Міністерство освіти і науки України

ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ

Методичні вказівки до виконання самостійної роботи
для здобувачів першого (бакалаврського) рівня вищої освіти
освітньої програми «Комп'ютерна інженерія»
галузь знані F Інформаційні технології
спеціальності F7 Комп'ютерна інженерія
денної та заочної форм навчання

Луцьк 2025

УДК 004.75(07)

П63

протокол № _____ від « ____» _____ 20 ____ року.

Голова вченої ради факультету КІТ _____ Інна КОНДІУС

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ

Директор бібліотеки _____ Наталія ПОЛІЩУК

Розглянуто і схвалено на засіданні кафедри комп'ютерної інженерії та безпеки ЛНТУ, протокол № _____ від « ____» _____ 20 ____ року.

Завідувач кафедри КІБ _____ Тарас ТЕРЛЕЦЬКИЙ

Укладач: _____ Катерина МЕЛЬНИК, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

Рецензент: _____ Петро ПЕХ, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

Відповідальний за випуск: _____ Тарас ТЕРЛЕЦЬКИЙ, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

П63 **Паралельні та розподілені обчислення**: методичні вказівки до самостійної роботи для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузі знань F Інформаційні технології, спеціальності F7 Комп'ютерна інженерія денної та заочної форм навчання / уклад. К.В. Мельник. Луцьк: ЛНТУ, 2025. 20 с.

Методичні вказівки до самостійної роботи з освітньої компоненти «**Паралельні та розподілені обчислення**» складені відповідно до діючої програми курсу.

Призначені для здобувачів вищої освіти спеціальності F7 Комп'ютерна інженерія освітньої програми «Комп'ютерна інженерія».

ЗМІСТ

ВСТУП

Самостійна робота сприяє глибшому засвоєнню та закріпленню знань. Для виконання самостійних робіт передбачається використання знань, що отримані на лекціях, лабораторних роботах, а також самостійно в межах вивчення курсу «Паралельні та розподілені обчислення».

Самостійна робота складається з теоретичної та практичної частини.

В теоретичній частині здобувач повинен показати розуміння матеріалу, набуті знання на лекціях та під час самостійного опрацювання окремих тем. Питання самостійної роботи включені до екзаменаційних білетів.

Практична частина передбачає написання програм, що вирішують певні, поставлені перед здобувачем задачі. Дані до тестування розроблених програм студент обирає самостійно або по рекомендації викладача. Тестові розрахунки повинні переконливо свідчити про те, що розроблена програма або програмні засоби працюють належним чином.

Тема 1. Огляд паралельних обчислень. Задача про філософів, що обідають

Література: [основна: 4-8; додаткова:1-7].

Питання для вивчення

1. Асиметрія: Один філософ завжди бере спочатку праву виделку, а потім ліву (порушення циклічного очікування).

2. Семафор-обмеження: Дозволити лише *N-1* філософу (наприклад, 4 з 5) сидіти за столом одночасно, гарантуючи, що принаймні один філософ зможе захопити обидві виделки.

3. Критична секція (critical section): що це таке і чому її необхідно захищати?

4. Стан гонки (race condition): які умови призводять до race condition і як lock (або Monitor) його запобігає?

5. Взаємоблокування (deadlock): які чотири необхідні умови мають бути виконані для виникнення deadlock?

6. Голодування (starvation): чим голодування відрізняється від взаємоблокування? Наведіть приклад механізму, який може викликати голодування.

7. Атомарні операції: коли і чому слід використовувати System.Threading.Interlocked замість lock? (Концепція Compare-and-Swap, CAS).

8. Закон Амдала (Amdahl's Law): як він обмежує максимальне прискорення програми при додаванні ядер? (Покажіть розрахунок з часткою послідовного коду).

9. Навантаження синхронізації (synchronization overhead): чому використання занадто частих або дорогих блокувань може зробити паралельну програму повільнішою за послідовну?

10. Масштабованість (scalability): що таке сильна та слабка масштабованість? яку з них складніше досягти?

11. Модель пам'яті (memory model): у чому ключова різниця між архітектурою з спільною пам'яттю та розподіленою пам'яттю?

12. Кеш-когерентність та volatile: що таке проблема кеш-когерентності та як ключове слово volatile намагається її пом'якшити в C# (та чому воно не завжди є достатнім для складної синхронізації)?

Контрольні питання

1. Що таке паралельні обчислення?
2. Які основні типи паралельних обчислень?
3. Які рівні паралелізму існують?
4. Які переваги та недоліки паралельних обчислень?
5. У яких сферах використовуються паралельні обчислення?
6. Які типи паралельних архітектур існують?
7. Які мови програмування використовуються для паралельних обчислень?
8. Які методи аналізу та оцінки паралельних алгоритмів?
9. Які фактори впливають на ефективність паралельних алгоритмів?
10. Наведіть приклади використання паралельних обчислень в наукових дослідженнях.

Тема 2. Процеси та синхронізація. Задача про письменників та читачів

Література: [основна: 4-8; додаткова:1-6].

Питання для вивчення

1. Чому письменники повинні мати взаємне виключення, а читачі — ні? Наведіть приклад, до чого призведе одночасний запис.

2. Яка небезпека виникає, якщо читач отримує доступ до ресурсу, коли письменник усе ще його змінює? (Відповідь: читання частково оновлених/некоректних даних).

3. Які примітиви синхронізації (наприклад, Семафори, М’ютекси, Умовні Змінні / Monitor.Wait/Pulse) необхідні для реалізації цієї задачі?

4. Роль лічильника читачів: як використовується спільна змінна-лічильник (наприклад, readCount) для керування доступом читачів? Чому сам цей лічильник вимагає захисту (наприклад, через м’ютекс)?

5. Пріоритет читачів (механізм): опишіть логіку, при якій перший читач, що входить, повинен блокувати письменників, а останній читач, що виходить, звільняти їх.

6. Пріоритет письменників (механізм): як можна модифікувати логіку, щоб новий читач, який приходить, блокувався, якщо є очікуючий письменник? (Це вимагає додаткового механізму черговості або блокування).

7. Як клас System.Threading.ReaderWriterLockSlim у .NET реалізує обидва варіанти пріоритету ефективніше, ніж класичні семафори?

8. Поясніть різницю між Shared Lock (для читачів) та Exclusive Lock (для письменників).

9. Чи може виникнути Deadlock у задачі «Письменники-Читачі» та за яких умов? (Наприклад, якщо письменник чекає на ресурс, який утримується читачем, який, своєю чергою, чекає на звільнення іншого ресурсу, який утримується першим письменником).

Контрольні питання

1. Чим відрізняються процеси від потоків?
2. Які переваги використання багатопроцесорних систем?
3. Які типи синхронізації існують?
4. Які проблеми виникають при синхронізації процесів?
5. Що таке семафори?
6. Що таке м'ютекси?
7. Що таке бар'єри?
8. Які мови програмування надають підтримку синхронізації процесів?

Тема 3. Потоки в C#. Задача про сплячого перукаря

Література: [основна: 4-8; додаткова:1-7].

Питання для вивчення

1. Поясніть, що означає «перукар засинає» в термінах потоків C#. Який механізм (Wait, Monitor.Wait, Semaphore.Wait) використовується для «усипляння» перукаря?

2. Опишіть сценарій, за якого в цій задачі може виникнути взаємоблокування (наприклад, якщо перукар чекає клієнта, а клієнти чекають сигналу від перукаря).
3. Чи може клієнт голодувати (ніколи не отримати стрижку)? Якщо так, за яких умов?
4. Як кімната очікування (буфер) запобігає нескінченному зростанню черги? Який примітив C# найкраще підходить для моделювання обмеженої кількості стільців?
5. Яку роль виконує lock (м'ютекс) у цій задачі? Чи повинен клієнт блокувати крісло, коли стрижеться, чи це робить перукар?
6. Якщо ми використовуємо SemaphoreSlim для моделювання вільних стільців, яке початкове значення повинен мати семафор? Хто викликає Release() (звільнення) – перукар чи клієнт?
7. Як можна використовувати Monitor.Wait() та Monitor.Pulse() для того, щоб клієнт будив сплячого перукаря?
8. Чи потрібен lock лише при зміні лічильника клієнтів у кімнаті очікування, чи для інших змінних також?
9. Наскільки задача «Сплячий перукар» схожа на класичну задачу «Виробник-Споживач»? Хто є виробником і хто є споживачем у цьому контексті?
10. Як зміниться логіка синхронізації та які додаткові примітиви знадобляться, якщо в перукарні працюватиме кілька перукарів (наприклад, три)?

Контрольні питання

1. Що таке потік в C#?
2. Чим потоки відрізняються від завдань?
3. Які переваги використання потоків?
4. Як синхронізувати роботу потоків?
5. Які класи та методи використовуються для роботи з потоками в C#?
6. Що таке ThreadPool клас?

Тема 4. Синхронізація за допомогою об'єктів ядра і конструкцій користувацького режиму

Література: [основна: 4-8; додаткова:2-7].

Питання для вивчення

1. Який зв'язок між оператором lock у C# та класом System.Threading.Monitor? Який об'єкт використовується як розділена змінна для блокування?

2. За яких умов доцільно використовувати "активне очікування" (SpinLock або SpinWait) замість традиційного блокування (lock)? Яка межа часу (time limit) зазвичай застосовується перед переходом до блокування на рівні ядра?

3. Чому операції Interlocked.Increment/CompareExchange є найшвидшими? Яка апаратна інструкція (інструкція процесора) лежить в основі їхньої атомарності?

4. Чим SemaphoreSlim відрізняється від Semaphore (який завжди є об'єктом ядра)? Коли SemaphoreSlim переходить у режим ядра?

5. Як цей механізм підвищує паралелізм порівняно з простим lock? Поясніть різницю між Shared Lock та Exclusive Lock у цьому контексті.

6. У чому ключова відмінність Mutex від lock (чи Monitor)? За яких обставин програмісту необхідно використовувати іменований Mutex?

7. У чому фундаментальна різниця між AutoResetEvent та ManualResetEvent? Який механізм краще підходить для послідовної сигналізації, а який — для широкомовної?

8. Об'єкти Ядра та Взаємоблокування (Deadlock): Чи схильні об'єкти ядра до проблеми взаємоблокування? Поясніть, як Mutex може призвести до Deadlock, якщо два потоки захоплюють ресурси у зворотному порядку.

9. Чому використання using або явного виклику Dispose() є критично важливим для об'єктів ядра (наприклад, Mutex або Semaphore)? Що може статися, якщо об'єкт ядра не буде коректно звільнено?

10. Якщо вам потрібно захистити спільну змінну в межах одного процесу, який механізм ви оберете (lock, Mutex чи Interlocked) і чому? Сформулюйте ієрархію вибору за пріоритетом швидкості/накладності (Overhead).

Контрольні питання

1. Що таке конструкції користувацького режиму для синхронізації?

2. Які переваги використання конструкцій користувацького режиму?

3. Які недоліки використання конструкцій користувацького режиму?

4. Які фактори слід враховувати при виборі між об'єктами ядра і конструкціями користувацького режиму?

Тема 5. Асинхронна модель програмування

Література: [основна: 4-8; додаткова:5-7]

Питання для вивчення

1. Яка єдина мета ключового слова async при оголошенні методу? Що відбувається в коді при додаванні цього модифікатора?

2. Що відбувається з потоком виконання (потоком, що викликає), коли він досягає ключового слова await? Куди передається керування, і що є «точкою продовження»?

3. Чому використання await в одному методі зазвичай вимагає, щоб метод, який його викликає, також був позначений як async?

4. Які три основні типи повернення можуть мати async методи, і коли слід використовувати кожен із них (Task, Task<T>, void)? Яке обмеження застосовується до використання async void?

5. Коли і чому слід використовувати ConfigureAwait(false)? Яку проблему це вирішує (проблема захоплення контексту), і які переваги це дає на серверних/бібліотечних додатках?

6. Опишіть класичний сценарій асинхронного Deadlock у застосунках з графічним інтерфейсом користувача (UI) або ASP.NET (до Core), коли синхронний код блокує асинхронний (.Wait() або .Result).

7. Як обробляються винятки, що виникають усередині Task, якщо він не був «очікуваний» (awaited)? Як try-catch використовується навколо await?

8. Який метод використовується для паралельного очікування завершення кількох незалежних асинхронних завдань (Task)? Чому простий послідовний await task1; await task2; є неефективним?

9. Як у C# правильно імітувати тривалу обчислювальну роботу (CPU-Bound) у фоновому потоці, використовуючи Task.Run(), і коли це має сенс?

Контрольні питання

1. Що таке асинхронна модель програмування?
2. Які переваги використання асинхронної моделі програмування?
3. Що таке async/await (C#)?
4. Які проблеми виникають при асинхронному програмуванні?
5. Які інструменти та бібліотеки використовуються для асинхронного програмування в різних мовах?

Тема 6. TPL. Паралельне програмування

Література: [основна: 4-8; додаткова:2-7].

Питання для вивчення

1. У чому фундаментальна різниця між Task (завданням) і Thread (низькорівневим потоком)? Чому TPL заохочує використання Task для більшості сценаріїв паралелізму?
2. Що таке TaskScheduler і яку роль він відіграє в TPL? Як TaskScheduler.Default допомагає оптимізувати планування завдань на процесорах?
3. Поясніть концепцію ієрархії завдань (Parent-Child Tasks). Як можна гарантувати, що батьківське завдання (Parent Task) не завершиться, доки не завершаться всі його дочірні завдання (Child Tasks)?
4. У чому перевага використання Parallel.For над звичайним циклом for? У яких випадках використання Parallel.For може насправді уповільнити виконання (наприклад, для CPU-Bound vs. I/O-Bound операцій)?
5. Як TPL автоматично розбиває колекцію для Parallel.ForEach? Які механізми (наприклад, Partitioner) можна використовувати для більш тонкого налаштування розбиття?
6. Як TPL обробляє винятки, що виникають у кількох паралельних завданнях? Чому для обробки винятків з Task або Parallel.For потрібно використовувати AggregateException?

7. У чому різниця між LINQ та PLINQ? Як оператор .AsParallel() інформує TPL про необхідність паралелізації запиту?

8. Чому при паралельному програмуванні життєво важливо використовувати токени скасування? Як Parallel.For реагує на запит скасування?

9. Як можна обмежити максимальну кількість потоків, що використовуються TPL, якщо ваша програма споживає занадто багато ресурсів (наприклад, через обмеження зовнішньої системи)?

10. При використанні Parallel.For або Parallel.ForEach, як можна уникнути стану гонки при агрегуванні результатів, використовуючи локальні змінні потоку (наприклад, через перевантаження Parallel.For з локальними змінними)?

Контрольні питання

1. Які переваги використання TPL?
2. Що таке Task клас?
3. Як використовувати Parallel.For цикл для обробки масивів?
4. Що таке Parallel.ForEach цикл?
5. Як використовувати Parallel.ForEach цикл для обробки колекцій?
6. Які механізми синхронізації пропонує TPL?

Тема 7. Async і Await

Література: [основна: 4-8; додаткова:1-7].

Питання для вивчення

1. Яка єдина основна мета ключового слова async при оголошенні методу? Що відбувається на етапі компіляції (зміна структури коду) при додаванні цього модифікатора?

2. Що відбувається з потоком виконання (потоком, що викликає), коли він досягає оператора await над незавершеним Task? Куди передається керування?

3. Поясніть роль машини станів, згенерованої компілятором. Як вона допомагає методу "запам'ятати", де продовжити виконання після завершення Task?

4. Що таке продовження (continuation) в контексті Task? Де саме виконується код, що слідує за await (наприклад, у тому ж потоці чи в пулі потоків)?

5. Що таке Контекст Синхронізації (SynchronizationContext) і яку роль він виконує в UI-додатках (WPF/WinForms)? Чому await за замовчуванням намагається "захопити" цей контекст?

6. Коли слід використовувати ConfigureAwait(false)? Які переваги це дає у бібліотечному чи серверному коді з погляду продуктивності та уникнення Deadlock?

7. Опишіть класичний сценарій Deadlock при використанні async та await у UI-середовищах. Як виклик .Wait() або .Result на async методі призводить до взаємоблокування?

8. Які три основні типи може повертати async метод, і коли слід використовувати кожен із них (Task, Task<T>, void)? Яка небезпека при використанні async void?

9. Який метод TPL слід використовувати для ефективного паралельного очікування завершення кількох незалежних асинхронних завдань? Чому послідовне використання await для незалежних завдань є неефективним?

10. Поясніть, як асинхронність обробляє I/O-Bound (введення/виведення) операції (без блокування потоку) і як слід правильно обробляти CPU-Bound (обчислювальні) операції в асинхронному коді (через Task.Run()).

Контрольні питання

1. Що таке асинхронне програмування?

2. Як async та await ключові слова використовуються в C# для асинхронного програмування?

3. Що таке асинхронні потоки?

4. Які переваги використання асинхронних потоків?

5. Які проблеми виникають при асинхронному програмуванні з async та await?

6. Які поширені помилки при використанні async та await?

Тема 8. Технологія OpenMP

Література: [основна: 4-8; додаткова:1-7]

Питання для вивчення

1. Що таке OpenMP і для яких архітектур (моделей пам'яті) вона призначена? Чим OpenMP принципово відрізняється від MPI?

2. Як OpenMP досягає паралелізму? Яку роль відіграють директиви компілятора (прагми) у перетворенні послідовного коду на паралельний?

3. Поясніть модель виконання Fork-Join в OpenMP. Коли створюються нові потоки (форк), і коли вони зливаються назад (джойн)?

4. Поясніть різницю між спільними (Shared) та приватними (Private) змінними в контексті OpenMP. Яка директива використовується для оголошення спільної області паралельною?

5. Як директива omp for розподіляє ітерації циклу між доступними потоками? Що таке схеми планування (schedule clauses) (наприклад, static, dynamic) і коли їх слід застосовувати?

6. Які механізми синхронізації пропонує OpenMP, і коли їх слід використовувати? (Наприклад, critical, atomic, barrier). У чому різниця між critical та atomic?

7. Яку проблему вирішує директива reduction? Наведіть приклад її використання для паралельного обчислення суми елементів масиву.

8. Які джерела накладних витрат можуть уповільнити виконання OpenMP-програми (наприклад, витрати на форк-джойн, синхронізацію)?

9. У яких випадках (наприклад, тип задачі, розмір циклу) використання OpenMP є найбільш ефективним, а коли воно може не дати прискорення або навіть уповільнити програму?

10. Поясніть проблему помилкового спільного доступу (False Sharing). Як вона пов'язана з кеш-лініями процесора, і як програміст OpenMP може її уникнути?

Контрольні питання

1. Які переваги використання OpenMP?

2. Які директиви OpenMP використовуються для розподілу завдань між потоками?

3. Які директиви OpenMP використовуються для синхронізації доступу до спільних даних?

4. Що таке parallel for цикл OpenMP?

5. Як використовувати parallel sections секцію для виконання незалежних блоків коду?

6. Що таке критичні секції, атомарні операції та бар'єри OpenMP?

Тема 9. Технологія MPI

Література: [основна: 4-8; додаткова:1-7].

Питання для вивчення

1. Що таке MPI і для яких архітектур вона є основною (наприклад, кластери, суперкомп'ютери)? Чим MPI принципово відрізняється від OpenMP у плані доступу до пам'яті?

2. Поясніть модель передачі повідомлень (message passing). Як процеси MPI взаємодіють між собою, і чому кожен процес має власний локальний адресний простір?

3. Що таке комунікатор у MPI? Яку роль відіграє стандартний комунікатор MPI_COMM_WORLD?

4. Що таке ранг процесу і що таке розмір комунікатора? Чому ранг процесу є критично важливим для умовного виконання коду?

5. Поясніть різницю між блокуючими (MPI_Send, MPI_Recv) та неблокуючими (MPI_Isend, MPI_Irecv) операціями пересилання. Коли слід використовувати неблокуючий обмін?

6. Що таке колективні операції? Опишіть призначення трьох основних операцій: MPI_Bcast (розсилка), MPI_Scatter (розподіл) та MPI_Gather (збір/агрегація).

7. Яку роль виконує операція MPI_Barrier? Як вона забезпечує синхронізацію всіх процесів у комунікаторі?

8. Яке основне джерело накладних витрат (overhead) при використанні MPI (окрім обчислень)? Як мінімізувати вплив мережевої затримки (latency)?

9. Що таке декомпозиція задачі (Domain Decomposition) у MPI? Які проблеми можуть виникнути при невдалій декомпозиції (наприклад, Незбалансованість Навантаження)?

10. Як може виникнути Deadlock при використанні MPI? Наведіть приклад, коли два процеси одночасно викликають блокуючий MPI_Send один до одного.

Контрольні питання

1. Що таке MPI (Message Passing Interface)?
2. Які моделі паралельного виконання пропонує MPI?
3. Які ключові компоненти MPI?
4. Які основні типи комунікацій MPI?
5. Які колективні операції MPI підтримує?
6. Як використовувати MPI для синхронізації роботи процесів?
7. Як MPI використовується для оптимізації продуктивності паралельних програм?
8. Як MPI використовується на кластерах комп'ютерів?
9. Як MPI використовується на хмарних платформах?

Тема 10. Розподілене програмування в хмарі

Література: [основна: 1-8; додаткова:1-7].

Питання для вивчення

1. Поясніть CAP-теорему (Consistency, Availability, Partition Tolerance). Які два з трьох властивостей можна гарантувати одночасно в розподіленій системі, і як цей вибір впливає на архітектуру хмарних баз даних (наприклад, NoSQL)?

2. Які переваги та недоліки мікросервісів порівняно з монолітною архітектурою в хмарі? Як мікросервіси впливають на обмін даними та синхронізацію?

3. Що таке FaaS (Function as a Service)? Як безсерверна модель (наприклад, AWS Lambda, Azure Functions) змінює спосіб написання та масштабування розподілених функцій?

4. У чому різниця між синхронним (наприклад, REST, gRPC) та асинхронним (наприклад, черги повідомлень) обміном даними між хмарними сервісами? Коли слід використовувати асинхронну чергу (наприклад, RabbitMQ, Kafka) для розподілених завдань?

5. Чому класичні ACID-транзакції (Atomic, Consistent, Isolated, Durable) важко реалізувати у широко розподілених системах? Які існують альтернативні моделі, як-от Saga pattern або Eventually Consistency?

6. Що таке ідемпотентність у розподілених API? Чому ідемпотентність критично важлива при обробці повідомлень з черг, де можлива повторна доставка?

7. Поясніть, що таке горизонтальне масштабування (Sharding) баз даних. Які основні проблеми виникають при реалізації Sharding (наприклад, проблема міграції даних, гарячі розділи)?

8. Як хмарні балансувальники навантаження (load balancers) допомагають розподіляти трафік? Поясніть різницю між балансуванням на рівні мережі (Layer 4) та балансуванням на рівні додатків (Layer 7).

9. Яку роль відіграють контейнери (Docker) та оркестрація (Kubernetes) у спрощенні розгортання та управління розподіленими додатками в хмарі?

10. Що таке розподілене трасування (наприклад, Jaeger, Zipkin) і чому воно необхідне для налагодження мікросервісних додатків? Яку проблему (наприклад, ланцюжок викликів) воно допомагає вирішити?

Контрольні питання

1. Що таке розподілене програмування в хмарі?
2. Які моделі розподіленого програмування в хмарі існують?
3. Які основні хмарні платформи для розподіленого програмування? (AWS, Azure, GCP, etc.)
4. Які послуги пропонують хмарні платформи для розподіленого програмування? (Compute, Storage, Networking, Databases, etc.)

5. Які інструменти та бібліотеки можна використовувати для розробки розподілених програм в хмарі?

6. Як масштабувати розподілені програми в хмарі?

7. Як забезпечити безпеку розподілених програм в хмарі?

8. Які найкращі практики при розробці та розгортанні розподілених програм в хмарі?

Перелік рекомендованих джерел

Базова

1. Carnell, J., & Sánchez, I. H. (2021). Spring microservices in action. Simon and Schuster.

2. Ford, N., Richards, M., Sadalage, P., & Dehghani, Z. (2021). Software architecture: The hard parts. «O'Reilly Media, Inc.»

3. Heather Adkins, Betsy Beyer, Paul Blankinship, Piotr Lewandowski, Ana Oprea, and Adam Stubblefield. Building Secure and Reliable Systems Published by O'Reilly Media, 2020. 557 с.

4. Корочкін О.В.. Русанова О.В. Паралельні та розподілені обчислення. Вибрані розділи: Навч. посібник до кредитного модуля «Паралельні та розподілені обчислення» для студентів освітньої програми «Комп'ютерні системи та мережі», за спеціальністю 123 "Комп'ютерна інженерія" К.: КПІ імені Ігоря Сікорського, 2020. 123 с.

5. Коцовський В. М. К75 Теорія паралельних обчислень: навчальний посібник. Ужгород: ПП «АУТДОР-Шарк», 2021. 188 с.

6. Кузьма К.Т., Мельник О.В. Паралельні та розподілені обчислення: навчальний посібник для вищих закладів освіти. Миколаїв: ФОП Швець В.М., 2020. 172 с.

7. Лисенко В. Ф. Паралельні та розподілені обчислення: навч. посіб. Кропивницький: Видавець, 2021. 153 с.

8. Наконечна О. А., Ярмоленко Т. А., Алексеєнко В. В., Якимчук Б. М. Інструктивно-методичні рекомендації з дисципліни «Технології розподілених систем та паралельних обчислень. Житомир:

Вид-во ЖДУ ім. Івана Франка, 2023. 74 с.

Інформаційні ресурси

9. Parallel Patterns Library (PPL). URL: https://cutt.ly/gr1uBNh (date of access: 16.05.2025).

10. Home - OpenMP. OpenMP. URL: http://www.openmp.org/ (дата звернення: 16.05.2025).

11. Мельник К., Мельник В., Григоришин А. Автоматичний збір інформації (парсинг) в мережі. Computer-integrated technologies: education, science, production. 2020. № 39. С. 151–156. URL: https://doi.org/10.36910/6775-2524-0560-2020-39-26.

12. Дослідження покращення внутрішніх та зовнішніх параметрів швидкодії зв’язку на кластері комунікуючих віртуальних машин / В. Мельник та ін. Computer-integrated technologies: education, science, production. 2020. № 39. С. 162–174. URL: https://doi.org/10.36910/6775-2524-0560-2020-39-28.

13. Multithreading and Parallel Programming in C#. http://surl.li/mfdjf. (дата звернення: 16.05.2025)

14. В.М. Мельник, К.В. Мельник, О.І. Кузьмич, Н.В. Багнюк, О.Р. Кравець. Підвищення параметрів швидкодії зв’язку на кластері комунікуючих віртуальних машин. // Програмовані логічні інтегральні схеми та мікропроцесорна техніка в освіті і виробництві: збірник тез міжнародного науково-практичного семінару молодих вчених та студентів (12-13 травня 2020 р.) / відп. ред. П.А. Пех. Луцьк, 2020. С. 41-43.

15. Khrystynets N., Melnyk K., Lavrenchuk S., Miskevych O., Kostiuchko S. Multiprocessing as a Way to Optimize Queries. Advances in Transdisciplinary Engineering, 2024, №48, pp. 455–464 / URL: https://doi.org/10.3233/ATDE231357.

П63

Паралельні та розподілені обчислення: методичні вказівки до самостійної роботи для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп’ютерна інженерія» галузі знань F Інформаційні технології, спеціальності F7 Комп’ютерна інженерія денної та заочної форм навчання / уклад. К.В. Мельник. Луцьк: ЛНТУ, 2025. 20 с.

Методичне видання до виконання самостійної роботи з дисципліни «**Паралельні та розподілені обчислення**» складене відповідно до діючої програми курсу.

Призначене для здобувачів вищої освіти спеціальності F7 Комп’ютерна інженерія освітньої програми «Комп’ютерна інженерія».

Комп’ютерний набір К.В. Мельник

Редактор К.В. Мельник

Підп. до друку «___» _____ 2025р.

Формат 60х84/16. Папір офс. Гарнітура Таймс.

Ум. друк. арк. ____. Тираж 10 прим. Зам. ___

Відділ іміджу та промоцій

Луцького національного технічного університету

43018, м. Луцьк, вул. Львівська, 75