Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра обчислювальної техніки

Основи програмування. Частина 2
Методології програмування

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни

Рівень вищої освіти	*Перший (бакалаврський)*
Галузь знань	*F (12) Інформаційні технології*
Спеціальність	*F2 (121) Інженерія програмного забезпечення*
Освітня програма	*Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»*
Статус дисципліни	*Нормативна*
Форма навчання	*Очна (денна) / дистанційна*
Рік підготовки, семестр	*1 курс, весняний семестр*
Обсяг дисципліни	*6 кредитів, 180 годин* *Лекцій 15 (30 годин), практичних занять 7 (14 годин), лабораторних робіт 9 (30 годин), СРС 106 годин*
Семестровий контроль / контрольні заходи	*Екзамен / МКР, лабораторні роботи*
Розклад занять	*Згідно з розкладом поточного навчального року за адресою https://schedule.kpi.ua*
Мова викладання	*Українська*
Інформація про керівника курсу / викладачів	*Лектор: ас. кафедри ОТ Нечай Д.О., nechaido@gmail.com* *Лабораторні: ас. кафедри ОТ Нечай Д.О.* *Практичні: ас. кафедри ОТ Нечай Д.О.*
Розміщення курсу	*Навчальний курс розміщений на платформі дистанційного навчання «Сікорський» в середовищі Google Workspace for Education:* *https://classroom.google.com/c/ODkzNTMyMDgwMjkw?cjc=morpltuj*

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчання та результати навчання

Метою дисципліни «Основи програмування. Частина 2. Методології програмування» є формування у студентів здатностей (компетентностей) вільного володіння методологіями та парадигмами програмування, синтаксисом мови JavaScript, базовими структурами даних і

колекціями, засобами функціонального програмування, патернами проєктування, асинхронним та реактивним програмуванням, а також практичних навичок побудови багатомодульних бібліотек і застосунків.

Предметом вивчення дисципліни є методології, парадигми, методи та засоби програмування, структури даних, патерни проєктування та асинхронні моделі виконання коду на платформі JavaScript/Node.js.

Завдання вивчення дисципліни:

– оволодіння методологіями та парадигмами програмування (імперативна, об’єктно-орієнтована, функціональна);

– набуття навичок роботи з ітераторами, генераторами та лінивими послідовностями;

– оволодіння базовими структурами даних і колекціями (стек, черга, дек, кеш) та оцінкою їх ефективності;

– набуття навичок функціональної композиції, роботи з функторами та узагальненим кодом;

– оволодіння асинхронним та реактивним програмуванням (callback, Promise, async/await, події, потоки);

– набуття навичок застосування патернів проєктування;

– набуття навичок побудови багатомодульних бібліотек, використання систем контролю версій та інтеграції коду в єдиний проєкт, що є основою курсової роботи.

Дисципліна є фундаментальною для підготовки фахівців з інженерії програмного забезпечення, оскільки формує системне розуміння методологій програмування та типових інженерних рішень, які застосовуються під час подальшого вивчення об’єктно-орієнтованого та системного програмування, компонентів програмної інженерії та розробки на платформі Node.js. Лабораторні роботи послідовно інтегруються в єдиний проєкт, який є основою курсової роботи з освітнього компонента «Основи програмування. Курсова робота».

Дисципліна формує такі компетентності:

Загальні компетентності:

– ЗК01. Здатність до абстрактного мислення, аналізу та синтезу.

– ЗК05. Здатність генерувати нові ідеї (креативність).

Фахові компетентності:

– ФК01. Здатність аналізувати предметні області, формувати та класифікувати вимоги до програмного забезпечення.

– ФК10. Здатність створювати та використовувати програмне забезпечення високопродуктивних комп’ютерних систем.

Результати навчання:

Знання:

– методологій та парадигм програмування, їх сильних сторін і сфер застосування;

– базових структур даних, колекцій та методів оцінки обчислювальної складності;

– принципів функціонального програмування, композиції функцій та функторів;

– моделей асинхронного та реактивного програмування (callback, Promise, async/await, події, потоки);

– типових патернів проєктування та засад узагальненого програмування.

Уміння:

– обґрунтовано обирати парадигму та методологію програмування для розв’язання задачі (ПРН09);

– реалізовувати ітератори, генератори, структури даних та колекції;

– застосовувати функціональну композицію, функтори та патерни проєктування;

– реалізовувати асинхронну та реактивну взаємодію компонентів, обробляти помилки та скасування операцій;

– будувати багатомодульні бібліотеки й інтегрувати код у єдиний проєкт із веденням історії у системі контролю версій;

– збирати та аналізувати науково-технічну інформацію для розв’язання прикладних задач (ПРН17).

Досвід:

– розроблення єдиного програмного проєкту, що інтегрує ітератори й генератори, структури даних, функціональні абстракції, асинхронність, реактивність та патерни проєктування.

2. Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)

Пререквізити: ПО 02.1 «Основи програмування. Частина 1. Базові конструкції», ПО 01.1 «Алгоритми та структури даних. Частина 1. Основи алгоритмізації», ЗО 01 «Комп’ютерна дискретна математика». Кореквізит: ПО 03 «Основи програмування. Курсова робота», що виконується в тому ж семестрі на основі лабораторних робіт дисципліни.

Постреквізити: ПО 07.2–ПО 07.4 «Компоненти програмної інженерії» (частини 2–4), ПО 12 «Об’єктно-орієнтоване програмування», ПО 14 «Системне програмування», ПО 21 «Програмне забезпечення високопродуктивних комп’ютерних систем», вибіркові освітні компоненти Ф-каталогу з розробки програмного забезпечення на платформі Node.js, ПО 11 «Дипломне проєктування».

Навчальний контент

3. Зміст навчальної дисципліни

Навчальний матеріал дисципліни структурований за такими розділами:

Розділ 1. Вступ. Ітератори та генератори (лекції 1–3)

Методології та парадигми програмування, критерії якості коду. Протокол ітерування, генератори та лініві послідовності.

Розділ 2. Інструменти розробки та засоби платформи (лекції 4–5)

Модульна організація коду, пакети та бібліотеки, система контролю версій Git. Типізовані масиви, бінарні дані, таймери.

Розділ 3. Кешування та структури даних (лекції 6–7)

Кешування, мемоізація та стратегії витіснення. Стек, черга, дек, пріоритетна черга.

Розділ 4. Функціональне програмування (лекції 8–9)

Функції вищого порядку, композиція, чисті функції, каррінг, незмінність. Функтори, монади, Maybe/Either.

Розділ 5. Асинхронне та реактивне програмування (лекції 10–13)

Callback, Promise, async/await, обробка помилок і скасування. Події, EventEmitter, потоки та обробка великих наборів даних.

Розділ 6. Патерни проєктування та узагальнене програмування (лекції 14–15)

Породжувальні, структурні та поведінкові патерни: фабрика, компонувальник, адаптер, проксі, декоратор, пул об’єктів, відкритий конструктор. Параметричний поліморфізм, узагальнені структури даних і функції, обмеження типів у TypeScript.

Лекційні заняття

№	Назва теми лекції та перелік основних питань	Год.
1	**Тема: Організація курсу. Методології та парадигми програмування** Ознайомлення з РСО, курсовою роботою та етапами курсу. Парадигми програмування: імперативна, об’єктно-орієнтована, функціональна. Методології програмування та критерії якості коду. Організація роботи у системі контролю версій.	2
2	**Тема: Ітератори та протокол ітерування** Протокол ітератора, Symbol.iterator, ітеровані об’єкти. Вбудовані ітератори колекцій. Ручна реалізація ітераторів. Споживання ітераторів, ітерування з обмеженням у часі.	2
3	**Тема: Генератори та ліниві послідовності** Функції-генератори, оператор yield, делегування yield*. Ліниві та нескінченні послідовності. Генератори як засіб керування потоком виконання.	2
4	**Тема: Модульна організація коду, пакети та система контролю версій Git** Модульні системи CommonJS та ECMAScript Modules. Файл package.json, менеджер пакетів npm, семантичне версіонування, побудова багатомодульних бібліотек. Робота з Git: гілки, коміти, історія, пул-реквести та код-рев’ю; feature-branch робочий процес.	2
5	**Тема: Типізовані масиви, бінарні дані та таймери** Типізовані масиви (TypedArray, ArrayBuffer, DataView), робота з бінарними даними. Таймери: setTimeout, setInterval, setImmediate, process.nextTick.	2
6	**Тема: Кешування, мемоізація та стратегії витіснення** Кешування результатів обчислень, мемоізація функцій. Стратегії витіснення кешу: LRU, LFU, за часом (TTL), власні стратегії. Обмеження розміру кешу.	2
7	**Тема: Стек, черга, дек** Структури даних: стек, черга, дек, пріоритетна черга. Способи реалізації та оцінка обчислювальної складності операцій.	2

№	Назва теми лекції та перелік основних питань	Год.
8	**Тема: Функції вищого порядку, композиція, чисті функції та каррінг** Функції вищого порядку та зворотного виклику. Композиція функцій (compose, pipe), часткове застосування та каррінг. Чисті функції, побічні ефекти, незмінність даних. Декларативна обробка колекцій.	2
9	**Тема: Функтори, монади, Maybe/Either** Контейнери та обгортки значень. Функтор та операція map. Контейнери Maybe та Either для обробки відсутніх значень і помилок. Поняття монади, ланцюжки обчислень.	2
10	**Тема: Асинхронне програмування на callback та Promise** Моделі асинхронності: callback, контракт error-first. Проміси (Promise), стани промісів, ланцюжки. Послідовне та паралельне виконання асинхронних операцій.	2
11	**Тема: Асинхронні функції: async/await, обробка помилок, скасування** Асинхронні функції, async/await, thenable. Обробка помилок в асинхронному коді. Скасування асинхронних операцій (AbortController), таймаути.	2
12	**Тема: Реактивне програмування: події та EventEmitter** Реактивна взаємодія через події. EventEmitter, підписка та відписка, кілька незалежних слухачів. Патерн «Спостерігач» та Observable.	2
13	**Тема: Потоки даних та обробка великих наборів даних** Потоки даних (Readable, Writable, Transform, Duplex), конвеєри (pipe), backpressure. Асинхронні ітератори. Інкрементальна, ощадлива до пам’яті обробка великих наборів даних.	2
14	**Тема: Патерни проєктування** Породжувальні, структурні та поведінкові патерни: фабрика, компонувальник (Composite), адаптер, проксі (Proxy), декоратор, пул об’єктів (Object Pool), відкритий конструктор (Revealing Constructor). Призначення, структура, приклади застосування у JavaScript та комбінування патернів.	2
15	**Тема: Узагальнене (generic) програмування** Параметричний поліморфізм та параметри типів. Узагальнені структури даних і функції. Обмеження типів (constraints). Застосування узагальнень у системі типів TypeScript.	2
	Всього	**30**

Лабораторні роботи

Кожна лабораторна робота відповідає темі лекційного матеріалу. Метою роботи є засвоєння та опанування теми; наведені завдання є лише прикладами, на яких студент опановує тему, а не самоціллю. Роботи виконуються мовою JavaScript/TypeScript із використанням системи контролю версій Git та публікацією коду під ліцензією MIT. Рекомендованим форматом є інтеграція всіх робіт у спільну кодову базу — єдиний проєкт, що є основою курсової роботи (окремий освітній компонент).

№	Тема лабораторної роботи	Мета та завдання	Год.
1	Генератори та ітератори	Засвоїти та опанувати протокол ітерування, генератори та ліниві послідовності. Приклад завдання: генератор нескінченної послідовності та споживач ітератора з обмеженням за часом.	4
2	Налаштування проєкту та бібліотеки	Засвоїти та опанувати модульну організацію коду, роботу з пакетами та системою контролю версій. Приклад завдання: Git-репозиторій і package.json з ліцензією, поділ коду на бібліотеку та проєкт-приклад з локальними залежностями.	2
3	Мемоізація	Засвоїти та опанувати кешування, мемоізацію та стратегії витіснення кешу. Приклад завдання: мемоізація чистої функції з конфігурованими стратегіями витіснення (LRU, LFU, за часом, власна).	4
4	Черги	Засвоїти та опанувати структури даних «черга», «дек», «пріоритетна черга» та оцінку складності їх операцій. Приклад завдання: двонапрямлена пріоритетна черга з вибором за пріоритетом і за порядком вставки.	4
5	Асинхронне програмування	Засвоїти та опанувати моделі асинхронності (callback, Promise, async/await), обробку помилок і скасування операцій. Приклад завдання: callback- та Promise-версії функції масиву з підтримкою AbortController.	4
6	Реактивне програмування	Засвоїти та опанувати реактивну взаємодію через події (EventEmitter, Observable, патерн «Спостерігач»). Приклад завдання: взаємодія сутностей через повідомлення з підпискою/відпискою та кількома слухачами.	4

№	Тема лабораторної роботи	Мета та завдання	Год.
7	Потоки даних	Засвоїти та опанувати потоки даних та асинхронні ітератори для інкрементальної, ощадливої до пам’яті обробки даних. Приклад завдання: обробка набору даних, що не вміщується в пам’ять.	4
8	Патерни проєктування. Проксі	Засвоїти та опанувати патерн «Проксі»: призначення, структуру, доречність застосування. Приклад завдання: проксі, що додає облікові дані (API Key, JWT) до HTTP-запитів.	2
9	Патерни проєктування. Декоратор	Засвоїти та опанувати патерн «Декоратор» для синхронних і асинхронних функцій. Приклад завдання: декоратор логування з конфігурованими рівнями та структурованим виводом.	2
		Всього	**30**

Практичні заняття

Практичні заняття проводяться у форматі практикуму та код-рев’ю: розбір типових рішень, спільне написання коду, обговорення рішень студентів та інтеграція виконаних лабораторних робіт у єдиний проєкт, який є основою курсової роботи (окремий освітній компонент «Основи програмування. Курсова робота»).

№	Тема практичного заняття та зміст	Год.
1	**Організація проєкту та середовища розробки** Налаштування репозиторію, package.json, лінтера та форматера. Вибір теми та планування структури інтегрованого (курсового) проєкту.	2
2	**Практикум: ітератори, генератори та модульна структура** Розбір рішень та код-рев’ю робіт з ітераторів і генераторів; розділення коду на бібліотеку та приклади використання.	2
3	**Практикум: структури даних та кешування** Реалізація та порівняння черг, деків і кешів; аналіз складності операцій; код-рев’ю робіт з мемоізації та черг.	2
4	**Практикум: функціональні абстракції** Композиція, каррінг, функтори та контейнери Maybe/Either на прикладах із проєктів студентів.	2
5	**Практикум: асинхронне програмування** Перехід від callback до Promise та async/await, обробка помилок і скасування операцій; код-рев’ю робіт з асинхронного програмування.	2

№	Тема практичного заняття та зміст	Год.
6	**Практикум: реактивність, потоки та патерни у проєкті** Код-рев'ю робіт з потоків даних, реактивного програмування та патернів «Проксі» і «Декоратор»; перевірка цілісності інтегрованого проєкту.	2
7	**Модульна контрольна робота (МКР)** Теоретичні питання та практичне завдання з написання коду за темами лекцій 1–13.	2
	Всього	**14**

4. Навчальні ресурси та матеріали

4.1. Базова література

1. Нечай Д.О. Конспект лекцій, приклади коду та методичні вказівки до виконання лабораторних робіт з дисципліни «Основи програмування. Частина 2. Методології програмування» (дистанційний курс на платформі «Сікорський»): https://classroom.google.com/c/ODkzNTMyMDgwMjkw?cjc=morpltuj

2. Мартін Р. Чистий код: створення і рефакторинг за допомогою Agile / пер. з англ. Харків: Фабула, 2019. 448 с.

3. Кормен Т., Лейзерсон Ч., Рівест Р., Штайн К. Вступ до алгоритмів / пер. з англ. Київ: K.I.C., 2019. 1288 с.

4.2. Додаткова література

4. Шемсединов Т.Г., Нечай Д.О. та інші. Приклади коду та проєктів (HowProgrammingWorks): https://github.com/HowProgrammingWorks

5. MDN Web Docs — JavaScript: https://developer.mozilla.org/uk/docs/Web/JavaScript

6. Документація Node.js: https://nodejs.org/uk/docs/

7. Fowler M. Refactoring: Improving the Design of Existing Code. 2nd ed. Boston: Addison-Wesley, 2018. 448 p.

8. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Reading: Addison-Wesley, 1994. 395 p.

9. McConnell S. Code Complete. 2nd ed. Redmond: Microsoft Press, 2004. 960 p.

10. TypeScript Documentation: https://www.typescriptlang.org/docs/

11. Технологічний стек Metarhia: https://github.com/metarhia/

4.3. Інформаційні ресурси

12. Дистанційний курс на платформі дистанційного навчання «Сікорський» в середовищі Google Workspace for Education (навчальні матеріали, завдання, оголошення): https://classroom.google.com/c/ODkzNTMyMDgwMjkw?cjc=morpltuj

13. Відеозаписи лекцій дисципліни на платформі YouTube: https://www.youtube.com/playlist?list=PLkIBctdSkz9D4bIEhdGxU50h__LA4XGlz

5. Методика опанування навчальної дисципліни (освітнього компонента)

Розподіл навчального часу за видами занять і завдань з дисципліни згідно з робочим навчальним планом. На вивчення дисципліни виділено 180 годин / 6 кредитів: лекцій 15 (30

годин), практичних занять 7 (14 годин), лабораторних робіт 9 (30 годин), самостійної роботи 106 годин.

Лекційні заняття

Лекційний матеріал спрямований на формування системного розуміння методологій та парадигм програмування. Половина часу лекційних занять проводиться у форматі практичного семінару (розбір і написання коду наживо). Особливу увагу приділено прикладам коду, що демонструють імплементацію концепцій. Відеозаписи лекцій розміщуються на дистанційному курсі.

Деталізація тем лекційних занять, конспект лекцій та відеоматеріали розміщено на платформі дистанційного навчання «Сікорський»: https://classroom.google.com/c/ODkzNTMyMDgwMjkw?cjc=morpltuj

Практичні заняття

Практичні заняття проводяться у форматі практикуму та код-рев'ю: розбір типових рішень, спільне написання коду та інтеграція виконаних лабораторних робіт у єдиний проєкт. На останньому практичному занятті проводиться модульна контрольна робота.

Лабораторні роботи

Метою лабораторних робіт є формування навичок реалізації вивчених концепцій у програмному коді. Рекомендованим форматом є інтеграція всіх лабораторних робіт у єдиний проєкт, що дозволяє одночасно виконувати вимоги лабораторних робіт і накопичувати практичну основу курсової роботи.

Перевага надається методам, що розвивають критичне мислення та самостійність, з акцентом на розумінні коду, який студент подає до захисту. Комунікативний підхід реалізується через код-рев'ю, обговорення рішень та пір-рев'ю коду колег.

6. Самостійна робота здобувача вищої освіти

Види самостійної роботи (106 годин):

– підготовка до лекційних занять та опрацювання матеріалів лекцій (0,5 год × 30 год лекцій = 15 годин);

– підготовка до практичних занять (0,5 год × 14 год практичних занять = 7 годин);

– виконання лабораторних робіт, їх інтеграція у єдиний проєкт та підготовка до захисту (1 год × 30 год лабораторних робіт = 30 годин);

– самостійне опрацювання тем, наведених у таблиці нижче (22 години);

– підготовка до модульної контрольної роботи (2 години);

– підготовка до екзамену (30 годин).

Теми для самостійного опрацювання

Теми для самостійного опрацювання (22 години) розподілено відповідно до розділів лекційного матеріалу:

№ з/п	Розділ / тема, що виноситься на самостійне опрацювання	Год. СРС
	Розділ 1. Вступ. Ітератори та генератори	
1	Порівняння імперативного, об'єктно-орієнтованого та функціонального підходів на прикладах коду	2
2	Генератори як співпрограми: передача значень у генератор, методи return і throw	2
	Розділ 2. Інструменти розробки та засоби платформи	

№ з/п	Розділ / тема, що виноситься на самостійне опрацювання	Год. СРС
3	Публікація пакета в npm-реєстрі, lock-файли та аудит залежностей	2
4	Робота з гілками Git, злиття та rebase, розв’язання конфліктів	2
	Розділ 3. Кешування та структури даних	
5	Зв’язний список, кільцевий буфер, пріоритетна черга на основі купи	2
6	Порівняльна оцінка стратегій витіснення кешу	2
	Розділ 4. Функціональне програмування	
7	Замикання та їх використання для інкапсуляції стану	2
	Розділ 5. Асинхронне та реактивне програмування	
8	Цикл подій (event loop), мікро- та макрозадачі; комбінатори промісів	2
9	Потоки Node.js та керування backpressure	2
	Розділ 6. Патерни проєктування та узагальнене програмування	
10	Патерни «Стратегія» та «Фасад»; засоби метапрограмування Proxy та Reflect	2
11	Узагальнені та утилітні типи TypeScript	2
Разом		**22**

Матеріали для самостійного опрацювання, питання для самоконтролю та приклади коду розміщено на платформі дистанційного навчання «Сікорський»: https://classroom.google.com/c/ODkzNTMyMDgwMjkw?cjc=morpltuj

Політика та контроль

7. Політика навчальної дисципліни (освітнього компонента)

Студент має вивчати дисципліну протягом семестру, дотримуючись календарного плану виконання лабораторних робіт та вивчення тем лекційного матеріалу. Студент виконує завдання самостійно; у груповій роботі кожен студент самостійно виконує та захищає власну частину.

Дедлайни. Для виконання лабораторних робіт встановлюються дедлайни, які уточнюються та оприлюднюються викладачем на початку семестру. Штрафні бали за невчасне виконання завдань не передбачаються; заохочувальні бали за вчасне виконання робіт не нараховуються.

Лабораторні роботи. Робота вважається зарахованою, якщо студент захистив її у викладача (показав працездатність, пояснив рішення, відповів на питання за темою роботи, за потреби вніс невеликі зміни під час захисту) та опублікував результат у Git-репозиторії. Уся робота ведеться у Git з історією комітів, що відображає реальний процес виконання; у груповій роботі кожен студент комітить власну частину від свого імені та захищає її окремо. Допускається асинхронна здача через надсилання посилання на репозиторій з подальшим захистом. Бали за кожну роботу та критерії оцінювання наведено в розділі 8.

Модульна контрольна робота. Модульна контрольна робота проводиться у письмовій формі на останньому практичному занятті за темами лекцій 1–13. Студент, який пропустив МКР з поважної причини, виконує її у термін, погоджений з викладачем.

Екзамен. Екзамен проводиться у письмовій формі з подальшою співбесідою за написаним; екзаменаційна робота виконується без застосування допоміжних засобів (мобільні телефони, планшети, засоби штучного інтелекту тощо). Бали за екзамен додаються до стартового

рейтингу. Умовою допуску до екзамену є зарахування всіх лабораторних робіт та стартовий рейтинг R_C не менше 36 балів.

Правила відвідування занять. Відвідування занять є вільним, бали за присутність на заняттях не додаються, відповідно штрафні бали не передбачаються.

Дистанційний режим навчання. У разі запровадження обмежень на відвідування університету, пов'язаних з введенням карантину або режиму воєнного стану в державі, освітній процес здійснюється у дистанційному режимі відповідно до Положення про дистанційне навчання в КПІ ім. Ігоря Сікорського (https://osvita.kpi.ua/index.php/node/188), Регламенту організації освітнього процесу в дистанційному режимі (https://profkom.kpi.ua/reglament-organizatsiyi-osvitnogo-protsesu-v-distantsiynomu-rezhimi) та Регламенту проведення семестрового контролю в дистанційному режимі (https://osvita.kpi.ua/node/148). Дистанційний курс дисципліни розміщено на платформі дистанційного навчання «Сікорський»; результати оцінювання висвітлюються в АС «Електронний кампус» (https://ecampus.kpi.ua).

Правила поведінки на заняттях. На заняттях слід дотримуватись норм етичної поведінки, визначених у Кодексі честі КПІ ім. Ігоря Сікорського (https://kpi.ua/code). На території університету здобувачі мають поводити себе відповідно до Правил внутрішнього розпорядку (https://kpi.ua/admin-rule).

Визнання результатів навчання, набутих у неформальній/інформальній освіті. Визнання регламентується Положенням про визнання результатів навчання, набутих у неформальній / інформальній освіті (https://osvita.kpi.ua/index.php/node/179):

– у разі навчання в неформальній освіті, яку студент обрав самостійно і хоче зарахувати результати навчання в рамках дисципліни, проходиться процедура валідації, що передбачає подання здобувачем заяви на ім'я декана та підтверджувальних документів; рішення про визнання чи невизнання приймає комісія у складі завідувача кафедри, викладача та гаранта освітньо-професійної програми;

– за рішенням викладача окремі теми дисципліни можуть бути зараховані на підставі результатів проходження здобувачем онлайн-курсу або іншого елементу неформальної освіти; перелік таких елементів визначається викладачем заздалегідь і розміщується на платформі дистанційного навчання «Сікорський», про що студенти повідомляються на початку семестру; у цьому випадку здобувач звільняється від виконання відповідних завдань і отримує максимальний бал згідно з РСО.

Політика використання штучного інтелекту. Використання штучного інтелекту регламентується «Політикою використання штучного інтелекту для академічної діяльності в КПІ ім. Ігоря Сікорського» (https://osvita.kpi.ua/node/1225). Усі завдання мають бути результатом власної оригінальної роботи здобувача. Допускається використання ШІ як допоміжного інструмента за умови, що студент повністю розуміє кожен рядок коду та кожне прийняте рішення і здатен їх пояснити та змінити під час захисту; автоматична генерація відповідей без подальшого аналізу та доопрацювання заборонена.

Порушення академічної доброчесності: використання плагіату. У разі виявлення фактів використання плагіату під час виконання навчальних завдань студент втрачає право на подальше виконання поточних завдань у межах навчальної дисципліни до з'ясування обставин порушення. Студент зобов'язаний аргументовано пояснити причини використання плагіату, засвідчити готовність дотримуватись принципів академічної доброчесності

(письмово, у вигляді пояснювальної записки) та повторно виконати завдання або, за потреби, виконати додаткове завдання відповідно до вимог щодо самостійності. Зазначені дії регламентуються Кодексом честі КПІ ім. Ігоря Сікорського (https://kpi.ua/code).

8. Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Рейтинг студента з дисципліни розраховується за 100-бальною шкалою і складається зі стартового рейтингу R_C (до 60 балів) та екзаменаційної складової R_E (40 балів): $R = R_C + R_E$. Стартовий рейтинг $R_C = R_{ЛР} + R_{МКР}$ формується протягом семестру за виконання та захист 9 лабораторних робіт ($R_{ЛР}$, до 45 балів) і модульну контрольну роботу ($R_{МКР}$, до 15 балів).

Таблиця 1. Оцінювання лабораторних робіт

№	Тема роботи	Мін. балів для зарахування	Макс. балів
1	Генератори та ітератори	3	5
2	Налаштування проєкту та бібліотеки	3	5
3	Мемоізація	3	5
4	Черги	3	5
5	Асинхронне програмування	3	5
6	Реактивне програмування	3	5
7	Потоки даних	3	5
8	Патерни проєктування. Проксі	3	5
9	Патерни проєктування. Декоратор	3	5
Всього за роботи			**45**

Кожна лабораторна робота оцінюється у 5 балів за наведеними нижче критеріями і вважається зарахованою за умови отримання не менше 3 балів. Бали за терміни здачі не знижуються; незарахована робота може бути доопрацьована та захищена повторно. Модульна контрольна робота проводиться на останньому практичному занятті та оцінюється у 15 балів.

Критерії оцінювання лабораторної роботи (5 балів)

Критерій	Опис	Бали
Виконання завдання (до 2 балів)	Реалізація повністю працює, відповідає завданню та коректно застосовує концепції теми роботи	2
	Реалізація працює частково або концепції теми застосовано з помилками	1
	Реалізація не працює або не відповідає завданню	0
Якість коду та репозиторій (до 1 бала)	Код структурований і читабельний; репозиторій має логічну структуру та історію комітів, що відображає процес виконання	1
	Код погано структурований або історія комітів відсутня чи формальна	0
Захист (до 2 балів)	Студент вільно пояснює код і прийняті рішення, відповідає на питання за темою лекції, за потреби вносить невеликі зміни під час захисту	2
	Пояснення неповні, на окремі питання студент відповідає з помилками	1
	Студент не може пояснити власний код	0

Критерії оцінювання модульної контрольної роботи (15 балів)

МКР складається з теоретичних питань (5 балів) та практичного завдання з написання коду (10 балів).

Критерій	Опис	Бали
Теоретичні питання	Відповіді правильні та повні, поняття пояснено з прикладами	5
	Відповіді в цілому правильні, з незначними неточностями або неповні	3–4
	Відповіді містять суттєві помилки або розкривають питання частково	1–2
Практичне завдання	Код працює правильно, відповідає умові, читабельний, рішення обґрунтоване	9–10
	Код працює, але має незначні недоліки (граничні випадки, стиль, обробка помилок)	7–8
	Код працює частково або містить помилки, підхід у цілому правильний	4–6
	Наведено лише фрагменти розв’язання або обрано неправильний підхід	1–3
Відсутність відповіді	Відповідь відсутня або не відповідає завданню	0

Критерії оцінювання екзамену (40 балів)

Екзаменаційний білет містить два теоретичні питання (по 10 балів) та практичне завдання з написання коду (20 балів).

Критерій	Опис	Бали
Теоретичне питання (кожне)	Повна, правильна та логічно викладена відповідь, поняття пояснено з прикладами коду	9–10
	Правильна відповідь з неповними поясненнями або незначними неточностями	7–8
	Відповідь містить помилки, питання розкрито частково	5–6
	Фрагментарна відповідь із суттєвими помилками	1–4
Практичне завдання	Код працює правильно, відповідає умові, рішення обґрунтоване, студент пояснює кожен рядок під час співбесіди	18–20
	Код працює, але має незначні недоліки або пояснення неповні	14–17
	Код працює частково або містить помилки, підхід у цілому правильний	10–13
	Наведено лише фрагменти розв’язання або обрано неправильний підхід	1–9
Відсутність відповіді	Відповідь відсутня або не відповідає завданню	0

Таблиця 2. Зведена таблиця оцінювання

Вид контролю	Макс. балів
Лабораторні роботи ($R_{ЛР}$)	45
Модульна контрольна робота ($R_{МКР}$)	15
Стартовий рейтинг R_C	**60**
Екзамен (R_E)	40
Всього	**100**

Умовою допуску до екзамену є зарахування всіх лабораторних робіт та стартовий рейтинг R_C не менше 36 балів (60% від 60).

Максимальна кількість балів за екзамен $R_E = 40$. Підсумковий рейтинг $R = R_C + R_E$ не перевищує 100 балів.

Курсова робота не входить до рейтингу дисципліни: вона є окремим освітнім компонентом «Основи програмування. Курсова робота» (ПО 03, 1 кредит ЄКТС) з власним силабусом і оцінюванням. Практичною основою курсової роботи є інтегрований проєкт, у який студент послідовно вбудовує виконані лабораторні роботи.

Календарний контроль

Календарна атестація студентів (на 8 та 14 тижнях семестру) проводиться за значенням поточного рейтингу студента на час атестації. Якщо значення цього рейтингу не менше 50 % від максимально можливого на час атестації, студент вважається атестованим; в іншому випадку в атестаційній відомості виставляється «неатестовано».

Семестровий контроль

Семестровий контроль результатів навчання проводиться у вигляді екзамену. Підсумковий рейтинг визначається як $R = R_C + R_E$ і переводиться в оцінку за університетською шкалою згідно з таблицею. Студенти, які не виконали умов допуску, вважаються не допущеними до екзамену.

Таблиця 3. Відповідність рейтингових балів оцінкам

Кількість балів	Оцінка
100–95	Відмінно
94–85	Дуже добре
84–75	Добре
74–65	Задовільно
64–60	Достатньо
Менше 60	Незадовільно
Не виконані умови допуску	Не допущено

Робочу програму навчальної дисципліни (силабус):

Складено: ас. кафедри ОТ Нечай Д.О.

Ухвалено: кафедрою ОТ (протокол №18 від 24.06.2026)

Погоджено: методичною комісією ФІОТ (протокол №12 від 26.06.2026)