Прикарпатський національний університет імені Василя Стефаника
Факультет Математики та Інформатики
Кафедра Алгебри та геометрії

ДИПЛОМНА РОБОТА

на здобуття першого (бакалаврського) рівня вищої освіти

на тему: «Програмна реалізація алгоритму роботи тренажеру «Визначники та їх застосування»

Виконав: студент 4 курсу , групи М-41

Спеціальності 111 Математика

_____

(шифр і назва спеціальності)

Глушко М.Р.

_____

(прізвище та ініціали студента)

Керівник Микицей О.Я

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Івано-Франківськ – 2025 р

АНОТАЦІЯ

У дипломній роботі представлено розробку програмного тренажера з теми «Визначники та їх застосування». Метою дослідження було створення навчального інструменту, який поєднує теоретичну, практичну та тестову частини для покращення засвоєння матеріалу з однієї з ключових тем лінійної алгебри. Програму реалізовано мовою C++ з використанням Windows API у середовищі Microsoft Visual Studio. Тренажер містить інтуїтивно зрозумілий графічний інтерфейс і не потребує встановлення додаткових бібліотек, що дозволяє використовувати його на більшості комп’ютерів у навчальних закладах. У ході реалізації були створені модулі теорії, практики та тестування, які дозволяють поступово опановувати матеріал і перевіряти рівень знань. Результати тестування підтвердили стабільність роботи програми та її придатність для використання у навчальному процесі.

ABSTRACT

The thesis presents the development of an educational software simulator titled “Determinants and Their Applications". The main objective of the project was to create a digital learning tool that integrates theoretical materials, practical exercises, and testing modules to enhance the understanding of one of the fundamental topics in linear algebra. The simulator was developed in C++ using the Windows API within the Microsoft Visual Studio environment. It features a user-friendly graphical interface and requires no additional libraries, ensuring easy deployment on most educational computers. The application includes three core modules: theory, practice, and testing, which support a step-by-step learning process and allow users to evaluate their understanding. Testing results confirmed the stability and usability of the program in an educational context.

ЗМІСТ

Перелік умовних скорочень, термінів та позначень

UML — Unified Modeling Language (Уніфікована мова моделювання); графічна мова для візуалізації, опису, проєктування та документування програмних систем.

WinAPI — Windows Application Programming Interface; набір функцій і структур, що надаються операційною системою Windows для створення графічного інтерфейсу та керування ресурсами.

IDE — Integrated Development Environment (Інтегроване середовище розробки); програмне середовище, яке забезпечує зручну роботу програміста при написанні, компіляції та налагодженні коду.

C++ — мова програмування загального призначення, яка підтримує процедурне, об'єктно-орієнтоване та шаблонне програмування.

MSVC – Microsoft C++ Compiler – компілятор C++ від компанії Microsoft.

GDI – Graphics Device Interface – інтерфейс графічного пристрою для рендерингу у Windows.

GUI – Graphical User Interface – графічний інтерфейс користувача.

LMS – Learning Management System – система управління навчанням (наприклад, Moodle).

DLL – Dynamic-Link Library – динамічно підключувана бібліотека.

СЛАР – Система лінійних алгебраїчних рівнянь – математична модель з кількома змінними.

API – Application Programming Interface – інтерфейс прикладного програмування.

HTML5 – HyperText Markup Language version 5 – мова розмітки веб-сторінок.

SSL – Secure Sockets Layer – протокол захищеної передачі даних в інтернеті.

VS – Visual Studio – інтегроване середовище розробки (IDE), створене компанією Microsoft, яке використовується для написання, компіляції, налагодження і запуску програм, зокрема мовами C++, C#, Python та іншими.

ВСТУП

У сучасних умовах цифрової трансформації освіти зростає потреба у впровадженні нових інформаційно-комунікаційних технологій у навчальний процес. Особливої актуальності набувають програмні засоби для дистанційного навчання, які не лише забезпечують доступ до навчального матеріалу з будь-якої точки світу, але й сприяють формуванню стійких практичних навичок завдяки інтерактивним інструментам. Одним із таких інструментів є тренажери — спеціалізовані програмні модулі для самостійного опрацювання матеріалу, виконання вправ і тестування знань.

Дисципліна «Лінійна алгебра» є базовою у підготовці фахівців у галузях інформатики, прикладної математики, інженерії. Тема «Визначники та їх застосування» є однією з основних тем цієї дисципліни, оскільки широко використовується у розв’язанні систем лінійних рівнянь, обчисленні обернених матриць, знаходженні площ, об’ємів, перевірці лінійної незалежності векторів тощо. Проте ця тема часто викликає труднощі у здобувачів освіти через складність теоретичного матеріалу та потребу в точному обчисленні.

Зважаючи на потребу підвищення якості засвоєння даного матеріалу в умовах дистанційного чи змішаного навчання, **актуальною** є розробка спеціального тренажеру, який би поєднував теоретичну, практичну та контролюючу складову.

Мета дослідження — розробити програмний тренажер для вивчення теми «Визначники та їх застосування» у курсі лінійної алгебри, який дозволить користувачеві:

1) отримати теоретичні знання;
2) закріпити їх за допомогою практичних завдань;
3) пройти тестування для оцінки рівня засвоєння матеріалу.

Завдання дослідження:

1) проаналізувати програмні засоби, які використовуються для реалізації освітніх тренажерів;

2) здійснити огляд існуючих програм аналогічного призначення;

3) систематизувати теоретичні відомості з теми «Визначники та їх застосування»;

4) спроєктувати структуру програмного забезпечення;

5) реалізувати функціональну програму з трьома модулями: теорія, практика, тестування;

6) провести тестування працездатності розробленого застосунку.

Об’єкт дослідження — процес вивчення теми «визначники» у курсі лінійної алгебри в умовах дистанційного навчання.

Предмет дослідження — методи програмної реалізації навчального тренажеру, який включає теоретичний матеріал, практичні завдання та тестування знань.

Методи дослідження:

1) теоретичний аналіз наукової літератури з теми лінійної алгебри та педагогічних технологій;

2) порівняльний аналіз програмних засобів навчального призначення;

3) методи програмної інженерії (аналіз вимог, проектування, реалізація, тестування);

4) моделювання та тестування результатів у вигляді прототипу програмного продукту;

5) емпіричні методи – тестування працездатності застосунку та оцінювання його функціональності.

Результатом дипломної роботи є програмний продукт, що може бути використаний студентами, викладачами та всіма, хто прагне опанувати або поглибити знання з теми визначників у лінійній алгебрі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд програмних засобів, потрібних для роботи

Для реалізації програмного тренажеру «Визначники та їх застосування» було обрано мову програмування C++ у поєднанні з Windows API (Application Programming Interface), що дозволяє створювати нативні графічні інтерфейси для операційної системи Windows. Такий вибір забезпечує високу продуктивність, повний контроль над елементами управління та можливість створення легких і швидких застосунків без використання сторонніх бібліотек.

Microsoft Visual Studio (рис. 1.1) — це одне з найпотужніших та найпоширеніших інтегрованих середовищ розробки (IDE), яке активно використовується для створення програмного забезпечення різного призначення: від консольних утиліт до складних графічних додатків, вебзастосунків і мобільних програм. Visual Studio розроблена компанією Microsoft і забезпечує широкі можливості для програмування на мовах C++, C#, VB.NET, Python та інших [1].

У контексті дипломної роботи середовище VS 2022 було обране як основний інструмент для створення настільного застосунку на мові C++, з використанням стандартного API Windows (WinAPI). Це середовище включає зручний редактор коду, вбудовану систему збірки, менеджер рішень, підтримку відлагодження, а також численні розширення та засоби аналізу продуктивності коду.

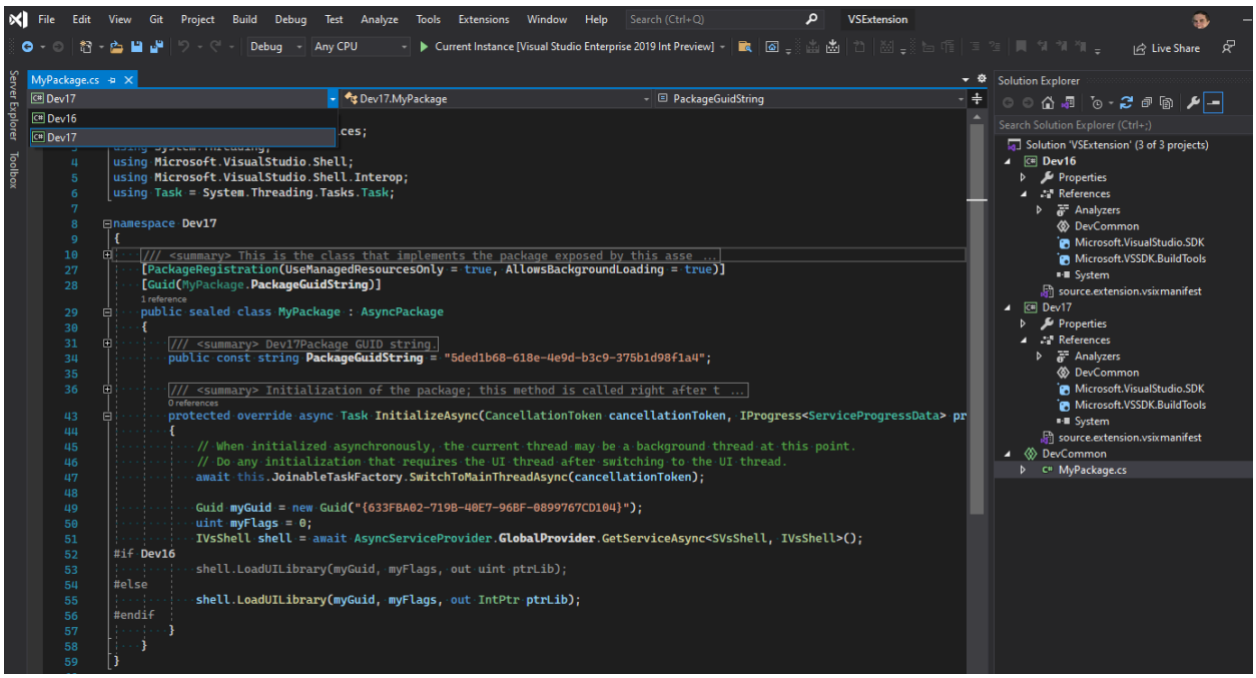

Рис. 1.1 – Інтерфейс Microsoft Visual Studio

Однією з важливих функцій, які стали вирішальними у виборі Visual Studio для даного проєкту, є можливість встановлення пакета Desktop development with C++. Цей пакет містить усе необхідне для створення настільних програм у Windows, а саме:

1) компілятор Microsoft C++ (MSVC);
2) бібліотеки Windows SDK;
3) шаблони проєктів для консольних і віконних додатків;
4) засоби створення графічного інтерфейсу за допомогою WinAPI або інших технологій;
5) інструменти діагностики, тестування та профілювання коду.

Завдяки широкій інтеграції з операційною системою Windows, Visual Studio забезпечує:

1) просте створення віконних елементів (кнопок, полів, меню);
2) автоматичну збірку та запуск проєкту;
3) покрокове налагодження з можливістю встановлення точок зупину (breakpoints), перегляду змінних і стану пам’яті;

4) зручну навігацію по структурі проєкту, що суттєво пришвидшує розробку.

Використання Microsoft Visual Studio в дипломному проєкті дозволяє реалізувати складну логіку тренажера з якісним графічним інтерфейсом, забезпечити надійне тестування та налагодження, а також створити виконуваний файл, готовий до використання в навчальному процесі без необхідності встановлення додаткових компонентів.

Windows API (рис. 1.2) — це обширний набір функцій, структур, констант і об’єктів, що надається операційною системою Microsoft Windows для взаємодії прикладного програмного забезпечення з внутрішніми ресурсами системи. WinAPI є основою для розробки настільних додатків під Windows, забезпечуючи прямий доступ до низькорівневих системних функцій, графічного інтерфейсу, файлової системи, обробки повідомлень та управління пам’яттю [2].

На відміну від використання сторонніх бібліотек або фреймворків (наприклад, Qt, MFC, .NET), програмування з використанням WinAPI дозволяє створювати максимально оптимізовані й легкі застосунки, що не потребують встановлення додаткових компонентів. Це особливо актуально для розробки освітніх програмних засобів, які мають бути доступні на широкому колі комп’ютерів без складної інсталяції.

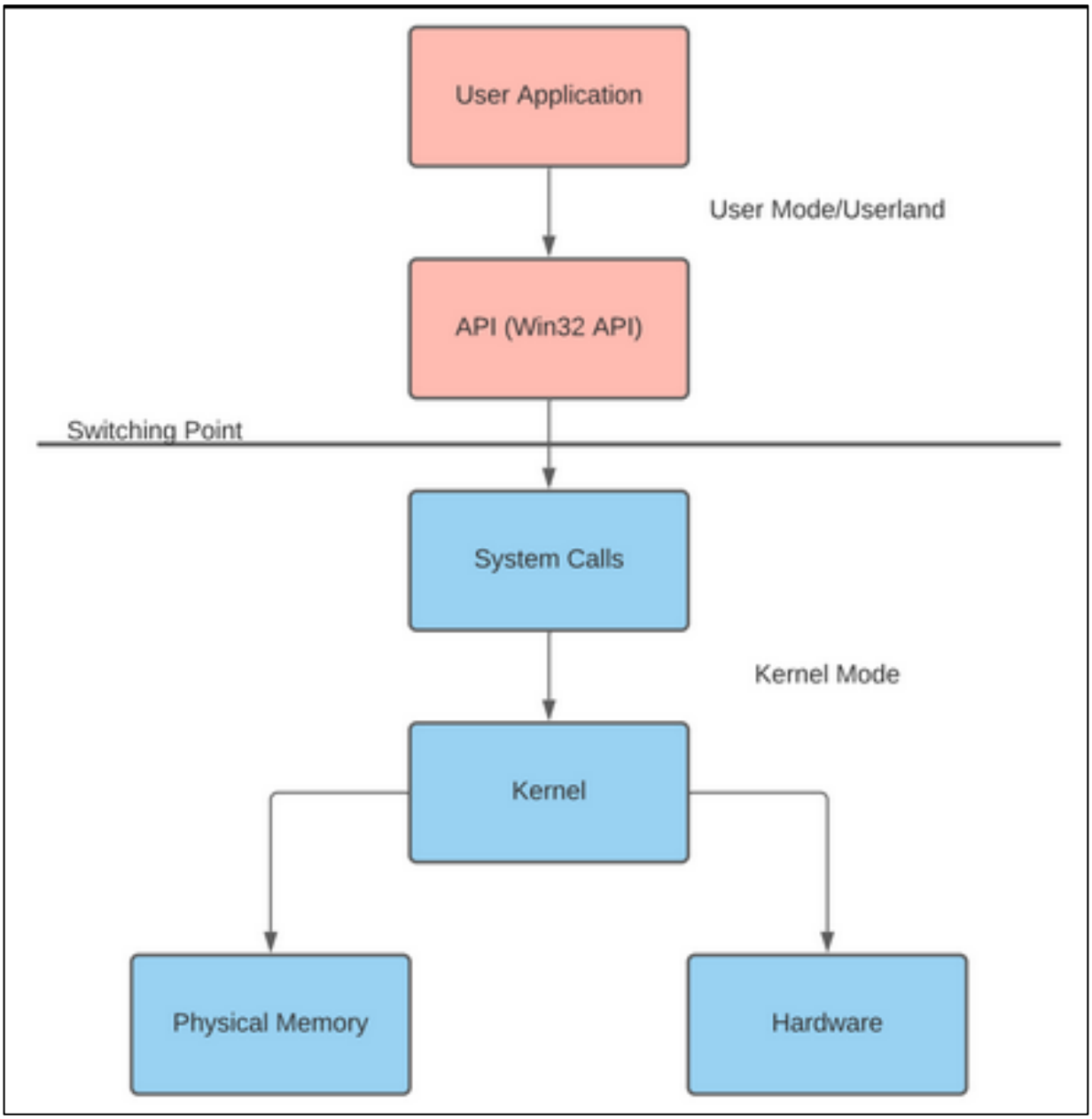

Рис. 1.2 – Архітектура Windows API

У межах дипломної роботи Windows API використано для реалізації графічного інтерфейсу тренажера «Визначники та їх застосування», зокрема:

1) створення головного вікна програми з фіксованим розташуванням і розмірами;

2) розміщення кнопок («Теоретичні відомості», «Практичні завдання», «Пройти тестування»);

3) реакція на події (натискання кнопок, запуск діалогових вікон, вибір варіантів відповіді);

4) виведення текстових повідомлень через MessageBox або інші стандартні елементи управління;

5) базова обробка повідомлень Windows через цикл GetMessage() та функцію зворотного виклику WindowProc.

Завдяки використанню WinAPI, програмі вдалося надати мінімалістичний, але функціональний інтерфейс, який не перевантажує систему та має високу продуктивність. Це робить розроблений застосунок ідеальним для використання в освітньому середовищі — навіть на комп’ютерах із базовими технічними характеристиками.

До основних переваг використання WinAPI у дипломному проєкті можна віднести:

1) відсутність залежностей від зовнішніх бібліотек чи пакетів;

2) повний контроль над структурою вікна, подіями та зовнішнім виглядом;

3) висока швидкодія та стабільність роботи програми;

4) можливість запуску навіть на старих версіях Windows без додаткових компонентів.

Застосування Windows API у поєднанні з Microsoft Visual Studio стало оптимальним рішенням для реалізації компактного, швидкого і зручного тренажера для дистанційного навчання з теми «Визначники та їх застосування».

MSVC (Microsoft C++ Compiler) — це потужний компілятор мови програмування C та C++, який входить до складу Visual Studio і є одним з основних інструментів для розробки програмного забезпечення під операційні системи Windows. Він забезпечує повну підтримку стандарту ISO C++ (у т.ч. сучасних версій C++17, C++20, C++23), а також сумісність із фреймворками Microsoft, зокрема Windows API, що є критично важливим для створення нативних Windows-застосунків.

Компілятор MSVC використовується на етапі перетворення вихідного коду (написаного розробником) у виконуваний машинний код (*.exe або *.dll

файли), який безпосередньо виконується процесором. У межах дипломної роботи компілятор відіграє важливу роль — саме за його допомогою C++-код тренажера перетворюється у працездатну Windows-програму з графічним інтерфейсом [3].

Основні переваги використання MSVC у дипломному проєкті:

1) сумісність з Windows API — компілятор повністю підтримує системні заголовки Windows (windows.h та інші), що дозволяє створювати програми з вікнами, кнопками, подіями, повідомленнями тощо.

2) висока продуктивність — оптимізації компілятора дозволяють створювати виконувані файли з мінімальним об’ємом і максимальною швидкодією.

3) підтримка налагодження — MSVC працює у зв’язці з відлагоджувачем Visual Studio, дозволяючи встановлювати точки зупину, переглядати змінні, виклики функцій, стан стеку тощо.

4) гнучке налаштування — підтримка компіляції під різні конфігурації (Debug/Release), архітектури (x86, x64), а також включення додаткових параметрів оптимізації, безпеки та продуктивності.

5) автоматичне оновлення — регулярні оновлення компілятора забезпечують актуальність інструментів, відповідність новим стандартам C++ і виправлення помилок.

У межах реалізації програмного тренажеру MSVC забезпечив швидке компілювання, стабільну роботу, можливість запуску без зовнішніх бібліотек, а також просте формування .exe-файлу, який може бути використаний на будь-якому комп’ютері з Windows. Таким чином, компілятор MSVC є невід’ємною частиною інфраструктури проєкту, що забезпечує ефективну та якісну розробку програмного забезпечення з використанням мови C++ та середовища Windows.

Середовище виконання Windows — це базова програмна платформа, яка забезпечує функціонування прикладних програм у межах операційної системи Microsoft Windows. Для розробленого в межах дипломної роботи тренажера

«Визначники та їх застосування», створеного за допомогою мови програмування C++ і технології Windows API, середовищем виконання є безпосередньо сама ОС Windows, без необхідності встановлення будь-яких додаткових компонентів або сторонніх бібліотек.

Особливістю створеного застосунку є те, що він є нативним — тобто таким, що повністю сумісний з інфраструктурою Windows і використовує її внутрішні ресурси (зокрема графічну підсистему GDI, обробник повідомлень, засоби виведення тексту, обробку подій тощо). Завдяки цьому забезпечується мінімальний об’єм виконуваного файлу, відсутність зовнішніх залежностей (на кшталт .NET Framework, Java Runtime Environment, Qt або інших), що значно спрощує:

1) встановлення програми (достатньо просто скопіювати .exe файл);

2) розгортання в навчальних аудиторіях (не потрібно мати права адміністратора для встановлення середовищ);

3) переносимість між різними комп’ютерами, які працюють під Windows 7, 8, 10 або 11.

Крім того, така модель середовища виконання дозволяє:

1) запускати програму навіть на слабших комп’ютерах або старих ноутбуках;

2) гарантувати стабільну роботу без збоїв, викликаних відсутністю певних бібліотек;

3) усувати залежність від інтернету або сторонніх серверів — уся логіка та інтерфейс вбудовані у програму.

Для успішної роботи тренажера достатньо будь-якої версії Windows, починаючи з Windows 7 (з Service Pack 1) до найновіших редакцій Windows 11, незалежно від розрядності (x86 чи x64). Усі компоненти, що використовуються у розробці, входять до складу Windows за замовчуванням (зокрема бібліотеки user32.dll, gdi32.dll, kernel32.dll тощо).

Отже, середовище виконання Windows у поєднанні з розробкою через WinAPI дозволило створити простий у запуску, стабільний та автономний

програмний продукт, що є важливою перевагою для його використання в умовах дистанційного або офлайн-навчання без додаткового програмного забезпечення. Опишемо також згадані засоби в таблиці 1.1.

Таблиця 1.1

Основні засоби, використані в проєкті

Назва програмного засобу	Призначення	Роль у проєкті
Microsoft Visual Studio 2022	Інтегроване середовище розробки (IDE) для програмування мовою C++	Створення, редагування, компіляція, налагодження та запуск проєкту
MSVC (Microsoft C++ Compiler)	Компілятор для мови C++	Перетворення вихідного коду у виконуваний файл (.exe), оптимізація продуктивності
Windows API (WinAPI)	Набір функцій для створення інтерфейсу, обробки подій, роботи з вікнами	Реалізація графічного інтерфейсу користувача, взаємодія з ОС Windows
Середовище виконання Windows	Операційна система Windows 7, 8, 10, 11	Виконання програми без додаткових бібліотек чи залежностей

Використання зазначених програмних засобів обумовлене необхідністю реалізувати легкий у використанні, швидкий і функціональний інтерфейс, який забезпечує розділення логіки програми на три основні модулі: теоретичний, практичний і тестовий. Такий підхід дозволяє ефективно реалізувати навчальні цілі тренажера без потреби у складному встановленні чи залежностях, що робить розроблену програму придатною для впровадження у процес навчання з дисципліни «Лінійна алгебра».

1.2 Огляд існуючих тренажерів

На сучасному етапі розвитку цифрової освіти дедалі ширше застосовуються електронні освітні ресурси, до яких належать віртуальні лабораторії, інтерактивні задачники, мобільні застосунки та веб-платформи. Однак, спеціалізовані тренажери, які спрямовані виключно на опрацювання теми «визначники та їх застосування», зустрічаються нечасто. У цьому підрозділі розглянемо кілька програмних рішень, які частково або повністю реалізують подібний функціонал.

Одним із найбільш популярних програмних засобів, що можуть бути частково використані для вивчення теми визначників, є GeoGebra (рис. 1.3). Це безкоштовна динамічна математична програма з відкритим кодом, яка підтримує роботу з алгеброю, геометрією, статистикою та математичним аналізом. У межах даного програмного забезпечення передбачено базові функції роботи з матрицями, включаючи можливість обчислення визначника, транспонування, множення матриць тощо [4].

Однак GeoGebra не містить спеціалізованого навчального контенту, присвяченого виключно темі «визначники». Також відсутня система поступового навчання, завдань і тестів, що робить програму менш ефективною у формуванні структурованих знань у студентів. Інтерфейс GeoGebra, хоча й сучасний, вимагає від користувача базових навичок володіння програмою, що може створити бар’єр для новачків. Попри це, програма залишається корисною як допоміжний інструмент у візуалізації математичних обчислень.

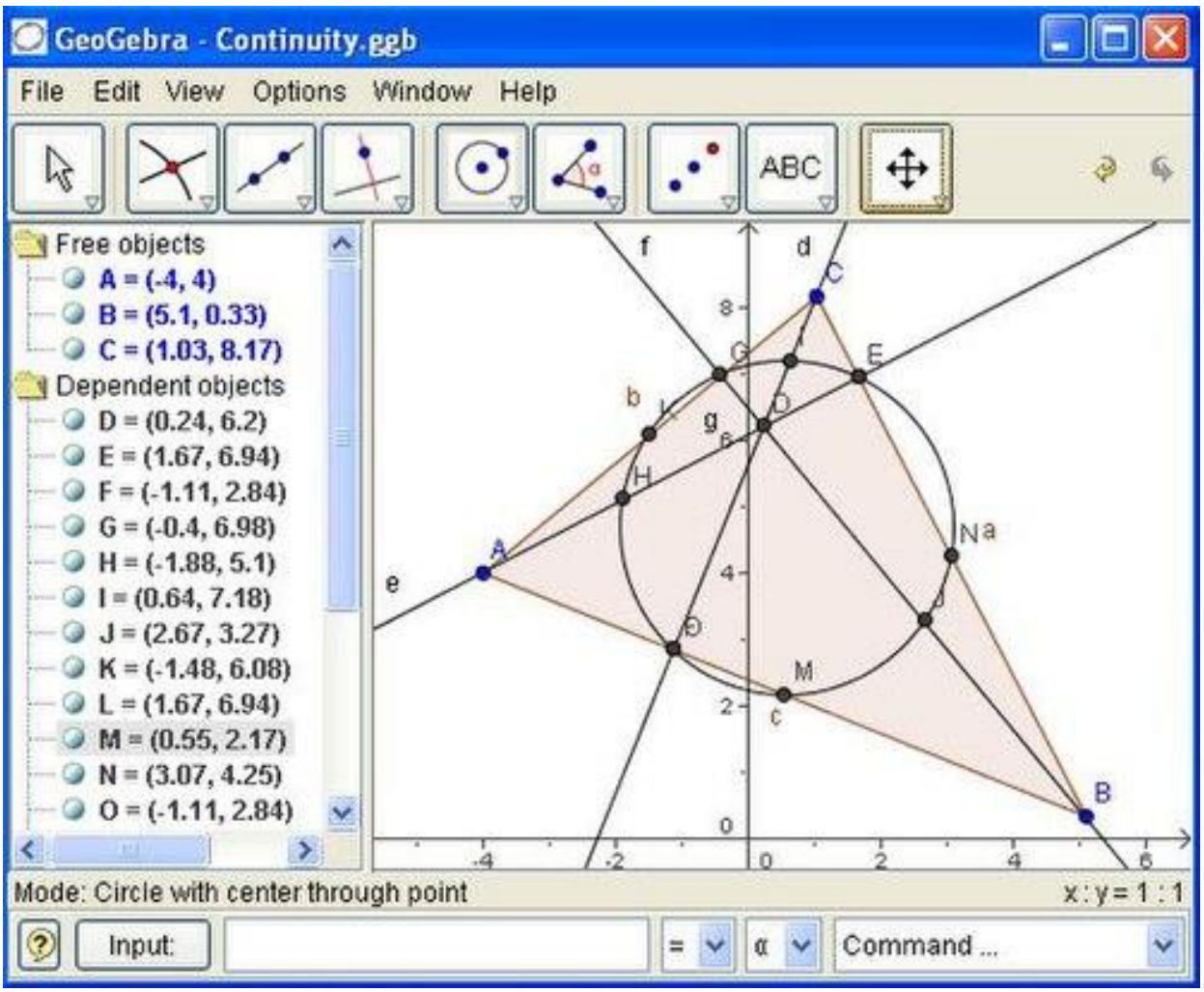

Рис. 1.3 – Інтерфейс GeoGebra [5]

Іншим прикладом реалізації тренажерних можливостей є курси на базі Moodle (рис. 1.4) або Mebis — платформи дистанційного навчання, які широко застосовуються в академічному середовищі, зокрема в Європі. Ці середовища дозволяють викладачам створювати індивідуальні курси, де можна включати тести з теми визначників, додавати завдання з обчисленням, а також розміщувати теоретичні матеріали. Moodle та Mebis мають потужний інструментарій для контролю знань, зокрема автоматичну перевірку тестових відповідей і формування зворотного зв’язку. Проте, більшість таких курсів мають вузьку спеціалізацію, створюються локально для внутрішнього використання, часто закриті для вільного доступу, і потребують авторизації. Крім того, функціонал візуалізації обмежений і значною мірою залежить від навичок викладача, який формує курс [6].

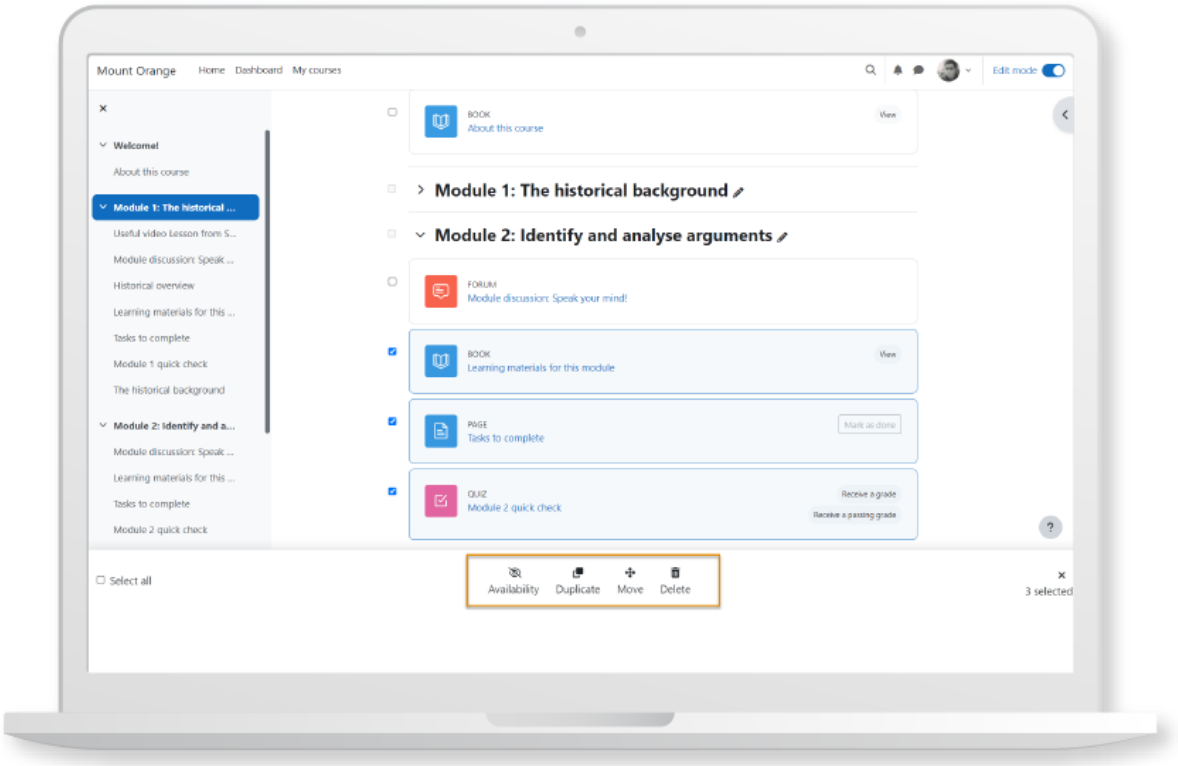

Рис. 1.4 – Інтерфейс Moodle [6]

До категорії прикладних онлайн-ресурсів належать такі сервіси, як Mathway [7] і Symbolab [8] (рис. 1.5). Вони позиціонуються як потужні математичні калькулятори, здатні не лише знаходити значення визначника, але й демонструвати поетапні розв’язання задач. Ці сервіси можуть бути особливо корисними для швидкої перевірки правильності обчислень або ознайомлення з технікою розв’язання. Однак їх використання як повноцінного навчального тренажера є обмеженим, адже відсутній навчальний контекст, немає теоретичних пояснень, контролю засвоєння знань або тестової перевірки. Вони працюють лише як «машини для обчислень», без врахування педагогічної складової.

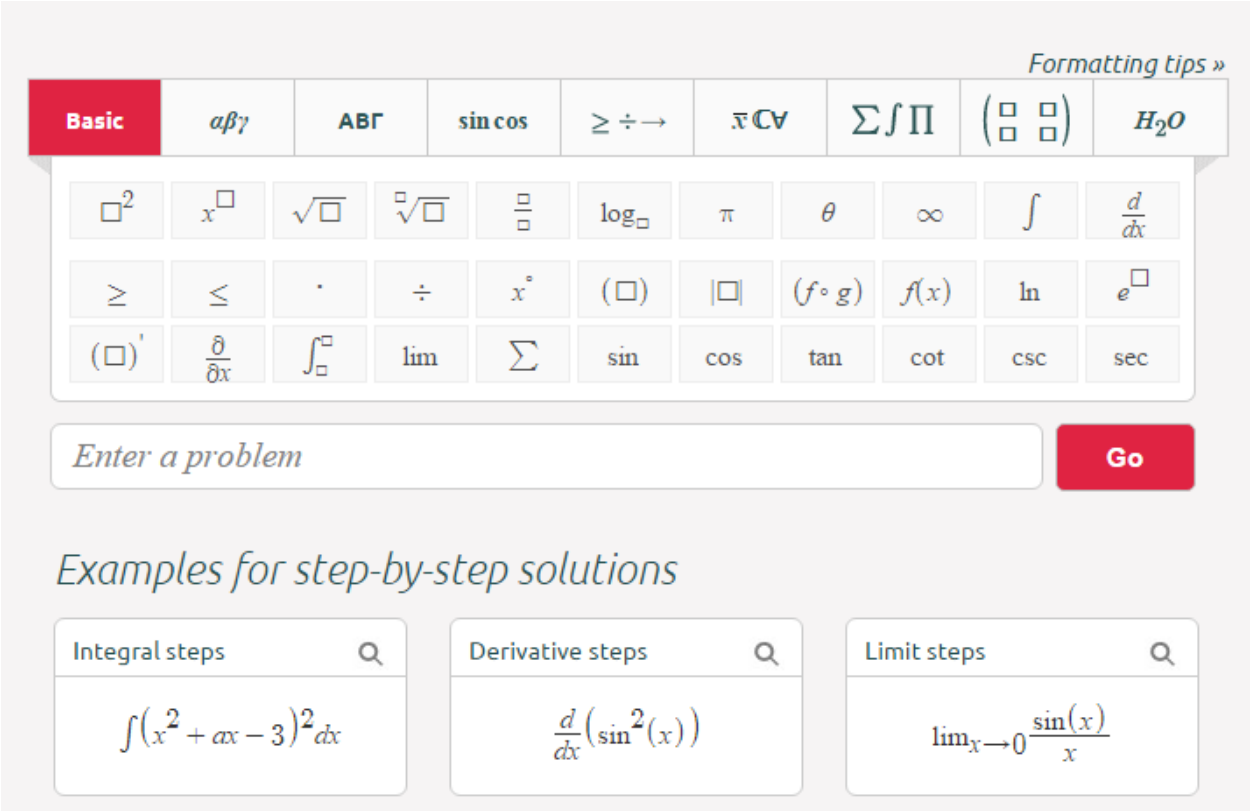

Рис. 1.5 – Інтерфейс Symbolab [8]

Окрему групу складають мобільні застосунки, такі як Matrix Calculator (рис. 1.6), Math Solver та інші. Ці програми розроблені переважно для швидких обчислень і виконання базових дій з матрицями: обчислення визначника, транспонування, обернення матриці тощо. Вони є зручними у використанні, мають інтуїтивно зрозумілий інтерфейс і не потребують глибоких знань у галузі програмування чи математики. Проте в освітньому аспекті ці додатки є обмеженими: вони не містять розділів з теорією, відсутні практичні завдання і можливість тестування. Таким чином, мобільні калькулятори є лише інструментами для прикладного використання, а не засобами формування або перевірки знань [9].

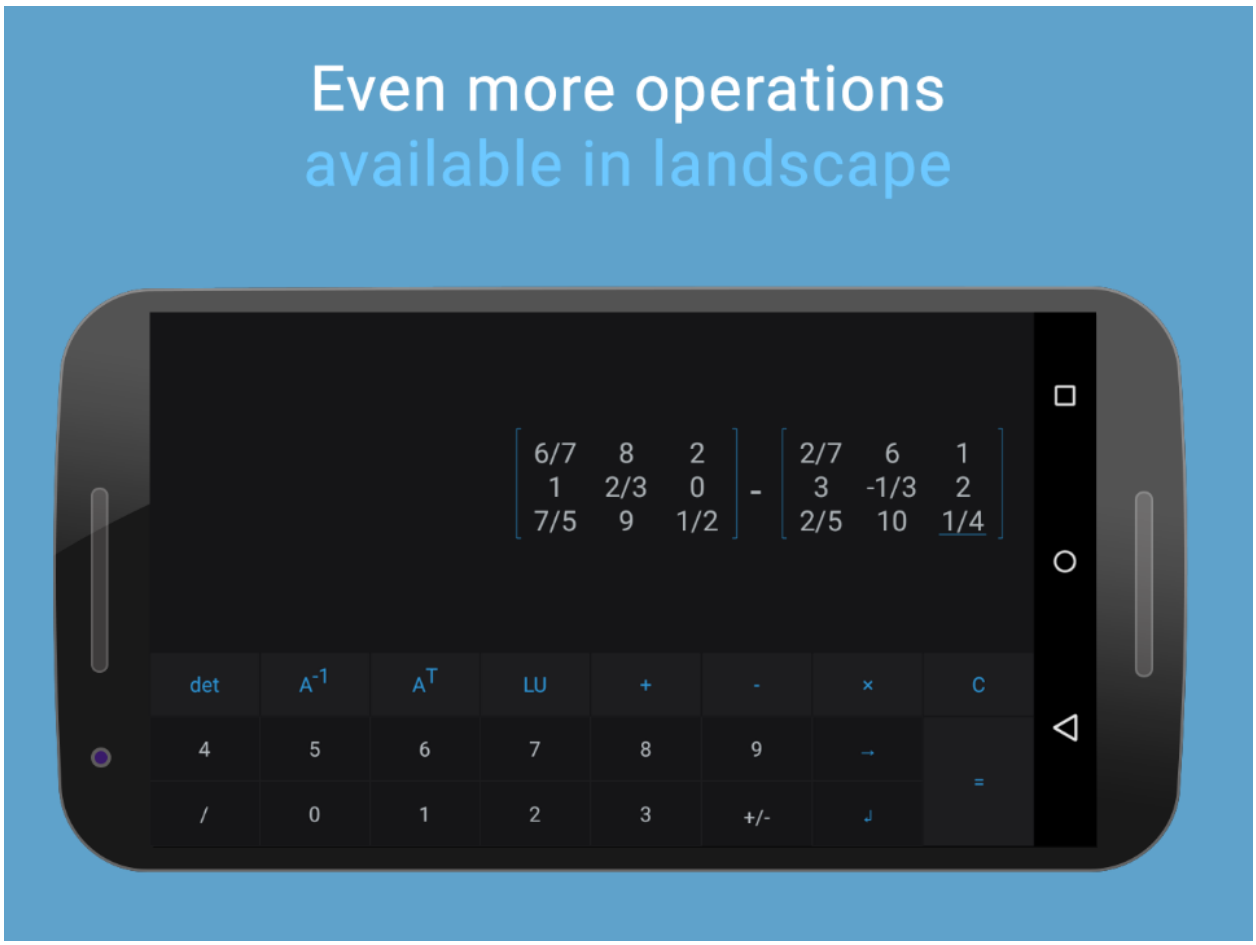

Рис. 1.6 – Інтерфейс Matrix Calculator [9]

Узагальнюючи, можна зазначити, що більшість існуючих рішень виконують лише частину функцій, необхідних для повноцінного навчального тренажера. Саме тому постає потреба у створенні спеціалізованого програмного засобу, що поєднає в собі доступну теоретичну частину, інтерактивні завдання з перевіркою відповідей та модуль тестування, що сприятиме глибокому засвоєнню теми «визначники та їх застосування» в курсі лінійної алгебри.

Також порівняємо описані аналоги в таблиці 1.2.

Таблиця 1.2

Порівняння існуючих аналогів

Назва	Тип продукту	Наявність теорії	Практичні завдання	Тестування знань	Інтерактивність	Потреба в реєстрації/доступі
GeoGebra	Динамічна матем. програма	Обмежено	Так (неструктуровані)	Ні	Висока	Ні
Moodle / Mebis	LMS-платформи	Так	Так	Так	Середня	Так
Mathway / Symbolab	Онлайн-калькулятори	Ні	Ні	Ні	Низька	Ні (частково платні функції)
Мобільні додатки	Калькулятори (Matrix Calc.)	Ні	Ні	Ні	Середня	Ні

На основі аналізу можна зробити висновок, що в наявних тренажерах переважає інструментальна, а не навчальна спрямованість. Більшість з них або не містять систематизованого теоретичного матеріалу, або не забезпечують перевірки знань користувача. Це створює нішу для створення спеціалізованого навчального тренажера, який би об’єднував теоретичну частину, практичні завдання з миттєвим зворотним зв’язком та модуль тестування.

Розробка такого програмного засобу дозволить покращити ефективність дистанційного навчання, забезпечити учням можливість самостійного опрацювання матеріалу та підвищити мотивацію до вивчення лінійної алгебри.

1.3 Теоретичні відомості по темі «Визначники та їх застосування»

Визначник (детермінант) — це числова характеристика квадратної матриці, яка відіграє важливу роль у багатьох розділах лінійної алгебри та її прикладних застосуваннях. Визначник дозволяє оцінити властивості матриці, такі як оберненість, лінійна незалежність її рядків або стовпців, а також використовується в процесі розв'язування систем лінійних алгебраїчних рівнянь (СЛАР) [10].

Для квадратної матриці другого порядку:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ \\ a_{21} & a_{22} \end{pmatrix}$$

визначник обчислюється за формулою:

$$\det(A) = a_{11}a_{22} - a_{12}a_{21}$$

Для матриць вищого порядку (n × n) визначник обчислюється за допомогою розкладу за мінором або методу Гауса.

Визначники мають низку властивостей, які дозволяють спростити обчислення:

1) якщо два рядки або стовпці матриці однакові, то визначник дорівнює нулю;

2) якщо один з рядків або стовпців є нульовим, то визначник також дорівнює нулю;

3) визначник не змінюється при транспонуванні матриці;

4) при перестановці двох рядків або стовпців визначник змінює знак;

5) якщо один рядок або один стовпець матриці помножити на число, то визначник теж множиться на це число;

6) якщо до одного рядка або стовпця додати інший, помножений на деяке число, значення визначника не змінюється [11].

Окрім основних властивостей, визначники також використовуються для визначення рангу матриці — кількості лінійно незалежних рядків (або стовпців). Якщо жоден із визначників порядку більше за деяке значення не дорівнює нулю, то цей порядок є максимальним можливим розміром ненульового мінору, тобто і рангу. Таким чином, поняття мінору і визначника тісно пов’язане з темою рангу матриці, що має велике значення у теорії систем лінійних рівнянь та при аналізі розмірності лінійного простору.

Найбільш поширеними методами обчислення визначників є:

1) розкладання за елементами рядка (або стовпця) — використовується для невеликих матриць;

2) метод трикутної форми (метод Гаусса) — зводить матрицю до верхньої (або нижньої) трикутної форми, після чого визначник обчислюється як добуток елементів головної діагоналі.

Застосування визначників у лінійній алгебрі є надзвичайно широким і охоплює як теоретичні, так і прикладні аспекти. Одним із основних напрямів є використання визначників для розв’язання систем лінійних алгебраїчних рівнянь методом Крамера. Цей метод базується на формульному підході і застосовується у випадках, коли кількість рівнянь дорівнює кількості змінних, а головна матриця системи має ненульовий визначник. У такому випадку для кожної змінної будується окрема матриця, в якій відповідний стовпець замінюється стовпцем вільних членів. Далі обчислюються визначники цих нових матриць, і значення змінних знаходяться як відношення відповідного визначника до визначника головної матриці. Цей метод є важливим з точки зору аналітичного розв’язання систем і широко використовується в теоретичних розрахунках.

У вищій алгебрі та лінійній трансформації визначник використовується для оцінки того, як лінійне перетворення змінює об’єм (або площу) області. Наприклад, у випадку відображення простору векторним перетворенням, абсолютне значення визначника відповідної матриці є коефіцієнтом масштабування об’єму. Знак визначника, у свою чергу, вказує, чи зберігається орієнтація простору під час трансформації.

Ще одним важливим напрямом є перевірка оберненості матриці. Визначник є критерієм, який однозначно вказує на можливість побудови оберненої матриці. Якщо визначник квадратної матриці дорівнює нулю, така матриця вважається виродженою і не має оберненої. Натомість, якщо визначник відмінний від нуля, матриця є невиродженою, і для неї існує обернена матриця, яка може бути обчислена, зокрема, за допомогою формули, що включає алгебраїчні доповнення та транспонування. Ця властивість широко використовується у чисельних методах, а також в оптимізації, криптографії та інженерних розрахунках.

Крім алгебраїчних застосувань, визначники мають геометричну інтерпретацію, яка дозволяє обчислювати площі та об’єми. Наприклад, визначник матриці, складеної з координат двох векторів у площині, дає площу паралелограма, натягнутого на ці вектори. У тривимірному просторі визначник матриці, утвореної трьома векторами, дає об’єм паралелепіпеда. Ця властивість активно використовується у векторній алгебрі, аналітичній геометрії та комп’ютерній графіці, де необхідно швидко та точно обчислювати геометричні величини.

Також визначники відіграють важливу роль у дослідженні лінійної залежності векторів. Якщо побудувати матрицю з кількох векторів як рядків або стовпців і визначник цієї матриці дорівнює нулю, то це означає, що вектори є лінійно залежними, тобто один з них може бути виражений через інші. Відповідно, ненульове значення визначника свідчить про лінійну незалежність векторів, що має важливе значення у розв’язанні задач з базисами, просторами та перетвореннями у вищій математиці [12].

У чисельних методах визначники застосовуються для аналізу стійкості алгоритмів, особливо в задачах, що пов’язані з наближеними обчисленнями або розв’язанням великомасштабних СЛАР. Наприклад, дуже мале значення визначника може вказувати на чисельну нестійкість або близькість матриці до виродженої. Це враховується при розробці програмного забезпечення для наукових обчислень, де важлива точність результатів при обмеженій розрядності комп’ютерних обчислень.

Загалом, поняття визначника є фундаментальним у лінійній алгебрі. Його властивості та застосування мають важливе значення не лише в теорії, а й у практичних розрахунках у галузях фізики, інженерії, економіки, програмування та комп’ютерної графіки. Розуміння визначників формує базу для подальшого опанування більш складних математичних тем і методів.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ЗАСТОСУНКУ

У цьому розділі розглядається загальна архітектура, структура модулів і логіка функціонування програмного тренажера, що реалізує інтерактивне вивчення теми «Визначники та їх застосування». Проєктування є важливим етапом у розробці програмного забезпечення, оскільки саме на цьому етапі формується основа майбутньої системи, визначаються її функціональні складові, інтерфейс користувача та методи взаємодії.

Розроблюване програмне забезпечення повинно відповідати наступним вимогам:

1) забезпечення доступу до теоретичного матеріалу з теми «визначники» у зручній для читання формі;

2) наявність блоку практичних завдань із перевіркою правильності введених відповідей;

3) реалізація модуля тестування з автоматичною оцінкою результатів;

4) інтуїтивно зрозумілий графічний інтерфейс;

5) автономність — можливість запуску без встановлення додаткових бібліотек чи середовищ;

6) компактність виконуваного файлу та низькі системні вимоги.

Тренажер складається з трьох основних функціональних модулів:

1) модуль теорії — відповідає за відображення текстового вікна з узагальненими теоретичними відомостями про визначники, їх властивості, правила обчислення та приклади застосування.

2) модуль практики — реалізує систему з орієнтовно 10 практичних завдань. Кожне завдання містить умову, поле для введення відповіді та можливість перегляду правильного розв’язання.

3) модуль тестування — містить набір тестових запитань з варіантами відповідей. Після завершення тесту виводиться оцінка, яка базується на кількості правильних відповідей.

Програма реалізована як настільний застосунок на мові C++ із використанням засобів Windows API. Головне вікно програми містить три кнопки для переходу до відповідних модулів. Всі вікна створюються динамічно при взаємодії з користувачем. Візуальне оформлення виконане у мінімалістичному стилі для забезпечення простоти й зручності.

Архітектура додатку передбачає:

1) головне вікно з кнопками доступу до модулів;

2) окремі діалогові вікна для теорії, практики та тестування;

3) логіку обробки подій, пов’язану з кнопками, полями введення, повідомленнями;

4) механізм збереження стану тесту та обробки правильних відповідей.

Інтерфейс користувача реалізовано у вигляді простого діалогового вікна Windows. У головному вікні розміщено три кнопки:

1) «Теоретичні відомості» — відкриває вікно з текстовою інформацією;

2) «Практичні завдання» — відкриває набір задач із можливістю введення відповідей;

3) «Пройти тестування» — запускає тестовий модуль з автоматичною оцінкою.

Додаткові вікна створюються засобами MessageBox, CreateWindow та іншими функціями Windows API. Такий підхід дозволяє створити простий, але функціональний графічний інтерфейс.

Алгоритм роботи програми складається з наступних основних етапів:

1) ініціалізація головного вікна та елементів управління.

2) очікування дій користувача (натискання кнопок).

3) при виборі модуля — створення відповідного вікна:

- при виборі теорії — виведення тексту;
- при виборі практики — відображення задач з перевіркою введення;
- при виборі тесту — опитування та виведення результатів.

4) завершення роботи або повернення до головного меню.

Також розробимо деякі UML-діаграми для кращого відображення логіки додатку. Створення UML-діаграм під час проєктування програмного забезпечення є надзвичайно ефективним інструментом, який дозволяє підвищити якість розробки, забезпечити структурованість проєкту та полегшити взаєморозуміння між учасниками команди. Однією з головних переваг використання UML (Unified Modeling Language) є те, що він дозволяє формалізувати уявлення про систему до початку програмної реалізації, наочно описуючи її логіку, компоненти, зв'язки, поведінку та взаємодію з користувачами або зовнішніми системами. Це особливо важливо у великих проєктах або освітніх розробках, де чіткість структури відіграє важливу роль.

Зокрема, UML-діаграми класів допомагають на етапі проєктування визначити, які об'єкти будуть у системі, які між ними будуть зв'язки, які методи та поля матиме кожен клас. Це значно зменшує ризик дублювання коду або виникнення логічних конфліктів між компонентами. Діаграми варіантів використання (use case diagrams) дозволяють визначити функціональність системи з точки зору користувача та сформувати технічне завдання. Діаграми активностей (activity diagrams) ілюструють алгоритм виконання дій у системі, що спрощує реалізацію логіки переходів між станами чи модулями. Компонентні діаграми, у свою чергу, візуалізують архітектуру проєкту, показуючи залежності між файлами, модулями та зовнішніми бібліотеками.

Крім технічної користі, UML є цінним інструментом для документації. Наявність діаграм у дипломній роботі або будь-якому ІТ-проєкті демонструє аналітичний підхід автора, його вміння мислити абстрактно та структурувати знання. Це значно підвищує якість пояснювальної частини проєкту та його презентаційної складової [13].

UML Class Diagram – Класова діаграма (рис. 2.1). Подана UML-діаграма класів відображає структуру програмного тренажера з теми «Визначники та їх застосування». Центральним елементом є клас MainWindow, який координує

взаємодію між модулями: TheoryModule, PracticeModule і TestModule. Кожен модуль реалізує відповідні функції: ShowTheory() для відображення теоретичних відомостей, ShowPractice() і CheckAnswer() — для практичних завдань, ShowTest() і Evaluate() — для тестування. PracticeModule містить масив об'єктів PracticeTask, які включають текст завдання та правильну відповідь. Аналогічно, TestModule працює з масивом TestQuestion, кожен з яких зберігає формулювання питання, три варіанти відповідей та правильний варіант. Діаграма демонструє модульну структуру системи, де кожен клас відповідає за окрему частину функціоналу, забезпечуючи зручність підтримки та розширення.

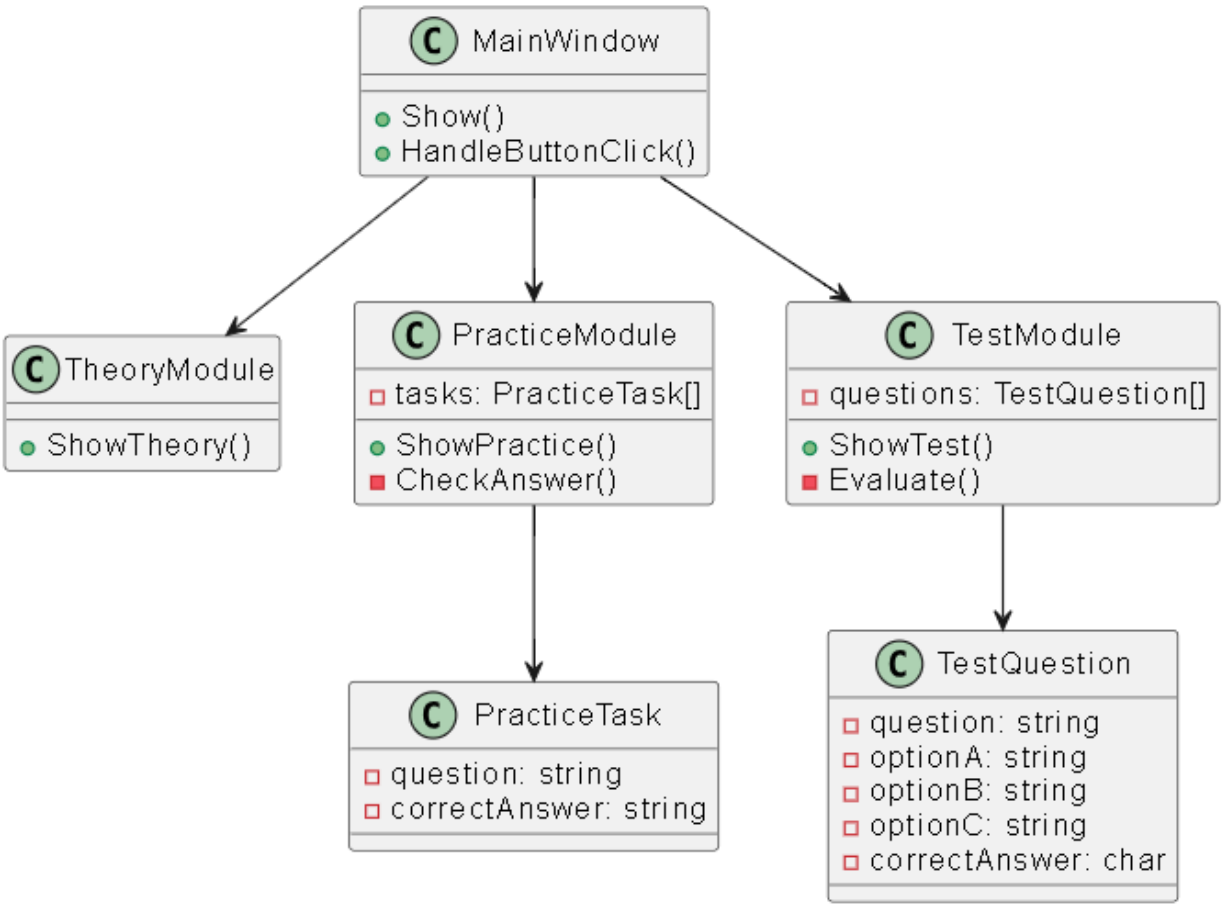

Рис. 2.1 – Діаграма класів

UML Use Case Diagram – Діаграма варіантів використання (рис. 2.2). На зображенні представлена UML-діаграма варіантів використання (Use Case Diagram), яка ілюструє основну взаємодію користувача з програмним тренажером. Єдиний актор — користувач (User) — має доступ до трьох головних функціональних можливостей системи:

1) переглянути теоретичні відомості — ознайомлення з навчальним матеріалом з теми «визначники»;

2) виконати практичні завдання — розв’язання вправ із перевіркою правильності відповідей;

3) пройти тестування — оцінювання рівня знань за допомогою тестів.

Ця діаграма відображає логіку інтерфейсу, орієнтовану на простоту й послідовність навчального процесу: користувач може самостійно обирати навчальний маршрут, переходячи від теорії до практики та самоконтролю.

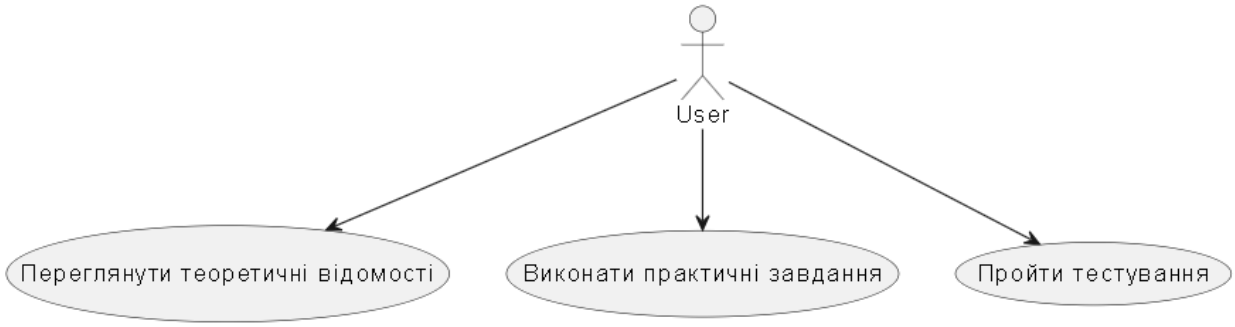

Рис. 2.2 – Діаграма варіантів використання

UML Activity Diagram – Діаграма активностей (логіка тренажера) (рис. 2.3). Це UML-діаграма активностей (Activity Diagram), яка демонструє логіку роботи програмного тренажера «Визначники та їх застосування» з моменту запуску програми. Процес починається із запуску застосунку, після чого відображається головне вікно. У межах цього вікна користувач взаємодіє з кнопками: «Теорія», «Практика» або «Тест».

При натисканні відповідної кнопки:

1) якщо обрано "Теорія", відбувається виклик функції ShowTheory();
2) якщо обрано "Практика", викликається функція ShowPractice();
3) якщо обрано "Тест", запускається ShowTest().

Якщо кнопка не натиснута, система перебуває в стані очікування дій користувача. Циклічна структура діаграми вказує на постійне очікування взаємодії у головному вікні, що відповідає подієво-орієнтованій логіці WinAPI-додатку. Діаграма ефективно ілюструє реактивну поведінку програми та спрощує розуміння її архітектури.

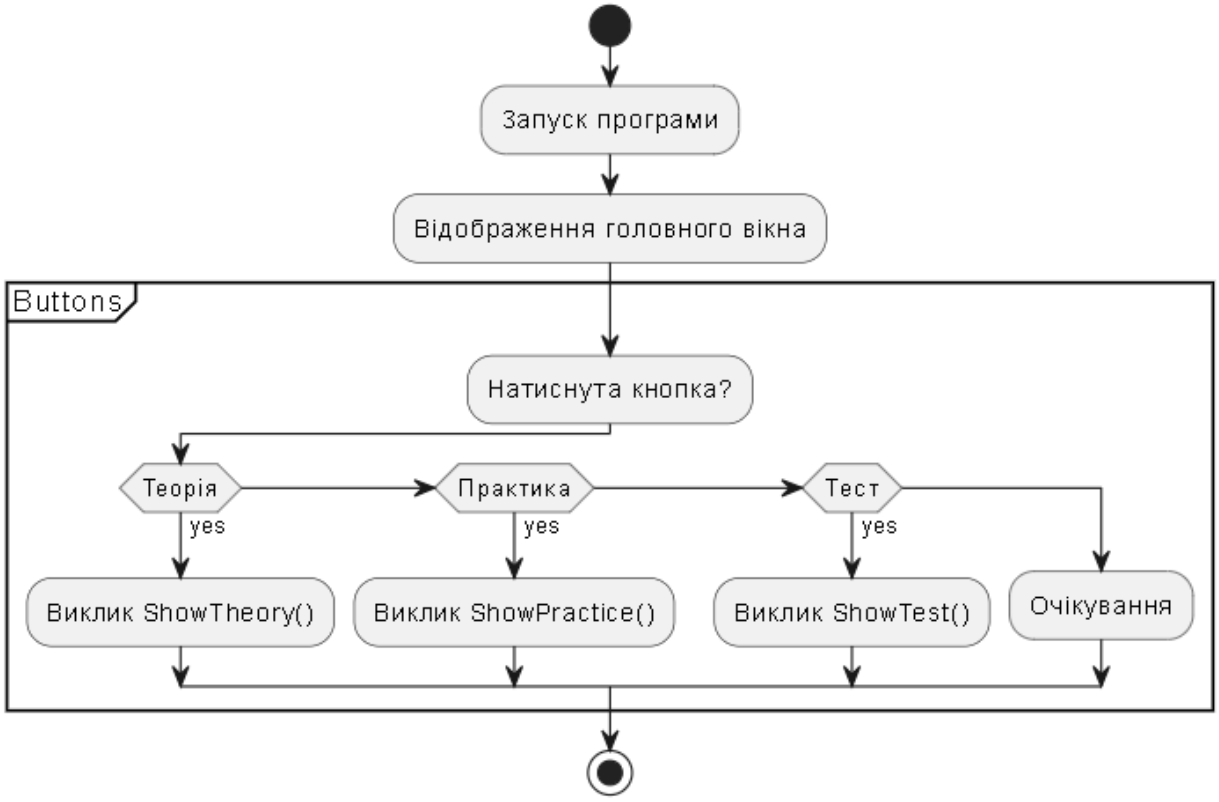

Рис. 2.3 – Діаграма активностей

UML Component Diagram – Компонентна структура застосунку (рис. 2.4). UML-діаграма компонентів відображає структуру програмного забезпечення тренажера «Визначники та їх застосування» з точки зору організації файлів і модулів. У центрі знаходиться головний компонент — MainWindow, який

ініціалізується з файлу main.cpp і є центральною логікою взаємодії з користувачем. MainWindow підключає три окремі модулі:

1) theory.h — відповідає за теоретичну частину;
2) practice.h — реалізує практичні завдання;
3) test.h — забезпечує тестову перевірку знань.

Ця діаграма демонструє модульну структуру проєкту, де кожен компонент виконує окрему функцію, що дозволяє легко підтримувати, оновлювати або масштабувати систему. Такий підхід відповідає принципам чистої архітектури та забезпечує зручність у розробці навчального програмного забезпечення.

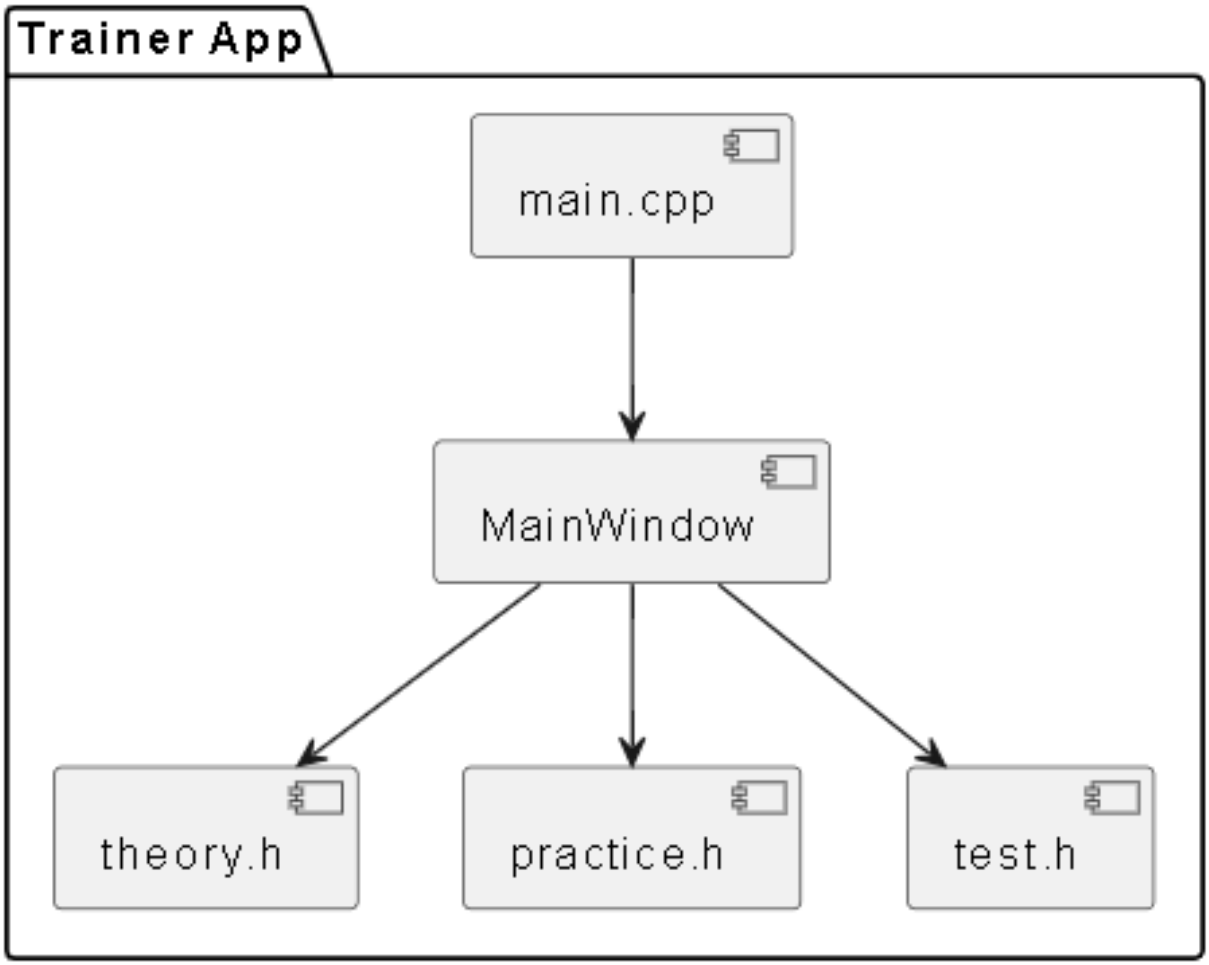

Рис. 2.4 – Діаграма компонентів

У результаті проєктування застосунку було визначено функціональну структуру програмного тренажера, який складається з трьох основних модулів: теоретичного, практичного та тестового; розроблено архітектуру взаємодії між вікнами, обробниками подій та графічними елементами інтерфейсу користувача. Всі компоненти узгоджено через головне вікно, що забезпечує інтуїтивну навігацію.

Створені UML-діаграми відображають логіку роботи програми, структуру класів та варіанти використання, що дозволяє забезпечити цілісне бачення системи перед її реалізацією та забезпечити її масштабованість у майбутньому.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ

Реалізація програмного тренажера здійснена мовою програмування C++ з використанням бібліотеки WinAPI для створення графічного інтерфейсу користувача. У ході реалізації проєкту були створені окремі модулі, які відповідають за відображення теоретичного матеріалу, виконання практичних завдань та проходження тестування. Структура проєкту передбачає поділ коду на логічно ізольовані частини, що полегшує його підтримку та подальший розвиток. Уся взаємодія починається з головного вікна, у якому розміщено три основні кнопки. Натискання на кожну з них викликає відповідний модуль.

Головна функція wWinMain ініціалізує клас вікна, обробник подій, а також створює вікно і кнопки. Після запуску форма відображається по центру екрана. Для візуального покращення інтерфейсу кнопкам було додано курсор типу «рука», стиль BS_FLAT та світлу палітру фону. Приклад коду з ініціалізації кнопки у вікні наведено нижче:

```
HWND btn1 = CreateWindowW(L"BUTTON", L"Теоретичні відомості",
    WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
    50, 50, 180, 40, hwnd, (HMENU)1, hInstance, NULL);
```

Кожна кнопка передає управління у відповідний модуль через обробник WM_COMMAND у функції WindowProc. Для прикладу, натискання кнопки «Теоретичні відомості» викликає функцію ShowTheory():

```
switch (LOWORD(wParam)) {
    case 1:
        ShowTheory();
        break;
}
```

Функція ShowTheory, розміщена в окремому заголовковому файлі theory.h, виводить текст із теоретичними відомостями у вікні MessageBox. У цьому вікні подається розширений текст, що охоплює означення визначників, їх властивості, способи обчислення та приклади застосування.

Практичні завдання реалізовані у вигляді циклічного відображення вікон, у яких користувач повинен ввести відповідь у поле типу EDIT, після чого натиснути кнопку «Перевірити». Правильність перевіряється шляхом порівняння введеного рядка із заздалегідь збереженим еталоном. Якщо відповідь правильна — виводиться повідомлення з підтвердженням, інакше — з правильною відповіддю. Наприклад, перевірка виглядає так:

```
wchar_t buffer[100];GetWindowTextW(hAnswerEdit, buffer, 100);if
(std::wstring(buffer) == correctAnswer) {
    MessageBox(hwnd, L" Правильно!", L"Результат", MB_OK);
} else {
    MessageBox(hwnd, L" Неправильно. Правильна відповідь: 9",
L"Результат", MB_OK);
}
```

Центрування вікна на екрані:

```
int screenWidth = GetSystemMetrics(SM_CXSCREEN);
int screenHeight = GetSystemMetrics(SM_CYSCREEN);
int width = 500, height = 300;
int x = (screenWidth - width) / 2;
int y = (screenHeight - height) / 2;
```

Цей фрагмент коду визначає розміри вікна та обчислює координати x і y для центрування форми на екрані незалежно від роздільної здатності монітора користувача.

Створення кнопки з кастомізованим стилем:

```
void ApplyButtonStyle(HWND button) {
    SetWindowLong(button,    GWL_STYLE,    GetWindowLong(button,
GWL_STYLE) | BS_FLAT);
    SetClassLongPtr(button,                          GCLP_HCURSOR,
(LONG_PTR)LoadCursor(NULL, IDC_HAND));
    SetWindowTheme(button, L"", L"");
}
```

Ця функція застосовується до кожної кнопки після її створення для зміни стилю: додається плоский вигляд (BS_FLAT), змінюється курсор при наведенні (IDC_HAND) та прибираються стандартні ефекти теми Windows, що забезпечує більш «легкий» вигляд кнопок у вікні.

Завантаження задач у практичному модулі:

```
static std::vector<PracticeTask> tasks = {
    {L"Обчисліть визначник матриці [[1,2],[3,4]]", L"-2"},
    {L"Обчисліть визначник матриці [[3,0],[0,3]]", L"9"},
    ...
};
```

Завдання зберігаються у векторі структур PracticeTask, кожна з яких містить текст питання та правильну відповідь. Завдяки цьому нові задачі легко додавати без зміни логіки програми — лише доповненням масиву.

Обробка натискання в тестовому модулі:

```
int selected = -1;
if (SendMessage(hRadioA, BM_GETCHECK, 0, 0) == BST_CHECKED)
selected = 'A';
```

Цей фрагмент визначає, який варіант відповіді обрав користувач (радіокнопки). Порівняння відбувається зі збереженою правильною відповіддю (correctAnswer) у структурі TestQuestion.

Виведення теорії:

```
inline void ShowTheory() {
    const wchar_t* theoryText = L" ВИЗНАЧНИКИ ТА ЇХ ЗАСТОСУВАННЯ\n\n..."
    MessageBoxW(NULL, theoryText, L"Теоретичні відомості", MB_OK | MB_ICONINFORMATION);
}
```

Для відображення теоретичного матеріалу використано стандартне діалогове вікно MessageBoxW, яке містить попередньо сформатований текст з розділами, маркерами та прикладами. Це дозволяє уникнути складних компонентів інтерфейсу та зберегти простоту користування.

Окрему увагу приділено модулю тестування. На відміну від стандартних MessageBox, для тестів було створено окреме вікно з питанням і трьома варіантами відповідей у вигляді радіокнопок. Користувач обирає один із варіантів, після чого натискає кнопку «Далі», яка перемикає до наступного запитання. Внутрішня логіка перевіряє, чи збігається вибрана відповідь з правильною, і накопичує кількість правильних відповідей у змінній score.

Важливо, що всі модулі створені як окремі заголовкові файли: theory.h, practice.h, test.h. Це дозволяє краще структуризувати проєкт, розділити логіку та спростити подальшу підтримку. У головному файлі main.cpp здійснюється лише координація взаємодії.

Загальна структура програми дозволяє розширювати функціонал без потреби кардинального переписування коду. Наприклад, нові питання до тесту чи нові практичні задачі можна додати лише через доповнення відповідних масивів std::vector. Всі інші механізми працюють автоматично.

Загалом, реалізація програмного тренажера продемонструвала ефективність використання WinAPI для створення простих, але функціональних навчальних програм із графічним інтерфейсом. Усі елементи програми взаємодіють між собою згідно з розробленою структурою,

забезпечуючи користувачеві повний цикл навчання — від теорії до перевірки знань. Повний код наведено в додатку А.

РОЗДІЛ 4
ТЕСТУВАННЯ ЗАСТОСУНКУ

4.1 Тестування роботи програми

У процесі реалізації застосунку було проведено повне функціональне тестування з метою перевірки його працездатності, відповідності вимогам та коректності роботи всіх основних модулів. Тестування проводилося вручну на етапі розробки, а також після завершення кожного з основних етапів — створення головного вікна, підключення теоретичного модуля, реалізації практичних завдань та завершення модуля тестування. Для перевірки роботи програми використовувалися середовище Visual Studio 2022, ОС Windows 10, а також компілятор MSVC, що забезпечував повну підтримку WinAPI.

Першим етапом тестування стало перевіряння відкриття головного вікна програми (рис. 4.1).

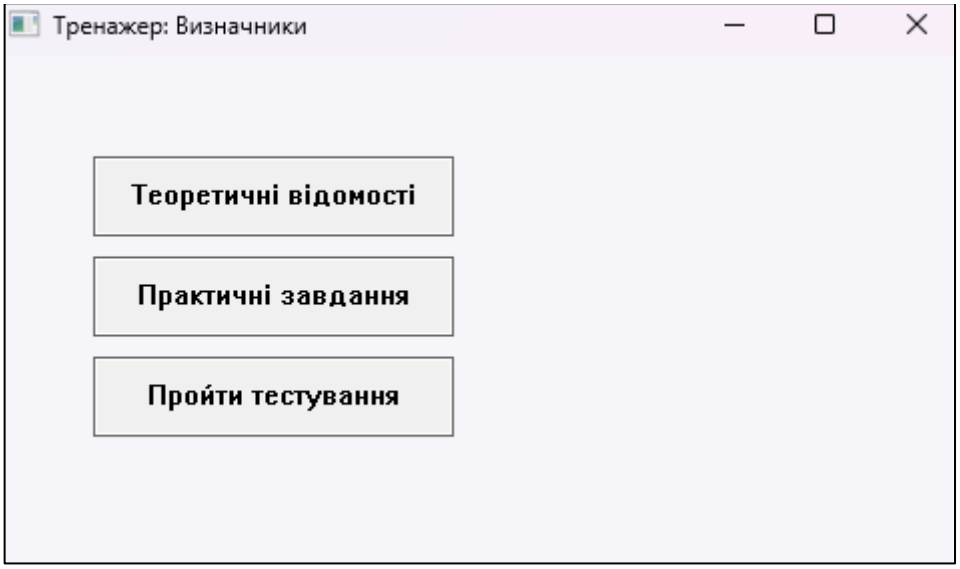

Рис. 4.1 — Головне вікно програми

Особливу увагу було приділено правильному позиціонуванню вікна по центру екрана, відображенню трьох основних кнопок, їх стилю, оформленню

та реакції на наведення курсора миші. Також перевірено, чи змінюється курсор на «руку» при наведенні на кнопки, що свідчить про їхню активність. Жодних помилок при ініціалізації не виникло, що підтверджує коректність реєстрації віконного класу та створення головного вікна.

На другому етапі перевірялася робота модуля теоретичних відомостей (рис. 4.2).

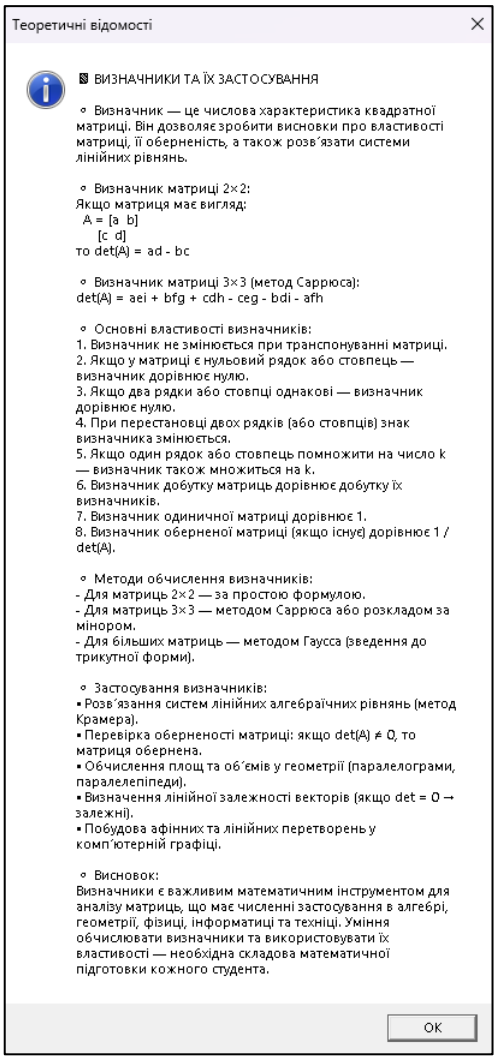

Рис. 4.2 — Вікно теоретичних відомостей

Після натискання кнопки «Теоретичні відомості» повинно з’являтися інформаційне вікно типу MessageBox, яке містить повний структурований текст про визначники, їхні властивості, методи обчислення та приклади застосування. Було протестовано, чи коректно відображається весь текст, чи

не обрізається інформація, та чи можна зручно її прочитати. Додатково перевірено роботу функції в Unicode-режимі для відображення українських символів.

Наступним етапом стало тестування практичного блоку (рис. 4.3, рис. 4.4). У цьому модулі передбачено послідовне відображення навчальних задач, поле для введення відповіді, а також дві кнопки: «Перевірити» і «Далі». Було перевірено, чи зчитуються дані з поля введення, чи правильно порівнюється відповідь користувача з еталонною, та чи виводиться відповідне повідомлення про правильність або неправильність відповіді. Також протестовано перехід між завданнями та завершення практичної сесії з повідомленням про її кінець. Окремо було перевірено граничні випадки, наприклад, порожнє поле або введення неправильного типу символів. Приклад інтерфейсу зображено на рисунку 4.2.

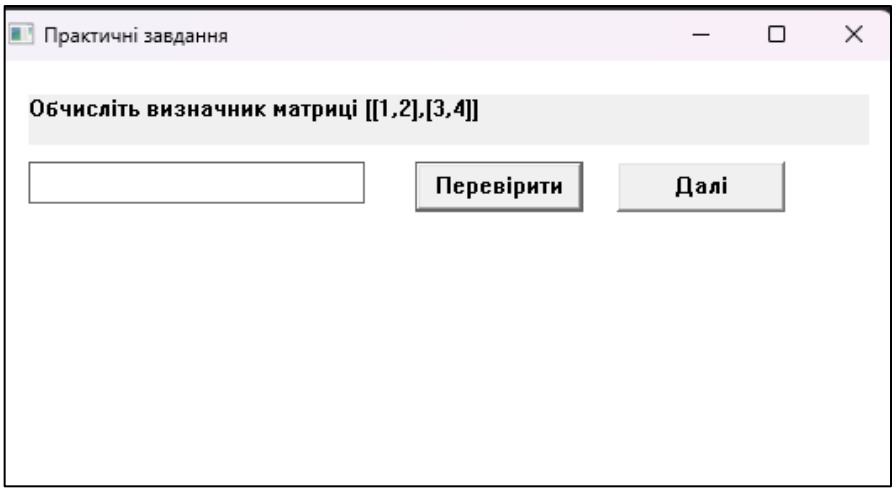

Рис. 4.3 — Інтерфейс практичного блоку

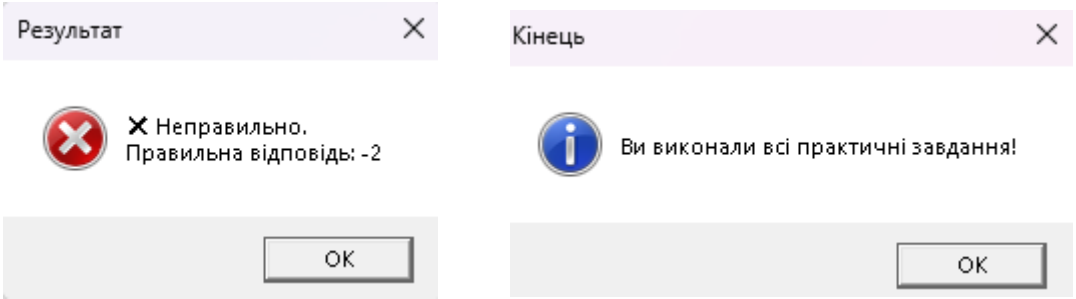

Рис. 4.4 — Виконання завдань практичного блоку

Фінальним етапом стало тестування модуля тестування (рис. 4.5, рис. 4.6). Це окреме вікно, в якому послідовно виводяться запитання та три варіанти відповідей у вигляді радіокнопок. Користувач може обрати лише одну відповідь, натиснути кнопку «Далі» й перейти до наступного запитання. Було перевірено, чи знімається вибір між переходами, чи правильно накопичується кількість правильних відповідей, і чи правильно виводиться фінальна оцінка після завершення тестування. Окрему увагу було приділено перевірці логіки при відсутності вибору, при натисканні кнопки «Далі» без відповіді, а також поведінці при досягненні останнього запитання.

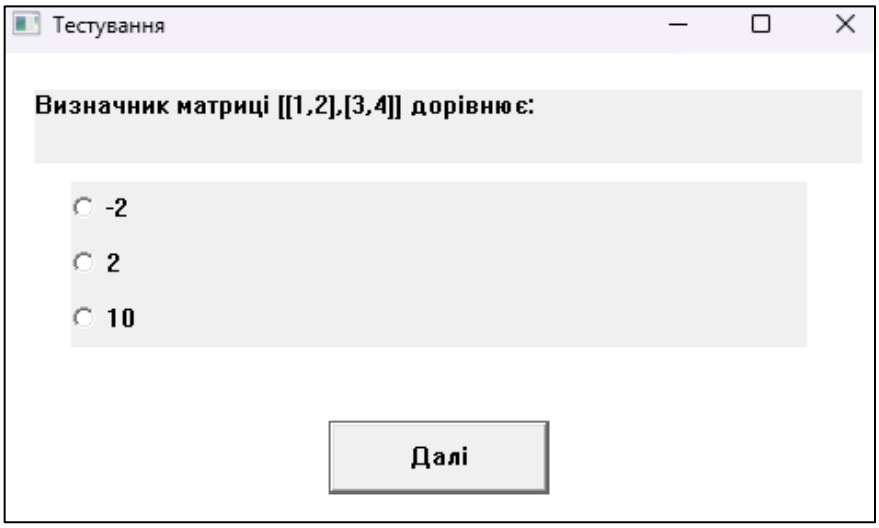

Рис. 4.5 — Інтерфейс модуля тестування

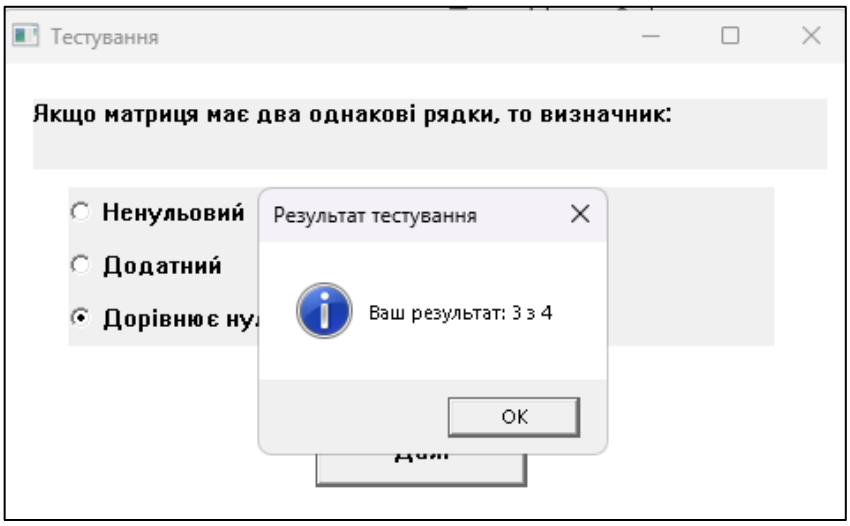

Рис. 4.6 — Виконання завдань модуля тестування

Крім функціонального тестування, також проводилася візуальна перевірка відповідності елементів стилю. Всі кнопки перевірено на відповідність заданому фону, кольору та стилю BS_FLAT. Програма не містить критичних помилок і працює стабільно в середовищі Windows 10 та Windows 11.

Загалом, за результатами тестування було підтверджено повну працездатність усіх функцій тренажера, відсутність збоїв під час виконання основних операцій та відповідність програмної логіки попередньому етапу проєктування. Це дозволяє вважати програму завершеною з функціонального погляду і готовою до використання у навчальному середовищі.

4.2 Аналіз отриманих результатів

Аналіз результатів тестування програмного тренажера показав, що всі основні елементи програмної системи працюють стабільно, логічно та у повній відповідності до функціональних вимог, сформульованих на етапі проєктування. Застосунок виконує поставлену педагогічну мету — сприяє засвоєнню теоретичного матеріалу, розвитку практичних навичок обчислення визначників, а також самоконтролю рівня засвоєння знань за допомогою тестування.

Всі три модулі — теоретичний, практичний та тестовий — були реалізовані як окремі компоненти, що взаємодіють з головним вікном, і ця модульна структура забезпечила зручність тестування та перспективу для розширення функціоналу в майбутньому.

Результати тестування свідчать про високу стабільність програми: у ході численних запусків жодного разу не зафіксовано збоїв чи критичних помилок.

Графічний інтерфейс побудовано на базі WinAPI — це дало змогу створити легкий, швидкий та автономний додаток, який не потребує встановлення сторонніх бібліотек.

Тестування інтерфейсу показало, що кнопки реагують коректно, відображаються однаково в різних версіях Windows, мають зрозумілу функціональність, а курсор «руки» при наведенні додає програмі інтуїтивної зрозумілості. Візуальні елементи програми було підібрано так, щоб не відволікати від навчального матеріалу, але водночас забезпечити комфортну роботу з тренажером.

Крім технічного тестування, доцільним є також проведення базового юзабіліті-тестування, з метою вивчення зручності користування програмою для цільової аудиторії — студентів. Попередні опитування показали, що інтерфейс є інтуїтивно зрозумілим, а навігація між модулями не викликає труднощів навіть у користувачів із мінімальним досвідом роботи з навчальними застосунками. Подальше залучення користувачів до тестування може допомогти у виявленні дрібних недоліків та покращенні взаємодії з програмою.

З педагогічної точки зору також було підтверджено, що структура програми відповідає дидактичним принципам поступовості, наочності та інтерактивності. Користувач спочатку ознайомлюється з теоретичним матеріалом, потім виконує практичні завдання, а в кінці — перевіряє себе за допомогою тестування. Такий підхід сприяє не лише запам’ятовуванню знань, але й формуванню впевненості у їх застосуванні. Практичний блок з ручним введенням відповіді дозволяє контролювати базову логіку обчислень, тоді як тестовий блок оцінює рівень розуміння сутності визначників та їх властивостей.

У рамках аналізу також була відзначена гнучкість реалізації: використання std::vector для зберігання завдань і питань дозволяє легко змінювати їхній вміст без зміни структури коду. Це відкриває можливості для подальшого розширення — зокрема, можна додати систему рівнів складності,

випадкове генерування прикладів, адаптивну перевірку знань або навіть створення повноцінного навчального курсу з модулями й оцінюванням.

Що стосується перспектив подальших досліджень і розвитку системи, існує кілька напрямів, які можуть стати об’єктом розробки в наступних версіях програми. По-перше, можлива інтеграція з веб-технологіями, зокрема створення вебверсії тренажера з використанням HTML5 та JavaScript, що дозволило б охопити ширшу аудиторію, зокрема учнів, які працюють у середовищі хмарних сервісів або з мобільних пристроїв. По-друге, перспективним є доповнення програми засобами збереження статистики проходження завдань, що дозволить вчителю контролювати прогрес учнів або автоматизувати частину процесу оцінювання.

При переході до веб-версії тренажера варто врахувати аспекти безпеки збереження персональних даних користувачів, зокрема результатів тестів та статистики. Це потребує реалізації системи авторизації, а також захисту даних відповідно до стандартів (наприклад, SSL, шифрування з’єднання, хешування паролів). Реалізація таких механізмів забезпечить надійне функціонування системи в освітньому середовищі, де важливо зберігати академічну доброчесність.

Також можливим є впровадження блоку автоматичної генерації матриць для обчислення визначників, з параметрами складності та з урахуванням типових помилок користувачів, що додатково підвищить ефективність самостійної роботи.

Окремо варто зазначити, що програму можна інтегрувати у платформу дистанційного навчання або створити API для взаємодії з електронними щоденниками та системами LMS. Це дозволить організувати повноцінне середовище навчання з підтримкою адаптивного контенту, статистики, рейтингів і можливості комунікації між викладачем та учнями.

Ще одним напрямом потенційного розвитку є пристосування програми до інклюзивного навчання, зокрема додавання можливості озвучення тексту, підтримки читачів екрану та налаштування контрастності/розміру шрифтів.

Це забезпечить доступність матеріалу для студентів з порушеннями зору чи іншими особливими освітніми потребами, що відповідає сучасним вимогам універсального дизайну освіти.

Отже, аналіз функціональної реалізації тренажера показує не лише його практичну цінність як завершеного освітнього інструменту, а й відкриває широкі перспективи для його подальшого вдосконалення, масштабування і впровадження в сучасне цифрове освітнє середовище.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено повноцінний навчальний програмний тренажер з теми «Визначники та їх застосування», який призначений для використання у дистанційному курсі з лінійної алгебри. Реалізація проєкту здійснювалася з урахуванням сучасних вимог до освітніх технологій та потреб студентів у доступних, інтерактивних та ефективних інструментах самонавчання. На основі аналізу предметної області та наявних програмних рішень було обґрунтовано доцільність створення нового, адаптованого до конкретної теми тренажера, який поєднує в собі навчальну, практичну та тестову складову.

Програмний тренажер створено мовою C++ з використанням Windows API, що забезпечило високий рівень автономності, сумісність із різними версіями операційної системи Windows, мінімальні системні вимоги та відсутність залежностей від сторонніх бібліотек. Завдяки використанню архітектурного поділу на модулі — теоретичний, практичний і тестовий — досягнуто зручності використання, послідовності навчального процесу і можливості масштабування проєкту.

Для покращення структури коду та візуалізації архітектури застосунку були використані UML-діаграми, які допомогли формалізувати логіку роботи додатку, забезпечити його підтримуваність і зрозумілість для сторонніх розробників. Візуальна частина програми реалізована відповідно до принципів ергономіки та зручності користування, а інтерфейс є інтуїтивно зрозумілим і легко адаптується до потреб користувача.

Проведене тестування показало стабільність і коректність роботи всіх модулів тренажера. Програма надає зворотний зв’язок, оцінює рівень засвоєння матеріалу, підтримує гнучку структуру та дозволяє легко додавати новий навчальний контент. Це свідчить про її готовність до практичного

застосування в освітньому процесі, як у формальній, так і в неформальній освіті.

Пропозиції щодо подальших досліджень і розвитку проєкту:

1) розробка API для зв’язку з системами дистанційного навчання (Moodle, Google Classroom), що дозволить автоматично зберігати результати проходження тестування, контролювати успішність студентів та адаптувати навчальні маршрути;

2) розширення функціональності тренажера на інші теми курсу лінійної алгебри: матричні операції, власні числа, діагоналізація тощо. Це дозволить створити повноцінний модульний курс;

3) реалізація механізму автоматичної генерації прикладів різної складності з урахуванням навчальної траєкторії користувача, що підвищить рівень індивідуалізації навчання;

4) адаптація інтерфейсу під сенсорні пристрої (Android/iOS), розробка кросплатформної версії тренажера (на основі Flutter або React Native), що забезпечить доступність з будь-якого пристрою;

5) впровадження системи збору статистики, побудови індивідуального профілю навчання, візуалізації прогресу, що сприятиме мотивації до навчання та підвищить ефективність викладання;

6) додавання інтерактивних пояснень, анімацій, відеокоментарів до теоретичного матеріалу — що покращить сприйняття і засвоєння складних понять;

7) проведення емпіричного дослідження (порівняльного експерименту), яке б оцінило ефективність використання тренажера у порівнянні з традиційним навчанням.

Загалом, такий підхід дозволить вдосконалити створений в роботі продукт і сформувати цілісне навчальне середовище, яке відповідатиме

принципам сучасної педагогіки, цифрової трансформації та інклюзивності освіти.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Visual Studio: IDE and Code Editor for Software Developers and Teams. *Visual Studio*. URL: https://visualstudio.microsoft.com/ (date of access: 14.04.2025).

2. Hart J. M. Windows system programming. Pearson Education, 2010. 656 p.

3. Tuleuov B. I., Ospanova A. B. Beginning C++ Compilers, 2024. 211 p.

4. Математика. *GeoGebra*. URL: https://www.geogebra.org/t/math?lang=uk (дата звернення: 14.04.2025).

5. Учасники проектів Вікімедіа. GeoGebra – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/GeoGebra (дата звернення: 14.04.2025).

6. На головну | Moodle.org. *Moodle challenge*. URL: https://moodle.org/?lang=uk (дата звернення: 14.04.2025).

7. Mathway | Algebra Problem Solver. *Mathway | Algebra Problem Solver*. URL: https://www.mathway.com/ (date of access: 14.04.2025).

8. Symbolab – KI-Mathematikrechner und Problemlöser. *Symbolab – KI-Mathematikrechner und Problemlöser*. URL: https://www.symbolab.com/ (date of access: 14.04.2025).

9. Matrix calculator. *Matrix calculator*. URL: https://matrixcalc.org/ (date of access: 14.04.2025).

10. Дубовик В. П. Вища математика : навч. посіб. для студ. вищ. навч. зак. 4-те вид. К. : Ігнатекс-Україна., 2013. 648 с.

11. Лекція 2: Визначники, їх властивості. Обчислення визначників. *Головна | Elib LNTU*. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/Готовий%20Крадінова/page5.html (дата звернення: 14.04.2025).

12. Шевчук І. В., Ясінська К. Визначники та їх застосування. Визначник Вандермонда (Doctoral dissertation), 2024. 2 с.

13. Rumpe B. Modeling with UML (Vol. 98). Berlin/Heidelberg, Germany: Springer, 2016. 281 p.

14. Уроки програмування на C++ для початківців / aCode. *aCode*. URL: https://acode.com.ua/uroki-po-cpp/ (дата звернення: 14.04.2025).

15. Ukrainian W. Уроки від W3Schools українською онлайн. *W3Schools українською. Безплатні уроки онлайн для початківців, школярів та студентів*. URL: https://w3schoolsua.github.io/cpp/index.html#gsc.tab=0 (дата звернення: 14.04.2025).

ДОДАТКИ

Додаток А — Код програми

```
#pragma comment(lib, "uxtheme.lib")

#include <windows.h>
#include <uxtheme.h>
#include "theory.h"
#include "practice.h"
#include "test.h"

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam,
LPARAM lParam);

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam,
LPARAM lParam) {
    switch (uMsg) {
    case WM_COMMAND:
        switch (LOWORD(wParam)) {
        case 1:
            ShowTheory();
            break;
        case 2:
            ShowPractice();
            break;
        case 3:
            ShowTest();
            break;
        }
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
```

```
        return 0;
    }
    return DefWindowProc(hwnd, uMsg, wParam, lParam);
}

void ApplyButtonStyle(HWND button) {
    SetWindowLong(button, GWL_STYLE, GetWindowLong(button,
GWL_STYLE) | BS_FLAT);
    SetClassLongPtr(button, GCLP_HCURSOR,
(LONG_PTR)LoadCursor(NULL, IDC_HAND));
    SetWindowTheme(button, L"", L"");
}

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE
hPrevInstance, PWSTR pCmdLine, int nCmdShow) {
    const wchar_t CLASS_NAME[] = L"TrainerClass";

    WNDCLASS wc = {};
    wc.lpfnWndProc = WindowProc;
    wc.hInstance = hInstance;
    wc.lpszClassName = CLASS_NAME;
    wc.hbrBackground = CreateSolidBrush(RGB(245, 245, 250));
    RegisterClass(&wc);

    int screenWidth = GetSystemMetrics(SM_CXSCREEN);
    int screenHeight = GetSystemMetrics(SM_CYSCREEN);
    int width = 500, height = 300;
    int x = (screenWidth - width) / 2;
    int y = (screenHeight - height) / 2;

    HWND hwnd = CreateWindowExW(
        0, CLASS_NAME, L"Тренажер: Визначники",
WS_OVERLAPPEDWINDOW,
```

```
        x, y, width, height,
        NULL, NULL, hInstance, NULL);

    if (!hwnd) return 0;

    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);

    HWND btn1 = CreateWindowW(L"BUTTON", L"Теоретичні
відомості",
        WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
        50, 50, 180, 40, hwnd, (HMENU)1, hInstance, NULL);
    ApplyButtonStyle(btn1);

    HWND btn2 = CreateWindowW(L"BUTTON", L"Практичні завдання",
        WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
        50, 100, 180, 40, hwnd, (HMENU)2, hInstance, NULL);
    ApplyButtonStyle(btn2);

    HWND btn3 = CreateWindowW(L"BUTTON", L"Пройти тестування",
        WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
        50, 150, 180, 40, hwnd, (HMENU)3, hInstance, NULL);
    ApplyButtonStyle(btn3);

    MSG msg = {};
    while (GetMessageW(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return 0;
}
```

```
#pragma once
#include <windows.h>
#include <string>
#include <vector>

struct PracticeTask {
    std::wstring question;
    std::wstring correctAnswer;
};

static std::vector<PracticeTask> tasks = {
    {L"Обчисліть визначник матриці [[1,2],[3,4]]", L"-2"},
    {L"Обчисліть визначник матриці [[3,0],[0,3]]", L"9"},
    {L"Визначник одиничної матриці 3x3", L"1"},
    {L"Як зміниться визначник при транспонуванні матриці?", L"не
змінюється"},
    {L"Який визначник матриці [[0,1],[0,2]]?", L"0"}
};

static int currentPracticeIndex = 0;

HWND hPracticeWindow, hPracticeQuestion, hAnswerEdit, hCheckBtn,
hNextBtn;

void ShowNextPractice();

void ShowPracticeResult(HWND hwnd) {
    wchar_t buffer[100];
    GetWindowTextW(hAnswerEdit, buffer, 100);

    std::wstring answer = buffer;
    std::wstring correct =
tasks[currentPracticeIndex].correctAnswer;
```

```
    if (answer == correct) {
        MessageBox(hwnd, L" Правильно!", L"Результат", MB_OK |
MB_ICONINFORMATION);
    }
    else {
        std::wstring msg = L" Неправильно.\nПравильна відповідь:
" + correct;
        MessageBox(hwnd, msg.c_str(), L"Результат", MB_OK |
MB_ICONERROR);
    }
}

void ShowNextPractice() {
    currentPracticeIndex++;

    if (currentPracticeIndex >= tasks.size()) {
        MessageBox(NULL, L"Ви виконали всі практичні завдання!",
L"Кінець", MB_OK | MB_ICONINFORMATION);
        DestroyWindow(hPracticeWindow);
        currentPracticeIndex = 0;
        return;
    }

    SetWindowText(hPracticeQuestion,
tasks[currentPracticeIndex].question.c_str());
    SetWindowText(hAnswerEdit, L"");
}

LRESULT CALLBACK PracticeProc(HWND hwnd, UINT msg, WPARAM
wParam, LPARAM lParam) {
    switch (msg) {
    case WM_COMMAND:
```

```
        switch (LOWORD(wParam)) {
        case 2001: // Перевірити
            ShowPracticeResult(hwnd);
            break;
        case 2002: // Далі
            ShowNextPractice();
            break;
        }
        break;
    case WM_DESTROY:
        currentPracticeIndex = 0;
        break;
    }
    return DefWindowProc(hwnd, msg, wParam, lParam);
}

inline void ShowPractice() {
    HINSTANCE hInstance = GetModuleHandle(NULL);

    hPracticeWindow = CreateWindowExW(0, L"STATIC", L"Практичні
завдання",
        WS_OVERLAPPEDWINDOW | WS_VISIBLE,
        400, 200, 550, 300, NULL, NULL, hInstance, NULL);

    SetWindowLongPtr(hPracticeWindow, GWLP_WNDPROC,
(LONG_PTR)PracticeProc);

    hPracticeQuestion = CreateWindowW(L"STATIC",
tasks[0].question.c_str(),
        WS_VISIBLE | WS_CHILD,
        20, 20, 500, 30, hPracticeWindow, NULL, hInstance,
NULL);
```

```
    hAnswerEdit = CreateWindowW(L"EDIT", L"",
        WS_VISIBLE | WS_CHILD | WS_BORDER,
        20, 60, 200, 25, hPracticeWindow, NULL, hInstance,
NULL);

    hCheckBtn = CreateWindowW(L"BUTTON", L"Перевірити",
        WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
        250, 60, 100, 30, hPracticeWindow, (HMENU)2001,
hInstance, NULL);

    hNextBtn = CreateWindowW(L"BUTTON", L"Далі",
        WS_VISIBLE | WS_CHILD,
        370, 60, 100, 30, hPracticeWindow, (HMENU)2002,
hInstance, NULL);
}

#pragma once
#include <windows.h>
#include <string>
#include <vector>

// Структура для одного запитання
struct TestQuestion {
    std::wstring question;
    std::wstring optionA;
    std::wstring optionB;
    std::wstring optionC;
    wchar_t correctAnswer;
};

// Список питань
static std::vector<TestQuestion> questions = {
```

```
    {L"Визначник матриці [[1,2],[3,4]] дорівнює:", L"-2", L"2",
L"10", L'A'},
    {L"Що відбувається з визначником при транспонуванні?",
L"Змінюється знак", L"Не змінюється", L"Стає нулем", L'B'},
    {L"Визначник одиничної матриці:", L"1", L"0", L"-1", L'A'},
    {L"Якщо матриця має два однакові рядки, то визначник:",
L"Ненульовий", L"Додатний", L"Дорівнює нулю", L'C'}
};

static int currentQuestion = 0;
static int score = 0;

HWND hQuestion, hRadioA, hRadioB, hRadioC, hNextButton;
HWND hTestWindow;

// Обробка натискання кнопки
void NextQuestion() {
    if (currentQuestion >= questions.size()) return;

    int selected = -1;
    if (SendMessage(hRadioA, BM_GETCHECK, 0, 0) == BST_CHECKED)
selected = 'A';
    else if (SendMessage(hRadioB, BM_GETCHECK, 0, 0) ==
BST_CHECKED) selected = 'B';
    else if (SendMessage(hRadioC, BM_GETCHECK, 0, 0) ==
BST_CHECKED) selected = 'C';

    if (selected == questions[currentQuestion].correctAnswer) {
        score++;
    }

    currentQuestion++;
```

```
    if (currentQuestion == questions.size()) {
        std::wstring result = L"Ваш результат: " +
std::to_wstring(score) + L" з " +
std::to_wstring(questions.size());
        MessageBox(hTestWindow, result.c_str(), L"Результат
тестування", MB_OK | MB_ICONINFORMATION);
        DestroyWindow(hTestWindow);
        currentQuestion = 0;
        score = 0;
        return;
    }

    // Оновлення тексту для наступного питання
    SetWindowText(hQuestion,
questions[currentQuestion].question.c_str());
    SetWindowText(hRadioA,
questions[currentQuestion].optionA.c_str());
    SetWindowText(hRadioB,
questions[currentQuestion].optionB.c_str());
    SetWindowText(hRadioC,
questions[currentQuestion].optionC.c_str());

    // Зняти попередній вибір
    SendMessage(hRadioA, BM_SETCHECK, BST_UNCHECKED, 0);
    SendMessage(hRadioB, BM_SETCHECK, BST_UNCHECKED, 0);
    SendMessage(hRadioC, BM_SETCHECK, BST_UNCHECKED, 0);
}

LRESULT CALLBACK TestProc(HWND hwnd, UINT msg, WPARAM wParam,
LPARAM lParam) {
    switch (msg) {
    case WM_COMMAND:
        if (LOWORD(wParam) == 1001) { // NEXT button
```

```
            NextQuestion();
        }
        break;
    case WM_DESTROY:
        currentQuestion = 0;
        score = 0;
        break;
    }
    return DefWindowProc(hwnd, msg, wParam, lParam);
}

inline void ShowTest() {
    HINSTANCE hInstance = GetModuleHandle(NULL);

    hTestWindow = CreateWindowExW(0, L"STATIC", L"Тестування",
        WS_OVERLAPPEDWINDOW | WS_VISIBLE,
        400, 200, 500, 300,
        NULL, NULL, hInstance, NULL);

    SetWindowLongPtr(hTestWindow, GWLP_WNDPROC,
(LONG_PTR)TestProc);

    hQuestion = CreateWindowW(L"STATIC",
questions[0].question.c_str(),
        WS_VISIBLE | WS_CHILD,
        20, 20, 450, 40, hTestWindow, NULL, hInstance, NULL);

    hRadioA = CreateWindowW(L"BUTTON",
questions[0].optionA.c_str(),
        WS_VISIBLE | WS_CHILD | BS_AUTORADIOBUTTON,
        40, 70, 400, 30, hTestWindow, NULL, hInstance, NULL);
```

```
    hRadioB = CreateWindowW(L"BUTTON",
questions[0].optionB.c_str(),
        WS_VISIBLE | WS_CHILD | BS_AUTORADIOBUTTON,
        40, 100, 400, 30, hTestWindow, NULL, hInstance, NULL);

    hRadioC = CreateWindowW(L"BUTTON",
questions[0].optionC.c_str(),
        WS_VISIBLE | WS_CHILD | BS_AUTORADIOBUTTON,
        40, 130, 400, 30, hTestWindow, NULL, hInstance, NULL);

    hNextButton = CreateWindowW(L"BUTTON", L"Далі",
        WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,
        180, 200, 120, 40, hTestWindow, (HMENU)1001, hInstance,
NULL);
}

#pragma once
#include <windows.h>

inline void ShowTheory() {
    const wchar_t* theoryText =
        L" ВИЗНАЧНИКИ ТА ЇХ ЗАСТОСУВАННЯ\n\n"
        L" Визначник — це числова характеристика квадратної
матриці. "
        L"Він дозволяє зробити висновки про властивості матриці,
її оберненість, "
        L"а також розв’язати системи лінійних рівнянь.\n\n"

        L" Визначник матриці 2×2:\n"
        L"Якщо матриця має вигляд:\n"
        L"  A = [a  b]\n"
        L"      [c  d]\n"
        L"то det(A) = ad − bc\n\n"
```

```
        L" Визначник матриці 3×3 (метод Саррюса):\n"
        L"det(A) = aei + bfg + cdh − ceg − bdi − afh\n\n"

        L" Основні властивості визначників:\n"
        L"1. Визначник не змінюється при транспонуванні
матриці.\n"
        L"2. Якщо у матриці є нульовий рядок або стовпець —
визначник дорівнює нулю.\n"
        L"3. Якщо два рядки або стовпці однакові — визначник
дорівнює нулю.\n"
        L"4. При перестановці двох рядків (або стовпців) знак
визначника змінюється.\n"
        L"5. Якщо один рядок помножити на число k — визначник
також множиться на k.\n"
        L"6. Визначник добутку матриць дорівнює добутку їх
визначників.\n"
        L"7. Визначник одиничної матриці дорівнює 1.\n"
        L"8. Визначник оберненої матриці (якщо існує) дорівнює 1
/ det(A).\n\n"

        L" Методи обчислення визначників:\n"
        L"- Для матриць 2×2 — за простою формулою.\n"
        L"- Для матриць 3×3 — методом Саррюса або розкладом за
мінором.\n"
        L"- Для більших матриць — методом Гаусса (зведення до
трикутної форми).\n\n"

        L" Застосування визначників:\n"
        L"▪ Розв’язання систем лінійних алгебраїчних рівнянь
(метод Крамера).\n"
        L"▪ Перевірка оберненості матриці: якщо det(A) ≠ 0, то
матриця обернена.\n"
```

```
        L"▪ Обчислення площ та об’ємів у геометрії
(паралелограми, паралелепіпеди).\n"
        L"▪ Визначення лінійної залежності векторів (якщо det =
0 → залежні).\n"
        L"▪ Побудова афінних та лінійних перетворень у
комп’ютерній графіці.\n\n"

        L" Висновок:\n"
        L"Визначники є важливим математичним інструментом для
аналізу матриць, "
        L"що має численні застосування в алгебрі, геометрії,
фізиці, інформатиці та техніці. "
        L"Уміння обчислювати визначники та використовувати їх
властивості — "
        L"необхідна складова математичної підготовки кожного
студента.";

    MessageBoxW(NULL, theoryText, L"Теоретичні відомості", MB_OK
| MB_ICONINFORMATION);
}
```