

	Силабус навчальної дисципліни <u>«Конструювання програмного забезпечення»</u> Освітньо-професійна програма <u>«Інженерія програмного забезпечення»</u> Спеціальність <u>F2 «Інженерія програмного забезпечення»</u> Галузь знань <u>F «Інформаційні технології»</u>
Рівень освіти	Фахова передвища освіта
Освітньо-професійний/освітній ступінь	Фаховий молодший бакалавр
Статус навчальної дисципліни	Нормативна
Семестр	на базі БЗСО – 4 семестр, на базі ПЗСО – 6 семестр
Обсяг дисципліни (кредити ЄКТС/загальна кількість годин)	<u>4 кредити ЄКТС / 120 годин</u>
Мова викладання	Українська
Оригінальність навчальної дисципліни	Однією з головних концепцій дисципліни «Конструювання програмного забезпечення» є наскрізне створення програмного продукту: від виявлення проблеми та формулювання вимог до моделювання, вибору архітектури, конструювання компонентів, перевірки якості, командної доставки й подальшого супроводу. Здобувачі освіти не виконують набір ізольованих вправ, а послідовно розвивають власний навчальний проєкт і на кожному етапі створюють взаємопов'язані інженерні артефакти. У межах дисципліни поєднуються: - аналіз і моделювання: визначення стейкхолдерів, функціональних і нефункціональних вимог, критеріїв приймання та меж системи; створення UML-діаграм прецедентів, діяльності, станів, класів, послідовностей і компонентів; - порівняння C# і C++ як інструментів для різних класів задач з урахуванням типобезпечності, керування ресурсами, продуктивності, переносності та швидкості розробки; - вибір архітектурного стилю відповідно до вимог до якості, оцінювання компромісів, ризиків і вартості рішень, документування рішень за допомогою ADR; - доставка й еволюція: автоматизована збірка та CI, версіонування, журнал змін, логування, діагностика, підготовка релізу, аналіз дефектів і ретроспектива.
Мета навчальної дисципліни	Метою навчальної дисципліни є формування здатності системно конструювати програмне забезпечення протягом усього життєвого циклу: аналізувати проблему та вимоги, моделювати програмну систему засобами UML, обирати відповідні парадигми, мови й архітектурні рішення, створювати модульні та тестовані компоненти, організовувати командну роботу, автоматизувати перевірку й доставку, а також забезпечувати якість, безпеку, супроводжуваність та еволюцію програмного продукту. Дисципліна також має на меті сформувати інженерне мислення, за якого технічне рішення обирається не за звичкою або популярністю інструменту, а на підставі вимог, обмежень, ризиків і усвідомлених компромісів. C# і C++ застосовуються як приклади мов із різними моделями виконання та сферами

	доцільного використання без надмірного заглиблення в їх синтаксис або специфічні бібліотеки.
Заплановані результати навчання	Загальні компетентності ЗК04. Здатність спілкуватися іноземною мовою. ЗК05. Знання та розуміння предметної області та розуміння професійної діяльності. ЗК06. Здатність до пошуку, оброблення та аналізу інформації з різних джерел. ЗК07. Здатність застосовувати знання у практичних ситуаціях. Спеціальні (фахові) компетентності СК01. Здатність алгоритмічно та логічно мислити. СК02. Здатність вдосконалювати знання і навички в галузі інформаційних технологій та усвідомлення важливості навчання протягом усього життя. СК04. Здатність дотримуватися стандартів при розробці програмного забезпечення. СК05. Здатність брати участь у визначенні та формулюванні вимог до програмного забезпечення. СК06. Здатність брати участь у проектуванні програмного забезпечення. СК07. Здатність розробляти модулі і компоненти програмного забезпечення за допомогою типових алгоритмів та інструментів. СК08. Здатність забезпечувати інформаційну та функціональну безпеку програмного забезпечення. СК09. Здатність вибирати та використовувати ефективні інструментальні засоби розробки програмного продукту. СК10. Здатність реалізовувати всі етапи життєвого циклу програмного забезпечення. Програмні результати навчання/результати навчання РН02. Систематизувати та узагальнювати інформацію про підходи, методи та засоби розробки супроводу програмного забезпечення. РН06. Використовувати основні методології та підходи до організації життєвого циклу програмного забезпечення. РН07. Застосовувати стандарти, специфікації в процесах життєвого циклу програмного забезпечення. РН12. Впроваджувати і супроводжувати програмні продукти.
Заплановані знання та вміння	Знати: - місце конструювання у життєвому циклі програмного забезпечення та принципи інкрементальної розробки; - способи виявлення, формулювання, пріоритизації та перевірки функціональних і нефункціональних вимог; - призначення, можливості та правила застосування основних UML-діаграм; - сутність процедурної, об'єктно-орієнтованої, функціональної та подієвої парадигм і критерії їх вибору; - відмінності C# і C++ щодо моделі виконання, типобезпечності, керування пам'яттю й ресурсами, продуктивності, переносності та швидкості розробки; - основи командної роботи з Git, проведення code review та організації безперервної інтеграції. Вміти: - перетворювати опис проблеми на узгоджений набір вимог, user stories/use cases і критеріїв приймання; - визначати межі програмної системи, моделювати її структуру та поведінку за допомогою UML;

	- обирати мову, парадигму, архітектурний стиль і спосіб декомпозиції відповідно до вимог та обмежень задачі; - розподіляти відповідальності між модулями й компонентами, визначати їх контракти та залежності.
Навчальна логістика	Зміст навчальної дисципліни: Розділ 1. Від задуму до моделі програмної системи <u>Теми розділу 1.</u> Конструювання програмного забезпечення як інженерна дисципліна. Життєвий цикл та інкрементальна розробка. Стейкхолдери, функціональні й нефункціональні вимоги, user stories, use cases та критерії приймання. Межі системи й доменна модель. UML-моделювання: діаграми прецедентів, діяльності, станів, класів, послідовностей і компонентів. Простежуваність вимог та керування змінами. Розділ 2. Конструкція коду та архітектура програмного продукту <u>Теми розділу 2.</u> Процедурна, об'єктно-орієнтована, функціональна та подієва парадигми. Вибір C# і C++ відповідно до вимог та обмежень задачі. Модульність, інкапсуляція, композиція, зв'язність і зчеплення. Принципи SOLID. Шаблони. Шарова архітектура, модульний моноліт, клієнт-серверна та hexagonal architecture. Контракти компонентів і API, робота з даними. Розділ 3. Якість, командна доставка та еволюція програмного забезпечення <u>Теми розділу 3.</u> Читабельність коду, code smells, рефакторинг і технічний борг. Unit, integration, contract та acceptance testing. Git, гілки, pull request, code review, розв'язання конфліктів і Definition of Done. Автоматизована збірка, CI, статичний аналіз, безпечна розробка, керування залежностями й секретами. Логування, метрики, діагностика, реліз, версіонування, супровід і ретроспектива програмного продукту. Види занять: лекції, лабораторні заняття. Методи навчання: - вербальні/словесні (пояснення, розповідь, бесіда); - практичні (практичні заняття); - пояснювально-ілюстративний або інформаційно-рецептивний, який передбачає пред'явлення готової інформації викладачем та її засвоєння здобувачами фахової передвищої освіти.
Пререквізити	«Технології розробки програмного забезпечення», «Програмування», «Алгоритми та методи обчислення»
Постреквізити	Навчальна практика, «Якість програмного забезпечення та тестування», «Інструментальні засоби візуального програмування»
Рекомендовані навчально-методичні матеріали для вивчення навчальної дисципліни	1. Баран С. В. Розробка програмного забезпечення з використанням патернів проектування: навчальний посібник. Кривий Ріг: Державний університет економіки і технологій, 2023. 203 с. 2. Ривкід Й., Лисенко Т., Чернікова Л., Шакотько В. Інформатика. [Розділ 5. Алгоритми та програми] – Київ: Генеза, 2022. – 277 с.: іл. 3. Професійна практика програмної інженерії : методичні вказівки до виконання лабораторних робіт / С. О. Карпова. – Миколаїв : НУК, 2024. – 52 с. 4. Гешева Г. В. Технології створення програмних продуктів: методичні вказівки до виконання курсової роботи «Проектування розробки програмного забезпечення». Запоріжжя: ТДАТУ, 2024. 39 с. 5. Рудий Т. В., Паранчук Я. С., Сенік В. В. Алгоритмізація та програмування. Частина 1. Структурне програмування : навчальний посібник. Львів : Львівський державний університет внутрішніх справ, 2023. 240 с.
Матеріально-технічне забезпечення	Мультимедійне обладнання, комп'ютерна лабораторія

Семестровий контроль, критерії оцінювання	Форма семестрового контролю – залік. Критерії оцінювання: 90–100 балів / А / Відмінно – здобувач освіти повністю володіє матеріалом дисципліни та може пройти основні етапи конструювання програмного продукту. Самостійно формулює функціональні й нефункціональні вимоги, user stories, use cases та критерії приймання, визначає межі системи. Коректно будує UML-діаграми, обирає парадигму, C# або C++ відповідно до вимог. Виконує декомпозицію на модулі та компоненти, застосовує SOLID і обґрунтовує вибір архітектури. Розрізняє види тестування, упевнено працює з Git, pull request, code review і CI, розуміє логування, версіонування та реліз. Аргументує прийняті інженерні рішення. 82–89 балів / В / Добре – здобувач освіти впевнено формулює вимоги, user stories/use cases та критерії приймання, будує основні UML-діаграми. Розуміє відмінності парадигм і C#/C++, може виконати декомпозицію системи, застосувати основні принципи SOLID та обрати типовий архітектурний підхід. Знає рівні тестування, Git, code review, CI, логування та версіонування. Допускає окремі неточності у складніших архітектурних рішеннях. 75–81 бал / С / Добре – здобувач освіти може визначити основні вимоги, сформувати прості user stories/use cases та побудувати основні UML-діаграми. Розуміє базові відмінності парадигм програмування і C#/C++. Може розділити нескладну систему на модулі або шари, пояснити призначення SOLID, тестування, Git, pull request і CI. У складніших питаннях архітектури та декомпозиції потребує спрямування викладача. 67–74 бали / D / Задовільно – здобувач освіти розуміє основні етапи життєвого циклу ПЗ, відрізняє функціональні та нефункціональні вимоги, за зразком формує user story/use case і будує просту UML-діаграму. Знає загальні відмінності C# і C++, основних парадигм, розуміє поняття модуля, компонента, SOLID, тестування, Git та CI. Практичні завдання виконує з помилками та частковою допомогою викладача. 60–66 балів / E / Задовільно – здобувач освіти володіє мінімально необхідними знаннями. Може назвати основні етапи розробки ПЗ, пояснити базові поняття вимог, UML, модулів, компонентів, тестування, Git і CI. Розпізнає основні UML-діаграми та за прикладом може створити просту модель. За допомогою викладача формулює прості вимоги та виконує елементарну декомпозицію системи. 35–59 балів / FX / Незадовільно – здобувач освіти має фрагментарні знання з вимог, UML, парадигм, архітектури, тестування та Git, але не розуміє їх взаємозв'язку в процесі створення ПЗ. Не може самостійно сформулювати вимоги, побудувати змістовну UML-модель або виконати декомпозицію системи. Практичне завдання виконує лише частково навіть із допомогою викладача. 0–34 бали / F / Незадовільно – здобувач освіти не демонструє мінімально необхідних знань. Не може пояснити основні етапи життєвого циклу ПЗ, поняття вимог, UML, модулів, компонентів, тестування та Git. Не здатний сформулювати просту вимогу, побудувати базову UML-діаграму або виконати найпростішу декомпозицію системи навіть за зразком.
Циклова комісія	Професійно-орієнтованих дисциплін та програмного забезпечення