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 основ 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 основ 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 № 11 від 07 червня 2024 р.
Голова комісії _____ Катерина СЛІДЗЕВСЬКА

Розглянуто на засіданні Методичної ради
Протокол № 7 від «____» 27.06.2024 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 основ програмування

Освітньо-професійний ступінь	*Фаховий молодший бакалавр*
Галузь знань	*F «Інформаційні технології»*
Спеціальність	*F2 «Інженерія програмного забезпечення»*
Освітньо-професійна програма	*«Інженерія програмного забезпечення»*

Житомир 2025 р.

Навчальна практика з основ 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 № 11 від 07 червня 2025 р.
Голова комісії _____ Дмитро ОЛЕНЮК

Розглянуто на засіданні Методичної ради
Протокол № 7 від «____» 27.06.2025 р.

УКРАЇНА
ЖИТОМИРСЬКИЙ КООПЕРАТИВНИЙ ФАХОВИЙ
КОЛЕДЖ БІЗНЕСУ І ПРАВА

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

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Навчальна практика з основ програмування

Освітньо-професійний ступінь	*Фаховий молодший бакалавр*
Галузь знань	*F «Інформаційні технології»*
Спеціальність	*F2 «Інженерія програмного забезпечення»*
Освітньо-професійна програма	*«Інженерія програмного забезпечення»*

Житомир 2026 р.

Навчальна практика з основ 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 № 11 від 07 червня 2026 р.
Голова комісії _____ Наталія ГОНГАЛО

Розглянуто на засіданні Методичної ради
Протокол № 7 від «____» 27.06.2026 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>72</td><td>–</td><td>–</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>108</td><td>–</td><td>–</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>ДЗ</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
</table>

Форма підсумкового контролю — диференційований залік.

2. МЕТА І ЗАВДАННЯ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Мета навчальної дисципліни: формування у здобувачів освіти стійких практичних навичок алгоритмізації та програмування мовою С++ через безперервну практичну діяльність — від розроблення найпростіших програм до проєктування та реалізації завершеного програмного застосунку із застосуванням об’єктно-орієнтованого підходу як базового інструменту фахівця у сфері інженерії програмного забезпечення.

Завдання навчальної дисципліни:

– закріпити практичні навички роботи в інтегрованому середовищі розробки, з компілятором та відлагоджувачем мови С++;

– сформувати вміння розробляти, компілювати, тестувати й відлагоджувати програми мовою С++;

– відпрацювати практичне застосування базових алгоритмічних конструкцій: лінійних, розгалужених та циклічних;

– сформувати навички опрацювання структурованих даних: одновимірних і двовимірних масивів, символьних рядків, структур та контейнерів стандартної бібліотеки;

– опанувати практичні прийоми декомпозиції задачі на функції та побудови модульної структури програми;

– відпрацювати роботу з покажчиками, посиланнями та динамічним розподілом пам’яті;

– сформувати навички організації введення й виведення даних із зовнішніх файлів та опрацювання виключних ситуацій;

– опанувати практичну реалізацію принципів об’єктно-орієнтованого програмування: інкапсуляції, успадкування та поліморфізму;

– сформувати вміння тестувати, відлагоджувати й документувати власний програмний код;

– виробити навички самостійного виконання індивідуального завдання та презентації результатів власної роботи.

Предметом вивчення навчальної дисципліни є практична діяльність з розроблення програмного забезпечення мовою C++: побудова алгоритмів, реалізація їх у програмному коді, застосування структур даних, функцій, засобів роботи з файлами та об’єктно-орієнтованих конструкцій для розв’язання прикладних задач.

Міждисциплінарні зв’язки: «Основи 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. Середовище розробки мовою С++	6	2	4
2	Структура програми мовою С++. Типи даних, змінні та константи. Введення й виведення даних	12	6	6
3	Операції та вирази. Перетворення типів. Лінійні алгоритми	6	2	4
4	Алгоритми розгалуження. Умовний оператор та оператор вибору	12	6	6
5	Циклічні алгоритми. Оператори циклу	18	6	12
6	Одновимірні масиви. Алгоритми пошуку та сортування	18	6	12
7	Двовимірні масиви. Опрацювання матриць	6	2	4
8	Символьні дані та рядки. Опрацювання текстової інформації	12	6	6
9	Функції. Модульна структура програми	12	6	6
10	Рекурсія. Рекурсивні алгоритми	6	2	4
11	Покажчики, посилання та динамічний розподіл пам’яті	12	6	6
12	Структури, перелічення та контейнери стандартної бібліотеки	6	2	4
13	Робота з файлами. Опрацювання виключних ситуацій. Тестування програм	12	6	6
14	Основи об’єктно-орієнтованого програмування. Класи та об’єкти	18	6	12
15	Успадкування та поліморфізм	12	4	8
16	Практикум з програмування: розроблення та захист прикладного проєкту	12	4	8
Разом		**180**	**72**	**108**

Форма підсумкового контролю — диференційований залік.

6. ЗМІСТ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Тема 1. Вступ до навчальної практики. Середовище розробки мовою С++

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. Історія, призначення та сфери застосування мови С++. Стандарти мови. Поняття компілятора та компонувальника. Огляд сучасних компіляторів (GCC/MinGW, MSVC, Clang) та інтегрованих середовищ розробки (Code::Blocks, Visual Studio, Visual Studio Code, CLion, онлайн-компілятори). Встановлення та налаштування середовища розробки. Створення, збирання та запуск першого проєкту. Ознайомлення із системою контролю версій Git.

Самостійна робота: опрацювання конспекту лекцій та рекомендованої літератури; повторення правил охорони праці та організації робочого місця; підготовка короткого огляду стандартів мови С++ (С++11, С++14, С++17, С++20) та порівняльної таблиці компіляторів і середовищ розробки; самостійне встановлення й налаштування компілятора та середовища розробки на домашньому комп'ютері; створення облікового запису на GitHub, створення локального репозиторію та виконання базових команд Git (init, add, commit, status, log); написання та запуск програми, що виводить інформацію про студента.

Студенти повинні:

знати: мету та завдання навчальної практики; правила охорони праці й техніки безпеки; призначення та сфери застосування мови С++; склад інструментальних засобів розроблення програм;

уміти: організовувати робоче місце; встановлювати й налаштовувати компілятор та середовище розробки; створювати, збирати та запускати проєкт мовою С++.

Тема 2. Структура програми мовою C++. Типи даних, змінні та константи. Введення й виведення даних

Етапи створення програми: редагування, препроцесування, компіляція, компонування, виконання. Структура програми: директиви препроцесора, функція main(), простір імен std, код завершення. Коментарі та правила оформлення програмного коду. Алфавіт і лексеми мови, ідентифікатори та ключові слова. Базові типи даних (int, short, long, long long, unsigned, float, double, char, bool), їхній діапазон і розмір у пам'яті, операція sizeof. Оголошення та ініціалізація змінних. Константи (const, constexpr) і літерали. Потокове введення й виведення: об'єкти cin, cout, cerr. Маніпулятори форматування endl, setw, setprecision, fixed. Особливості зберігання даних у пам'яті комп'ютера.

Самостійна робота: опрацювання конспекту лекцій; вивчення етапів побудови програми та призначення проміжних файлів (.i, .o/.obj, .exe); складання таблиці типів даних із зазначенням розміру та діапазону значень (за допомогою sizeof і бібліотеки <climits>); розв'язання задач на введення, виведення та форматування даних (вивід таблиці з вирівнюванням стовпців за допомогою setw, setprecision, fixed); написання програми-візитівки з введенням даних користувача; дослідження переповнення цілочисельних типів; вивчення правил іменування змінних та оформлення коду.

Студенти повинні:

знати: структуру програми мовою C++; базові типи даних, їхні діапазони та розміри; правила оголошення змінних і константів; засоби потокового введення й виведення та маніпулятори форматування;

уміти: створювати, компілювати та запускати програми; оголошувати й ініціалізувати змінні та константи; організовувати форматоване введення й виведення даних; документувати код коментарями.

Тема 3. Операції та вирази. Перетворення типів. Лінійні алгоритми

Арифметичні операції. Цілочисельне ділення та остача. Операції інкременту й декременту. Складені операції присвоєння. Пріоритет та асоціативність операцій. Неявне та явне перетворення типів, операція static_cast. Математичні функції бібліотеки <cmath>. Генерування псевдовипадкових чисел. Поняття лінійного алгоритму. Розроблення програм, що реалізують обчислення за формулами, геометричні та фізичні задачі, опрацювання цифр числа, переведення одиниць вимірювання.

Самостійна робота: опрацювання конспекту лекцій; вивчення таблиці пріоритетів операцій; обчислення значень виразів вручну з наступною перевіркою програмою; розв'язання задач на обчислення за формулами (геометричні, фізичні, фінансові); програмування опрацювання цифр тризначного та чотиризначного чисел; переведення одиниць вимірювання (час, довжина, температура, швидкість); дослідження втрати точності при неявному перетворенні типів; використання функцій <cmath> та генератора rand()/srand() для моделювання простих випадкових величин; побудова блок-схем лінійних алгоритмів.

Студенти повинні:

знати: систему операцій мови С++ та пріоритет їх виконання; правила перетворення типів даних; склад математичної бібліотеки;

уміти: записувати та обчислювати арифметичні вирази; виконувати перетворення типів даних; застосовувати математичні функції; реалізовувати лінійні алгоритми.

Тема 4. Алгоритми розгалуження. Умовний оператор та оператор вибору

Поняття розгалуженого алгоритму. Операції порівняння. Логічні операції &&, ||, !. Побудова складених логічних умов, скорочене обчислення логічних виразів. Умовний оператор if, конструкції if–else та if–else if.

Вкладені розгалуження. Умовна (тернарна) операція ?:. Оператор вибору switch–case, роль операторів break і default. Побудова блок-схем розгалужених алгоритмів. Оптимізація складних логічних умов.

Самостійна робота: опрацювання конспекту лекцій; складання таблиць істинності для логічних операцій; розв'язання задач на складені умови (належність числа проміжку, тип трикутника, високосний рік, розв'язання квадратного рівняння, визначення чверті координатної площини); розроблення програми-калькулятора з використанням switch; перетворення конструкції if–else if у switch та навпаки; побудова блок-схем розгалужених алгоритмів у редакторі (draw.io або аналог); дослідження скороченого обчислення логічних виразів на прикладах; спрощення складних логічних умов.

Студенти повинні:

знати: поняття розгалуженого алгоритму; синтаксис операторів if та switch; правила побудови логічних виразів; правила побудови блок-схем;

уміти: складати програми з розгалуженням; застосовувати вкладені умови та оператор вибору; опрацьовувати логічні вирази; будувати блок-схеми алгоритмів.

Тема 5. Циклічні алгоритми. Оператори циклу

Поняття циклічного алгоритму. Оператор циклу з параметром for та його складові частини. Цикл з передумовою while. Цикл з постумовою do–while. Вибір доцільного виду циклу для конкретної задачі. Оператори керування циклом break і continue. Вкладені цикли. Табулювання функцій, обчислення сум і добутків числових послідовностей, ітераційні обчислення із заданою точністю, перебір варіантів, перевірка чисел на простоту, знаходження найбільшого спільного дільника, побудова символьних зображень. Типові помилки під час роботи з циклами (нескінченні цикли, помилки межових

значень) та способи їх усунення. Покрокове відлагодження програм з використанням точок зупину та спостереження за значеннями змінних.

Самостійна робота: опрацювання конспекту лекцій; ручне виконання циклів із заповненням таблиці трасування значень змінних; розв’язання задач на табулювання функцій, обчислення сум і добутків, наближене обчислення значень функцій рядами із заданою точністю; програмування перевірки числа на простоту та пошуку НСД (алгоритм Евкліда); виведення фігур із символів (трикутники, ромб, ялинка) за допомогою вкладених циклів; пошук і виправлення навмисно допущених помилок у циклах (нескінченний цикл, помилка на одиницю); освоєння роботи з відлагоджувачем (точки зупину, покроковий режим, вікно спостереження за змінними).

Студенти повинні:

знати: види циклічних алгоритмів; синтаксис операторів for, while та do–while; призначення операторів керування циклом; типові помилки під час програмування циклів;

уміти: складати програми з циклами; обирати доцільний вид циклу; програмувати вкладені цикли; застосовувати відлагоджувач для пошуку логічних помилок.

Тема 6. Одновимірні масиви. Алгоритми пошуку та сортування

Поняття масиву як однорідної структури даних. Оголошення, ініціалізація та індексація одновимірних масивів. Введення й виведення елементів масиву. Типові алгоритми опрацювання: обчислення суми, добутку, кількості та середнього значення елементів; пошук максимального й мінімального елементів та їхніх індексів; підрахунок елементів за умовою; реверс, циклічний зсув, вставлення та вилучення елементів; злиття масивів. Лінійний та двійковий пошук. Алгоритми сортування: бульбашкове, вибором, вставленням; порівняння їхньої ефективності. Вихід за межі масиву як типова помилка.

Самостійна робота: опрацювання конспекту лекцій; розв’язання задач на типові алгоритми опрацювання масивів (заповнення випадковими числами, статистика, пошук екстремумів, реверс, зсув, вилучення та вставлення елементів); програмування лінійного та двійкового пошуку з підрахунком кількості порівнянь; реалізація трьох алгоритмів сортування (бульбашкове, вибором, вставленням) та експериментальне порівняння кількості операцій і часу виконання на масивах різного розміру; дослідження наслідків виходу за межі масиву; оформлення порівняльної таблиці ефективності алгоритмів.

Студенти повинні:

знати: поняття масиву; правила оголошення й індексації одновимірних масивів; типові алгоритми опрацювання масивів; алгоритми пошуку та сортування;

уміти: оголошувати, заповнювати та виводити масиви; реалізовувати типові алгоритми опрацювання масивів; програмувати алгоритми пошуку та сортування й порівнювати їхню ефективність.

Тема 7. Двовимірні масиви. Опрацювання матриць

Поняття двовимірного масиву. Оголошення, ініціалізація, введення й виведення елементів. Обхід матриці за рядками та стовпцями. Опрацювання головної й допоміжної діагоналей. Транспонування матриці. Додавання та множення матриць. Пошук екстремальних елементів у рядках і стовпцях. Обчислення сум і середніх значень за рядками та стовпцями. Прикладні задачі з використанням таблиць даних.

Самостійна робота: опрацювання конспекту лекцій; повторення основ матричної алгебри (додавання, множення, транспонування); розв’язання задач на обхід матриці за рядками, стовпцями та діагоналями; програмування транспонування, додавання й множення матриць із перевіркою сумісності розмірів; пошук екстремальних елементів, сум і середніх значень за рядками та стовпцями; розв’язання прикладних задач із табличними даними (відомість

оцінок, таблиця продажів, температурні спостереження); виведення матриці у відформатованому вигляді; побудова магічного квадрата або заповнення матриці «змійкою» та по спіралі.

Студенти повинні:

знати: правила оголошення та індексації двовимірних масивів; способи обходу матриці; типові алгоритми опрацювання матриць;

уміти: створювати та опрацьовувати двовимірні масиви; виконувати операції над матрицями; розв’язувати прикладні задачі з табличними даними.

Тема 8. Символьні дані та рядки. Опрацювання текстової інформації

Символьний тип char. Таблиці кодування ASCII та Unicode. Функції бібліотеки <cctype>. Рядки у стилі мови C: масиви символів, нуль-термінатор, функції бібліотеки <cstring>. Клас std::string та його методи (length, size, substr, find, replace, insert, erase, compare, c_str). Конкатенація, порівняння та перетворення рядків. Перебір символів рядка. Опрацювання тексту: підрахунок слів, символів і речень, зміна регістру, пошук і заміна підрядків, перевірка на паліндром, розбиття рядка на лексеми, найпростіше шифрування тексту.

Самостійна робота: опрацювання конспекту лекцій; вивчення таблиці ASCII та особливостей кодування кирилиці; розв’язання задач із використанням функцій <cctype> (підрахунок літер, цифр, пробілів, зміна регістру); реалізація власних версій функцій <cstring> (довжина, копіювання, порівняння рядків); програмування підрахунку слів і речень, пошуку та заміни підрядка, перевірки на паліндром; розбиття рядка на слова; реалізація шифру Цезаря (шифрування та розшифрування); порівняння рядків у стилі C та std::string за зручністю й безпекою використання.

Студенти повинні:

знати: поняття символьного типу та рядка; особливості рядків у стилі С і класу std::string; основні методи опрацювання рядків;

уміти: виконувати операції з рядками; застосовувати методи опрацювання тексту; розв’язувати прикладні задачі опрацювання текстової інформації.

Тема 9. Функції. Модульна структура програми

Поняття функції та підпрограми. Декомпозиція задачі на підзадачі. Прототип (оголошення) та визначення функції. Формальні й фактичні параметри, оператор return, функції типу void. Локальні та глобальні змінні, область видимості й час життя, статичні локальні змінні. Передавання параметрів за значенням, за посиланням і через покажчик. Параметри зі значеннями за замовчуванням. Перевантаження функцій. Вбудовані (inline) функції. Шаблони функцій (ознайомлення). Передавання масивів у функції. Розподіл програми на кілька файлів: заголовкові файли (.h) та файли реалізації (.cpp), охорона включення (include guard, #pragma once). Простори імен. Збирання багатофайлового проєкту.

Самостійна робота: опрацювання конспекту лекцій; переробка раніше написаних програм (масиви, рядки, цикли) із виділенням логіки у функції; розв’язання задач на різні способи передавання параметрів (за значенням, за посиланням, через покажчик) із поясненням результатів; дослідження області видимості та статичних локальних змінних (лічильник викликів); написання перевантажених функцій і функцій із параметрами за замовчуванням; створення власної бібліотеки функцій (наприклад, mathutils.h / mathutils.cpp) із include guard; збирання багатофайлового проєкту та аналіз помилок компонування.

Студенти повинні:

знати: поняття функції, її прототипу та визначення; способи передавання параметрів; правила області видимості та часу життя змінних; принципи побудови багатофайлового проєкту;

уміти: декомпозувати задачу на функції; розробляти та використовувати власні функції; застосовувати перевантаження та параметри за замовчуванням; організовувати програму у вигляді багатофайлового проєкту.

Тема 10. Рекурсія. Рекурсивні алгоритми

Поняття рекурсії. Базовий випадок і крок рекурсії. Глибина рекурсії та стек викликів. Класичні рекурсивні алгоритми: обчислення факторіала, чисел Фібоначчі, найбільшого спільного дільника, ханойські башти, швидке піднесення до степеня, реверс рядка, двійковий пошук. Пряма та непряма рекурсія. Порівняння рекурсивної та ітеративної реалізацій за швидкодією та витратами пам’яті. Типові помилки під час програмування рекурсії.

Самостійна робота: опрацювання конспекту лекцій; трасування рекурсивних викликів на папері (схема стеку викликів для факторіала та чисел Фібоначчі); програмування рекурсивних функцій (сума цифр числа, степінь, НСД, реверс рядка, перевірка паліндрома, двійковий пошук); реалізація задачі про ханойські башти з підрахунком кількості переміщень; порівняння рекурсивної та ітеративної версій за часом виконання (наприклад, числа Фібоначчі) і формулювання висновків; дослідження переповнення стеку при відсутності базового випадку; рефакторинг ітеративних алгоритмів у рекурсивні.

Студенти повинні:

знати: поняття рекурсії, базового випадку та кроку рекурсії; класичні рекурсивні алгоритми; переваги й недоліки рекурсивного підходу;

уміти: розробляти рекурсивні функції; реалізовувати класичні рекурсивні алгоритми; обирати між рекурсивною та ітеративною реалізацією.

Тема 11. Покажчики, посилання та динамічний розподіл пам’яті

Поняття адреси та покажчика. Оголошення покажчиків, операції взяття адреси та розіменування. Арифметика покажчиків. Зв’язок покажчиків і масивів. Нульовий покажчик. Покажчики на функції (ознайомлення). Посилання та їхні відмінності від покажчиків. Динамічний розподіл пам’яті: операції new, delete, new[], delete[]. Динамічні одновимірні та двовимірні масиви. Поняття витоку пам’яті та способи його запобігання. Передавання динамічних структур у функції.

Самостійна робота: опрацювання конспекту лекцій; малювання схем розміщення змінних і покажчиків у пам’яті; виведення адрес змінних і елементів масиву, дослідження арифметики покажчиків; розв’язання задач на обробку масивів через покажчики; реалізація функцій обміну значень через покажчик і через посилання; створення динамічного одновимірного масиву розміром, заданим користувачем, та динамічної матриці з коректним звільненням пам’яті; дослідження витоку пам’яті (навмисне створення та виправлення помилки); вивчення типових помилок (висячий покажчик, подвійне звільнення пам’яті, розіменування nullptr); ознайомлення з покажчиками на функції на простому прикладі.

Студенти повинні:

знати: поняття покажчика та посилання; правила арифметики покажчиків; механізм динамічного розподілу пам’яті; поняття витоку пам’яті;

уміти: оголошувати та використовувати покажчики й посилання; створювати та коректно звільняти динамічні масиви; передавати динамічні структури у функції.

Тема 12. Структури, перелічення та контейнери стандартної бібліотеки

Поняття структури як складеного типу даних. Оголошення структури, доступ до полів, ініціалізація. Вкладені структури. Масиви структур та їх опрацювання: введення, виведення, пошук, фільтрація та сортування записів за полем. Передавання структур у функції. Перелічувані типи (enum, enum class). Псевдоніми типів (typedef, using). Огляд стандартної бібліотеки шаблонів (STL). Контейнер std::vector: створення, додавання й вилучення елементів, ітератори, обхід за допомогою циклу range-based for. Ознайомлення з контейнерами std::array, std::map, std::set. Алгоритми бібліотеки <algorithm>: sort, find, count, max_element, reverse, accumulate.

Самостійна робота: опрацювання конспекту лекцій; розроблення структур для предметних областей (студент, товар, книга, співробітник) та масивів записів; реалізація введення, виведення, пошуку, фільтрації й сортування записів за обраним полем; використання enum class для кодування станів і категорій; переписування задач на масиви із застосуванням std::vector; розв’язання задач із використанням std::map (підрахунок частоти слів) та std::set (унікальні елементи); застосування алгоритмів sort, find, count, max_element, reverse, accumulate замість ручних циклів; порівняння роботи з масивом, vector та array за зручністю й безпекою.

Студенти повинні:

знати: поняття структури та перелічуваного типу; склад і призначення контейнерів стандартної бібліотеки шаблонів; основні алгоритми бібліотеки <algorithm>;

уміти: оголошувати та опрацьовувати структури й масиви структур; застосовувати контейнери та алгоритми стандартної бібліотеки; обирати доцільну структуру даних для розв’язання задачі.

Тема 13. Робота з файлами. Опрацювання виключних ситуацій. Тестування програм

Поняття файлу. Файлові потоки бібліотеки <fstream>: ifstream, ofstream, fstream. Режими відкриття файлів, перевірка успішності відкриття. Запис даних у текстовий файл і читання з нього. Порядкове опрацювання файлу, контроль кінця файлу. Опрацювання табличних даних формату CSV. Основи роботи з двійковими файлами. Види помилок: синтаксичні, помилки компонування, логічні та помилки виконання. Механізм виключних ситуацій: try, catch, throw; стандартні класи виключень. Перевірка коректності вхідних даних, принципи захисного програмування. Тестування програми: підбір тестових наборів, межові значення. Використання відлагоджувача й засобів діагностики компілятора.

Самостійна робота: опрацювання конспекту лекцій; розв'язання задач на запис результатів обчислень у текстовий файл і читання даних із файлу (статистика числових даних, підрахунок слів); програмування опрацювання CSV-файлу (завантаження таблиці, обчислення підсумків, збереження результату); запис і читання масиву структур у двійковому файлі; написання функцій із перевіркою коректності введення (нечислове значення, значення поза діапазоном, відсутній файл) та генерацією виключень; складання тестових наборів (типові, межові та некоректні значення) для власних програм із таблицею очікуваних результатів; пошук помилок за допомогою відлагоджувача та попереджень компілятора.

Студенти повинні:

знати: засоби роботи з файловими потоками та режими доступу до файлів; види помилок і механізм опрацювання виключних ситуацій; принципи тестування та відлагодження програм;

уміти: зчитувати та записувати дані у текстові файли; опрацьовувати дані формату CSV; опрацьовувати виключні ситуації та перевіряти коректність вхідних даних; складати тестові набори й відлагоджувати програми.

Тема 14. Основи об’єктно-орієнтованого програмування. Класи та об’єкти

Обмеження структурного підходу та передумови виникнення об’єктно-орієнтованого програмування. Базові поняття: абстракція, інкапсуляція, успадкування, поліморфізм. Клас як тип даних, об’єкт як екземпляр класу. Оголошення класу, дані-члени (поля) та функції-члени (методи). Специфікатори доступу private, protected, public. Реалізація методів у тілі класу та за його межами. Інкапсуляція, методи доступу та модифікації. Конструктори: за замовчуванням, з параметрами, копіювання; список ініціалізації членів. Деструктор і звільнення ресурсів. Покажчик this. Статичні поля та статичні методи. Константні методи. Дружні функції. Перевантаження операцій: арифметичних, порівняння, присвоєння, потокових операцій << і >>. Композиція та агрегація об’єктів.

Самостійна робота: опрацювання конспекту лекцій; перетворення раніше розробленої структури (наприклад, «Студент») на клас із закритими полями, методами доступу та валідацією значень; розроблення класів предметної області (Рахунок, Дріб, Вектор, Комплексне число, Точка); реалізація різних видів конструкторів (за замовчуванням, з параметрами, копіювання) та деструктора з виведенням повідомлень для спостереження за порядком створення й знищення об’єктів; використання статичних полів (лічильник об’єктів) і константних методів; перевантаження операцій +, −, ==, = та << і >> для власного класу; побудова класу з композицією об’єктів; розділення класу на файли .h та .cpp.

Студенти повинні:

знати: основні поняття об’єктно-орієнтованого програмування; структуру класу та призначення специфікаторів доступу; призначення конструкторів, деструктора та покажчика this; правила перевантаження операцій;

уміти: оголошувати класи, поля й методи та створювати об’єкти; застосовувати інкапсуляцію та методи доступу; реалізовувати конструктори й деструктор; перевантажувати операції.

Тема 15. Успадкування та поліморфізм

Успадкування як механізм повторного використання коду. Базовий і похідний класи. Види успадкування (public, protected, private). Порядок виклику конструкторів і деструкторів під час успадкування. Перевизначення методів та звертання до членів базового класу. Багаторівневе успадкування. Поліморфізм: віртуальні методи, механізм пізнього зв’язування, таблиця віртуальних методів. Чисто віртуальні методи та абстрактні класи. Віртуальний деструктор. Побудова простої діаграми класів. Принципи об’єктно-орієнтованого проєктування.

Самостійна робота: опрацювання конспекту лекцій; побудова діаграм класів (UML) для обраної предметної області; розроблення ієрархії класів (наприклад, Фігура → Коло, Прямокутник, Трикутник; Співробітник → Менеджер, Інженер) з використанням успадкування; спостереження за порядком виклику конструкторів і деструкторів у багаторівневій ієрархії; перевизначення методів базового класу та виклик їх через покажчик і посилання на базовий клас із virtual і без нього, порівняння результатів; створення абстрактного класу з чисто віртуальними методами та масиву покажчиків на базовий клас для поліморфної обробки об’єктів; демонстрація проблеми невіртуального деструктора та її усунення.

Студенти повинні:

знати: механізми успадкування та поліморфізму; види успадкування; призначення віртуальних методів і віртуального деструктора; поняття абстрактного класу;

уміти: будувати ієрархію класів; перевизначати методи базового класу; реалізовувати поліморфну поведінку; застосовувати об’єктно-орієнтований підхід для розв’язання прикладних задач.

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

Етапи розроблення програмного продукту. Постановка задачі та формулювання функціональних вимог. Проєктування структури програми: виділення класів, функцій і структур даних, розроблення схеми меню. Інтеграція вивчених конструкцій у завершений проєкт. Реалізація прикладного застосунку із застосуванням об’єктно-орієнтованого підходу, файлового зберігання даних та опрацювання помилок. Тестування й відлагодження програми. Документування коду та підготовка інструкції користувача. Виконання індивідуального завдання. Презентація та захист результатів роботи.

Самостійна робота: вибір теми індивідуального проєкту та узгодження її з викладачем; формулювання функціональних вимог і складання плану роботи; проєктування структури програми (діаграма класів, схема меню, формат файлів даних); поетапна реалізація проєкту з використанням системи контролю версій Git (регулярні коміти з осмисленими повідомленнями); розроблення тестових наборів і тестування програми, виправлення виявлених помилок; коментування коду та підготовка інструкції користувача; підготовка презентації та короткої доповіді для захисту проєкту; репетиція захисту та підготовка відповідей на контрольні запитання.

Студенти повинні:

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

уміти: розробляти завершені програми; інтегрувати різні мовні конструкції 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. Середовище розробки мовою С++ Практичне заняття 1. Інструктаж з охорони праці та техніки безпеки. Налаштування середовища розробки та написання першої програми мовою С++.	2
2	2	Тема 2. Структура програми мовою С++. Типи даних, змінні та константи Практичне заняття 2. Структура програми. Типи даних, оголошення змінних і константів. Визначення розміру типів у пам'яті.	2
3	3	Тема 2. Структура програми мовою С++. Типи даних, змінні та константи Практичне заняття 3. Організація введення й виведення даних. Форматування результатів за допомогою маніпуляторів.	2
4	4	Тема 2. Структура програми мовою С++. Типи даних, змінні та константи Практичне заняття 4. Розв'язання простих задач з використанням змінних різних типів. Оформлення програмного коду та коментарі.	2
5	5	Тема 3. Операції та вирази. Перетворення типів. Лінійні алгоритми Практичне заняття 5. Розв'язання задач на обчислення з використанням змінних та операцій. Реалізація лінійних алгоритмів.	2
6	6	Тема 4. Алгоритми розгалуження. Умовний оператор та оператор вибору Практичне заняття 6. Розв'язання задач із використанням умовного оператора та вкладених розгалужень.	2
7	7	Тема 4. Алгоритми розгалуження. Умовний оператор та оператор вибору Практичне заняття 7. Програмування з оператором вибору switch. Побудова блок-схем розгалужених алгоритмів.	2
8	8	Тема 4. Алгоритми розгалуження. Умовний оператор та оператор вибору Практичне заняття 8. Розв'язання простих задач із розгалуженням: порівняння чисел, визначення парності та знака числа.	2
9	9	Тема 5. Циклічні алгоритми. Оператори циклу Практичне заняття 9. Розв'язання задач із використанням циклу з параметром for. Табулювання функцій.	2
10	10	Тема 5. Циклічні алгоритми. Оператори циклу Практичне заняття 10. Розв'язання задач із використанням циклів while та do–while. Ітераційні обчислення із заданою точністю.	2
11	11	Тема 5. Циклічні алгоритми. Оператори циклу Практичне заняття 11. Програмування вкладених циклів. Покрокове відлагодження програм.	2
12	12	Тема 6. Одновимірні масиви. Алгоритми пошуку та сортування	2

		Практичне заняття 12. Розв’язання задач на обробку одновимірних масивів.	
13	13	Тема 6. Одновимірні масиви. Алгоритми пошуку та сортування Практичне заняття 13. Реалізація алгоритмів лінійного та двійкового пошуку.	2
14	14	Тема 6. Одновимірні масиви. Алгоритми пошуку та сортування Практичне заняття 14. Реалізація алгоритмів сортування масиву та порівняння їхньої ефективності.	2
15	15	Тема 7. Двовимірні масиви. Опрацювання матриць Практичне заняття 15. Робота з двовимірними масивами. Виконання операцій над матрицями.	2
16	16	Тема 8. Символьні дані та рядки. Опрацювання текстової інформації Практичне заняття 16. Розв’язання задач на обробку символьних даних та рядків.	2
17	17	Тема 8. Символьні дані та рядки. Опрацювання текстової інформації Практичне заняття 17. Опрацювання текстових даних: статистика тексту, пошук і заміна підрядків.	2
18	18	Тема 8. Символьні дані та рядки. Опрацювання текстової інформації Практичне заняття 18. Розбиття рядка на слова (лексеми). Перевірка рядка на паліндром. Реалізація шифру Цезаря.	2
19	19	Тема 9. Функції. Модульна структура програми Практичне заняття 19. Розроблення програм із використанням власних функцій. Передавання параметрів.	2
20	20	Тема 9. Функції. Модульна структура програми Практичне заняття 20. Перевантаження функцій, параметри зі значеннями за замовчуванням. Передавання масивів у функції. Область видимості та статичні локальні змінні.	2
21	21	Тема 9. Функції. Модульна структура програми Практичне заняття 21. Створення багатофайлового проєкту та власного заголовкового файлу.	2
22	22	Тема 10. Рекурсія. Рекурсивні алгоритми Практичне заняття 22. Реалізація рекурсивних алгоритмів та порівняння їх з ітеративними.	2
23	23	Тема 11. Покажчики, посилання та динамічний розподіл пам’яті Практичне заняття 23. Розв’язання задач із використанням покажчиків і посилань.	2
24	24	Тема 11. Покажчики, посилання та динамічний розподіл пам’яті Практичне заняття 24. Використання покажчиків для доступу до елементів масиву. Обмін значень змінних через посилання.	2
25	25	Тема 11. Покажчики, посилання та динамічний розподіл пам’яті Практичне заняття 25. Створення та опрацювання динамічних одновимірних і двовимірних масивів.	2
26	26	Тема 12. Структури, перелічення та контейнери стандартної бібліотеки Практичне заняття 26. Розв’язання задач із використанням структур, масивів структур та контейнера std::vector.	2
27	27	Тема 13. Робота з файлами. Опрацювання виключних ситуацій	2

		Практичне заняття 27. Розв’язання задач на читання та запис даних у файли. Опрацювання даних формату CSV.	
28	28	Тема 13. Робота з файлами. Опрацювання виключних ситуацій Практичне заняття 28. Запис і читання масиву структур у двійкових файлах. Порядкове опрацювання текстових файлів із контролем кінця файлу.	2
29	29	Тема 13. Робота з файлами. Опрацювання виключних ситуацій Практичне заняття 29. Розроблення програм з опрацюванням виключних ситуацій та перевіркою коректності даних.	2
30	30	Тема 14. Основи об’єктно-орієнтованого програмування. Класи та об’єкти Практичне заняття 30. Розроблення програм із використанням класів та об’єктів. Інкапсуляція та методи доступу.	2
31	31	Тема 14. Основи об’єктно-орієнтованого програмування. Класи та об’єкти Практичне заняття 31. Реалізація конструкторів, деструктора та статичних членів класу.	2
32	32	Тема 14. Основи об’єктно-орієнтованого програмування. Класи та об’єкти Практичне заняття 32. Перевантаження операцій. Розроблення класу з перевантаженими операціями.	2
33	33	Тема 15. Успадкування та поліморфізм Практичне заняття 33. Побудова ієрархії класів із застосуванням успадкування.	2
34	34	Тема 15. Успадкування та поліморфізм Практичне заняття 34. Реалізація поліморфізму за допомогою віртуальних методів та абстрактних класів.	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 практичного 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 базових конструкцій мови С++;

– 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 C++, 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 12-бальною шкалою відповідно до чотирьох рівнів навчальних досягнень: початкового (1–3 бали), середнього (4–6 балів), достатнього (7–9 балів) та високого (10–12 балів). 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.

Критерії оцінювання навчальних досягнень студентів за 12-бальною шкалою

Рівень навчальних досягнень	Бали	Критерії оцінювання навчальних досягнень
Початковий	**1**	Студент розпізнає окремі елементи мови програмування (ключові слова, оператори) серед запропонованих, але не може пояснити їх призначення; не володіє середовищем розробки.
	2	Студент має фрагментарні уявлення про структуру програми мовою C++; за детальною допомогою викладача відтворює окремі рядки коду за зразком; програма не компілюється.
	3	Студент за зразком набирає найпростішу програму, але не може самостійно її скомпілювати та запустити; не розуміє призначення основних конструкцій мови; завдання практичного заняття не виконує.
Середній	**4**	Студент під керівництвом викладача створює та запускає найпростішу програму лінійної структури; допускає численні синтаксичні помилки, які усуває лише з допомогою; роботу програми пояснює фрагментарно.
	5	Студент самостійно набирає й компілює програму за зразком, розуміє призначення змінних та операцій введення-виведення, проте плутає назви функцій і синтаксичні конструкції; помилки в алгоритмі виправляє з допомогою викладача.
	6	Студент розв'язує прості типові задачі за наведеним зразком; знає базові типи даних та керуючі конструкції, але не завжди правильно їх застосовує; не вміє самостійно локалізувати помилку в коді.
Достатній	**7**	Студент самостійно розв'язує типові задачі з використанням розгалужень і циклів; програма працездатна, проте код неоптимальний і слабо документований; потребує окремих уточнень викладача.
	8	Студент самостійно складає алгоритм і реалізує його мовою C++, застосовує масиви, рядки та функції; припускається несуттєвих помилок, які самостійно виявляє й усуває; аргументовано пояснює роботу власної програми.
	9	Студент упевнено застосовує вивчені конструкції мови, у тому числі структури даних та файлове введення-виведення; код оформлений і документований; неточності допускає лише у складніших завданнях.
Високий	**10**	Студент самостійно розробляє працездатні програми з раціональною структурою, застосовує функції, динамічну пам'ять та класи; тестує програму на різних наборах даних; допускає окремі неточності в оформленні або документуванні коду.
	11	Студент вільно володіє синтаксисом мови C++, обґрунтовано обирає структури даних і алгоритми, реалізує об'єктно-орієнтований підхід, самостійно відлагоджує програму та аргументовано пояснює прийняті рішення.
	12	Студент виконує завдання підвищеної складності, пропонує власні раціональні розв'язки, проєктує ієрархію класів, забезпечує надійне опрацювання помилок, документує код і презентує результат; виявляє творчий підхід та здатність самостійно опановувати новий матеріал; вільно володіє державною мовою.

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

Високий рівень (10–12 балів) — студент дає повну, аргументовану відповідь на теоретичні питання, глибоко розуміє принципи алгоритмізації та структуру програми, вільно складає працездатний код мовою С++, доцільно застосовує розгалуження, цикли, масиви, рядки, функції та структури даних, демонструє вміння розробляти модульні й об’єктно-орієнтовані програми та опрацьовувати виключні ситуації, робить обґрунтовані висновки щодо коректності та ефективності розробленого алгоритму.

Достатній рівень (7–9 балів) — студент правильно розкриває більшість питань, але допускає незначні неточності; знає синтаксис мови та принципи побудови алгоритмів, проте частково помиляється у складніших конструкціях; демонструє достатні практичні навички написання коду, але не завжди може оптимізувати алгоритм або вичерпно пояснити його роботу; здатний виконати завдання з невеликими недоліками.

Середній рівень (4–6 балів) — студент поверхово розкриває теорію, не демонструє системного розуміння принципів програмування, порушує логіку під час складання програмного коду, при виконанні практичних завдань припускається грубих помилок у реалізації алгоритму, не вміє інтерпретувати результати роботи програми та виправляти помилки, обмежено володіє навичками роботи у середовищі розробки.

Початковий рівень (1–3 бали) — студент не розкриває основних теоретичних положень, не володіє базовими поняттями (змінні, типи даних, оператори, цикли, масиви, функції, класи), 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. Зеленський О. С., Лисенко В. С. Об’єктно-орієнтоване програмування на C++ : навч. посіб. Кривий Ріг : ДУЕТ, 2023. 215 с.
2. Ярошко С. А., Ярошко О. С. Методи розробки алгоритмів. Програмування мовою C++ : навч. посіб. Львів : ЛНУ імені Івана Франка, 2022. 248 с.

Додаткова:

3. Міловідов Ю. О. Об’єктно-орієнтоване програмування : навч. посіб. 3-тє вид., допов. і переробл. Київ : НУБіП України, 2025. 334 с.
4. Stroustrup B. Programming: Principles and Practice Using C++. 3rd ed. Boston : Addison-Wesley, 2024. 656 p.

Інформаційні ресурси:

5. C++ Core Guidelines : вебсайт. URL: https://isocpp.github.io/CppCoreGuidelines/ (дата звернення: 11.09.2025).
6. C++ Insights : вебсайт. URL: https://cppinsights.io (дата звернення: 05.09.2025).
7. Codewars : вебсайт. URL: https://www.codewars.com (дата звернення: 22.10.2025).
8. Compiler Explorer (Godbolt) : вебсайт. URL: https://godbolt.org (дата звернення: 28.09.2025).
9. cplusplus.com : вебсайт. URL: https://cplusplus.com (дата звернення: 21.09.2025).
10. cppreference.com : вебсайт. URL: https://en.cppreference.com (дата звернення: 29.09.2025).
11. e-olymp : вебсайт. URL: https://www.e-olymp.com (дата звернення: 18.10.2025).

12. Exercism. C++ Track : вебсайт. URL: https://exercism.org/tracks/cpp (дата звернення: 13.10.2025).
13. freeCodeCamp : вебсайт. URL: https://www.freecodecamp.org (дата звернення: 03.11.2025).
14. GeeksforGeeks. C++ : вебсайт. URL: https://www.geeksforgeeks.org/c-plus-plus/ (дата звернення: 14.09.2025).
15. Git : вебсайт. URL: https://git-scm.com/doc (дата звернення: 09.09.2025).
16. GitHub : вебсайт. URL: https://github.com (дата звернення: 07.09.2025).
17. GNU Compiler Collection (GCC) : вебсайт. URL: https://gcc.gnu.org (дата звернення: 08.10.2025).
18. HackerRank. C++ : вебсайт. URL: https://www.hackerrank.com/domains/cpp (дата звернення: 17.09.2025).
19. LearnCpp.com : вебсайт. URL: https://www.learncpp.com (дата звернення: 24.10.2025).
20. LeetCode : вебсайт. URL: https://leetcode.com (дата звернення: 16.09.2025).
21. OnlineGDB : вебсайт. URL: https://www.onlinegdb.com (дата звернення: 05.11.2025).
22. Programiz. Learn C++ Programming : вебсайт. URL: https://www.programiz.com/cpp-programming (дата звернення: 07.10.2025).
23. Prometheus : вебсайт. URL: https://prometheus.org.ua (дата звернення: 07.11.2025).
24. Replit : вебсайт. URL: https://replit.com (дата звернення: 06.11.2025).
25. Stack Overflow : вебсайт. URL: https://stackoverflow.com (дата звернення: 06.09.2025).
26. Standard C++ Foundation : вебсайт. URL: https://isocpp.org (дата звернення: 02.11.2025).
27. Visual Studio Code : вебсайт. URL: https://code.visualstudio.com/docs (дата звернення: 09.11.2025).