УКРАЇНА
ЖИТОМИРСЬКИЙ КООПЕРАТИВНИЙ ФАХОВИЙ
КОЛЕДЖ БІЗНЕСУ І ПРАВА

ЗАТВЕРДЖУЮ
Директор Житомирського
кооперативного фахового коледжу
бізнесу і права
_____ Олена СИЛАНТЬЄВА
«30» серпня 2024 р.

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Навчальна практика з об'єктно-орієнтованого програмування

Освітньо-професійний ступінь	*Фаховий молодший бакалавр*
Галузь знань	*12 «Інформаційні технології»*
Спеціальність	*121 «Інженерія програмного забезпечення»*
Освітньо-професійна програма	*«Інженерія програмного забезпечення»*

Житомир 2024 р.

Навчальна практика з об'єктно-орієнтованого програмування. Програма обов'язкової навчальної дисципліни для підготовки фахових молодших бакалаврів у ЖКФКБП. Спеціальність: 121 Інженерія програмного забезпечення. Освітньо-професійна програма: Інженерія програмного забезпечення.

Укладач: Філінський Олександр Олександрович — спеціаліст

Розглянуто на засіданні циклової комісії
інженерії програмного забезпечення та математики
Протокол № ___ від «____» _____ 2024 р.
Голова комісії _____ Катерина СЛІДЗЕВСЬКА

Розглянуто на засіданні Методичної ради
Протокол № ___ від «____» _____ 2024 р.

УКРАЇНА
ЖИТОМИРСЬКИЙ КООПЕРАТИВНИЙ ФАХОВИЙ
КОЛЕДЖ БІЗНЕСУ І ПРАВА

ЗАТВЕРДЖУЮ
Директор Житомирського
кооперативного фахового коледжу
бізнесу і права
_____ Олена СИЛАНТЬЄВА
«28» серпня 2025 р.

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Навчальна практика з об'єктно-орієнтованого програмування

Освітньо-професійний ступінь	*Фаховий молодший бакалавр*
Галузь знань	*12 «Інформаційні технології»*
Спеціальність	*121 «Інженерія програмного забезпечення»*
Освітньо-професійна програма	*«Інженерія програмного забезпечення»*

Житомир 2025 р.

Навчальна практика з об'єктно-орієнтованого програмування. Програма обов'язкової навчальної дисципліни для підготовки фахових молодших бакалаврів у ЖКФКБП. Спеціальність: 121 Інженерія програмного забезпечення. Освітньо-професійна програма: Інженерія програмного забезпечення.

Укладач: Філінський Олександр Олександрович — спеціаліст

Розглянуто на засіданні циклової комісії
інженерії програмного забезпечення та математики
Протокол № ___ від «____» _____ 2025 р.
Голова комісії _____ Дмитро Оленюк

Розглянуто на засіданні Методичної ради
Протокол № ___ від «____» _____ 2025 р.

УКРАЇНА
ЖИТОМИРСЬКИЙ КООПЕРАТИВНИЙ ФАХОВИЙ
КОЛЕДЖ БІЗНЕСУ І ПРАВА

ЗАТВЕРДЖУЮ
Директор Житомирського
кооперативного фахового коледжу
бізнесу і права
_____ Олена СИЛАНТЬЄВА
«28» серпня 2026 р.

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Навчальна практика з об'єктно-орієнтованого програмування

Освітньо-професійний ступінь	*Фаховий молодший бакалавр*
Галузь знань	*12 «Інформаційні технології»*
Спеціальність	*121 «Інженерія програмного забезпечення»*
Освітньо-професійна програма	*«Інженерія програмного забезпечення»*

Житомир 2026 р.

Навчальна практика з об'єктно-орієнтованого програмування. Програма обов'язкової навчальної дисципліни для підготовки фахових молодших бакалаврів у ЖКФКБП. Спеціальність: 121 Інженерія програмного забезпечення. Освітньо-професійна програма: Інженерія програмного забезпечення.

Укладач: Філінський Олександр Олександрович — спеціаліст

Розглянуто на засіданні циклової комісії
інженерії програмного забезпечення та математики
Протокол № ___ від «____» _____ 2026 р.
Голова комісії _____ Наталія ГОНГАЛО

Розглянуто на засіданні Методичної ради
Протокол № ___ від «____» _____ 2026 р.

ЗМІСТ

1. ОПИС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ 8
2. МЕТА І ЗАВДАННЯ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ 9
3. ПЕРЕЛІК ЗАГАЛЬНИХ І СПЕЦІАЛЬНИХ КОМПЕТЕНТНОСТЕЙ 11
5. ІНФОРМАЦІЙНИЙ ОБСЯГ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ 15
6. ЗМІСТ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ 16
7. ТЕМИ ПРАКТИЧНИХ ЗАНЯТЬ 32
8. МЕТОДИ НАВЧАННЯ, КОНТРОЛЮ ТА ЗАСОБИ ОЦІНЮВАННЯ 35
9. ПОРЯДОК ТА КРИТЕРІЇ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ НАВЧАННЯ. 37
10. СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ 40

1. ОПИС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

<table>
<tr><td rowspan="2">Характеристика навчальної дисципліни</td><td rowspan="2">Галузь знань, спеціальність, освітньо-професійна програма, освітньо-професійний ступінь</td><td colspan="8">Розподіл годин</td></tr>
<tr><td colspan="8">Семестри</td></tr>
<tr><td rowspan="11">Обов’язкова

Загальний обсяг:
Годин — 180
Кредитів — 6

Тижневих годин:
Аудиторні — 72
(6 тижнів по 12 годин)

Індивідуальне завдання —
передбачено навчальним планом</td><td rowspan="11">Галузь знань:
12 «Інформаційні технології»

Спеціальність:
121 «Інженерія програмного забезпечення»

Освітньо-професійна програма:
«Інженерія програмного забезпечення»

Освітньо-професійний ступінь: фаховий молодший бакалавр</td><td>1-й</td><td>2-й</td><td>3-й</td><td>4-й</td><td>5-й</td><td>6-й</td><td>7-й</td><td>8-й</td></tr>
<tr><td colspan="8">Лекції</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td colspan="8">Практичні заняття</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>72</td><td>–</td><td>–</td></tr>
<tr><td colspan="8">Семінарські заняття</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td colspan="8">Лабораторні заняття</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td colspan="8">Самостійна робота</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>108</td><td>–</td><td>–</td></tr>
<tr><td colspan="8">Індивідуальні завдання</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>+</td><td>–</td><td>–</td></tr>
<tr><td colspan="8">Форма підсумкового контролю</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>ДЗ</td><td>–</td><td>–</td></tr>
</table>

Форма підсумкового контролю — диференційований залік.

2. МЕТА І ЗАВДАННЯ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Мета навчальної дисципліни: поглиблене практичне опанування мови C++ та стандартної бібліотеки шаблонів (STL) на основі базових знань і навичок, здобутих у курсі «Основи програмування»: формування у здобувачів освіти стійких навичок проєктування та реалізації програмних застосунків із використанням об’єктно-орієнтованого підходу, сучасних можливостей стандартів C++ та засобів стандартної бібліотеки як базового інструменту фахівця у сфері інженерії програмного забезпечення.

Завдання навчальної дисципліни:

– закріпити та поглибити практичні навички роботи із сучасними інструментами розроблення мовою C++: компілятором, системою збирання CMake, відлагоджувачем та системою контролю версій Git;
– сформувати вміння проєктувати й реалізовувати класи з коректною інкапсуляцією, конструкторами, деструкторами та константними методами;
– опанувати практичні прийоми керування ресурсами: ідіому RAII, копіювання та переміщення об’єктів, правила трьох, п’яти та нуля, розумні вказівники;
– відпрацювати перевантаження операцій, застосування функціональних об’єктів та лямбда-виразів;
– опанувати практичну реалізацію успадкування та поліморфізму: віртуальних методів, абстрактних класів та інтерфейсів, множинного й віртуального успадкування;
– сформувати навички розроблення шаблонів функцій і класів, застосування концептів та узагальненого програмування;
– сформувати навички опрацювання виключних ситуацій та забезпечення безпеки програм щодо винятків;

– опанувати контейнери, ітератори та алгоритми стандартної бібліотеки шаблонів (STL), бібліотеку діапазонів, а також засоби роботи з рядками, потоками введення-виведення та файловою системою;
– сформувати вміння застосовувати принципи SOLID і базові шаблони проєктування, тестувати, відлагоджувати й документувати власний програмний код;
– виробити навички самостійного виконання індивідуального проєкту та презентації результатів власної роботи.

Предметом вивчення навчальної дисципліни є практична діяльність з розроблення програмного забезпечення мовою C++ із застосуванням об’єктно-орієнтованого програмування та стандартної бібліотеки шаблонів: проєктування класів та ієрархій класів, керування ресурсами, узагальнене програмування, використання контейнерів, ітераторів і алгоритмів STL для розв’язання прикладних задач.

Міждисциплінарні зв’язки: «Основи програмування», «Навчальна практика з основ програмування», «Алгоритми та структури даних», «Дискретна математика», «Об’єктно-орієнтоване програмування», «Архітектура комп’ютерів та операційні системи», «Технології розроблення програмного забезпечення», «Бази даних».

3. ПЕРЕЛІК ЗАГАЛЬНИХ І СПЕЦІАЛЬНИХ КОМПЕТЕНТНОСТЕЙ

Перелік загальних (ЗК) і спеціальних (СК) компетентностей випускника відповідно до Освітньо-професійної програми підготовки фахового молодшого бакалавра зі спеціальності 121 «Інженерія програмного забезпечення»:

- ЗК 01. Здатність реалізувати свої права і обов'язки як члена суспільства, усвідомлювати цінності громадянського (вільного демократичного) суспільства та необхідність його сталого розвитку, верховенства права, прав і свобод людини і громадянина в Україні.
- ЗК 02. Здатність зберігати та примножувати моральні, культурні, наукові цінності і досягнення суспільства на основі розуміння історії та закономірностей розвитку предметної області, її місця у загальній системі знань про природу і суспільство та у розвитку суспільства, техніки і технологій, використовувати різні види та форми рухової активності для активного відпочинку та ведення здорового способу життя.
- ЗК 03. Здатність спілкуватися державною мовою як усно, так і письмово.
- ЗК 04. Здатність спілкуватися іноземною мовою.
- ЗК 05. Знання та розуміння предметної області та розуміння професійної діяльності.
- ЗК 06. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.
- ЗК 07. Здатність застосовувати знання у практичних ситуаціях.
- СК 01. Здатність алгоритмічно та логічно мислити.
- СК 02. Здатність вдосконалювати знання і навички в галузі інформаційних технологій та усвідомлення важливості навчання протягом усього життя.
- СК 03. Здатність застосовувати теоретичні та емпіричні знання для розроблення, тестування, впровадження та супроводу програмного забезпечення.

– СК 04. Здатність дотримуватися стандартів при розробці програмного забезпечення.

– СК 05. Здатність брати участь у визначенні та формулюванні вимог до програмного забезпечення.

– СК 06. Здатність брати участь у проектуванні програмного забезпечення.

– СК 07. Здатність розробляти модулі і компоненти програмного забезпечення за допомогою типових алгоритмів та інструментів.

– СК 08. Здатність забезпечувати інформаційну та функціональну безпеку програмного забезпечення.

– СК 09. Здатність вибирати та використовувати ефективні інструментальні засоби розробки програмного продукту.

– СК 10. Здатність реалізовувати всі етапи життєвого циклу програмного забезпечення.

4. Нормативний зміст підготовки здобувачів фахової передвищої освіти, сформульований у термінах результатів навчання

– РН 01. Застосовувати основні принципи професійної етики у галузі програмної інженерії, усвідомлювати їх соціальну значимість та культурні аспекти в професійній діяльності.

– РН 02. Систематизувати та узагальнювати інформацію про підходи, методи та засоби розробки супроводу програмного забезпечення.

– РН 03. Застосовувати спеціалізовані емпіричні та теоретичні знання у сфері інженерії програмного забезпечення.

– РН 04. Використовувати знання математичних методів на рівні, необхідному для розв'язання типових задач програмної інженерії.

– РН 05. Розробляти та супроводжувати програмне забезпечення.

– РН 06. Використовувати основні методології та підходи до організації життєвого циклу програмного забезпечення.

– РН 07. Застосовувати стандарти, специфікації в процесах життєвого циклу програмного забезпечення.

– РН 08. Аналізувати вимоги до програмного забезпечення.

– РН 09. Розуміти основні принципи командної роботи при розробці програмного забезпечення.

– РН 10. Обирати та застосовувати ефективні методи оптимізації алгоритмів.

– РН 11. Обирати інструментальні засоби, ефективні методи та здійснювати тестування програмних систем.

– РН 12. Впроваджувати і супроводжувати програмні продукти.

– РН 13. Спілкуватися українською та іноземною мовою усно і письмово з питань інженерії програмного забезпечення.

– РН 14. Розуміти предметну область, застосовувати знання у професійній діяльності.

– РН 15. Аналізувати та узагальнювати необхідну інформацію з різних джерел та ресурсів для розв'язання професійних задач з урахуванням сучасних досягнень інформаційних технологій.

5. ІНФОРМАЦІЙНИЙ ОБСЯГ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

№ з/п	Назва теми	Усього годин	З них	
			практичні заняття	Самостійна робота
1	Вступ до навчальної практики. Сучасне середовище розробки мовою C++	6	2	4
2	Класи та об’єкти. Інкапсуляція	12	6	6
3	Керування ресурсами. Копіювання та переміщення об’єктів	18	6	12
4	Перевантаження операцій. Функціональні об’єкти та лямбда-вирази	12	6	6
5	Успадкування	12	6	6
6	Поліморфізм. Абстрактні класи та інтерфейси	12	6	6
7	Розумні вказівники та сучасне керування пам’яттю	6	2	4
8	Опрацювання виключних ситуацій	12	6	6
9	Шаблони функцій і класів. Узагальнене програмування	18	6	12
10	Послідовні контейнери та ітератори STL	12	6	6
11	Асоціативні контейнери та адаптери контейнерів	12	4	8
12	Алгоритми STL. Бібліотека діапазонів	12	6	6
13	Рядки, потоки введення-виведення та файлова система	6	2	4
14	Багатопотокове програмування в C++	6	2	4
15	Принципи проєктування. Шаблони проєктування. Модульне тестування	6	2	4
16	Практикум з програмування: розроблення та захист прикладного проєкту	18	4	14
Разом		**180**	**72**	**108**

Форма підсумкового контролю — диференційований залік.

6. ЗМІСТ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Тема 1. Вступ до навчальної практики. Сучасне середовище розробки мовою C++

Мета, завдання та структура навчальної практики. Вимоги до виконання та захисту завдань. Інструктаж з охорони праці, техніки безпеки та безпеки життєдіяльності під час роботи з персональним комп'ютером. Організація робочого місця програміста. Еволюція стандартів мови C++: C++11, C++14, C++17, C++20, C++23; підтримка стандартів компіляторами GCC, Clang та MSVC. Повторення базових конструкцій мови, вивчених у курсі «Основи програмування»: типи даних, керуючі оператори, функції, покажчики, посилання, структури. Інструментальні засоби розроблення: опції компілятора (-std, -Wall, -Wextra), система збирання CMake, відлагоджувач, статичні аналізатори та санітайзери (ознайомлення). Основні команди системи контролю версій Git. Структура багатофайлового проєкту: заголовкові файли та файли реалізації, захист від повторного підключення (#pragma once, include guards).

Самостійна робота: опрацювання конспекту лекцій; повторення правил охорони праці та організації робочого місця; підготовка короткого порівняння можливостей стандартів C++11, C++14, C++17, C++20 та C++23 (по три нововведення кожного); самостійне встановлення компілятора, CMake та Git на домашньому комп'ютері; створення багатофайлового проєкту з файлом CMakeLists.txt, збирання з прапорцями -std=c++20 -Wall -Wextra та аналіз попереджень; запуск програми з AddressSanitizer; створення репозиторію на GitHub та виконання базових команд Git (init, add, commit, branch, merge, push); повторення раніше вивчених конструкцій мови (функції, покажчики, посилання, структури) на задачах із розділу «Основи програмування».

Студенти повинні:

знати: мету та завдання навчальної практики; правила охорони праці й техніки безпеки; стандарти мови C++ та їхні основні можливості; склад інструментальних засобів розроблення програм; основні команди Git;

уміти: організовувати робоче місце; налаштовувати компілятор, систему збирання та середовище розробки; створювати, збирати й запускати багатофайловий проєкт мовою C++; користуватися основними командами Git.

Тема 2. Класи та об’єкти. Інкапсуляція

Обмеження процедурного підходу та передумови виникнення об’єктно-орієнтованого програмування. Клас як тип даних, об’єкт як екземпляр класу. Поля та методи класу, специфікатори доступу private, protected, public. Розділення оголошення та реалізації класу. Інкапсуляція, інваріанти класу, методи доступу. Константні методи та const-коректність. Конструктори: за замовчуванням, параметризовані, список ініціалізації, делегувальні конструктори, специфікатор explicit, = default та = delete. Ініціалізація полів за замовчуванням, уніфікована ініціалізація {}. Деструктор. Порядок створення та знищення об’єктів. Покажчик this, ланцюжки викликів методів. Статичні члени класу. Дружні функції та класи. Вкладені класи. Простори імен. Перелічення enum class.

Самостійна робота: опрацювання конспекту лекцій; перетворення раніше розробленої структури (наприклад, «Студент», «Товар») на клас із закритими полями, інваріантами та методами доступу; розроблення класів предметної області (Рахунок, Точка, Прямокутник, Календар) з різними видами конструкторів (за замовчуванням, параметризований, делегувальний, explicit); використання списку ініціалізації, = default та = delete; спостереження за порядком створення й знищення об’єктів за допомогою виведення повідомлень у конструкторах і деструкторі; реалізація лічильника об’єктів на статичному полі; застосування const-коректності та ланцюжків викликів методів; розділення класу на файли .h та .cpp із include guard.

Студенти повинні:

знати: поняття класу й об’єкта; призначення специфікаторів доступу; види конструкторів та призначення деструктора; правила const-коректності; призначення статичних членів і дружніх функцій;

уміти: оголошувати класи та розділяти їх оголошення й реалізацію; реалізовувати конструктори, деструктор та методи доступу; застосовувати інкапсуляцію, статичні члени та константні методи; створювати та використовувати об’єкти.

Тема 3. Керування ресурсами. Копіювання та переміщення об’єктів

Поняття ресурсу та часу його життя. Ідіома RAII. Динамічна пам’ять у складі класу, витоки пам’яті. Поверхневе та глибоке копіювання. Конструктор копіювання та оператор присвоєння копіюванням. Lvalue та rvalue, rvalue-посилання, std::move, std::forward (ознайомлення). Конструктор переміщення та оператор присвоєння переміщенням. Оптимізація копіювання: copy elision, RVO. Правила трьох, п’яти та нуля. Специфікатор noexcept для операцій переміщення. Заборона копіювання об’єктів. Ідіома copy-and-swap.

Самостійна робота: опрацювання конспекту лекцій; розроблення класу-обгортки над динамічним масивом (власний аналог std::vector) із застосуванням ідіоми RAII; реалізація конструктора копіювання й оператора присвоєння та демонстрація помилок поверхневого копіювання; доповнення класу конструктором і оператором присвоєння переміщенням із noexcept; реалізація ідіоми copy-and-swap; дослідження правил трьох, п’яти та нуля на прикладах; перевірка відсутності витоків пам’яті за допомогою санітайзерів або Valgrind; вимірювання часу виконання копіювання та переміщення великих об’єктів; створення класу, копіювання якого заборонене.

Студенти повинні:

знати: ідіому RAII; різницю між поверхневим і глибоким копіюванням; поняття rvalue-посилання та семантики переміщення; правила трьох, п’яти та нуля;

уміти: розробляти класи, що володіють ресурсами; реалізовувати конструктори та оператори копіювання й переміщення; застосовувати std::move; запобігати витокам пам’яті та подвійному звільненню ресурсів.

Тема 4. Перевантаження операцій. Функціональні об’єкти та лямбда-вирази

Правила та обмеження перевантаження операцій. Перевантаження операцій як методів класу та як вільних функцій. Арифметичні операції та операції порівняння, індексування [], виклику функції (), інкременту й декременту, потокового введення та виведення (<<, >>). Оператор тривимірного порівняння <=> та автоматично згенеровані оператори порівняння (C++20). Користувацькі перетворення типів: конвертувальні конструктори та оператори перетворення, специфікатор explicit. Функціональні об’єкти (функтори). Лямбда-вирази: список захоплення, параметри, узагальнені лямбда-вирази, зберігання у std::function. Принцип найменшого здивування під час перевантаження операцій.

Самостійна робота: опрацювання конспекту лекцій; розроблення класів «Комплексне число», «Раціональний дріб», «Вектор», «Матриця» з перевантаженими арифметичними операціями, операціями порівняння, індексування [], виклику функції () та потокових операцій << і >>; використання оператора <=> та автоматично згенерованих операторів порівняння (C++20); реалізація конвертувальних конструкторів та операторів перетворення типу з explicit; написання функціонального об’єкта для порівняння записів; розв’язання задач із лямбда-виразами (різні способи захоплення, узагальнені лямбда-вирази, зберігання в std::function); аналіз

випадків, коли перевантаження операцій порушує принцип найменшого здивування.

Студенти повинні:

знати: правила та обмеження перевантаження операцій; способи перевантаження (метод класу, вільна функція, дружня функція); призначення функторів і лямбда-виразів; особливості оператора <=>;

уміти: перевантажувати арифметичні операції, операції порівняння, індексування, виклику функції та потокові операції; реалізовувати користувацькі перетворення типів; створювати функціональні об’єкти та лямбда-вирази.

Тема 5. Успадкування

Успадкування як механізм повторного використання коду та відношення «є» (is-a). Базовий і похідний класи. Види успадкування public, protected, private. Порядок виклику конструкторів і деструкторів, передавання параметрів у конструктор базового класу. Успадкування конструкторів (using). Приховування й перевизначення методів, звертання до членів базового класу. Багаторівневе успадкування. Множинне успадкування, неоднозначність імен, проблема ромба, віртуальне успадкування. Композиція та агрегація, принцип «композиція замість успадкування». Проблема зрізання об’єктів (slicing).

Самостійна робота: опрацювання конспекту лекцій; побудова діаграм класів UML для обраної предметної області; розроблення ієрархії класів (наприклад, Транспорт → Автомобіль, Автобус, Мотоцикл; Співробітник → Менеджер, Програміст) з використанням одиночного та багаторівневого успадкування; спостереження за порядком виклику конструкторів і деструкторів у ієрархії; передавання параметрів у конструктор базового класу та успадкування конструкторів (using); моделювання проблеми ромба при множинному успадкуванні та її усунення за допомогою віртуального

успадкування; демонстрація проблеми зрізання об’єктів (slicing); порівняння успадкування та композиції на двох реалізаціях однієї задачі.

Студенти повинні:

знати: призначення та види успадкування; порядок виклику конструкторів і деструкторів у ієрархії; проблему ромба та способи її розв’язання; різницю між успадкуванням, композицією та агрегацією;

уміти: будувати ієрархії класів; передавати параметри у конструктори базових класів; перевизначати методи; застосовувати множинне й віртуальне успадкування; обирати між успадкуванням і композицією.

Тема 6. Поліморфізм. Абстрактні класи та інтерфейси

Статичний і динамічний поліморфізм. Віртуальні функції, таблиця віртуальних методів, механізм пізнього зв’язування. Специфікатори override та final. Віртуальний деструктор. Чисто віртуальні функції, абстрактні класи, інтерфейси. Особливості виклику віртуальних функцій у конструкторах і деструкторах. Ідіома NVI (non-virtual interface), ознайомлення. Інформація про типи під час виконання: typeid, dynamic_cast, static_cast, const_cast. Зберігання поліморфних об’єктів у контейнерах за допомогою покажчиків. Віртуальне копіювання (метод clone()).

Самостійна робота: опрацювання конспекту лекцій; розроблення поліморфної ієрархії (геометричні фігури, банківські рахунки, графічні об’єкти) з віртуальними методами, специфікаторами override та final; створення абстрактного класу й інтерфейсу з чисто віртуальними функціями; зберігання поліморфних об’єктів у std::vector за допомогою покажчиків та їх обробка в циклі; демонстрація проблеми невіртуального деструктора та її усунення; дослідження виклику віртуальних функцій у конструкторах і деструкторах; реалізація віртуального копіювання (метод clone()); використання dynamic_cast, typeid та static_cast із перевіркою результату; порівняння статичного й динамічного поліморфізму.

Студенти повинні:

знати: механізм динамічного поліморфізму; призначення специфікаторів override та final; поняття абстрактного класу та інтерфейсу; призначення віртуального деструктора; засоби визначення типу під час виконання;

уміти: реалізовувати поліморфну поведінку за допомогою віртуальних методів; створювати абстрактні класи та інтерфейси; застосовувати dynamic_cast та typeid; зберігати поліморфні об’єкти в контейнерах.

Тема 7. Розумні вказівники та сучасне керування пам’яттю

Проблеми «сирих» покажчиків: витоки пам’яті, висячі покажчики, подвійне видалення. Поняття власності на об’єкт. Розумні вказівники бібліотеки <memory>: std::unique_ptr (виключна власність, передавання власності), std::shared_ptr (спільна власність, лічильник посилань), std::weak_ptr (розрив циклічних залежностей). Функції std::make_unique та std::make_shared. Користувацькі функції видалення. Передавання розумних вказівників у функції та повернення з функцій. Застосування розумних вказівників у поліморфних ієрархіях та структурах даних.

Самостійна робота: опрацювання конспекту лекцій; пошук і виправлення помилок роботи з «сирими» покажчиками (витік пам’яті, висячий покажчик, подвійне видалення) у наданих фрагментах коду; заміна «сирих» покажчиків на std::unique_ptr та std::shared_ptr у раніше розробленій ієрархії класів; передавання власності на об’єкт через std::move; реалізація двозв’язного списку або дерева з використанням std::shared_ptr і std::weak_ptr для розриву циклічних залежностей; спостереження за лічильником посилань (use_count); створення користувацької функції видалення; обґрунтування вибору типу розумного вказівника для параметрів і значень, що повертаються з функцій; використання std::make_unique та std::make_shared.

Студенти повинні:

знати: проблеми використання «сирих» покажчиків; призначення та відмінності std::unique_ptr, std::shared_ptr і std::weak_ptr; поняття власності та циклічної залежності;

уміти: застосовувати розумні вказівники для керування динамічною пам'яттю; передавати власність за допомогою std::move; розривати циклічні залежності; використовувати make_unique та make_shared.

Тема 8. Опрацювання виключних ситуацій

Недоліки кодів повернення як способу сигналізування про помилки. Механізм виключень: throw, try, catch, перехоплення за посиланням, порядок обробників, catch(...). Розкручування стеку та виклик деструкторів. Ієрархія класів стандартних виключень std::exception: logic_error, runtime_error, out_of_range, bad_alloc, bad_cast; метод what(). Користувацькі класи виключень. Специфікатор noexcept, функція std::terminate. Гарантії безпеки щодо виключень: базова, строга, гарантія відсутності виключень. Виключення в конструкторах і деструкторах. Альтернативні засоби: std::optional, std::variant та std::visit, std::expected (C++23, ознайомлення), assert та static_assert.

Самостійна робота: опрацювання конспекту лекцій; порівняння обробки помилок за допомогою кодів повернення та виключень на прикладі однієї задачі; розроблення програм із генеруванням і перехопленням виключень стандартних типів (out_of_range, invalid_argument, bad_alloc); створення власної ієрархії класів виключень із методом what(); спостереження за розкручуванням стеку та викликом деструкторів; написання класу з гарантією безпеки щодо виключень (базова та строга гарантії); застосування noexcept; використання std::optional, std::variant і std::visit як альтернативи виключенням; застосування assert та static_assert; перевірка коректності введення користувача.

Студенти повинні:

знати: механізм виключень мови C++; ієрархію стандартних виключень; гарантії безпеки щодо виключень; призначення noexcept; альтернативні засоби повідомлення про помилки;

уміти: генерувати та перехоплювати виключення; створювати власні класи виключень; забезпечувати безпеку коду щодо виключень; застосовувати std::optional та std::variant.

Тема 9. Шаблони функцій і класів. Узагальнене програмування

Поняття узагальненого програмування. Шаблони функцій: виведення типів аргументів, явне задання параметрів, перевантаження шаблонів. Шаблони класів: параметри-типи та нетипові параметри, шаблонні методи, параметри за замовчуванням. Розміщення оголошення та реалізації шаблонів. Повна та часткова спеціалізація шаблонів. Залежні імена, ключове слово typename. Шаблони зі змінною кількістю параметрів (variadic templates), згортальні вирази (C++17). Ключові слова auto та decltype, виведення параметрів шаблонів класів (CTAD). Специфікатори constexpr та consteval. Концепти та обмеження шаблонів requires (C++20). Бібліотека <type_traits> та static_assert (ознайомлення). Шаблони псевдонімів using.

Самостійна робота: опрацювання конспекту лекцій; розроблення шаблонів функцій (пошук максимуму, обмін значень, сортування) із явним і неявним виведенням типів; створення шаблонів класів «Стек», «Черга», «Зв’язний список», «Матриця» з нетиповими параметрами; реалізація повної та часткової спеціалізації шаблонів; написання шаблону зі змінною кількістю параметрів і згортального виразу; використання auto, decltype та виведення параметрів шаблонів класів (CTAD); обмеження шаблонів за допомогою концептів і requires (C++20); перевірка властивостей типів за допомогою <type_traits> і static_assert; аналіз повідомлень компілятора про помилки в шаблонах.

Студенти повинні:

знати: поняття шаблону та інстанціювання; види параметрів шаблонів; правила спеціалізації; призначення variadic templates, constexpr та концептів;

уміти: розробляти шаблони функцій і класів; застосовувати спеціалізацію шаблонів; обмежувати шаблони за допомогою концептів; використовувати constexpr та auto.

Тема 10. Послідовні контейнери та ітератори STL

Склад стандартної бібліотеки шаблонів: контейнери, ітератори, алгоритми, функціональні об’єкти. Послідовні контейнери: std::array, std::vector (розмір та ємність, reserve, shrink_to_fit, push_back та emplace_back), std::deque, std::list, std::forward_list. Обчислювальна складність операцій. Категорії ітераторів, функції begin, end, cbegin, rbegin, цикл for для діапазону, структуровані прив’язки. Інвалідація ітераторів. Ідіома erase–remove. Критерії вибору контейнера. std::span (C++20). Допоміжні типи std::pair та std::tuple.

Самостійна робота: опрацювання конспекту лекцій; складання таблиці складності основних операцій контейнерів std::array, std::vector, std::deque, std::list та std::forward_list; розв’язання задач із використанням послідовних контейнерів; експериментальне порівняння швидкодії вставлення, видалення й доступу для різних контейнерів на великих обсягах даних; дослідження росту ємності std::vector (size, capacity, reserve, shrink_to_fit); демонстрація інвалідації ітераторів та правильне видалення елементів за ідіомою erase–remove; використання зворотних ітераторів, структурованих прив’язок, std::pair, std::tuple та std::span; обґрунтування вибору контейнера для заданої задачі.

Студенти повинні:

знати: склад та призначення STL; властивості послідовних контейнерів та складність їхніх операцій; категорії ітераторів; правила інвалідації ітераторів;

уміти: застосовувати послідовні контейнери STL; працювати з ітераторами; обирати доцільний контейнер для розв'язання задачі; уникати помилок, пов'язаних з інвалідацією ітераторів.

Тема 11. Асоціативні контейнери та адаптери контейнерів

Впорядковані асоціативні контейнери: std::set, std::multiset, std::map, std::multimap; обчислювальна складність операцій. Методи insert, emplace, try_emplace, find, count, contains (C++20), lower_bound, upper_bound; відмінність operator[] та at. Невпорядковані асоціативні контейнери: std::unordered_set, std::unordered_map, хеш-функції, коефіцієнт завантаження. Користувацькі компаратори та хеш-функції. Адаптери контейнерів: std::stack, std::queue, std::priority_queue. Використання у контейнерах об'єктів власних класів. Порівняння та вибір контейнерів.

Самостійна робота: опрацювання конспекту лекцій; розв'язання задач на використання std::set, std::multiset, std::map, std::multimap, std::unordered_set та std::unordered_map (підрахунок частоти слів, словник, телефонний довідник, пошук унікальних елементів); порівняння швидкодії впорядкованих і невпорядкованих контейнерів; вивчення відмінності operator[] та at, методів try_emplace, contains, lower_bound; написання користувацького компаратора та хеш-функції для власного класу; розв'язання задач із використанням std::stack, std::queue та std::priority_queue (перевірка дужкових послідовностей, обчислення виразу, планувальник завдань); оформлення порівняльної таблиці контейнерів.

Студенти повинні:

знати: властивості та складність операцій асоціативних контейнерів; відмінності між впорядкованими та невпорядкованими контейнерами; призначення адаптерів контейнерів;

уміти: застосовувати асоціативні контейнери та адаптери; задавати користувацькі компаратори та хеш-функції; обирати доцільний контейнер для розв’язання задачі.

Тема 12. Алгоритми STL. Бібліотека діапазонів

Бібліотеки <algorithm> та <numeric>. Алгоритми без модифікації послідовності: find, find_if, count, all_of, any_of, none_of, for_each, equal, mismatch. Алгоритми, що змінюють послідовність: copy, transform, replace, remove, unique, reverse, rotate, fill, generate. Сортування та пошук: sort, stable_sort, partial_sort, nth_element, binary_search, lower_bound, merge. Числові алгоритми: accumulate, reduce, iota, partial_sum, inner_product. Предикати, компаратори, лямбда-вирази як аргументи алгоритмів. std::function та std::bind (ознайомлення). Ітератори-адаптери: back_inserter, istream_iterator. Політики паралельного виконання алгоритмів (C++17, ознайомлення). Бібліотека діапазонів <ranges> (C++20): алгоритми над діапазонами, представлення views (filter, transform, take, drop), конвеєри з оператором |.

Самостійна робота: опрацювання конспекту лекцій; розв’язання задач із використанням алгоритмів без модифікації та зі зміною послідовності (find_if, count_if, transform, remove_if, unique, rotate, copy_if); застосування алгоритмів сортування й пошуку (sort, stable_sort, partial_sort, nth_element, binary_search, merge) із лямбда-виразами як компараторами; використання числових алгоритмів accumulate, reduce, iota, partial_sum та inner_product; заміна ручних циклів на виклики алгоритмів; побудова конвеєрів обробки даних за допомогою бібліотеки діапазонів (views::filter, views::transform, views::take); порівняння ітеративного підходу, алгоритмів STL і діапазонів за читабельністю та швидкодією.

Студенти повинні:

знати: склад та призначення алгоритмів STL; вимоги до предикатів і компараторів; призначення бібліотеки діапазонів та представлень;

уміти: застосовувати алгоритми STL разом з контейнерами та лямбда-виразами; користуватися ітераторами-адаптерами; будувати конвеєри оброблення даних за допомогою діапазонів.

Тема 13. Рядки, потоки введення-виведення та файлова система

Клас std::string: методи, особливості збереження коротких рядків. Клас std::string_view. Перетворення між рядками та числами: to_string, stoi, stod, from_chars, to_chars. Ієрархія потоків введення-виведення, стан потоку, маніпулятори форматування. Рядкові потоки std::stringstream. Файлові потоки std::fstream, текстові та двійкові файли. Збереження й відновлення стану об’єктів (серіалізація), перевантаження операцій << та >> для власних класів. Бібліотека <filesystem>: path, directory_iterator. Форматування виведення за допомогою std::format (C++20). Регулярні вирази <regex> та засоби роботи з часом <chrono> (ознайомлення).

Самостійна робота: опрацювання конспекту лекцій; розв’язання задач на обробку тексту з використанням std::string та std::string_view (підрахунок слів, заміна підрядків, розбиття на лексеми); перетворення між рядками та числами (stoi, stod, from_chars, to_chars); форматування виведення за допомогою маніпуляторів і std::format; використання std::stringstream для розбору рядків; запис і читання об’єктів власних класів у текстових і двійкових файлах (серіалізація) із перевантаженням операцій << та >>; опрацювання CSV-файлу з обробкою помилок відкриття й читання; використання <filesystem> для перегляду каталогу та перевірки існування файлів; ознайомлення з <regex> та <chrono> на простих прикладах.

Студенти повинні:

знати: можливості та методи класів std::string і std::string_view; ієрархію потоків введення-виведення; засоби роботи з файлами та файловою системою;

уміти: опрацьовувати текстові дані; зберігати та відновлювати стан об’єктів у файлах; застосовувати std::format та засоби бібліотеки <filesystem>.

Тема 14. Багатопотокове програмування в C++

Поняття процесу та потоку виконання. Класи std::thread та std::jthread (C++20), передавання аргументів у потік, очікування завершення потоку. Гонка даних та критична секція. Взаємне виключення: std::mutex, std::lock_guard, std::unique_lock, std::scoped_lock; взаємне блокування (deadlock). Умовні змінні std::condition_variable. Атомарні типи std::atomic. Асинхронні завдання: std::async, std::future, std::promise. Потокобезпека контейнерів STL.

Самостійна робота: опрацювання конспекту лекцій; створення багатопотокових програм із використанням std::thread та std::jthread (паралельне обчислення суми елементів масиву, пошук у кількох файлах); навмисне відтворення гонки даних та її усунення за допомогою std::mutex, std::lock_guard і std::scoped_lock; моделювання взаємного блокування (deadlock) та способів його запобігання; реалізація задачі «виробник–споживач» із std::condition_variable; використання std::atomic для лічильників; застосування std::async, std::future та std::promise; вимірювання прискорення при різній кількості потоків; перевірка програм за допомогою ThreadSanitizer.

Студенти повинні:

знати: поняття потоку виконання; причини виникнення гонки даних та взаємного блокування; засоби синхронізації потоків у стандартній бібліотеці;

уміти: створювати та запускати потоки; захищати спільні дані за допомогою м’ютексів; застосовувати std::async та std::future; запобігати гонці даних.

Тема 15. Принципи проєктування. Шаблони проєктування. Модульне тестування

Принципи SOLID, DRY та KISS. Зв’язність і залежність модулів. Діаграми класів UML. Базові шаблони проєктування та їх реалізація мовою

C++: породжувальні (Singleton, Factory Method, Builder, Prototype), структурні (Adapter, Decorator, Composite, Facade), поведінкові (Strategy, Observer, Command, State, Iterator). Модульне тестування: тестові фреймворки (Google Test, Catch2), тестові випадки, перевірочні макроси, фікстури. Рефакторинг. Документування коду (Doxygen). Стиль кодування та настанови C++ Core Guidelines.

Самостійна робота: опрацювання конспекту лекцій; аналіз власного коду на відповідність принципам SOLID, DRY і KISS та його рефакторинг; побудова діаграм класів UML; реалізація шаблонів проєктування мовою C++ (Singleton, Factory Method, Builder, Adapter, Decorator, Composite, Strategy, Observer, Command, State) на прикладах предметних областей; написання модульних тестів для раніше розроблених класів за допомогою Google Test або Catch2 (звичайні, межові та помилкові випадки, фікстури); підключення тестів до проєкту CMake; документування коду коментарями Doxygen; вивчення настанов C++ Core Guidelines.

Студенти повинні:

знати: принципи SOLID, DRY та KISS; призначення та класифікацію шаблонів проєктування; основи модульного тестування;

уміти: застосовувати принципи SOLID під час проєктування класів; реалізовувати базові шаблони проєктування; розробляти модульні тести; документувати програмний код.

Тема 16. Практикум з програмування: розроблення та захист прикладного проєкту

Етапи розроблення програмного продукту. Постановка задачі та формулювання функціональних вимог. Проєктування ієрархії класів, діаграма класів UML, розподіл програми на модулі. Вибір контейнерів та алгоритмів STL. Реалізація прикладного застосунку (наприклад, система обліку, інформаційна система з консольним інтерфейсом) із застосуванням

інкапсуляції, успадкування, поліморфізму, шаблонів, розумних вказівників, опрацювання виключень та файлового введення-виведення. Модульне тестування, збирання проєкту за допомогою CMake, версіонування у Git. Документування коду. Підготовка презентації та захист проєкту.

Самостійна робота: вибір теми індивідуального проєкту та узгодження її з викладачем; формулювання функціональних вимог і складання плану роботи; проєктування ієрархії класів, діаграми класів UML та структури модулів; вибір контейнерів і алгоритмів STL для зберігання та опрацювання даних; поетапна реалізація застосунку із застосуванням інкапсуляції, успадкування, поліморфізму, шаблонів, розумних вказівників, виключень і файлового введення-виведення; написання модульних тестів; налаштування збирання за допомогою CMake; ведення історії розроблення в Git (регулярні коміти з осмисленими повідомленнями); підготовка інструкції користувача, презентації та доповіді для захисту; репетиція захисту.

Студенти повинні:

знати: етапи розроблення програмного продукту; правила документування програмного коду; вимоги до оформлення й захисту індивідуального проєкту;

уміти: розробляти завершені програми з використанням об’єктно-орієнтованого підходу та STL; інтегрувати вивчені засоби мови в один проєкт; тестувати, відлагоджувати й документувати код; презентувати та захищати власні програмні рішення.

7. ТЕМИ ПРАКТИЧНИХ ЗАНЯТЬ

№ з/п	№ заняття	Тема заняття	Кількість годин
1	1	Тема 1. Вступ до навчальної практики. Сучасне середовище розробки мовою C++ Практичне заняття 1. Інструктаж з охорони праці та техніки безпеки. Налаштування середовища розробки, збирання проєкту за допомогою CMake, робота з Git. Повторення базових конструкцій мови.	2
2	2	Тема 2. Класи та об’єкти. Інкапсуляція Практичне заняття 2. Класи та об’єкти. Поля, методи, специфікатори доступу, інкапсуляція. Константні методи.	2
3	3	Тема 2. Класи та об’єкти. Інкапсуляція Практичне заняття 3. Конструктори (за замовчуванням, параметризовані, делегувальні), список ініціалізації, деструктор.	2
4	4	Тема 2. Класи та об’єкти. Інкапсуляція Практичне заняття 4. Статичні члени класу, покажчик this, дружні функції.	2
5	5	Тема 3. Керування ресурсами. Копіювання та переміщення об’єктів Практичне заняття 5. Ідіома RAII. Розроблення класу-обгортки над динамічним масивом. Правило трьох.	2
6	6	Тема 3. Керування ресурсами. Копіювання та переміщення об’єктів Практичне заняття 6. Конструктор і оператор присвоєння копіюванням та переміщенням.	2
7	7	Тема 3. Керування ресурсами. Копіювання та переміщення об’єктів Практичне заняття 7. Правила п’яти та нуля. Використання std::move	2
8	8	Тема 4. Перевантаження операцій. Функціональні об’єкти та лямбда-вирази Практичне заняття 8. Перевантаження арифметичних операцій та операцій порівняння. Оператор <=>.	2
9	9	Тема 4. Перевантаження операцій. Функціональні об’єкти та лямбда-вирази Практичне заняття 9. Перевантаження операцій індексування, виклику функції, інкременту й декременту, потокових операцій. Користувацькі перетворення типів.	2
10	10	Тема 4. Перевантаження операцій. Функціональні об’єкти та лямбда-вирази Практичне заняття 10. Функціональні об’єкти та лямбда-вирази. Використання std::function.	2
11	11	Тема 5. Успадкування Практичне заняття 11. Одиночне та багаторівневе успадкування. Порядок виклику конструкторів і деструкторів.	2
12	12	Тема 5. Успадкування Практичне заняття 12. Види успадкування. Приховування й перевизначення методів. Композиція та агрегація.	2
13	13	Тема 5. Успадкування Практичне заняття 13. Множинне та віртуальне успадкування. Проблема ромба.	2
14	14	Тема 6. Поліморфізм. Абстрактні класи та інтерфейси	2

		Практичне заняття 14. Віртуальні методи, специфікатори override та final. Віртуальний деструктор.	
15	15	Тема 6. Поліморфізм. Абстрактні класи та інтерфейси Практичне заняття 15. Абстрактні класи та інтерфейси. Побудова поліморфної ієрархії.	2
16	16	Тема 6. Поліморфізм. Абстрактні класи та інтерфейси Практичне заняття 16. Інформація про типи під час виконання: dynamic_cast, typeid. Зберігання поліморфних об’єктів у контейнерах. Метод clone().	2
17	17	Тема 7. Розумні вказівники та сучасне керування пам’яттю Практичне заняття 17. Розумні вказівники std::unique_ptr, std::shared_ptr, std::weak_ptr. Заміна «сирих» покажчиків у ієрархії класів.	2
18	18	Тема 8. Опрацювання виключних ситуацій Практичне заняття 18. Генерування та перехоплення виключень. Ієрархія стандартних виключень.	2
19	19	Тема 8. Опрацювання виключних ситуацій Практичне заняття 19. Користувацькі класи виключень. Специфікатор noexcept. Гарантії безпеки щодо виключень.	2
20	20	Тема 8. Опрацювання виключних ситуацій Практичне заняття 20. Застосування std::optional та std::variant. Використання assert і static_assert.	2
21	21	Тема 9. Шаблони функцій і класів. Узагальнене програмування Практичне заняття 21. Шаблони функцій. Виведення типів, перевантаження та спеціалізація шаблонів.	2
22	22	Тема 9. Шаблони функцій і класів. Узагальнене програмування Практичне заняття 22. Шаблони класів. Розроблення шаблонів стека, черги та зв’язного списку.	2
23	23	Тема 9. Шаблони функцій і класів. Узагальнене програмування Практичне заняття 23. Шаблони зі змінною кількістю параметрів. Специфікатор constexpr. Концепти та обмеження шаблонів.	2
24	24	Тема 10. Послідовні контейнери та ітератори STL Практичне заняття 24. Контейнери std::array та std::vector. Ємність і розмір, методи вставлення та видалення.	2
25	25	Тема 10. Послідовні контейнери та ітератори STL Практичне заняття 25. Контейнери std::deque, std::list та std::forward_list. Порівняння ефективності послідовних контейнерів.	2
26	26	Тема 10. Послідовні контейнери та ітератори STL Практичне заняття 26. Ітератори. Інвалідація ітераторів. Ідіома erase–remove. Вибір контейнера.	2
27	27	Тема 11. Асоціативні контейнери та адаптери контейнерів Практичне заняття 27. Контейнери std::set, std::multiset, std::map, std::multimap. Користувацькі компаратори.	2
28	28	Тема 11. Асоціативні контейнери та адаптери контейнерів Практичне заняття 28. Контейнери std::unordered_set та std::unordered_map. Адаптери std::stack, std::queue, std::priority_queue.	2
29	29	Тема 12. Алгоритми STL. Бібліотека діапазонів Практичне заняття 29. Алгоритми пошуку, перетворення та сортування бібліотеки <algorithm>. Лямбда-вирази як предикати.	2

30	30	Тема 12. Алгоритми STL. Бібліотека діапазонів Практичне заняття 30. Числові алгоритми бібліотеки <numeric>. Ітератори-адаптери.	2
31	31	Тема 12. Алгоритми STL. Бібліотека діапазонів Практичне заняття 31. Бібліотека діапазонів <ranges>. Представлення та конвеєри.	2
32	32	Тема 13. Рядки, потоки введення-виведення та файлова система Практичне заняття 32. Опрацювання рядків: std::string, std::string_view, рядкові та файлові потоки. Збереження й завантаження об’єктів. Форматування std::format.	2
33	33	Тема 14. Багатопотокове програмування в C++ Практичне заняття 33. Створення потоків std::thread. Синхронізація за допомогою м’ютексів. Використання std::async та std::future.	2
34	34	Тема 15. Принципи проєктування. Шаблони проєктування. Модульне тестування Практичне заняття 34. Принципи SOLID. Реалізація шаблонів проєктування. Розроблення модульних тестів.	2
35	35	Тема 16. Практикум з програмування: розроблення та захист прикладного проєкту Практичне заняття 35. Розроблення індивідуального проєкту: проєктування, реалізація та тестування застосунку.	2
36	36	Тема 16. Практикум з програмування: розроблення та захист прикладного проєкту Практичне заняття 36. Документування, презентація та захист підсумкового проєкту.	2
Усього			**72**

8. МЕТОДИ НАВЧАННЯ, КОНТРОЛЮ ТА ЗАСОБИ ОЦІНЮВАННЯ

Методи навчання

– словесні (інструктаж, пояснення, бесіда, обговорення розв’язків, консультація);

– наочні (демонстрація роботи середовища розробки та відлагоджувача, показ зразків програмного коду, презентації, відеоматеріали);

– практичні (вправи, практичні роботи, написання, компіляція та відлагодження програмного коду, практико-орієнтовані завдання);

– репродуктивні та частково-пошукові (аналіз готового коду, пошук і виправлення помилок у запропонованих програмах, рефакторинг);

– проблемно-пошукові (самостійна постановка задачі та вибір способу її розв’язання);

– метод проєктів (виконання індивідуального завдання як завершеного програмного продукту);

– робота з технічною документацією, довідковими та мережевими джерелами інформації;

– взаємонавчання та робота в парах під час перевірки й аналізу коду;

– індивідуальні та групові консультації.

Методи контролю

Усне опитування (фронтальне, індивідуальне); тестування на знання синтаксису мови C++; оцінка за виконання та захист практичних робіт; перевірка працездатності та якості розробленого програмного коду; контроль виконання індивідуального завдання; комплексна контрольна робота; самоконтроль; спостереження за навчальною діяльністю.

Засоби оцінювання

– листинги та виконувані файли розроблених програм;

– протоколи тестування програм на контрольних наборах даних;

– презентації результатів виконаних завдань;

– стандартизовані тести з конструкцій мови С++ та засобів стандартної бібліотеки;

– письмове опитування;

– усне фронтальне та індивідуальне опитування;

– навчальні проєкти (прикладний програмний застосунок);

– комплексна контрольна робота.

9. ПОРЯДОК ТА КРИТЕРІЇ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ НАВЧАННЯ

Результати навчання студента, що відображають досягнутий ним рівень компетентностей відносно очікуваних, ідентифікують і вимірюють у процесі контрольних заходів за допомогою критеріїв. Оцінювання студентів охоплює низку письмових, усних та практичних контрольних процедур, які використовують відповідно до компетентнісних характеристик (знання, вміння/навички, комунікація, відповідальність і автономія).

Оцінювання результатів навчання студентів проводиться під час поточного і підсумкового контролю. Поточний контроль здійснюється на кожному практичному занятті у формі усного опитування, комп'ютерного тестування на знання синтаксису мови С++, оцінювання правильності написання та працездатності програмного коду, перевірки виконання індивідуального завдання. Підсумковий контроль проводиться у формі диференційованого заліку в обсязі матеріалу, визначеного програмою.

Оцінювання здійснюється за 5-бальною (національною) шкалою відповідно до чотирьох рівнів навчальних досягнень: початкового (оцінка «2» — «незадовільно»), середнього (оцінка «3» — «задовільно»), достатнього (оцінка «4» — «добре») та високого (оцінка «5» — «відмінно»). Оцінка за кожне практичне заняття виставляється у журнал. Підсумкова оцінка за навчальну дисципліну визначається як середнє арифметичне оцінок поточного контролю з урахуванням оцінок за індивідуальне завдання, комплексну контрольну роботу та захист підсумкового практичного проєкту й округлюється до цілого числа за правилами математичного округлення.

Критерії оцінювання навчальних досягнень студентів за 5-бальною шкалою

Рівень навчальних досягнень	Оцінка	Критерії оцінювання навчальних досягнень
Початковий	**2** «незадовільно»	Студент розпізнає окремі терміни об’єктно-орієнтованого програмування (клас, об’єкт, метод, успадкування), але не може пояснити їх призначення; має фрагментарні уявлення про структуру класу мовою C++; лише за зразком і з допомогою викладача відтворює окремі рядки коду; програма не компілюється; не розуміє призначення специфікаторів доступу й конструкторів; завдання практичного заняття не виконує.
Середній	**3** «задовільно»	Студент під керівництвом викладача створює класи з конструкторами та кількома методами, компілює й запускає програму за зразком; розв’язує прості типові задачі з використанням класів і базових контейнерів (std::vector, std::string); допускає синтаксичні та логічні помилки (плутає специфікатори доступу, порядок виклику конструкторів і деструкторів), які виправляє лише з допомогою викладача; роботу програми пояснює фрагментарно.
Достатній	**4** «добре»	Студент самостійно розробляє класи з конструкторами, перевантаженими операціями, успадкуванням і віртуальними методами; застосовує контейнери й алгоритми STL, шаблони, розумні вказівники та опрацювання виключень; програма працездатна; припускається несуттєвих помилок (наприклад, у керуванні ресурсами чи оформленні коду), які самостійно виявляє й усуває; аргументовано пояснює роботу власної програми, потребує лише окремих уточнень викладача.
Високий	**5** «відмінно»	Студент самостійно розробляє працездатні програми з раціональною структурою класів; коректно керує ресурсами (RAII, правило п’яти, розумні вказівники); обґрунтовано обирає контейнери, алгоритми й ітератори STL, розробляє шаблони класів і функцій, застосовує семантику переміщення, концепти та діапазони; забезпечує безпеку коду щодо виключень, дотримується принципів SOLID; відлагоджує, документує та тестує код модульними тестами; аргументовано пояснює прийняті рішення; виявляє здатність самостійно опановувати новий матеріал; вільно володіє державною мовою.

Критерії оцінювання рівня знань та вмінь студентів при виконанні комплексної контрольної роботи

Високий рівень (оцінка «5» — «відмінно») — студент дає повну, аргументовану відповідь на теоретичні питання, глибоко розуміє принципи об’єктно-орієнтованого програмування та засоби стандартної бібліотеки, вільно складає працездатний код мовою C++, доцільно застосовує класи, успадкування, поліморфізм, шаблони, контейнери й алгоритми STL, розумні вказівники та опрацювання виключень, робить обґрунтовані висновки щодо коректності та ефективності розробленого програмного коду.

Достатній рівень (оцінка «4» — «добре») — студент правильно розкриває більшість питань, але допускає незначні неточності; знає синтаксис мови та принципи об’єктно-орієнтованого програмування, проте частково помиляється у складніших конструкціях (шаблони, семантика переміщення, ітератори); демонструє достатні практичні навички написання коду, але не завжди може оптимізувати розв’язок або вичерпно пояснити його роботу; здатний виконати завдання з невеликими недоліками.

Середній рівень (оцінка «3» — «задовільно») — студент поверхово розкриває теорію, не демонструє системного розуміння принципів об’єктно-орієнтованого програмування, порушує логіку під час складання програмного коду, при виконанні практичних завдань припускається грубих помилок (наприклад, у керуванні пам’яттю та використанні ітераторів), не вміє інтерпретувати результати роботи програми та виправляти помилки, обмежено володіє навичками роботи у середовищі розробки.

Початковий рівень (оцінка «2» — «незадовільно») — студент не розкриває основних теоретичних положень, не володіє базовими поняттями (клас, об’єкт, конструктор, успадкування, поліморфізм, шаблон, контейнер), не здатний виконати практичне завдання, не розуміє алгоритму розв’язання задачі, а програмний код не компілюється або містить критичні помилки, які студент не може виправити.

10. СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна:

1. Бублик В. В. Об’єктно-орієнтоване програмування : підручник. Київ : ІТ-книга, 2015. 624 с.
2. Зеленський О. С., Лисенко В. С. Об’єктно-орієнтоване програмування на С++ : навч. посіб. Кривий Ріг : ДУЕТ, 2023. 215 с.
3. Міловідов Ю. О. Об’єктно-орієнтоване програмування : навч. посіб. 3-тє вид., допов. і переробл. Київ : НУБіП України, 2025. 334 с.
4. С++. Алгоритмізація та програмування : підручник / О. Г. Трофименко та ін. 2-ге вид., перероб. і допов. Одеса : Фенікс, 2019. 477 с.

Додаткова:

5. Васильєв О. Програмування на С++ в прикладах і задачах : навч. посіб. Київ : Ліра-К, 2017. 382 с.
6. Про затвердження стандарту фахової передвищої освіти зі спеціальності 121 «Інженерія програмного забезпечення» галузі знань 12 «Інформаційні технології» для освітньо-професійного ступеня «фаховий молодший бакалавр» : наказ МОН України від 21 верес. 2021 р. № 1006. URL: https://osvita.ua/legislation/faxova-peredvyshha-osvita/85816/ (дата звернення: 29.10.2025).
7. Про фахову передвищу освіту : Закон України від 6 черв. 2019 р. № 2745-VIII. URL: https://zakon.rada.gov.ua/laws/show/2745-19 (дата звернення: 12.11.2025).
8. Stroustrup B. Programming: Principles and Practice Using C++. 3rd ed. Boston : Addison-Wesley, 2024. 656 p.

Інформаційні ресурси:

9. C++ Core Guidelines : вебсайт. URL: https://isocpp.github.io/CppCoreGuidelines/ (дата звернення: 31.10.2025).

10. C++ Insights : вебсайт. URL: https://cppinsights.io (дата звернення: 15.11.2025).
11. Codewars : вебсайт. URL: https://www.codewars.com (дата звернення: 06.11.2025).
12. Compiler Explorer (Godbolt) : вебсайт. URL: https://godbolt.org (дата звернення: 26.09.2025).
13. cplusplus.com : вебсайт. URL: https://cplusplus.com (дата звернення: 25.09.2025).
14. cppreference.com : вебсайт. URL: https://en.cppreference.com (дата звернення: 11.11.2025).
15. e-olymp : вебсайт. URL: https://www.e-olymp.com (дата звернення: 01.11.2025).
16. Exercism. C++ Track : вебсайт. URL: https://exercism.org/tracks/cpp (дата звернення: 09.11.2025).
17. freeCodeCamp : вебсайт. URL: https://www.freecodecamp.org (дата звернення: 14.09.2025).
18. GeeksforGeeks. C++ : вебсайт. URL: https://www.geeksforgeeks.org/c-plus-plus/ (дата звернення: 14.11.2025).
19. Git : вебсайт. URL: https://git-scm.com/doc (дата звернення: 21.09.2025).
20. GitHub : вебсайт. URL: https://github.com (дата звернення: 11.09.2025).
21. GNU Compiler Collection (GCC) : вебсайт. URL: https://gcc.gnu.org (дата звернення: 07.09.2025).
22. HackerRank. C++ : вебсайт. URL: https://www.hackerrank.com/domains/cpp (дата звернення: 06.10.2025).
23. LearnCpp.com : вебсайт. URL: https://www.learncpp.com (дата звернення: 23.10.2025).
24. LeetCode : вебсайт. URL: https://leetcode.com (дата звернення: 28.10.2025).
25. OnlineGDB : вебсайт. URL: https://www.onlinegdb.com (дата звернення: 16.10.2025).

26. Programiz. Learn C++ Programming : вебсайт. URL: https://www.programiz.com/cpp-programming (дата звернення: 12.10.2025).
27. Prometheus : вебсайт. URL: https://prometheus.org.ua (дата звернення: 04.09.2025).
28. Replit : вебсайт. URL: https://replit.com (дата звернення: 10.10.2025).
29. Stack Overflow : вебсайт. URL: https://stackoverflow.com (дата звернення: 27.09.2025).
30. Standard C++ Foundation : вебсайт. URL: https://isocpp.org (дата звернення: 30.09.2025).
31. Visual Studio Code : вебсайт. URL: https://code.visualstudio.com/docs (дата звернення: 13.10.2025).
32. W3Schools. C++ Tutorial : вебсайт. URL: https://www.w3schools.com/cpp/ (дата звернення: 19.10.2025).