81,9	**C**	
64 – 73,9	**D**	задовільно
60 – 63,9	**E**	
35 – 59,9	**FX**	незадовільно з можливістю повторного складання
0 – 34,9	**F**	незадовільно з обов'язковим повторним вивченням дисципліни

10 Методичне забезпечення

1. Рольщиков В.Б. Застосування засобів інтерфейсу передачі повідомлень при програмуванні розподілених систем мовою Java: Навчальний посібник / Одеса, 2018. 209 с.

11 Рекомендована література

11.1 Базова

1. Рольщиков В.Б. Технології розподілених систем та паралельних обчислень / Конспект лекцій (Змістовний модуль №2) – Одеса: ОДЕКУ 2016. – 155с.

2. Рольщиков В.Б. Технології розподілених систем та паралельних обчислень / Конспект лекцій (Змістовний модуль №1) – Одеса: ОДЕКУ 2018. – 186с.

3. Рольщиков В.Б Застосування засобів інтерфейсу передачі повідомлень при програмуванні розподілених систем мовою Java: навчальний посібник / Одеса, 2018. 209 с.

4. Таненбаум Э., ван Стеен М. Распределенные системы. Принципы и парадигмы. – СПб.: Питер, 2003. – 877с.

11.2 Допоміжна

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления – СПб.: БХВ-Петербург, 2002. – 608с.

2. Г.Р. Эндрюс Основы многопоточного, параллельного и распределенного программирования – М.: Вильямс, 2003. – 506с.

3. Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI – Минск: БГУ, 2002. – 323с.

4. Оленев Н.Н. Основы параллельного программирования в системе MPI – М: ВЦ РАН, 2005. 81с.

12 Інформаційні Інтернет ресурси

1. https://parallel.ru/
2. http://www.eu-datagrid.org/
3. http://cluster.linux-ekb.info/
4. http://parallelcompute.sourceforge.net/index_ru.php
5. http://ru.wn.com/Параллельные_вычисления
6. http://www.myshared.ru/search/?q=%D0%93%D0%B5%D1%80%D0%B3%D0%B5%D0%BB%D1%8C
7. http://it.kgsu.ru/ParalAlg/
8. http://bigor.bmstu.ru/?cnt/?doc=Parallel/base.cou

Загальні методичні вказівки до виконання контрольної роботи

Вибір варіантів розділів контрольної роботи виконується за формулою:

(трьохзначне число) mod 12+1

трьохзначне число у формулі задається наступним чином:

– для першої частини роботи – це останні три цифри залікової книжки;

– для другої частини – нове число створюється циклічним зсувом цифр у вихідному числі на один десятковий розряд вліво;

– для третьої частини – повторюються дії з попереднього пункту.

Наприклад, номер залікової книжки 12285, тоді номер варіанту для першого завдання буде 285 mod 12+1=10, відповідно для другого завдання вийде 852 mod 12+1=1.

Якщо у варіанті завдання до контрольної роботи потребується написання програмного коду, то відповідні програми пишуться мовою програмування Java з застосуванням методів класів бібліотеки MPJExpress. Правильність написання програм перевіряється шляхом їхнього запуску на виконання під керуванням віртуальної машини Java в любому з операційних середовищ Linux, Unix, MacOS або, на сам кінець, Windows.

Нижче до кожного розділу контрольної роботи надані деякі теоретичні відомості які можуть бути корисними при виконанні завданій.

1 Перша частина

Література до виконання завдання: [2] – стор. 44 – 60. Крім того, нижче наведені деякі відомості з принципів побудови ярусно паралельної форми (ЯПФ) програми.

Метою Ознайомлення з поняттям графа програми (алгоритму), операційною і інформаційною залежністю, графом керування, інформаційним графом, операційною і інформаційною історіями, ЯПФ програми і її характеристиками.

Результатом виконання завдання повинен бути граф ярусно паралельної

форми інформаційної історії деякого вихідного графа згідно з варіантом завдання.

2 Друга частина

Література до виконання завдання: [2] стор. 60 – 80, [3]. Крім того, нижче наведені деякі відомості з теоретичного матеріалу потрібного для виконання завдання і наданий приклад програми, яка застосовує циклічний розподіл ітерацій і модифікованим методом прямокутників обчислює визначений інтеграл $\int_0^{\infty} \frac{4}{1 + x^2}\, dx$.

Метою цієї частини є вивчення методів передачі повідомлень типу точка-точка з бібліотеки MPI і механізмів розпаралелювання даних між процесами (потоками) для скорочення часу необхідного для обробки даних.

Відповідна до варіанту завдання паралельна програма повинна розроблятись мовою програмування Java з використанням технології MPI за допомогою методів класів бібліотеки MPJ Express в середовищі будь-якої операційної системи (Linux, Windows тощо).

Настроювання оперативної системи для роботи з бібліотекою класів MPJ Express, компіляція і запуск програми на виконання виконується згідно з [3] в мультиядерному режимі, але, якщо є така можливість, бажаним є використання мультипроцесорного режиму.

Використання віконного інтерфейсу в програмі є необов’язковим, застосування термінального режиму є бажанішим тому, що робить більш наглядним саме паралельний алгоритм роботи програми.

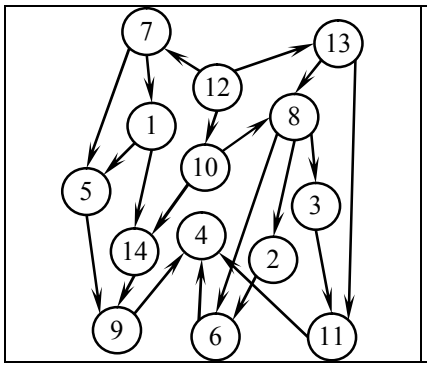
Програма повинна обробляти вкидання винятків, які можуть виникнути під час її роботи.

Завдання до контрольної роботи

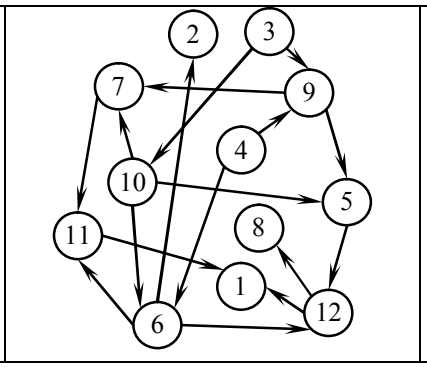
1 Завдання перше

Відповідно до свого варіанту перетворити у канонічну ярусно-паралельну форму вихідний граф деякої умовної програми, яка представлена у вигляді орієнтованого графа. Нарисувати одержану ярусно-паралельну форму

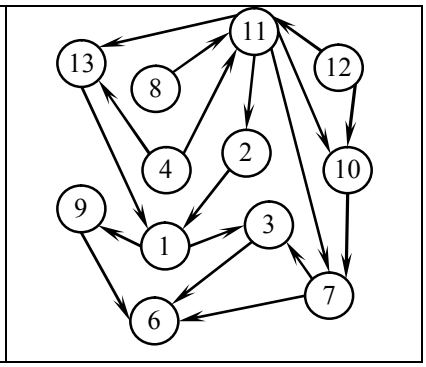
програми, визначити висоту й ширину одержаного графа, визначити і нанести на ньому критичний шлях.



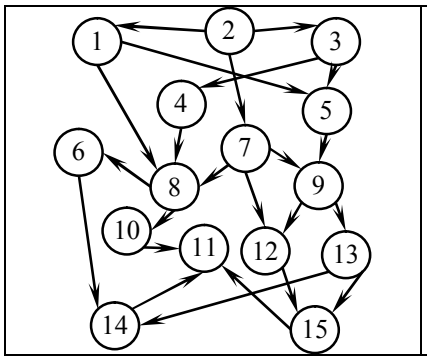
Варіант 1



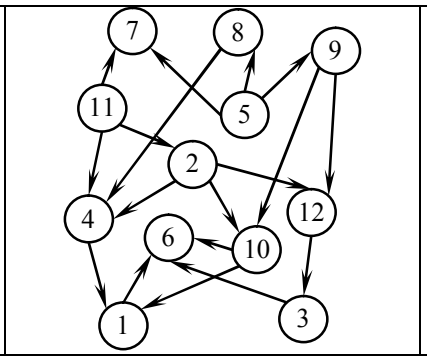
Варіант 2



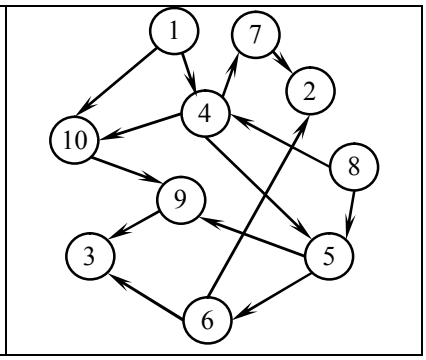
Варіант 3



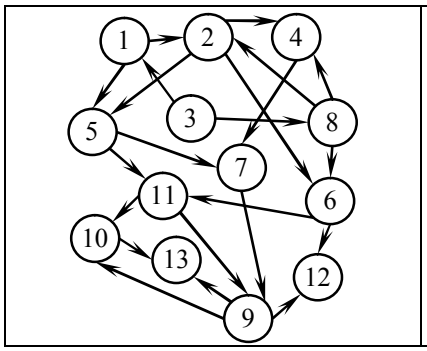
Варіант 4



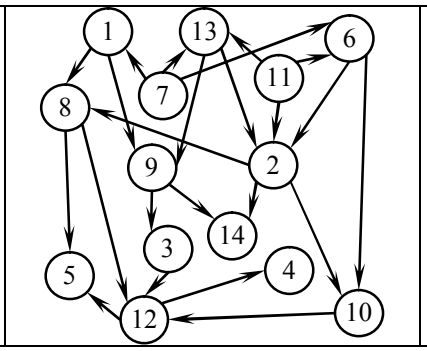
Варіант 5



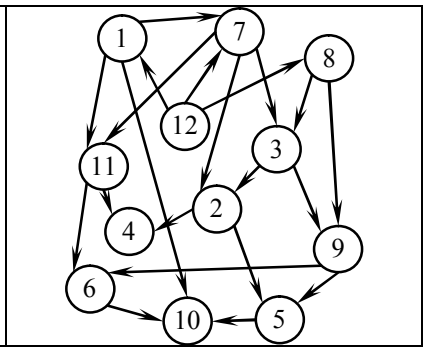
Варіант 6



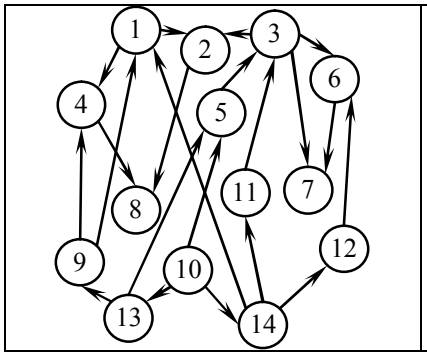
Варіант 7



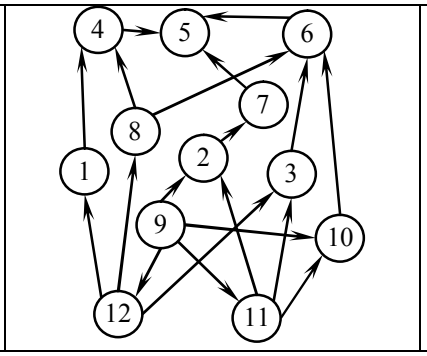
Варіант 8



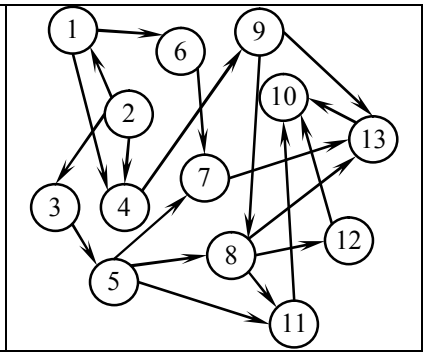
Варіант 9



Варіант 10



Варіант 11



Варіант 12

2 Завдання друге

При виконанні варіантів цього завдання, межі і крок інтегрування передати у програму як параметри командного рядка системи. Роботу програми організувати в паралельному режимі по моделі «master-slave», тобто один процес розподіляє всю роботу проміж іншими і отримує від них результати.

Інтегрування у варіантах необхідно виконувати за такими формулами:

- для методу прямокутників – $I = h\sum_{i=0}^{N} f\left(x_i + \frac{h}{2}\right),\ N = \frac{b-a}{h},\ x_i = a + hi$;
- для методу трапецій – $I = \frac{h}{2}\sum_{i=0}^{N} f(x_i) + f(x_{i+1}),\ N = \frac{b-a}{h},\ x_i = a + hi$;
- для методу Сімпсона – $I = \frac{h}{3}\left(f(a) + f(b) + \sum_{i=1}^{N-1} 2((i \bmod 2) + 2) f(x_i) \right),$ $N = \frac{b-a}{h},\ x_i = a + hi.$

В цих формулах a і b – відповідно нижня та верхня межі інтегрування.

1. При кількості процесів (потоків) P = 7, застосовуючи блочно-циклічний метод розпаралелювання даних, методом Сімпсона з кроком h = 0.01 обчислити $\int_{0}^{3150} x^2 \sin x\, dx$. Кількість даних, яка передається у блоці за один раз, не може перевищувати 3000.
2. Застосовуючи 12 процесів (потоків) і блочний метод розпаралелювання циклів, методом трапецій з кроком h = 0.01 обчислити визначений інтеграл $\int_{0.01}^{1000} \frac{\sin x \cos x}{x} dx$.
3. За допомогою методу циклічного розпаралелювання даних методом прямокутників обчислити визначений інтеграл від функції $\int_{1}^{1000} \ln^2 x\, dx$. Крок інтегрування покласти рівним 0.01, кількість процесів (потоків) P = 8.

4. Методом Сімпсона з кроком h = 0.001 обчислити визначений інтеграл $\int_{0.001}^{350} \frac{\sin x}{x^2} dx$. Розпаралелювання даних виконати за допомогою циклічного розподілу циклів. Кількість процесів (потоків) задати рівною 10.
5. Застосовуючи блочний метод розпаралелювання циклів і метод трапецій з кроком h = 0.1 обчислити визначений інтеграл $\int_{0}^{10000} \frac{\cos x}{1+x^2} dx$. Кількість паралельних процесів (потоків) P = 8.
6. За допомогою 11 паралельних процесів (потоків), методом прямокутників з кроком h = 0.01 обчислити визначений інтеграл $\int_{1}^{2500} \frac{\ln x}{x^2} dx$. Розпаралелювання даних виконати блочно-циклічним методом з максимальною кількістю даних в блоці 700.
7. У паралельному режимі за допомогою 10 процесів (потоків) обчислити визначений інтеграл $\int_{0}^{315} x^3 \cos x \, dx$ з кроком h = 0.001. Обчислення виконати методом Сімпсона. Дані по процесах розподілити циклічним способом.
8. Методом трапецій з кроком h = 1e-6 обчислити визначений інтеграл $\int_{0}^{5} \frac{x}{e^x + 1} dx$. При обчисленні використати 12 процесів (потоків). Дані по процесах розподілити блочним методом.
9. Застосовуючи блочно-циклічне розподілення даних з кількістю даних в блоці не більше за 1700, методом прямокутників з кроком h = 1e-4 обчислити визначений інтеграл $\int_{1}^{100} x^2 \ln x \, dx$. Кількість процесів (потоків) P = 8.
10. Використовуючи 9 процесів (потоків) методом Сімпсона з кроком h = 1e-5 обчислити визначений інтеграл $\int_{1e-5}^{62} \frac{\cos x}{x^3} dx$. При розподіленні даних по процесах застосувати блочне розподілення даних.

11. Використовуючи 11 процесів (потоків) і блочно-циклічне розподілення даних по процесах з кількістю даних у блоці не більше за 200, методом трапецій з кроком h = 1e-6 обчислити визначений інтеграл $\int_{1e-6}^{10} \frac{x}{e^x - 1} dx$.

12. Методом прямокутників обчислити визначений інтеграл $\int_{1}^{1000} \frac{\ln^2 x}{x} dx$. Обчислення виконати застосовуючи циклічний розподіл даних у системі з 10 процесами (потоками). Крок інтегрування h взяти рівним 0.001.

Основні теоретичні відомості і деякі приклади застосування методів бібліотеки MPI, які необхідні при виконанні контрольної роботи

1 Короткі теоретичні відомості до першої частини

Представлення алгоритму програми у вигляді набору ансамблів операцій, причому в кожному ансамблі всі операції не зв'язані одна з одною, не є остаточним етапом по створенню паралельної програми. Необхідно ще вирішити питання розподілу цих ансамблів між процесорами або іншими обробними пристроями. Якщо архітектура паралельної системи дозволяє реалізовувати одночасно всі операції кожного ансамблю, то без урахування часу на передачі даних час виконання алгоритму буде майже прямо пропорційним числу ансамблів. Число ансамблів при такому представленні називається ***висотою алгоритму***. А алгоритми, у яких висота менше загального числа операцій називаються ***паралельними***. Представлення алгоритму через виконання послідовності ансамблів з незалежних операцій і визначає ***паралельну форму алгоритму***.

Найбільш очевидним способом розподілу завдань по процесорах є наступний. В інформаційній історії позначаються ті операції, які залежать тільки від зовнішніх даних програми й говорять, що такі операції належать до першого ярусу інформаційної історії. На другий ярус поміщаються операції, які залежать тільки від операцій першого ярусу й зовнішніх даних. Третій ярус містить операції інформаційно залежні від результатів виконання операцій другого й першого ярусів і вихідних даних і так далі. При розподілених у такий спосіб операціях інформаційної історії, утворюється так звана ***ярусно-паралельна фо-***

рма (***ЯПФ***) програми. На рис. 1.1 наведений приклад такої структури.

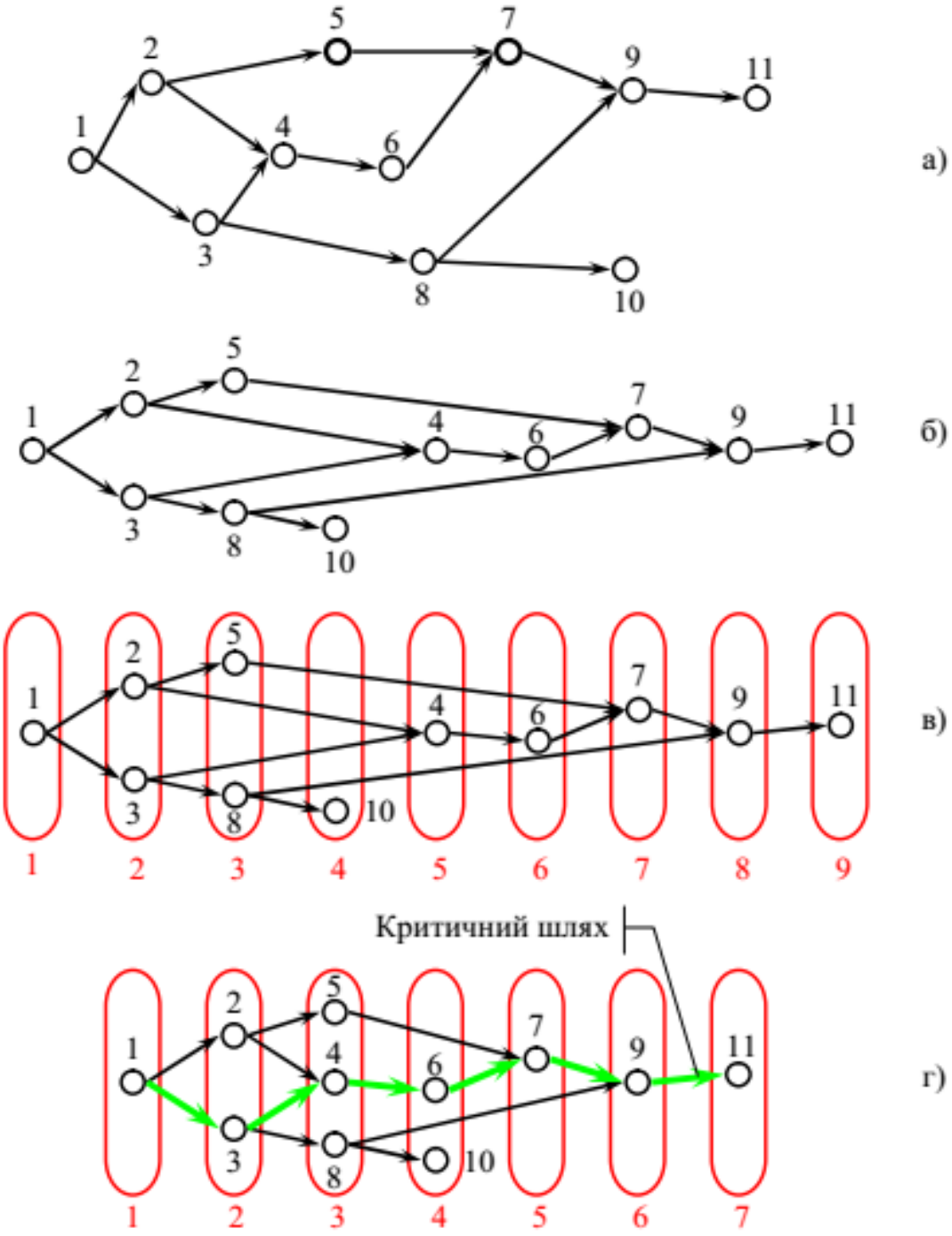


Рисунок 1.1 − Приклад переходу від графа програми до її канонічної ярусно-паралельної формі

а) вихідний граф програми, б) еквівалентно перебудований граф,

в) яруси ярусно-паралельної форми програми,

г) канонічна ярусно-паралельна форма програми

Аналіз вихідного графа (рис. 1.1 а) трохи скрутний через нерегулярне розташування його вершин. Незначні перестановки вершин графа зі збереженням його топології приводять до більш зручної форми (рис. 1.1 б). На цьому графі вже можна виділити 9 ярусів ЯПФ (рис. 1.1 в). Таким чином, видно, що вихідний граф алгоритму зводиться до паралельної форми з висотою яка дорівнює числу ярусів – 9 при загальному числі операцій одинадцять. При уважному розгляді отриманої ЯПФ можна помітити, що вершина 4, знаходячись на п’ятому ярусі, інформаційно залежить лише від вершин 2 і 3, які належать другому ярусу, тому вона може бути перенесена з п'ятого ярусу на третій. Наступним кроком буде перенесення вершини 6 із шостого ярусу на четвертий, тому що вона інформаційно залежить від операцій на третьому ярусі (вершина 4). І, на завершення, вершини 7, 9, 11 можуть бути перенесені на яруси п'ять, шість і сім відповідно. Тим самим висота отриманої ярусно-паралельної форми програми буде дорівнювати семи.

У будь-якому орієнтованому ациклічному графі завжди можна виділити так званий ***критичний шлях*** – шлях максимальної довжини від початкової вершини до кінцевої. У розглянутому прикладі такий шлях у може, наприклад, пролягати через вершини $1\rightarrow3\rightarrow4\rightarrow6\rightarrow7\rightarrow9\rightarrow11$.

Довжина критичного шляху визначає *мінімальну висоту* із всіх можливих висот ярусно-паралельної форми даного ациклічного графа. ЯПФ, у якій кожна операція з ярусу з номером `k` більшим за одиницю, як один з аргументів одержує результат виконання деякої операції з попереднього (`k - 1`)-го ярусу і має мінімальну висоту, називається ***канонічною ярусно-паралельною формою*** програми. Для будь-якого алгоритму при заданих вхідних даних канонічна форма існує завжди і є єдиною. Висота канонічної ЯПФ на одиницю більше довжини критичного шляху. Таким чином, на рис. 1.1 г) представлена канонічна ЯПФ із висотою, яка дорівнює семи. Відповідно, довжина критичного шляху цієї ЯПФ дорівнює шести.

Кількість операцій на одному ярусі обумовлює ***ширину ярусу***, а максималь-

на ширина ярусів у канонічної ЯПФ називається ***шириною ЯПФ***. Для наведеного прикладу ширина ЯПФ дорівнює трьом – це ширина третього ярусу (рис. 1.1 г). У канонічній паралельній формі, як і в будь-якій іншій формі мінімальної висоти, яруси в середньому мають максимально можливу ширину.

Побудова ярусів канонічної ЯПФ виконувалась таким чином, щоб на один ярус потрапляли операції, які не перебувають в інформаційної залежності відносно одна до одної, а тому вони можуть бути виконані одночасно.

Однак подібне побудування канонічної ЯПФ програми можна зробити лише для дуже простих графів з достатньо структурованим розташуванням вершин і дуг. Але такі випадки зустрічаються дуже і дуже рідко, тому було б бажаним мати інший, більш строгий метод створення ЯПФ програми. Такий метод існує і заснований він на наступному.

З курсу дискретної математики ви вже добро знаєте, що будь який граф може бути описаний за допомогою так званої ***матриці суміжності*** **A** у який кількість стовпців і рядків дорівнює кількості вершин графа, а значення елементів a_{ij} встановлюються у одиницю, якщо відповідні вершини зв'язані між собою, і дорівнюють нулю, якщо між вершинами дуги немає.

У свою чергу матриця суміжності дає можливість шляхом її перетворення безпосередньо одержати канонічну ярусно-паралельну форму графа програми. Алгоритм одержання канонічної ЯПФ програми вимагає виконання наступних кроків:

1) складається матриця суміжності графа;
2) в одержаній матриці суміжності виконується пошук нульових стовпців;
3) вершини, яким відповідають нульові стовпці, поміщаються в поточний ярус ЯПФ;
4) з матриці суміжності викреслюються стовпці й рядки, які відповідають вершинам поточного ярусу;
5) пункти 2, 3 і 4 даного алгоритму повторюються доти, поки не будуть охоплені всі вершини;

6) по вихідній матриці суміжності відновлюються дуги між вершинами.

Так, наприклад, для графа, який зображений на рис. 1.1 а) матриця суміжності має вигляд:

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	1	0	0	0	0	0	0	0	0
2	0	0	0	1	1	0	0	0	0	0	0
3	0	0	0	1	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	1	0
9	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0

.

З матриці цілком очевидно, що на першому ярусі ЯПФ повинна розташовуватися вершина за номером 1 оскільки стовпчик, який відповідає цій вершині має лише нульові значення.

Після викреслювання першого стовпця і першого рядка перетворена матриця має вигляд:

	2	3	4	5	6	7	8	9	10	11
2	0	0	1	1	0	0	0	0	0	0
3	0	0	1	0	0	0	1	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	1	1	0
9	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0

.

В цій матриці вже потрібно викреслити стовпці і рядки з номерами 2 і 3, а відповідні вершини занести до другого ярусу ЯПФ. Знову перетворена матриця приймає вже такий вигляд:

	4	5	6	7	8	9	10	11
4	0	0	1	0	0	0	0	0
5	0	0	0	1	0	0	0	0
6	0	0	0	1	0	0	0	0
7	0	0	0	0	0	1	0	0
8	0	0	0	0	0	1	1	0
9	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0

.

З неї вже потрібно викреслити 4, 5 і 8 стопці й рядки. Це говорить о тім, що до третього ярусу ЯПФ належать три відповідні вершини. Зі знов одержаної матриці

	6	7	9	10	11
6	0	1	0	0	0
7	0	0	1	0	0
9	0	0	0	0	1
10	0	0	0	0	0
11	0	0	0	0	0

,

з викреслюванням двох стовпців і двох рядків, відповідні їм шоста і десята вершини заносяться до четвертого ярусу. До останніх ярусів ЯПФ: п’ятого, шостого і сьомого, згідно з наступними перетвореними матрицями

	7	9	11
7	0	1	0
9	0	0	1
11	0	0	0

,

	9	11
9	0	1
11	0	0

і

	11
11	0

належать сьома, дев’ята і одинадцята вершини графа відповідно.

Як видно, в результаті послідовного виконання кроків, які пропонуються алгоритмом, одержана канонічна ЯПФ, зображена на рис. 1.1 г).

Саме процес одержання паралельної програми виконується так: спочатку між процесорами розподіляються операції першого ярусу, після їхнього завер-

шення – операції другого ярусу, потім третього й так далі. Оскільки будь-яка операція n-го ярусу залежить хоча б від однієї операції попереднього (n - 1)-го ярусу, то її не можна почати виконувати раніше, ніж завершиться виконання операцій цього ярусу. Тому канонічна ярусно-паралельна форма, в певному розумінні, визначає максимально паралельну реалізацію програми. При цьому довжина критичного шляху характеризує кількість паралельних кроків, необхідних для її виконання.

2 Короткі теоретичні відомості до другої частини роботи

Як показує практика, найбільший ресурс паралелізму в програмах зосереджений у циклах. Тому найпоширенішим способом розпаралелювання є той або інший спосіб ***розподілу ітерацій циклів***. Якщо між ітераціями деякого циклу немає інформаційних залежностей, то їх можна тим або іншим способом роздати різним процесорам для одночасного виконання. Умовно це може бути виражено приблизно наступною конструкцією псевдокоду:

```
for (i=0; i < k; i++)
  if (i ~ MyProc) {/* операції i-ої ітерації для виконання
                      процесором MyProc */}
```

Тут конструкція `i ~ MyProc` застосована для того, щоб указати, що номер ітерації `i` якимсь чином співвідноситься з номером процесора `MyProc`. Конкретний спосіб завдання цього співвідношення обумовлює те, які ітерації циклу на які процесори будуть розподілені. У принципі, ніщо не заважає, наприклад, роздати всім процесорам по одній ітерації, а всі інші ітерації виконати якимсь одним процесором. Однак очевидно, що у дуже великій кількості випадків такий розподіл виявляється неефективним, оскільки всі процесори, крім одного, виконавши свою ітерацію, будуть скоріш за все простоювати. Таким чином, однією з вимог до розподілу ітерацій (як, втім, і до процесу розпаралелювання взагалі) є по можливості *рівномірне завантаження процесорів*. Найпоширеніші способи розподілу ітерацій циклів у тому чи іншому ступені задовольняють цій вимозі. Серед цих способів розподілу виділяють:

− блокове розподілення;

− блочно-циклічне розподілення;

− циклічне розподілення.

Блокове розподілення ітерацій припускає, що розподіл ітерацій циклу по процесорах ведеться блоками по кілька послідовних ітерацій. У найпростішому випадку для кількості ітерацій у циклі, яка дорівнює `n` і кількості процесорів у системі `p`, функція ***стеля*** обумовлює кількість ітерацій у блоці `k = [n / p]`, тобто кількість ітерацій ділиться на число процесорів і результат округляється до найближчого цілого зверху. Програмно це можна визначити так:

```
n % p ? n / p + 1 : n / p.
```

При цьому майже всі процесори одержують однакову кількість ітерацій, однак один або кілька останніх процесорів можуть простоювати.

Вибір меншого розміру блоку може зменшити цей дисбаланс, однак при цьому частина ітерацій залишиться нерозподіленою. Якщо нерозподілені ітерації, після завершення виконання першого блоку ітерацій, знову почати розподіляти такими ж блоками, починаючи з першого процесора, то утворюється так званий ***блочно-циклічний розподіл*** ітерацій.

Якщо й надалі зменшувати кількість ітерацій, то дійдемо до розподілу по одній ітерації. Такий розподіл ітерацій має назву ***циклічний розподіл ітерацій***. Циклічний розподіл дозволяє мінімізувати дисбаланс у завантаженні процесорів, який найчастіше виникає при блокових розподілах.

Наприклад, блоковий розподіл ітерацій для звичайного циклу підсумовування

```
for (i = 0; i < n; i++) a[i] = a[i] + b[i];,
```

у випадку якщо в цільовому комп'ютері є `p` процесорів з номерами від 0 до `p-1` для процесора з номером `MyProc` можна записати в такий спосіб (фрагмент 1.7):

Фрагмент коду 2.1

```
k = (n - 1) / p + 1;                // обчислення розміру блоку
                                    // ітерацій
ibeg = MyProc * k;                  // індекс початку блоку для
                                    // процесора MyProc
iend = (MyProc + 1) * k - 1;        // індекс кінця блоку для
                                    // процесора MyProc
if(ibeg >= n) iend = ibeg - 1;      // якщо процесору не
                                    // дісталося ітерацій
else if(iend >= n) iend = n - 1;    // якщо дісталося менше
                                    // ітерацій
// нарешті цикл для процесора з номером MyProc
for(i = ibeg; i <= iend; i++) a[i] = a[i] + b[i];
```

Циклічний же розподіл ітерацій для того ж вихідного циклу можна записати так:

```
for (i = MyProc; i < n; i += p) a[i] = a[i] + b[i];
```

На рис. 2.1 зображені графи алгоритмів для обох розглянутих способів розподілу ітерацій циклів для паралельного виконання одинадцяти ітерацій на системі, що складається з трьох процесорів.

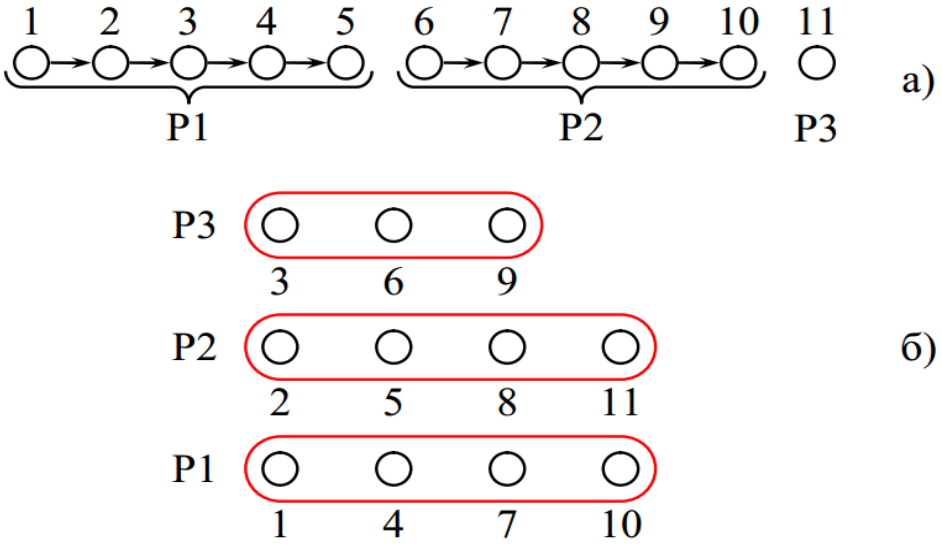


Рисунок 2.1 − Графи алгоритмів розподілу ітерацій циклів
а) операційна історія для блокового розподілу,
б) ЯПФ для циклічного розподілу ітерацій

Рис. (2.1 а) представляє операційну історію блокового розподілу ітерацій циклу. Для зазначених параметрів – кількості ітерацій циклу `n = 11` і кількості процесорів `p = 3` при блоковому розподілі ітерацій, функція стеля дає значення кількості ітерацій на процесор `k`, яке дорівнює п’яти. Тому першому й другому процесорам виділяється 5 ітерацій, а на долю третього залишається одна ітерація, тобто спостерігається явний дисбаланс завантаження процесорів. ЯПФ для циклічного розподілу ітерацій (рис. 2.1 б), коли на кожному ярусі, кожен з процесорів виконує тільки одну ітерацію, наочно демонструє більш рівномірний розподіл ітерацій (ширина перших трьох ярусів дорівнює трьом, а останнього двом).

Слід відмітити, що всі міркування про розподіл ітерацій циклів з метою досягнення рівномірності завантаження процесорів мають рацію тільки в припущенні, що розподіляються ітерації, які приблизно рівноцінні за часом виконання. Але у багатьох реальних випадках (наприклад, при розв'язанні матричних задач із трикутною матрицею) таке припущення може виявитись практично не вірним, а це означає, що можуть знадобитися цілком інші способи розподілу циклів.

Однак, тільки рівномірного завантаження процесорів у більшості практичних випадків ще недостатньо для одержання ефективної паралельної програми. Тільки у вкрай рідких випадках програма не містить інформаційних залежностей взагалі. Якщо ж є інформаційна залежність між операціями, які при обраній схемі розподілу потрапляють на різні процесори, то в цьому разі буде потрібним виконувати пересилання даних між процесорами. На практиці пересилання даних вимагають досить великого часу для свого здійснення. Тому іншим важливим завданням при розпаралелюванні коду є мінімізація необхідної *кількості* й *обсягу* пересилань даних. Так, наприклад, при наявності інформаційних залежностей між `i`-ою і (`i + 1`)-ою ітераціями деякого циклу, блоковий розподіл ітерацій може виявитися ефективнішим за циклічний, тому що при блоковому розподілі сусідні ітерації попадають на один процесор, а відповідно, буде потрібна менша кількість пересилань, ніж при циклічному розподілі. Це

твердження для коду з наявністю вказаних інформаційних зв'язків

```
for(i=MyProc; i<n; i += p) a[i] = a[i-1] + b[i];
```

наочно ілюструється рис. 2.2.

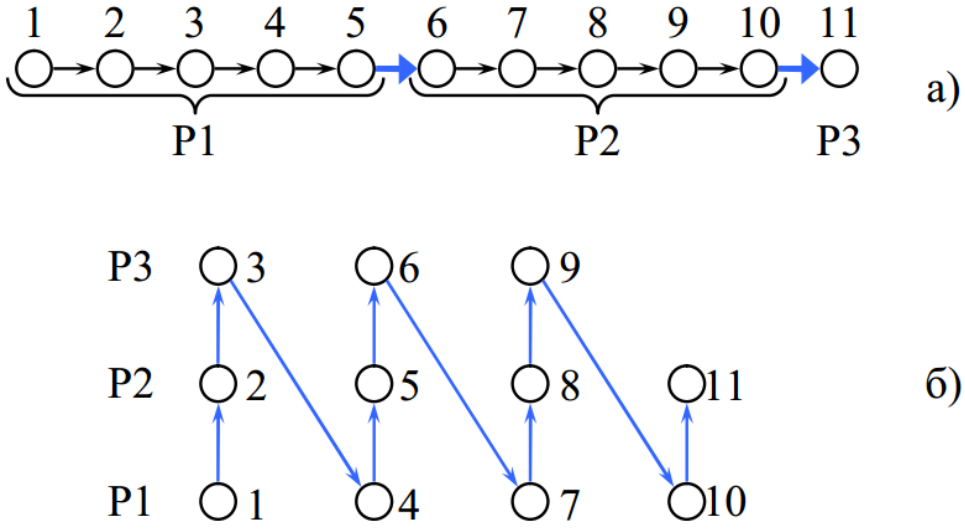


Рисунок 2.2 − Графи алгоритмів розподілу ітерацій циклів
а) інформаційна історія для блокового розподілу,
б) ЯПФ для циклічного розподілу ітерацій

На рис. 2.2 а) видно, що пересилання даних між процесорами у випадку блокового розподілу й наявності інформаційних зв'язків у циклі виконується всього лише два рази (грубі стрілки). У той же самий час у випадку циклічного розподілу ітерацій (рис. 2.2 б) присутні неприпустимі інформаційні зв'язки усередині ярусів ЯПФ, і виконується десять операцій пересилання даних.

Нижче наведений приклад програми мовою Java з використанням методів класів бібліотеки MPJ Express, яка за допомогою циклічного розподілу ітерацій методом прямокутників обчислює визначений інтеграл $\int_0^\infty \frac{4}{1+x^2}\,dx$, значення якого дорівнює числу π. Обчислення виконуються с кроком 10^{-9}, тобто процеси сукупно виконують мільярд обчислень в той час коли кожен з них виконує лише їх частину.

```
import mpi.*;
class Pi
  {
    static public void main(String []args) throws MPIException
      {
        int myid, numprocs, i;
        double PI25DT = 3.14159265358979323846264 3;
        double h, sum, x;
        double startwtime = 0.0, endwtime;
        long[] n = {1000000000L};
        double[] mypi = new double[1];
        double[] pi = new double[1];
        String  processor_name;
        MPI.Init(args);
        numprocs = MPI.COMM_WORLD.Size();
        myid     = MPI.COMM_WORLD.Rank();
        processor_name = MPI.Get_processor_name();
        System.out.println ("Process " + myid + " of " +
                            numprocs + " is on "+
                            processor_name);
        if (myid == 0) startwtime = MPI.Wtime();
        MPI.COMM_WORLD.Bcast(n, 0, 1, MPI.LONG, 0);
        h   = 1.0 / (double) n[0];
        sum = 0.0;
        for (i = myid + 1; i <= n[0]; i += numprocs)
          {
            x = h * ((double)i - 0.5);
            sum += f(x);
          }
        mypi[0] = h * sum;
        MPI.COMM_WORLD.Reduce(mypi, 0, pi, 0, 1,
                              MPI.DOUBLE, MPI.SUM, 0);
        if (myid == 0)
          {
            endwtime = MPI.Wtime();
            System.out.println ("pi is approximately " +
                                pi[0] + " Error is " +
                                Math.abs(pi[0] - PI25DT));
            System.out.println ( "wall clock time = " +
                                (endwtime-startwtime));
          }
        MPI.Finalize();
      }

    static double f(double a)
      {
        return (4.0 / (1.0 + a*a));
      }
  }
```

МЕТОДИЧНІ ВКАЗІВКИ

для самостійної роботи студентів
і виконання контрольної роботи №1 з дисципліни
«Технології розподілених систем та паралельних обчислень»

для студентів заочної форми навчання.
рівень підготовки “бакалавр”

Укладач: Рольщиков В.Б.

Підписано до друку Формат Папір офсетний
Ум.друк.арк. Тираж Замовлення
Видавництво та друкарня

Одеський державний екологічний університет
65016, Одеса 16, вул. Львівська, 1