

Чернівецький національний університет імені Юрія Федьковича

(повне найменування закладу вищої освіти)

Факультет математики та інформатики

(назва факультету/навчально-наукового інституту)

Кафедра математичного моделювання

(назва кафедри, що забезпечує викладання)

“ЗАТВЕРДЖУЮ”

**Декан факультету
математики та інформатики**

Ольга МАРТИНЮК

“28” серпня 2026 року

РОБОЧА ПРОГРАМА

навчальної дисципліни

Програмування

(назва навчальної дисципліни)

обов’язкова

(вказати: обов’язкова)

Освітньо-професійна програма Комп’ютерні науки та проєктування програмних систем

(назва програми)

Спеціальність Ф3 Комп’ютерні науки

(вказати: код, назва)

Галузь знань Ф Інформаційні технології

(вказати: шифр, назва)

Рівень вищої освіти перший (бакалаврський)

(вказати: перший (бакалаврський) / другий (магістерський) / третій (освітньо-науковий))

Факультет математики та інформатики

(назва факультету/ навчально-наукового інституту, на якому здійснюється підготовка фахівців за вказаною освітньою програмою)

Мова навчання українська

(вказати: на якій мові читається дисципліна)

Чернівці 2026 рік

Робоча програма навчальної дисципліни «Програмування» складена відповідно до освітньо-професійної програми «Комп'ютерні науки та проектування програмних систем»

Розробники:

Івасюк Галина Петрівна, доцент кафедри математичного моделювання, кандидат фіз.-мат. наук, доцент

Викладачі, що забезпечують читання даної навчальної дисципліни:

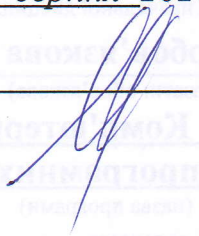
Івасюк Галина Петрівна, доцент кафедри математичного моделювання, кандидат фіз.-мат. наук, доцент;

Косован Василь Михайлович, асистент кафедри математичного моделювання, кандидат фіз.-мат. наук;

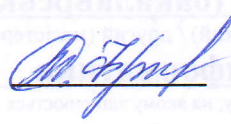
Косович Ігор Тарасович, асистент кафедри математичного моделювання

Погоджено з гарантом ОП  **Ігор ЮРЧЕНКО**

Затверджено на засіданні кафедри математичного моделювання
Протокол № 1 від «28» серпня 2026 року

Завідувач кафедри  **Ігор ЧЕРЕВКО**

Схвалено методичною радою факультету математики та інформатики
Протокол № 1 від «28» серпня 2026 року

Голова методичної ради  **Тоня ФРАТАВЧАН**

Мета навчальної дисципліни:

формування у студентів системних знань та практичних навичок розробки програм мовою С, включаючи засвоєння базового синтаксису, керування потоком виконання, роботи з пам'яттю, модульного програмування, структур даних і файлових систем, а також оволодіння сучасними підходами до компіляції, налагодження, аналізу та оптимізації програмного коду.

Пререквізити: Навчальна дисципліна «Програмування» відповідно до структурно-логічної схеми освітньо-професійної програми слухається здобувачами освіти в першому та другому семестрі першого року навчання паралельно із такими дисциплінами як «Алгебра і геометрія», «Дискретна математика», «Архітектура обчислювальних систем». Вивчення дисципліни «Програмування» є основою для засвоєння інших обов'язкових дисциплін, зокрема «Об'єктно-орієнтоване програмування», «Теорія алгоритмів». Для успішного оволодіння курсу здобувач має вільно володіти знаннями з курсу шкільної інформатики.

Результати навчання: Відповідно до освітньо-професійної програми підготовки бакалаврів галузі знань F Інформаційні технології за спеціальністю F3 Комп'ютерні науки (освітньо-професійна програма «Комп'ютерні науки та проєктування програмних систем») вивчення дисципліни сприяє формуванню:

загальних та фахових компетентностей:

ЗК1. Здатність до абстрактного мислення, аналізу та синтезу.

ЗК2. Здатність застосовувати знання у практичних ситуаціях.

ЗК3. Знання та розуміння предметної області та розуміння професійної діяльності.

ЗК7. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.

ЗК8. Здатність генерувати нові ідеї (креативність).

ЗК9. Здатність працювати в команді.

ЗК10. Здатність бути критичним і самокритичним.

ЗК11. Здатність приймати обґрунтовані рішення.

ЗК12. Здатність оцінювати та забезпечувати якість виконуваних робіт.

ФК3. Здатність до логічного мислення, побудови логічних висновків, використання формальних мов і моделей алгоритмічних обчислень, проєктування, розроблення й аналізу алгоритмів, оцінювання їх ефективності та складності, розв'язності та нерозв'язності алгоритмічних проблем для адекватного моделювання предметних областей і створення програмних та інформаційних систем.

програмних результатів навчання:

ПРН1. Застосовувати знання основних форм і законів абстрактнологічного мислення, основ методології наукового пізнання, форм і методів вилучення, аналізу, обробки та синтезу інформації в предметній області комп'ютерних наук.

Опис навчальної дисципліни

Загальна інформація

Форма навчання	Рік підготовки	Семестр	Кількість		Кількість годин						Вид підсумкового контролю
			кредитів	всього годин	лекції	практичні	семінарські	лабораторні	самостійна робота	індивідуальні завдання	
Денна	1	1,2	12	360	74	46	–	60	180	–	залік, екзамен

Структура змісту навчальної дисципліни

Назви змістових модулів і тем навчальних занять	Кількість годин											
	<i>денна форма</i>						<i>заочна форма</i>					
	усього	у тому числі					усьо го	у тому числі				
		л	п	лаб	інд	с.р.		л	п	лаб	інд	с.р.
1	2	3	4	5	6	7	8	9	10	11	12	13
Змістовий модуль 1. Базовий синтаксис та керування потоком												
Тема 1. Вступ до C. Процес компіляції (GCC/Clang). Типи даних, змінні, стандартне введення/виведення (printf, scanf). Базові алгоритмічні структури.	18	4	2	2		10						
Тема 2. Оператори та вирази. Арифметичні, логічні та побітові операції (Bitwise logic).	22	4	4	2		12						
Тема 3. Керуючі конструкції. Умовні переходи (if-else, switch). Цикли (for, while, do-while).	34	8	6	8		12						
Тема 4. Основи роботи з пам'яттю. Показники (Pointers) та адресна арифметика.	18	4	2	2		10						
Разом за ЗМ 1	92	20	14	14		44						
Змістовий модуль 2. Модульність та робота з пам'яттю												
Тема 1. Функції користувача. Передача аргументів за значенням та за посиланням. Рекурсія.	26	6	4	4		12						
Тема 2. Масиви (одновимірні та багатовимірні). Зв'язок масивів з показниками.	30	8	4	4		14						

Тема 3. Рядки та символні масиви. Стандартна бібліотека string.h. Безпечна робота з рядками.	28	6	4	4		14						
Тема 4. Динамічне керування пам'яттю. Функції malloc, calloc, realloc, free.	22	4	4	4		10						
Разом за ЗМ 2	106	24	16	16		50						
Змістовий модуль 3. Структурування даних та файлова система												
Тема 1. Препроцесор C. Макроси, умовна компіляція. Створення власних заголовних файлів (.h).	22	4	4	4		10						
Тема 2. Користувачькі типи даних. Структури (struct), Об'єднання (union), перерахування (enum), typedef.	26	4	4	4		14						
Тема 3. Динамічні структури даних. Реалізація Стейку, Черги та Зв'язного списку.	26	4	4	4		14						
Тема 4. Робота з файлами та потоками. Текстові та бінарні файли. Позиціонування у файлі.	26	4	4	6		12						
Разом за ЗМ3	100	16	16	18		50						
Змістовий модуль 4. Сучасна розробка та продуктивність												
Тема 1. Сучасні можливості мови C: оновлення стандартів C11–C23	12	2		2		8						
Тема 2. Компілятори та системи збірки. Поглиблений розгляд етапів трансляції. Робота з Makefiles та CMake.	14	4		2		8						
Тема 3. Швидкодія програм та основи оптимізації: роль організації пам'яті (кеш-локальність, вирівнювання, restrict)	18	4		4		10						
Тема 4. Відладка та аналіз програм: інструменти пошуку помилок і підвищення якості коду	18	4		4		10						
Разом за ЗМ4	62	14		12		36						
Усього годин	360	74	46	60		180						

Тематика лекційних занять з переліком питань

№	Назва теми з основними питаннями
Змістовий модуль 1. Базовий синтаксис та керування потоком	
1	Тема 1. Вступ до С. Процес компіляції (GCC/Clang). Типи даних, змінні, стандартне введення/виведення (printf, scanf). • Мова програмування С. Структура програми мовою С. • Правила оформлення коду програми мовою С. • Змінні. Іменовані константи. Стандартні типи даних мови С. • Введення та виведення даних. • Процес компіляції (GCC/Clang).
2	Тема 2. Оператори та вирази. Арифметичні, логічні та побітові операції (Bitwise logic). • Арифметичні вирази, арифметичні операції і стандартні математичні функції. Пріоритет математичних операцій. • Інкремент. Декремент. Оператор присвоєння. Порожній оператор. • Логічні оператори і вирази. • Побітові операції (Bitwise logic).
3	Тема 3. Керуючі конструкції. Умовні переходи (if-else, switch). Цикли (for, while, do-while). • Оператор умовного переходу if-else. • Оператор безумовного переходу goto. • Оператор множинного вибору switch • Вкладені розгалуження. Сходинковий оператор if-else-if. Оператор «? :» • Цикл з лічильником for. Деякі особливості циклу for. • Оператор послідовного виконання «,». • Цикли while та do-while • Алгоритми поєднання розгалуження та повторення • Вкладені цикли. Покрокове введення та виведення даних. • Рекурентні послідовності
4	Тема 4. Основи роботи з пам'яттю. Показчики (Pointers) та адресна арифметика. • Оголошення та ініціалізація вказівників (створення вказівників на змінні різних типів, виведення адреси та значення) • Розіменування вказівників (доступ і зміна значення змінної через вказівник) • Арифметика вказівників (операції ++, --, додавання/віднімання цілих чисел до вказівника) • Практичне застосування (прості приклади: обмін значеннями двох змінних через вказівники, робота з подвійним вказівником)
Змістовий модуль 2. Модульність та робота з пам'яттю	
5	Тема 1. Функції користувача. Передача аргументів за значенням та за посиланням. Рекурсія. • Основні поняття • Передача аргументів за значенням та за посиланням (адресою) у функцію. • Написання функцій із різними типами повернутих значень (int, double, char). • Використання вказівників для повернення кількох результатів з функції (наприклад, одночасне знаходження суми та добутку чисел). • Рекурсивні алгоритми для роботи з числами (НСД, сума цифр числа). • Прості рекурсивні функції (обчислення факторіалу, степеня числа). • Приклади: рекурсивне обчислення чисел Фібоначчі, рекурсивний вивід елементів послідовності.

6	Теми 2-3. Масиви (одновимірні та багатовимірні). Зв'язок масивів з покажчиками. Динамічне керування пам'яттю. Функції malloc, calloc, realloc, free. • Ініціалізація одновимірних масивів. • Типові помилки при роботі з масивами (вихід за межі, неправильна ініціалізація). • Використання циклів для обробки масивів (введення, виведення, обчислення). • Найпростіші алгоритми роботи з одновимірними масивами (пошук заданого елемента, пошук мінімального/максимального елемента). • Оголошення двовимірних масивів. • Використання вкладених циклів для обробки. • Приклади застосування двовимірних масивів (таблиці, матриці). • Доступ до елементів масиву через вказівник. • Перегляд і обробка масиву за допомогою вказівників. • Методи ітеративного обходу елементів двовимірного масиву. • Алгоритми визначення мінімального та максимального значень у двовимірному масиві. • Обчислення сумарних та середніх характеристик елементів масиву за рядками, стовпцями та в цілому. • Приклади застосування двовимірних масивів у практичних задачах програмування. • Функції для ініціалізації масиву (заповнення випадковими числами або введення з клавіатури). • Функції для обробки масиву (підрахунок кількості від'ємних/додатних елементів, пошук у масиві першого елемента, що задовольняє певну умову, наприклад, парний або більший за задане число).
7	Тема 4. Рядки та символьні масиви. Стандартна бібліотека string.h. Безпечна робота з рядками. • Поняття рядка як символьного масиву в мові C. • Методи оголошення та ініціалізації символьних масивів. • Доступ до окремих символів рядка та операції над ними. • Стандартні бібліотечні функції для обробки рядків (strlen, strcpy, strcat, strcmp та ін.). • Практичні приклади використання рядків у програмуванні. • Пошук символів або підрядків у рядку. • Модифікація символів та перетворення рядків. • Безпечна робота з рядками. • Передача рядків у функцію як символьних масивів: реалізація власних аналогів strlen, strcpy.
Змістовий модуль 3. Структурування даних та файлова система	
8	Тема 1. Препроцесор C. Макроси, умовна компіляція. Створення власних заголовних файлів (.h). • Поняття препроцесора мови C та його роль у процесі компіляції. • Основні директиви препроцесора. • Директива #include: призначення та способи підключення файлів. • Макроси в мові C: визначення та класифікація. • Константні макроси та макроси з параметрами. • Особливості використання макросів та типові помилки. • Умовна компіляція: поняття та призначення. • Директиви умовної компіляції (#if, #ifdef, #ifndef, #else, #endif). • Захист від повторного включення заголовних файлів (include guard). • Заголовні файли (.h): призначення та структура. • Організація багатофайлових програм із використанням заголовних файлів.

	• Типові помилки при використанні препроцесора та заголовних файлів.
9	Тема 2. Користувацькі типи даних. Структури (struct), Об'єднання (union), перерахування (enum), typedef. • Оголошення та ініціалізація структур, доступ до полів. • Масиви структур: збереження та вивід даних (наприклад, відомості про студентів). • Структури в структурах: створення складніших моделей (адреса у структурі «Студент»). • Приклад: підрахунок кількості структур, які задовольняють певну умову (наприклад, студенти з балом вище 90). • Передача структур у функції за значенням і за адресою. • Функції для обробки структур (заповнення, вивід, пошук за критерієм). • Використання вказівників на структури, доступ до полів через оператор ->. • Приклад: функція, яка отримує масив структур і обчислює середнє значення певного поля. • Оголошення та ініціалізація union, доступ до полів. • Практичні приклади: збереження даних різних типів у union (наприклад, числа та символи). • Використання enum для створення злічених (порядкових) типів даних. • Приклад: створення перерахування для днів тижня та використання його у програмі. • Створення псевдонімів типів за допомогою typedef для структур, масивів, вказівників. • Практичні приклади: спрощення оголошень складних типів (наприклад, масив структур). Використання typedef для підвищення читабельності алгоритмів обробки даних.
10	Тема 3. Динамічні структури даних. Реалізація Стейку, Черги та Зв'язного списку. • Створення динамічного стека та черги через масиви та вказівники. • Операції над стеком: додавання (push), видалення (pop), перегляд верхнього елемента. • Операції над чергою: додавання (enqueue), видалення (dequeue), перегляд першого елемента. • Оголошення структури вузла та створення простого однозв'язного списку. • Додавання і видалення елементів списку на початку, в кінці та за ключем. • Перегляд і обробка елементів списку (пошук, підрахунок елементів). • Приклади: обчислення суми елементів списку або пошук максимального значення.
11	Тема 4. Робота з файлами та потоками. Текстові та бінарні файли. Позиціонування у файлі. • Поняття потоку (stream) у мові C. • Види потоків: стандартні потоки введення, виведення та помилок. • Поняття файлу та його роль у зберіганні даних. • Відкриття та закриття файлів (fopen, fclose). • Режими відкриття файлів. • Читання даних з файлу (fscanf, fgets, fread). • Запис даних у файл (fprintf, fputs, fwrite). • Текстові файли: особливості та способи обробки. • Бінарні файли: особливості та відмінності від текстових. • Робота з файлами через покажчики (FILE *). • Позиціонування у файлі (fseek, ftell, rewind). • Обробка помилок при роботі з файлами. • Кінець файлу та перевірка стану потоку (feof, ferror). • Типові помилки при роботі з файлами та потоками.

Змістовий модуль 4. Основи сучасної розробки та налагодження програм	
	Тема 1. Сучасні можливості мови C: оновлення стандартів C11–C23 • Поняття стандартів мови програмування C та їх роль у розвитку мови • Огляд сучасних стандартів C11, C17 та C23 • Основні оновлення та покращення мови C у нових стандартах • Узагальнений вибір типів (_Generic) та його призначення • Перевірки на етапі компіляції (static assertions) • Анонімні структури та об'єднання: призначення та використання • Покращена підтримка Unicode у сучасному C • Практичне значення нових можливостей для розробки програм
	Тема 2. Процес компіляції та системи автоматичної збірки програм • Поняття компіляції програмного забезпечення та її роль у процесі розробки • Основні етапи трансляції вихідного коду у виконуваний файл • Функції препроцесора, компілятора та компоувальника (лінкера) у процесі збірки • Проблематика ручної організації процесу збірки програмних проєктів • Поняття систем автоматизації збірки програмного забезпечення • Система автоматизації збірки Make: принципи роботи та структура Makefile • Система CMake: призначення, основні концепції та особливості використання • Значення систем автоматизації збірки у сучасній програмній інженерії
	Тема 3. Швидкодія програм та основи оптимізації: роль організації пам'яті (кеш-локальність, вирівнювання, restrict) • Поняття швидкодії програм та її значення • Основні фактори, що впливають на продуктивність • Загальна ідея організації пам'яті комп'ютера • Кеш-локальність як принцип швидкого доступу до даних • Вирівнювання пам'яті та його вплив на швидкодію • _Alignas як засіб керування розміщенням даних • restrict як підказка компілятору для оптимізації • Прості приклади впливу коду на продуктивність
	Тема 4. Відладка та аналіз програм: інструменти пошуку помилок і підвищення якості коду • Поняття налагодження (debugging) та його роль у розробці програм. • Класифікація типів помилок у програмах: синтаксичні, логічні та помилки виконання. • Базові методи пошуку помилок у програмному коді. • Трасування виконання програми як метод аналізу її роботи. • Використання виводу (printf) для діагностики програм. • Основи роботи з відладчиком (GDB) (запуск програм у режимі відладки, встановлення точок зупину (breakpoints), перегляд значень змінних). • Типові помилки при роботі з пам'яттю. • Огляд інструментів налагодження та автоматизованого аналізу програм (GDB, Valgrind, AddressSanitizer).

Тематика практичних занять з переліком питань

№	Назва теми (питання/завдання)
1	Вступ до C. Процес компіляції (GCC/Clang). Типи даних, змінні, стандартне введення/виведення Завдання: - Розробити просту програму мовою C з використанням функції виведення даних. - Реалізувати введення та обробку числових даних із клавіатури. - Дослідити процес компіляції програми із застосуванням GCC або Clang.
2	Оператори та вирази. Арифметичні, логічні та побітові операції Завдання: - Реалізувати обчислення арифметичних виразів із врахуванням пріоритету операцій. - Розробити програму з використанням логічних операторів для перевірки умов. - Продемонструвати застосування побітових операцій для обробки цілих чисел.
3	Керуючі конструкції. Умовні переходи. Цикли Завдання: - Реалізувати алгоритми з використанням умовних операторів if-else та switch. - Розробити програму з використанням циклів для обробки послідовностей даних. - Проаналізувати відмінності між різними типами циклів (for, while, do-while).
4	Основи роботи з пам'яттю. Показчики та адресна арифметика Завдання: - Реалізувати роботу з вказівниками: оголошення, ініціалізація та розіменування. - Розробити програму для обміну значень змінних із використанням вказівників. - Дослідити особливості арифметики вказівників.
5	Функції користувача. Передача аргументів. Рекурсія Завдання: - Розробити набір функцій із різними типами повернених значень. - Реалізувати передачу аргументів за значенням і за адресою. - Реалізувати рекурсивні алгоритми для обчислення числових характеристик.
6	Масиви (одновимірні та багатовимірні). Зв'язок із показниками Завдання: - Реалізувати обробку одновимірних масивів (пошук, обчислення характеристик). - Алгоритми сортування. - Розробити алгоритми роботи з двовимірними масивами. - Вивести елементи масиву, використовуючи лише вказівники. - Обчислити суму елементів масиву через вказівник. - Продемонструвати різницю між передачею змінної "за значенням" і "за адресою". - Динамічне виділення пам'яті ('malloc', 'calloc').

	• Звільнення пам'яті ('free'). • Сортування обміном (bubble sort). • Сортування вибором (selection sort). • Сортування вставками (insertion sort). • Принцип роботи алгоритму сортування підрахунком (counting sort). • Реалізація цифрового сортування (radix sort). • Порівняння ефективності методів на масиві однакових даних.
7	Рядки та символьні масиви. Бібліотека string.h Завдання: • Реалізувати обробку рядків із використанням стандартних функцій бібліотеки. • Розробити алгоритми пошуку та модифікації рядків. • Реалізувати власні функції для роботи з символьними масивами.
8	Узагальнення: алгоритми обробки даних Завдання: • Розробити комплексну програму, що поєднує масиви, функції та умовні конструкції. • Проаналізувати ефективність реалізованих алгоритмів.
9	Препроцесор C. Макроси. Заголовні файли Завдання: • Реалізувати використання макросів для опису констант і функціональних залежностей. • Розробити власний заголовний файл для організації програмного коду. • Дослідити застосування умовної компіляції.
10	Користувачькі типи даних: struct, union, enum, typedef Завдання: • Розробити структури для опису предметної області. • Реалізувати обробку масивів структур. • Оголошення та ініціалізація об'єднань (union), доступ до полів. • Створення та використання злічених типів даних (enum). • Порівняння роботи struct і union у спільних прикладах. • Приклади: опис елементів меню або днів тижня через enum і збереження даних різних типів у union. • Продемонструвати використання typedef для підвищення читабельності коду.
11	Динамічні структури даних Завдання: • Реалізація стека з використанням масиву або списку. • Операції над стеком: push, pop, перегляд верхнього елемента. • Реалізація черги та основні операції enqueue, dequeue. • Оголошення структури вузла та створення однозв'язного списку. • Додавання і видалення елементів списку. • Перегляд і пошук елементів у списку. • Проаналізувати особливості динамічного керування пам'яттю.
12	Робота з файлами та потоками Завдання: • Реалізувати читання та запис даних у текстові файли. • Розробити програму для обробки даних із файлу. • Дослідити особливості роботи з бінарними файлами.

13	Сучасні можливості мови C (C11–C23) Завдання: • Ознайомитися з новими можливостями стандартів C11–C23. • Продемонструвати використання <code>_Generic</code> та статичних тверджень. • Проаналізувати доцільність використання нових можливостей у практиці програмування.
14	Процес компіляції та системи автоматичної збірки Завдання: • Дослідити етапи трансляції програмного коду. • Реалізувати збірку багатофайлового проєкту. • Розробити базовий Makefile або CMake-конфігурацію.
15	Швидкодія програм та основи оптимізації Завдання: • Проаналізувати вплив структури коду на продуктивність програми. • Продемонструвати принципи ефективної роботи з пам'яттю. • Реалізувати приклади оптимізації обчислень.
16	Відладка та аналіз програм Завдання: • Виявити та виправити помилки у програмному коді. • Реалізувати відладку програми з використанням базових інструментів. • Ознайомитися з інструментами автоматизованого аналізу програм.

Тематика лабораторних занять з переліком питань

№	Назва теми (завдання)
1	Лінійні та розгалужені програми (ознайомитися з лінійними та розгалуженими алгоритмами, навчитися застосовувати оператори розгалуження для реалізації різних сценаріїв виконання програм і перевірки умов)
2	Вкладені розгалуження та множинний вибір (ознайомитися з вкладеними розгалуженнями та конструкціями множинного вибору, навчитися реалізовувати складніші умови та альтернативні варіанти виконання програми)
3	Циклічні програми (ознайомитися з циклічними структурами програм, навчитися реалізовувати повторювані обчислення та обробку даних за допомогою циклів)
4	Функції користувача та рекурсія (розробити програмні модулі у вигляді функцій користувача, реалізувати передачу параметрів за значенням і за адресою; дослідити особливості використання вказівників для передачі даних у функції та повернення результатів; реалізувати рекурсивні алгоритми для обробки числових даних).
5	Масиви та алгоритми їх обробки (реалізувати алгоритми обробки одновимірних і двовимірних масивів, включаючи пошук, обчислення характеристик та аналіз даних, з використанням власних функцій; продемонструвати використання вказівників для доступу до елементів масивів та дослідити особливості їх обробки через адресну арифметику; реалізувати базові підходи до динамічного виділення пам'яті для масивів).
6	Рядки, покажчики та робота з пам'яттю (розробити програми для обробки символьних масивів із використанням стандартних бібліотечних засобів та власних функцій; дослідити застосування вказівників для роботи з рядками та передавання даних у функції; реалізувати динамічне виділення та звільнення пам'яті для символьних масивів).

7	Препроцесор та модульна організація програм (реалізувати багатофайлову програму з використанням директив препроцесора, макросів і заголовних файлів для структуризації програмного коду).
8	Користувачькі типи даних (розробити програму з використанням структур, об'єднань та перерахувань, реалізувати обробку масивів структур та застосувати typedef для підвищення читабельності коду).
9	Динамічні структури даних (реалізувати структури даних (стек, черга, зв'язний список) із використанням динамічного виділення пам'яті та забезпечити виконання базових операцій над елементами).
10	Робота з файлами та потоками (розробити програму для зчитування, обробки та збереження даних у текстових і бінарних файлах із використанням стандартних засобів роботи з потоками).
11	Процес компіляції та автоматизація збірки програм (дослідити етапи трансляції програм, реалізувати збірку багатофайлового проєкту та створити конфігурацію збірки із використанням Make або CMake).
12	Оптимізація, відладка та аналіз програм (проаналізувати продуктивність програмного коду, реалізувати базові методи оптимізації, виконати налагодження програм та ознайомитися з інструментами автоматизованого аналізу).

Завдання для самостійної роботи студентів

№	Назва теми	Завдання для самостійної роботи	К-сть год.
1.	Вступ до C. Процес компіляції (GCC/Clang). Типи даних, змінні, стандартне введення/виведення (printf, scanf). Базові алгоритмічні структури.	Розробити блок-схеми алгоритмів знаходження розв'язку наступних задач: *Обчислити суму перших k непарних чисел; *Обчислити добуток усіх чисел від 1 до k, кратних 3; *Визначити кількість цифр у числі k^3; *Знайти найменший дільник числа k, що більший за 1; k – номер залікової книжки студента.	10
2.	Оператори та вирази. Арифметичні, логічні та побітові операції	*Реалізувати програму для обчислення площі прямокутника або трикутника за заданими сторонами. *Написати програму для конвертації температури ($^{\circ}\text{C} \leftrightarrow ^{\circ}\text{F}$). *Розробити програму для обчислення вартості товару з ПДВ або знижкою.	12
3.	Керуючі конструкції. Умовні переходи (if-else, switch). Цикли (for, while, do-while).	*Написати програму, яка визначає, чи введене число додатне, від'ємне чи нуль. *Реалізувати програму, що перевіряє, чи можна утворити трикутник зі сторін a, b, c. *Створити програму, яка обчислює податок залежно від доходу: до 5000 – 10%, 5001–10000 – 15%, більше 10000 – 20%. *Реалізувати програму для знаходження	12

		добутку непарних чисел від 1 до N. *Створити програму, яка обчислює суму цифр цілого числа.	
4.	Основи роботи з пам'яттю. Показчики (Pointers) та адресна арифметика.	* Реалізувати програму для обміну значеннями двох змінних за допомогою показчиків. * Написати функцію для знаходження мінімального та максимального елементів масиву з використанням показчиків * Реалізувати програму для динамічного виділення пам'яті під дійсне число та обчислення його квадрата.	10
5.	Функції користувача. Передача аргументів за значенням та за посиланням. Рекурсія.	* Написати функцію для визначення кількості цифр у заданому цілому числі. * Написати функцію для перевірки, чи є задане число простим. * Реалізувати рекурсивну функцію для обчислення добутку цифр натурального числа. * Написати функцію для визначення, чи є задане число паліндромом. * Реалізувати функцію для знаходження найбільшої цифри заданого натурального числа. *Реалізувати функцію, яка повертає новий динамічний масив — копію переданого масиву. *Написати функцію void printLine(char c, int n), яка виводить символ c задану кількість разів. *Створити функцію void circle(float r, float *length, float *area), яка за радіусом обчислює довжину кола та площу круга (результати повертає через параметри). *Написати функцію, яка перевіряє, чи є задане число простим. *Реалізувати програму, що містить функції для роботи з рядками: * перетворення літер на великі.	12
6.	Масиви (одновимірні та багатовимірні). Зв'язок масивів з показниками. Динамічне керування пам'яттю. Функції malloc, calloc, realloc, free.	*Створити масив дійсних чисел і підрахувати кількість елементів більших за середнє. *Реалізувати пошук першого від'ємного елемента в масиві. *Написати програму для обчислення суми елементів із парними індексами. *Розробити програму для циклічного зсуву елементів масиву праворуч. *Знайти суму елементів заданого стовпця матриці. *Реалізувати програму для транспонування квадратної матриці. *Написати програму, яка обчислює суму всіх елементів, розташованих вище головної діагоналі.	24

		*Обчислити добуток двох матриць. *Реалізувати програму для підрахунку кількості елементів, кратних заданому числу. *Написати функцію для підрахунку кількості унікальних елементів масиву. *Написати функцію для видалення елементів масиву, які повторюються.	
7.	Рядки та символічні масиви. Стандартна бібліотека string.h. Безпечна робота з рядками.	*Написати програму, яка підраховує кількість слів у рядку. *Створити функцію для перетворення всіх малих літер у великі. *Реалізувати перевірку, чи містить рядок лише цифри. *Реалізувати функцію для пошуку студента з найвищим середнім балом. *Створити функцію для пошуку студента за прізвищем у масиві структур. *Створити функцію int findAll(int *arr, int n, int key, int *positions), яка знаходить усі позиції заданого елемента.	14
8.	Препроцесор C. Макроси, умовна компіляція. Створення власних заголовних файлів (.h).	Створити власний заголовний файл із прототипами функцій для обчислення площі та периметра геометричних фігур і підключити його до програми.	10
9.	Користувацькі типи даних. Структури (struct), Об'єднання (union), перерахування (enum), typedef.	*Створити структуру «Точка» та обчислити відстань між двома точками. *Реалізувати структуру «Товар» і програму для підрахунку загальної вартості товарів. *Написати програму, що сортує масив структур «Студент» за середнім балом. *Розробити функцію, яка знаходить студента за номером залікової книжки. *Реалізувати програму, що використовує enum для задання режиму роботи (AUTO, MANUAL, TEST). *Використати typedef для оголошення скорочених імен структурних типів. *Створити union для збереження різних типів даних у полі структури «Дані». *Пошук у масивах структур. *Реалізувати функцію для пошуку студента за номером id (лінійним пошуком).	14
10.	Динамічні структури даних. Реалізація Стейку, Черги та	*Створити програму, яка використовує чергу для моделювання черги пацієнтів у реєстратурі.	14

	Зв'язного списку.	*Реалізувати функцію <code>int countElements(struct Node *head)</code> — підрахунок кількості елементів. *Написати функцію для пошуку елемента у списку за значенням. *Створити програму, що реалізує однозв'язний список для зберігання відомостей про товари (назва, ціна, кількість). Реалізувати: • додавання нового товару; • пошук за назвою; • обчислення загальної вартості всіх товарів. *Нехай номер варіанта студента — k. • Створити список із k елементів, обчислити суму всіх заданих елементів. • Додати до списку елемент зі значенням k^2 після третього вузла. • Зі стеку видалити всі елементи, менші за k.	
11.	Робота з файлами та потоками. Текстові та бінарні файли. Позиціонування у файлі.	* Написати програму для пошуку найдовшого слова в текстовому файлі. * Реалізувати програму для додавання нових записів про товари до бінарного файлу та пошуку товару за кодом. * Написати програму для визначення кількості чисел у текстовому файлі, що перевищують задане значення. * Реалізувати програму для читання заданого фрагмента бінарного файлу з використанням позиціонування. * Написати програму для об'єднання вмісту двох текстових файлів в один. * Реалізувати програму для пошуку заданого слова в текстовому файлі та визначення номерів рядків, у яких воно зустрічається. *Написати програму, яка зчитує з файлу список студентів (ПІБ і бал), сортує їх за балами та зберігає відсортований список у новий файл. *Реалізувати функцію, що сортує динамічний масив дійсних чисел за допомогою вказівників. *Нехай номер варіанта студента — k. • Створити масив із $k+10$ випадкових чисел і відсортувати його методом вибору. • Упорядкувати перші k елементів масиву методом вставок.	12
12.	Сучасні можливості мови C: оновлення стандартів C11–C23	• Реалізувати програму для безпечного виділення динамічної пам'яті з перевіркою результату та обробкою помилок. • Написати програму з використанням	8

		узагальнених макросів <code>_Generic</code> для визначення типу переданого аргументу. Реалізувати програму для вимірювання часу виконання фрагмента коду засобами стандартної бібліотеки C.	
13.	Компілятори та системи збірки. Поглиблений розгляд етапів трансляції. Робота з Makefiles та CMake.	- Налаштувати компіляцію програми мовою C з кількох вихідних файлів і створити виконуваний файл. - Створити <code>Makefile</code> для автоматизації компіляції програми з кількох модулів із можливістю очищення файлів збірки. - Створити проєкт на C із використанням CMake для налаштування компіляції та компонування програми.	8
14.	Швидкодія програм та основи оптимізації: роль організації пам'яті (кеш-локальність, вирівнювання, <code>restrict</code>)	- Реалізувати два варіанти обробки двовимірного масиву з різним порядком обходу елементів і порівняти час їх виконання. - Написати програму для порівняння швидкодії обробки масиву за допомогою індексів і покажчиків. - Дослідити вплив використання <code>restrict</code> на швидкодію функції обробки масивів. - Реалізувати програму для порівняння часу виконання операцій над даними з різним вирівнюванням у пам'яті.	10
15.	Відладка та аналіз програм: інструменти пошуку помилок і підвищення якості коду	- Виявити та виправити логічні помилки у програмі за допомогою покрокового виконання в налагоджувачі GDB. - Використати <code>AddressSanitizer</code> для виявлення помилок роботи з динамічною пам'яттю.	10

Самостійна робота також складається з повторення матеріалу, засвоєного на лекціях, самостійного опанування частини теоретичного матеріалу, роботи з контрольними запитаннями та завданнями лабораторних робіт.

Методи навчання

Лекції, лабораторні роботи, тестування, аудиторне та онлайн-навчання з використанням систем Moodle та Google Meet.

Методи навчання:

- вербальні методи (лекція, бесіда, диспут, пояснення, розповідь тощо);
- практичні методи (лабораторні роботи);
- наочні методи (демонстрація, ілюстрація);
- робота з інформаційними ресурсами: з навчально-методичною, науковою, нормативною літературою та інтернет-ресурсами;
- самостійна робота над індивідуальним завданням або за програмою навчальної дисципліни;
- навчання з використанням відповідних онлайн-платформ.

Система контролю та оцінювання

Засобами оцінювання та демонстрування результатів навчання є:

- усне та письмове опитування;
- виконання практичних та лабораторних робіт;
- стандартизовані тести.

Поточний контроль знань відбувається на практичних та лабораторних заняттях шляхом усного та письмового опитування теоретичних основ теми, виконання практичних та лабораторних завдань, а також у вигляді контрольних робіт та тестування.

Форми підсумкового контролю:

1 семестр – залік, 2 семестр – екзамен.

Критерії оцінювання поточного та підсумкового контролю

Система оцінювання рівня навчальних досягнень ґрунтується на принципах ECTS та є накопичувальною. Впродовж першого та другого семестрів студенти виконують 12 лабораторних робіт (по шість у кожному). Лабораторні роботи оцінюються по 6 або 7 балів за кожну (див. таблицю нижче).

Виконуючи завдання лабораторної роботи (ЛР), студент повинен оформити і завантажити для подальшої перевірки на сайт електронного навчання звіт разом із працездатними файлами програмної реалізації завдань ЛР (правила оформлення, варіанти завдань розміщені на платформі <https://moodle.chnu.edu.ua/> на сторінці навчальної дисципліни).

50% балів, відведених на оцінювання ЛР, студент отримує за повністю правильно виконану ЛР та оформлений звіт. Решта 50% балів виставляється після захисту студентом виконаного звіту. У випадку неістотної помилки знімається 10-20% балів, а у випадку істотної 20-40% балів, якщо ж студент не опанував теоретичний матеріал, плутається в означеннях, то знімається до 50% балів від усієї суми балів за ЛР.

По завершенню кожного модуля студенти проходять тестове онлайн опитування на сайті електронного навчання ЧНУ. Протягом одного модуля студенти можуть отримати 20 балів за захист лабораторних завдань та 10 балів за успішне проходження тестів. Весь річний курс розбито на чотири модулі.

Залік проводиться у формі тестового опитування з використанням системи електронного навчання Moodle. Кожному студенту випадковим чином генерується 20 запитань. Сумарна максимальна кількість балів за виконання тестів становить 40 балів. Час виконання тесту – 40 хвилин. Загальна оцінка за залік визначається з урахуванням всіх отриманих балів у першому семестрі (модулі 1, 2 та підсумковий тест).

Екзамен проводиться у формі тестового опитування з використанням системи електронного навчання Moodle. Кожному студенту випадковим чином генерується 22 запитання чотирьох рівнів складності. 1 рівень – 4 запитання по 0,25 бала, 2 рівень – 8 запитань по 0,5 бала, 3 рівень – 5 запитань по 1 балу, 4

рівень – 5 запитань по 2 бали. Сумарна максимальна кількість балів за виконання тестів становить 20 балів. Час виконання тесту – 30 хвилин.

Практичною частиною екзамену є розв’язання студентами двох задач, що містяться у базі даних даного курсу і також генерується випадковим чином. Виконання цього завдання оцінюється по 10 балів за кожне. Час виконання становить 30 хвилин.

Сумарна максимальна оцінка за екзамен становить 40 балів. Загальна оцінка за навчальну дисципліну визначається з урахуванням всіх отриманих балів у другому семестрі (модулі 3, 4 та підсумковий модуль).

Розподіл балів, які отримують студенти у першому семестрі

Поточне тестування та самостійна робота								Підсумковий тест (залік)	Сума
Змістовий модуль 1				Змістовий модуль 2				40	100
T1	T2	T3-T4	тест	T1	T2	T3-T4	тест		
6	7	7	10	7	7	6	10		

Розподіл балів, які отримують студенти у другому семестрі

Поточне тестування та самостійна робота								Підсумковий модуль (екзамен)	Сума
Змістовий модуль 3				Змістовий модуль 4				40	100
T1	T2	T3-T4	тест	T1	T2	T3-T4	тест		
7	7	6	10	6	7	7	10		

Шкала оцінювання: національна та ECTS

100-бальна шкала	Оцінка за національною шкалою		Оцінка за шкалою ECTS	
			Оцінка	Пояснення за розширеною шкалою
90-100	Зараховано	Відмінно	A	відмінно
80-89		Добре	B	дуже добре
70-79			C	добре
60-69		Задовільно	D	задовільно
50-59			E	достатньо
35-49	Незараховано	Незадовільно	FX	(незадовільно) з можливістю повторного складання
1-34			F	(незадовільно) з обов’язковим самостійним повторним опрацюванням освітнього компонента до перескладання

Перелік питань для самоконтролю та підсумкового контролю навчальних досягнень студентів

1. Основні поняття алгоритмізації та програмування. Базові алгоритмічні структури
2. Мови програмування. Класифікація мов програмування. Мова програмування C. Структура програми мовою C.
3. Елементи мови програмування C (алфавіт, ідентифікатори, команди, директиви препроцесора). Типи даних
4. Лінійні програми. Структура лінійних програм. Приклади
5. Арифметичні вирази. Інструкція присвоєння. Приклади
6. Введення та виведення даних . Приклади
7. Оператор умовного переходу. Приклади
8. Оператор вибору (перемикач). Приклади
9. Оператор безумовного переходу. Поняття мітки. Оператор “?”. Приклади
10. Оператор циклу з повторенням for. Приклади
11. Оператори циклу з передумовою while та з післяумовою do/while. Приклади
12. Використання у циклах операторів break, exit(), continue. Приклади
13. Вкладені цикли. Приклади
14. Покрокове введення та виведення даних. Приклади
15. Рекурентні послідовності. Приклади
16. Одновимірні масиви. Приклади
17. Пошук заданого елемента в послідовності. Алгоритм бінарного пошуку в упорядкованій послідовності. Приклади
18. Пошук мінімального/максимального елемента в послідовності. Приклади
19. Константне задання елементів масиву. Ініціалізація масивів. Генератор випадкових чисел. Приклади
20. Використання файлів для введення/виведення даних. Приклади
21. Двовимірні масиви. Багатовимірні масиви. Приклади
22. Створення тестових файлів для роботи з масивами. Приклади
23. Пошук мінімального та максимального елементів у двовимірному масиві. Приклад
24. Обробка елементів двовимірних масивів по рядках та по стовпцях. Приклади
25. Рядки та символьні масиви. Введення та виведення символьних та рядкових даних. Приклади
26. Спеціальні функції для введення та виведення значень символьних і рядкових величин. Приклади
27. Стандартні функції для роботи з рядками. Приклади
28. Лінійні пошукові алгоритми для рядкових величин. Приклади тестів
29. Алгоритми зміни довжин рядкових величин. Приклади
30. Порівняння рядкових величин. Посимвольне введення рядкових величин. Перетворення символьних значень у числові. Приклади
31. Масиви рядків. Приклади
32. Поняття динамічних змінних. Опис динамічних змінних або змінних-вказівників. Оператори для роботи з вказівниками. Приклади
33. Арифметичні дії з вказівниками. Порівняння вказівників. Приклади
34. Динамічне виділення пам'яті та вказівники. Приклади
35. Обробка елементів масиву за допомогою адресної арифметики. Приклади
36. Індексція елементів одновимірного масиву за допомогою вказівників. Приклади
37. Обробка елементів двовимірного масиву за допомогою вказівників. Приклади
38. Вказівники на символьні масиви. Ініціалізація вказівників. Масиви вказівників. Приклади
39. Поняття функції. Загальний вигляд опису функції. Локальні та глобальні параметри або змінні. Фактичні та формальні аргументи. Прототип функції. Приклади
40. Способи передачі і повернення параметрів у функцію: параметри-значення. Приклади

- 41.Способи передачі і повернення параметрів у функцію: параметри-вказівники. Приклади
- 42.Масиви як параметри функцій. Приклади
- 43.Додаткові можливості передавання параметрів у функції. Функція main(). Приклади
- 44.Поняття рекурсії. Принцип виконання рекурсивних функцій. Приклади
- 45.Вказівники на функції. Приклади
- 46.Сучасні технології програмування. Переваги та недоліки використання функцій
- 47.Поняття структури. Оголошення змінних описаної структури. Особливості оголошення структур. Звернення до полів структури. Приклади
- 48.Масиви структур. Структури в структурах. Структури з масивів. Приклади
- 49.Використання структур для роботи з функціями: передавання полів структур в функції. Приклади
- 50.Використання структур для роботи з функціями: передавання структури у функцію. Приклади
- 51.Використання вказівників на структури. Приклади
- 52.Бітові поля. Операції над бітовими полями. Приклади
- 53.Об'єднання (union). Приклади
- 54.Злічені типи даних. Операції над елементами злічених типів. Приклади
- 55.Використання операторів typedef та sizeof. Алгоритми обробки структурованих даних.
- 56.Поняття про системи числення. Зв'язок між системами числення з основою 2k.
- 58.Бітові оператори. Приклади.
- 59.Робота з файлами. Тип змінних FILE.
60. Використання можливостей середовища програмування для роботи з файлами.
- 61.Рекурсивний пошук у впорядкованому одновимірному масиві. Оцінка алгоритму. Приклади тестів
- 62.Основні поняття алгоритмів сортування. Класифікація методів сортування. Прямі методи сортування. Оцінка ефективності
- 65.Сортування вибором. Оцінка алгоритму. Приклади тестів
- 66.Сортування обміном. Оцінка алгоритму. Приклади тестів
- 67.Сортування вставленням. Оцінка алгоритму. Приклади тестів
- 68.Лінійні методи сортування: сортування підрахунком. Оцінка алгоритму. Приклади тестів
- 69.Лінійні методи сортування: цифрове сортування. Оцінка алгоритму. Приклади тестів
- 70.Директиви препроцесора. Створення власних заголовочних файлів. Приклади.
71. Поясніть поняття стандарту мови програмування C та охарактеризуйте основні стандарти C11, C17 і C23.
72. Які основні нові можливості були додані у сучасних стандартах мови C? Наведіть приклади.
73. Поясніть призначення механізму узагальненого вибору типів (_Generic) та статичних тверджень (_Static_assert).
74. Охарактеризуйте поняття Unicode та його підтримку в сучасних стандартах мови C.
75. Поясніть процес компіляції програм мовою C та охарактеризуйте його основні етапи.
76. Яку роль відіграють препроцесор, компілятор і компонувальник у процесі трансляції програм?
77. Поясніть призначення систем автоматизації збірки програм. Порівняйте підходи Make та CMake.
78. Що таке швидкодія програми? Які основні фактори впливають на продуктивність програмного забезпечення?
79. Поясніть поняття кеш-локальності, вирівнювання пам'яті та використання restrict. Як вони впливають на ефективність програм?
80. Поясніть поняття налагодження програм. Охарактеризуйте типи помилок та основні інструменти їх пошуку (наприклад, GDB, Valgrind).

Зарахування результатів неформальної/інформальної освіти

Здобувачі вищої освіти має право на участь у неформальній/інформальній освіті.

У межах поточного контролю можуть визнаватися результати неформальної/інформальної освіти за умови наявності сертифікату або освітньої декларації про результати неформальної/інформальної освіти з питань, що відповідає тематиці курсу («Порядок визнання у Чернівецькому національному університеті імені Юрія Федьковича результатів навчання, здобутих шляхом неформальної та/або інформальної освіти», <https://www.chnu.edu.ua/media/4g5fzssb/poriadok-vyznannia-rezultativ-navchannia-zdobutykh-shliakhom-neformalnoi-ta-abo-informalnoi-osvity.pdf>).

Студентам можуть бути зараховані додаткові бали (максимум 5 балів), отримані через неформальну освіту, до загальної суми балів, набраної з освітньої компоненти, за умови, що результати з проблеми, за якою відбувалося навчання, відповідають тематиці курсу.

Рекомендована література **Основна**

1. Караванова Т.П. Методика розв'язування алгоритмічних задач. Основи алгоритмізації та програмування: Навчально-методичний посібник для вчителів / Т.П.Караванова. – Кам'янець-Подільський: Аксіома, 2013, 460 с.
2. Караванова Т.П. Методика розв'язування алгоритмічних задач. Побудова алгоритмів: Навчально-методичний посібник для вчителів / Т.П.Караванова. – Кам'янець-Подільський: Аксіома, 2014, 344 с.
3. Вінник В.Ю. Алгоритмічні мови та основи програмування: мова С. Житомир: ЖДТУ. 2007, 328 с. URL: <http://programming.in.ua/programming/c-language/327-vinnyk-algorithmic-languages-and-the-basics-of-programming-c-language.html>
4. Керніган Б.В., Річі Д.М. Мова програмування С, друге видання. Englewood Cliffs: Prentice Hall. 1988, 272 с. URL: <http://programming.in.ua/programming/c-language/227-book-programming-c-kernighan.html>
5. Мова програмування С/С++. Практикум: навчальний посібник / О. В. Прокопенко, М. О. Попов, Г. Л. Чумак. – К.: Київський національний університет імені Тараса Шевченка, 2024. – 375 с. URL: https://iht.knu.ua/wp-content/uploads/2024/06/C_C-%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D1%83%D0%BC_2024.pdf
6. Програмування. Частина 2. Навчальний посібник / Укл.: Івасюк Г.П., Фратавчан Т.М., Івасюк Р.В. Чернівці: Чернівецький нац. ун-т, 2025. 164 с.
7. Gustedt J. Modern C. Shelter Island: Manning Publications. 2019, 400 с.

URL: <https://inria.hal.science/hal-02383654v2/file/modernC.pdf>

8. Klemens B. 21st Century C: C Tips from the New School. Sebastopol: O'Reilly Media. 2012, 408 с. URL: <https://karadev.net/uroci/filespdf/files/21st-century-c-o-reilly-ben-klemens.pdf>
9. King K.N. C Programming: A Modern Approach. New York: W. W. Norton & Company. 2008, 832 с. URL: <https://books.google.com.ua/books?id=-5plzQEACAAJ>
10. cppreference.com C Reference Manual (онлайн довідник). URL: <https://en.cppreference.com/w/c.html>

Допоміжна

1. Herbert Schildt C++: A Beginner's Guide, Second Edition, Publisher: McGraw-Hill Prof Med/Tech, 2003, 567 p.
2. Herbert Schildt Herb Schildt's C++ Programming Cookbook, Publisher: McGraw Hill Professional, 2008, 509 p.
3. Robert Sedgewick Algorithms in C++, Parts 1-4: Fundamentals, Data Structure, Sorting, Searching, Third Edition 3rd Edition. — Publisher: Addison-Wesley Professional; 3rd edition (July 13, 1998). — 738 pages.
4. Stephen Prata C Primer Plus — Publisher: Addison-Wesley Professional, 2013. — 1037 pages.
5. Караванова Т.П. Інформатика: основи алгоритмізації та програмув.: 777 задач з рек. та прикл.: Навч. посіб. для 8-9 кл. із поглибленим вивч. інф-ки / За заг. ред. М.З.Згуровського — К.: Генеза, 2012. — 286 с.: іл. — Бібліограф.: 286 с.
6. Караванова Т.П. Інформатика: методи побудови алгоритмів та їх аналіз: необчислювальні алгоритми: Навч. посіб. для 9-10 кл. із поглибл. вивч. інф-ки — К.: Генеза. — 2007.— 216 с.:іл. — Бібліограф.: 212 с.

Інформаційні ресурси

1. <https://moodle.chnu.edu.ua/enrol/index.php?id=2535> — розміщення курсу на платформі <https://moodle.chnu.edu.ua/>.
2. <https://moodle.chnu.edu.ua/course/view.php?id=2536> — розміщення курсу на платформі <https://moodle.chnu.edu.ua/>.

Політика академічної доброчесності

Дотримання політики щодо академічної доброчесності учасниками освітнього процесу при вивченні освітнього компонента регламентовано чинними документами ЧНУ:

- "Кодекс академічної доброчесності та етики Чернівецького національного університету імені Юрія Федьковича"
<https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/kodeks-akademichnoi-dobrochesnosti-ta-etyky-chernivetskoho-natsionalnoho-universytetu-imeni-yuriiia-fedkovycha/>

- "Положення про виявлення та запобігання академічного плагіату у Чернівецькому національному університеті імені Юрія Федьковича"
<https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/polozhennia-pro-vyjavlennia-ta-zapobihannia-akademichnomu-plahiatu-u-chernivetskomu-natsionalnomu-universyteti-imeni-yuriiia-fedkovycha/>
- "Політика використання штучного інтелекту в Чернівецькому національному університеті імені Юрія Федьковича"
<https://www.chnu.edu.ua/universytet/normatyvni-dokumenty/polityka-vykorystannia-shtuchnoho-intelektu-v-chernivetskomu-natsionalnomu-universyteti-imeni-yuriiia-fedkovycha-zi-zminamy-zatverdzhenymy-vchenoiu-radoiu-chnu-imeni-yuriiia-fedkovycha/>

Використання здобувачами освіти інструментів штучного інтелекту (ШІ) під час виконання навчальних завдань:

- допускається за умови дотримання принципів академічної доброчесності (здобувач освіти несе відповідальність за зміст, достовірність і результати поданої роботи, а у разі використання ШІ зобов'язаний зазначити факт і спосіб його застосування);
- не допускається використання ШІ як заміни самостійного виконання завдань або з метою порушення вимог академічної доброчесності;
- додаткові обмеження чи вимоги щодо використання ШІ *можуть визначатися* викладачем в освітньому процесі для конкретних видів навчальної діяльності відповідно до особливостей дисципліни.