Чернівецький національний університет імені Юрія Федьковича
Факультет математики та інформатики
Кафедра прикладної математики та інформаційних технологій

“ЗАТВЕРДЖУЮ”
Декан факультету
математики та інформатики

_____ Ольга МАРТИНЮК

“ 28 ” серпня 2026 року

РОБОЧА ПРОГРАМА
навчальної дисципліни

Паралельне програмування
(назва навчальної дисципліни)

Обов’язкова
(вказати: обов’язкова / вибіркова)

Освітньо-професійна програма Технології програмування та комп’ютерне моделювання
(назва програми)

Спеціальність F1 Прикладна математика
(вказати: код, назва)

Галузь знань F Інформаційні технології

Рівень вищої освіти другий (магістерський)
(вказати: перший бакалаврський/другий магістерський)

Факультет математики та інформатики
(назва факультету/інституту, на якому здійснюється підготовка фахівців за вказаною освітньо-професійною програмою)

Мова навчання українська
(вказати: на якій мові читається дисципліна)

Чернівці – 2026

Робоча програма навчальної дисципліни "*Паралельне програмування*" **складена відповідно до** освітньо-професійної програми "Технології програмування та комп'ютерне моделювання" та навчального плану підготовки здобувачів вищої освіти за спеціальністю F1 Прикладна математика (галузь знань F Інформаційні технології). Дисципліна реалізує обов'язковий освітній компонент ОК2 "Паралельне програмування" в обсязі 4.5 кредитів ECTS (135 годин).

Розробник:

Краснокутська Інесса Володимирівна, доцент кафедри прикладної математики та інформаційних технологій, кандидат фізико-математичних наук, доцент

Викладачі, що забезпечують читання даної навчальної дисципліни:

Краснокутська Інесса Володимирівна, доцент кафедри прикладної математики та інформаційних технологій, кандидат фізико-математичних наук, доцент

Погоджено з гарантом ОП _____ **Микола ФІЛІППЧУК**
(підпис)

Затверджено на засіданні кафедри прикладної математики та інформаційних технологій

Протокол № 2 від « 28 » серпня 2026 року

Завідувач кафедри _____ **Ярослав БІГУН**
(підпис)

Схвалено методичною радою факультету математики та інформатики

Протокол № 1 від « 28 » серпня 2026 року

Голова методичної ради _____ **Тоня ФРАТАВЧАН**
(підпис)

МЕТА ОСВІТНЬОГО КОМПОНЕНТА

Мета дисципліни:

- формування у студентів необхідного рівня теоретичної і практичної підготовки для організації ними ефективного розв‘язку великих задач на обчислювальних системах з паралельною архітектурою з використанням сучасних пакетів та стандартів паралельного програмування;
- оволодіння методами та технологіями побудови паралельних алгоритмів;
- набуття практичних навичок гетерогенного програмування;
- вивчення методів оптимізації програм для паралельних обчислювальних систем та забезпечення вміннями узгодження потенціального паралелізму алгоритму з наявним апаратним забезпеченням.

Завдання дисципліни:

- вивчення стандартів паралельного програмування (багатопотоковість, обмін повідомленнями між процесами, гетерогенні обчислення) та їх реалізацій;
- вивчення технологій паралельного програмування (CUDA, OpenCL), паралельних числових методів рішення задач з метою використання набутих знань для розробки паралельних алгоритмів та створення паралельних програм;
- отримання знань і умінь для розробки паралельних алгоритмів з метою виконання на GPU;
- засвоєння знань, умінь та навичок гетерогенного програмування.

Знання, які студент отримає в результаті вивчення даної дисципліни, відіграватимуть важливу роль у процесі його професійного формування та зростання.

ПРЕРЕКВІЗИТИ

Для успішного засвоєння дисципліни необхідні добрі знання класичного програмування на CPU мовами C, C++, Java, Python, JavaScript, C#, що закладаються під час навчання в бакалавраті в навчальних курсах, пов’язаних з вивченням мов програмування.

РЕЗУЛЬТАТИ НАВЧАННЯ

В результаті вивчення дисципліни студент має набути таких **компетентностей**:

знати:

- мови та спеціальні бібліотеки паралельного програмування (CUDA, OpenCL та ін.);
- принципи організації неграфічних обчислень на графічних процесорах;
- специфіку розробки багатопотокових додатків;
- проблеми синхронізації та засоби їх вирішення при паралельному програмуванні;
- критерії вибору програмно-апаратної платформи для вирішення обчислювально-складних задач заданого класу;

вміти:

- проводити декомпозицію поставленого завдання на менші частини, які можуть бути виконані одночасно і незалежно одна від одної, з метою побудови паралельного алгоритму заданої задачі;
- проводити оптимізацію паралельних алгоритмів;
- розробляти ефективні паралельні програми;
- створювати багатопотокові додатки та задіювати систему планування виконання потоків;
- самостійно розробляти схему взаємодії процесів, обирати оптимальні засоби реалізації взаємодії процесів;
- працювати на паралельних комп'ютерах та кластерах;
- виконувати гетерогенні обчислення (CPU та GPU);
- використовувати принципи роботи з пам'яттю та потоками в програмах на CUDA;
- задіювати засоби синхронізації паралельних обчислень без втрати продуктивності.

Під час вивчення дисципліни, відповідно до освітньо-професійної програми, формуються наступні

загальні компетентності:

ЗК3. Знання та розуміння предметної області та розуміння професійної діяльності;

ЗК6. Здатність вчитися і оволодівати сучасними знаннями, інформаційними технологіями та комп'ютерною технікою;

фахові компетентності:

ФК9. Здатність проектувати та розробляти програмне забезпечення для розв'язання формалізованих задач із застосуванням різних парадигм програмування, зокрема об'єктно-орієнтованого, методів паралельного програмування та захисту даних

та отримуються наступні **програмні результати навчання**:

ПРН9. Створювати та програмно реалізовувати алгоритми розв'язання задач, розробляти системне та прикладне забезпечення інформаційних систем і технологій з урахуванням вимог до його якості, надійності, виробничих характеристик;

ПРН10. Аргументувати вибір програмних та технічних засобів для розробки програмних засобів на основі аналізу їх властивостей, призначення і технічних характеристик з урахуванням вимог замовника; мати навички налагодження та тестування програмних і технічних засобів;

ПРН11. Вміти проектувати архітектуру системи з великими обсягами даних.

ОПИС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Загальна інформація

Форма навчання	Рік підготовки	Семестр	Кількість		Кількість годин						Вид підсумкового контролю
			кредитів	годин	лекції	практичні	семінарські	лабораторні	самостійна робота	індивідуальні завдання	
Денна	5	10	4.5	135	30	-	-	30	75	-	іспит
Заочна	-	-	-	-	-	-	-	-	-	-	-

Структура змісту навчальної дисципліни

Назви змістових модулів і тем	Кількість годин											
	денна форма						Заочна форма					
	усього	у тому числі										
		л	п	лаб	інд	с.р.						
1	2	3	4	5	6	7						
Змістовий модуль 1. ***Архітектура GPU та основи гетерогенного програмування GPGPU***												
Тема 1. Архітектура та структура пам'яті GPU	12	3		3		6						
Тема 2. Реалізація масивного паралелізму за допомогою графічних процесорів	12	3		3		6						
Тема 3. Принципи організації паралельних обчислень	12	3		3		6						
Тема 4. Гетерогенне програмування GPGPU	12	3		3		6						

Разом за змістовим модулем 1	48	12		12		24						
Змістовий модуль 2. ***Технологія CUDA від компанії NVIDIA***												
Тема 1. Вступ до CUDA: архітектура та модель програмування	10	2		2		7						
Тема 2. Основи написання CUDA-ядер	10	4		2		6						
Тема 3. Управління пам'яттю в CUDA	10	4		2		6						
Тема 4. Оптимізація продуктивності CUDA-програм	10	2		2		7						
Тема 5. Розробка паралельних алгоритмів з використанням CUDA та додаткових бібліотек	10	2		2		7						
Разом за змістовим модулем 2	50	10		10		33						
Змістовий модуль 3. ***Гетерогенне програмування за допомогою OpenCL***												
Тема 1. Реалізація масивного паралелізму за допомогою графічних процесорів	9	2		2		6						
Тема 2. Написання та виконання OpenCL-ядер	7	2		2		4						
Тема 3. Оптимізація OpenCL-програм: продуктивність та налагодження	7	2		2		4						
Тема 4. Гетерогенні обчислення з	7	2		2		4						

OpenCL: використання CPU, GPU та їх комбінація												
Разом за змістовим модулем 3	30	8		8		16						
Усього годин	120	30		30		75						

Тематика та питання лекційних занять

МОДУЛЬ 1. АРХІТЕКТУРА GPU ТА ОСНОВИ ГЕТЕРОГЕННОГО ПРОГРАМУВАННЯ GPGPU	
Тема 1	Архітектура та структура пам'яті GPU 1. Потокові мультипроцесори (SM) та обчислювальні ядра. 2. Ієрархія пам'яті (глобальна, розділювана, локальна, константна та текстурна). 3. Порівняння архітектури GPU та CPU за орієнтацією на пропускну здатність і латентність.
Тема 2	Реалізація масивного паралелізму за допомогою графічних процесорів 1. Модель виконання SIMT (Single Instruction, Multiple Threads). 2. Формування та планування варпів (warps) і хвильових фронтів (wavefronts). 3. Апаратне приховування затримок доступу до пам'яті шляхом швидкого перемикання контексту потоків.
Тема 3	Принципи організації паралельних обчислень 1. Декомпозиція задач за даними та функціями. 2. Синхронізація паралельних потоків і бар'єри. 3. Типові проблеми паралелізму (гонка даних, взаємне блокування, розгалуження потоків / warp divergence).
Тема 4	Гетерогенне програмування GPGPU 1. Розподіл обчислювального навантаження між хостом (CPU) та прискорювачем (GPU). 2. Життєвий цикл гетерогенної програми та обмін даними через шину PCIe/NVLink. 3. Огляд сучасних програмних платформ та API для GPGPU (CUDA, OpenCL, SYCL, HIP).
МОДУЛЬ 2. ТЕХНОЛОГІЯ CUDA ВІД КОМПАНІЇ NVIDIA	
Тема 1	Вступ до CUDA: архітектура та модель програмування 1. Ієрархія потоків у CUDA (сітка grid, блоки block, потоки thread). 2. Індексація потоків у багатовимірних просторах. 3. Базовий шаблон взаємодії CPU-коду (Host) та GPU-коду (Device).
Тема 2	Основи написання CUDA-ядер 1. Ієрархія потоків у CUDA (сітка grid, блоки block, потоки thread). 2. Індексація потоків у багатовимірних просторах. 3. Базовий шаблон взаємодії CPU-коду (Host) та GPU-коду (Device).
Тема 3	Управління пам'яттю в CUDA 1. Динамічне виділення, копіювання та звільнення пам'яті пристрою (cudaMalloc, cudaMemcpy, cudaFree). 2. Ефективне використання швидкої розділюваної пам'яті (__shared__) та регістрів. 3. Технологія уніфікованої пам'яті (Unified Memory) та закріплена пам'ять (Pinned Memory).

Тема 4	Оптимізація продуктивності CUDA-програм 1. Коалесцентний (об'єднаний) доступ до глобальної пам'яті. 2. Усунення конфліктів банків у розділюваній пам'яті. 3. Максимізація завантаженості мультипроцесорів (Occupancy) та асинхронне виконання за допомогою потоків CUDA (Streams).
Тема 5	Розробка паралельних алгоритмів з використанням CUDA та додаткових бібліотек 1. Реалізація базових паралельних патернів (редукція, префіксна сума, множення матриць). 2. Використання стандартних високооптимізованих бібліотек (cuBLAS, cuFFT, Thrust). 3. Профілювання та пошук вузьких місць за допомогою NVIDIA Nsight.
МОДУЛЬ 3. ГЕТЕРОГЕННЕ ПРОГРАМУВАННЯ ЗА ДОПОМОГОЮ OPEN CL	
Тема 1	Реалізація масивного паралелізму за допомогою графічних процесорів 1. Платформна модель OpenCL (хост та обчислювальні пристрої). 2. Модель виконання на основі NDRange (робочі елементи work-items та робочі групи work-groups). 3. Модель пам'яті OpenCL (global, constant, local, private) та її відображення на апаратне забезпечення GPU.
Тема 2	Написання та виконання OpenCL-ядер 1. Ініціалізація платформи, створення контексту та черг команд (Command Queues). 2. Написання ядер на мові OpenCL C та динамічна компіляція програм під час виконання. 3. Управління буферами даних та аргументами ядра.
Тема 3	Оптимізація OpenCL-програм: продуктивність та налагодження 1. Векторизація обчислень із використанням вбудованих векторних типів даних (float4, int8). 2. Оптимізація розмірів робочих груп та використання локальної пам'яті. 3. Налагодження коду ядер та профілювання подій (OpenCL Events).
Тема 4	Гетерогенні обчислення з OpenCL: використання CPU, GPU та їх комбінація 1. Динамічне виявлення та вибір доступних обчислювальних пристроїв різних вендорів. 2. Координація одночасного виконання задач на CPU та GPU в межах одного контексту. 3. Балансування навантаження та синхронізація між різнорідними обчислювачами за допомогою списків очікування подій.

Тематика та завдання семінарських занять

Семінарські заняття з даної навчальної дисципліни навчальним планом не передбачені.

Тематика та завдання практичних занять

Практичні заняття з даної навчальної дисципліни навчальним планом не передбачені.

Тематика та завдання лабораторних занять

МОДУЛЬ 1. АРХІТЕКТУРА GPU ТА ОСНОВИ ГЕТЕРОГЕННОГО ПРОГРАМУВАННЯ GPGPU	
ЛР 1.1	Реалізація масивного паралелізму за допомогою графічних процесорів 1. Дослідження апаратних характеристик GPU (кількість мультипроцесорів, розмір варпу, обсяги пам'яті) за допомогою системних утиліт. 2. Написання паралельної програми для поелементної обробки великих масивів даних. 3. Порівняння часу виконання та прискорення обчислень на GPU відносно однопотокової та багатопотокової реалізацій на CPU.
ЛР 1.2	Гетерогенне програмування GPGPU 1. Проєктування та реалізація базового циклу гетерогенної програми (виділення пам'яті на пристрої, передача вхідних даних, запуск обчислень, зчитування результатів). 2. Оцінка накладних витрат на пересилання даних через шину PCIe для різних обсягів даних. 3. Дослідження межі доцільності перенесення обчислень на графічний прискорювач.
МОДУЛЬ 2. ТЕХНОЛОГІЯ CUDA ВІД КОМПАНІЇ NVIDIA	
ЛР 2.1	Вступ до CUDA: архітектура та модель програмування 1. Налаштування середовища розробки CUDA Toolkit та програмне опитування властивостей встановлених GPU (cudaGetDeviceProperties). 2. Реалізація базового ядра для додавання векторів із використанням одновимірної та двовимірної індексації потоків. 3. Експериментальне дослідження впливу конфігурації сітки та розміру блоку потоків на швидкість виконання.
ЛР 2.2	Основи написання CUDA-ядер 1. Розробка CUDA-ядер для двовимірної обробки матриць і зображень (наприклад, множення матриць або застосування згорткових фільтрів). 2. Використання допоміжних __device__-функцій та реалізація механізму перевірки помилок CUDA API. 3. Аналіз впливу розгалужень у коді ядра (warp divergence) на продуктивність обчислень.
ЛР 2.3	Управління пам'яттю в CUDA 1. Оптимізація доступу до даних шляхом використання розділюваної пам'яті (__shared__) та синхронізації потоків (__syncthreads()) на прикладі блочного множення матриць або редукції. 2. Реалізація коалесцентного доступу до глобальної пам'яті та усунення конфліктів банків. 3. Порівняння швидкодії стандартного копіювання (cudaMemcpy), закріпленої пам'яті (Pinned Memory) та уніфікованої пам'яті (Unified Memory).
МОДУЛЬ 3. ГЕТЕРОГЕННЕ ПРОГРАМУВАННЯ ЗА ДОПОМОГОЮ OPEN CL	
ЛР 3.1	Написання та виконання OpenCL-ядер 1. Ініціалізація платформи OpenCL, створення контексту, черги команд та компіляція коду ядра з вихідного тексту під час виконання. 2. Налаштування буферів пам'яті (clCreateBuffer), передача аргументів та запуск ядра у двовимірному просторі індексів NDRange. 3. Вимірювання чистого часу виконання ядра на пристрої за допомогою профілювання подій (OpenCL Events).
ЛР 3.2	Гетерогенні обчислення з OpenCL: використання CPU, GPU та їх комбінація

	1. Програмний пошук та порівняння продуктивності виконання одного і того ж OpenCL-ядра на CPU та GPU. 2. Реалізація гібридної обробки масиву даних із одночасним розподілом навантаження між CPU та GPU в межах спільного контексту. 3. Налаштування синхронізації між пристроями через списки очікування подій та пошук оптимальної пропорції розподілу задач.

Індивідуальні науково-дослідні завдання (ІНДЗ)

ІНДЗ з даної навчальної дисципліни не передбачені.

Завдання для самостійної роботи студентів

Назва теми	Кіль-кість годин
Порівняльний аналіз CPU та GPU для масивно-паралельних обчислень (архітектурні відмінності, сфери застосування, обмеження)	8
Порівняльний аналіз CPU та GPU для масивно-паралельних обчислень (архітектурні відмінності, сфери застосування, обмеження)	8
Оптимізація продуктивності CUDA-програм: типові помилки та кращі практики (occupancy, coalesced access, latency hiding)	14
Застосування GPGPU у реальних задачах (обробка зображень, машинне навчання, фізичне моделювання — на вибір студента)	14
Порівняння технологій CUDA та OpenCL для гетерогенних обчислень (портативність, продуктивність, складність розробки)	8
Сучасні тенденції розвитку гетерогенного програмування (multi-GPU, unified memory, heterogeneous computing у HPC та AI)	8
Разом	60

Самостійна робота студента полягає в опрацюванні лекційного матеріалу, більш детальному розгляді окремих питань курсу, підготовці до виконання та захисту лабораторних робіт.

Самостійна робота студента загалом складає 60 годин. Розподіл цих годин за видами робіт: опрацювання лекційного матеріалу, більш детальний розгляд окремих питань курсу – 30 годин; підготовка до виконання та захисту лабораторних робіт – 30 годин.

Оцінювання самостійної роботи студента є складовою частиною оцінювання захисту лабораторних робіт (30% оцінки кожної лабораторної роботи складає оцінювання самостійної роботи).

МЕТОДИ НАВЧАННЯ

Під час викладання навчальної дисципліни «Алгоритми та структури даних» застосовується комплексний підхід до вибору методів та освітніх технологій, спрямований на формування як ґрунтовного теоретичного фундаменту з алгоритмізації, так і стійких практичних навичок розробки, відлагодження та тестування програмного забезпечення мовою C++ / Python.

Освітній процес поєднує такі класичні та сучасні методи й освітні технології:

1. **За джерелами передачі та сприйняття інформації:**
   - *Словесні методи:* розповідь, лекція-пояснення, навчальний діалог та лекція-дискусія для розкриття теоретичних основ алгоритмів та структур даних.
   - *Наочні класичні та цифрові методи:* метод мультимедійних презентацій, демонстрація розбору та виконання програмного коду в режимі реального часу в інтегрованому середовищі розробки (IDE), візуалізація алгоритмів (трасування стеку викликів, схеми розподілу динамічної пам’яті).
   - *Практичні методи:* розв'язання алгоритмічних задач на практичних заняттях та виконання лабораторного практикуму на ПК з індивідуальним захистом робіт.
2. **За характером пізнавальної діяльності здобувачів освіти:**
   - *Пояснювально-ілюстративний та репродуктивний методи:* засвоєння принципів організації динамічних структур даних (однозв’язних, двозв’язних і кільцевих списків, стеків, черг), узагальненого програмування та ітераторів; відтворення класичних алгоритмів пошуку, сортування (бульбашкою, вибором, вставками), а також базових операцій на бінарних деревах пошуку та пірамідах за поданими зразками.
   - *Частково-пошуковий та проблемний методи:* розв’язання алгоритмічних задач підвищеної складності, аналіз та порівняння альтернативних рішень (зокрема ітераційних та рекурсивних підходів), самостійний пошук та усунення помилок у програмному коді (дебагінг).
   - *Дослідницький метод:* аналіз ефективності алгоритмів за часом виконання та використанням пам’яті, проектування складних структур даних, побудова алгоритмів.
3. **За формою організації та практичної спрямованості (основний блок):**
   - *Практико-орієнтований лабораторний практикум:* проведення лабораторних занять у комп’ютерних класах із персональним виконанням та захистом лабораторних робіт.
   - *Алгоритмічний тренінг та розв'язання задач:* систематична робота на практичних заняттях із подальшим аналізом і рев'ю коду (code review).
   - *Інформаційно-комунікаційні технології та електронне навчання:* використання електронного навчального курсу в LMS Moodle ЧНУ для доступу до лекційних матеріалів, складання поточних модульних і підсумкових тестів та подання звітів із скріншотами виконання програм.
   - *Самонавчання та самостійна робота:* опрацювання рекомендованої літератури, виконання завдань для самоконтролю з навчального посібника та підготовка до контрольних заходів.

СИСТЕМА КОНТРОЛЮ ТА ОЦІНЮВАННЯ

Методами та формами поточного контролю є:

- усні поточні опитування;
- захист виконаних лабораторних робіт;

Формою підсумкового контролю є усно-письмовий іспит.

Політика освітнього процесу

Політика щодо відвідування. Відвідування занять є обов'язковим. За наявності поважних причин (хвороба, індивідуальний графік навчання) навчання, за погодженням із викладачем, може відбуватись у змішаній (очно-дистанційній) формі. За необхідності, з метою з'ясування всіх незрозумілих під час самостійної роботи питань, потрібно відвідувати консультації викладача.

Політика щодо академічної доброчесності. Студент зобов'язаний самостійно, своєчасно та добросовісно виконувати усі отримані завдання, дотримуючись принципів академічної доброчесності. Під час проведення контрольних заходів заборонені використання сторонньої допомоги, навчальної літератури, конспектів, мобільних пристроїв, інших джерел інформації.

Політика щодо дедлайнів та перескладання. Якщо студент був відсутній на заняттях (з будь-якої причини), він повинен самостійно опрацювати пропущений матеріал та прозвітувати викладачу про виконання відповідних завдань у встановлені ним терміни під час консультацій. Якщо студент з поважної причини пропустив захист лабораторної роботи, він може з дозволу викладача захистити її під час консультації. Відсутність студента на захисті лабораторної роботи без поважної причини відповідає оцінці «0 балів», тобто призводить до заборгованості, яка повинна бути ліквідована студентом до початку підсумкового контролю з наступного модуля. Кінцевий термін ліквідації заборгованості з модульного контролю обмежується початком заліково-екзаменаційної сесії. Лабораторні роботи, які захищаються невчасно без поважних причин, оцінюються на нижчу оцінку (від -10% до -50% від максимальної кількості балів – залежно від терміну затримки здачі). Перезахист лабораторних робіт не дозволяється. Складання (перескладання) іспиту відбувається за встановленим деканатом розкладом.

КРИТЕРІЇ ОЦІНЮВАННЯ ПОТОЧНОГО ТА ПІДСУМКОВОГО КОНТРОЛЮ

Оцінювання знань студентів здійснюється на основі результатів поточного, модульного та підсумкового контролю. Оцінювання здійснюється за програмним матеріалом навчальної дисципліни, засвоєння якого перевіряється пропонованими видами контролю.

Форми проведення поточного контролю та періодичність їх здійснення визначаються робочим планом викладача відповідно до програми дисципліни. Поточний контроль здійснюється під час проведення лабораторних занять і перевірки самостійної роботи студентів, а також під час читання лекцій. Модульний контроль здійснюється за результатами виконаних модульних контрольних робіт та перевірки лабораторних робіт і проектів. Завданнями поточного та модульного контролю є перевірка рівня розуміння та засвоєння лекційного матеріалу, набуття практичних навичок і досвіду виконання індивідуальних і комплексних задач.

Форми підсумкового семестрового контролю визначаються навчальним планом спеціальності F1 «Прикладна математика». Для дисципліни «Паралельне програмування» встановлено семестровий іспит по завершенню вивчення курсу.

Завданням підсумкового контролю (іспиту) є перевірка розуміння студентом програмного матеріалу в цілому, здатності успішно розв'язувати поставлені практичні задачі та комплексно використовувати отримані знання.

Оцінювання знань здійснюється за 100-бальною шкалою. Результати роботи впродовж навчального семестру оцінюються в ході поточного та модульного контролю на інтервалі оцінок від 0 до 60 балів, а результати підсумкового контролю (іспиту) оцінюються максимум у 40 балів. Загальна підсумкова оцінка з навчальної дисципліни виставляється за загальною сумою балів поточного та модульного контролю і підсумкового контролю.

Нормативно-процесуальні умови оцінювання, допуску та апеляції

Оцінювання знань здобувачів вищої освіти здійснюється відповідно до «Положення про організацію освітнього процесу в Чернівецькому національному університеті імені Юрія Федьковича» та «Положення про контроль і систему оцінювання результатів навчання здобувачів вищої освіти у Чернівецькому національному університеті імені Юрія Федьковича».

Умови допуску до підсумкового семестрового контролю (іспиту):

- Обов'язкове виконання, подання та успішний захист усіх 5 лабораторних робіт, передбачених робочою програмою;
- Проходження поточного комп'ютерного тестування за трьома змістовими модулями;
- Набрання за результатами поточного оцінювання не менше 35 балів із 60 можливих.

Політика ліквідації заборгованостей та апеляції:

- Здобувачі освіти, які не були допущені до іспиту або отримали під час семестрового контролю незадовільну оцінку FX (35–49 балів), мають право на ліквідацію академічної заборгованості за затвердженим графіком у визначені терміни до початку наступного навчального семестру.
- Перескладання іспиту дозволяється не більше двох разів: один раз — викладачу, вдруге – комісії, створеній деканом факультету. Перескладання семестрового контролю з метою підвищення позитивної оцінки не допускається.
- Здобувачі, які отримали оцінку F (1-34 бали), зобов'язані пройти повторний курс вивчення дисципліни.
- У разі незгоди з отриманою на семестровому іспиті оцінкою здобувач вищої освіти має право подати письмову апеляційну заяву на ім'я декана факультету (або ректора Університету) не пізніше 12:00 наступного робочого дня після оголошення результату відповідно до Положення про апеляцію на результати підсумкового семестрового контролю ЧНУ.

Розподіл балів, які отримують студенти

Поточне тестування та самостійна робота							Іспит	Сума
Змістовий модуль №1		Змістовий модуль №2			Змістовий модуль №3			
Т1	Т2	Т1	Т2	Т3	Т1	Т2	40	100
10	5	10	10	10	10	5		

Іспит (40 балів) складається з онлайн-тестування на 20 балів та комплексного практичного завдання на 20 балів. Оцінювання результатів проходження онлайн-тестування здійснюється автоматично, на основі кількості наданих правильних відповідей на питання тесту. За суттєві помилки в розв’язуванні практичного завдання знімається 10-14 балів. За несуттєві помилки в розв’язуванні практичного завдання знімається 2-8 балів. Часткове розв’язання практичного завдання із грубими помилками оцінюється не вище 6 балів.

Загальна підсумкова оцінка з навчальної дисципліни виставляється за загальною сумою балів, набраних студентом, згідно з наступною таблицею:

Шкала оцінювання: національна та ЄКТС

Оцінка за національною шкалою	**Оцінка за шкалою ECTS**	
	Оцінка (бали)	**Пояснення за розширеною шкалою**
Зараховано	A (90-100)	відмінно
	B (80-89)	дуже добре
	C (70-79)	добре
	D (60-69)	задовільно
	E (50-59)	достатньо
Не зараховано	FX (35-49)	(незадовільно) з можливістю повторного складання
	F (0-34)	(незадовільно) з обов'язковим самостійним опрацюванням освітнього компонента до перескладання

ЗАРАХУВАННЯ РЕЗУЛЬТАТІВ НЕФОРМАЛЬНОЇ ОСВІТИ

Порядок та умови визнання й зарахування результатів навчання, здобутих шляхом неформальної та/або інформальної освіти, регламентуються чинним у Чернівецькому національному університеті імені Юрія Федьковича «Порядком визнання у Чернівецькому національному університеті імені Юрія Федьковича результатів навчання, здобутих шляхом неформальної та/або інформальної освіти».

Здобувачі вищої освіти мають право на визнання результатів навчання, отриманих під час проходження онлайн-курсів (на платформах Coursera, edX, Prometheus, Udemy, Stepik, Pluralsight тощо), участі у воркшопах, хакатонах, літніх/зимових школах з програмування за наявності підтверджуючих документів (сертифікатів, свідоцтв).

Механізм та умови визнання результатів:

1. **Перезарахування окремих тем / лабораторних робіт / поточного контролю:**
   - У разі часткової відповідності змісту онлайн-курсу або сертифікованої програми тематиці окремих змістових модулів дисципліни «Програмування», здобувач звертається в усній формі до викладача, подаючи підтверджуючі документи.
   - За умови підтвердження відповідності результатів навчання здобувачу зараховується максимальний бал з відповідної теми (лабораторної роботи, тесту) та звільняється від виконання відповідних завдань. Загальний обсяг балів, який може бути зарахований здобувачу за результатами неформальної/інформальної освіти з окремих тем, не може перевищувати 25% від загальної кількості балів з дисципліни (до 25 балів зі 100).
2. **Повне зарахування освітнього компонента (підсумкового контролю):**

- У разі необхідності повного зарахування навчальної дисципліни здобувач подає письмову заяву на ім’я декана факультету, до якої додає освітню декларацію про попереднє навчання та сертифікати.
- Розгляд документів, співставлення результатів навчання із програмою дисципліни та оцінювання здійснює Предметна комісія, призначена розпорядженням декана.

3. Процедура підтвердження та оцінювання:

- У разі прийняття Предметною комісією рішення про проведення контрольного заходу (співбесіди, контрольного завдання), здобувачу надається до 10 робочих днів для підготовки за питаннями і завданнями, що виносяться на контроль.
- Загальний обсяг визнаних результатів неформальної освіти за весь строк навчання за ОПП «Технології програмування та комп’ютерне моделювання» (галузь знань F «Інформаційні технології») не може перевищувати 35% від загального обсягу освітньої програми.

ПЕРЕЛІК ПИТАНЬ ДЛЯ САМОКОНТРОЛЮ ТА ПІДСУМКОВОГО КОНТРОЛЮ НАВЧАЛЬНИХ ДОСЯГНЕНЬ СТУДЕНТІВ

1. Що таке GPU-архітектура та чим вона відрізняється від архітектури CPU?
2. Поясніть поняття масивного паралелізму та його реалізацію на графічних процесорах.
3. У чому полягає суть гетерогенного програмування GPGPU?
4. Які основні компоненти апаратної архітектури GPU?
5. Опишіть модель виконання програм у середовищі GPGPU.
6. Що таке технологія CUDA та які її основні переваги?
7. Опишіть архітектуру CUDA та модель програмування CUDA.
8. Що таке CUDA-ядро та як відбувається його виконання на GPU?
9. Які типи пам’яті використовуються в CUDA та для чого вони призначені?
10. Поясніть принципи управління пам’яттю в CUDA-програмах.
11. Що таке OpenCL та яку роль воно відіграє в гетерогенному програмуванні?
12. Опишіть модель платформи та виконання програм в OpenCL.
13. Як відбувається написання та запуск OpenCL-ядер?
14. У чому полягають особливості гетерогенних обчислень із використанням CPU та GPU в OpenCL?
15. Порівняйте технології CUDA та OpenCL: основні відмінності та сфери застосування.
16. Які основні напрямки застосування гетерогенного програмування та GPGPU в сучасних обчислювальних системах?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

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

2. CUDA by Example. An Introduction to General-Purpose GPU Programming. Addisson-Wesley - Boston. July 2010.
3. R. Hochberg. Matrix Multiplication with CUDA. A basic introduction to the CUDA programming model. Cambridge University Press, August 11, 2012.
4. David B. Kirk, Wen-mei Hwu. Programming Massively Parallel Processors. A Hands-on Approach. - Published by Elsevier Inc. - 2010.
5. El Zein, A.H.; Rendell, A.P., Generating optimal CUDA sparse matrix–vector product implementations for evolving GPU hardware, Concurrency and Computation: Practice and Experience (2011).
6. B. Wilkinson and M. Allen. Parallel Programming Techniques & Applications Using Networked Workstations & Parallel Computers, 2nd ed. Toronto, Canada: Pearson, 2004.
7. Sun Micorsystems. Multithreading Programming Guide. Santa Clara, CA: Sun Microsystems, 2008.

Додаткова (допоміжна)

1. Качко О.Г. Паралельне програмування. – Харків. нац. ун-т радіоелектроніки. – Харків : ХНУРЕ, 2016. – 403 с.
2. Williams, S. W., D. Patterson, et. al. “The Roofline Model: A pedagogical tool for auto-tuning kernels on multicore architectures.” In Hot Chips, A Symposium on High Performance Chips, Vol. HC20 (August 10, 2008).
3. A. Grama, A. Gupta, G. Karypis, and V. Kumar. Introduction to Parallel Computing, 2nd ed. Reading, MA: Addison Wesley, 2003.
4. Suchard, M.A.; Wanq, Q.; Chan, C.; Frelinger, J.; Cron, A.; West, M., Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures, J Comput Graph Stat. 19 (2) (2010, Jun 1) 419–438.
5. Micikevicius, P., 3D Finite Difference Computation on GPUs using CUDA. Proceeding GPGPU-2 Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units. (2010) ACM, New York, NY.

ІНФОРМАЦІЙНІ РЕСУРСИ

1. Електронний курс в системі Moodle
   https://moodle.chnu.edu.ua/course/view.php?id=9434
2. Технологія CUDA
   https://developer.nvidia.com/cuda
3. Технологія OpenCL
   https://www.khronos.org/opencl

ПОЛІТИКА АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Дотримання політики щодо академічної доброчесності учасниками освітнього процесу при вивченні освітнього компонента регламентовано чинними документами ЧНУ:

- «Кодекс академічної доброчесності та етики Чернівецького національного університету імені Юрія Федьковича»

https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/kodeks-akademichnoi-dobrochesnosti-ta-etyky-chernivetskoho-natsionalnoho-universytetu-imeni-yuriia-fedkovycha/

- «Положенням про виявлення та запобігання академічного плагіату у Чернівецькому національному університеті імені Юрія Федьковича» https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/polozhennia-pro-vyiavlennia-ta-zapobihannia-akademichnomu-plahiatu-u-chernivetskomu-natsionalnomu-universyteti-imeni-yuriia-fedkovycha/
- «Політика використання штучного інтелекту в Чернівецькому національному університету імені Юрія Федьковича» https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/polityka-vykorystannia-shtuchnoho-intelektu-v-chernivetskomu-natsionalnomu-universyteti-imeni-yuriia-fedkovycha-zi-zminamy-zatverdzhenymy-vchenoiu-radoiu-chnu-imeni-yuriia-fedkovycha/

Використання здобувачами освіти інструментів штучного інтелекту (ШІ) під час виконання навчальних завдань:

- допускається за умови дотримання принципів академічної доброчесності (здобувач освіти несе відповідальність за зміст, достовірність і результати поданої роботи, а у разі використання ШІ зобов'язаний зазначити факт і спосіб його застосування);
- не допускається використання ШІ як заміни самостійного виконання завдань або з метою порушення вимог академічної доброчесності;
- додаткові обмеження чи вимоги щодо використання ШІ можуть визначатися викладачем в освітньому процесі для конкретних видів навчальної діяльності відповідно до особливостей дисципліни.