Чернівецький національний університет імені Юрія Федьковича
Факультет математики та інформатики
Кафедра прикладної математики та інформаційних технологій

“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 / вибіркова)

Освітньо-професійна програма _Технології програмування та комп’ютерне моделювання_
(назва програми)

Спеціальність _____F1 Прикладна математика_____
(вказати: код, назва)

Галузь знань _____F Інформаційні технології_____

Рівень вищої освіти _____перший (бакалаврський)_____
(вказати: перший бакалаврський/другий магістерський)

_____Факультет математики та інформатики_____
(назва факультету/інституту, на якому здійснюється підготовка фахівців за вказаною освітньо-професійною програмою)

Мова навчання _____українська_____
(вказати: на якій мові читається дисципліна)

Чернівці – 2026

Робоча програма навчальної дисципліни “Програмування” складена відповідно до освітньо-професійної програми “Технології програмування та комп’ютерне моделювання” підготовки здобувачів вищої освіти за спеціальністю F1 Прикладна математика (галузь знань F Інформаційні технології). Дисципліна реалізує другу частину обов’язкового освітнього компонента ОК6 “Програмування” (загальний обсяг компонента - 13,0 кредитів ЄКТС) в обсязі 6,5 кредитів ECTS (195 годин).

Розробник:

Данилюк Іван Михайлович, доцент кафедри прикладної математики та інформаційних технологій, кандидат фізико-математичних наук, доцент

Викладачі, що забезпечують читання даної навчальної дисципліни:

Данилюк Іван Михайлович, доцент кафедри прикладної математики та інформаційних технологій, кандидат фізико-математичних наук, доцент

Унгурян Галина Михайлівна, асистент кафедри прикладної математики та інформаційних технологій, кандидат фізико-математичних наук, доцент

Погоджено з гарантом ОП _____ **Василь МАЦЕНКО**
(підпис)

Затверджено на засіданні кафедри прикладної математики та інформаційних технологій

Протокол № 2 від « 28 » серпня 2026 року

Завідувач кафедри _____ **Ярослав БІГУН**
(підпис)

Схвалено методичною радою факультету математики та інформатики

Протокол № 1 від « 28 » серпня 2026 року

Голова методичної ради _____ **Тоня ФРАТАВЧАН**
(підпис)

Мета навчальної дисципліни полягає у розвитку логічного та аналітичного мислення студентів, уміння застосовувати індукцію, дедукцію, аналіз і синтез під час розв'язування завдань. Вона спрямована на формування ґрунтовної теоретичної бази знань щодо побудови алгоритмів та їх реалізації мовою програмування. Курс передбачає розвиток практичних навичок розв'язування алгоритмічних задач із використанням математичного апарату, спеціалізованої літератури та сучасної комп'ютерної техніки. Вивчення дисципліни має довести студентів до творчого рівня оволодіння матеріалом і сформувати бачення шляхів застосування здобутих знань у майбутній професійній діяльності. Окрім цього, курс інтегрується з іншими навчальними дисциплінами, забезпечуючи цілісність підготовки фахівців. Таким чином, мета дисципліни полягає у гармонійному поєднанні теоретичних знань і практичних умінь, необхідних для формування компетентного спеціаліста.

Пререквізити: базові знання з математики, інформатики.

Результати навчання: у результаті вивчення навчальної дисципліни студент буде

знати:

- правила запису розробленого алгоритму мовою програмування С;
- можливості використання середовища компіляції розробленої С-програми;
- загальні принципи розв'язування алгоритмічних задач; постановка задачі, побудова алгоритму, реалізація алгоритму мовою програмування, тестування реалізованого алгоритму;
- методи розв'язання алгоритмічних задач.

вміти:

- застосовувати теоретичні знання для розв'язування практичних завдань;
- використовувати навички роботи з інтегрованим середовищем програмування;
- аналізувати відомі методи побудови алгоритмів та визначати найоптимальніші з них для розв'язування конкретної задачі;
- реалізовувати побудовані алгоритми мовою програмування С;
- розробляти власні тести для перевірки коректності розроблених алгоритмів;
- тестувати розроблені алгоритми;
- використовувати навички техніки програмування.

Вивчення навчальної дисципліни передбачає формування та розвиток у студентів компетентностей, передбачених освітньо-професійною програмою:

- загальних:

ЗК01 - здатність учитися і оволодівати сучасними знаннями;

ЗК02 - здатність застосовувати знання у практичних ситуаціях;

ЗК03 - здатність генерувати нові ідеї (креативність);

ЗК04 - здатність бути критичним і самокритичним;

ЗК06 - здатність до абстрактного мислення, аналізу та синтезу;

ЗК07 - здатність до пошуку, оброблення та аналізу інформації з різних джерел;

ЗК08 - знання та розуміння предметної області та розуміння професійної діяльності;

ЗК10 - навички у використанні інформаційних і комунікаційних технологій;

ЗК12 - визначеність і наполегливість щодо поставлених завдань і взятих обов'язків;

ЗК13 - навички міжособистісної взаємодії;

- фахових:

ФК02 - здатність виконувати завдання, сформульовані у математичній формі;

ФК04 - здатність розробляти алгоритми та структури даних, програмні засоби та програмну документацію;

ФК06 - здатність розв'язувати професійні задачі за допомогою комп'ютерної техніки, комп'ютерних мереж та Інтернету, в середовищі сучасних операційних систем, з використанням стандартних офісних додатків;

ФК07 - здатність експлуатувати та обслуговувати програмне забезпечення автоматизованих та інформаційних систем різного призначення;

ФК08 - здатність використовувати сучасні технології програмування та тестування програмного забезпечення;

ФК10 - здатність створення документів встановленої звітності, використання нормативно-правових документів;

ФК17 - здатність до використання новітніх інформаційно-комунікаційних технологій;

ФК19 - здатність застосовувати сучасні інформаційні технології для комп'ютерної реалізації математичних моделей і алгоритми комп'ютерної графіки для розробки графічного програмного забезпечення.

Вивчення дисципліни деталізує такі програмні результати навчання:

ПРН11 - вміти застосовувати сучасні технології програмування та розроблення програмного забезпечення, програмної реалізації чисельних і символьних алгоритмів;

ПРН14 - виявляти здатність до самонавчання та продовження професійного розвитку;

ПРН15 - уміти організувати власну діяльність та одержувати результат у рамках обмеженого часу;

ПРН18 - ефективно спілкуватися з питань інформації, ідей, проблем та рішень зі спеціалістами та суспільством загалом.

Опис навчальної дисципліни

Загальна інформація

Форма навчання	Рік підготовки	Семестр	Кількість		Кількість годин						Вид підсумкового контролю
			кредитів	годин	лекції	практичні	семінарські	лабораторні	самостійна робота	індивідуальні	
Денна	1	2	6,5	195	30	30		30	105		Іспит
Заочна	-	-	-	-	-	-	-	-	-	-	-

Структура змісту навчальної дисципліни

Назви змістових модулів і тем	Кількість годин											
	денна форма						Заочна форма					
	усього	у тому числі					усього	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. Базові конструкції мови С												
Тема 1. Мова програмування С. Основи.	10	2	2	2		4						
Тема 2. Логічні оператори і вирази.	12	2	2	2		6						
Тема 3. Циклічні конструкції.	12	2	2	2		6						
Тема 4. Алгоритми поєднання розгалуження та повторення. Вкладені цикли. Покрокове введення та виведення даних. Рекурентні послідовності.	12	2	2	2		6						
Разом за змістовим модулем 1	46	8	8	8		22						
Змістовий модуль 2. Масиви, рядки та робота з даними												
Тема 5. Одновимірні масиви. Найпростіші алгоритми роботи з одновимірними масивами. Пошук заданого елемента, пошук мінімального/ максимального елемента.	12	2	2	2		6						
Тема 6. Додаткові способи введення даних. Основи роботи з файлами і потоками. Генератор випадкових чисел. Константні вхідні дані, ініціалізація масивів.	12	2	2	2		6						
Тема 7. Двовимірні масиви. Базові алгоритми для обробки елементів двовимірних масивів.	14	2	2	2		8						
Тема 8. Рядки і символьні масиви. Стандартні функції для роботи з рядками.	13	2	2	2		7						
Тема 9. Найпростіші алгоритми	13	2	2	2		7						

роботи із символьними та рядковими величинами.												
Разом за змістовим модулем 2	64	10	10	10		34						
Змістовий модуль 3. Вказівники, функції та структури												
Тема 10. Вказівники. Операції з вказівниками. Практичне застосування вказівників.	19	2	4	4		9						
Тема 11. Використання вказівників для роботи з масивами. Динамічні масиви.	16	2	2	2		10						
Тема 12. Функції користувача. Масиви і функції. Передача масиву у функцію.	14	2	2	2		8						
Тема 13. Рекурсивні алгоритми.	12	2	2	2		6						
Тема 14. Структури. Масиви структур. Структури в структурах.	12	2	1	1		8						
Тема 15. Використання структур для роботи з функціями. Використання вказівників для роботи зі структурами.	12	2	1	1		8						
Разом за змістовим модулем 3	85	12	12	12		49						
Усього годин	**195**	**30**	**30**	**30**		**105**						

Тематика лекційних занять з переліком питань

№	Назва теми з основними питаннями
1.	**Мова програмування С. Основи** 1. Поняття алгоритму та програми 2. Різниця між void main() і int main() 3. Структура програми на С 4. Змінні та типи даних 5. Базові операції над змінними 6. Інкрементні та декрементні операції 7. Коментарі в коді 8. Форматований ввід/вивід 9. Особливості написання коду на С 10. Практичні приклади
2.	**Логічні оператори і вирази. Розгалуження** 1. Логічні оператори та булева логіка в С 2. Операції відношення 3. Логічні операції 4. Пріоритет операцій 5. Особливості булевих виразів у С 6. Умовний вираз (?:) 7. Умовний оператор (if, else) 8. Вкладені умови 9. Оператор switch 10. Практичні приклади 11. Порівняння з іншими мовами

3.	**Циклічні конструкції** 1. Поняття циклу 2. Цикл із лічильником (for) 3. Цикли за умовою 4. Вкладені цикли 5. Практичні приклади 6. Порівняння з іншими мовами
4.	**Алгоритми поєднання розгалуження та повторення. Вкладені цикли. Покрокове введення та виведення даних. Рекурентні послідовності** 1. Поєднання розгалужень із циклами 2. Вкладені цикли 3. Поєднання з розгалуженнями 4. Покрокове введення та виведення даних 5. Використання while для невизначеної кількості даних 6. Рекурентні послідовності 7. Практичні приклади 8. Порівняння з іншими мовами
5.	**Одновимірні масиви. Найпростіші алгоритми роботи з одновимірними масивами.** 1. Пошук заданого елемента, пошук мінімального/максимального елемента 2. Поняття масиву 3. Оголошення та ініціалізація масивів 4. Базові операції з масивами 5. Найпростіші алгоритми роботи з одновимірними масивами 6. Розширені приклади роботи з масивами 7. Порівняння з іншими мовами 8. Рекомендації щодо використання
6.	**Додаткові способи введення даних** 1. Робота з файлами та потоками 2. Генерація випадкових чисел 3. Ініціалізація масивів константами 4. Практичні приклади 5. Порівняння з іншими мовами
7.	**Двовимірні масиви. Базові алгоритми для обробки елементів двовимірних масивів** 1. Поняття двовимірного масиву 2. Оголошення та ініціалізація двовимірних масивів 3. Введення та виведення двовимірних масивів 4. Базові алгоритми обробки двовимірних масивів 5. Порівняння з іншими мовами
8.	**Рядки і символьні масиви. Стандартні функції для роботи з рядками** 1. Поняття рядка в мові С 2. Основні операції з рядками 3. Стандартні функції бібліотеки <string.h> 4. Практичні приклади 5. Порівняння з іншими мовами
9.	**Найпростіші алгоритми роботи з рядками. Вступ до алгоритмів роботи з рядками** 1. Основи алгоритмів із рядками 2. Практичні приклади
10.	**Вказівники. Операції з вказівниками. Практичне застосування вказівників** 1. Оператор адреси (&)

	2. Оператор розіменування (*) 3. Вказівники: оголошення та основи 4. Присвоювання значень вказівнику 5. Розіменування вказівників 6. Розіменування некоректних вказівників 7. Розмір вказівників. Залежність від архітектури 8. Практичне застосування вказівників 9. Виділення та звільнення пам’яті 10. Додаткові приклади 11. Порівняння з іншими мовами
11.	**Використання вказівників для роботи з масивами. Динамічні масиви** 1. Вказівники і масиви: схожість і відмінності 2. Передача масивів у функції 3. Адресна арифметика і індексація масивів 4. Динамічні масиви 5. Практичні приклади
12.	**Функції користувача** 1. Основи функцій: параметри та аргументи 2. Способи передачі аргументів 3. Повернення значень із функцій 4. Практичні приклади
13.	**Рекурсивні алгоритми** 1. Стек і купа: організація пам’яті в С 2. Рекурсія: основи та принципи 3. Рекурсивні алгоритми: приклади 4. Рекурсія vs Ітерація 5. Додаткові приклади
14.	**Структури** 1. Навіщо потрібні структури? 2. Оголошення та визначення структур 3. Доступ до членів структур 4. Ініціалізація структур 5. Вкладені структури 6. Розмір структур і вирівнювання
15.	**Використання структур для роботи з функціями. Використання вказівників для роботи зі структурами** 1. Передача структур у функції 2. Повернення структур із функцій 3. Використання вказівників зі структурами 4. Вкладені структури та вказівники 5. Практичні приклади

Тематика практичних занять з переліком питань

№	Назва теми	Перелік питань та практичних завдань
1.	**Мова програмування С. Основи та лінійні алгоритми.**	1. Структура програми на С, підключення бібліотеки <stdio.h>. 2. Оголошення змінних та форматований ввід/вивід (printf, scanf).

		3. Практичні завдання: Розробка програм для обчислення площі кола, переведення часу (секунди у год:хв:сек), обміну значень двох змінних та обчислення суми цифр числа.
2.	**Умовні оператори та розгалужені алгоритми.**	1. Логічні операції, операції відношення та тернарний оператор ?:. 2. Синтаксис if-else та багатоваріантний вибір switch. 3. Практичні завдання: Перевірка належності точки координатному квадранту/області, перевірка існування трикутника, розрахунок знижки та програмування калькулятора.
3.	**Циклічні програми та ітераційні обчислення.**	1. Застосування циклів for, while, do-while. 2. Практичні завдання: Обчислення добутку через додавання, реалізація схеми Горнера, знаходження максимуму послідовності, обчислення n!! та суми гармонічного ряду.
4.	**Комбіновані та рекурентні алгоритми.**	1. Вкладені цикли та покрокове введення/виведення. 2. Практичні завдання: Покрокове введення невизначеної кількості даних до появи 0, виведення геометричних фігур (трикутник, ромб), обчислення чисел Фібоначчі та перевірка числа на простоту.
5.	**Одновимірні масиви: базові алгоритми.**	1. Ініціалізація та обхід одновимірного масиву. 2. Практичні завдання: Пошук заданого елемента (лінійний пошук), знаходження мінімального й максимального елементів та їх індексів, підрахунок кількості парних/непарних елементів.
6.	**Додаткові способи введення даних та робота з файлами.**	1. Файлові потоки, відкриття/закриття файлів (fopen, fclose). 2. Використання генератора pseudo-випадкових чисел rand(). 3. Практичні завдання: Запис та читання текстових/двійкових файлів, генерація масиву випадкових чисел у файл та заміна символів у файлі.
7.	**Двовимірні масиви та алгоритми обробки матриць.**	1. Двовимірні масиви в пам'яті, вкладені цикли для обходу матриць. 2. Практичні завдання: Обчислення суми елементів кожного рядка та середнього арифметичного стовпчиків, пошук першого додатного елемента, обертання матриці на 90° та витягнення підматриці.
8.	**Опрацювання рядків за допомогою бібліотечних функцій.**	1. Символьні масиви та нуль-термінатор. 2. Використання функцій бібліотеки <string.h> (strlen, strcpy, strcat, strcmp, strchr). 3. Практичні завдання: Безпечне введення рядків, порівняння та конкатенація текстів, пошук підрядків.
9.	**Алгоритми аналізу та модифікації символьних рядків.**	1. Посимвольний аналіз та пряма маніпуляція кодами ASCII. 2. Практичні завдання: Розробка функцій для підрахунку кількості голосних літер, перетворення малих літер на великі, видалення пробілів із тексту та інверсія рядка.
10.	**Операції з вказівниками та адресація пам'яті.**	1. Застосування операторів адреси & та розіменування *. 2. Практичні завдання: Оголошення вказівників, зміна значень змінних через розіменування, обмін значень двох змінних у функції та ітерація по масиву за допомогою вказівників.
11.	**Адресна арифметика та динамічні масиви.**	1. Взаємозв'язок вказівників і масивів, адресна арифметика. 2. Використання функцій malloc, calloc, realloc, free. 3. Практичні завдання: Створення та заповнення

		динамічного одновимірного масиву заданого користувачем розміру, безпечне розширення масиву.
12.	**Функції користувача та способи передачі аргументів.**	1. Передача параметрів за значенням та за адресою. 2. Повернення значень та вказівників із функцій. 3. Практичні завдання: Написання функцій обчислення суми елементів масиву, виділення динамічного масиву у функції та захист даних за допомогою const.
13.	**Рекурсивні алгоритми та аналіз стеку.**	1. Принципи рекурсії, базовий випадок, структура фрейму стеку. 2. Практичні завдання: Реалізація рекурсивних функцій для обчислення факторіала, переведення числа у двійкову систему числення, обчислення суми цифр та порівняння продуктивності з ітерацією.
14.	**Структуровані типи даних (структури).**	1. Оголошення та ініціалізація структур struct, доступ через оператор .. 2. Практичні завдання: Опис структури для збереження даних про особистість/працівника, створення вкладених структур (наприклад, дата народження), визначення розміру через sizeof.
15.	**Робота зі структурами через вказівники та функції.**	1. Передача структур у функції за адресою, оператор ->. 2. Практичні завдання: Розробка функцій для оновлення полів структури, обчислення середини відрізка між двома точками, управління динамічним масивом структур (список студентів).

Тематика лабораторних занять з переліком питань

№	Назва теми	Зміст роботи та перелік завдань
1.	**Лінійне програмування та умовний оператор.**	1. Засвоєння структури програми та візуального проектування програм. 2. Засвоєння простих типів даних, операцій та виразів. 3. Засвоєння основних операторів мови програмування (присвоєння, складеного, умовної і безумовної передачі керування, вибору). 4. Отримання навичок в організації введення та виведення даних. 5. Отримання практичних навичок роботи в інтерактивному режимі.
2.	**Циклічні програми.**	1. Засвоєння операторів циклу. 2. Отримання практичних навичок розробки ітераційних алгоритмів і програм для наближеного обчислення нескінченних сум та пошуку членів у нескінченних послідовностях, що задовольняють певні умови.
3.	**Алгоритми обробки масивів.**	1. Засвоєння структурованих даних типу масив. 2. Отримання практичних навичок розробки традиційних обчислювальних алгоритмів обробки масивів даних. 3. Засвоєння операторів циклу та побудови програм з використанням вкладених циклів. 4. Отримання практичних навичок розробки традиційних обчислювальних алгоритмів обробки багатовимірних масивів.
4.	**Упорядкування і пошук даних.**	1. Засвоєння методів упорядкування і пошуку даних. 2. Отримання практичних навичок розробки алгоритмів і

		програм упорядкування і пошуку даних.
5.	**Організація роботи з файлами. Символьні рядки.**	1. Засвоєння структурованих даних типу символьний рядок і операцій над цими даними. 2. Отримання практичних навичок обробки і редагування текстів.
6.	**Вказівники. Динамічний розподіл пам'яті.**	1. Ознайомитися з концепцією вказівників та їх використанням у мові програмування С для роботи з пам'яттю. 2. Освоїти методи динамічного розподілу пам'яті за допомогою функцій calloc, malloc та звільнення пам'яті за допомогою free. 3. Навчитися застосовувати два способи доступу до елементів масиву при динамічному виділенні пам'яті: через індексацію [] та через оператор розіменування *. 4. Поглибити розуміння роботи з одновимірними масивами та стрічками при використанні вказівників. 5. Розвинути навички оптимізації програмного коду шляхом ефективного управління пам'яттю.
7.	**Створення і використання функцій.**	1. Ознайомлення з поняттям функції у програмуванні. 2. Дослідження особливостей локальних і глобальних змінних та механізм повернення значень. 3. Закріплення навичок структурованого програмування та оптимізації коду за допомогою функцій.
8.	**Рекурсивні підпрограми.**	1. Ознайомлення з основними принципами рекурсії. 2. Застосування рекурсивних алгоритмів, дослідження переваг та недоліків рекурсивного підходу в порівнянні з ітеративним методом. 3. Закріплення навичок програмування рекурсивних функцій та аналізу їх складності.
9.	**Структуровані типи даних.**	1. Ознайомлення з поняттями структури у програмуванні. 2. Вивчення принципів їх використання для організації та зберігання даних. 3. Створення, ініціалізування та робота із структурами. 4. Дослідження переваг у порівнянні з іншими типами даних. 5. Закріплення навичок роботи зі структурами через практичне виконання завдань.

Індивідуальні науково-дослідні завдання (ІНДЗ)

Індивідуальні завдання з навчальної дисципліни не передбачено.

Завдання для самостійної роботи студентів

№	Назва теми	Завдання для самостійної роботи	Кіль кість годин
1.	**Мова програмування С. Основи.**	Дослідження стандартів мови С (ANSI C, C99, C11, C23). Відмінності між void main() та int main(), специфікатори форматованого вводу/виводу printf/scanf, правила опрацювання розгалужень формату та коментарів.	4
2.	**Логічні оператори і**	Моделювання булевої логіки цілими числами в С.	6

	вирази. Розгалуження.	Пріоритет арифметичних та логічних операцій. Вивчення "лінивих обчислень" (short-circuit evaluation) та застереження щодо вкладених умов if-else й оператора switch.	
3.	**Циклічні конструкції.**	Порівняльний аналіз циклів із передумовою (while), післяумовою (do-while) та лічильником (for). Програмування обчислення наближених сум ітераційних рядів, алгоритму Горнера та подвійного факторіала n!!.	6
4.	**Алгоритми поєднання розгалуження та повторення. Рекурентні послідовності.**	Побудова алгоритмів обробки невизначеної кількості даних (з умовою зупинки). Опрацювання рекурентних послідовностей (числа Фібоначчі, алгоритм Евкліда) без використання рекурсії. Вкладені цикли для форматування геометричних фігур.	6
5.	**Одновимірні масиви. Базові алгоритми.**	Механізми розміщення статичних масивів у пам'яті. Реалізація алгоритмів лінійного пошуку, обчислення крайніх значень (min/max), реверсу масиву та перевірки впорядкованості елементів.	6
6.	**Додаткові способи введення даних. Файли та випадкові числа.**	Опрацювання принципів роботи з файловими потоками FILE*. Використання функцій fopen, fclose, fprintf, fscanf, fgets, fputs. Використання rand() та srand(time(NULL)) для генерації масивів.	6
7.	**Двовимірні масиви. Базові алгоритми обробки.**	Порядок зберігання матриць у пам'яті за рядками. Алгоритми обходу елементів за стовпчиками та рядками, обчислення діагональних елементів квадратних матриць, обертання матриці на 90° та транспонування.	8
8.	**Рядки і символьні масиви. Бібліотека <string.h>.**	Роль нуль-термінатора \0 у символьних масивах. Опрацювання стандартних функцій: strlen, strcpy, strncpy, strcat, strncat, strcmp, strchr, strstr. Забезпечення безпеки буфера при введенні рядків (fgets замість gets).	7
9.	**Найпростіші алгоритми роботи із символьними та рядковими величинами.**	Алгоритми поелементного аналізу та трансформації текстів: інверсія рядка, конвертація регістру ASCII, підрахунок та видалення заданих символів/пробілів, перевірка рядка на паліндром.	7
10.	**Вказівники. Операції з вказівниками.**	Фізична адресація пам'яті, оператори адреси (&) та розіменування (*). Типові помилки під час роботи з неініціалізованими ("сміттєвими") вказівниками. Робота з вказівниками на вказівник (**ptr).	9
11.	**Використання вказівників для роботи з масивами. Динамічні масиви.**	Адресна арифметика та концепція "розпаду масиву у вказівник". Динамічний розподіл пам'яті в купі (malloc, calloc, realloc, free). Запобігання витокам пам’яті та утворенню "висячих" вказівників.	10
12.	**Функції користувача.**	Оголошення прототипів та визначення функцій. Передача аргументів за значенням та за адресою	8

	Передача аргументів.	(через вказівники). Використання модифікатора const для захисту вхідних даних. Заборона повернення адрес локальних змінних.	
13.	**Рекурсивні алгоритми та організація пам'яті.**	Аналіз сегментів пам’яті програми (Код, BSS, Data, Стек, Купа). Механізм фреймів стеку викликів (LIFO, push/pop). Базовий випадок у рекурсії. Причини та запобігання переповненню стеку (stack overflow).	6
14.	**Структури. Вкладені структури.**	Визначення користувацьких типів даних struct. Оператор вибору члена ., списки ініціалізації. Особливості вирівнювання пам’яті компілятором (padding) та розрахунок розміру через sizeof.	8
15.	**Використання структур із функціями та вказівниками.**	Передача екземплярів структур у функції за адресою. Оператор стрілочки -> для доступу до полів через вказівник. Масиви структур та динамічне виділення пам’яті під структури.	8
	Разом		**105**

Методи навчання

Під час вивчення дисципліни «Програмування» використовуються словесні методи навчання (розповідь, лекція-пояснення, діалог, дискусія), метод мультимедійних презентацій, демонстрація розбору й виконання програмного коду в IDE, наочні методи (візуалізація алгоритмів, трасування стеку викликів, схеми розподілу динамічної пам'яті). Практичні методи охоплюють розв'язання алгоритмічних задач із подальшим рев'ю коду, виконання лабораторних робіт на ПК із індивідуальним захистом, налагодження програм (дебагінг), самостійну роботу та онлайн-тестування на платформі Moodle. Проте основне навчання відбувається за допомогою виконання 9 лабораторних робіт і написанням програм на практичних заняттях.

У разі дистанційного навчання лекції, консультації та захист лабораторних робіт проводяться у режимі відеоконференції в Google Meet з демонстрацією екрана й презентацій, а навчальні матеріали, звіти зі скріншотами виконання програм та тести розміщуються на платформі Moodle (moodle.chnu.edu.ua).

Система контролю та оцінювання, критерії оцінювання поточного та підсумкового контролю

Форми проведення поточного контролю та періодичність їх здійснення визначаються робочим планом викладача відповідно до програми дисципліни. Поточний контроль проводиться у формі тематичного комп’ютерного тестування за темами 1-3, 5-9, 10-15 та захисту (заліків) за виконані лабораторні роботи №1-9.

Форми підсумкового семестрового контролю визначаються навчальним планом спеціальності F1 «Прикладна математика». Для дисципліни «Програмування» встановлено семестровий іспит по завершенню вивчення курсу.

Підсумковий іспит проводиться з використанням електронного навчального середовища Moodle ЧНУ і складається з двох частин:

- Тестова частина – максимальна оцінка 15 балів.

- Практична частина – максимальна оцінка 25 балів.

Сума балів, отриманих за поточний (модульний) контроль (максимум 60 балів), та балів, отриманих на іспиті (максимум 40 балів), становить підсумкову семестрову оцінку за 100-бальною шкалою.

Тестова частина іспиту (тестовий блок у Moodle) містить 15 тестових питань із матеріалів курсу, що розглядалися на лекціях та під час виконання лабораторних робіт. Правильна відповідь на кожне питання оцінюється в 1 бал. Питання обираються системально-випадковим чином із підготовленого банку питань.

Практична частина іспиту (практичний блок у Moodle) перевіряє вміння розробляти, налагоджувати та аналізувати програмне забезпечення засобами мови С. Кожен здобувач отримує білет із трьох практичних задач, що обираються випадковим чином із заздалегідь підготовленого банку завдань. Написані програми демонструються у середовищі розробки на ПК, після чого здобувач оформлює та подає звіт із програмним кодом і скріншотами результатів виконання.

Критерії оцінювання практичної частини іспиту:

- Задача № 1 оцінюється до 5 балів;
- Задачі № 2 та № 3 оцінюються до 10 балів кожна;
- За похибки та незначні помилки у поясненнях або назвах функцій/змінних, які не призвели до принципового викривлення алгоритму чи логіки відповіді, знімається до 3 балів;
- Часткова відповідь, у якій відсутнє повне розуміння базових понять та обґрунтування отриманих результатів, оцінюється не вище 40% від максимальної кількості балів за відповідну задачу.

Розподіл балів, які отримують студенти

Поточне тестування та самостійна робота												Іспит	Сума
Змістовий модуль №1			Змістовий модуль №2				Змістовий модуль №3						
ЛБ1	ЛБ2	Тести	ЛБ3	ЛБ4	ЛБ5	Тести	ЛБ6	ЛБ7	ЛБ8	ЛБ9	Тести	40	100
5	5	5	5	5	5	5	5	5	5	5	5		

Шкала оцінювання: національна та ЄКТС

Оцінка за національною шкалою	**Оцінка за шкалою ЄКТС**	
	Оцінка (бали)	**Пояснення за розширеною шкалою**
Відмінно	A (90-100)	відмінно
Добре	B (80-89)	дуже добре
	C (70-79)	добре
Задовільно	D (60-69)	задовільно

	E (50-59)	достатньо
Незадовільно	FX (35-49)	(незадовільно) з можливістю повторного складання
	F (1-34)	(незадовільно) з обов'язковим самостійним опрацюванням освітнього компоненту до перескладання

Нормативно-процесуальні умови оцінювання, допуску та апеляції

Оцінювання результатів навчання здобувачів вищої освіти здійснюється відповідно до Положення про організацію освітнього процесу у Чернівецькому національному університеті імені Юрія Федьковича (2026) та Положення про контроль і систему оцінювання результатів навчання здобувачів вищої освіти у Чернівецькому національному університеті імені Юрія Федьковича.

Підсумковою формою семестрового контролю з дисципліни є **екзамен**. Максимальна кількість балів за поточний контроль становить 60, за екзамен – 40. Підсумкова оцінка визначається як сума балів поточного та екзаменаційного контролю (максимум 100 балів) і переводиться в національну шкалу («відмінно», «добре», «задовільно», «незадовільно») та шкалу ЄКТС (A, B, C, D, E, FX, F).

Умови допуску до екзамену. Здобувач вищої освіти може бути не допущений до семестрового контролю, якщо у встановлені графіком освітнього процесу терміни він не підтвердив опанування на мінімальному рівні запланованих результатів навчання. Підставами для недопуску можуть бути недосягнення мінімального порогового рівня тих результатів навчання, які неможливо оцінити під час екзамену, або недостатність набраних балів для отримання позитивної підсумкової оцінки навіть за максимального результату екзамену. Рішення про недопуск за поданням завідувача кафедри (викладача) приймає декан факультету або уповноважений заступник декана з навчально-методичної роботи. Рішення оприлюднюється до початку відповідного контролю.

Ліквідація академічної заборгованості та апеляція.

- Здобувачі, які отримали під час семестрового контролю незадовільні оцінки FX (35-49 балів) або F (1-34 бали), мають право ліквідувати академічну заборгованість у визначені терміни до початку наступного навчального семестру. Для оцінки F перед перескладанням передбачено обов'язкове самостійне повторне опрацювання освітнього компонента.
- Повторне складання екзамену допускається не більше двох разів із кожної дисципліни: перший раз – викладачу, другий – комісії, яку створює декан факультету. Якщо під час складання екзамену комісії здобувач знову отримує незадовільну оцінку (FX або F), він відраховується з університету або залишається на повторний курс. Рішення комісії з перескладання є остаточним. Перескладання семестрового контролю з метою підвищення позитивної оцінки не допускається.

- У разі незгоди з оцінкою, отриманою під час семестрового екзамену, здобувач має право особисто подати письмову апеляційну заяву декану факультету протягом двох робочих днів після оголошення результатів підсумкового оцінювання. Порядок подання та розгляду апеляції визначається Положенням про апеляцію на результати підсумкового семестрового контролю знань студентів Чернівецького національного університету імені Юрія Федьковича.

Перелік питань для самоконтролю та підсумкового контролю

1. Поняття алгоритму та програми.
2. Різниця між void main() і int main().
3. Структура програми мовою С.
4. Змінні та типи даних у С.
5. Базові операції над змінними.
6. Інкрементні та декрементні операції.
7. Використання коментарів у програмному коді.
8. Форматований ввід та вивід у С.
9. Особливості написання коду мовою С.
10. Логічні оператори та булева логіка у С.
11. Операції відношення та їх використання.
12. Пріоритет логічних і арифметичних операцій.
13. Умовний вираз ?:.
14. Умовні оператори if, else, вкладені умови.
15. Оператор switch та його застосування.
16. Поняття циклу та його різновиди.
17. Цикл із лічильником (for).
18. Цикли за умовою (while, do…while).
19. Вкладені цикли та їх застосування.
20. Поєднання розгалужень і циклів.
21. Покрокове введення та виведення даних.
22. Використання циклів для обробки невизначеної кількості даних.
23. Рекурентні послідовності та способи їх реалізації.
24. Масиви: поняття, оголошення та ініціалізація.
25. Базові операції з одновимірними масивами.
26. Пошук заданого елемента в масиві.
27. Пошук мінімального та максимального елементів.
28. Алгоритми обробки масивів (сортування, пошук).
29. Додаткові способи введення даних у С.
30. Основи роботи з файлами та потоками.
31. Генерація випадкових чисел у програмах.
32. Ініціалізація масивів константами.
33. Поняття двовимірного масиву.
34. Оголошення та ініціалізація двовимірних масивів.
35. Введення та виведення елементів двовимірного масиву.

36. Базові алгоритми обробки двовимірних масивів.
37. Поняття рядка в мові С.
38. Символьні масиви та їх використання.
39. Основні функції бібліотеки <string.h>.
40. Алгоритми роботи з рядками (пошук, конкатенація, порівняння).
41. Практичні приклади роботи з текстами.
42. Поняття вказівника та його призначення.
43. Оператор адреси & та оператор розіменування *.
44. Оголошення вказівників та операції з ними.
45. Розіменування та некоректні вказівники.
46. Розмір вказівників залежно від архітектури.
47. Динамічний розподіл пам’яті (malloc, calloc, free).
48. Використання вказівників для роботи з масивами.
49. Адресна арифметика та індексація.
50. Передача масивів у функції.
51. Динамічні масиви: створення та використання.
52. Поняття функції у програмуванні.
53. Параметри та аргументи функцій.
54. Способи передачі аргументів у функцію.
55. Повернення значень із функцій.
56. Масиви та функції: передача масиву у функцію.
57. Використання рекурсії у програмуванні.
58. Рекурсія та ітерація: порівняння.
59. Приклади рекурсивних алгоритмів.
60. Стек і купа: організація пам’яті.
61. Структури: поняття, оголошення та ініціалізація.
62. Доступ до елементів структур.
63. Масиви структур та вкладені структури.
64. Використання структур у функціях.
65. Використання вказівників для роботи зі структурами.
66. Алгоритми сортування (методи впорядкування даних).
67. Алгоритми пошуку у структурованих даних.
68. Обробка текстів і символьних рядків.
69. Організація роботи з файлами: читання та запис.
70. Використання динамічного розподілу пам’яті для оптимізації коду.
71. Програмування та відлагодження ітераційних алгоритмів.
72. Програмування рекурсивних підпрограм.
73. Організація даних у структурованому вигляді.
74. Порівняння основних можливостей мови С з іншими мовами програмування.
75. Тестування та відлагодження програм у середовищі розробки.

Зарахування результатів неформальної освіти

Порядок та умови визнання й зарахування результатів навчання, здобутих шляхом неформальної та/або інформальної освіти, регламентуються чинним у Чернівецькому національному університеті імені Юрія Федьковича «Порядком визнання у Чернівецькому національному університеті імені Юрія Федьковича результатів навчання, здобутих шляхом неформальної та/або інформальної освіти».

Здобувачі вищої освіти мають право на визнання результатів навчання, отриманих під час проходження онлайн-курсів (на платформах Coursera, edX, Prometheus, Udemy, Stepik, Pluralsight тощо), участі у воркшопах, хакатонах, літніх/зимових школах з програмування за наявності підтверджуючих документів (сертифікатів, свідоцтв).

Механізм та умови визнання результатів:

1. **Перезарахування окремих тем / лабораторних робіт / поточного контролю:**
   - У разі часткової відповідності змісту онлайн-курсу або сертифікованої програми тематиці окремих змістових модулів дисципліни, здобувач звертається в усній формі до викладача, подаючи підтверджуючі документи.
   - За умови підтвердження відповідності результатів навчання здобувачу зараховується максимальний бал з відповідної теми (лабораторної роботи, тесту) та звільняється від виконання відповідних завдань. Загальний обсяг балів, який може бути зарахований здобувачу за результатами неформальної/інформальної освіти з окремих тем, не може перевищувати 25% від загальної кількості балів з дисципліни (до 25 балів зі 100).
2. **Повне зарахування освітнього компонента (підсумкового контролю):**
   - У разі необхідності повного зарахування навчальної дисципліни здобувач подає письмову заяву на ім’я декана факультету, до якої додає освітню декларацію про попереднє навчання та сертифікати.
   - Розгляд документів, співставлення результатів навчання із програмою дисципліни та оцінювання здійснює Предметна комісія, призначена розпорядженням декана.
3. **Процедура підтвердження та оцінювання:**
   - У разі прийняття Предметною комісією рішення про проведення контрольного заходу (співбесіди, контрольного завдання), здобувачу надається до 10 робочих днів для підготовки за питаннями і завданнями, що виносяться на контроль.
   - Загальний обсяг визнаних результатів неформальної освіти за весь строк навчання за ОПП «Технології програмування та комп’ютерне моделювання» (галузь знань F «Інформаційні технології») не може перевищувати 35% від загального обсягу освітньої програми.

Рекомендована література

Основна:

1. Данилюк І. М., Унгурян Г. М. Програмування. Мова С : навчальний посібник. Чернівці : Чернівецький нац. ун-т ім. Ю. Федьковича, 2025. 268 с. ISBN 978-617-8703-21-9. URL: https://archer.chnu.edu.ua/xmlui/handle/123456789/12958
2. Гюстедт Й. Modern C. 2nd ed. Shelter Island, NY : Manning Publications, 2019. 360 p. URL: https://gustedt.gitlabpages.inria.fr/modern-c/
3. Прокопенко О. В., Попов М. О., Чумак Г. Л. Мова програмування C/C++. Практикум : навчальний посібник. Київ : Київський національний університет імені Тараса Шевченка, 2024. 375 с. URL: https://iht.knu.ua/wp-content/uploads/2024/12/C_C-%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D1%83%D0%BC.pdf

Додаткова:

1. Мартін Р. Чистий код. Київ : Фабула, 2019. 416 с.
2. Іванов Є. О., Ліндер Я. М., Жереб К. А. Основи мови програмування C++ : посібник першокурсника. Київ : Логос, 2020. 90 с. URL: https://iss.csc.knu.ua/library/study-guides/foundations-of-c++-language.pdf

Інформаційні ресурси:

1. Електронний курс «Програмування». URL: https://moodle.chnu.edu.ua/course/view.php?id=3450
2. Hall B. J. «Beej». Beej's Guide to C Programming. Version 0.10.5. 2026. URL: https://beej.us/guide/bgc/
3. ISO/IEC 9899:2024. Programming languages - C (working draft N3220). Geneva : ISO/IEC JTC 1/SC 22/WG 14, 2024. 759 p. URL: https://www.open-std.org/jtc1/sc22/wg14/www/docs/n3220.pdf
4. Mike Banahan, Declan Brady, Mark Doran. The C Book. URL: http://publications.gbdirect.co.uk/c_book/
5. Steve Oualline. C Elements of Style. URL: http://www.oualline.com/books.free/style/index.html
6. Jens Gustedt. Modern C. URL: http://icube-icps.unistra.fr/img_auth.php/d/db/ModernC.pdf
7. cppreference.com : C Programming Language reference. URL: https://en.cppreference.com/w/c
8. Brian Gough. An Introduction to GCC. URL: http://www.network-theory.co.uk/docs/gccintro/

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-yuriia-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-vyiavlennia-ta-zapobihannia-akademichnomu-plahiatu-u-chernivetskomu-natsionalnomu-universyteti-imeni-yuriia-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-yuriia-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.