Чернівецький національний університет імені Юрія Федьковича

(повне найменування головного закладу вищої освіти)

Відокремлений структурний підрозділ «Фаховий коледж

Чернівецького національного університету імені Юрія Федьковича»

(повна назва відокремленого структурного підрозділу)

Циклова комісія *комп'ютерна інженерія*

(назва циклової/предметної комісії)

"ПОГОДЖЕНО"
Завідувач
Природничого відділення
(підпис) *Христина МЕЛЬНИЧУК* (ім'я та прізвище)
"27" 08 20 26 року

"ЗАТВЕРДЖУЮ"
Заступник директора
з навчально-методичної роботи
(підпис) *Микола ДЕРЕВ'ЯНЧУК* (ім'я та прізвище)
"28" 08 20 26 року

ПРОГРАМА

навчальної дисципліни

"Програмування Ч.3 Спеціалізовані мови програмування"

(вкажіть назву навчальної дисципліни (іноземною, якщо дисципліна викладається іноземною мовою))

обов'язкова

(вказати: обов'язкова/вибіркова)

Освітньо-професійна програма

"Комп'ютерна інженерія"

(назва освітньо-професійної програми)

Спеціальність

123 Комп'ютерна інженерія

(код і назва спеціальності)

Галузь знань

12 Інформаційні технології

(код і назва галузі знань)

Освітньо-професійний ступінь

фаховий молодший бакалавр

(назва освітньо-професійного ступеня)

Мова навчання

українська

(вказати: на якій мові читається предмет)

Чернівці, 2026

Програма навчальної дисципліни *"Програмування Ч.3 Спеціалізовані мови програмування"*

(назва навчальної дисципліни)

складена відповідно до освітньо-професійної програми *"Комп'ютерна інженерія"*

(назва освітньо-професійної програми)

затвердженої Вченою радою Чернівецького національного університету імені Юрія Федьковича (Протокол № 12 від «2» вересня 2024 року) та введеної в дію наказом ректора № 315 від «28» серпня 2026 року.

Розробники: (вказати авторів, їхні посади, наукові ступені та вчені (педагогічні) звання)
викладач, І.С. Хамула, викладач, спеціаліст першої категорії, Г.П. Горський

Профайл викладача https://ce-college.chnu.edu.ua/tsyklova-komisiia/sklad-tsyklovoi-komisii/khamula-illia-serhiiovych/
https://ce-college.chnu.edu.ua/tsyklova-komisiia/sklad-tsyklovoi-komisii/horskyi-hennadii-petrovych/

E-mail i.hamula@chnu.edu.ua, h.horskiy@chnu.edu.ua

Сторінка курсу в Moodle https://moodle.chnu.edu.ua/enrol/index.php?id=9406

Робоча програма обговорена та узгоджена на засіданні циклової комісії *комп'ютерної інженерії*

Протокол № 1 від "27" 08 2026 року

Голова циклової комісії _____ *Олександр ТАЩУК*
(підпис) (ім'я та прізвище)

Схвалена Методичною радою

Протокол № 1 від "27" 08 2026 року

Голова методичної ради _____ *Ольга БІЛОКРИЛА*
(підпис) (ім'я та прізвище)

ПЕРЕЗАТВЕРДЖЕНО

Протокол № ____ від _____, 20 __ р. _____ _____
(підпис) (ініціали та прізвище голови ЦК)

Протокол № ____ від _____, 20 __ р. _____ _____
(підпис) (ініціали та прізвище голови ЦК)

Протокол № ____ від _____, 20 __ р. _____ _____
(підпис) (ініціали та прізвище голови ЦК)

1. Мета та завдання навчальної дисципліни

Анотація: Дисципліна «Спеціалізовані мови програмування» спрямована на формування у студентів системного розуміння спеціалізованих мов (DSL), їх призначення, класифікації та місця у сучасних програмних і комп'ютерних системах. Розглядаються мови розмітки й опису інтерфейсів, запитів і роботи з даними, конфігурації та серіалізації, сценарної автоматизації, побудови й інфраструктури, наукових обчислень, логічного та функціонального програмування, комп'ютерної графіки, паралельних обчислень і опису апаратури. Окрема увага приділяється принципам проєктування власної DSL, визначенню синтаксису й семантики та базовим підходам до її реалізації.

Мета: формування у студентів цілісного уявлення про спеціалізовані мови програмування та декларативні засоби опису, що використовуються для розв'язання задач у різних предметних областях. Набуття знань, умінь і практичних навичок вибору та застосування відповідної DSL для опису даних, інтерфейсів, запитів, конфігурацій, автоматизації, збірки, інфраструктури, наукових обчислень, графічних і апаратних систем, а також ознайомлення з принципами проєктування власних спеціалізованих мов. Також метою є формування необхідного рівня теоретичної та практичної підготовки студентів для розуміння структури та призначення базових класів мови що входять до так званого ядра застосування цих знань на практиці для вирішення конкретних базових задач, вміння застосовувати відлагоджувач для пошуку помилок, все це стане базою для подальшого розвитку у напрямку програмування на мовах високого рівня , об'єктно-орієнтоване програмування, реалізація специфічних податків – «аплетів», то що.

Завдання:

- формування розуміння поняття предметно-орієнтованої мови (DSL), її відмінностей від мов загального призначення, переваг, обмежень і сфер застосування;
- набуття навичок використання мов розмітки, опису стилів та інтерфейсів, зокрема HTML, XML, Markdown, CSS, QML і XAML;
- оволодіння засобами роботи зі структурованими даними та запитами: SQL, JSON, YAML, TOML і Protocol Buffers;
- формування практичних навичок використання регулярних виразів, Bash, PowerShell та сценарних засобів для автоматизації типових задач;
- ознайомлення зі спеціалізованими мовами та форматами опису процесів збірки, контейнеризації й інфраструктури: Makefile, CMake, Gradle, Maven, Dockerfile, Docker Compose, Terraform та Ansible;
- ознайомлення зі спеціалізованими засобами для наукових обчислень, аналізу даних, логічного, функціонального та символьного програмування;
- формування розуміння мов програмування графічних процесорів, паралельних обчислень і опису цифрової апаратури на прикладах GLSL, HLSL, CUDA, Verilog та VHDL;
- набуття базових умінь проєктувати власну DSL, визначати її синтаксис і семантику та реалізовувати простий інтерпретатор або транслятор.
- ознайомлення з основними принципами та засадами для побудови програм на мові високого рівня "Java";

- підбір та встановлення необхідного програмного забезпечення яке потрібне для вивчення мови програмування, вирішення конкретних завдань та подальшого вдосконалення навичок програмування у цьому напрямі;.
- вивчення загальних принципів побудови з базових конструкцій мови, синтаксу, компіляції , інтерпритації та відлагоджування програм на мові високого рівня, за допомогою командного рядка та за інтегрованих середовищ розробки при вирішенні конкретних задач, необхідних для освоєння системи взаємопов'язаних дисциплін;.
- набуття теоретичних і практичних навичок проведення компіляції, інтерпритації, відлагоджування програм, математичних операцій, обчислення виразів , що потребують застосування оперцій галуженння та використання спеціальних класів та застосування методів, які реалізують іх фуекціонал, достатній для практичної діяльності за спеціальністю;
- формування у студентів спроможності подальшого самостійного засвоєння знань та вмінь, розвиток пізнавального хисту, освоєння раціональних прийомів оволодіння знаннями..

Предмет навчальної дисципліни: поняття та класифікація спеціалізованих мов програмування; мови розмітки HTML, XML, Markdown; декларативні мови опису стилів та інтерфейсів CSS, QML, XAML; SQL та розширені можливості роботи з реляційними даними; JSON, YAML, TOML і Protocol Buffers; регулярні вирази; Bash, PowerShell та Python-скрипти; Makefile, CMake, Gradle і Maven; Dockerfile, Docker Compose, Terraform і Ansible; R, MATLAB, NumPy і Pandas; Prolog; Lisp-подібні DSL і Wolfram Language; GLSL, HLSL і CUDA; Verilog і VHDL; принципи проєктування власної DSL, визначення синтаксису й семантики, лексичний і синтаксичний аналіз, AST, інтерпретація та трансляція.

Пререквізити: Дисципліна «Спеціалізовані мови програмування» може вивчатись студентами після набуття базових компетентностей у межах дисциплін «Алгоритмізація та програмування» та «Об'єктно-орієнтоване програмування». Бажаними є базове розуміння структур даних, принципів побудови програм, робота з командним рядком та вміння читати технічну документацію.

Дисципліна є складовою циклу професійної підготовки фахового молодшого бакалавра з комп'ютерної інженерії. Набуті знання можуть використовуватись під час подальшого вивчення баз даних, системного програмування, комп'ютерної графіки, комп'ютерних систем і мереж, автоматизованих систем, Інтернету речей та мікропроцесорної техніки, а також під час виконання курсових і кваліфікаційних робіт та проходження практики.

Студент повинен знати:

- основні поняття алгоритмізації, структур даних та базові конструкції мов програмування;
- принципи об'єктно-орієнтованого програмування та основні способи організації програмного коду;
- базовий синтаксис однієї або декількох мов загального призначення та відмінності між імперативним і декларативним описом;

- основи структурування програмних проєктів, роботи з файлами, конфігураціями та зовнішніми даними;
- призначення командного рядка, змінних середовища та базові принципи взаємодії програм з операційною системою;
- основні принципи читання технічної документації, специфікацій та описів форматів і мов.

Студент повинен вміти:

- аналізувати задачу та визначати, чи доцільно використовувати спеціалізовану мову для її розв’язання;
- створювати та редагувати документи, конфігурації та описи інтерфейсів за допомогою спеціалізованих декларативних мов;
- формувати SQL-запити та працювати з текстовими і бінарними форматами серіалізації даних;
- застосовувати регулярні вирази та сценарні мови для пошуку, обробки даних і автоматизації типових операцій;
- використовувати спеціалізовані засоби опису процесів збірки, контейнеризації, розгортання та інфраструктури;
- читати й модифікувати приклади коду спеціалізованих мов для наукових обчислень, логічного програмування, графічних і апаратних систем;
- порівнювати синтаксис, семантику та моделі виконання різних класів DSL;
- проєктувати просту спеціалізовану мову та реалізовувати базові етапи її обробки.

Перелік предметних компетентностей здобувача освіти:

Загальні:
ЗК.03. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.
ЗК.04. Здатність застосовувати знання у практичних ситуаціях.
ЗК.05. Здатність спілкуватися державною мовою як усно, так і письмово.
ЗК.07. Здатність працювати в команді.
ЗК.08. Здатність вчитися і оволодівати сучасними знаннями.

Спеціальні:
СК.02. Здатність застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування апаратних, програмних та інструментальних засобів комп'ютерної інженерії.
СК.03. Здатність вільно користуватись сучасними комп'ютерними та інформаційними технологіями, прикладними та спеціалізованими комп'ютерно інтегрованими середовищами для розробки, впровадження та обслуговування апаратних та програмних засобів комп'ютерної інженерії.

СК.04. Здатність брати участь у розробці системного та прикладного програмного забезпечення засобів комп'ютерної інженерії з використанням ефективних алгоритмів, сучасних методів і мов програмування.
СК.06. Здатність брати участь у модернізації апаратних та програмних засобів комп'ютерної інженерії.
СК.10. Здатність аргументувати вибір методів розв'язування спеціалізованих задач, критично оцінювати отримані результати, обґрунтовувати прийняті рішення.
СК.11. Здатність здійснювати вибір, розгортати, інтегрувати, діагностувати, адмініструвати та експлуатувати комп'ютерні системи та мережі, мережеві ресурси, сервіси та інфраструктуру організації.
СК.15. Здатність проектувати, розробляти та обслуговувати web-застосунки з динамічним контентом, використовуючи web-технології, технології комп'ютерної графіки та анімації.
СК.16. Здатність генерувати нові ідеї (креативність).
СК.17. Здатність до абстрактного мислення, аналізу і синтезу.

Програмні результати навчання:

РН.03. Знати сучасні методи та технології для розв'язання прикладних задач комп'ютерної інженерії.
РН.06. Тестувати, діагностувати та обслуговувати апаратні та програмні засоби комп'ютерної інженерії.
РН.08. Застосовувати знання технічних характеристик, конструктивних особливостей, призначення і правил експлуатації апаратних та програмних засобів комп'ютерної інженерії для вирішення технічних задач у професійній діяльності.
РН.09. Розробляти, тестувати, впроваджувати, експлуатувати програмне забезпечення для вбудованих і розподілених систем.
РН.10. Здійснювати пошук інформації з різних джерел для розв'язання задач комп'ютерної інженерії.
РН.18. Системно мислити та застосовувати творчі здібності до формування нових та нестандартних рішень при розв'язуванні задач комп'ютерної інженерії.
РН.19. Вміти використовувати методи аналізу та синтезу при розробці апаратних та програмних засобів комп'ютерної інженерії.

2. Опис навчальної дисципліни

2.1. Загальна інформація

Форма навчання	Рік підготовки	Семестр	Кількість			Кількість годин						Вид підсумкового контролю
			кредитів	годин	модулів	лекції	семінарські	практичні	лабораторні	самостійна робота	підготовка до екзамену	
Денна	4	7	3	90	3	34	-	-	34	22	–	залік
	4	8	3	90	3	24	-	-	24	12	30	екзамен

2.2. Дидактична карта навчальної дисципліни

Назви змістових модулів і тем	Кількість годин					
	денна форма					
	усього	у тому числі				
		л	п	лаб	інд	с.р.
1	2	3	4	5	6	7
I семестр						
Модуль 1. Основи DSL, мови розмітки, інтерфейсів і даних						
Тема 1. Вступ до спеціалізованих мов програмування	4	2	-	2	-	
Тема 2. Класифікація спеціалізованих мов	6	2	-	2	-	2
Тема 3. Мови розмітки: HTML, XML, Markdown	6	2	-	2	-	2
Тема 4. CSS, QML, XAML: мови опису стилів та інтерфейсів	6	2	-	2	-	2
Тема 5. SQL: спеціалізована мова роботи з реляційними даними	4	2	-	2	-	-
Тема 6. Розширені можливості SQL	6	2	-	2	-	2
Тема 7. JSON, YAML, TOML та Protocol Buffers	5	2	-	2	-	1
Разом за Модулем 1	37	14	-	14	-	9
Модуль 2. Автоматизація, збірка, інфраструктура та обчислення						
Тема 8. Регулярні вирази	5	2	-	2	-	1

Тема 9. Bash, PowerShell та Python-скрипти	4	2	-	2	-	
Тема 10. Makefile, CMake, Gradle та Maven	6	2	-	2	-	2
Тема 11. Dockerfile, Docker Compose, Terraform та Ansible	5	2	-	2	-	1
Тема 12. R, MATLAB, NumPy та Pandas	5	2	-	2	-	1
Тема 13. Prolog: логічне програмування як спеціалізований підхід	6	2	-	2	-	2
Разом за Модулем 2	31	12	-	12	-	7
Модуль 3. Декларативні, графічні й апаратні мови та проєктування DSL						
Тема 14. Lisp-подібні DSL та Wolfram Language	6	2	-	2	-	2
Тема 15. GLSL, HLSL та CUDA	6	2	-	2	-	2
Тема 16. Verilog та VHDL	5	2	-	2	-	1
Тема 17. Проєктування власної спеціалізованої мови	5	2	-	2	-	1
Разом за Модулем 3	22	8	-	8	-	6
Усього за І семестр	**90**	**34**	-	**34**	-	**22**

	Назва розділу (модуля), теми (змістового модуля), теми занять	*Кількість годин*							*Навч.-метод. література*	*Форми та засоби контролю*
		ВСЬОГО	*у тому числі:*							
			Лек.	*Сем.*	*Прак.*	*Лаб.*	*Інд.*	*С.р.*		
II семестр										
Модуль 1. Вступ. Принцип побудови простого додатку, класифікація типів, базові конструкції та масиви										
1	Вступ. Що таке Java? Виконання Java-програми. Поняття JVM.	2	2	-	-	-	-	-		
2	Що таке JDK? Що таке JRE? Як встановити JDK? Інтегровані середовища розробки.	4	2	-	-	2	-	-		Усне опит., тести
3	Перший додаток на Java. Опис синтаксичних особливостей мови Java. Коментарі. Константи.Символи. Рядки. Імена.	4	2	-	-	2	-	-		Усне опит., С.Р.
4	Цілі типи.Операції над цілими типами. Операції порівняння. Дійсні типи.Операції присвоєння. Умовна операція. Пріоритет операцій.	4	2	-	-	2	-	-		Усне опит., тести
5	Оператори. Оператори присвоєння. Умовний оператор. Оператор continue та мітки.Оператори циклу. Оператор continue та мітки. Оператор break. Оператор варіанта.	4	2			2				Усне опит., С.Р.
6	Масиви.Оголошення.Визначення.Ініціалізація. Створення безіменних масивів. Багатовимірні масиви. Оголошення. Визначення. Ініціалізація.	4	2	-	-	2	-	-		Усне опит., С.Р.
Разом за модуль 1		**22**	**12**	**-**	**-**	**10**	**-**	**-**		
Модуль 2. Парадигми та принципи реалізації ООП, специфікатори доступу, структура класів ядра, виключні ситуації, потоки та аплети										
7	ООП в Java. Парадигми програмування. Принципи ООП: Абстракція, Ієрархія. Інкапсуляція . Модульність. Опис класу та підкласу. Передача аргументів в метод. Перевантаження методів. Перевизначення методів. Реалізація поліморфізму. Абстрактні методи і класи. Остаточні методи і класи.	6	2	-	-	4	-	-		Усне опит.
8	Пакети. Пакет та підпакет. Права доступу до членів класу. Розміщення пакетів у файловій системі . Імпорт класів та пакетів. Java-файли. Інтерфейси. Варіанти наслідування. Перерахування. Аннотації.	4	2	-	-	2	-	-		Усне опит.
9	Клас Object. Конструктори класу. Операція new. Статичні члени класу static. Метод main(). Методи	4	2	-	-	2	-	-		Усне опит.

	зі змінною кількістю аргументів. Області видимості змінних. Вкладені класи.									
10	Обробка виключних ситуацій. Блоки перехоплення виключних ситуацій. Частина заголовка метода throws. Оператор throw. Ієрархія класів виключень. Порядок обробки виключень.	4	2	-	-	2	-	-		Усне опит., С.Р.
11	Підпроцеси. Клас Thread та його конструкторы. Синхронізація підпроцесів. Узгодження роботи декількох процесів. Пріоритети процесів. Процеси-демони. Групи підпроцесів.	4	2	-	-	2	-	-		Усне опит., С.Р.
12	Аплети . Особливості написання та роботи з html. Передача параметрів в аплет. Параметри тегу . Відомості про оточення аплета. Зображення та звук в аплетах. Відслідковування процесу завантаження. Класс MediaTracker. Захист від аплета. Аплети бібліотеки.	4	2			2				Усне опит.
Разом за модуль 2		**26**	**12**	**-**	**-**	**14**	**-**	**-**		
Модуль 3. Робота з рядками за допомогою спеціалізованих класів. Ієрархія класів-колекцій. Знайомство з класом вектор										
13	Робота із рядками. Клас String. Створення рядків. Зчеплення рядків. Маніпуляції зі рядками. Розбиття на підрядки. Порівняння рядків. Знаходження символу та підрядка. Заміна символу та підрядка. Метод trim(). Перетворення інших типів даних у рядок.	4	-	-	-	-		4		Усне опит., С.Р.
14	Клас StringBuilder. Конструкторы. Додавання та вставка підрядока. Видалення підрядока та символу. Заміна підряду. Перевертання рядка. Синтаксичний розбір рядку. Клас StringTokenizer.	4	-	-	-	-	-	4		
15	Класи-колекції структура ієрархії . Клас Vector , Створення вектора. Додавання елементу у вектор. Заміна елементу. Визначення розміру вектора. Звернення та перевірка наявності е-ту. Індекс е-ту. Видалення е-ту.	4	-	-	-	-	-	4		Усне опит., С.Р.
Разом за модуль 3		12	-			-		12		
Всього годин		**60**	**24**	**-**	**-**	**24**	**-**	**12**		**Екзамен**
Підготовка до екзамену		**30**						**30**		
Всього за курс		**180**	**58**			**58**		**64**		

3. Теми лекційних занять

№ з/п	Назва теми	Кількість годин
1.	**Тема 1. Вступ до спеціалізованих мов програмування.** 1. Поняття предметно-орієнтованої мови (DSL) 2. DSL та мови загального призначення 3. Приклади використання DSL у програмних і комп'ютерних системах	2
2.	**Тема 2. Класифікація спеціалізованих мов.** 1. Внутрішні та зовнішні DSL 2. Декларативні та імперативні DSL 3. Класифікація за предметною областю 4. Критерії вибору спеціалізованої мови	2
3.	**Тема 3. Мови розмітки: HTML, XML, Markdown.** 1. Структура HTML-документа 2. XML: структура, атрибути та well-formedness 3. Markdown: базові конструкції та практичне використання 4. Порівняння мов розмітки	2
4.	**Тема 4. CSS, QML, XAML: мови опису стилів та інтерфейсів.** 1. CSS: селектори, каскад і властивості 2. QML: декларативний UI та зв'язок із Qt 3. XAML: декларативний UI у .NET 4. Порівняння підходів CSS, QML та XAML	2
5.	**Теми 5–6. SQL та розширені можливості SQL.** 1. Основи реляційної моделі та SQL 2. SELECT, JOIN, GROUP BY, підзапити 3. CTE, віконні функції та транзакції 4. Індекси, подання та практичні аспекти SQL	4
6.	**Тема 7. JSON, YAML, TOML та Protocol Buffers.** 1. Формати структурованих даних 2. JSON, YAML і TOML: синтаксис та сфери застосування 3. Серіалізація та десеріалізація 4. Protocol Buffers і бінарне представлення даних	2
7.	**Тема 8. Регулярні вирази.** 1. Синтаксис регулярних виразів 2. Класи символів, групи та квантифікатори 3. Пошук, перевірка та заміна тексту 4. Практичне використання регулярних виразів	2
8.	**Тема 9. Bash, PowerShell та Python-скрипти. Сценарна автоматизація** 1. Bash: змінні, команди та керування потоком 2. PowerShell: об'єктний pipeline 3. Python як засіб автоматизації	2
9.	**Тема 10. Makefile, CMake, Gradle та Maven.**	2

	1. Автоматизація процесу збірки 2. Makefile: цілі та залежності 3. CMake: опис кросплатформної збірки 4. Gradle та Maven: проєкти і залежності	
10.	**Тема 11. Dockerfile, Docker Compose, Terraform та Ansible. Контейнеризація та Dockerfile** 1. Опис багатоконтейнерних систем у Docker Compose 2. Infrastructure as Code 3. Terraform та Ansible	2
11.	**Тема 12. R, MATLAB, NumPy та Pandas.** 1. Спеціалізовані середовища наукових обчислень 2. R та MATLAB 3. NumPy: багатовимірні масиви й обчислення	2
12.	**Теми 13–14. Prolog, Lisp-подібні DSL та Wolfram Language.** 1. Логічне програмування: факти, правила, запити 2. Уніфікація та пошук у Prolog 3. Lisp-подібні мови, S-вирази та макроси	4
13.	**Теми 15–16. GLSL, HLSL, CUDA, Verilog та VHDL.** 1. Програмування графічного конвеєра: GLSL і HLSL 2. Паралельні обчислення та модель CUDA 3. Мови опису апаратури	4
14.	**Тема 17. Проєктування власної спеціалізованої мови.** 1. Визначення предметної області та вимог 2. Синтаксис і семантика DSL 3. Лексичний та синтаксичний аналіз, AST 4. Інтерпретатор, транслятор і тестування DSL	2
Всього		**34**

№з/п	*Назва теми*	*Кількість годин*
1	**Вступ. Що таке Java? Виконання Java-програми. Поняття JVM.** 1. Походження назви мови та історія виникнення. 2. Опис процесу компіляції та інтерпретації, виконня програми на віртуальній машині. 3. Принцип функціонування віртуальної машини (JVM). 4. Використання та інтеграція JVM у різні ОС.	**2**
2	**Що таке JDK? Що таке JRE? Як встановити JDK? Інтегровані середовища розробки.** 1. Призначення та дитальний опис пакету JDK. 2. Хронологія розвитку пакету JDK. 3. Опис пакету JRE. 4. Опис процесу інсталяції пакету JDK на різних ОС. 5. Робота з пакетом JDK за допомогою терміналу. 6. Огляд інтгрованих середовищ розробки (IDE) та поєднання іх з пакетом JDK.	**2**
3	**Перший додаток на Java. Опис синтаксичних особливостей мови Java. Коментарі. Константи. Символи. Рядки. Імена.** 1. Огляд простого додатку на Java та розгляд синтаксичних особливостей його складу. 2. Великі та малі літери у програмі , загально прийняті правила використання. 3. Синтаксичні особливості при компіляції та інтерпритації за допомогою командного рядка. 4. Анотації та спеціальні коментарі як інструмент формування динамічної документції. 5. Різновиди констант, правила запису, осбливості використання. 6. Різновиди символів та іх опису, керуючі символи та їх використання. 7. Опис рядкі та методів маніпуляції з ними, різновиди кодування рядків. 8. Загально прийняті правила написання імен у Java, використання спец. символів у іменах.	**2**
4	**Цілі типи. Операції над цілими типами. Операції порівняння. Дійсні типи. Операції присвоєння. Умовна операція. Пріоритет операцій.** 1. Загальна схема типів використовуваних у Java, стислий огляд примітивних типів. 2. Логічний тип, операції на ним, таблиці істиності для логічних операторів, особливості використання. 3. Цілі типи , розмірності в байтах , діапазони допстимих значень, порядок оголошення та ініціалізації, двійкове представлення у памяті. 4. Арифметичні, логічні , побітові логічні операції над цілими числам, синтаксис та особливості застосування, можливі помилки при операціях 5. Дійсні типи , розмірності в байтах , діапазони допстимих значень, порядок оголошення та ініціалізації, визначення безмежності та невизначеності. 6. Особливості операцій над дісними типми, та првила приведення типів. 7. Операції присвоєння , порядок дії, поєднання з іншими. операціями(складові операції), можливі помилки при використанні.	**2**

	8. Умовна операція (тринарна), особливості роботи та синтаксису, застосування. 9. Особливості обчислення виразів, порядок присвоєння та розкриття дужок. 10. Приорітет операцій та їхне синтаксичне зображення.	
5	**Оператори. Оператори присвоєння. Умовний оператор. Оператор continue та мітки. Оператори циклу. Оператор continue та мітки. Оператор break. Оператор варіанта.** 1. Загальний набір операторів, стисла характеристика, осбливості та аналогії з іншими мовами прграмування. 2. Осбливості оператору присвоєння, синтаксис 3. Умовний оператор (if-then-else statement) , синтаксис та застосування , скорочений оператор (if-then statement) , осбливості використаня та комбінування з повним. 4. Оператори циклу (while, do-while) , синтаксис та застосування , відмінності. 5. Оператори циклу (for), варіації застосування, синтаксис. 6. Оператор (continue та мітка) , операtр (break), поєднання та застосування у складних конструкціях. 7. Оператор варіанта (switch- case, switch- case- default) , важливість застосуваня , операtру (break) в цих конструкціях.	**2**
6	**Масиви.Оголошення. Визначення. Ініціалізація. Створення безіменних масивів. Багатовимірні масиви. Оголошення. Визначення. Ініціалізація.** 1. Одномірні масиви , оголошеня, визначення розмірнсті (new). 2. Способи ініціалізації елементів масиву, використання арифметичних дій на цьому етапі. 3. Поєднання оголошеня та визначення, оголошеня та ініціалізації масиву в одній конструкції. 4. Створитрення безіменного масиву, синтаксис та використання. 5. Константа length , важливість та використання для роботи з масивами. 6. Цикл (for-each) для роботи з масивами, зручність застсування. 7. Багатовимірні масиви**,** оголошеня, визначення, ініціалізація. 8. Поєднання етапів оголошеня, визначення, ініціалізації в одну конструкцію у різних варіаціях. 9. Організація роботи з багоатовимірними масивами за допомогою вкладених циклів.	**2**
7	**ООП в Java. Парадигми програмування. Принципи ООП: Абстракція, Ієрархія. Інкапсуляція . Модульність. Опис класу та підкласу. Передача аргументів в метод. Перевантаження методів. Перевизначення матодів. Поліморфізм** 1. Об'єктно-орієнтоване пограмування в Java, історія та перадумови виникнення. 2. Парадигми програмування, що передували винекненню ООП та розповсюджнню на всі мови високого рівня. 3. Принципи ООП в загальному та конкретна реалізація засобами Java. 4. Абстракція, як рівень диталізації обєкту, адаптація під конкретні потреби. 5. Ієрархія, як спеціальний засіб ООП , скорочення обсягу програм 6. Модульність, як засіб взаємодії між класами , типи взаємодії між ними 7. Опис класу та підкласу, синтаксис та ключові слова (class, extends, this, super)	**2**

	8. Передача аргументів в метод, сигнатура метода, методи зі змінною кількістю параметрів 9. Перевантаження методів (overloading). 10. Перевизначення методів, як засіб поліморфізму, застосування . 11. Остаточні члени та класи	
8	**Клас Object. Конструктори класу. Операція new. Статичні члени класу static. Метод main(). Методи зі змінною кількістю аргументів. Області видимості змінних. Вкладені класи.** 1. 1. Клас Object , як вершина ієрархії класів, його методи та іх реалізація. 2. . Конструктори класу іх призначення, різновиди , застосування модифікаторів доступу , ззастосування ключових слів super() та this() . 3. Операція new при визначенні розмірності масиву та при створенні екземпляру класу, відмінності та прнцип застсування 4. Статичні члени класу static, різновиди, застосування та обмеження 5. Метод main(), призначення, обовязкові модифікатори та робота з аументами 6. Методи зі змінною кількістю аргументів, варіанти синтаксису та методиди роботи з аргументами в тілі циклу 7. Області видимості змінних , локальні , змінні класу, змінні екземпляру 8. Вкладені класи, означення та варіанти побудови, використання та звернення до елементів таких класів, безіменні класи , структура імен створених файлів після компіляції, критерії для доцільності використання.	**2**
9	**Пакети. Пакет та підпакет. Права доступу до членів класу. Розміщення пакетів у файловій системі . Імпорт класів та пакетів. Java-файли. Інтерфейси. Варіанти наслідування. Перерахування. Аннотації.** **1.** Пакети, призначення та розгляд структури пакетів на прикладі API . **2.** Пакет та підпакет , створення , структура доменних імен підпакетів, застосування повних імен для доступу до е-тів класу . **3.** Права доступу до членів класу, (private, protected і public, "package"), зведена таблиця **4.** Розміщення пакетів у файловій системі, файли та директорії, схематичне зображення структури варінти компіляції з різними ключам **5.** Імпорт класів та пакетів, використання повних та неповних імен, застосування спеціального символу «*» **6.** Java-файли , зведені правила для написання тексту програми та його структура. **7.** Інтерфейси (interface), означення, застосування механізмів множинного наслідування, варіанти наслідування , ромбовидне наслідування , реалізація за допомогою інтерфейсі та комбінації інтерфейсів та класів, труднощі його інтерпретації, використання, ефект поліморфізму, використання шнтерфейсів тільки з константами, для реалізації світлофора **8.** Перерахування , як самостійний тип, використання тільки з константами, з константами та методами, реалізація світлофора,	**2**

	використанн як структури вкладених класів **9.** Анотації , оголошення, інтерфейс-анотація, створення користувацьких анотацій , що містять змінні та методи, реалізція класами, застосування	
10	**Обробка виключних ситуацій. Блоки перехоплення виключних ситуацій. Частина заголовка метода throws. Оператор throw. Ієрархія класів виключень. Порядок обробки виключень.** **1**. Обробка виключних ситуацій, поимилки виконання(runtime), перехоплення умовними операторами. **2.** Блоки перехоплення виключних ситуацій , консрукція (try-catch) та її застосування, об'єкт-виключення (throws), консрукції (try-catch-finally, try-finally) та їх застосування, **3.** Частина заголовка метода throws, обвязковість застсування в окремих випадках, винятки (checked та . checked), класи винятків Error, RuntimeException **4.** Оператор throw, створення нових виключень(користувацьких) **5.** Ієрархія класів виключень, схематичне .зображення, класи Throwable, Error і Exception, RuntimeException місце та роль в ієрархії **6.** Порядок обробки виключень, блоки перехоплення (catch), поняття терміну (multi-catch), правило ієрархії , включення влсних винятків у ієрархію **7.** Частина заголовка методу throws, варіанти його застосування, декілька виключень, правило ієрархії **8.** Застосування оператору Оператор try{}catch() як вкладеного **9.** Оператор throw, як иструмнет викидання виключень в потрібному місці, **10.** Клас Throwable констуктори та методи, класи Error і Exception **11.** Побрядок обробки винтків та застсування multi-catch , синтаксис **12.** Створення користувацьких виключень, як нащадків , перевизначення методів	**2**
11	**Підпроцеси. Клас Thread та його конструкторы. Синхронізація підпроцесів. Узгодження роботи декількох процесів. Пріоритети процесів. Процеси-демони. Групи підпроцесів.** 1. Підпроцеси, загальні положення та концепції реалізації у різних ОС та JVM **2.** Клас Thread , константи стану та методи **3.** інтерфейс Runnable та метод . run () **4.** інтерфейс Callable та метод call() **5.** Додаткові засоби створення та керування підпроцесами java.util.concurrent, java.util.concurrent.atomic, java.util.concurrent.locks **6.** інтерфейсом Future та метод get(),isDone(),cancel(),isCancelled(). **7.** Абстрактний клас ForkJoinTask та метод adapt() **8.** інтерфейсом RunnableFuture у ієрархії та реалізація класом FutureTask **9.** інтерфейс Executor та метод execute (Runnable), **10.** інтерфейс ExecutorService та метод submit ( ) **11.** абстрактний клас AbstractExecutorService перевизначення методу newTaskFor() для використання методу submit ( ) **12.** абстрактний клас AbstractExecutorService та його звязок ThreadPoolExecutor **13.** клас ForkJoinPool його використання та звязок з класом ForkJoinTask, інтерфейсами Runnable або Callable, застосування методів execute(), invoke() та submit() **14.** абстрактні класи RecursiveTask та RecursiveAction принцип їїх роботити та перевизначений метод compute ()	**2**

	15. Конструктори класу Thread **16.** Конструктори класу Thread **17.** клас Executors що містить статичні методи, що створюють об'єкти типу ExecutorService для роботи у нових підпроцесах **18.** Створення декількох потоків та запуск на виконання , порядок завершення, **19.** Припинення роботи потоків користувачем **20.** Синхронізація підпроцесів **21.** Поняття так званго монітора (monitor) та застсування оператора synchronized () **22.** позначення метода словом synchronized та поняття (critical section) **23.** Поняття (mutually exclusive) виконання підпроцесів та механізм очікування та сповіщення (wait-notify) **24.** Методи sleep() та wait() спільні риси та відмінності **25.** Метод notify() та notifyall() **26.** Класи-колекції ArrayBlockingQueue та ArrayBlockingDeque , ConcurrentLinkedQueue та ConcurrentLinkedDeque , SynchronousQueue які забезпечують взаємодію підпроцесів , які реалізують інтерфейси BlockingQueue або BlockingDeque **27.** Приорітети підпроцесів, константи norm_priority, min_priority, max_priority іх значення та застосування **28.** Встановленя приорітету підпроцесів методом setPriority(int newPriority) перевірка методом checkAccess ( ) та виняток SecurityException, поняття пиорітету батьківського підпроцесу **29.** Підпроцеси-демони , метод setDaemon(true), checkAccess() і SecurityException , при роботі з такими процесами, нащадки цих процесів та метод setDaemon (false) **30.** Групи підпроцесів , конструктори класу ThreadGroup та використання екземплярів цього класу , як аргумету для конструкторів класу Thread, утворення ієрархії та вплив на неї операторм setMaxPriority(int maxPri) та обмеження maxPri у ієрархії	
12	**Аплети . Особливості написання та роботи з html. Передача параметрів в аплет. Параметри тегу . Відомості про оточення аплета. Зображення та звук в аплетах. Відслідковування процесу завантаження. Класс MediaTracker. Захист від аплета. Аплети бібліотеки.** **1.** Поняття аплетів та їх призначення **2.** Відмінності аплетів від автономних Java-додатків **3.** Середовище виконання аплетів і роль браузера **4.** Клас Applet та його місце в ієрархії AWT **5.** Аплети в бібліотеці Swing (JApplet) **6.** Життєвий цикл аплету **7.** Методи init(), start(), stop(), destroy() **8.** Відсутність методу main() в аплетах **9.** Створення найпростішого аплету **10.** Запуск аплету за допомогою HTML-тега <applet> **11.** Атрибути та параметри тега <applet> **12.** Передача параметрів в аплет через <param> **13.** Контекст аплету та взаємодія з браузером **14.** Налагодження аплетів і appletviewer **15.** Робота з графікою та зображеннями в аплетах **16.** Відтворення звуку в аплетах **17.** Безпека аплетів і модель «sandbox» **18.** Java Plug-in та сумісність аплетів **19.** Сучасний стан аплетів і альтернативні технології	**2**
	Разом	**24**

Контроль виконання завдань, винесених на підготовку до лекційних занять та опрацювання лекційного матеріалу, здійснюється в межах модульного контролю. Однією з опор для опрацювання лекційного матеріалу є конспект лекцій, що ведеться студентами.

Бали за цю роботу входять у загальну кількість балів за конкретний модуль.

4. Теми лабораторних занять

№ з/п	Коротка характеристика	Кількість годин
1.	Мови розмітки та структуровані формати: HTML, XML, Markdown (до Тем 3, 7)	4
2.	Декларативний UI: CSS, QML та XAML (до Теми 4)	4
3.	SQL: створення запитів та робота з реляційними даними (до Тем 5–6)	4
4.	JSON, YAML, TOML, Protocol Buffers та регулярні вирази (до Тем 7–8)	4
5.	Сценарна автоматизація: Bash, PowerShell і Python (до Теми 9)	4
6.	Автоматизація збірки та інфраструктури: CMake, Docker, Terraform/Ansible (до Тем 10–11)	4
7.	Спеціалізовані обчислювальні й логічні мови: NumPy/Pandas, Prolog, Lisp (до Тем 12–14)	4
8.	GLSL та основи проєктування власної DSL (до Тем 15–17)	6
Всього		**34**

№ з/п	*Назва теми*	*Кількість годин*
1	Лабораторна робота №1. Вступ. Підготовка робочого місця , встановлення JDK та IDE Компіляція, запуск простого додатку та аплету з командного та за допомогою IDE. Запуск аплета за допомогою HTML-файлу у Web-браузері.	**6**
2	Лабораторна робота №2. Робота з простими типами – опис, визначення. Опис масивів та їх використання. Різновиди опису рядків, властивості. Функції для роботи з рядками. Операнди:арифметичні, цілочисельні бітові, порівняння, логічні опис та застосування. Перетворення типів. Керуючі конструкції – принцип роботи та застосування. Використання циклів та їх різновиди.	**8**
3	Лабораторна робота №3. Клас. Інтерфейс. Успадкування одного інтерфейсу іншим. Реалізація інтерфейсів класами. Абстрактні класи. Механізми множинного спадкування. Поліморфізм. Створення конструкторів, блоків ініціалізації та інших елементів класу, Методи класу та їх виклик. Метод main() та робота його з параметрами.	**10**
4	Лабораторна робота № 4. Реалізація багатопоточності, застосування методу run, Пріоритет процесу, види пріоритетів. Робота з методами призначеними для потоків та їх функції. Обробка виключних ситуації за допомогою. Життєвий цикл аплету, застосування основних методів аплету, синхронізація. Подія та її обробка, огляд обробників подій.	**10**
	Разом	**24**

Контроль виконання завдань, винесених на підготовку до лабораторних занять здійснюється в межах відповідного заняття. Бали за цю роботу входять у загальну кількість балів за конкретне заняття.

5. Самостійна робота

Принципи організації самостійної роботи

- використання міждисциплінарних зв’язків з уже вивченими курсами;
- робота з фрагментами першоджерел, зокрема, й англомовними;
- втілення результатів виконання роботи у різних формах творчих завдань;
- діалогічність і ідейний взаємообмін (зокрема, позаурочний) студентів щодо змісту самосійної роботи; взаємне збагачення студентами знань одне одного через представлення власних тематичних траєкторій.

№ з/п	Назва теми	Короткий зміст
1.	Тема 1. Вступ до спеціалізованих мов програмування.	1. Приклади DSL у різних предметних областях
2.	Тема 2. Класифікація спеціалізованих мов.	1. Порівняння внутрішніх і зовнішніх DSL 2. Критерії вибору DSL
3.	Тема 3. Мови розмітки: HTML, XML, Markdown.	1. Створення HTML/XML/Markdown документів

		2. Валідація XML 3. Розширені можливості Markdown
4.	Тема 4. CSS, QML, XAML: мови опису стилів та інтерфейсів.	1. CSS-селектори та макетування 2. Огляд QML 3. Огляд XAML
5.	Теми 5–6. SQL та розширені можливості SQL.	1. JOIN та агрегатні функції 2. CTE і віконні функції
6.	Тема 7. JSON, YAML, TOML та Protocol Buffers.	1. Порівняння JSON, YAML і TOML 2. Схеми даних і Protocol Buffers
7.	Тема 8. Регулярні вирази.	1. Побудова та тестування регулярних виразів
8.	Тема 9. Bash, PowerShell та Python-скрипти.	1. Автоматизація типових задач у Bash/PowerShell 2. Робота з параметрами та pipeline
9.	Тема 10. Makefile, CMake, Gradle та Maven.	1. Створення CMake-конфігурації 2. Огляд Gradle і Maven 3. Керування залежностями
10.	Тема 11. Dockerfile, Docker Compose, Terraform та Ansible.	1. Створення Dockerfile 2. Опис сервісів у Docker Compose
11.	Тема 12. R, MATLAB, NumPy та Pandas.	1. Робота з масивами NumPy 2. Обробка таблиць Pandas
12.	Теми 13–14. Prolog, Lisp-подібні DSL та Wolfram Language.	1. Факти, правила і запити Prolog 2. S-вирази та символьні вирази
13.	Теми 15–16. GLSL, HLSL, CUDA, Verilog та VHDL.	1. Базові GLSL-шейдери 2. Огляд CUDA
14.	Тема 17. Проєктування власної спеціалізованої мови.	1. Проєктування синтаксису DSL 2. Побудова AST та простого інтерпретатора

№ з/п	*Змістовий модуль*	*Назва теми*	*Короткий зміст*
1.	Команди передачі керування	Робота із рядками. Клас String. Створення рядків. Зчеплення рядків. Маніпуляції зі рядками. Розбиття на підрядки. Порівняння рядків. Знаходження символу та підрядка. Заміна символу та підрядка. Метод trim(). Перетворення інших типів даних у рядок.	Робота із рядками в Java ґрунтується на використанні класу String, який призначений для зберігання та обробки текстових даних і є незмінним (immutable). Рядки можна створювати за допомогою літералів або конструкторів, а їх зчеплення виконується оператором + або методом concat(). Клас String надає широкий набір методів для маніпуляцій із рядками: отримання довжини,

			доступу до окремих символів, розбиття рядка на підрядки (substring(), split()), порівняння рядків (equals(), compareTo()), пошуку символів і підрядків (indexOf(), lastIndexOf()), а також заміни символів і підрядків (replace()). Метод trim() дозволяє видаляти пробіли на початку та в кінці рядка. Для перетворення інших типів даних у рядок використовуються методи String.valueOf() або автоматичне перетворення під час конкатенації з рядком.
		Клас StringBuilder. Конструкторы. Додавання та вставка підрядока. Видалення підрядока та символу. Заміна підряду. Перевертання рядка. Синтаксичний розбір рядку. Клас StringTokenizer.	Клас StringBuilder у Java призначений для роботи зі змінними рядками, які можна модифікувати без створення нових об’єктів, що підвищує ефективність програм. Конструктори класу дозволяють створювати порожній об’єкт, об’єкт із початковим рядком або із заданою ємністю буфера. Для роботи з рядками використовуються основні методи: append() для додавання підрядка в кінець, insert() для вставки у визначену позицію, delete() і deleteCharAt() для видалення підрядка або символу, replace() для заміни частини рядка та reverse() для перевертання рядка. Синтаксичний розбір рядка полягає у поділі тексту на окремі елементи для подальшої обробки. Для цього також використовується клас StringTokenizer, який дозволяє розбивати рядок на токени за заданими роздільниками та послідовно отримувати їх під час обробки.
		Класи-колекції, структура ієрархії . Клас Vector , Створення вектора. Додавання елементу у вектор. Заміна елементу. Визначення розміру вектора. Звернення та перевірка наявності е-ту. Індекс е-ту. Видалення е-ту.	Класи-колекції в Java призначені для зберігання та обробки груп об’єктів і входять до ієрархії **Java Collections Framework**, основою якої є інтерфейс Collection та його похідні. Клас Vector є динамічним масивом, який автоматично змінює розмір і підтримує синхронізацію. Вектор створюється за допомогою конструктора, елементи додаються методом add(), замінюються методом set(), а кількість елементів визначається методом size(). Доступ до елементів здійснюється методом get(), перевірка наявності виконується методом contains(), індекс елемента визначається методом indexOf(), а видалення елементів здійснюється методом remove().

6. Методи навчання

При вивченні дисципліни передбачене проведення лекцій, семінарських та практичних занять. Самостійна робота, пов’язана з опрацюванням відповідного теоретичного та практичного матеріалу, може супроводжуватись підготовкою конспектів, розв’язуванням завдань домашніх чи самостійних робіт, підготовкою проєктів.

Викладання дисципліни передбачає поєднання основних традиційних та інноваційних методів навчання з елементами електронного консультування, що передбачає використання сучасних інформаційних технологій, платформ та забезпечує засвоєння програмних результатів навчання.

Для досягнення освітньої мети й прогнозованих результатів навчання у дисципліні «Спеціалізовані мови програмування» використовуються такі технології та методи навчання:

1. Пояснювально-ілюстративний метод (виклад теоретичного матеріалу на лекціях, ілюстрація методів розв’язання завдань та прикладів застосування в професійній сфері, показ презентацій).

2. Метод проблемного навчання, що передбачає засвоєння студентами фундаментальних знань з дисципліни та пошуку способів розв'язання поставлених на лекції завдань.

3. Репродуктивний метод, що передбачає безпосереднє застосування набутих базових знань до демонстрації практичних умінь та навичок при розв'язуванні поставлених завдань.

4. Частково-пошуковий метод – організація пошуку об’єктів дослідження, розв'язування запропонованих викладачем завдань.

5. Дослідницький метод, що передбачає пошук розв'язку практичних завдань дисципліни з можливістю консультацій з викладачем.

6. Використання мультимедійних та хмарних технологій при організації комунікації з викладачем чи при онлайн-навчанні, а також використання платформи для електронного навчання Moodle (https://moodle.chnu.edu.ua).

7. Групові та індивідуальні методи роботи на практичних заняттях та при виконанні завдань самостійної роботи, формуванням напрацювань для виконання і захисту практичних завдань.

8. Інтерактивні методи навчання: робота в малих групах та тренінги; методи проєктів, кейс-метод; метод «мозкового штурму»; ділова гра, рольова гра та інші освітні технології;

9. Застосування електронних курсів та ресурсів, а також платформи для електронного навчання Moodle (https://moodle.chnu.edu.ua).

Отримані навички та знання в подальшому будуть використані під час виконання завдань практичного характеру.

7. Система контролю та оцінювання

Методи контролю

Методи та засоби оцінювання та демонстрування результатів навчання даної дисципліни наступні:

- поточні та модульні контрольні роботи з використанням різнорівневих завдань та стандартизованих тестів у системі дистанційного навчання Moodle;
- презентації результатів виконаних завдань;
- розрахункові, графічні, розрахунково-графічні роботи та аналітичні звіти щодо виконаних практичних завдань;
- захист творчих робіт (есе, творів-роздумів, проєктів, рефератів, доповідей);
- інші види індивідуальних та групових завдань.

У разі проведення навчального процесу та оцінювання в дистанційній формі використовуються засоби Moodle (у тому числі тестування).

Контроль самостійної роботи і оцінка її результатів включає:

- самоконтроль і самооцінку студента;
- контроль і оцінку з боку викладача, циклової комісії, державних кваліфікаційних комісій та ін.

Основними формами контролю самостійної роботи є:

- проведення екзамену;
- тестування;
- проведення модульних контрольних робіт;
- письмові чи усні опитування студентів;
- захист практичних завдань.

Форми контролю

Даною дисципліною передбачені наступні форми поточного контролю:

- усне опитування (фронтальне, індивідуальне та комбіноване), захист завдань практичного характеру, а також усні відповіді студентів;
- письмове опитування (модульні контрольні, тестування і самостійні роботи).

Форма підсумкового контролю – залік в І семестрі; екзамен в 2 семестрі.

Політика академічної доброчесності

Дотримання політики щодо академічної доброчесності учасниками освітнього процесу при вивченні освітнього компонента регламентовано чинними документами ВСП «Фаховий коледж ЧНУ»:

- Етичний кодекс Відокремленого структурного підрозділу «Фаховий коледж Чернівецького національного університету імені Юрія Федьковича»
- Положенням про виявлення та запобігання академічному плагіату у Відокремленому структурному підрозділі «Фаховий коледж Чернівецького національного університеті імені Юрія Федьковича»
- Політика використання штучного інтелекту в Чернівецькому національному університету імені Юрія Федьковича

Використання здобувачами фахової передвишої 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 порушенням інших вимог академічної доброчесності.

Розподіл балів, які отримують студенти

І семестр

Поточне оцінювання (*аудиторна та самостійна робота*)												Кількість балів	Кількість балів (залік)	Сумарна кількість балів
Модуль №1						Модуль № 2			Модуль № 3					
Лаб1	Лаб2	Лаб3	Лаб4	Лаб5	МКР	Лаб6	Лаб7	МКР	Лаб8	Лаб9	МКР	50	50	100
4	4	4	4	4	6	4	4	4	4	4	4			

ІІ семестр

Поточне тестування та самостійна робота					**Підсумковий екзамен**	**Сума балів**
Модуль 1	***Модуль 2***	***Модуль 3***	***Л.Р.***	***Всього***		
Т 1 – Т 6	Т 7 – Т 12	Т 12 – Т 15		50	50	100
5	7	3	35			

Критерії оцінювання відповідей студентів

Рівень знань	**Бали**	**Критерії оцінювання відповіді**
Високий	90 -100 % від максимальної кількості встановлених балів	Відповідь студента: - містить повний, розгорнутий, правильний виклад матеріалу з поставленого питання; - демонструє знання основних понять і категорій та взаємозв'язку між ними, правильне розуміння змісту основних теоретичних положень; - вказує на вміння давати змістовний та логічний аналіз матеріалу з поставленого питання; - містить послідовний та аргументований розв'язок задачі (завдання), правильно зроблені розрахунки до задачі; - здатність робити власні висновки в разі неоднозначності, спірного чи проблемного характеру поставленого питання чи проблеми.

Достатній	70-89 % від максимальної кількості встановлених балів	Студент дав досить змістовну відповідь на поставлене питання, але відповідь містить наступні недоліки: - недостатня повнота, незначні неточності чи прогалини при поясненні того чи іншого аспекту питання; - окремі формулювання є нечіткими, міститься інформація, котра не відноситься до змісту екзаменаційного питання; - алгоритм розв'язку задачі є правильним, однак допущені помилки при розрахунках.
Середній	50-69 % від максимальної кількості встановлених балів	Студент дав відповідь на поставлене питання, однак допустив суттєві помилки як змістовного характеру, так і при відповіді на питання: - зміст відповіді свідчить про прогалини у знаннях з відповідного питання або ж про неправильне розуміння окремих аспектів поставленого питання; - відповідь викладена недостатньо аргументовано та/або з порушенням правил логіки при поданні матеріалу; - порушено алгоритм розв'язку задачі і/або присутні помилки при розрахунках.
Початковий	0-49 % від максимальної кількості встановлених балів	Студент взагалі не відповів на питання, або його відповідь є неправильною, тобто містить грубі змістовні помилки щодо принципових аспектів поставленого питання. Аргументація відсутня взагалі (є безсистемною чи алогічною). Задача розв'язана неправильно.

8. Критерії оцінювання навчальних досягнень студентів під час проведення підсумкового контролю

В кінці вивчення дисципліни «Спеціалізовані мови програмування» студенти згідно із навчальним планом складають екзамен. Екзамен проводиться в усній або письмовій формах. Екзаменаційне завдання може бути у вигляді екзаменаційного білету, який включає 2 теоретичні питання, 3 тестові завдання, або тестування на платформі Moodle. Підсумкова оцінка знань студентів з дисципліни «Спеціалізовані мови програмування» здійснюється на основі результатів поточного, модульного та екзаменаційного контролю.

На оцінювання впливають такі моменти:

- рівень опанування основних проблемних питань відповідно до програми дисципліни та до завдань білетів;
- знання студентом ключових термінів і категорій;
- вміння вирішувати ситуаційні завдання та робити висновок;
- ступінь ознайомлення із основною та додатковою науковою літературою відповідно до програми дисципліни;
- вміння пов'язати теоретичні положення з практикою;
- здатність студента відстоювати власні погляди;
- вміння логічно у відведений час викласти основні положення запропонованих в екзаменаційному завданні питань і зробити відповідні висновки і узагальнення.

50,0-45,0 балів ставиться у випадках, коли студент має систематичні та глибокі знання навчального матеріалу, вміє без помилок виконувати практичні завдання, передбачені програмою дисципліни, засвоїв основу та ознайомився з додатковою літературою, викладає матеріал у логічній послідовності, робить узагальнення і висновки, наводить практичні приклади у контексті тематичного теоретичного матеріалу.

44,9-40,0 балів ставиться, якщо студент повністю засвоїв навчальний матеріал, знає основну літературу, вміє виконувати практичні завдання, викладає матеріал у логічній послідовності, робить певні узагальнення і висновки, але не наводить практичних прикладів у контексті тематичного теоретичного матеріалу або допускається незначних помилок у формулюванні термінів, категорій, розв'язанні практичних завдань.

39,9-35,0 балів ставиться, якщо студент повністю засвоїв навчальний матеріал, знає основну літературу, вміє виконувати практичні завдання, викладає матеріал у логічній послідовності, але не вміє робити певні узагальнення і висновки, не наводить практичних прикладів у контексті тематичного теоретичного матеріалу або допускається незначних помилок у формулюванні термінів, категорій, розв'язанні практичних завдань.

34,9-30,0 балів ставиться у випадках, коли студент засвоїв матеріал не у повному обсязі, дає неповні відповіді на поставлені теоретичне запитання, допустився грубих помилок у розв'язанні практичного завдання.

29,9-25,0 балів ставиться у випадках, коли студент засвоїв матеріал не у повному обсязі, дає неповні відповіді на поставлені теоретичне запитання, не вміє чітко узагальнити опрацьований матеріал, допустився грубих помилок у розв'язанні практичного завдання.

24,9-17,5 балів ставиться студенту, який не засвоїв навчальний матеріал, дає неправильні відповіді на більшість поставлених теоретичних запитань, не неправильно виконує розрахунки та розв'язувати практичні завдання.

17,4-0 балів ставиться студенту, який не засвоїв навчальний матеріал, дає неправильні відповіді на поставлені теоретичні запитання, не вміє виконувати розрахунки та розв'язувати практичні завдання.

Критерії оцінювання результатів навчання з навчальної дисципліни
Шкала оцінювання: національна та ЄКТС

<table>
<tr><td rowspan="2">Сума балів за всі види навчальної діяльності</td><td rowspan="2">Оцінка ЄКТС</td><td colspan="2">Оцінка за національною шкалою</td></tr>
<tr><td>для екзамену, курсового проекту (роботи), практики</td><td>для заліку</td></tr>
<tr><td>90-100</td><td>A</td><td>відмінно</td><td rowspan="5">зараховано</td></tr>
<tr><td>80-89</td><td>B</td><td rowspan="2">добре</td></tr>
<tr><td>70-79</td><td>C</td></tr>
<tr><td>60-69</td><td>D</td><td rowspan="2">задовільно</td></tr>
<tr><td>50-59</td><td>E</td></tr>
<tr><td>35-49</td><td>FX</td><td>незадовільно з можливістю повторного складання</td><td>не зараховано з можливістю повторного складання</td></tr>
<tr><td>0-34</td><td>F</td><td>незадовільно з обов'язковим самостійним повторним вивченням дисципліни</td><td>не зараховано з обов'язковим самостійним повторним вивченням дисципліни</td></tr>
</table>

9. Перелік питань для заліку

1. Що таке спеціалізована мова програмування (DSL) і для чого вона використовується?
2. У чому полягає відмінність DSL від мови програмування загального призначення?
3. Які основні переваги та обмеження використання DSL?
4. Які існують основні способи класифікації спеціалізованих мов?
5. У чому відмінність внутрішньої (embedded) та зовнішньої (external) DSL?
6. Наведіть приклади спеціалізованих мов у різних предметних областях.
7. Яке призначення HTML і які базові структурні елементи HTML-документа?
8. Яке призначення XML та чим XML відрізняється від HTML?
9. Що означає поняття добре сформованого (well-formed) XML-документа?
10. Для чого використовується Markdown і які його основні конструкції?
11. Яке призначення CSS та як працюють селектори CSS?
12. Що таке каскадування та специфічність у CSS?
13. Що таке QML і для яких типів застосунків він використовується?
14. Що таке XAML і в яких технологіях він застосовується?
15. Порівняйте декларативний опис інтерфейсу у QML та XAML.
16. Що таке SQL і які основні групи команд SQL існують?
17. Як працюють SELECT, WHERE, ORDER BY та GROUP BY?
18. Для чого використовуються JOIN і які основні типи JOIN існують?
19. Що таке підзапити та загальні табличні вирази (CTE)?
20. Що таке транзакція і які властивості ACID?
21. Яке призначення JSON та які типи значень він підтримує?
22. Які особливості YAML і в чому його відмінність від JSON?
23. Для чого використовується TOML?
24. Що таке Protocol Buffers та чим бінарна серіалізація відрізняється від текстової?
25. Що таке регулярний вираз і для яких задач він використовується?
26. Поясніть призначення метасимволів, квантифікаторів і груп у регулярних виразах.
27. Що таке жадібні та ліниві квантифікатори?
28. Яке призначення Bash як сценарної мови?
29. Які базові конструкції керування потоком використовуються у shell-скриптах?
30. Яке призначення PowerShell та чим його pipeline відрізняється від традиційного Unix pipeline?
31. У яких випадках Python доцільно використовувати як мову сценарної автоматизації?
32. Що таке Makefile і як у ньому описуються цілі та залежності?
33. Яке призначення CMake і чим він відрізняється від Make?
34. Для чого використовуються Gradle та Maven?

35. Що таке декларативний опис процесу збірки?
36. Яке призначення Dockerfile і які базові інструкції він містить?
37. Для чого використовується Docker Compose?
38. Що таке Infrastructure as Code?
39. Яке призначення Terraform і якою є базова структура його конфігурації?
40. Яке призначення Ansible і що таке playbook?
41. Для яких задач використовується мова R?
42. Які типові задачі розв’язуються у MATLAB?
43. Яке призначення NumPy у наукових обчисленнях Python?
44. Яке призначення Pandas та які основні структури даних він надає?
45. Що таке логічне програмування?
46. Які основні поняття Prolog: факти, правила та запити?
47. Як працює уніфікація та пошук розв’язку у Prolog?
48. Які особливості Lisp-подібних мов і S-виразів?
49. Що таке homoiconicity і чому вона важлива для Lisp-подібних DSL?
50. Яке призначення Wolfram Language та символьних обчислень?
51. Що таке шейдер і для чого використовується GLSL?
52. Які основні стадії графічного конвеєра програмуються за допомогою шейдерів?
53. Що таке HLSL і як він співвідноситься з GLSL?
54. Що таке CUDA і яка модель паралельного виконання використовується в CUDA?
55. Що таке мова опису апаратури (HDL)?
56. У чому відмінність мов Verilog/VHDL від звичайних мов програмування?
57. Які базові конструкції використовуються для опису комбінаційної та послідовнісної логіки?
58. Що таке синтаксис мови і чим він відрізняється від семантики?
59. Що таке лексичний аналіз і яку роль виконує лексер?
60. Що таке синтаксичний аналіз і яку роль виконує парсер?
61. Що таке абстрактне синтаксичне дерево (AST)?
62. У чому відмінність інтерпретатора від компілятора або транслятора?
63. Які етапи доцільно пройти під час проєктування власної DSL?
64. Які критерії слід враховувати при виборі синтаксису спеціалізованої мови?
65. Як перевірити коректність і зручність спроєктованої DSL на практиці?

II семестр

Питання до екзамену

Модуль 1. Вступ. Принцип побудови простого додатку, класифікація типів, базові конструкції та масиви

1. Вступ. Що таке Java? Виконання Java-програми.Поняття JVM.
2. Що таке JDK? Що таке JRE? Як встановити JDK? Інтегровані середовища розробки.
3. Перший додаток на Java. Опис синтаксичних особливостей мови Java.
4. Коментарі. Константи. Символи. Рядки.
5. Цілі типи. Операції над цілими типами. Операції порівняння.
6. Дійсні типи. Операції присвоєння. Умовна операція. Пріоритет операцій.
7. Оператори. Оператори присвоєння. Умовний оператор (if-then-else statement, if-then statement)
8. Оператор continue та мітки. Оператори циклу(while, do-while for) Оператор break. Оператор варіанта (switch-case).
9. Масиви. Оголошення. Визначення. Ініціалізація. Визначення масивів.
10. Створення безіменних масивів. Багатовимірні масиви. Визначення. Ініціалізація.

Модуль 2. Парадигми та принципи реалізації ООП, специфікатори доступу, структура класів ядра, виключні ситуації, потоки та аплети

11. Принципи ООП: Абстракція. Ієрархія.Інкапсуляція (incapsulation). Модульність.
12. Опис классу та підкласу. Передача аргументів в метод.
13. Перевантаження методів. Перевизначення методів.Реалізація поліморфізму.
14. Абстрактні методи і класи. Остаточні методи і класи (final)
15. Клас Object. Конструктори класу. Операція new. Статичні члени класу static.
16. Метод main(). Методи зі змінною кількістю аргументів.
17. Області видимості змінних. Вкладені класи(inner).
18. Пакети, Інтерфейси, Перерахування опис.
19. Пакет та підпакет . Права доступу до членів класу. Розмішення пакетів по файлах.
20. Імпорт класів та пакетів. Java-файли.Інтерфейси.
21. Варіанти наслідування. Реалізація перерахування.
22. Обробка виключних ситуацій (Exeption).Блоки перехоплення виключних ситуацій(try-catch),(t catch-finally).
23. Частина заголовка метода throws. Оператор throw.
24. Ієрархія класів виключень. Порядок обробки виключень
25. Підпроцесси. Класс Thread та його конструктори.Синхронізація підпроцесів (synchronized).
26. Узгодження роботи декількох процесів (wait(),notify()). Приорітети підроцесів.Процеси-демони(daemons).Групи підпроцесів.

Модуль 3. Робота з рядками за допомогою спеціалізованих класів. Ієрархія класів-колекцій. Знайомство з класом вектор

27. Робота з рядками. Клас String. Створення рядків. Зчеплення рядків.
28. Маніпуляції з рядками.
29. Розбиття на підрядки. Порівняння рядків. Знаходження символу та підрядка.
30. Клас StringBuilder. Конструктори. Додавання та вставка підрядка. Видалення підрядка та символу. Заміна підрядка.

31. Перевернення рядка. Синтаксичний розбір рядка. Клас StringTokenizer.
32. Класи- колекції. Клас Vector , Створення вектора. Додавання елементу у вектор. Заміна елеме
    Визначення розміру вектора. Звернення та перевіка наявності елементу. Індекс елементу.Видалення елементу
33. Розбиття на підрядки. Порівняння рядків. Знаходження символу та підрядка.
34. Клас StringBuilder. Конструктори. Додавання та вставка підрядка. Видалення підрядка та символу. Заміна підрядка.
35. Перевернення рядка. Синтаксичний розбір рядка. Клас StringTokenizer.
36. Класи- колекції. Клас Vector , Створення вектора. Додавання елементу у вектор. Заміна елеме
    Визначення розміру вектора. Звернення та перевіка наявності елементу. Індекс елементу.Видалення елементу

10. Рекомендована література

1. Martin Fowler. Domain-Specific Languages. — Addison-Wesley Professional, 2010.
2. Marjan Mernik, Jan Heering, Anthony M. Sloane. When and How to Develop Domain-Specific Languages. — ACM Computing Surveys, Vol. 37, No. 4, 2005.
3. Terence Parr. Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages. — Pragmatic Bookshelf, 2009.
4. Terence Parr. The Definitive ANTLR 4 Reference, 2nd Edition. — Pragmatic Bookshelf, 2013.
5. Dick Grune, Ceriel J.H. Jacobs. Parsing Techniques: A Practical Guide, 2nd Edition. — Springer, 2008.
6. Alan Beaulieu. Learning SQL, 3rd Edition. — O'Reilly Media, 2020.
7. John L. Hennessy, David A. Patterson. Computer Architecture: A Quantitative Approach, 6th Edition. — Morgan Kaufmann, 2017.

Інформаційні ресурси:

1. https://developer.mozilla.org — MDN Web Docs: HTML, XML, CSS та регулярні вирази.
2. https://doc.qt.io/qt-6/qmlapplications.html — офіційна документація Qt QML.
3. https://learn.microsoft.com/dotnet/desktop/wpf/xaml/ — документація Microsoft щодо XAML.
4. https://www.postgresql.org/docs/ — офіційна документація PostgreSQL і довідник з SQL.
5. https://www.rfc-editor.org/rfc/rfc8259 та https://yaml.org/spec/ — специфікації JSON і YAML.
6. https://protobuf.dev — офіційна документація Protocol Buffers.
7. https://www.gnu.org/software/bash/manual/ та https://learn.microsoft.com/powershell/ — документація Bash і PowerShell.
8. https://cmake.org/documentation/, https://docs.gradle.org та https://maven.apache.org/guides/ — документація CMake, Gradle і Maven.
9. https://docs.docker.com, https://developer.hashicorp.com/terraform та https://docs.ansible.com — документація Docker, Terraform і Ansible.
10. https://www.r-project.org, https://numpy.org/doc/, https://pandas.pydata.org/docs/ та https://www.mathworks.com/help/ — ресурси для R, NumPy, Pandas і MATLAB.
11. https://www.swi-prolog.org/pldoc/ — офіційна документація SWI-Prolog.
12. https://www.lispworks.com/documentation/HyperSpec/ та https://reference.wolfram.com/language/ — Common Lisp HyperSpec і Wolfram Language.
13. https://registry.khronos.org/OpenGL/specs/gl/GLSLangSpec.4.60.pdf та https://docs.nvidia.com/cuda/ — специфікація GLSL і документація CUDA.
14. https://www.antlr.org та https://craftinginterpreters.com — матеріали з проєктування парсерів, інтерпретаторів і власних мов.