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

Емблема кафедри (за наявності)

Кафедра інженерії програмного забезпечення в енергетиці

Новітні мови програмування

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни

Рівень вищої освіти	***Перший***
Галузь знань	*12 Інформаційні технології*
Спеціальність	*121 Інженерія програмного забезпечення*
Освітня програма	*Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці*
Статус дисципліни	*Вибіркова*
Форма навчання	*очна(денна),заочна*
Рік підготовки, семестр	*4 курс, осінній*
Обсяг дисципліни	*4 кредити/120 год. (Лекції: 36 год., комп. прак.: 18 год., СРС: 66 год.)*
Семестровий контроль/ контрольні заходи	*Залік*
Розклад занять	*Згідно розкладу на осінній семестр поточного навчального року (rozklad.kpi.ua)*
Мова викладання	*Українська*
Інформація про керівника курсу / викладачів	Лектор: *Пироговська Тетяна Володимирівна* Практичні: Пироговська Тетяна Володимирівна
Розміщення курсу	Google classroom, Е- кампус, платформа дистанційного навчання «Сікорський»

Програма навчальної дисципліни

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

Чому майбутньому фахівцю варто вчити саме цю дисципліну?

Вивчення дисципліни "Новітні мови програмування" є необхідним компонентом професійної підготовки сучасного фахівця в галузі комп'ютерних наук та інформаційних технологій. У контексті стрімкого розвитку програмного забезпечення та зростання складності інформаційних систем, знання лише класичних мов програмування вже не гарантує конкурентоспроможності на ринку праці. Новітні мови створюються як реакція на обмеження попередників і пропонують інноваційні підходи до керування пам'яттю, паралелізму, типізації та безпеки. Опанування таких мов дозволяє студенту осягнути актуальні тенденції в інженерії програмного забезпечення та критично оцінювати еволюцію мовних конструкцій. Це не лише розширює технічний кругозір, але й формує здатність швидко адаптуватися до нових парадигм у розробці. Дисципліна сприяє розвитку абстрактного мислення, аналітичного підходу та глибшого розуміння внутрішніх механізмів сучасних мов. Крім того, практична робота з новітніми мовами дозволяє застосовувати принципи безпечного кодування, що особливо важливо в умовах кіберзагроз. Такий досвід стає основою для участі в інноваційних проєктах і міждисциплінарних дослідженнях. У результаті студент формує не лише набір навичок, а й гнучкість до змін, що є критичною в умовах технологічної невизначеності.

Опанування новітніх мов програмування також сприяє формуванню майбутнього фахівця як інженера, здатного мислити не інструментально, а концептуально. На відміну від традиційного підходу, який орієнтується на реалізацію вже відомих алгоритмів, ця дисципліна

заохочує до пошуку нових рішень, які враховують як технічні, так і архітектурні обмеження. Це особливо важливо в умовах переходу до хмарних технологій, багатоядерних процесорів та розподілених обчислень. Завдяки вивченню таких мов як Rust, Zig, Julia або Elm, студенти ознайомлюються з сучасними підходами до продуктивності, надійності та масштабованості програм. Крім того, дисципліна формує вміння аналізувати доцільність вибору мовних інструментів для конкретних задач, що є ознакою професійної зрілості. Практика міжмовного порівняння розвиває стратегічне бачення та критичне ставлення до інженерних рішень. Знання новітніх мов стає не лише додатковим технічним активом, але й основою для міжплатформеного та мультипарадигмального мислення. Такий підхід особливо актуальний в умовах швидкої еволюції IT-галузі, де нові інструменти з'являються щороку. Отже, дисципліна має як прикладне, так і концептуальне значення у формуванні фахівця майбутнього.

Мета дисципліни. *Ознайомити студентів з сучасними мовами програмування на прикладі мов Rust, Julia та Zig.*

Предмет дисципліни. *Огляд мов Rust, Julia та Zig, включаючи:*

- *володіння та запозичення,*
- *тривалість життя та риси;*
- *гарантована безпека програм;*
- *тестування, оброблення помилок та ефективний рефакторинг;*
- *розумні вказівники, багатопотоковість та зіставлення зі зразком;*

Очікувані результати навчання.

Фахові компетентності.

ФК 16. Володіти скриптовими та декларативними мовами програмування.

Програмні результати навчання.

ПРН 15. Мотивовано обирати мови програмування та технології розробки для розв'язання завдань створення і супроводження програмного забезпечення.

ПРН 29. Вміти створювати інтерактивні, компактні Веб-застосунки та Веб-системи, володіти методичними основами та технологіями створення інформаційних систем та мережевого програмного забезпечення з врахуванням специфіки предметної області енергетичної галузі.

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

Дисципліна вивчається у сьомому семестрі. Пререквізитами є курси "Комп'ютерна дискретна математика", "Основи програмування" та "Архітектура системного програмного забезпечення", базовий рівень володіння англійською мовою не нижче А2 Постреквізитів у даного курсу на бакалаврському рівні немає.

3. Зміст навчальної дисципліни

Розділ 1. Вступ

Тема 1.1 Еволюція мов програмування, класифікація мов

Тема 1.2. Огляд сучасних вузько направлених мов програмування

Тема 1.3. Новітні мови системного рівня

Розділ 2. Мова програмування Rust

Тема 2.1. Типи, змінні, контроль пам'яті (ownership, borrowing)

Тема 2.2. Функції, структури, енумерації, матчінг

Тема 2.3. Модулі, Crates, підключення бібліотек

Тема 2.4. Обробка помилок, Result, Option, паніка

Тема 2.5. Асинхронне програмування в Rust (async/await)

Розділ 3. Мова програмування Julia

Тема 3.1. Синтаксис, змінні, типізація

Тема 3.2. Масиви, цикли, функції.

Тема 3.3. Продуктивність і компіляція

Тема 3.4. Використання Julia у Data Science та науці

Розділ 4. Мова програмування Zig

Тема 4.1. Синтаксис, компіляція, змінні

Тема 4.2. Пам'ять, manual allocation, safety

Тема 4.3. Управління помилками

Тема 4.4. Структури, модулі та інтерфейси

Розділ 5. Аналітика та сучасна мовна екосистема

Тема 5.1. Мови майбутнього: які ідеї ще не реалізовані, дослідження мовних моделей

Тема 5.2. Доцільність створення нових мов програмування.

4. Навчальні матеріали та ресурси

Базова література:

*1.Klabnik, S. & Nichols, C. (2021). The Rust Programming Language. 2-ге вид. Офіційний та найповніший підручник по Rust. Доступно онлайн: **https://doc.rust-lang.org/book/***

*2.Lauwens, B. & Dejonghe, B. (2019). Think Julia: How to Think Like a Computer Scientist. Навчальний посібник з програмування на Julia для початківців. Онлайн-версія: **https://benlauwens.github.io/ThinkJulia.jl/latest/book/***

*3. Duarte Faria, P. (2022). Introduction to Zig: a project-based book. Офіційна книга з глибоким зануренням в Zig. Доступно онлайн: **https://ziglang.org/documentation/master/introduction/***

Додаткова література:

*1. Rust Project Contributors. Rust By Example. Інтерактивна книга з прикладами коду, доповнює "The Book". Доступно онлайн: **https://doc.rust-lang.org/rust-by-example/***

*2. Julia Project Contributors. The Julia Language. Офіційний довідник та документація мови Julia. Доступно онлайн: **https://docs.julialang.org/***

*3. Zig Community. The Zig Book. Повний довідник по мові та стандартній бібліотеці. Онлайн-версія: **https://ziglang.org/learn/***

4. Isaacson, W. (2014). The Innovators: How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution. Simon & Schuster.

5. Friedman, D. P., Wand, M. & Haynes, C. T. (2008). Essentials of Programming Languages (3-ге вид.). MIT Press.

6. Chowdhary, K. R. (2020). "On the Evolution of Programming Languages.".

7. Crafa, S. (2015). "Modelling the Evolution of Programming Languages."

8. Harper, R. (2016). Practical Foundations for Programming Languages. Cambridge University Press.

9. Felleisen, M. et al. (2022). Structure and Interpretation of Computer Programs (JavaScript ed.). MIT Press.

Бажано зазначати не більше п'яти базових джерел, які є вільно доступними, та не більше 20 додаткових.

Навчальний контент

5. Методика опанування навчальної дисципліни (освітнього компонента)

№ з/п	Тип навчального заняття	Опис навчального заняття
	Розділ 1. Вступ	
1	*Лекція 1. Еволюція мов програмування, класифікація мов*	*Історія розвитку мов програмування, класифікація мов(імперативні, функціональні, об'єктно-орієнтовані, логічні), передумови виникнення нових мов програмування.* *СРС: Ознайомитись з класифікацією мов програмування.*
2	*Лекція 2. Огляд сучасних вузько направлених мов програмування*	*Nim: Python-подібна мова з компіляцією в C — історія, переваги, застосування; Gleam: Erlang з сучасним синтаксисом, Bicep: інфраструктура як код (IaC) для Azure; Wit & WITX, WebAssembly-мови: нові формати опису інтерфейсів для вбудованих рішень; порівняння вузькоспеціалізованих мов за цілями та ефективністю.* *СРС: Ознайомитись з можливостями мов програмування Nim, Gleam, Bicep, Wit & WITX, проаналізувати причини їх створення.*
3	*Комп'ютерний практикум 1. Представлення новітньої мови програмування.*	*Обрати мову програмування, створену після 2015 року. Проаналізувати передумови її виникнення, переваги та недоліки. Результати аналізу представити у вигляді звіту.*
3	*Лекція 3. Новітні мови системного рівня*	*Carbon (Google): історія появи, порівняння з C++, Swift, Rust; переваги та концептуальні новації; Vale: історія, фокус на безпечну пам'ять через контроль потоків та ownership, Vale vs Rust: інші підходи до memory safety, експериментальні особливості; Bosque (Microsoft): філософія детермінованого коду, виключення мутацій.* *СРС: Ознайомитись з можливостями мов програмування Carbon та Vale*
4	*Комп'ютерний практикум 2. Представлення новітньої мови програмування.*	*Презентація проведених досліджень обраної новітньої мови програмування у вигляді виступу.*
	Розділ 2. Мова програмування Rust	
5	*Лекція 4. Типи, змінні, контроль пам'яті (ownership, borrowing)*	*Опис система типів, оголошення змінних та ключові особливості управління пам'яттю — механізми володіння (ownership) та запозичення (borrowing), опис Rust реалізації безпечного використання ресурсів без втрат продуктивності, демонстрація прикладів*

		помилок компіляції, пов'язаних з неправильним життєвим циклом змінних. *СРС: Ознайомитись з матеріалами, розібрати представлені приклади.*
6	*Лекція 5. Функції, структури, енумерації, матчінг*	*Створення функцій, передача параметрів та використання повертаних значень. Огляд структурованих типів даних — структури (struct), переліки (enum) та механізмів шаблонного порівняння match, безпечне опрацювання варіантів і розгалужень.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
7	*Лекція 6. Модулі, Crates, підключення бібліотек*	*Основи модульності у Rust — як створювати модулі, організовувати структуру проєкту. Ознайомлення з системою пакетів Cargo, підключенням зовнішніх бібліотек (crates) та керуванням залежностями. Побудова великих програм з використанням відкритих бібліотек.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
8	*Комп'ютерний практикум 3. Rust на практичних прикладах*	*Ілюструються приклади розглянутих особливостей мови на прикладі стандартних задач.* *СРС: Вирішити завдання за допомогою інструментів мови Rust.*
9	*Лекція 7. Обробка помилок, Result, Option, паніка*	*Ознайомлення з механізмами обробки помилок у Rust без винятків. Типи Result, Option, оператор ?, макроси panic! І способи уникнення фатальних помилок.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
10	*Лекція 8. Асинхронне програмування в Rust (async/await)*	*Реалізація асинхронності у Rust: async, await та Future-и. Робота екзек'юторів, неблокуючі операції та переваги асинхронного підходу. Приклади з обробкою файлів або HTTP-запитів.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
11	*Комп'ютерний практикум 4. Порівняльний аналіз мови Rust з її попередниками.*	*Реалізувати надані завдання мовами Rust та C++ або Java, порівнюючи синтаксис, контроль пам'яті, безпеку та продуктивність. Результати аналізу представити у формі звіту. Мета заняття — відчути відмінність парадигм і можливості Rust в контексті традиційних мов.*
Розділ 3. Мова програмування Julia		
12	*Лекція 9. Синтаксис, змінні, типізація*	*Базовий синтаксис Julia, включно з правилами оголошення змінних та особливостями типізації.*

		Багатопарадигмності мови, включно з динамічною і статичною типізацією. Базові вирази і оператори. *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
13	*Лекція 10. Масиви, цикли, функції.*	*Робота з масивами, обчислювальними циклами та функціями. Аналіз ефективних способів ітерації та векторизації. Функції вищого порядку, замикання та передачу параметрів.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
14	*Комп'ютерний практикум 5. Julia на практичних прикладах*	*Ілюструються приклади розглянутих особливостей мови на прикладі стандартних задач.* *СРС: Вирішити завдання за допомогою інструментів мови Julia.*
15	*Лекція 11. Продуктивність і компіляція*	*Огляд механізму JIT-компіляції у Julia та його впливу на продуктивність. Пояснення принципів оптимізації коду, аналітики типів та спеціалізації функцій. Інструменти профілювання продуктивності.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
16	*Лекція 12. Використання Julia у Data Science та науці*	*Поглиблене вивчення застосування Julia в аналізі даних, машинному навчанні та візуалізації. Огляд популярних бібліотек (DataFrames.jl, Flux.jl, Plots.jl), побудова моделей. Практичне використання в аналітиці.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
17	*Комп'ютерний практикум 6. Порівняльний аналіз мови Julia з її попередниками.*	*Реалізація математичної задачі мовами Julia та Python/Matlab із подальшим аналізом швидкодії та зручності синтаксису. Результати аналізу представити у формі звіту. Мета – навчитись обґрунтовувати вибір мови залежно від обчислювальних потреб..*
Розділ 4. Мова програмування Zig		
18	*Лекція 13. Синтаксис, компіляція, змінні*	*Опис синтаксису Zig, структура програм, робота з компілятором Zig. Особливості оголошення змінних, відмінності між const, var, comptime. Демонстрація створення найпростіших утилітт.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
19	*Лекція 14. Пам'ять, manual allocation, safety*	*Управління пам'яттю вручну, механізми безпеки (наприклад, bounds checking). Взаємодія з malloc, аллокаторами та робота без garbage collector. Правила безпечного і контрольованого коду.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*

20	*Комп'ютерний практикум 7. Zig на практичних прикладах*	*Ілюструються приклади розглянутих особливостей мови на прикладі стандартних задач.* *СРС: Вирішити завдання за допомогою інструментів мови Zig.*
21	*Лекція 15. Управління помилками*	*Огляд унікальної моделі обробки помилок у Zig, зокрема через error union, try, catch, defer. Пояснення відмови від класичних винятків задля передбачуваності. Приклади написання кодів з надійною обробкою несподіваних ситуацій.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
22	*Лекція 16. Структури, модулі та інтерфейси*	*Оголошення структур та модулів у Zig, побудова інтерфейсів за допомогою гнучкої типізації. Організація коду на проєктному рівні. Використання композиції замість спадкування.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
23	*Комп'ютерний практикум 8. Порівняльний аналіз мови Zig.*	*Реалізація задач на Zig та її аналогу на C. Порівняти код, ефективність, зручність управління пам'яттю та повідомлення компілятора. Оцінити практичність Zig у реальних умовах. Результати аналізу представити у формі звіту.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
Розділ 5. Аналітика та сучасна мовна екосистема		
24	*Лекція 17. Мови майбутнього: які ідеї ще не реалізовані, дослідження мовних моделей*	*Огляд інноваційних ідей у програмуванні, які поки що залишаються концепціями або експериментами: наприклад, мови з формально виведеними гарантіями, нейромережеві DSL, візуальні або природномовні інтерфейси. Вплив дослідженнь у сфері мовних моделей (наприклад, GPT, Codex) на розробку мов нового покоління. Питання автоматичної генерації коду, синтаксичної мінімізації та інтеграції компіляторів з AI-системами.* *СРС: Ознайомитись з матеріалами лекції, розібрати представлені приклади.*
25	*Лекція 18. Підсумкова лекція: Чи варто створювати ще одну мову програмування?*	*Аналіз проблеми надлишкового розмаїття мов програмування. Об'єктивні причини створення нових мов, критерії їх доцільності та типові сценарії невдач. Кожен студент має змогу висловити власну точку зору, спираючись на здобуті знання — лекція підсумовує курс і стимулює критичне мислення щодо майбутнього програмування.* *СРС: Підготувати аналіз проблематики створення мов програмування.*

26	*Комп'ютерний практикум 9. Обговорення сучасних тенденцій інформаційних технологій*	*Дискусія «Проблематики створення мов програмування.»*

6. Самостійна робота здобувача вищої освіти

Дисципліна «Новітні мови програмування» ґрунтується на самостійній підготовці до аудиторних занять на теоретичні та практичні теми

№ з/п	*Назва теми, що виноситься на самостійне опрацювання*	*Кількість годин*
1	*Підготовка до лекції 1*	*2*
2	*Підготовка до лекції 2*	*2*
3	*Підготовка до лекції 3*	*2*
4	*Підготовка до лекції 4*	*2*
5	*Підготовка до лекції 5*	*2*
6	*Підготовка до лекції 6*	*2*
7	*Підготовка до лекції 7*	*2*
8	*Підготовка до лекції 8*	*2*
9	*Підготовка до лекції 9*	*2*
10	*Підготовка до лекції 10*	*2*
11	*Підготовка до лекції 11*	*2*
12	*Підготовка до лекції 12*	*2*
13	*Підготовка до лекції 13*	*2*
14	*Підготовка до лекції 14*	*2*
15	*Підготовка до лекції 15*	*2*
16	*Підготовка до лекції 16*	*2*
17	*Підготовка до лекції 17*	*2*
18	*Підготовка до лекції 18*	*2*
19	*Підготовка до комп'ютерного практикуму 1*	*4,5*
20	*Підготовка до комп'ютерного практикуму 2*	*4,5*
21	*Підготовка до комп'ютерного практикуму 3*	*4,5*
22	*Підготовка до комп'ютерного практикуму 4*	*4,5*
23	*Підготовка до комп'ютерного практикуму 5*	*4,5*
24	*Підготовка до комп'ютерного практикуму 6*	*4,5*
25	*Підготовка до комп'ютерного практикуму 7*	*4,5*
26	*Підготовка до комп'ютерного практикуму 8*	*4,5*
27	*Підготовка до комп'ютерного практикуму 9*	*4,5*

29	*Підготовка до заліку*	*8*

Політика та контроль

7. Політика навчальної дисципліни (освітнього компонента)

Студенти отримують бали за правильне та вчасне виконання комп'ютерних практикумів. На кожній лекції студенти можуть отримати бали за Студент повинен здати правильно виконаний комп'ютерний практикум протягом трьох тижнів з дня видачі завдання для отримання повної кількості балів, в іншому випадку застосовуються штрафні бали не більше 40% від загальної кількості за лабораторну роботу.

У разі виявлення недотримання студентом академічної доброчесності, робота студента анулюється. Для отримання балів за роботу студент має виконати додаткове завдання.

Протягом семестру на довільних лекціях відбувається бліц-опитування за темами минулих лекцій. Максимальна кількість балів за опитування, яку можна отримати протягом семестру: 8 балів.

На першій атестації (8-й тиждень) студент отримує «зараховано», якщо його поточний рейтинг не менше 40 % від максимальної кількості балів, яку може отримати студент до першої атестації.

На другій атестації (14-й тиждень) студент отримує «зараховано», якщо його поточний рейтинг не менше 60 % від максимальної кількості балів, яку може отримати студент до другої атестації.

Семестровий рейтинг (кількість балів) складається з: 1) Комп'ютерних практикумів (у формі практичних завдань з програмування) 52%, 2) проходження бліц опитування на лекціях 8% 3) заліку 40%.

За семестрового рейтингу не менше 60 балів та зарахуванні усіх робіт комп'ютерного практикуму студент отримує залік «автоматом» відповідно до таблиці (Таблиця відповідності рейтингових балів оцінкам за університетською шкалою). В іншому разі або коли студент хоче підвищити свої оцінку студент має скласти залік.

Необхідною умовою допуску до заліку є виконання і захист всіх комп'ютерних практикумів.

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

Елементи контролю	*Бали оцінювання*
Експрес опитування на лекціях	*6*
Комп'ютерний практикум 1	*6*
Комп'ютерний практикум 2	*6*
Комп'ютерний практикум 3	*6*
Комп'ютерний практикум 4	*6*
Комп'ютерний практикум 5	*6*
Комп'ютерний практикум 6	*6*
Комп'ютерний практикум 7	*6*
Комп'ютерний практикум 8	*6*
Комп'ютерний практикум 9	*6*
Залік	*40*

Поточний контроль: експрес-опитування

Календарний контроль: провадиться двічі на семестр як моніторинг поточного стану виконання вимог силабусу.

Семестровий контроль: залік

Умови допуску до семестрового контролю: зарахування усіх комп'ютерних практикумів, семестровий рейтинг менше 40 балів.

Таблиця відповідності рейтингових балів оцінкам за університетською шкалою:

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

9. Додаткова інформація з дисципліни (освітнього компонента)

- *перелік питань, які виносяться на семестровий контроль (наприклад, як додаток до силабусу);*

Робочу програму навчальної дисципліни (силабус):

Складено асистентом кафедри ІПЗЕ, Пироговською Тетяною Володимирівною

Ухвалено кафедрою _____ (протокол № ___ від _____)

Погоджено Методичною комісією факультету[1] (протокол № ___ від _____)