УДК 512.58:004.42:378.147

https://doi.org/10.52058/2786-5274-2026-5(57)-2035-2046

Матурін Юрій Петрович кандидат фізико-математичних наук, доцент, Дрогобицький державний педагогічний університет імені Івана Франка, м. Дрогобич, https://orcid.org/0000-0002-0544-1329

Хаць Руслан Васильович кандидат фізико-математичних наук, доцент, Дрогобицький державний педагогічний університет імені Івана Франка, Львівська область, м. Дрогобич, https://orcid.org/0000-0001-9905-5447

Комарницька Леся Іванівна кандидат фізико-математичних наук, доцент, Дрогобицький державний педагогічний університет імені Івана Франка, Львівська область, м. Дрогобич, https://orcid.org/0009-0001-0907-1038

МЕТОДИЧНІ ПІДХОДИ У ВИВЧЕННІ ТЕОРІЇ КАТЕГОРІЙ

Анотація. У статті обґрунтовано методичні підходи до вивчення теорії категорій у контексті її реальних застосувань у комп'ютерних науках у межах магістерської підготовки здобувачів вищої освіти за предметною спеціальністю А4.04 «Середня освіта (Математика)». Теорія категорій розглядається не як абстрактна галузь алгебри, відірвана від практики, а як уніфікована математична мова, що точно описує структурні закономірності сучасних комп'ютерних систем, архітектурні принципи функціональних мов програмування та формальні основи теорії баз даних.

Показано, що основні поняття теорії категорій (категорія, функтор, природне перетворення, ад'юнкція, монада) мають точні та практично значущі аналоги в таких галузях комп'ютерних наук, як теорія типів і функціональне програмування, семантика мов програмування, формальна верифікація програм і теорія баз даних. Зокрема, система типів мови програмування Haskell є безпосередньою реалізацією категоріальних конструкцій: типи утворюють об'єкти категорії, чисті функції − морфізми, параметричні типи-контейнери − ендофунктори, а монади забезпечують структурований опис обчислювальних ефектів як специфічних моноїдів у категорії ендофункторів.

Детально розглянуто відповідність Каррі-Говарда-Ламбека, яка встановлює глибокий тристоронній ізоморфізм між інтуїціоністською пропозиційною логікою, системою простого лямбда-числення і декартово замкненими категоріями. Кожна логічна формула відповідає типу, кожне доведення − програмі, а структурні правила логічних висновків − категоріальним морфізмам. Цей ізоморфізм є теоретичною основою систем автоматизованого доведення теорем, мов із залежними типами та сучасних засобів статичної верифікації

програм. Для майбутнього вчителя математики таке поєднання алгебри, логіки і програмування формує єдину концептуальну картину, важливу для міждисциплінарного навчання.

Окрему увагу приділено категоріальному підходу до теорії баз даних, де схема реляційної бази даних моделюється як невелика категорія, а конкретна база даних − як функтор до категорії множин. Такий підхід, розвинений у працях Д. Співака, дозволяє описувати міграцію даних між різними схемами через функтори та ад'юнкції, перетворюючи завдання трансляції даних на точно сформульоване завдання категоріального відображення. Розглянуто також застосування F-алгебр для опису рекурсивних структур даних та алгоритм catamorphism (згортки) як канонічного морфізму до фінальної коалгебри.

Запропоновано п'ятиетапну послідовну методичну модель вивчення теорії категорій, що будується від базових понять категорії і функтора через природні перетворення і ад'юнкції до монад і декартово замкнених категорій. На кожному етапі теоретичні поняття закріплюються відповідними прикладами з функціонального програмування, теорії типів і теорії баз даних. Практична цінність дослідження полягає у формуванні в магістрів здатності пояснювати архітектурні рішення сучасного програмного забезпечення через математичну структуру. Запропонований підхід може бути використаний у курсах абстрактної алгебри, теорії програмування, математичної логіки та методики навчання математики.

Ключові слова: теорія категорій, функтор, природне перетворення, монада, типи даних, функціональне програмування, відповідність Каррі-Говарда-Ламбека, декартово замкнена категорія, категоріальна семантика баз даних, методика навчання математики.

Yuriy Maturin Candidate of Sciences in Physics and Mathematics, Docent, Drohobych Ivan Franko State Pedagogical University, Drohobych, https://orcid.org/0000-0002-0544-1329

Ruslan Khats' Candidate of Sciences in Physics and Mathematics, Docent, Drohobych Ivan Franko State Pedagogical University, Drohobych, https://orcid.org/0000-0001-9905-5447

Lesia Komarnytska Candidate of Sciences in Physics and Mathematics, Docent, Drohobych Ivan Franko State Pedagogical University, Drohobych, https://orcid.org/0009-0001-0907-1038

METHODOLOGICAL APPROACHES IN STUDYING CATEGORY THEORY

Abstract. The paper substantiates methodological approaches to the study of category theory in the context of its real applications in computer science within the

framework of master's training of students in the subject specialty A4.04 «Secondary Education (Mathematics)». Category theory is treated not as an abstract branch of algebra divorced from practice, but as a unified mathematical language that precisely describes the structural regularities of modern computer systems, the architectural principles of functional programming languages, and the formal foundations of database theory.

It is shown that the basic notions of category theory (category, functor, natural transformation, adjunction, and monad) have precise and practically significant counterparts in such areas of computer science as type theory and functional programming, programming language semantics, formal program verification, and database theory. In particular, the type system of the Haskell programming language is a direct implementation of categorical constructions: types form the objects of a category, pure functions form morphisms, parametric container types form endofunctors, and monads provide a structured description of computational effects as specific monoids in the category of endofunctors.

The Curry-Howard-Lambek correspondence is examined in detail. It establishes a deep three-way isomorphism between intuitionistic propositional logic, the system of simple lambda calculus, and Cartesian closed categories. Every logical formula corresponds to a type, every proof to a program, and the structural rules of logical inference to categorical morphisms. This isomorphism constitutes the theoretical foundation of automated theorem provers, languages with dependent types, and modern static program verification tools. For a future mathematics teacher, such a unification of algebra, logic, and programming forms a single conceptual picture that is fundamentally important for interdisciplinary instruction.

Special attention is given to the categorical approach to database theory, in which a relational database schema is modelled as a small category and a specific database instance as a functor to the category of sets.

A five-stage sequential methodological model for studying category theory is proposed. It is organized as a progression from the basic notions of category and functor through natural transformations and adjunctions to monads and Cartesian closed categories. At each stage, theoretical concepts are consolidated by corresponding examples from functional programming, type theory, and database theory. The practical value of the study lies in developing in master's students the ability to explain the architectural decisions of modern software through mathematical structure. The proposed approach may be used in courses on abstract algebra, programming theory, mathematical logic, and mathematics teaching methodology.

Keywords: category theory, functor, natural transformation, monad, data types, functional programming, Curry-Howard-Lambek correspondence, Cartesian closed category, categorical database semantics, mathematics teaching methodology.

Постановка проблеми. Сучасний учитель математики в умовах цифровізації освіти дедалі частіше стикається з необхідністю пояснювати

принципи роботи програмних систем і встановлювати зв’язки між математичними структурами та архітектурними рішеннями сучасного програмного забезпечення. У традиційному університетському курсі алгебри теорія категорій, якщо і включається до навчальних програм, подається переважно як засіб уніфікації алгебраїчних структур − груп, кілець, модулів. Зв’язок між цією теорією та комп’ютерними науками залишається поза навчальним контентом.

Водночас у міжнародній комп’ютерно-науковій спільноті теорія категорій із 1980-х років слугує активним дослідницьким інструментом. Денотаційна семантика мов програмування, теорія типів, функціональне програмування, алгоритми компіляції, теорія паралельних обчислень і формальні методи верифікації програм − усі ці галузі використовують категоріальні конструкції як рідну мову. Система типів мов Haskell і Agda є прямою практичною реалізацією понять функтора, монади та природного перетворення. Бібліотеки функціонального програмування явно посилаються на категоріальні закони. Стандарт SQL у версії, розробленій Е.Ф. Коддом, містить приховані категоріальні структури, які стають явними в підході Д. Співака.

Виникає дидактична прогалина: майбутній учитель математики вивчає теорію категорій як абстрактну алгебраїчну структуру і не отримує уявлення про те, що ці самі поняття складають теоретичний фундамент систем, якими він щодня користується. Це знижує мотивацію до вивчення абстрактного матеріалу та позбавляє здобувача освіти можливості формувати цілісний міждисциплінарний погляд. Для предметної спеціальності А4.04 «Середня освіта (Математика)» ця проблема є особливо актуальною, оскільки майбутні педагоги повинні вміти встановлювати зв’язки між різними предметними областями і забезпечувати математично строге пояснення принципів роботи обчислювальних систем.

Аналіз останніх досліджень і публікацій. Теорія категорій як самостійна математична дисципліна була заснована С. Ейленбергом і С. МакЛейном [1], де введено поняття категорії, функтора та природного перетворення для формалізації поняття «природної» еквівалентності в алгебраїчній топології. Ф.В. Ловер і С. Шануель [2] розвинули доступну та концептуально сильну версію теорії категорій для широкої математичної аудиторії, заклавши основи категоріальної логіки і теорії топосів як середовищ конструктивної математики.

Систематичний виклад теорії категорій для потреб комп’ютерних наук здійснили М. Барр і Ч. Велс [3], що охоплює застосування до теорії автоматів, специфікації програм та алгебраїчної семантики. Б. Пірс [4] сформулював мінімальний набір категоріальних понять, достатній для розуміння функціональних мов програмування та теорії типів. С. Аводі [5] побудував систематичний вступ до теорії категорій, незалежний від конкретної алгебраїчної галузі.

Застосування монад у функціональному програмуванні формалізував П. Уодлер [7], який показав, що монади забезпечують уніфікований механізм моделювання обчислювальних ефектів: введення-виведення, виняткових

ситуацій, стану та недетермінізму. Б. Мілевський [6] здійснив спробу доступного пояснення категоріальних понять через реалізації на мові Haskell.

Категоріальний підхід до теорії баз даних розроблений у праці Д. Співака [9], де функтори служать формальним моделюванням схем та екземплярів даних. Загальний маніфест застосування теорії категорій у комп'ютерних науках сформульований Дж. Ґоґуеном [10], де підкреслено єдність понять структури, відображення та композиції в цій дисципліні.

Разом із тим, методичний аспект включення теорії категорій до підготовки майбутніх учителів математики з акцентом на реальних застосуваннях у комп'ютерних науках залишається нерозробленим у вітчизняній педагогічній літературі. Відсутні методичні моделі, здатні органічно поєднати академічну строгість теорії категорій із практичними прикладами з функціонального програмування, теорії типів і теорії баз даних.

Мета статті − обґрунтувати методичні підходи до вивчення теорії категорій у магістерських програмах підготовки вчителів математики через призму її реальних застосувань у комп'ютерних науках. Запропонувати послідовну дидактичну модель, що дозволяє перейти від абстрактних категоріальних понять до конкретних архітектурних принципів сучасних програмних систем і мов програмування, зберігаючи при цьому вимогу математичної строгості.

Виклад основного матеріалу. Теорія категорій виникла як мова уніфікації математики, але стала водночас мовою опису структур комп'ютерних наук. Причина цього збігу полягає у тому, що і математика, і програмування оперують композицією відображень. Функції між множинами, логічні висновки, морфізми модулів, переходи між станами автомата − усе це є конкретизаціями єдиного поняття «морфізм у категорії». Категоріальне мислення дозволяє виявляти структурну аналогію між математичними і програмними конструкціями там, де поверхневий синтаксичний аналіз не бачить зв'язку.

Для педагогічного введення важливо підкреслити, що теорія категорій не потребує апарату аналізу чи геометрії. Вона вимагає лише розуміння структур із морфізмами та їх композиціями. Це робить її органічним компонентом алгебраїчного курсу та природним мостом між алгеброю і програмуванням. У широкому розумінні категорія є «математикою математики»: вона вивчає не окремі об'єкти, а відображення між ними і закони їх поєднання.

Теорія категорій як структурна мова комп'ютерних наук. Категорія C складається з класу об'єктів Ob(C) та для кожної пари об'єктів A, B − з множини морфізмів Hom(A, B) разом з асоціативною операцією композиції та тотожним морфізмом idA для кожного об'єкта A [5]. У програмному контексті найпростішим прикладом є категорія Hask, де об'єктами є типи мови Haskell, а морфізмами − чисті функції між цими типами. Тотожний морфізм відповідає функції id, а композиція − оператору (.) мови Haskell. Закони категорії відповідають фундаментальним законам рефакторингу: асоціативність означає, що дужки при ланцюжку застосувань функцій не мають значення, а наявність

тотожного морфізма гарантує нейтральний елемент для будь-якого зв’язування функцій [4].

Другим важливим прикладом є категорія частково впорядкованих множин Pos, де морфізмами є монотонні відображення. Ця категорія безпосередньо пов’язана з типами уточнення (refinement types) і теорією Ґалуа − апаратом, що лежить в основі статичного аналізу програм. Ад’юнктивні пари монотонних відображень між решітками моделюють пари «абстракція-конкретизація» в теорії абстрактної інтерпретації, що є фундаментом для інструментів верифікації програм [3]. Ключовою методичною ідеєю на цьому етапі є демонстрація того, що одні й ті самі категоріальні закони виконуються у принципово різних предметних галузях, що підкреслює структурну єдність математики і комп’ютерних наук [6, 10].

Функтори та природні перетворення в теорії типів. Функтор F: C → D − це відображення між категоріями, що зберігає структуру: воно переводить об’єкти у об’єкти, морфізми − у морфізми і зберігає тотожні морфізми та композицію. У категорії типів Hask ендофунктор − це параметричний тип-контейнер [4, 6]. Типи Maybe, [], Tree, Either e − усі вони є функторами: для кожного з них визначена операція fmap, яка «піднімає» звичайну функцію a → b до функції F a → F b, перетворюючи значення всередині контейнера без зміни самої контейнерної структури.

Закони функтора мають безпосередній педагогічний зміст. Перший закон (fmap id = id) означає збереження тотожності: якщо підняти тотожну функцію, вона не повинна змінювати контейнер. Другий закон (fmap (g ∘ f) = fmap g ∘ fmap f) означає збереження композиції: підняття двох послідовних функцій еквівалентне послідовному підняттю кожної. Ці закони є специфікацією коректної поведінки «контейнера», яку може автоматично перевіряти бібліотека QuickCheck. Для здобувача освіти такий зв'язок між абстрактним математичним законом і автоматизованим тестуванням є прикладом того, як алгебраїчна структура безпосередньо забезпечує якість програмного забезпечення [7].

Природне перетворення α: F ⟹ G між функторами F і G − це сім’я морфізмів αA: F(A) → G(A), параметризована об’єктами A і задовольняє умову натуральності: для кожного морфізму f: A → B виконується рівність G(f) ∘ αA = αB ∘ F(f). У програмному контексті природне перетворення − це поліморфна функція між контейнерними типами. Наприклад, функція safeHead :: [a] → Maybe a є природним перетворенням з функтора [] до функтора Maybe. Умова натуральності означає, що результат не залежить від порядку застосування: чи ми спочатку перетворимо всі елементи, а потім візьмемо перший, чи спочатку візьмемо перший, а потім перетворимо − результат буде той самий [4, 5].

Цей факт, відомий як «теорема про безкоштовні теореми» П. Уодлера, має глибоке методичне значення. Поліморфні функції в Haskell автоматично задовольняють певні рівності, виведені виключно з їхнього типу, без жодного аналізу їх тіла. Компілятор може використовувати ці рівності для оптимізацій,

наприклад, для підняття map через fold або для злиття суміжних перетворень. Такий зв'язок між типами, теоремами і оптимізаціями компілятора є одним із найяскравіших прикладів того, як математична структура безпосередньо впливає на архітектуру програмних систем.

Монади як категоріальна теорія обчислювальних ефектів. Монада − це ендофунктор T: C → C разом із двома природними перетвореннями: одиницею η: IdC ⟹ T та множенням μ: T∘T ⟹ T, що задовольняють аксіомам моноїда: μ ∘ (T ∘ η) = id = μ ∘ (η ∘ T) та μ ∘ (T ∘ μ) = μ ∘ (μ ∘ T) [3, 5]. Цей абстрактний математичний об'єкт набуває надзвичайно конкретного змісту у функціональному програмуванні.

П. Уодлер [7] показав, що монади забезпечують уніфікований механізм для роботи з обчислювальними ефектами в чистих функціональних мовах. Монада Maybe моделює обчислення, які можуть завершитися невдачею. Монада List моделює недетерміновані обчислення − обчислення, що повертають нуль або більше результатів. Монада State s моделює обчислення з глобальним станом, який змінюється. Монада IO інкапсулює взаємодію з зовнішнім світом, зберігаючи чистоту функціональної семантики мови. У всіх цих різноманітних випадках структура монади залишається тією самою − відрізняються лише конкретний функтор T і конкретна реалізація η та μ.

Методично важливо пояснити вислів «монада − це моноїд у категорії ендофункторів», який часто цитують як приклад математичного гумору. Насправді це строге твердження: категорія ендофункторів [C, C] із операцією горизонтальної композиції функторів є моноїдальною категорією, а монада − рівно моноїдним об'єктом у ній. Одиниця монади η відповідає одиничному елементу моноїда, а множення μ відповідає бінарній операції. Таке пояснення демонструє, як єдине абстрактне поняття «моноїд» конкретизується у принципово несхожих структурах − від натуральних чисел зі звичайним множенням до структур керування потоком у програмуванні [5].

F-алгебри є ще одним важливим застосуванням теорії категорій у програмуванні. F-алгебра для ендофунктора F − це пара (A, α: FA → A), де A є носієм, а α − структурним відображенням. Рекурсивні типи даних (списки, дерева) є початковими F-алгебрами.

Відповідність Каррі-Говарда-Ламбека та декартово замкнені категорії. Відповідність Каррі-Говарда встановлює ізоморфізм між системою природного виводу для інтуїціоністської пропозиційної логіки та системою типів простого лямбда-числення: формули відповідають типам, доведення − термам (програмам), нормалізація доведень − обчисленню за значенням. Категоріальна частина цього тристороннього ізоморфізму, додана Ламбеком, стверджує, що обидві ці системи еквівалентні теорії декартово замкнених категорій [3, 5].

Декартово замкнена категорія (CCC) − це категорія з фінітними добутками та «внутрішнім Hom»: для кожної пари об'єктів A, B існує об'єкт [A, B] (тип функцій), що задовольняє природну бієкцію Hom(C × A, B) ≅ Hom(C, [A, B]). Ця

бієкція є рівно каррюванням (currying) у функціональних мовах: функція двох аргументів еквівалентна функції, яка повертає функцію. У категорії Set декартово замкнена структура − це звичайний простір функцій BA. В категорії типів Haskell [A, B] є типом A -> B.

Таблиця 1

Відповідність Каррі-Говарда-Ламбека між логікою, теорією типів і теорією категорій

Логіка (пропозиції)	**Теорія типів (Haskell)**	**Теорія категорій**	**Необхідна алгебраїчна структура**
Істина ($\top$)	Одиниця ()	Термінальний об'єкт 1	Декартова категорія
Кон'юнкція $A \wedge B$	Тип добутку (A, B)	Добуток $A \times B$	Декартова категорія
Імплікація $A \supset B$	Тип функцій A -> B	Експоненціал B^A	Декартово замкнена категорія (CCC)
Хибність ($\bot$)	Тип Void	Початковий об'єкт 0	Бідекартово замкнена (BiCCC)
Диз'юнкція $A \vee B$	Тип суми Either A B	Копродукт $A + B$	Бідекартово замкнена (BiCCC)
Універсальна квантифікація $\forall x : A.B(x)$	Залежний добуток (Π-тип)	Правий спряжений до функтора заміни бази	Локально декартово замкнена категорія / Топос
Екзистенційна квантифікація $\exists x : A.B(x)$	Залежна сума (Σ-тип)	Лівий спряжений до функтора заміни бази	Локально декартово замкнена категорія / Топос

Ця відповідність має принципове значення для підготовки вчителя математики. Вона показує, що компілятор мови Haskell або Agda виконує таку саму перевірку, що й верифікатор математичних доведень. Тип-правильна програма є доведенням відповідної теореми. Коли IDE сигналізує про помилку типів, вона сигналізує про некоректне логічне міркування. Такий погляд перетворює процес програмування на форму математичної практики, що є важливим аргументом для майбутніх учителів математики [8, 10].

Категоріальна теорія баз даних. Д. Співак [9] розробив формалізм категоріальних баз даних, в якому схема реляційної бази даних моделюється як невелика категорія S. Таблиці відповідають об'єктам категорії, зовнішні ключі та атрибути − морфізмам, а обмеження цілісності (транзитивність зовнішніх ключів, рефлективність) − комутативним діаграмам у цій категорії. Конкретна база даних (набір таблиць з рядками) моделюється як функтор I: S → Set, де кожній таблиці ставиться у відповідність конкретна множина рядків, а кожному зовнішньому ключу − функція між відповідними множинами рядків.

Умова функторіальності означає рівно те, що зовнішні ключі є цілісними та коректними. Тотожний морфізм у схемі відповідає тотожній функції на рядках таблиці. Закон збереження композиції відповідає транзитивному замиканню зовнішніх ключів. Таким чином, математична коректність схеми бази даних − це рівно функторіальність відповідного відображення. Це є простим і точним формулюванням поняття «реляційної цілісності», яке зазвичай описується довгим переліком правил без єдиного структурного принципу [9].

Для навчання у педагогічному університеті ця тема є зручним місцем, де теорія категорій безпосередньо зустрічається з практикою баз даних − дисципліною, яку більшість здобувачів освіти вже вивчали. Вони розуміють зовнішні ключі та реляційну цілісність; тепер їм пропонується побачити, що ця знайома концепція є конкретним прикладом функторіального закону. Такий підхід формує компетентність «бачити структуру за синтаксисом», яка є ключовою для педагога математики.

Методична модель вивчення теорії категорій для магістрів математики. На основі аналізу структури предметної галузі та особливостей підготовки фахівців за предметною спеціальністю А4.04 «Середня освіта (Математика)» пропонується наступна п'ятиетапна послідовна методична модель.

Перший етап: категорії та морфізми. Вводяться базові визначення на трьох рівнях абстракції: конкретна математична категорія (Set, Grp, Top), категорія типів (Hask), категорія малої схеми бази даних. На цьому ж рівні розглядаються ізоморфізми, початкові та термінальні об'єкти, добутки та копродукти. Методично важливим є проведення паралелей між цими поняттями: початковий об'єкт у Set − порожня множина, у Hask − тип Void, у Grp − тривіальна група.

Спільний структурний принцип − визначення через універсальну властивість, який є першим прикладом категоріального мислення [4, 5].

Другий етап: функтори та природні перетворення. Параметричні типи-контейнери як ендофунктори. Закони функтора та їх зв'язок з автоматизованим тестуванням. Поліморфні функції як природні перетворення. Теорема Йонеди: будь-який об'єкт категорії цілком визначається функтором Hom(A, −), і кожне природне перетворення між репрезентаційними функторами відповідає елементу носія. Це один із найглибших результатів теорії категорій, що знаходить застосування у принципі «програмування через інтерфейс» в об'єктно-орієнтованих мовах [5, 6].

Третій етап: ад'юнкції. Ад'юнктивні пари між категоріями. Зв'язок між «забутим» функтором і вільними алгебраїчними структурами: список як вільний моноїд є результатом ад'юнкції між категорією Set і категорією моноїдів Mon. Ад'юнкції в теорії баз даних ($\sigma_- \dashv \Delta_- \dashv \pi_-$). Закони ад'юнкції та їх вираження через одиницю η та коодиницю ε. Зв'язок між ад'юнкціями і монадами: будь-яка ад'юнкція породжує монаду на лівій категорії [3].

Четвертий етап: монади. Монада як ендофунктор з одиницею та множенням. Приклади монад у функціональному програмуванні (Maybe, List, State, IO, Writer, Reader). Категорія Клейслі та її зв'язок із ланцюговими обчисленнями. F-алгебри. Еквівалентність монадичного і прикладного (applicative) функторного підходу. Теорема про монадичний трансформер − спосіб комбінування монад у складних програмах [7].

П'ятий етап: декартово замкнені категорії та відповідність Каррі-Говарда-Ламбека. Типи даних як пропозиції. Програми як доведення. Обчислення як нормалізація доведень. Системи залежних типів як предикатна логіка. Огляд систем Agda і Coq та їх застосування для формальної верифікації математичних тверджень і алгоритмів. Огляд напряму «HoTT» (гомотопна теорія типів) як приклад активного фронту дослідження [5, 8].

На кожному етапі рекомендується поєднувати чотири компоненти: строге математичне визначення, конкретний приклад із базової математики, відповідний приклад на мові Haskell або псевдокоді та вправи на розпізнавання категоріальних структур у задачах з різних предметних областей. Методично важливим є принцип «вертикальної когерентності»: кожне нове поняття вводиться через три узгоджені рівні — математичний, програмний і педагогічний. Математичний рівень формулює визначення строго, у стилі стандартного університетського тексту. Програмний рівень демонструє конкретну реалізацію у функціональній мові. Педагогічний рівень відповідає на питання: як пояснити це поняття учням середньої школи або студентам початкових курсів?

Така побудова курсу розвиває у здобувачів освіти здатність до «переключення регістрів» − уміння вільно переходити між рівнями абстракції, яке є ключовою компетентністю сучасного вчителя математики в умовах інтеграції предметів. Важливим педагогічним наслідком є формування у майбутніх учителів цілісного уявлення про математику та програмування як взаємодоповнювальні й взаємно збагачувальні дисципліни, що створює вагоме

підґрунтя для підвищення мотивації учнів до вивчення математики через демонстрацію її прикладного потенціалу в умовах розвитку цифрових технологій [6, 10].

Висновки. Теорія категорій є не лише абстрактним розділом алгебри, але й реальною математичною мовою, якою описуються архітектурні рішення сучасних програмних систем. У рамках магістерської підготовки за предметною спеціальністю А4.04 «Середня освіта (Математика)» ця теорія займає природне місце на перетині алгебри та комп'ютерних наук, відповідаючи вимогам інтегрованого викладання математики та інформатики.

Вивчення теорії категорій виховує у майбутнього вчителя глибоке структурне математичне мислення. Вчитель, який розуміє алгебраїчні структури на мета-рівні функторів, ізоморфізмів та ад'юнкцій, отримує здатність викладати елементарну алгебру та геометрію набагато концептуальніше. Він бачить загальні закономірності та універсальні властивості (universal properties) там, де учень або менш кваліфікований вчитель бачить лише набір розрізнених, ізольованих формул.

Запропонована п'ятиетапна методична модель − від категорій і функторів через природні перетворення і ад'юнкції до монад і декартово замкнених категорій − дозволяє органічно поєднати строгість математичного апарату з практичними прикладами з функціонального програмування, теорії типів і теорії баз даних. Відповідність Каррі-Говарда-Ламбека, монади як інструмент опису обчислювальних ефектів і функторіальний підхід Співака до баз даних є трьома ключовими точками, де категоріальна математика безпосередньо перетинається з інженерною практикою.

Навчання майбутніх учителів математики з опорою на ці точки перетину формує у них здатність пояснювати принципи сучасних обчислювальних систем через математичну структуру. Запропонований підхід може бути реалізований у курсах абстрактної алгебри, теорії програмування, математичної логіки та методики навчання математики, і створює основу для подальших досліджень методики вивчення формальної верифікації, теорії типів і функціональних парадигм програмування у педагогічних університетах.

Література:

1. Eilenberg S., MacLane S. General Theory of Natural Equivalences. Transactions of the American Mathematical Society. 1945. Vol. 58. P. 231–294. https://doi.org/10.2307/1990284

2. Lawvere F. W., Schanuel S. H. Conceptual Mathematics: A First Introduction to Categories. Cambridge University Press, 2009. 390 p.

3. Barr M., Wells C. Category Theory for Computing Science. Prentice Hall, 1995. 432 p.

4. Pierce B. C. Basic Category Theory for Computer Scientists. MIT Press, 1991. 100 p.

5. Awodey S. Category Theory. 2nd ed. Oxford University Press, 2010. 311 p.

6. Milewski B. Category Theory for Programmers. Blurb, 2018. 498 p.

7. Wadler P. Monads for Functional Programming. In: Broy M. (ed.) Program Design Calculi. Springer, 1993. P. 233–264. https://doi.org/10.1007/978-3-662-02880-3_8

8. Meijer E., Fokkinga M., Paterson R. Functional Programming with Bananas, Lenses, Envelopes and Barbed Wire. In: Hughes J. (ed.) Functional Programming Languages and Computer Architecture. Springer, 1991. P. 124–144.

9. Spivak D. I. Category Theory for the Sciences. MIT Press, 2014.

10. Goguen J. A. A Categorical Manifesto. Mathematical Structures in Computer Science. 1991. Vol. 1, No. 1. P. 49–67. https://doi.org/10.1017/S0960129500000050

References:

1. Eilenberg, S., & MacLane, S. (1945). General theory of natural equivalences. Transactions of the American Mathematical Society, 58, 231–294. https://doi.org/10.2307/1990284

2. Lawvere, F. W., & Schanuel, S. H. (2009). Conceptual mathematics: A first introduction to categories (2nd ed.). Cambridge University Press.

3. Barr, M., & Wells, C. (1995). Category theory for computing science. Prentice Hall.

4. Pierce, B. C. (1991). Basic category theory for computer scientists. MIT Press.

5. Awodey, S. (2010). Category theory (2nd ed.). Oxford University Press.

6. Milewski, B. (2018). Category theory for programmers. Blurb.

7. Wadler, P. (1993). Monads for functional programming. In M. Broy (Ed.), Program design calculi (pp. 233–264). Springer. https://doi.org/10.1007/978-3-662-02880-3_8

8. Meijer, E., Fokkinga, M., & Paterson, R. (1991). Functional programming with bananas, lenses, envelopes and barbed wire. In J. Hughes (Ed.), Functional programming languages and computer architecture (pp. 124–144). Springer.

9. Spivak, D. I. (2014). Category theory for the sciences. MIT Press.

10. Goguen, J. A. (1991). A categorical manifesto. Mathematical Structures in Computer Science, 1(1), 49–67. https://doi.org/10.1017/S0960129500000050

Дата першого надходження статті до видання: 23.04.2026
Дата прийняття статті до друку після рецензування: 07.05.2026