https://doi.org/10.31891/2219-9365-2024-77-4
УДК: 004.438:514.1

СМИШ Олег
Інститут програмних систем НАН України
https://orcid.org/0000-0002-8074-9745
e-mail: o.smysh@ukma.edu.ua
ЗАГОРУЛЬКО Андрій
Національний університет «Києво-Могилянська академія»
e-mail: andrii.zahorulko@ukma.edu.ua

ВИКОРИСТАННЯ МОВИ LEAN ДЛЯ ПЕРЕВІРКИ НА НЕСУПЕРЕЧНІСТЬ МАТЕМАТИЧНИХ ЗАДАЧ У НАВЧАЛЬНІЙ РЕКОМЕНДАЦІЙНІЙ СИСТЕМІ

У статті розглянуто проблему забезпечення якості вивчення математики в умовах зростання популярності самонавчання. Запропоновано підхід на основі функційної мови Lean, що уможливлює автоматичну перевірку на точність та несуперечність даних математичної задачі та розв'язку цієї задачі. Цей підхід впроваджено в створювану рекомендаційну систему, що призначено для розв'язування математичних задач, з інтерфейсом для взаємодії та навчання кінцевих користувачів. Створена підсистема на основі мови Lean здатна виявляти потенційні помилки та суперечності в умовах задачі, а також сприяти коректному розв'язку задач.

Мову програмування Lean першочергово розроблено, щоб слугувати ефективним асистентом у доведенні теорем, через це цю мову й обрано.

Розроблено підсистему перевірки на Lean, де додано аксіоми та теореми з планіметрії, що дає змогу забезпечити інтеграцію з іншими створеними модулями системи та виконувати перевірку на несуперечність даних. Для ілюстрації потенціалу Lean використано декілька прикладів задач, що демонструють, які є переваги мови в забезпеченні точності розв'язків та виявленні помилок у математичних задачах.

У висновках до статті підкреслено, що Lean ефективно доповнює рекомендаційну систему для розв'язування простих планіметричних задач, забезпечуючи не лише перевірку на несуперечність умов задачі, але й коректність розв'язків. Додатково висвітлено певні вади в мові, що ускладнюють її використання й потребують подальших досліджень та розробок для оптимізації роботи підсистеми. Порівняно з іншими підходами, як-от використання онтологій в програмі Protégé, Lean надає значні переваги в гнучкості визначення математичних моделей і виконанні нетривіальних доведень, що робить Lean потужним інструментом для підтримки якісного вивчення математики.

Ключові слова: Lean, Protégé, доведення теорем, перевірка на несуперечність, рекомендаційна система, геометрія, онтологія, Reasoner, SWRL.

SMYSH Oleh
Institute of Software Systems of the National Academy of Sciences of Ukraine
ZAHORULKO Andrii
National University of Kyiv-Mohyla Academy

USING LEAN PROGRAMMING LANGUAGE TO CHECK INCONSISTENCY IN MATHEMATICAL PROBLEMS IN THE EDUCATIONAL RECOMMENDATION SYSTEM

This article describes the problem of ensuring the quality of mathematics learning in the conditions of the growing popularity of self-study. An approach based on the Lean functional programming language is proposed, which enables automatic checking for the accuracy and consistency of the data in a mathematical problem and the solution of this problem. This approach is implemented in the recommendation system that is designed for solving mathematical problems, with an interface for interaction and training the users. The created subsystem based on the Lean language can identify potential errors and inconsistencies in the conditions of the task, as well as the contribution to the correct answer of the problem.

The Lean programming language was primarily designed to serve as an efficient assistant in theorem proving, which is why this language was primarily chosen.

The Lean verification subsystem was developed by adding axioms and theorems from planimetry, which allowed the integration with the other created system modules and also allowed to perform the data consistency checks. To illustrate the potential of Lean, several examples of problems are used to demonstrate the advantages of language in ensuring the accuracy of solutions and detecting errors in mathematical problems.

In the conclusions of the article, it is emphasized that Lean effectively complements the recommendation system for solving simple planimetric problems, providing not only the verification for the inconsistency of the conditions of the problem but also the correctness of the final solutions. In addition, certain defects in the language are highlighted, which complicates its use and requires further research and development to optimize the subsystem. Compared to other approaches, such as the use of ontologies in Protégé, Lean provides significant advantages in the flexibility of defining mathematical models and performing non-trivial proofs, making Lean a powerful tool to support quality mathematics learning.

Key words: Lean, Protégé, theorem proving, consistency checking, recommendation system, geometry, ontology, Reasoner, SWRL.

Постановка проблеми у загальному вигляді
та її зв'язок із важливими науковими чи практичними завданнями

В епоху розвитку дистанційного навчання та на тлі ситуації в Україні, дедалі вагоміше постає питання навчання у сфері здобуття знань. Учні, абітурієнти та студенти все частіше звертаються до самонавчання, без залучення вчителів, викладачів чи репетиторів. Очевидно, це економить людські та грошові ресурси, проте не завжди гарантує якісний кінцевий результат.

Аби покращити та спростити навчання, запропоновано створення автоматичної рекомендаційної системи, що сформує новий підхід до вивчення геометрії. Система здатна приймати на вхід тексти математичних задач українською мовою, знаходити розв'язок цих задач та інтерактивно взаємодіяти з користувачем, відслідковуючи його хід (кроки) розв'язування задачі [1].

Створювана рекомендаційна система демонструє можливість розв'язування задач через використання текстів зі шкільних підручників, де зберігається точність та несуперечність подання висвітлюваних умов задач. Хоча евристично можливо припустити, що незначний відсоток деяких умов все ж можуть бути некоректними. Якщо ж мова йде, до прикладу, про задачі, які створюються не для друку в підручнику вчителями чи викладачами, без належної додаткової перевірки сформованого, тоді похибки у формулюванні можуть траплятися частіше, що унеможливлює коректний розв'язок задачі. Фактично виникає ситуація, коли потрібно не лише пересвідчуватися в правильності знайденого рішення від користувача, проте й першочергово перевіряти на несуперечність умову поставленої математичної задачі. Тобто ввівши некоректну задачу в створювану рекомендаційну систему чи, коли під час знаходження шуканого користувач припустився помилки, системі не вдасться визначити цю суперечність.

Можна навести приклад тривіальної хиби під час створення умови, коли автор випадково припускається помилки та надає трьом сторонам трикутника значення 2, 3 та 5 сантиметрів відповідно. Фактично, автор такої задачі нехтує властивістю про нерівність трикутника, яка стверджує, що будь-яка сторона довільного трикутника є меншою за суму двох інших його сторін та одночасно більша за їхню різницю.

Так само, важливим аспектом є розв'язок задачі, як відповідь до задачі, що запропонував користувач рекомендаційної системи. Аби пересвідчитися в правильності поданого рішення й уникнути суперечностей в умові та розв'язку, потрібно сформувати апарат перевірки даних задачі, де можливо додавати аксіоми та теореми геометрії, на базі яких унеможливити суперечливі дані.

Формулювання цілей статті

Метою цієї статті є огляд та аналіз можливостей функційної мови програмування Lean для впровадження як додаткового модуля в створювану рекомендаційну систему з розв'язування математичних задач, аби уможливити перевірку, як самих даних задачі, так і результатів пошуку шуканого в задачі безпосередньо від кінцевого користувача системи.

Виклад основного матеріалу

Для перевірки правильності отримуваних значень у системі використано мову програмування Lean. Lean — це функційна мова програмування, яка дає змогу легко писати правильний і підтримуваний код. Також є змога використовувати Lean як інтерактивний засіб перевірки теорем. Програмування на Lean передусім передбачає визначення типів і функцій, що уможливлює зосередження на проблемній галузі та маніпулюванні її даними, а не на деталях програмування [2].

На Lean самостійно написано основні теореми та леми з планіметрії.

Взаємодія з Lean відбувається таким чином: у програму надсилаються отримані дані із задачі (значення, що дано з умови задачі та результати обрахунків користувача); далі програма автоматично перевіряє дані на несуперечність, зіставляючи дані з описаними лемами та теоремами; програма виводить висновки щодо отриманих даних зі своєю базою; у кінці висновок передається назад в основний модуль системи, аби засвідчити користувача чи все коректно.

У рекомендаційній системі, програма Lean відіграє роль додаткового модуля (підсистема) перевірки задач на несуперечність.

Далі описано дослідження можливостей використання Lean для розв'язання геометричних задач із різними плоскими фігурами на основі властивостей та означень. Підсистему сформовано на трьох невизначених термінах: Point, Line, Plane. Визначення відношень реалізовано за допомогою аксіом у Lean (див. Рис. 1).

ISSN 2219-9365

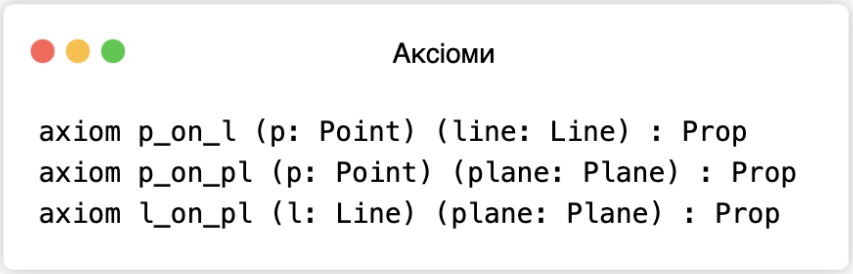

Рис. 1. Приклад визначення аксіом у Lean

Також, для спрощення роботи, визначено інфіксні нотації (див. Рис. 2), що надають можливість визначати приналежність точки/прямої до прямої/площини за допомогою символів «∈» та «⊂».

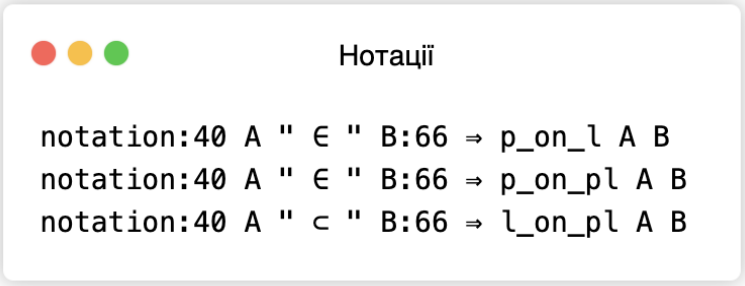

Рис. 2. Приклад визначення інфіксних нотацій у Lean

Lean надає можливість визначати аксіоми та теореми за допомогою термінів логіки першого порядку. Наприклад, аксіому про існування лише однієї прямої, яка перетинає дві точки, реалізовано через використання нотації «∃!», що позначає існування унікальної прямої (Див. Рис. 3).

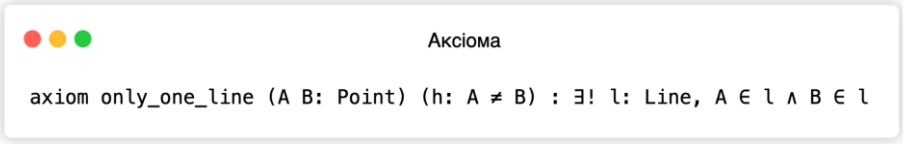

Рис. 3. Визначення аксіоми про одну пряму для точок

На основі точки та прямої побудовано такі структури, як: Segment, Ray, Angle, Triangle, Quadrilateral та інші. Кожна зі структур містить певні вимоги до даних, адже Lean не надає можливості обмежувати створення структур. Як наслідок, усі базові властивості є атрибутами.

Зокрема, визначення кута містить лівий промінь, правий промінь, доведення рівності початкових точок та доведення, що промені не лежать на одній прямій (Див. Рис. 4).

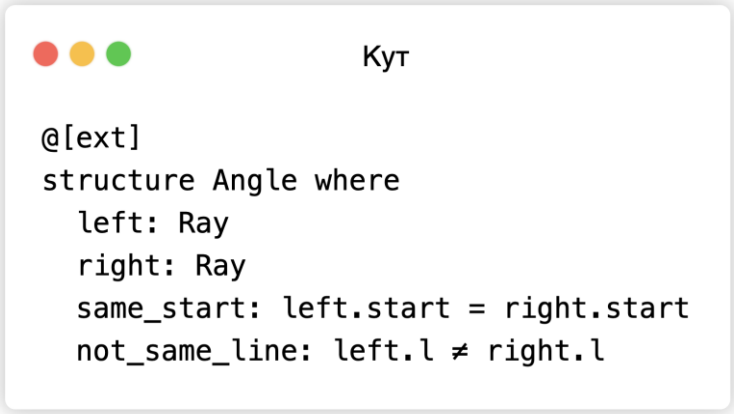

Рис. 4. Визначення кута

Задля уникнення роботи з множинами більшість структур використовують атрибути для кожного елемента множини. Геометричні фігури містять усі точки як атрибути в певному порядку. Як наслідок, трикутники ABC та BCA відрізнятимуться в межах системи.

Рішення про порядок атрибутів потребувало створення додаткових теорем та аксіом. Наприклад, для міри кута додано аксіому про рівність мір звичайного кута з кутом із перевернутими променями (Див. Рис. 5).

Аксіома

```
@[simp] axiom angle_flip_measure (a: Angle) : angle_measure a = angle_measure a.flip
```

Рис. 5. Визначення додаткових аксіом у Lean

Визначення, що можна застосувати до параметрів у різному порядку обмежено до одного. Наприклад, прямокутний трикутник у системі передбачає прямий кут в «першого» кута. Оскільки система дає можливість легко переставити порядок точок у чотирикутника і, як наслідок порядок кутів, то це обмеження не створило критичних вад.

Через демонстраційну мету підсистеми більшість проміжних теорем уведено, проте написано без доведення. Доведення усіх теорем на основі фундаментальних аксіом не є необхідним для тестування та демонстрації можливостей Lean. Задля уникнення виникнення математичного софізму, теореми без доведення обмежено до загальновідомих та тривіальних.

Тестування проведено на основі чотирьох задач. Головною метою тестування є дослідити можливості підсистеми, що сформовано на Lean, в різних аспектах та показати її переваги та недоліки у порівнянні з використанням Protégé. Protégé — це безплатний редактор онтологій з відкритим вихідним кодом і фреймворк для створення інтелектуальних систем [3].

Демонстрація задачі №1 про доведення однієї точки перетину.

Задачу №1 призначено для перевірки можливості доведення теорем лише за допомогою аксіом. Теорема про перетин прямих в одній точці потребує лише використання аксіоми про одну пряму для двох точок та базових теорем з основної бібліотеки Mathlib4 [4]. Вирішення відбувається за допомогою розділення існування унікальної точки на наявність точки, яка задовольняє умову, та доведення рівності точок, які лежать на обох прямих (Див. Рис. 6).

Теорема

```
theorem only_one_intersection (l1 l2: Line) (l1_intersects_l2: l_intersects_l l1 l2) (l1_ne_l2: l1 ≠ l2)
    : ∃! p: Point, p ∈ l1 ∧ p ∈ l2 := by
  unfold l_intersects_l at l1_intersects_l2
  apply exists_unique_of_exists_of_unique

  . show ∃ x, x ∈ l1 ∧ x ∈ l2
    apply l1_intersects_l2

  . show ∀ (y₁ y₂ : Point), y₁ ∈ l1 ^ y₁ ∈ l2 → y₂ ∈ l1 ^ y₂ ∈ l2 → y₁ = y₂
    intro a b
    intro r1 r2
    obtain (a_on_l1, a_on_l2) := r1
    obtain (b_on_l1, b_on_l2) := r2
    by_contra v
    rw [← @ne_eq] at v
    let l1_eq_l2 := (only_one_line a b v).unique (And.intro a_on_l1 b_on_l1) (And.intro a_on_l2 b_on_l2)
    exact l1_ne_l2 l1_eq_l2
```

Рис. 6. Теорема про існування лише однієї точки перетину сторін

Доведення рівності точок використовує тактику «by_contra», що надає можливість довести теорему від супротивного. Для цієї задачі суперечність знайдено за допомогою формування твердження «l1 ≠ l2» та порівняння з умовою, що «l1 = l2».

Під час доведення теорем, інструменти Lean надають можливість побачити контекст для кожного етапу розв'язку. Наприклад, для останнього рядка розв'язку, Lean демонструє всі відомі змінні та мету доведення «False» (Див. Рис. 7).

```
1 goal
▼ case hunique.intro.intro
l1 l2 : Line
l1_intersects_l2 : ∃ p, p ∈ l1 ∧ p ∈ l2
l1_ne_l2 : l1 ≠ l2
a b : Point
a_on_l1 : a ∈ l1
a_on_l2 : a ∈ l2
b_on_l1 : b ∈ l1
b_on_l2 : b ∈ l2
v : a ≠ b
l1_eq_l2 : l1 = l2 := ExistsUnique.unique (only_one_line a b v) { left :=
a_on_l1, right := b_on_l1 } { left := a_on_l2, right := b_on_l2 }
⊢ False
```

Рис. 7. Контекстна панель Lean

Отже, Lean ефективно продемонстрував свою здатність доведення цієї теореми, без виявлення будь-яких недоліків у процесі розв'язання.

Демонстрація задачі №2 на сторони паралелограма.

Задачу №2 призначено для перевірки базових здібностей системи та роботи з теоремами. Текст задачі взято зі шкільного підручника геометрії за 8 клас автора А. Мерзляка: «Периметр паралелограма ABCD дорівнює 24 см, ∠ABC = 160°, діагональ AC утворює зі стороною AD кут 10°. Знайдіть сторони паралелограма.» [5].

Lean надає можливість повністю відтворити умову цієї задачі. Оскільки секції з прикладами у програмі Lean потребують вказування мети, використано тривіальне твердження «true = true» (Див. Рис. 8).

Задача

```
example (ABCD: ConvexQuadrilateral) (h: Parallelogram ABCD)
  (per: ABCD.q.perimeter = 24) (h3: ABCD.angle2.value = 160)
  (h4: Angle.can_be_formed_by_segments ABCD.q.side4.inverse ABCD.q.diagonal13)
  (h5: (Angle.form_angle_from_segment ABCD.q.side4.inverse ABCD.q.diagonal13 h4).value = 10)
  : true = true := by
```

Рис. 8. Визначення задачі №2 в Lean

Знаходження сторін відбувається за допомогою доведення рівнобедреності трикутників, що сформовано діагоналлю, та використання рівності сторін між собою, разом із визначенням периметра чотирикутника. Lean із бібліотекою Mathlib4 надає можливість ефективно розв'язувати задачі за допомогою теорем, а також із використанням тактики «linarith», що спроможна автоматично розв'язувати певні перетворення в рівняннях (Див. Рис. 9).

Задача

```
have side4_length : ABCD.q.side4.Length = 6 := by
  calc
    ABCD.q.side4.length = 4 * ABCD.q.side4.length / 4 := by linarith
    _ = (ABCD.q.side4.length + ABCD.q.side4.length + ABCD.q.side4.length + ABCD.q.side4.length) / 4 := by linarith
    _ = (ABCD.q.side4.length + ABCD.q.side4.length + ABCD.q.side3.length + ABCD.q.side4.length) / 4 := by
      nth_rewrite 3 [AD_eq_CD]
      simp
    _ = (ABCD.q.side4.length + ABCD.q.side2.length + ABCD.q.side3.length + ABCD.q.side4.length) / 4 := by
      nth_rewrite 2 [← Parallelogram.side2_con_side4 ABCD h]
      simp
    _ = (ABCD.q.side1.length + ABCD.q.side2.length + ABCD.q.side3.length + ABCD.q.side4.length) / 4 := by
      nth_rewrite 1 [AD_eq_CD]
      nth_rewrite 1 [← (Parallelogram.side1_con_side3 ABCD h)]
      simp
    _ = _ := by
      rw [per]
      linarith
```

Рис. 9. Приклад знаходження довжини сторони DA

Цю задачу додатково попередньо розв'язано за допомогою системи в Protégé з використанням SWRL правил, де різонер уможливлює автоматичне знаходження результату.

Тому, варто зауважити, що мова програмування Lean не має змоги самостійно обраховувати результати розв'язання задачі, а лише полегшує роботу та впевнюється в несуперечності даних.

Демонстрація задачі №3 на відношення кутів.

Задача №3 аналізує можливості створення відношень між значеннями різних елементів. Задачу сформульовано таким чином: «Дано опуклий чотирикутник ABCD із відношенням між кутами 1:2:3:2. Знайти кути чотирикутника ABCD.».

Задачу можливо запрограмувати в Lean без жодних труднощів, проте цей приклад використано, аби унаочнити перевагу мови Lean у порівнянні з програмою для онтологій Protégé. Оскільки SWRL правила в Protégé потребують конкретних значень для застосування, розв'язання цієї задачі потребує побудови особливих класів, які зберігатимуть інформацію про відношення між кутами. Також, через відсутність можливості маніпуляції рівнянь, розробник змушений самостійно створювати формули обрахунку для кожної вільної змінної.

Наприклад, визначення суми кутів чотирикутника має 4 вільні змінні, для обрахунку кожної з яких розробнику потрібно створювати окреме SWRL правило. На противагу Lean, де можна сформувати одну теорему про суму кутів чотирикутника.

Демонстрація задачі №4 на доведення теореми Піфагора.

Задачу №4 призначено для перевірки можливостей Lean у комплексних теоремах. Задачу сформульовано таким чином: «Дано прямокутний трикутник ABC. Довести, що квадрат гіпотенузи дорівнює сумі квадратів катетів.». Розв'язання задачі в Lean виконано на основі подібності трикутників (Див. Рис. 10). Доведення теореми проведено частково через складнощі повного доведення всіх аспектів цієї теореми. Як наслідок, подібність трикутників ABC, HBA та HAC не доведено. Фокусом розв'язку є використання властивостей подібності для знаходження відношення між сторонами.

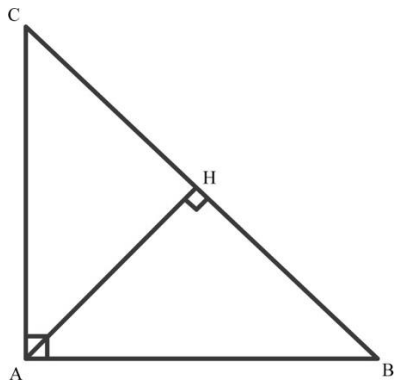

Рис. 10. Зображення трикутника до задачі №4

Доведення теореми Піфагора продемонструвало вади роботи із геометрією в Lean. Далі перелічено ці вади.

Відсутність можливості перейменування та зведення назв елементів ускладнює роботу, оскільки користувачу необхідно обраховувати значення самостійно. Наприклад, Lean вимагатиме від користувача доведення рівності точок сторін трикутника замість прямого іменування точок (Див. Рис. 11).

Рис. 11. Приклад вади іменування значень у Lean

Доведення в прикладі потребує використання тактики рефлективності у Lean, але спершу користувачу необхідно це самостійно зрозуміти.

У деяких випадках при зміні визначень, Lean спрощує терми до аксіоми вибору, що суттєво ускладнює розуміння самого твердження. Як наслідок, використання теорем із квантором існування вимагають обережності.

Кількість тактик, що необхідно для доведення теорем, суттєво збільшують вимоги до користувача підсистеми.

Висновки з дослідження та подальші перспективи у цьому напрямі

На основі продемонстрованих математичних задач можна стверджувати, що мова програмування Lean придатна для доведення теорем у планіметрії. Сформована підсистема на основі Lean, що використані в створюваній рекомендаційній системі, дає змогу перевіряти умову задачі та результати розв'язку задачі на несуперечність за допомогою сформованої бази теорем та лем.

Мова надає необхідні та достатні інструменти для опису аксіоматики та подальшого доведення теорем. Хоча, варто зазначити, що існують наразі певні вади в мові Lean, що ускладнюють сформовану підсистему, проте не унеможливлюють роботу з нею.

Потрібно додати, що в порівнянні з системою Protégé, у створеній підсистемі з Lean визначено такі переваги, як-от: можливість визначати аксіоми та теореми за допомогою логіки першого порядку; можливість маніпулювати рівняннями, підставляти значення, що суттєво зменшує кількість необхідних теорем; можливість працювати зі складеними виразами без необхідності обрахунку (коренями чисел, многочленами), що уможливлює отримання точних результатів.

Література

1. Жежерун О. П., Смиш О. Р. Автоматизація розв'язування задач із планіметрії, записаних природною українською мовою. Проблеми програмування. 2020. № 4. С. 071–080. URL: https://doi.org/10.15407/pp2020.04.071
2. LEAN (4). (2024). *About*. https://lean-lang.org/about/ (дата звернення: 10.03.2024)
3. Musen, M.A. The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003.
4. Leanprover-Community. *GitHub - leanprover-community/mathlib4: The math library of Lean 4*. GitHub. URL: https://github.com/leanprover-community/mathlib4 (дата звернення: 10.03.2024)
5. Мерзляк А. Г. Геометрія: підруч. для 8 кл. закладів заг. серед. освіти. 2-ге вид. Харків, Україна: Гімназія, 2021. 208 с.

References

1. Zhezherun, O. P., & Smysh, O. R. (2020). Avtomatyzatsiia rozviazuvannia zadach z planimetrii, zapysanykh pryrodnoiu ukrainskoiu movoiu [Automation of solving planimetry problems written in Ukrainian]. Problemy Prohramuvannia – PROBLEMS IN PROGRAMMING, (4), 71-80 [in Ukrainian].
2. LEAN (4). (2024). *About*. Retrieved from https://lean-lang.org/about/
3. Musen, M.A. The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003.
4. Leanprover-Community. *GitHub - leanprover-community/mathlib4: The math library of Lean 4*. GitHub. Retrieved from https://github.com/leanprover-community/mathlib4
5. Merzliak , A. H. (2021). *Heometriia: pidruch. dlia 8 kl. zakladiv zah. sered. osvity.* (2nd ed.) [in Ukrainian].