КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Кафедра прикладної статистики

«ЗАТВЕРДЖУЮ»
Заступник декана
з навчальної роботи
__________ Кашпур О.Ф.
«28» 08 2020 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
ПАРАДИГМИ ПРОГРАМУВАННЯ

для студентів

галузь знань	**12 «Інформаційні технології»**
спеціальність	**124 «Системний аналіз»**
освітній рівень	**бакалавр**
освітня програма	**«Системний Аналіз»**
вид дисципліни	**вибіркова**

Форма навчання	**денна**
Навчальний рік	**2020/2021**
Семестр	**3, 4**
Кількість кредитів ECTS	**7**
з них семестр 3	**4**
семестр 4	**3**
Мова викладання, навчання та оцінювання	**українська**
Форма заключного контролю	**залік, іспит**

Пролонговано: на 20__/20__ н.р. __________(___________) «__»___ 20__р.
(підпис, ПІБ, дата)

на 20__/20__ н.р. __________(___________) «__»___ 20__р.
(підпис, ПІБ, дата)

КИЇВ – 2020

Розробник:

Шарапов Михайло Михайлович, к.ф.-м.н, доцент кафедри прикладної статистики,

ЗАТВЕРДЖЕНО
Завідувач кафедри Прикладної Статистики

_____________________ (Лебєдєв Є.О.)

Протокол № 1 від «27» серпня 2020 р.

Схвалено Гарантом освітньо-професійної програми першого рівня вищої освіти

«Системний аналіз» _______________ М.М. Шарапов

«28» серпня 2020 року

Схвалено науково-методичною комісією факультету комп'ютерних наук та кібернетики

Протокол від «28» серпня 2020 року № 1
Голова науково-методичної комісії _______________ (Омельчук Л.Л.)
(підпис) (прізвище та ініціали)

«28» серпня 2020 року

1. Мета дисципліни – засвоєння базових знань щодо використання різних парадигм програмування: функціональної, логічної, декларативної, об’єктно-орієтованої. Оволодіння навичками проектування та розробки програмних систем із застосуванням різноманітних парадигм програмування з відповідними структурами даних, механізмами управління та технологіями.

2. Попередні вимоги до опанування або вибору навчальної дисципліни:

1. Знати: основні поняття та концепції програмування, алгебру та математичний аналіз на базовому рівні (об’єм першого курсу університету), суть поняття алгоритму.

2. Вміти: створювати програми будь-якою мовою, читати та аналізувати математичні тексти, реалізовувіти прості алгоритми.

3. Володіти елементарними навичками: роботи з комп’ютером, пошуку інформації в інтернеті, користування системами перекладу.

3. Анотація навчальної дисципліни:

Навчальна дисципліна “Об’єктно-орієнтоване програмування” є складовою освітньо-професійної програми підготовки фахівців за першим (*бакалаврським*) рівнем вищої освіти. В її рамках досліджуються переваги та недоліки об’єктно-орієнтованого підходу, розглядаються підходи до ефективної розробки програмного забезпечення, практикується використання інструментів індустріальної розробки програмних систем, систем контролю версій. Також відбувається розвиток вмінь реалізації алгоритмів та вивчаються основи аналізу складності.

Дисципліна викладається у **3** та **4** семестрах **2** курсу в обсязі **210 год. (7 кредитів ЕCTS)** зокрема: *лекції – 54 год., лабораторні – 48 год., консультації – 2 год., самостійна робота – 106 год.* У курсі передбачено **3** *частини* та **3** *контрольні роботи*. Семестр **3** завершується **заліком**, семестр **4** завершується **іспитом**.

В результаті вивчення навчальної дисципліни студент повинен

знати: суть і підґрунтя парадигми логічного програмування та парадигми функціонального програмування, основні конструкції мов програмування Prolog та Haskell, суть декларативних засобів стосовно об'єктно-орієнтованої парадигми;

вміти:

- застосовувати логічний та функціональний стилі програмування при розв’язуванні програмістських задач;
- використовувати відомі та створювати власні декларативні засоби програмування стосовно об'єктно-орієнтованої парадигми, зокрема, на у Java та .Net.

програми «Системний аналіз» студент повинен опанувати компетентності та результати навчання, які надає дисципліна „Програмування” програми «Системний аналіз».

4. Завдання (навчальні цілі):

Отримання компетентностей в професійній розробці програмного забезпечення для індустріальних та академічних проектів. Відповідно до освітньої кваліфікації бакалавра з системного аналізу дисципліна спрямована на досягнення таких компетентностей випускника:

- Здатність застосовувати знання у практичних ситуаціях (К02)
- Здатність планувати і управляти часом (К03)
- Здатність до пошуку, оброблення та аналізу інформації з різних джерел (К07)
- Здатність працювати автономно (К10)
- Здатність до комп’ютерної реалізації математичних моделей реальних систем і про цесів; проектувати, застосовувати і супроводжувати програмні засоби моделювання,

прийняття рішень, оптимізації, обробки інформації, інтелектуального аналізу даних (К22)

- Здатність використовувати сучасні інформаційні технології для комп'ютерної реалізації математичних моделей та прогнозування поведінки конкретних систем а саме: об'єктно-орієнтований підхід при проектуванні складних систем різної природи, прикладні математичні пакети, застосування баз даних і знань (К23)
- Здатність організовувати роботу з аналізу та проектування складних систем, створення відповідних інформаційних технологій та програмного забезпечення (К24)

5. Результати навчання за дисципліною:

Результат навчання (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)		Форми (та/або методи і технології) викладання і навчання	Методи оцінювання та пороговий критерій оцінювання (за необхідності)	Відсоток у підсумковій оцінці з дисципліни
Код	**Результат навчання**			
РН1.1	*Знати суть і підґрунтя парадигми функціонального програмування, знати засоби підтримки функціонального стилю програмування та їх подання у мові програмування Haskell.*	*Лекція, лабораторне заняття*	*контрольна робота, іспит*	15%
РН1.2	*Знати суть і підґрунтя парадигми логічного програмування, знати засоби підтримки логічного стилю програмування.*	*Лекція, лабораторне заняття*	*контрольна робота, іспит*	5%
РН1.3	*Знати суть декларативних засобів стосовно об'єктно-орієнтованої парадигми, знати засоби декларативного програмування для .Net та Java.*	*Лекція, лабораторне заняття*	*контрольна робота, іспит*	5%
РН2.1	*Вміти застосовувати функціональний стиль програмування, використовуючи мову програмування Haskell.*	*Лабораторне заняття, самостійна робота*	*захист лабораторних робіт (ЛР), контрольна робота, іспит*	15%
РН2.2	*Вміти застосовувати логічне програмування, використовуючи Prolog-процесори, зокрема SWI-Prolog.*	*Лабораторне заняття, самостійна робота*	*захист ЛР*	20%
РН2.3	*Вміти використовувати декларативні засоби програмування для .Net та Java.*	*Лекція, лабораторне заняття, самостійна робота*	*захист ЛР, іспит*	10%

РН3.1	*Аргументувати власний вибір підходів до розв'язання задачі, спілкуватися з колегами з питань проектування та розробки програм*	*Лабораторне заняття*	*захист ЛР*	10%
РН4.1	*Організовувати свою самостійну роботу для досягнення результату*	*Самостійна робота*	*захист ЛР*	10%
РН4.2	*Відповідально ставитися до виконуваних робіт, нести відповідальність за їх якість*	*Лабораторна робота*	*захист ЛР*	10%

6. Співвідношення результатів навчання дисципліни із програмними результатами навчання

Результати навчання дисципліни / **Програмні результати навчання**	**РН 1.1**	**РН 1.2**	**РН 1.3**	**РН 2.1**	**РН 2.2**	**РН 2.3**	**РН 3.1**	**РН 4.1**	**РН 4.2**
(з опису освітньої програми)									
ПР08. Володіти сучасними методами розробки програм і програмних комплексів та прийняття оптимальних рішень щодо складу програмного забезпечення, алгоритмів процедур і операцій.	+	+	+	+		+			
ПР09. Вміти створювати ефективні алгоритми для обчислювальних задач системного аналізу та систем підтримки прийняття рішень.		+		+	+			+	+
ПР13. Проектувати, реалізовувати, тестувати, впроваджувати, супроводжувати, експлуатувати програмні засоби роботи з даними і знаннями в комп'ютерних системах і мережах.				+	+	+	+	+	+
ПРСАПР 3. Вміти проектувати, реалізовувати, тестувати, впроваджувати, супроводжувати та експлуатувати програмне забезпечення комп'ютерних систем і мереж обробки даних і знань				+	+	+	+	+	+

7. Схема формування оцінки.

7.1 Форми оцінювання студентів:

- семестрове оцінювання:

Перший семестр

1. Контрольна робота 1: РН 1.1, РН 2.1 — 20 балів/10 балів.
2. Контрольна робота 2: РН 1.1, РН 1.3, РН 2.1 — 20 балів/10 балів.
3. Лабораторна робота 1 (проект): РН2.1, РН3.1, РН4.1, РН4.2 – 20 балів/10 балів.
4. Лабораторна робота 2 (проект): РН2.1, РН2.3, РН3.1, РН4.1, РН4.2 – 20 балів/10 балів.
5. Лабораторна робота 3 (проект): РН1.3, РН2.1, РН2.3, РН3.1, РН4.1, РН4.2 – 20 балів/10 балів.

Другий семестр

1. Контрольна робота 1: РН 1.2, РН 1.3, РН 2.1 — 15 балів/9 балів.

*2. Лабораторна робота 1 (проект): РН2.1, РН2.2, РН2.3, РН3.1, РН4.1, РН4.2 – 15 балів/*9 *балів.*
*3. Лабораторна робота 1 (проект): РН2.1, РН2.2, РН2.3, РН3.1, РН4.1, РН4.2 – 15 балів/*9 *балів.*
*4. Лабораторна робота 1 (проект): РН2.1, РН2.2, РН2.3, РН3.1, РН4.1, РН4.2 – 15 балів/*9 *балів.*

- підсумкове оцінювання

***Перший семестр* (у формі заліку)**

Згідно пп. 4.6.1 та 7.1.5 «Положення про організацію освітнього процесу у Київському національному університеті імені Тараса Шевченка» залік виставляється на підставі поточного контролю (див. семестрове оцінювання) як сума оцінок/балів за всіма успішно оціненими результатами навчання; оцінки нижче від мінімального порогового рівня до підсумкової оцінки не додаються.
До заліку допускаються всі студенти.

***Другий семестр* (у формі іспиту)**

- максимальна кількість балів які можуть бути отримані студентом: 40 балів;
- результати навчання які будуть оцінюватись: РН1.1, РН1.2, РН1.3, РН2.1, РН2.3;
- форма проведення і види завдань: письмова із захистом відповіді, два теоретичних питання (5 та 10 балів) та два практичних завдання (по 10 балів), і ще 5 балів додаються при наявності комітів практичних завдань з інтервалом до 10 хвилин в систему контролю версій.

Студент допускається до іспиту, якщо він під час семестру набрав більше 36 балів, причому лабораторні роботи було виконано як мінімум на 60%.

Для отримання загальної позитивної оцінки з дисципліни оцінка за іспит не може бути меншою 24 балів.

Структура екзаменаційної роботи та критерії оцінювання:
2 теоретичних питання (по 8 балів), 2 задачі (по 12 балів)

Запитання для підготовки до іспиту

- Парадигма функціонального програмування. Мова *Haskell*. Типи даних (елементарні та складені).
- Мова *Haskell*. Функції. Типи функцій, каррінг.
- Поліморфні типи та поліморфні функції. Параметричний поліморфізм. Виведення типів.
- Визначення функцій рівняннями. Зіставлення зі зразком. Приклади.
- Визначення функцій рівняннями. Рекурсивні функції. Приклади.
- Мова *Haskell*. Інфіксні оператори та бінарні функції. Операторна форма бінарних функцій. Функціональна форма інфіксних операторів .
- Мова *Haskell*. Умовні конструкції при визначенні функцій. Використання двовимірної структуризації.
- Каррінгові та кортежні функції. Функції як аргументи. Функції вищих порядків *curry* та *uncurry*.
- Мова *Haskell*. Типи користувача. Конструктори. Приклади.
- Рекурсивні типи користувача. “Власний” списковий тип.
- Рекурсивні типи користувача. Тип *Tree*.
- Ліниві та енергійні обчислення. Приклади.
- Використання нескінчених списків. Списки арифметичних послідовностей.

- Ліниві та енергійні обчислення. Використання нескінчених списків на прикладі списку простих чисел.
- Ліниві та енергійні обчислення. Використання нескінчених списків на прикладі списку чисел Фібоначчі.
- *List comprehension.* Приклади. Функція швидкого сортування.
- Функції вищих порядків. Приклади (*map, filter, zip, zipWith*).
- Функції вищих порядків. Приклади (*foldl, foldl*1 *foldr, foldr*1).
- Мова *Haskell*. Класи типів. Клас типів рівності (*Eq*). Розширення класів (*class extension*). Клас типів *Ord*. Варіанти втілення класів *Eq* та *Ord* у типі *Tree*.
- Мова *Haskell*. Похідні втілення класів типів. Похідні втілення *Eq* та *Ord* у типі *Tree*.
- Мова *Haskell*. Клас типів *Show*. Варіанти втілення класу *Show* у типі *Tree*.
- Суперпозиція функцій та композиція (.) у мові *Haskell.*
- Композиції та монадні композиції. Оператор монадної композиції (>=>).
- Оператори аплікації. Оператор зв'язування *bind* (>>=).
- Клас типів *Monad.* Монадна композиція та приємна (прозора) версія монадних законів. Дві версії монадних законів.
- Інтеграція монадних функцій зі «звичайними» функціями. Монадна функція *return.*
- Не всюди визначені функції. Тип *Maybe.* Монада *Maybe* (монадне втілення *Maybe*).
- Компонування *Maybe* монадних функцій. Приклади. Монада *Maybe* та монадні закони.
- Інтеграція *Maybe* монадних функцій зі «звичайними» функціями. Приклади.
- *Do*-нотація для монадних функцій. Перетворення *do*-нотації до виразу з монадною композицією. Приклади.
- Ввід-вивід. *IO* монада. Деякі функції вводу-виводу. Приклади.
- Оператор монадної послідовності (>>). Приклади використання у випадку *IO* монади.
- Ввід-вивід. Монадна функція *return.* Приклади.
-
- Скулемівські нормальні форми, множини диз'юнктів. Метод резолюцій для логіки висловлювань.
- Уніфікація. Алгоритм уніфікації. Метод резолюцій для логіки предикатів.
- Чистий (недетермінований) Пролог. Синтаксис. Логічна (декларативна) семантика.
- Чистий (недетермінований) Пролог. Операційна семантика. Приклади.
- Пролог. Списковий тип. Приклади.
- Приклади використання Прологу (задача про 8 ферзів, родинні зв'язки).
- Приклади використання Прологу (задача Ейнштейна).
- Пролог-процесори. Огляд особливостей (стратегія обчислень, відкати).
- Пролог-процесори. Техніка програмування (управління відкатами, моделювання циклів, "повтори" на основі предиката *repeat*, повтори з лічильником).
- Моделювання стеків та черг.
- Експертні системи та стратегії пошуку відповіді. Експертні системи із прямою стратегією.
- Експертні системи з оберненою стратегією. Подання знань у вигляді правил Прологу.
- Подання знань в експертних системах. Мережі знань.
- Експертні системи з поясненнями. Питання "Навіщо?" ("*Why*?").
- Експертні системи з поясненнями. Питання "Як?" ("*How*?").
- Декларативні засоби Java-програмування на прикладі веб-служб.
- Декларативні засоби *.Net* програмування на прикладі веб-служб.

- Веб-служби (*Web Services*) та сервісно-орієнтована архітектура (СОА). Стандарти веб-служб. Протокол *SOAP*. Конверт *SOAP*-повідомлення.
- Документування веб-служб: генерація документації для сприйняття людиною (з використанням веб-браузерів), генерація документації, орієнтованої на використання програмами – *wsdl*-файли.
- Розробка *.Net* веб-служб. Тест-форми веб-служб.
- Розробка клієнтських *.Net* програм для веб-служб. Утиліта *Wsdl.exe*.
- Розробка веб-служб на *Java*.
- Розробка клієнтських *Java* -програм для веб-служб.

7.2. Організація оцінювання.

Терміни проведення форм оцінювання в 3 семестрі:

1. Контрольна робота 1: до 6 тижня семестру.
2. Контрольна робота 2: до 14 тижня семестру.
3. Лабораторна робота 1 (проект): до 3 тижня семестру.
4. Лабораторна робота 2 (проект): до 8 тижня семестру.
5. Лабораторна робота 3 (проект): до останнього тижня семестру.

Терміни проведення форм оцінювання в 4 семестрі:

1. Контрольна робота 1: до 10 тижня семестру.
2. Лабораторна робота 1 (проект): до 4 тижня семестру.
3. Лабораторна робота 2 (проект): до 9 тижня семестру.
4. Лабораторна робота 3 (проект): до останнього тижня семестру.

Для контрольної роботи дозволяється одне перескладання, але без поважних причин (як то хвороба) можна отримати не більше 80% балів.

Лабораторні роботи можна здавати після визначеного терміну, але за перший тиждень запізнення віднімається 20% балів, і за кожен наступний по 10% (після 4-го тижня бали далі не віднімаються, на 60% можна здати завжди).

7.3 Шкала відповідності оцінок

Відмінно / Excellent	90-100
Добре / Good	75-89
Задовільно / Satisfactory	60-74
Незадовільно / Fail	0-59
Зараховано / Passed	60-100
Не зараховано / Fail	0-59

8. Структура навчальної дисципліни. Тематичний план лекцій і лабораторних занять

СЕМЕСТР 3

№ лекції	Назва лекції	Кількість годин		
		Лекції	Лаборат. занять	Сам. р-та
	Частина 1. Парадигма функціонального програмування.			
1.	**Тема 1.** Основні концепції функціональної парадигми. Мова Haskell. Елементарні типи даних. Вбудовані складені типи (списки, кортежі, функції).	4	2	2

2.	**Тема 2.** Функції. Типи функцій, визначення функцій рівняннями, застосування функцій. Автоматичне виведення типів. Функції від кількох аргументів. Карінг.	4	2	4
3.	**Тема 3.** Визначення функцій кількома рівняннями. Рекурсивні функції та теорема про нерухому точку.	4	2	4
4.	**Тема 4.** Поліморфні типи та поліморфні функції. Параметричний поліморфізм.	4	2	6
5.	**Тема 5.** Лямбда-функції у Haskell. Функції вищих порядків.	2	2	4
	Контрольна робота 1	2		
6.	**Тема 6.** Ліниві та енергійні обчислення. Нескінчені списки. Списки арифметичних послідовностей.	2	2	6
7.	**Тема 7.** List comprehension. Приклади.	2	4	6
8.	**Тема 8.** Поняття класів у Haskell та поліморфізм класів.	2	2	4
9.	**Тема 9.** Проблеми вводу-виводу,поняття монадних обчислень та монад.	4	4	8
10.	**Тема 10.** Стандартні монади. Проектування монад. Монада Maybe.	2	4	8
	Контрольна робота 2	2		
	ВСЬОГО	34	28	56

Загальний обсяг *120* *год.,* в тому числі:
Лекцій *– **34** год.*
Лабораторні заняття - ***28*** *год.*
Консультації – ***2*** *год.*
Самостійна робота **- 56** *год.*

СЕМЕСТР 4

№ лекції	Назва лекції	Кількість годин		
		Лекції	Лаборат. занять	Сам. р-та
Частина 2. Парадигма логічного програмування				
11.	**Тема 11.** "Чистий" (недетермінований) Пролог. Синтаксис, логічна семантика.	2	2	4
12.	**Тема 12.** Операційна семантика Пролог-програм та метод резолюцій.	2	2	6
13.	**Тема 13.** Процесори для мови Пролог. Використання відкатів.	2	2	4
14.	**Тема 14.** Специфіка техніки програмування у Пролозі.	1	2	8
	Тема 15. Використання Прологу при реалізації експертних систем.	1		
15.	**Тема 16.** Експертні системи з поясненнями. Експертні системи в умовах неповної визначеності.	2	2	6
Частина 3. Парадигма об'єктно-орієнтованого програмування та декларативні засоби програмування				
16.	**Тема 17.** Об'єкти та метадані. Рефлексія об'єктів. Декларативний стиль програмування у Java. Анотації.	2	2	6

17.	**Тема 18.** Анотації, розробка анотацій. Приклад.	4	4	8
18.	**Тема 19.** Застосування декларативних засобів програмування у Java та .Net на прикладі технології веб-сервісів SOAP	2	4	8
	Контрольна робота 1	2		
	ВСЬОГО	20	20	50

Загальний обсяг ***90*** *год.,* в тому числі:
Лекцій ***– 20*** *год.*
Лабораторні заняття - ***20*** *год.*
Самостійна робота **- 50** *год.*

9. Рекомендовані джерела

Основні:

1. Липовача М. Изучай Haskell во имя добра! М.: ДМК Пресс, 2012.
2. Lipovača M. Learn You a Haskell for Great Good! Miran Lipovača. : No Starch Press», 2011.
3. Душкин Р. В. Функциональное программирование на языке Haskell. М.: ДМК Пресс, 2007.
4. Братко И. Программирование на языке PROLOG для искусственного интеллекта. М.: Мир, 1990.
5. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М.: Наука, 1983.
6. Марселлус Д. Программирование экспертных систем на Турбо-Прологе, М., Финансы и статистика, 1994. 410с.
7. Хохгуртль Б., C# и Java: межплатформенные Web-сервисы, М., Кудиц-образ, 2004, 410с.
8. Машнин Т.С. Web-сервисы Java, БХВ-Петербург, 2012, 560 с.

Додаткові:

1. Ин Ц., Соломон Д. Использование Турбо-Пролога. М.: Мир, 1990.
2. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог. М.: Мир, 1990.
3. Стобо Дж. Язык программирования Пролог. М.: Радио и связь, 1993.
4. Клоксин У., Меллиш К. Программирование на языке пролог. М.: Мир, 1987.
5. Янсон А. Турбо-Пролог в сжатом изложении. М.: Мир, 1991.
6. Душкин Р. В. Справочник по языку Haskell. М.: ДМК Пресс, 2008.
7. Душкин Р. В. Практика работы на языке Haskell. М.: ДМК Пресс, 2009.
8. Роганова Н. А. Функциональное программирование, 2002.
9. Филд А., Харрисон П. Функциональное программирование. М.: Мир, 1993.
10. Хендерсон П. Функциональное программирование. Применение и реализация. М.: Мир, 1983.
11. Хьюдак П., Петерсон Дж., Джозеф Фасел Дж. Мягкое введение в Haskell. Учебник.