

ДЕРЖАВНИЙ УНІВЕРСИТЕТ  
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ  
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ  
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

**КВАЛІФІКАЦІЙНА РОБОТА**

на тему: «Додаток для розв'язку задач векторного числення на мовах програмування JavaScript, PHP з використанням MySQL, Adobe Photoshop»

на здобуття освітнього ступеня магістра  
зі спеціальності F3 Комп'ютерні науки  
(код, найменування спеціальності)  
освітньо-професійної програми Комп'ютерні науки  
(назва)

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

\_\_\_\_\_  
(підпис)

Андрій ДОЛІНСЬКИЙ  
(Ім'я, ПРИЗВИЩЕ здобувача)

Виконав:  
здобувач вищої освіти  
група КНДМ-62

Андрій ДОЛІНСЬКИЙ

Керівник:  
науковий ступінь,  
вчене звання

Олена ШИКУЛА  
д.ф.-м.н., професор

Рецензент:  
науковий ступінь,  
вчене звання

\_\_\_\_\_  
(Ім'я, ПРИЗВИЩЕ)

Київ 2025

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ  
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**  
**Навчально-науковий інститут інформаційних технологій**

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність F3 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

**ЗАТВЕРДЖУЮ**

Завідувач кафедру Комп'ютерних наук

\_\_\_\_\_ Віктор ВИШНІВСЬКИЙ  
« \_\_\_\_\_ » \_\_\_\_\_ 2025 р.

**ЗАВДАННЯ  
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Долінському Андрію Андрійовичу

*(прізвище, ім'я, по батькові здобувача)*

1. Тема кваліфікаційної роботи: Додаток для розв'язку задач векторного числення на мовах програмування JavaScript, PHP з використанням MySQL, Adobe Photoshop

керівник кваліфікаційної роботи Олена ШИКУЛА д.ф.-м.н., професор,

*(Ім'я, ПРИЗВИЩЕ науковий ступінь, вчене звання)*

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467

2. Строк подання кваліфікаційної роботи «26» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, інструменти для розробки клієнтської частини (HTML, CSS, JavaScript, JQuery, AJAX, Bootstrap, Adobe Photoshop) та інструменти для розробки серверної частини (PHP, OpenServer, MySQL) додатку.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Вивчення призначення та класифікації програмних засобів комп'ютерної математики, огляд наявних програмних засобів, визначення їх функціоналу, переваг та недоліків.

Розробка додатку для розв'язку задач векторного числення.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз роботи системи комп'ютерної алгебри Maxima
2. Структурна схема додатку та його структура.
3. Екранні форми додатку для розв'язку задач векторного числення.
4. Фрагменти програмного коду.

6. Дата видачі завдання «15» вересня 2025 р.

### КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи                         | Строк виконання етапів роботи | Примітка |
|-------|-------------------------------------------------------------|-------------------------------|----------|
| 1     | Аналіз наявної науково-технічної літератури                 | 15.09-27.09.25                | виконано |
| 2     | Огляд існуючих систем комп'ютерної математики               | 28.09-11.10.25                | виконано |
| 3     | Визначення функціоналу та основних характеристик додатка    | 12.10-25.10.25                | виконано |
| 4     | Вибір інструментів для розробки клієнтської частини додатку | 26.10-04.11.25                | виконано |
| 5     | Вибір інструментів для розробки серверної частини додатку   | 05.11-12.11.25                | виконано |
| 6     | Розробка додатку для розв'язку задач векторного числення    | 12.11-02.12.25                | виконано |
| 7     | Оформлення роботи: вступ, висновки, реферат                 | 02.12-08.12.25                | виконано |
| 8     | Розробка демонстраційних матеріалів                         | 09.12-13.12.25                | виконано |

Здобувач вищої освіти

\_\_\_\_\_  
(підпис)

Андрій ДОЛІНСЬКИЙ

(Ім'я, ПРІЗВИЩЕ)

Керівник  
кваліфікаційної роботи

\_\_\_\_\_  
(підпис)

Олена ШИКУЛА

(Ім'я, ПРІЗВИЩЕ)





## РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 83 стор., 47 рис., 45 джерел.

*Наукове завдання* – вивчення призначення та класифікації програмних засобів комп'ютерної математики, огляд наявних програмних засобів, визначення їх функціоналу, переваг та недоліків, підбір інструментів для розробки клієнтської та серверної частин додатку; розробка додатку для розв'язування задач з векторного числення зі зручним інтуїтивно-зрозумілим інтерфейсом.

*Мета роботи* – створення програмного додатку, що забезпечуватиме розв'язування задач з векторного числення і сприятиме кращому засвоєнню матеріалу учнями та студентами.

*Об'єкт дослідження* – процес створення додатку для розв'язку задач з векторного числення.

*Предмет дослідження* – додаток для розв'язку задач з векторного числення.

*Короткий зміст роботи:*

Вивчено призначення та класифікація програмних засобів комп'ютерної математики, розглянуто наявні програмні засоби, визначено їх функціонал, переваги та недоліки.

За допомогою мов програмування JavaScript, PHP з використанням СКБД MySQL, Adobe Photoshop було розроблено додаток для розв'язування задач векторного числення із зручним та зрозумілим користувачеві інтерфейсом. У додатку користувач зможе ознайомитися з теоретичними відомостями про вектори, засвоїти правила виконання дій з ними, а також перевірити отримані знання на практиці.

Ключові слова: додаток, задачі векторного числення, JavaScript, PHP, СКБД MySQL, Adobe Photoshop

## ABSTRACT

Text part of the master's qualification work: 83 pages, 47 pictures, 45 sources.

*The scientific task* is to study of the purpose and classification of software tools for computer mathematics, review of existing software tools, determination of their functionality, advantages and disadvantages, selection of tools for development of client and server parts of the application; development of an application for solving vector calculus problems with a convenient intuitive and understandable interface.

*The purpose of the work* is to create a software application that will provide solving vector calculus problems and contribute to better assimilation of the material by pupils and students.

*The object of research* is the process of creating an application for solving vector calculus problems.

*The subject of research* is an application for solving vector calculus problems.

*Summary of the work:* The purpose and classification of software tools for computer mathematics are studied, the existing software tools are considered, their functionality, advantages and disadvantages are determined. Using the programming languages JavaScript, PHP using the MySQL database, Adobe Photoshop, an application was developed for solving vector calculus problems with a convenient and user-friendly interface. In the application, the user will be able to familiarize themselves with theoretical information about vectors, learn the rules for performing actions with them, and also test the acquired knowledge in practice.

Keywords: testing systems, application, vector calculus problems, JavaScript, PHP, MySQL database, Adobe Photoshop

## ЗМІСТ

|                                                                               |     |
|-------------------------------------------------------------------------------|-----|
| ВСТУП.....                                                                    | 9   |
| 1 ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ.....                                     | 11  |
| 1.1 Програмні засоби комп'ютерної математики.....                             | 11  |
| 1.2 Призначення програмних засобів комп'ютерної математики.....               | 14  |
| 1.2.1 Недоліки комп'ютерних обчислень.....                                    | 15  |
| 1.2.2 Класифікація програмних засобів комп'ютерної математики.....            | 17  |
| 1.2.3 Призначення систем комп'ютерної алгебри.....                            | 20  |
| 1.3 Огляд програмних засобів комп'ютерної математики.....                     | 21  |
| 1.4 Аналіз роботи вільної системи комп'ютерної алгебри Maxima.....            | 24  |
| 2.1 Векторний метод та його застосування.....                                 | 32  |
| 2.2. Інструменти для розробки клієнтської частини додатку.....                | 50  |
| 2.2.1 Мова розмітки гіпертексту HTML.....                                     | 50  |
| 2.2.2 Каскадні таблиці стилів CSS.....                                        | 51  |
| 2.2.3 Динамічна мова програмування JavaScript та бібліотека JQuery.....       | 53  |
| 2.2.4 Технологія взаємодії з сервером без перезавантаження сторінки AJAX..... | 56  |
| 2.2.5 Фреймворк Bootstrap.....                                                | 60  |
| 2.2.6 Додаток Adobe Photoshop.....                                            | 62  |
| 2.3 Інструменти для розробки серверної частини додатку.....                   | 65  |
| 2.3.1 Мова програмування PHP.....                                             | 65  |
| 2.3.2 Сервер OpenServer.....                                                  | 67  |
| 2.3.3 СКБД MySQL.....                                                         | 68  |
| 2.3.4 Система управління контентом BlizAnt.....                               | 69  |
| 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ.....                                              | 72  |
| 3.1 Елементи керування додатку.....                                           | 72  |
| 3.2 Архітектура додатку.....                                                  | 73  |
| 3.3 Екранні форми додатку та його робота.....                                 | 76  |
| ВИСНОВКИ.....                                                                 | 87  |
| ПЕРЕЛІК ПОСИЛАНЬ.....                                                         | 88  |
| ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ.....                                    | 92  |
| ДОДАТОК Б. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація).....                        | 103 |

## ВСТУП

Розв'язування геометричних задач часто викликає значні труднощі у школярів. Це пояснюється насамперед тим, що більшість завдань потребує виконання кількох логічних кроків, які не завжди можна відразу побачити чи передбачити. У процесі роботи над задачею учням доводиться шукати приховані зв'язки між елементами фігур, робити додаткові побудови або спиратися на вже відомі властивості геометричних об'єктів. Саме тому розв'язок далеко не завжди лежить на поверхні.

Досить часто одна й та сама задача може мати кілька шляхів розв'язання. Це надає геометрії особливого шарму й демонструє багатство її методів. Серед них вирізняють різні підходи, серед яких особливе місце займають синтетичний та векторний методи. Синтетичний метод передбачає використання інтуїтивних припущень, творчих здогадок та додаткових побудов, що дозволяють виявити приховані властивості фігур. Його витонченість полягає у вмінні поєднати логіку з уявою, а також у красі самих побудов. На відміну від нього, векторний метод не вимагає таких інтуїтивних рішень, адже ґрунтується на чітких правилах векторної алгебри.

Багато геометричних задач підпорядковуються певним алгоритмам розв'язування, що значно полегшує пошук правильної відповіді. Використання векторної алгебри особливо корисне у випадках, коли необхідно довести теореми чи розв'язати задачі, які потребують складних просторових міркувань. Завдяки цьому складні логічні конструкції та геометричні доведення часто зводяться до відносно нескладних алгебраїчних обчислень. Такий підхід не лише спрощує процес, а й допомагає учням глибше зрозуміти сутність задачі, поєднуючи красу геометрії з точністю алгебри.

Більшість завдань з векторного числення, які часто створюють значні труднощі для школярів і студентів, підпорядковуються певному алгоритму, що відкриває можливість розробити спеціальний програмний додаток, призначений для виконання вправ з векторного числення. Використання такого додатку

дозволить у позаурочний та позааудиторний час самостійно вдосконалювати вміння та навички роботи з векторами. Крім того, цей інструмент стане у пригоді викладачам, оскільки може слугувати наочним матеріалом для пояснення теми. З огляду на це, проведене дослідження має беззаперечну *актуальність*.

*Об'єкт дослідження* – процес створення додатку для розв'язку задач з векторного числення.

*Предмет дослідження* – додаток для розв'язку задач з векторного числення.

*Метою дослідження* є створення програмного додатку, що забезпечуватиме розв'язування задач з векторного числення і сприятиме кращому засвоєнню матеріалу учнями та студентами. Передбачається, що у додатку користувач зможе ознайомитися з теоретичними відомостями про вектори, засвоїти правила виконання дій з ними, а також перевірити отримані знання на практиці. Завдяки зручному інтерфейсу та інтерактивним елементам такий додаток може слугувати ефективним посібником для опанування та закріплення навичок роботи з векторами.

Для досягнення мети роботи були поставлені наступні *завдання*:

- вивчити призначення та класифікацію програмних засобів комп'ютерної математики, розглянути наявні програмні засоби, визначити їх функціонал, переваги та недоліки.
- підібрати інструменти для розробки клієнтської та серверної частин додатку;
- розробити додаток для розв'язування задач з векторного числення зі зручним інтуїтивно-зрозумілим інтерфейсом, що сприятиме кращому засвоєнню матеріалу учнями та студентами.

*Практичне значення* додатку полягає в тому, що він може використовуватися як учнями у позашкільний час для опанування матеріалом та вдосконалення набутих навичок, так і вчителями для більш наочного пояснення нового матеріалу учням.

## 1 ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ

### 1.1 Програмні засоби комп'ютерної математики

Відповідно до тлумачень В. П. Д'яконова та Ю. В. Триуса, поняття комп'ютерна математика можна розглядати як комплекс теоретичних основ, методичних підходів, алгоритмічних прийомів, а також апаратних і програмних засобів, призначених для ефективного розв'язування за допомогою комп'ютера широкого спектра математичних задач. Важливою особливістю цього напрямку є високий рівень візуалізації кожного етапу обчислювального процесу [2, 11].

На сучасному етапі розвитку науки й освіти все більшого поширення набувають інструменти комп'ютерної математики, серед яких чільне місце займають програмні рішення. Їх прийнято називати програмними засобами комп'ютерної математики (ПЗКМ).

Програмні засоби комп'ютерної математики – це спеціалізовані програмні продукти, які дозволяють автоматизувати виконання найрізноманітніших математичних операцій: від чисельних розрахунків до аналітичних перетворень і побудови графічних моделей. У цих засобах утілено багатовіковий досвід розвитку математичної науки, що робить їх надзвичайно потужними інструментами для практичного використання. Завдяки ПЗКМ користувач отримує можливість розв'язувати навіть складні математичні завдання, які раніше вимагали значних інтелектуальних і часових затрат [8].

Важливість автоматизації не лише чисельних, а й аналітичних обчислень була усвідомлена ще на початку 60-х років XX століття академіком В. М. Глушковим. Саме під його керівництвом у Києві були створені перші у світі зразки персональних комп'ютерів, точніше – їхні прообрази, серії «Мир». Ці машини мали апаратну підтримку мов програмування високого рівня та надавали унікальні можливості для виконання як чисельних, так і аналітичних розрахунків.

Важливою особливістю системи була структурна інтерпретація мов високого рівня МИР і АНАЛІТИК, що дозволяла ефективно працювати з дійсними числами довільної точності, цілими числами без обмежень розрядності, а також виконувати точні операції з дробовими раціональними числами. Створена під керівництвом

Глушкова система АНАЛІТИК стала однією з перших у світі систем комп'ютерної алгебри. Ба більше, саме в мові АНАЛІТИК уперше було застосовано метод переписування алгебраїчних виразів (тобто використання співвідношень), який у сучасних умовах є основою декларативного програмування [9]. За швидкістю виконання аналітичних перетворень ці системи не мали аналогів у світі.

На жаль, інноваційні розробки академіка В. М. Глушкова не отримали належної підтримки в колишньому Радянському Союзі. Унаслідок цього подальший розвиток і лідерство в галузі комп'ютерної алгебри поступово перейшли до науковців і компаній західних країн.

Сьогодні провідні програмні засоби комп'ютерної математики розробляються та підтримуються насамперед великими західними компаніями, такими як MathSoft, MathWorks, Waterloo Maple та інші. Ці продукти перетворилися на потужні інструменти не лише для професійних математиків, а й для широкого кола користувачів, які застосовують математичні методи з метою побудови та дослідження математичних моделей у різноманітних галузях науки й техніки, зокрема й у сфері освіти [8].

Завдяки таким програмам стає можливим ефективне розв'язування як наукових і технічних завдань, так і навчальних задач різного рівня складності. Вони забезпечують зручну візуалізацію даних, дозволяють представити результати обчислень у наочній формі, а також виконують роль розгорнутих довідників із сучасних методів математичних обчислень. Важливою перевагою цих засобів є інтеграція потужної графіки, елементів візуального програмування та мультимедійних технологій, що значно розширює їхні можливості та виводить застосування ПЗКМ далеко за межі традиційних математичних розрахунків.

В освітньому процесі програмні засоби комп'ютерної математики дедалі активніше використовуються як універсальні інструменти. Вони допомагають викладачам створювати електронні уроки, мультимедійні курси лекцій і навіть електронні підручники з інтерактивними прикладами. Такий формат дає змогу учням і студентам не лише спостерігати готові розв'язки, а й самостійно змінювати вихідні умови та проводити власні навчальні дослідження.

Завдяки появі ПЗКМ значно зросли можливості професійних математиків і спеціалістів інших галузей, що активно застосовують математичні методи у своїй діяльності. Використання таких програмних продуктів дозволяє підвищити інтенсивність та ефективність інтелектуальної праці, забезпечуючи автоматизацію не лише чисельних, але й аналітичних (символьних) обчислень, а також графічних побудов.

Сфера аналітичних можливостей ПЗКМ надзвичайно широка: вони дають змогу знаходити границі та похідні функцій, обчислювати визначені й невизначені інтеграли, здійснювати розкладання функцій у ряди, розв'язувати різні класи диференціальних рівнянь у символічному вигляді. Крім того, системи цього типу дозволяють виконувати широкий спектр алгебраїчних операцій – від спрощення та перетворення виразів до підстановок і роботи з довільними алгебраїчними структурами.

На думку В. П. Д'яконова [2], системи комп'ютерної математики можна класифікувати за сімома основними класами: до них належать системи, орієнтовані на виконання чисельних обчислень; табличні процесори; матричні системи; програмні комплекси для статистичних розрахунків; системи для спеціалізованих обчислень; програмні продукти для аналітичних перетворень (системи комп'ютерної алгебри); а також універсальні системи, які поєднують у собі функції кількох попередніх категорій.

Протягом останніх десятиліть було створено значну кількість математичних пакетів, серед яких можна виокремити як спеціалізовані (наприклад, Eureka, MacMath, StatGraph, Reduse, MacSyma, SketchPad, Cabri тощо), так і універсальні (зокрема, Derive, MathCad, MathLab, Maple, Mathematica, MuPad) [1, 2, 3, 7, 11]. Усі вони відзначаються зручними інтерфейсами та широким спектром функцій: реалізацією великої кількості стандартних і спеціальних математичних операцій, наявністю потужних інструментів для дво- та тривимірної графіки, використанням власних мов програмування, засобами для підготовки математичних текстів до друку, можливістю експорту даних в інші програмні середовища й імпорту інформації з них для подальшої обробки. Такі характеристики роблять математичні

пакети надзвичайно корисними для ефективної діяльності фахівців у найрізноманітніших галузях.

Системи комп'ютерної математики набули широкого поширення в освітньому середовищі в усьому світі. Наприклад, за інформацією з офіційного сайту розробників системи Mathematica, тисячі університетів із 61 країни є офіційними користувачами цього програмного продукту. Серед них такі престижні навчальні заклади, як Пекінський університет, Кембриджський університет, Колумбійський університет, Гарвардський університет, Стенфордський університет, Московський державний університет, Австралійський національний університет, Каліфорнійський університет, Оксфордський університет, а також Лондонська школа економіки та політичних наук і багато інших.

Додатково, у результатах досліджень, проведених норвезькими науковцями [12], підкреслюється, що використання систем комп'ютерної математики в освітньому процесі є вже сформованою практикою, яка демонструє високу ефективність. Дослідники наголошують, що впровадження подібних систем повинно стати одним із пріоритетних завдань для технічної освіти. Зокрема, ними запропоновано такі рекомендації: забезпечення єдиного програмного середовища в межах одного навчального закладу; побудова математичних курсів з урахуванням можливостей обраної системи; створення спеціалізованих комп'ютерних лабораторій, що надають змогу максимально ефективно використовувати програмні засоби комп'ютерної математики.

## **1.2 Призначення програмних засобів комп'ютерної математики**

Програмні засоби комп'ютерної математики – це спеціалізоване програмне забезпечення, яке дає змогу виконувати на комп'ютері не лише числові обчислення, але й здійснювати аналітичні (символьні) перетворення різноманітних математичних об'єктів, а також використовувати інструменти графічної візуалізації результатів.

До цього класу належать як комерційні (пропріетарні) продукти – Maple, Mathematica, MATLAB, MathCad та інші, так і системи з відкритим вихідним

кодом. Вони активно застосовуються в наукових дослідженнях і поступово стають невід'ємним елементом комп'ютерних технологій, які широко використовуються у сфері освіти.

Подібні програмні комплекси відзначаються дружнім інтерфейсом, наявністю значного набору стандартних і спеціалізованих математичних операцій, потужними засобами дво- і тривимірної графіки, а також власними мовами програмування. Таке поєднання характеристик забезпечує широкі можливості для ефективної роботи фахівців різних галузей.

Найвідоміші математичні пакети, зокрема Maple, MATLAB та Mathematica, дозволяють користувачеві виконувати як символьні обчислення, так і чисельні методи аналізу. Сьогодні ці системи розглядаються як основні інструменти комп'ютерного моделювання в реальному часі й активно використовуються у різних напрямках науки та техніки. Використання програмних засобів комп'ютерної математики істотно підвищує ефективність праці науковців, викладачів і вчителів, надаючи їм можливість значно прискорити обчислювальні процеси й зосередитися на аналітичній частині дослідження.

Особливе місце серед зазначених засобів посідає комп'ютерна алгебра – галузь математики, що поєднує методи класичної алгебри з обчислювальними технологіями та орієнтована на символьне опрацювання математичних об'єктів.

### **1.2.1 Недоліки комп'ютерних обчислень**

Перші програмні системи комп'ютерної математики, серед яких Eureka, Mercury, Excel, Lotus-123, Mathcad для MS-DOS, PC MATLAB та інші, створювалися передусім для виконання чисельних розрахунків. Фактично вони перетворювали персональний комп'ютер на потужний програмований калькулятор, здатний із високою швидкістю й точністю автоматично здійснювати арифметичні та логічні операції над окремими числами або масивами числових даних відповідно до заданої програми. Підсумком таких операцій завжди був конкретний результат: певне число чи сукупність чисел, які виводилися на екран або у вигляді акуратно оформлених таблиць та графіків.

Разом з тим, отримані результати не завжди були абсолютно точними у строгому математичному сенсі. Це пояснюється тим, що під час обчислень із дійсними числами неминуче відбувається округлення, пов'язане з обмеженням розрядної сітки комп'ютера для представлення чисел у пам'яті. Крім того, більшість чисельних методів ґрунтується на наближених алгоритмах, які за певних умов можуть втрачати обчислювальну стійкість. Унаслідок накопичення похибок результати могли спотворюватися або навіть призводити до повного збою системи – аж до її «зависання».

Саме через ці особливості чисельні математичні системи та програмні реалізації чисельних методів зазнавали значної критики з боку науковців. Основним недоліком вважався їх суто конкретний характер: такі системи не дозволяли отримувати загальних формул чи виразів, що описували б сам процес обчислення. Відповідно, користувач був обмежений у можливості робити глибокі теоретичні узагальнення, а практичні висновки часто потребували додаткового підтвердження.

У результаті для виконання серйозних наукових досліджень доводилося залучати математиків-аналітиків, які знаходили аналітичні розв'язки задач і пропонували методи їх подальшої чисельної реалізації на комп'ютері. Такий підхід, однак, вимагав значних ресурсів – як фінансових, так і часових, що робило його не надто ефективним. Саме ці проблеми стимулювали пошук нових рішень і, зрештою, призвели до появи програмних засобів комп'ютерної математики (ПЗКМ), які поєднали чисельні та аналітичні можливості в межах єдиної системи.

Термін «комп'ютерна алгебра» використовується як синонім до таких понять, як «символьні обчислення», «аналітичні обчислення», «аналітичні перетворення» та інших подібних термінів. У французькій мові навіть сьогодні він має буквально значення – «формальні обчислення».

Коли йдеться про обчислювальні методи, зазвичай вважається, що всі обчислення здійснюються у множині дійсних чисел. Проте на практиці будь-яка програма, що виконується на електронно-обчислювальній машині, оперує лише з обмеженим набором раціональних чисел, оскільки тільки такі значення можуть

бути закодовані та збережені у комп'ютері. Для представлення цілих чисел у пам'яті зазвичай відводиться 16 або 32 біти, а для дійсних чисел – 32 або 64 біти. Такий обмежений набір чисел не є замкнутим відносно арифметичних операцій, що призводить до появи переповнень та інших технічних проблем.

Ще важливішою особливістю обчислювальної математики є те, що операції додавання, віднімання, множення чи ділення, які виконує комп'ютер, не повністю збігаються з аналогічними операціями над математичними раціональними числами. Однією з головних характеристик комп'ютерних обчислень є неминуча наявність похибки та обмеженої точності. Кожну задачу потрібно розв'язати з використанням наявних ресурсів електронно-обчислювальної машини за прийнятний проміжок часу і з наперед визначеним рівнем точності. Саме тому оцінка похибок є одним із ключових завдань обчислювальної математики.

Частково ці труднощі вдалося подолати завдяки розвитку систем комп'ютерної алгебри. Такі системи, що реалізують аналітичні обчислення, активно працюють із множиною раціональних чисел, а також із алгебраїчними числами, які задаються за допомогою многочленів. У сучасних наукових дослідженнях та інженерно-технічних розрахунках фахівці значно частіше мають справу не з безпосередніми числовими обчисленнями, а з перетвореннями складних математичних виразів і формул.

Перші спроби здійснення символьних перетворень для прикладних задач датуються ще 1950-ми роками. Проте активна розробка повноцінних систем комп'ютерної алгебри розпочалася лише наприкінці 1960-х років. Відтоді було створено велику кількість різноманітних програмних систем. Деякі з них продовжують успішно розвиватися й нині, інші поступово зникли з наукової практики, а паралельно постійно з'являються нові рішення, які відповідають сучасним вимогам науки та техніки.

### **1.2.2 Класифікація програмних засобів комп'ютерної математики**

На сучасному етапі розвитку науки та техніки програмні засоби комп'ютерної математики можна класифікувати на сім основних груп:

1. Системи для чисельних розрахунків – призначені для проведення точних обчислень за допомогою числових методів.

2. Табличні процесори – програми, які дозволяють опрацьовувати дані у вигляді таблиць, виконувати обчислення, будувати графіки та діаграми.

3. Матричні системи – орієнтовані на роботу з багатовимірними масивами даних і матрицями, що особливо важливо для задач лінійної алгебри та обробки сигналів.

4. Системи для статистичних розрахунків – забезпечують аналіз великих масивів даних, побудову моделей та виявлення закономірностей.

5. Системи для спеціалізованих розрахунків – створюються для виконання вузькопрофільних задач у певних наукових або технічних галузях.

6. Системи для аналітичних розрахунків (системи комп'ютерної алгебри) – призначені для виконання символьних перетворень, спрощення виразів, обчислення похідних, інтегралів та інших операцій у символьному вигляді.

7. Універсальні системи – поєднують у собі можливості кількох вищезгаданих класів і забезпечують найбільш широкий спектр застосування.

Кожен програмний засіб комп'ютерної математики має свої індивідуальні особливості побудови, однак для універсальних систем можна виділити низку спільних структурних рис:

- Ядро системи – це набір заздалегідь відкомпільованих вбудованих функцій, процедур і операторів, які забезпечують основну роботу програми.
- Інтерфейс користувача – надає можливість взаємодії з ядром, формулювання запитів і отримання результатів у зручній формі на екрані комп'ютера.
- Бібліотеки процедур і функцій – містять додаткові математичні засоби, що використовуються рідше, але дають змогу значно розширити можливості системи без перевантаження її ядра. Такий підхід спрямований на підтримку максимальної швидкості роботи базових операцій.
- Пакети розширення – спеціальні модулі, які створюються для підвищення функціональності системи або її пристосування до конкретних завдань. Ці

пакети зазвичай пишуться мовою програмування, що вбудована у систему, тому їх можуть створювати навіть досвідчені користувачі.

- Довідкова система – розвинений інформаційний блок, який містить пояснення щодо використання функцій і можливостей програми.

Таким чином, ядро, додаткові бібліотеки, пакети розширення та довідкова система сучасних програмних засобів комп'ютерної математики акумулюють у собі величезний обсяг знань з математики, що формувалися й удосконалювалися протягом багатьох століть.

Зростання інтересу до алгоритмів алгебри пояснюється усвідомленням того, що саме алгоритми відіграють центральну роль у сучасній інформатиці. Вони зручні тим, що їх можна описати чіткою, формальною та строгою мовою, що дозволяє забезпечити правильне розв'язання поставлених завдань.

Якщо традиційна алгебра здебільшого має справу з конструктивними методами, то комп'ютерна алгебра розглядає значно ширший спектр питань. Вона цікавиться не лише правильністю застосування методів, але й їх ефективністю, можливостями практичної реалізації, а також апаратними і програмними аспектами використання алгоритмів.

Під час оцінювання ефективності та визначення продуктивності алгебраїчних методів виникає потреба у залученні додаткових математичних засобів. Серед них можна назвати теорію рекурсивних функцій, математичну логіку, аналіз, комбінаторику та інші галузі, які допомагають сформувати комплексне уявлення про роботу алгоритмів.

Ще на початкових етапах використання обчислювальних машин у галузі символічної алгебри стало очевидно, що методи, які традиційно пропонувалися у підручниках, часто виявлялися надзвичайно неефективними для практичного застосування. Саме тому комп'ютерна алгебра взяла на себе завдання систематично досліджувати джерела такої неефективності, а також розробляти нові підходи та методи, які здатні значно підвищити продуктивність обчислень або навіть повністю замінити застарілі алгоритми.

### 1.2.3 Призначення систем комп'ютерної алгебри

Перші електронно-обчислювальні машини створювалися передусім для виконання складних математичних розрахунків, які у людини займали надто багато часу та зусиль. Їхня поява дозволила значно прискорити процес обчислень і забезпечила новий рівень точності. Наступним етапом розвитку обчислювальної техніки стали персональні комп'ютери, які відкрили широкі можливості використання ЕОМ у різних сферах людської діяльності. Вони здатні виконувати обчислення різного рівня складності – від елементарних арифметичних операцій до надзвичайно складних аналітичних задач, що зробило їх незамінними у науці, техніці, освіті та бізнесі.

Розвиток комп'ютерних математичних систем зумовив появу окремого класу програмного забезпечення, який отримав назву *системи комп'ютерної алгебри* (Computer Algebra Systems, CAS). Основним завданням таких систем є опрацювання математичних виразів у символьній формі.

До найважливіших символьних операцій, які реалізують CAS, належать:

- обчислення символьних і числових значень виразів;
- спрощення та перетворення виразів;
- знаходження похідних за однією або кількома змінними;
- розв'язування лінійних і нелінійних рівнянь;
- розв'язування диференціальних рівнянь;
- обчислення границь та інтегралів;
- робота з множинами та матрицями.

Окрім символьних перетворень, більшість систем комп'ютерної алгебри підтримують і чисельні методи. Зокрема, вони дозволяють обчислювати значення виразів при заданих значеннях змінних, будувати графіки функцій на площині й у просторі, а також виконувати візуалізацію результатів.

Важливою складовою CAS є вбудовані високорівневі мови програмування, що надають користувачам можливість створювати власні алгоритми для розв'язання специфічних завдань. Наукова дисципліна, яка вивчає принципи

побудови та алгоритми, що застосовуються у системах комп'ютерної алгебри, отримала назву комп'ютерна алгебра.

Комп'ютерна алгебра – це окремий розділ інформатики, який займається розробкою, аналізом, реалізацією та практичним застосуванням алгоритмів алгебри. Відмінною рисою таких алгоритмів є те, що вони мають прості й формально чіткі описи, для них існують строго доведені умови правильності, а також визначені межі часу виконання. Ці характеристики стають можливими завдяки добре розвинутій математичній теорії, що лежить в їх основі.

Ще однією важливою особливістю є те, що об'єкти алгебри можуть бути точно представлені у пам'яті обчислювальної машини. Це дозволяє виконувати алгебраїчні перетворення без втрати точності, що вигідно відрізняє комп'ютерну алгебру від чисельних методів, де похибки є неминучими.

Алгоритми алгебри зазвичай реалізуються у спеціалізованих програмних системах, які підтримують введення та виведення інформації у символьних позначеннях. Така можливість робить ці системи зручним інструментом як для навчання і досліджень, так і для прикладного використання.

Завдяки цим особливостям комп'ютерна алгебра привертає дедалі більший інтерес фахівців, що працюють у сфері інформатики, математики, а також у прикладних галузях науки й техніки.

### **1.3 Огляд програмних засобів комп'ютерної математики**

Програмні засоби комп'ютерної математики почали створюватися у 1970-х роках і розвивалися в межах наукових проєктів, пов'язаних із дослідженнями у сфері штучного інтелекту. Завдяки цьому сфера їх застосування з самого початку була надзвичайно широкою та різноманітною.

Першими відомими й популярними системами стали Reduce, Derive та Macsyma. У подальшому з'явилася велика кількість інших програмних продуктів, як комерційних, так і вільно поширюваних. На сьогодні лідерами серед комерційних систем є пакети Maple та Mathematica, які активно застосовуються в математичних дослідженнях, інженерній практиці та багатьох інших науково-

технічних галузях. Серед інших відомих комерційних продуктів можна назвати MathCad. Водночас існує чимало безкоштовних систем, серед яких – Axiom, Eigenmath, Maxima, Yacas та інші.

Успіх сучасних програмних засобів комп'ютерної математики пояснюється їхньою інтеграцією з усіма основними машинними можливостями. Сучасні системи поєднують символний та числовий інтерфейс, підтримують вбудовану графіку, анімацію, роботу з базами та банками даних тощо.

Практично всі сучасні комерційні системи комп'ютерної математики (зокрема Mathematica, Maple, MatLab і Reduce) мають типовий набір функціональних можливостей:

- наявність вхідної макромови для спілкування користувача із системою, яка містить спеціалізований набір функцій для розв'язання математичних задач;
- підтримка основних символних математичних об'єктів: многочленів, рядів, раціональних функцій, виразів загального вигляду, векторів і матриць;
- використання чисел різних типів: цілих, раціональних, дійсних та комплексних;
- забезпечення декількох режимів роботи (редагування, діагностика, діалог, протоколювання), які взаємодоповнюють одне одного;
- наявність зв'язку із засобами розробки програм, що дозволяє здійснювати підстановки, обчислювати значення, автоматично генерувати програми та застосовувати стандартні математичні бібліотеки;
- підтримка інтерфейсів для взаємодії з офісними додатками, базами даних, графічними програмами та іншими зовнішніми засобами.

Таким чином, сучасні системи комп'ютерної математики являють собою потужні інтегровані програмні комплекси, що поєднують символні, чисельні та графічні методи обчислень і мають широке застосування в наукових, освітніх та інженерно-технічних дослідженнях.

Системи комп'ютерної алгебри можна умовно поділити на два основних типи: системи загального призначення та спеціалізовані системи.

До систем загального призначення належать такі відомі програмні продукти, як Macsyma, Reduce, Mathematica, Maple, Axiom та інші. Вони орієнтовані на широкий спектр завдань і можуть застосовуватися як у фундаментальних математичних дослідженнях, так і в прикладних науках чи технічних галузях.

Спеціалізовані системи вирізняються вищою ефективністю в обмеженій сфері застосування. Прикладом таких програм є CALEY та GAP, які розроблені спеціально для виконання обчислень у теорії груп.

Розглянемо характеристики деяких найбільш відомих систем:

**Macsyma** – одна з перших систем комп'ютерної алгебри, розробка якої здійснювалася з 1968 по 1982 рік у Массачусетському технологічному інституті (Massachusetts Institute of Technology, MIT) у межах лабораторії Project MAC. Згодом система поширювалася на комерційній основі. Вона стала першою системою символьної математики широкого застосування та однією з перших програм, заснованих на знаннях. Багато ідей, упроваджених у Macsyma, пізніше були використані у таких продуктах, як Mathematica та Maple.

**Maple** – програмний пакет компанії Waterloo Maple Inc., орієнтований на виконання як аналітичних, так і чисельних обчислень. Maple має розвинену систему команд, зручний графічний інтерфейс і широкі можливості візуалізації результатів. Система широко застосовується у задачах математичного моделювання. Архітектура Maple складається з ядра, оптимізованих процедур та великої бібліотеки, написаної власною мовою, схожою на Pascal. Ядро виконує основні операції, тоді як бібліотека містить тисячі команд і процедур, що запускаються в інтерпретованому режимі.

**Mathematica** – система комп'ютерної алгебри компанії Wolfram Research. Вона поєднує функціонал для аналітичних перетворень, чисельних розрахунків, а також для роботи з графікою і мультимедійними даними. Mathematica підтримує побудову дво- і тривимірних графіків, створення довільних геометричних фігур, імпорт та експорт зображень і звуку. Особливістю системи є те, що вона виступає не лише ПЗКМ, але й потужною мовою програмування. Архітектура Mathematica

складається з ядра, яке інтерпретує код та виконує обчислення, і користувацького інтерфейсу, що відображає результати у зрозумілій формі. Основну аудиторію користувачів, за даними компанії-розробника, складають професіонали у сфері науки і техніки.

**MathCad** – система комп'ютерної алгебри, що належить до класу систем автоматизованого проектування. Вона орієнтована на створення інтерактивних документів, які поєднують обчислення з графічним супроводом. Відмінними рисами MathCad є простота використання, інтуїтивний інтерфейс і підтримка колективної роботи. Введення формул і даних може здійснюватися як із клавіатури, так і за допомогою спеціальних панелей інструментів. Обчислення виконуються у візуальному середовищі робочого аркуша за принципом WYSIWYG (What You See Is What You Get – «що бачиш, те й отримуєш»).

**Axiom** – вільна система комп'ютерної алгебри загального призначення. Вона включає середовище інтерпретатора, компілятор та бібліотеку, яка містить математично коректно побудовану ієрархію типів. Завдяки цьому об'єкти, що використовуються в Axiom, є особливо зручними для математиків і науковців.

#### 1.4 Аналіз роботи вільної системи комп'ютерної алгебри Maxima

Розглянемо вільну систему комп'ютерної алгебри **Maxima**, яка походить від системи DOE Macsyma. Програмний пакет Maxima складається з інтерпретатора спеціальної макромови, написаної мовою програмування *Lisp*, а також із декількох поколінь додаткових модулів, що створювалися як за допомогою цієї макромови, так і безпосередньо мовою *Lisp*.

Розробка Maxima здійснювалася американським науковцем Вільямом Шелтером упродовж майже двадцяти років – з 1982 до 2001 року. У 1998 році він отримав офіційний дозвіл на поширення системи у вигляді відкритого програмного забезпечення на основі ліцензії GNU GPL. Невдовзі після цього Вільям передав подальший розвиток та підтримку Maxima спільноті користувачів і розробників. Завдяки їхній активній роботі система й сьогодні розвивається досить швидкими

темпами і за багатьма характеристиками не поступається провідним комерційним системам комп'ютерної математики, таким як Maple або Mathematica.

Система Maxima дає змогу розв'язувати широкий спектр математичних задач, серед яких:

- виконання операцій з многочленами (пошук коренів, розкладання на множники, спрощення виразів шляхом зведення подібних доданків тощо);
- проведення обчислень з використанням елементарних функцій, включаючи логарифмічні, показникові та тригонометричні функції;
- знаходження границь і похідних;
- аналітичне обчислення інтегралів;
- розв'язування алгебраїчних рівнянь та систем рівнянь.

До основних переваг Maxima належать:

- вільне поширення і можливість безкоштовного використання (система є прикладом відкритого програмного забезпечення на основі ліцензії GNU);
- сумісність із різними операційними системами, зокрема Windows та Linux;
- невеликий розмір програмного пакета, що полегшує встановлення та використання;
- здатність виконувати широкий спектр математичних завдань;
- можливість роботи як у консольному режимі, так і з використанням зручних графічних інтерфейсів (наприклад, *xMaxima* або *wxMaxima*);
- наявність довідкових матеріалів і детальних інструкцій для користувачів;
- включення до пакета розширення *wxMaxima*, яке забезпечує інтуїтивно зрозумілий інтерфейс, що значно спрощує роботу і знімає необхідність у детальному вивченні синтаксису команд для розв'язування стандартних задач.

Система Maxima є консольною програмою, яка відображає математичні формули за допомогою звичайних текстових символів (див. рисунок 1.1). Такий підхід має щонайменше дві істотні переваги. По-перше, Maxima може слугувати ядром, для якого створюються графічні інтерфейси різного рівня складності та

зручності. По-друге, у базовому вигляді, без використання додаткових інтерфейсних оболонок, програма не висуває високих вимог до апаратних ресурсів комп'ютера. Крім того, консольний режим є зручним інструментом для пакетного виконання великої кількості завдань.

На сьогодні розроблено чимало графічних інтерфейсів для Maxima, проте особливу увагу заслуговує один із найпоширеніших, найбільш наочних і зручних у використанні – wxMaxima (див. рисунок 1.1).

Основними перевагами wxMaxima є:

- можливість графічного відображення математичних формул;
- зручне введення найбільш уживаних функцій за допомогою діалогових вікон;
- поділ робочого вікна програми на окремі функціональні блоки, що підвищує наочність і зручність у роботі.

Далі розглянемо робоче вікно програми wxMaxima та його основні елементи.

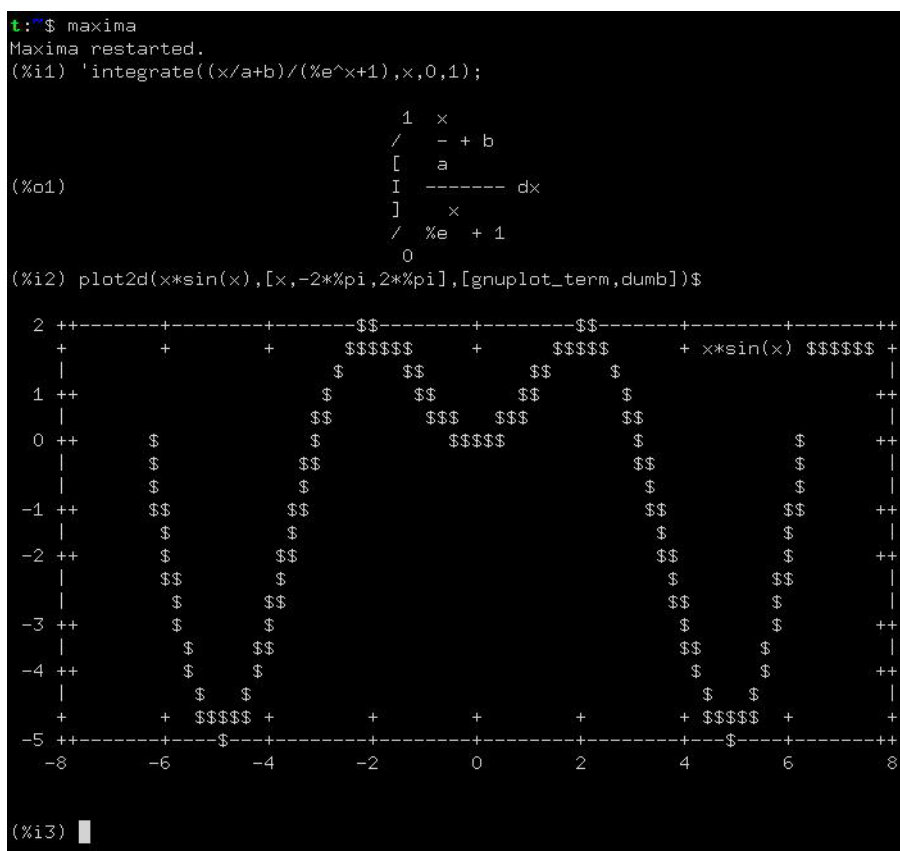


Рисунок 1.1 – СКМ Maxima в консольному режимі

У верхній частині робочого вікна програми розташоване текстове меню, яке забезпечує доступ до основних функцій та параметрів налаштування. Меню wxMaxima містить широкий набір команд для розв'язування різних типових математичних задач. Для зручності користувача ці команди згруповані за тематичними розділами: рівняння, алгебра, математичний аналіз, спрощення виразів, побудова графіків, чисельні обчислення тощо.

Використання діалогових вікон для введення команд значно полегшує роботу з програмою, особливо для початківців, оскільки знімає потребу у детальному запам'ятовуванні синтаксису.

Під текстовим меню розташована панель інструментів, яка дозволяє швидко виконувати найчастіше вживані дії. З лівого боку вікна розміщується панель виклику найуживаніших команд. Приклад діалогового вікна однієї з таких команд показано на рисунку 1.2.

Центральну і водночас найбільшу частину інтерфейсу займає вікно документа. Саме тут користувач вводить математичні вирази, виконує розрахунки та отримує результати у зручному для аналізу вигляді.

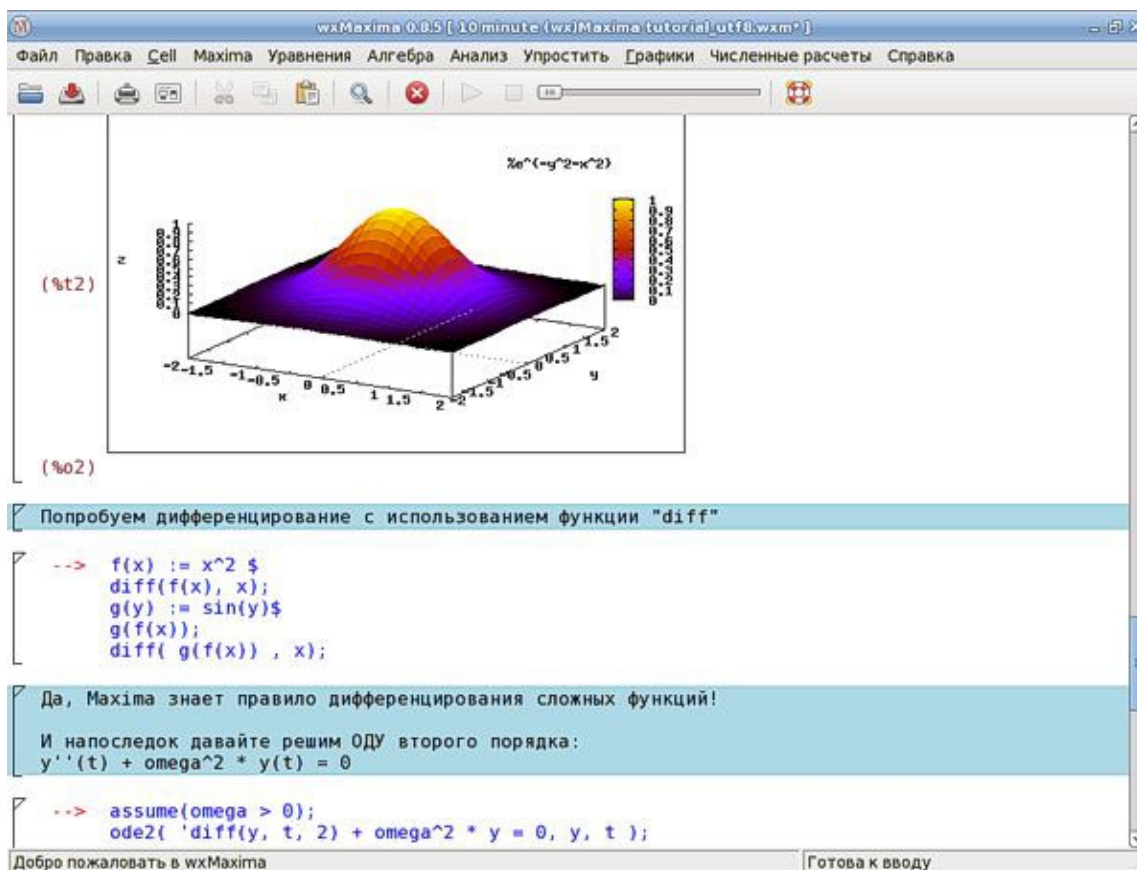


Рисунок 1.2 – Вікно wxMaxima

Кожен документ у програмі wxMaxima розглядається як набір окремих блоків. У літературі такі блоки можуть називатися елементами, секціями або клітинками (див. рисунок 1.3). У подальшому викладі ми будемо користуватися першим варіантом перекладу англійського слова *cell* – терміном «блок».

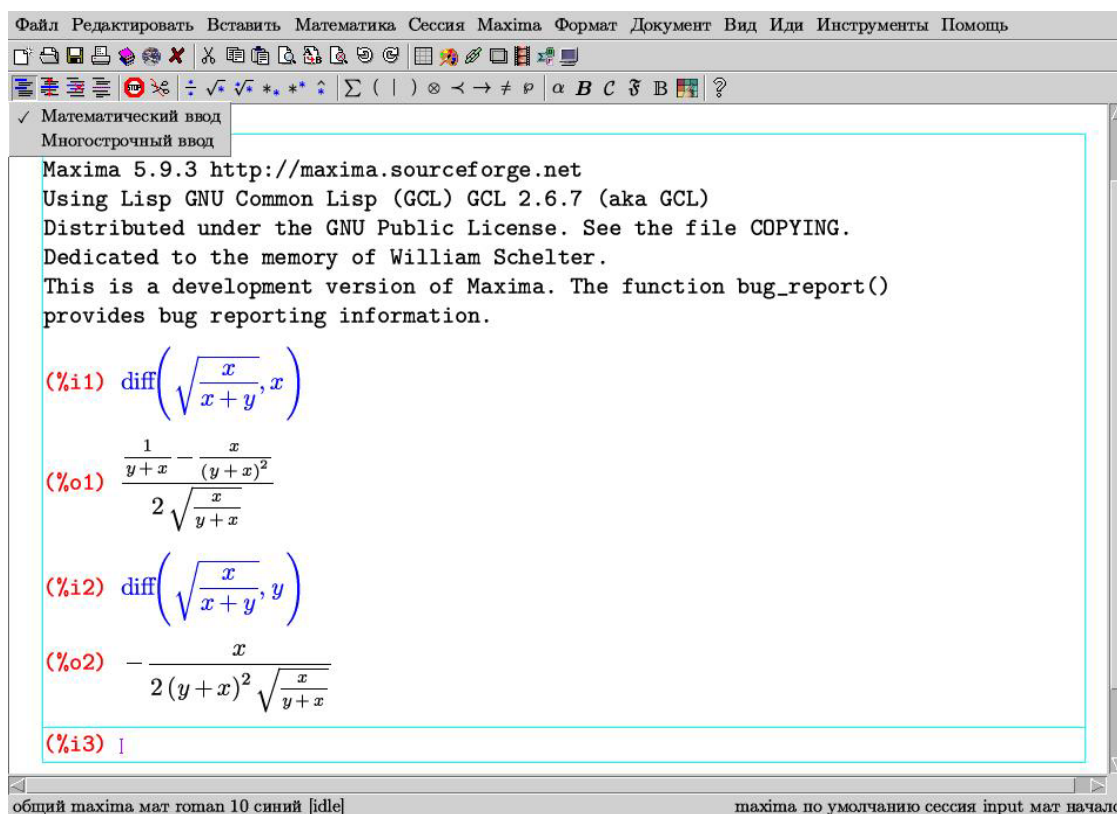


Рисунок 1.3 – Документ wxMaxima

Кожен блок має спеціальне графічне позначення: зліва відображається квадратна дужка, яка вказує на початок та кінець цього блоку. Залежно від призначення розрізняють різні типи блоків, наприклад блок заголовка чи блок тексту. Найважливішим є блок введення, у якому користувач вводить математичні вирази. Обчислення вмісту блоку здійснюється за допомогою комбінації клавіш Shift+Enter.

Перед надсиланням команди на обробку система wxMaxima перевіряє правильність завершення рядка. Кожен вираз повинен закінчуватися символом «;» або «\$». Якщо користувач забув поставити цей символ, програма автоматично завершує рядок крапкою з комою («;»). Це пов'язано з тим, що обчислювальна

система Maxima вимагає обов'язкової наявності одного з цих символів наприкінці кожного рядка.

Будь-який результат, отриманий від Maxima, автоматично додається програмою wxMaxima як елемент виведення безпосередньо до відповідного блоку введення.

Редагування вмісту блоку здійснюється дуже просто: потрібно виділити блок натисканням лівої кнопки миші. У цей момент з'являється курсор, а ліва дужка блоку змінює колір на червоний, сигналізуючи про можливість внесення змін. Якщо ж користувач клацне мишею за межами будь-якого блоку, з'являється горизонтальна лінія – так званий «горизонтальний курсор». Натиснувши будь-яку клавішу, у цьому місці можна створити новий блок введення.

Крім того, за допомогою комбінацій клавіш Shift+↑ або Shift+↓ можна виділяти кілька блоків у відповідному напрямку. Виділені блоки можна видаляти клавішами Backspace або Delete, а також копіювати, вирізати чи вставляти за допомогою команд меню «Редагувати» (Cut/Copy/Paste).

Іноді результати обчислень можуть займати значний обсяг. У такому випадку користувач має можливість приховати вивід: достатньо клацнути на порожньому трикутнику, що знаходиться у верхній частині лівої дужки блоку. Після цього трикутник стає заповненим, що означає прихований результат. Повторне натискання на нього знову відкриває вивід, роблячи його доступним для перегляду.

Якщо результат роботи над документом користувача задовольняє, його можна зберегти за допомогою команди Файл → Зберегти. Слід мати на увазі, що частина виводу блоків введення при цьому не зберігається. Після повторного відкриття документа можна швидко виконати всі обчислення у блоках, скориставшись командою Cell → Evaluate All Cells.

У процесі роботи з інтерфейсом wxMaxima користувач має можливість виділити у вікні виводу потрібну формулу й, викликавши контекстне меню правою кнопкою миші, скопіювати її у різних форматах: у вигляді звичайного тексту, у форматі TeX, а також як графічне зображення для подальшого вставлення в інший документ.

Крім того, у контекстному меню, яке з'являється під час вибору результату обчислення, пропонується ціла низка додаткових операцій з обраним виразом, наприклад: спрощення, розкриття дужок, інтегрування, диференціювання та інші дії.

Команда у системі *Maxima* – це будь-яка послідовність математичних виразів і вбудованих функцій, яка у найпростішому випадку завершується крапкою з комою («;»). Після введення команди та натискання комбінації клавіш *Shift+Enter* *Maxima* виконує її, виводить результат і переходить у режим очікування наступної команди.

Кожна введена команда отримує власний порядковий номер. Наприклад, на рисунку 1.3 команди мають позначення (%i1), (%i2), (%i3). Літера *i* у скороченні походить від англійського слова *input* («введення»). Відповідні результати обчислень позначаються як (%o1), (%o2), (%o3), де літера *o* є скороченням від слова *output* («виведення»).

Такий спосіб нумерації є зручним, оскільки дозволяє у наступних командах звертатися до вже використаних виразів чи результатів. Наприклад, запис (%i1)+(%i2) означає додавання виразу першої команди до виразу другої з подальшим обчисленням результату. Аналогічно можна використовувати й номери результатів: наприклад, (%o1)-(%o3) означає віднімання результату третього обчислення від результату першого.

Для останньої виконаної команди у *Maxima* існує спеціальне позначення – символ «%». Це дає змогу швидко звертатися до найсвіжішого результату без необхідності уточнювати його номер.

Таким чином, інтерфейс *wxMaxima* є зручним і наочним середовищем для роботи з математичною системою *Maxima*. Він поєднує в собі простоту консольного режиму з додатковими можливостями графічного відображення результатів, що робить роботу більш зрозумілою і доступною для користувачів різного рівня підготовки.

Основними перевагами є:

- зручний поділ документа на блоки з можливістю редагування, копіювання, приховування та повторного використання виразів;

- підтримка графічного відображення формул та експорт результатів у різних форматах (текст, TeX, графіка);
- наявність меню з готовими командами для розв'язання широкого спектра математичних задач;
- невибагливість до апаратних ресурсів, що дозволяє працювати навіть на відносно слабких комп'ютерах;
- інтеграція з базовою системою Maxima, що забезпечує аналітичні обчислення високого рівня.

Разом із тим, система має і певні недоліки. Серед них можна відзначити:

- порівняно менш сучасний інтерфейс у порівнянні з комерційними системами, такими як Maple чи Mathematica;
- обмежені можливості роботи з великими наборами даних і складною візуалізацією;
- необхідність знати базові принципи введення команд у Maxima, адже іноді діалогових вікон недостатньо;
- неповне збереження результатів під час збереження документа – повторно відкривши файл, користувач має виконати обчислення знову.

У цілому wxMaxima є потужним і водночас доступним інструментом для навчання, наукових досліджень і прикладних розрахунків. Його відкритість, багатофункціональність і зручність роблять систему конкурентною альтернативою відомим комерційним пакетам, особливо у сфері освіти та самостійного вивчення математики.

Після проведеного детального огляд та аналізу тематичної області, огляду наявних програмних засобів, визначення їх функціоналу, переваг та недоліків було поставлено завдання розробити додаток для розв'язку задач векторного числення зі зручним інтуїтивно-зрозумілим інтерфейсом.

## 2 АНАЛІЗ ТА ШЛЯХИ РОЗВ'ЯЗКУ ПРОБЛЕМИ

### 2.1 Векторний метод та його застосування

Поняття вектора є одним з основоположних у сучасній математиці. Його визначення може мати різні форми: як напрямлений відрізок; як впорядковану пару точок, що відповідають початку і кінцю відрізка; як множину відрізків однакової довжини, паралельних і однаково напрямлених; як впорядковану пару чисел; як операцію паралельного перенесення.

Вперше ідея вектора у вигляді напрямленого відрізка була використана в механіці для наочного подання фізичних величин векторного характеру, серед яких швидкість, прискорення, сила, момент сили та інші. Завдяки зрозумілості та наочності геометричних операцій з векторами у вигляді напрямлених відрізків це поняття отримало широке визнання і почало застосовуватися у багатьох розділах фізики: у кінематиці, статиці, динаміці точки та динаміці системи, у теорії потенціалу, у гідродинаміці. Згодом воно стало одним із головних у таких галузях науки, як векторна алгебра, векторний аналіз, теорія поля, тензорний аналіз та інші.

Хоча своє перше застосування поняття вектора знайшло саме у фізиці, за своєю природою воно є математичним. Усі дії з векторами виконуються відповідно до законів математики.

На сьогодні вектор як математичне поняття міцно утвердився у шкільному курсі математики, а також у численних дисциплінах, що не належать до математики. У навчанні векторний метод є надзвичайно важливим, адже з його допомогою розв'язуються задачі, для яких інші способи є неможливими чи значно складнішими.

Таким чином, вивчення поняття вектора має виняткове значення в умовах сучасного розвитку математичних наук.

## Поняття вектора

В елементарній геометрії відрізком  $AB$  називають множину всіх точок прямої, що розташовані між точками  $A$  і  $B$ . Точки  $A$  та  $B$  прийнято називати кінцями відрізка. При цьому порядок, у якому розглядаються кінцеві точки, значення не має. Проте у випадках, коли геометрія використовується для опису фізичних процесів, зокрема в механіці, виникає необхідність у розгляді напрямлених відрізків, тобто таких відрізків, для яких визначено початкову та кінцеву точки. Таким чином, хоча відрізки  $AB$  і  $BA$  є однаковими з геометричної точки зору, у контексті напрямлених відрізків вони задають різні об'єкти.

**Означення 1.** Відрізок  $AB$  називається напрямленим, якщо враховується порядок його кінцевих точок. При цьому перша точка ( $A$ ) вважається початком, а друга ( $B$ ) – кінцем відрізка. Направлений відрізок позначають як  $\overrightarrow{AB}$ .

**Означення 2.** Довжиною напрямленого відрізка  $\overrightarrow{AB}$  називається довжина відрізка  $AB$ . Позначення має вигляд:  $|\overrightarrow{AB}| = |\overrightarrow{BA}|$ .

**Означення 3.** Направлені відрізки  $AB$  і  $CD$  називаються однаково напрямленими (співнаправленими), якщо однойменні промені  $AB$  і  $CD$  мають однаковий напрямок, та протилежно напрямленими, якщо відповідні промені напрямлені у протилежні сторони.

**Означення 4.** Вектором називають множину всіх напрямлених відрізків, які мають однакову довжину та однаковий напрямок.

**Означення 4.1.** Вектором також називають напрямлений відрізок, для якого чітко вказано, яка з кінцевих точок вважається початковою, а яка – кінцевою. Початкова точка напрямленого відрізка називається початком вектора, а кінцева точка – його кінцем.

На кресленнях напрям вектора позначається стрілкою, що спрямована гострим кінцем у бік його кінця. У текстах вектори зазвичай записуються двома великими літерами латинського алфавіту з ризикою зверху, де перша літера позначає початок, а друга – кінець вектора. Наприклад,  $\overrightarrow{AB}$ ,  $\overrightarrow{CD}$  (див. рис. 2.1.а), при цьому точки  $A$ ,  $C$  є початками, а  $B$ ,  $D$  – відповідно кінцями векторів. У деяких

випадках допускається використання позначень у вигляді однієї малої літери, наприклад  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{c}$ ,... (див. рис. 2.1.b).

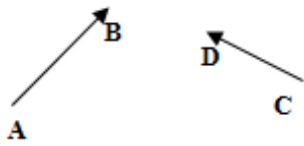


Рисунок 2.1.a – Означення вектору

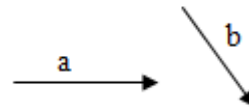


Рисунок 2.1.b – Означення вектору

**Означення 5.** Два вектори  $\overrightarrow{AB}$  і  $\overrightarrow{CD}$  називаються однаково напрямленими (співнаправленими), якщо відповідні їм напрямлені відрізки  $\overline{AB}$  і  $\overline{CD}$  є співнаправленими (див. рис. 2.2.a). Якщо ж ці відрізки мають протилежні напрямки, то відповідні вектори  $\overrightarrow{AB}$  і  $\overrightarrow{CD}$  називають протилежно напрямленими (див. рис. 2.2.b).

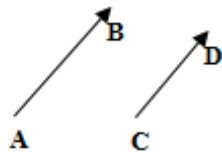


Рисунок 2.2.a – Співнаправлені вектори

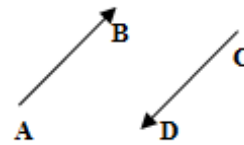


Рисунок 2.2.b – Протилежно напрямлені вектори

**Означення 6.** Довжиною або модулем вектора називають довжину будь-якого з напрямлених відрізків, що належать до класу співнаправлених і рівних за довжиною відрізків, які визначають даний вектор. Іншими словами, довжина вектора збігається з довжиною будь-якого напрямленого відрізка, що його зображає.

Модуль вектора  $\vec{a}$  позначається як  $|\vec{a}|$ , а модуль вектора  $\overrightarrow{AB}$  – як  $|\overrightarrow{AB}|$ .

Вектор, у якого початок і кінець збігаються, називається нульовим вектором. Його позначають  $\overrightarrow{AA}$  або  $\vec{0}$ . Нульовий вектор не має напрямку, а його довжина дорівнює нулю.

Вектор, довжина якого дорівнює одиниці, називається одиничним вектором, або ортом.

## Рівність векторів

**Означення 1.** Два вектори називаються рівними, якщо множини напрямлених відрізків, які відповідають цим векторам, збігаються. У цьому випадку пишуть:  $\vec{a} = \vec{b}$ .

Усі нульові вектори вважаються рівними один одному.

З цього означення випливає важлива ознака рівності двох векторів.

**Теорема 1.** (Перша ознака рівності векторів).

Для того щоб два вектори були рівними, необхідно й достатньо, щоб вони були співнаправленими та мали однакову довжину.

Доведення.

1. Необхідність. Нехай  $\vec{AB} = \vec{CD}$ . Доведемо, що  $\vec{AB} \uparrow\uparrow \vec{CD}$  і  $|\vec{AB}| = |\vec{CD}|$ .

Якщо  $\vec{AB} = \vec{CD}$ , то, згідно означенню 1, множини напрямлених відрізків, які відповідають цим векторам, збігаються. Отже, будь-які представники цих множин мають однакову довжину і однаковий напрям. Таким чином, ці вектори є співнаправленими та рівними за довжиною:  $\vec{AB} \uparrow\uparrow \vec{CD}$ ,  $|\vec{AB}| = |\vec{CD}|$ , що і потрібно було довести.

2. Достатність. Нехай вектори співнаправлені та рівні за довжиною:  $\vec{AB} \uparrow\uparrow \vec{CD}$ ,  $|\vec{AB}| = |\vec{CD}|$ . Потрібно довести, що  $\vec{AB} = \vec{CD}$ .

Якщо вектори співнаправлені та рівні за довжиною, то відповідні напрямлені відрізки належать одній і тій самій множині однаково напрямлених відрізків однакової довжини. Це значить, що  $\vec{AB} = \vec{CD}$ . Теорему доведено.

**Наслідок.** Якщо два вектори дорівнюють третьому, то вони рівні між собою.

**Теорема 2.** (Теорема про відкладання вектора).

Від будь-якої точки простору можна відкласти єдиний вектор, рівний даному.

Доведення.

Нехай дано вектор  $\vec{AB}$ . Виберемо у просторі довільну точку  $O$ . З'єднаємо точку  $B$  з точкою  $O$  і позначимо через  $C$  середину відрізка  $OB$  (див. рис. 2.3). Проведемо відрізок  $AC$  і відкладемо на його продовженні відрізок  $CM = AC$ .

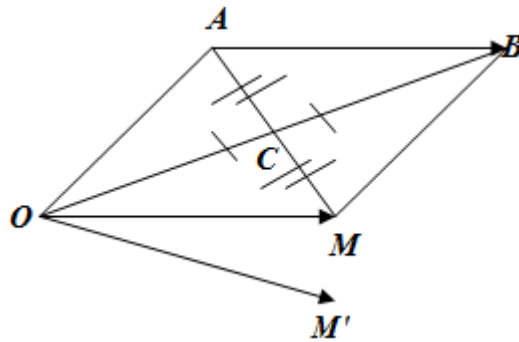


Рисунок 2.3 – Доведення Теорема 2

Розглянемо чотирикутник  $ABMO$ . Він є паралелограмом, оскільки точка перетину ділить його діагоналі навпіл. Звідси випливає, що промені  $AB$  і  $OM$  є однаково спрямованими, а відрізки  $AB$  і  $OM$  рівні один одному. Тому  $\overrightarrow{OM} = \overrightarrow{AB} = \vec{a}$ .

Тепер залишилося довести, що такий вектор єдиний. Нехай існує ще один вектор  $\overrightarrow{OM'} = \vec{a}$ , який є відмінним від  $\overrightarrow{OM}$ . Але ж і  $\overrightarrow{OM} = \vec{a}$ , тому, згідно наслідку Теорема 1,  $\overrightarrow{OM'} = \overrightarrow{OM}$ . Отже,  $\overrightarrow{OM'} \uparrow \overrightarrow{OM}$ ,  $|\overrightarrow{OM'}| = |\overrightarrow{OM}|$ , тому точки  $M$  і  $M'$  співпадають, а це суперечить припущенню. Отже, від точки  $O$  можна відкласти тільки один вектор, рівний вектору  $\vec{a}$ . Теорему доведено.

**Означення 2.** Два вектори називаються протилежними, якщо вони протилежно спрямовані та мають однакові довжини. Вектор, що є протилежним до  $\vec{a}$ , позначається  $-\vec{a}$  (див. рис. 2.4). Зрозуміло, що  $\overrightarrow{AB} = -\overrightarrow{BA}$ ,  $-(-\vec{a}) = \vec{a}$ .

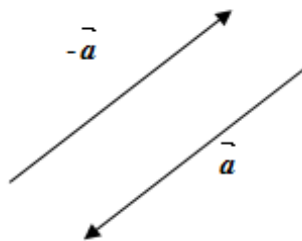


Рисунок 2.4 – Протилежні вектори

### Додавання векторів

Розглянемо операцію додавання векторів, яка має важливе значення в векторній алгебрі.

**Означення.** Нехай у нас задано два вектори  $\vec{a}$  і  $\vec{b}$ . Від точки **A** відкладемо вектор  $\overrightarrow{AB} = \vec{a}$ , а від точки **B** – вектор  $\overrightarrow{BC} = \vec{b}$ . Вектор  $\overrightarrow{AC} = \vec{c}$  називається сумою векторів  $\vec{a}$  і  $\vec{b}$  і позначається як  $\vec{c} = \vec{a} + \vec{b}$  (див. рис. 2.5). Відзначимо, що для знаходження двох неколінеарних векторів потрібно будувати трикутник. Отому правило додавання векторів називають ще правилом трикутника. Правило трикутника можна висловити так: для будь-яких трьох точок **A**, **B** і **C**  $\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}$ , або: сумою векторів  $\vec{a}$  і  $\vec{b}$  є вектор  $\vec{c}$ , який з'єднує початок вектора  $\vec{a}$  з кінцем вектора  $\vec{b}$  в випадку, коли вектор  $\vec{b}$  відкладено від кінця вектора  $\vec{a}$ .

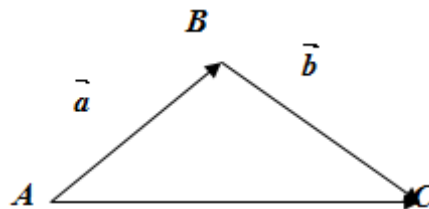


Рисунок 2.5 – Додавання векторів (Правило трикутника)

З правила трикутника спливає правило паралелограма: якщо вектори  $\vec{a}$  і  $\vec{b}$  відкладено від спільного початку **O**,  $\overrightarrow{OA} = \vec{a}$ ,  $\overrightarrow{OB} = \vec{b}$  (див. рис. 2.6) і на них побудовано паралелограм **OACB**, то сумою векторів  $\vec{a} + \vec{b}$  є вектор  $\vec{c} = \overrightarrow{OC}$ , який виходить з спільного початку **O** і збігається з діагоналлю **OC** паралелограма.

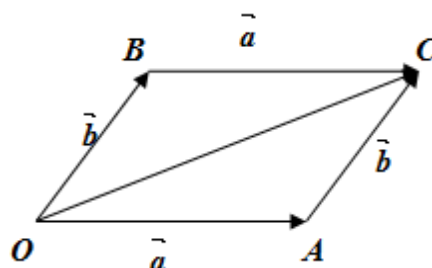


Рисунок 2.6 – Правило паралелограма

Вивчимо властивості операції додавання векторів.

**Властивість 1.** Операція додавання векторів є **комутативною**: для будь-яких векторів  $\vec{a}$  і  $\vec{b}$ :  $\vec{a} + \vec{b} = \vec{b} + \vec{a}$ .

Доведення:

Згідно правила трикутника виконується (див. рис. 2.6):

$$\begin{array}{l} \vec{a} + \vec{b} = \overrightarrow{OA} + \overrightarrow{AC} = \\ \overrightarrow{OC}, \end{array} \quad \left| \Rightarrow \vec{a} + \vec{b} = \vec{b} + \vec{a} \right.$$

**Властивість 2.** Операція додавання векторів є **асоціативною**: для будь-яких векторів  $\vec{a}, \vec{b}, \vec{c}$ :  $(\vec{a} + \vec{b}) + \vec{c} = \vec{a} + (\vec{b} + \vec{c})$

Доведення:

Виберемо довільну точку  $A$ . Відкладемо від точки  $A$  вектори  $\overrightarrow{AB} = \vec{a}$ ,  $\overrightarrow{BC} = \vec{b}$ ,  $\overrightarrow{CD} = \vec{c}$  (див. рис. 2.7). Тоді  $\vec{a} + \vec{b} = \overrightarrow{AC}$ ,  $(\vec{a} + \vec{b}) + \vec{c} = \overrightarrow{AD}$ ;  $\vec{b} + \vec{c} = \overrightarrow{BD}$ ;  $\vec{a} + (\vec{b} + \vec{c}) = \overrightarrow{AD}$ . Таким чином,  $(\vec{a} + \vec{b}) + \vec{c} = \vec{a} + (\vec{b} + \vec{c})$ .

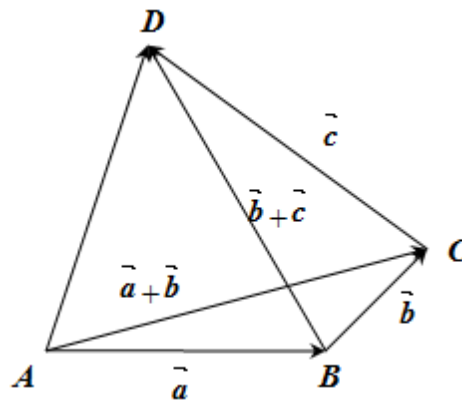


Рисунок 2.7 – Доведення Властивості 2

**Властивість 3.** Сумою протилежних векторів є нуль-вектор:  $\vec{a} + (-\vec{a}) = \vec{0}$ .

Доведення.

Припустимо  $\vec{a} = \overrightarrow{AB}$ , тоді  $-\vec{a} = \overrightarrow{BA}$ . Згідно правила трикутника:

$$\vec{a} + (-\vec{a}) = \overrightarrow{AB} + \overrightarrow{BA} = \overrightarrow{AA} = \vec{0}.$$

**Властивість 4.** Нуль-вектор є нейтральним елементом операції додавання:  $\vec{a} + \vec{0} = \vec{0} + \vec{a}$ .

Доведення:

Візьмемо  $\vec{a} = \overrightarrow{AB}$ ,  $\vec{0} = \overrightarrow{BB}$ . Згідно правила трикутника:  $\vec{a} + \vec{0} = \overrightarrow{AB} + \overrightarrow{BB} = \overrightarrow{AB} = \vec{a}$ .

Таким чином, властивості додавання векторів показують, що операція додавання векторів має ті ж самі властивості, що і операція додавання чисел. Отже

при перетворенні сум векторів будемо діяти так само, як і при перетворенні числових виразів:  $(\vec{a} + \vec{b}) + \vec{c} = \vec{a} + (\vec{b} + \vec{c}) = (\vec{b} + \vec{c}) + \vec{a} = \vec{b}(\vec{c} + \vec{a})$ .

Сума кількості векторів, більшої двох, визначається згідно правила багатокутника. Щоб знайти суму  $n$  векторів  $\vec{a}_1, \vec{a}_2, \dots, \vec{a}_n$  (див. рис. 2.8), з довільної точки  $O$  відкладемо вектор  $\vec{OA}_1 = \vec{a}_1$ , з кінця цього вектора – вектор  $\vec{A_1A_2} = \vec{a}_2, \dots, \vec{A_{n-1}A_n} = \vec{a}_n$  (тобто, початок кожного наступного вектора-доданка є кінцем попереднього вектора). Вектор  $\vec{c} = \vec{OA_n}$  буде сумою цих векторів.

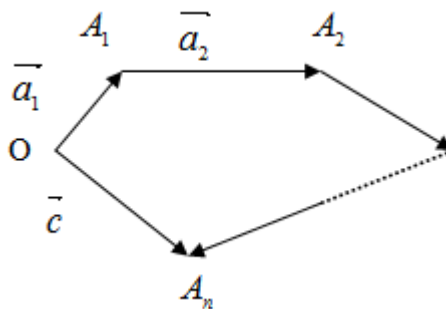


Рисунок 2.8 – Додавання векторів (Правило багатокутника)

### Віднімання векторів

Операція віднімання векторів вводиться як протилежна до їх додавання.

**Означення.** Різницею векторів  $\vec{a}$  і  $\vec{b}$  називається такий вектор  $\vec{x}$ , який в сумі з вектором  $\vec{b}$  дорівнює вектору  $\vec{a}$ :  $\vec{a} - \vec{b} = \vec{x}$ , якщо

$$\vec{b} + \vec{x} = \vec{a}. \quad (2.1)$$

Потрібно довести, що вектор  $\vec{x}$  існує і що такий вектор єдиний. Нехай вектор  $\vec{x}$  існує. Отоді додамо до обох частин рівності вектор  $(-\vec{b})$  і скористуємось властивостями суми векторів. В результаті отримаємо:

$$(-\vec{b}) + \vec{b} + \vec{x} = (-\vec{b}) + \vec{a}. \quad (2.2)$$

Тобто, якщо існує такий вектор  $\vec{x}$ , то він визначається рівністю (2.2), а тому єдиний. Справді, якщо ми підставимо (2.2) в (2.1), то отримаємо вірну рівність:  $\vec{b} + \vec{a} + (-\vec{b}) = \vec{a}$ .

Тому вектор, що рахується за формулою (2.2), є різницею векторів  $\vec{a}$  і  $\vec{b}$ :  $\vec{a} - \vec{b} = \vec{a} + (-\vec{b}) = \vec{x}$ . Згідно правила трикутника,  $\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}$ . Відси  $\overrightarrow{BC} = \overrightarrow{AC} - \overrightarrow{AB}$  (див. рис. 2.9).

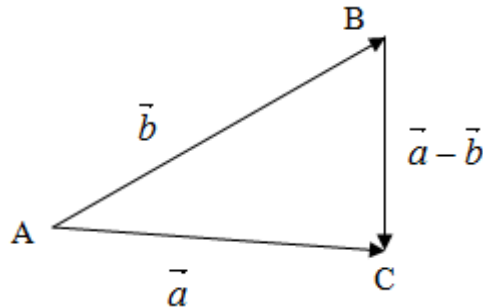


Рисунок 2.9 – Віднімання векторів

Отож, для того, щоб побудувати різницю векторів  $\vec{a}$  і  $\vec{b}$ , достатньо відкласти оці вектори від спільного початку  $A$  ( $\overrightarrow{AC} = \vec{a}$ ,  $\overrightarrow{AB} = \vec{b}$ ) та накреслити вектор  $\overrightarrow{BC}$  від кінця вектора  $B$ , що віднімається, до кінця вектора  $C$ , від якого він віднімається. Оцей вектор  $\overrightarrow{BC}$  є шуканою різницею  $\vec{a} - \vec{b}$ :  $\overrightarrow{BC} = \vec{a} - \vec{b}$ .

### Множення вектора на число

**Означення.** Добутком вектора  $\vec{a}$  на дійсне число  $\alpha$  називається вектор  $\vec{p}$ , що задовольняє таким умовам:

- 1)  $|\vec{p}| = |\alpha| * |\vec{a}|$ ;
- 2)  $\vec{p} \uparrow\uparrow \vec{a}$  при  $\alpha > 0$  та  $\vec{p} \uparrow\downarrow \vec{a}$  при  $\alpha < 0$ .

Оцей вектор запмсується як  $\vec{p} = \alpha \vec{a}$ .

Властивості операції добутку вектора на число:.

**Властивість 1.** Для усякого дійсного числа  $\alpha$  та усякого вектора  $\vec{a}$  виконується:  $\alpha * \vec{0} = 0 * \vec{a} = \vec{0}$ . Оця властивість є наслідком умови 1) означення.

**Властивість 2.** Для усякого вектора  $\vec{a}$  виконується:  $1 * \vec{a} = \vec{a}$ ;  $-1 * \vec{a} = -\vec{a}$ . Оця властивість спливає з означення.

**Властивість 3.** Для усякого вектора  $\vec{a}$  та усяких двох дійсних чисел  $\alpha$  і  $\beta$  виконується:  $\alpha(\beta \vec{a}) = (\alpha\beta) \vec{a}$ .

Доведення. Припустимо,  $\alpha(\beta\vec{a})=\vec{p}$ ,  $(\alpha\beta)\vec{a}=\vec{g}$ . Потрібно довести, що  $\vec{p}=\vec{g}$ .

Запишемо:

$$|\vec{p}|=|\alpha|*|\beta\vec{a}|=|\alpha|*|\beta|*|\vec{a}|,$$

$$|\vec{g}|=|\alpha\beta|*|\vec{a}|=|\alpha|*|\beta|*|\vec{a}|.$$

Таким чином,  $|\vec{p}|=|\vec{g}|$ . Потрібно показати, що  $\vec{p} \uparrow\uparrow \vec{g}$ . У випадку, коли  $\alpha$  і  $\beta$  мають один знак, вектор  $\vec{p}$  є однаково спрямованим з  $\vec{a}$ , а  $\vec{g}$  однаково спрямованим з  $\vec{a}$ . Тому  $\vec{p} \uparrow\uparrow \vec{g}$ . Якщо ж числа  $\alpha$  і  $\beta$  мають протилежні знаки,  $\vec{p} \uparrow\downarrow \vec{a}$ ,  $\vec{g} \uparrow\downarrow \vec{a}$ . Таким чином, теж  $\vec{p} \uparrow\uparrow \vec{g}$ , що і потрібно було довести.

**Властивість 4.** Операція множення вектора на число є дистрибутивною при додаванні векторів, себто  $\alpha(\vec{a}+\vec{b})=\alpha\vec{a}+\alpha\vec{b}$  для  $\forall \vec{a}, \vec{b}$  і  $\alpha \in \mathbb{R}$ .

Доведення. Розглянемо випадок  $\alpha > 0$ . Зобразимо вектори  $\vec{OA}=\vec{a}$ ,  $\vec{AB}=\vec{b}$ ,  $\vec{OA}'=\alpha\vec{a}$ ,  $\vec{A'B'}=\alpha\vec{b}$  (див. рис. 2.10). Відтак  $\vec{a}+\vec{b}=\vec{OB}$  та  $\alpha\vec{a}+\alpha\vec{b}=\vec{OB'}$ . Потрібно показати, що  $\vec{OB'}=\alpha\vec{OB}$ . Оскільки вектори  $\vec{a}$  і  $\alpha\vec{a}$  та  $\vec{b}$  і  $\alpha\vec{b}$  відповідно є однаково спрямованими, то кути  $A$  і  $A'$  у трикутників  $OAB$  і  $OA'B'$  будуть рівними (як кути, що утворюються перетином двох паралельних прямих третьою прямою). До того ж сторони трикутників  $OAB$  і  $OA'B'$ , які прилягають до рівних кутів, є пропорційними:  $\frac{OA'}{OA}=\frac{A'B'}{AB}=\alpha$ . Звідси випливає, що  $\triangle OAB \sim \triangle OA'B'$ . Звідси випливає, що промені  $OB$  і  $OB'$  співпадають, себто  $\vec{OB'} \uparrow\uparrow \vec{OB}$ . Окрім того  $|\vec{OB'}|=\alpha*|\vec{OB}|$ . Звідси  $\vec{OB'}=\alpha*\vec{OB}$ .

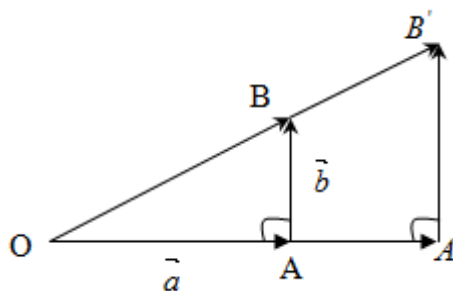


Рисунок 2.10 – Доведення Властивості 4 при  $\alpha > 0$

Аналогічно розглядається випадок, коли  $\alpha < 0$  (див. рис. 2.11).

Випадок  $\alpha = 0$  тривіальний. Отже,  $\alpha (\vec{a} + \vec{b}) = \alpha \vec{a} + \alpha \vec{b}$ .

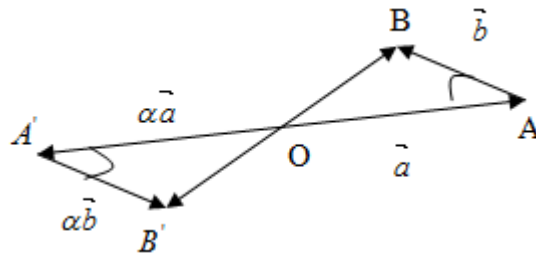


Рисунок 2.11 – Доведення Властивості 4 при  $\alpha < 0$

**Властивість 5.** Операція множення вектора на число є дистрибутивною при додаванні чисел, себто  $(\alpha + \beta)\vec{a} = \alpha\vec{a} + \beta\vec{a}$ ,  $\forall \vec{a}$  і  $\alpha, \beta \in \mathbb{R}$ .

Доведення. Будемо розглядати два ймовірних випадки:  $\alpha\beta > 0$  і  $\alpha\beta < 0$  (випадок  $\alpha\beta = 0$  є тривіальним).

1. Припустимо, що  $\alpha\beta > 0$ , тобто числа  $\alpha$  і  $\beta$  мають один знак. Відтак вектори

$(\alpha + \beta)\vec{a}$  і  $\alpha\vec{a} + \beta\vec{a}$  є однаково спрямованими. До того ж,

$$|\alpha\vec{a} + \beta\vec{a}| = |\alpha\vec{a}| + |\beta\vec{a}| = |\alpha| \cdot |\vec{a}| + |\beta| \cdot |\vec{a}| = (|\alpha| + |\beta|) \cdot |\vec{a}|;$$

$$|(\alpha + \beta)\vec{a}| = |\alpha + \beta| \cdot |\vec{a}| = (|\alpha| + |\beta|) \cdot |\vec{a}|.$$

Таким чином,  $|\alpha\vec{a} + \beta\vec{a}| = |(\alpha + \beta)\vec{a}|$ , отже вектори  $(\alpha + \beta)\vec{a}$  і  $\alpha\vec{a} + \beta\vec{a}$  є рівними.

2. Припустимо, що  $\alpha\beta < 0$ , тобто числа  $\alpha$  і  $\beta$  мають різні знаки. В випадку  $\alpha = -\beta$  має місце  $(\alpha + \beta)\vec{a} = (-\beta + \beta)\vec{a} = 0\vec{a} = 0$ ;  $\alpha\vec{a} + \beta\vec{a} = -\beta\vec{a} + \beta\vec{a} = 0$ . Тобто властивість виконується.

Розглянемо випадок  $\alpha \neq -\beta$ . Відтак  $-\alpha$ ,  $\alpha + \beta$  або  $-\beta$ ,  $\alpha + \beta$  мають один знак. Припустимо, наприклад,  $-\alpha$ ,  $\alpha + \beta$  мають один знак. Відтак за доведеним  $(-\alpha)\vec{a} + (\alpha + \beta)\vec{a} = (-\alpha + \alpha + \beta)\vec{a} = \beta\vec{a} \Rightarrow (\alpha + \beta)\vec{a} = \alpha\vec{a} + \beta\vec{a}$ . Це і потрібно було довести.

### Колінеарність векторів

**Означення.** Два ненульових вектори  $\vec{a}$  і  $\vec{b}$  називаються *колінеарними*, якщо відповідні їм спрямовані відрізки паралельні або належать одній прямій.

Вони позначаються:  $\vec{a} \parallel \vec{b}$ .

Зрозуміло, що колінеарні вектори є або однаково, або протилежно спрямованими. Нульовий вектор рахується колінеарним до будь-якого вектора. Тому, якщо два вектори є неколінеарними, то ніякий з них не є нульовим вектором.

**Теорема.** (перша ознака колінеарності двох векторів). Два ненульових вектори  $\vec{a}$  і  $\vec{b}$  колінеарні тоді і тільки тоді, коли існує деяке число  $\alpha$ , таке, що

$$\vec{b} = \alpha \vec{a}. \quad (2.3)$$

Доведення.

1. Необхідність. Припустимо, що  $\vec{a} \parallel \vec{b}$ . В такому випадку виконується: або  $\vec{a} \uparrow \vec{b}$ , або  $\vec{a} \downarrow \vec{b}$ . Коли  $\vec{a} \uparrow \vec{b}$ , має місце  $\vec{b} = \frac{|\vec{b}|}{|\vec{a}|} \vec{a}$ , тому що ці вектори є однаково спрямованими. Для їх модулів виконується:  $\left| \frac{|\vec{b}|}{|\vec{a}|} \vec{a} \right| = \frac{|\vec{b}|}{|\vec{a}|} |\vec{a}| = |\vec{b}|$ . Позначимо  $\alpha = \frac{|\vec{b}|}{|\vec{a}|}$ , отримаємо  $\vec{b} = \alpha \vec{a}$ .

У випадку  $\vec{a} \downarrow \vec{b}$  можна аналогічно довести, що  $\vec{b} = -\frac{|\vec{b}|}{|\vec{a}|} \vec{a}$ . Позначимо  $\alpha = -\frac{|\vec{b}|}{|\vec{a}|}$ , отримаємо також  $\vec{b} = \alpha \vec{a}$ .

2. Достатність. Припустимо, що виконується рівність (2.3). В такому випадку вектори  $\vec{a}$  і  $\vec{b}$  або однаково, або протилежно спрямовані, тобто вони є колінеарними. Отже, теорему доведено.

**Зауваження 1.** У випадку  $\vec{b} = 0$ ,  $\vec{a} \neq 0$  теорема також виконується. Тоді  $\alpha = 0$ .

**Зауваження 2.** Оскільки для колінеарних векторів  $\vec{a}$  і  $\vec{b}$  завжди існує єдине число  $\alpha$ , таке, що  $\vec{a} = \alpha \vec{b}$ , то відсінь формально можна записати:  $\alpha = \frac{\vec{a}}{\vec{b}}$ . Отже можна розглядати відношення двох колінеарних векторів.

Під відношенням  $\vec{a} : \vec{b}$  двох колінеарних векторів розглядають число, на яке потрібно помножити вектор  $\vec{b}$ , щоб отримати вектор  $\vec{a}$ . Тобто під відношенням двох колінеарних векторів розуміють число, що дорівнює відношенню модулів цих

векторів, взяті зі знаком «плюс», якщо вектори  $\vec{a}$  і  $\vec{b}$  однаково спрямовані, і зі знаком «мінус», якщо вектори протилежно спрямовані.

### Координати вектора

Розглянемо  $(\vec{e}_1, \vec{e}_2, \vec{e}_3)$  – якийсь базис простору  $V_3$  та  $\vec{a}$  – довільний вектор цього простору. Згідно теореми про розклад вектора за трьома некомпланарними векторами, існують тільки один набір чисел  $a_1, a_2, a_3$ , таких, що виконується

$$\vec{a} = a_1 \vec{e}_1 + a_2 \vec{e}_2 + a_3 \vec{e}_3.$$

Коефіцієнти  $a_1, a_2, a_3$  розкладання вектора за даними базисними векторами називають координатами вектора в цьому базисі. Число  $a_1$  називають першою координатою, число  $a_2$  – другою, а число  $a_3$  – третьою.

Позначення того, що вектор  $\vec{a}$  в даному базисі має координати  $a_1, a_2, a_3$ , записується у вигляді  $\vec{a}(a_1, a_2, a_3)$  або  $\overline{(a_1, a_2, a_3)}$ .

Дослідимо, який геометричний зміст мають координати вектору в даному базисі. Відкладемо вектори  $\vec{e}_1, \vec{e}_2, \vec{e}_3$  і  $\vec{a}$  від довільної точки  $O$  простору (див. рис. 2.12):  $\vec{e}_1 = \overrightarrow{OE_1}$ ,  $\vec{e}_2 = \overrightarrow{OE_2}$ ,  $\vec{e}_3 = \overrightarrow{OE_3}$ ,  $\vec{a} = \overrightarrow{OA}$ .

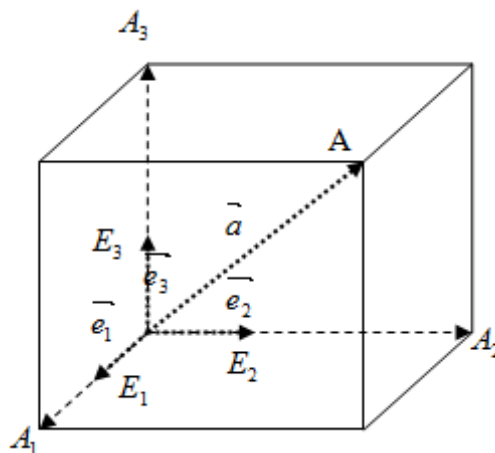


Рисунок 2.12 – Геометричний зміст координат вектору в просторі  $V_3$

Побудуємо паралелепіпед з ребрами, спрямованими поздовж прямих  $OE_1, OE_2, OE_3$ . Діагоналлю паралелепіпеда є відрізок  $OA$ . Відтак  $\vec{a} = \overrightarrow{OA_1} + \overrightarrow{OA_2} + \overrightarrow{OA_3}$ , де  $\overrightarrow{OA_1} = a_1 \vec{e}_1$ ,  $\overrightarrow{OA_2} = a_2 \vec{e}_2$ ,  $\overrightarrow{OA_3} = a_3 \vec{e}_3$ .

$$\text{Отже } \overrightarrow{OA_1} = |a_1| \cdot \overrightarrow{e_1} \Rightarrow |a_1| = \frac{OA_1}{|\overrightarrow{e_1}|};$$

$a_1$  має знак:  $a_1 > 0$  у випадку  $\overrightarrow{OA_1} \uparrow\uparrow \overrightarrow{e_1}$  і  $a_1 < 0$  у випадку  $\overrightarrow{OA_1} \uparrow\downarrow \overrightarrow{e_1}$ ;

$$\overrightarrow{OA_2} = |a_2| \cdot \overrightarrow{e_2} \Rightarrow |a_2| = \frac{OA_2}{|\overrightarrow{e_2}|};$$

$a_2$  має знак:  $a_2 > 0$  у випадку  $\overrightarrow{OA_2} \uparrow\uparrow \overrightarrow{e_2}$  і  $a_2 < 0$  у випадку  $\overrightarrow{OA_2} \uparrow\downarrow \overrightarrow{e_2}$ .

$$\text{Аналогічно, } \overrightarrow{OA_3} = |a_3| \cdot \overrightarrow{e_3} \Rightarrow |a_3| = \frac{OA_3}{|\overrightarrow{e_3}|};$$

$a_3$  має знак:  $a_3 > 0$  у випадку  $\overrightarrow{OA_3} \uparrow\uparrow \overrightarrow{e_3}$  і  $a_3 < 0$  у випадку  $\overrightarrow{OA_3} \uparrow\downarrow \overrightarrow{e_3}$ .

Таким чином, координата  $a_i$  з точністю до знака дорівнює довжині відрізка  $OA_i$ , яка вимірюється в одиницях довжини  $\overrightarrow{e_i}$ . Знак же координати  $a_i$  залежить від напрямку векторів  $\overrightarrow{OA_i}$  і  $\overrightarrow{e_i}$ :  $a_i > 0$  у випадку  $\overrightarrow{OA_i} \uparrow\uparrow \overrightarrow{e_i}$  та  $a_i < 0$  у випадку  $\overrightarrow{OA_i} \uparrow\downarrow \overrightarrow{e_i}$ . Аналогічно визначається геометричний зміст двох інших координат  $a_2$  і  $a_3$ .

Базисні вектори в своєму базисі мають координати:  $\overrightarrow{e_1} (1; 0; 0)$ ,  $\overrightarrow{e_2} (0; 1; 0)$ ,  $\overrightarrow{e_3} (0; 0; 1)$ .

Аналогічно можна визначити координати вектора в просторі  $V_2$ . Базисом цього підпростору є два неколінеарні вектори. Розглянемо систему векторів  $\overrightarrow{e_1}$ ,  $\overrightarrow{e_2}$  яє базис підпростору  $V_2$ . Відтак за теоремою про розклад вектора за двома неколінеарними векторами для кожного вектора  $\vec{a}$  із підпростору  $V_2$  існує тільки один набір чисел  $a_1$ ,  $a_2$ , для яких  $\vec{a} = a_1 \overrightarrow{e_1} + a_2 \overrightarrow{e_2}$ . Коефіцієнти  $a_1$ ,  $a_2$  такого розкладання називають координатами вектора  $\vec{a}$  в базисі  $(\overrightarrow{e_1}, \overrightarrow{e_2})$ . Число  $a_1$  називають першою координатою, а число  $a_2$  – другою. Аналогічно визначається геометричний зміст координат вектора в підпросторі  $V_2$  (рис. 2.13):

$$\overrightarrow{OA} = \overrightarrow{OA_1} + \overrightarrow{OA_2} = a_1 \overrightarrow{e_1} + a_2 \overrightarrow{e_2}.$$

$$\overrightarrow{OA_1} = |a_1| \cdot \overrightarrow{e_1} \Rightarrow |a_1| = \frac{OA_1}{|\overrightarrow{e_1}|},$$

$a_1$  має знак:  $a_1 > 0$  у випадку  $\overrightarrow{OA_1} \uparrow\uparrow \overrightarrow{e_1}$  і  $a_1 < 0$  у випадку  $\overrightarrow{OA_1} \uparrow\downarrow \overrightarrow{e_1}$ ;

$$\overrightarrow{OA_2} = |a_2| \cdot \overrightarrow{e_2} \Rightarrow |a_2| = \frac{OA_2}{|\overrightarrow{e_2}|};$$

$a_2$  має знак:  $a_2 > 0$  у випадку  $\overrightarrow{OA_2} \uparrow\uparrow \overrightarrow{e_2}$  і  $a_2 < 0$  у випадку  $\overrightarrow{OA_2} \uparrow\downarrow \overrightarrow{e_2}$ .

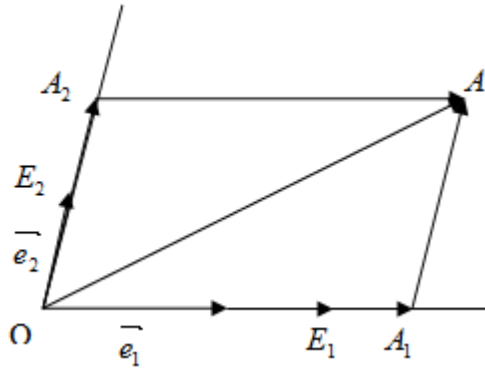


Рисунок 2.13 – Геометричний зміст координат вектору в просторі  $V_2$

Базисні вектори в своєму базисі мають координати:  $\overrightarrow{e_1}(1; 0)$ ,  $\overrightarrow{e_2}(0; 1)$ . Координати вектора в даному базисі повністю визначають вектор.

Вивчимо властивості координат векторів.

**Теорема (2-га ознака рівності векторів):** для рівності двох векторів необхідно і достатньо, щоб були рівними їх відповідні координати.

Доведення цієї теореми є очевидне, воно є наслідком єдиності розкладу вектора за трьома некопланарними векторами.

**Теорема:** Такі твердження є вірними:

- 1) координати суми двох векторів дорівнюють сумі відповідних координат цих векторів;
- 2) координати різниці двох векторів дорівнюють різниці відповідних координат цих векторів;
- 3) координати добутку вектора на число дорівнюють добутку відповідних координат цього вектора на дане число.

Доведення: доведемо, наприклад, перше твердження теореми. Розглянемо у деякому базисі  $(\overrightarrow{e_1}, \overrightarrow{e_2}, \overrightarrow{e_3})$ ,  $\vec{a}(a_1, a_2, a_3)$ ,  $\vec{b}(b_1, b_2, b_3)$ . Отоді за означенням координат вектора

$$\vec{a} = a_1 \vec{e}_1 + a_2 \vec{e}_2 + a_3 \vec{e}_3, \vec{b} = b_1 \vec{e}_1 + b_2 \vec{e}_2 + b_3 \vec{e}_3.$$

Таким чином,  $\vec{a} + \vec{b} = a_1 \vec{e}_1 + a_2 \vec{e}_2 + a_3 \vec{e}_3 + b_1 \vec{e}_1 + b_2 \vec{e}_2 + b_3 \vec{e}_3 = (a_1 + b_1) \vec{e}_1 + (a_2 + b_2) \vec{e}_2 + (a_3 + b_3) \vec{e}_3$ .

Звідси витікає, що координати вектора  $\vec{a} + \vec{b}$  відповідно дорівнюють  $a_1 + b_1, a_2 + b_2, a_3 + b_3$ . Це і потрібно було довести.

Аналогічно можна довести й інші властивості.

**Теорема (2-га ознака колінеарності двох векторів):** для колінеарності двох векторів  $\vec{a}(a_1, a_2, a_3), \vec{b}(b_1, b_2, b_3)$ , заданих в деякому базисі  $(\vec{e}_1, \vec{e}_2, \vec{e}_3)$  необхідно і достатньо, щоб їх координати були пропорційними.

Доведення: у випадку  $\vec{b} = \vec{0}$  доведення очевидне. Нехай  $\vec{b} \neq \vec{0}$ .

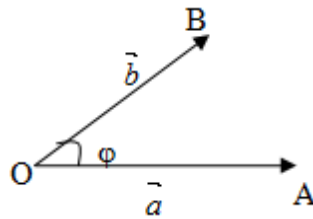
1. Необхідність. Припустимо, що  $\vec{a} \parallel \vec{b}$ . Тоді є таке число  $\lambda$ , що  $\vec{a} = \lambda \vec{b}$ , звідки витікає, що  $a_1 = \lambda b_1, a_2 = \lambda b_2, a_3 = \lambda b_3$ ;  $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \frac{a_3}{b_3} = \lambda$ .

Таким чином, якщо вектори є колінеарними, то їх координати пропорційні.

2. Достатність. Припустимо, що  $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \frac{a_3}{b_3} = \lambda$ , тоді  $a_1 = \lambda b_1, a_2 = \lambda b_2, a_3 = \lambda b_3$ . Помножимо ці рівності на вектори  $\vec{e}_1, \vec{e}_2, \vec{e}_3$  відповідно, отримаємо  $a_1 \vec{e}_1 = \lambda b_1 \vec{e}_1, a_2 \vec{e}_2 = \lambda b_2 \vec{e}_2, a_3 \vec{e}_3 = \lambda b_3 \vec{e}_3$ . Додамо ці рівності та одержимо  $a_1 \vec{e}_1 + a_2 \vec{e}_2 + a_3 \vec{e}_3 = \lambda b_1 \vec{e}_1 + \lambda b_2 \vec{e}_2 + \lambda b_3 \vec{e}_3$  або  $a_1 \vec{e}_1 + a_2 \vec{e}_2 + a_3 \vec{e}_3 = \lambda(b_1 \vec{e}_1 + b_2 \vec{e}_2 + b_3 \vec{e}_3)$ , тобто  $\vec{a} = \lambda \vec{b} \Rightarrow \vec{a} \parallel \vec{b}$ . Теорему доведено.

### Скалярний добуток векторів

Розглянемо  $\vec{a}, \vec{b}$  – ненульові вектори. Відкладемо від якої-небудь точки  $O$  вектори  $\vec{OA} = \vec{a}, \vec{OB} = \vec{b}$ . Кутом між векторами  $\vec{a}$  і  $\vec{b}$  називають кут між променями  $OA$  і  $OB$  (див. рис. 2.14). Використовується позначення:  $(\vec{a}, \vec{b}) = \varphi$ . Для будь-котрих векторів  $\vec{a}$  і  $\vec{b}$  маємо  $0 \leq (\vec{a}, \vec{b}) \leq \pi$ .

Рисунок 2.14 – Кут між векторами  $\vec{a}$  і  $\vec{b}$ 

**Означення:** скалярним добутком двох векторів називається число, яке дорівнює добутку їх довжин та косинусу кута між ними:  $\vec{a} \cdot \vec{b} = |\vec{a}| |\vec{b}| \cos(\vec{a}, \vec{b})$ .

**Теорема:** формула для обчислення скалярного добутку векторів  $\vec{a}(a_1, a_2, a_3)$ ,  $\vec{b}(b_1, b_2, b_3)$ , заданих в ортонормованому базисі, має вигляд:

$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + a_3 b_3. \quad (2.4)$$

*Доведення.* В випадку, коли один із векторів або обидва дорівнюють нулю, то доведення формули є очевидним. Нехай  $\vec{a} \neq \vec{0}$ ,  $\vec{b} \neq \vec{0}$ . Будемо розглядати два випадки.

1. Нехай вектори  $\vec{a}$  і  $\vec{b}$  є неколінеарними. Відкладемо вектори  $\vec{OA} = \vec{a}$ ,  $\vec{OB} = \vec{b}$  від деякої точуи  $O$  (див. рис. 2.15). Визначимо  $(\vec{a}, \vec{b}) = \varphi$ .

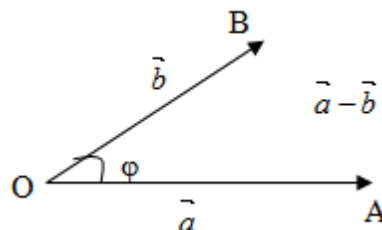


Рисунок 2.15 – Доведення теореми

Розглянемо  $\triangle OAB$ . Згідно теоремі косинусів

$$AB^2 = OA^2 + OB^2 - 2 OA \cdot OB \cos \varphi, \text{ або } |\vec{a} - \vec{b}|^2 = |\vec{a}|^2 + |\vec{b}|^2 - 2 \vec{a} \cdot \vec{b},$$

$$\text{звідси } 2 \vec{a} \cdot \vec{b} = |\vec{a}|^2 + |\vec{b}|^2 - |\vec{a} - \vec{b}|^2 = a_1^2 + a_2^2 + a_3^2 + b_1^2 + b_2^2 + b_3^2 - (a_1 - b_1)^2 - (a_2 - b_2)^2 - (a_3 - b_3)^2 =$$

$$= 2a_1 b_1 + 2a_2 b_2 + 2a_3 b_3. \text{ Таким чином, } \vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + a_3 b_3.$$

2. Нехай вектори  $\vec{a}$  і  $\vec{b}$  є колінеарними. В такому випадку

$$\vec{a} = \lambda \vec{b}, \quad a_1 = \lambda b_1, \quad a_2 = \lambda b_2, \quad a_3 = \lambda b_3;$$

$$\vec{a} \vec{b} = \lambda \vec{b} \vec{b} = |\lambda \vec{b}| |\vec{b}| \cos(\lambda \vec{b}, \vec{b}) = \lambda |\vec{b}|^2 = \lambda (b_1^2 + b_2^2 + b_3^2) = \lambda b_1 b_1 + \lambda b_2 b_2 + \lambda b_3 b_3 = a_1$$

$$b_1 + a_2 b_2 + a_3 b_3$$

Теорему доведено.

З теореми і означення витікають такі властивості скалярного добутку векторів:

1. У випадку  $\vec{a} \neq \vec{0}$ ,  $\vec{b} \neq \vec{0}$   $\vec{a} \vec{b} = 0$  тоді і тільки тоді, коли  $\vec{a} \perp \vec{b}$ .

$$2. \vec{a} \vec{a} \stackrel{def}{=} \vec{a}^2 = |\vec{a}|^2 \Rightarrow |\vec{a}| = \sqrt{\vec{a}^2}.$$

$$3. \vec{a} \vec{b} = \vec{b} \vec{a}.$$

$$4. (\alpha \vec{a}) \vec{b} = \alpha (\vec{a} \vec{b}), \forall \alpha \in \mathbb{R};$$

$$5. (\vec{a} + \vec{b}) \vec{c} = \vec{a} \vec{c} + \vec{b} \vec{c}.$$

Формула, аналогічна до формули (2.4), виконується і в просторі  $V_2$ . Дійсно, задамо в ортонормованому базисі простору  $V_2$  вектори  $\vec{a}(a_1, a_2)$ ,  $\vec{b}(b_1, b_2)$ . Відтак, використовуючи властивості 1–5, отримаємо:  $\vec{a} \vec{b} = (a_1 \vec{i} + a_2 \vec{j})(b_1 \vec{i} + b_2 \vec{j}) = a_1 b_1 \vec{i}^2 + (a_1 b_2 + a_2 b_1) \vec{i} \vec{j} + a_2 b_2 \vec{j}^2 = a_1 b_1 + a_2 b_2$ . Таким чином,

$$\vec{a} \vec{b} = a_1 b_1 + a_2 b_2 \quad (2.5)$$

З означення скалярного добутку і (2.4), (2.5) витікають такі формули для визначення косинуса кута між векторами:

– у просторі  $V_3$ :

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} \vec{b}}{|\vec{a}| |\vec{b}|} = \frac{a_1 b_1 + a_2 b_2 + a_3 b_3}{\sqrt{a_1^2 + a_2^2 + a_3^2} \sqrt{b_1^2 + b_2^2 + b_3^2}};$$

– у просторі  $V_2$ :

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} \vec{b}}{|\vec{a}| |\vec{b}|} = \frac{a_1 b_1 + a_2 b_2}{\sqrt{a_1^2 + a_2^2} \sqrt{b_1^2 + b_2^2}}.$$

Векторну алгебру можна успішно використовувати для побудови розв'язків задач елементарної геометрії.

## **2.2. Інструменти для розробки клієнтської частини додатку**

Для розробки клієнтської частини додатку будуть використани мови програмування HTML, CSS, JavaScript, бібліотека JQuery, технологія Ajax, фреймворк Bootstrap та додаток Adobe Photoshop.

### **2.2.1 Мова розмітки гіпертексту HTML**

*HTML* (від англ. HyperText Markup Language – «мова розмітки гіпертексту») – це базова мова для створення веб-сторінок, головним призначенням якої є структуроване відображення інформації у веб-браузерах, таких як Google Chrome. Використання HTML забезпечує можливість представлення на сторінці різноманітного контенту: текстових блоків, гіперпосилань, графічних елементів та інших компонентів.

Особливістю цієї мови є те, що її вихідний код не потребує трансляції у двійковий формат, адже HTML безпосередньо інтерпретується браузером. Водночас такий принцип роботи накладає певні обмеження на функціональні можливості мови та створює труднощі із сумісністю нових конструкцій у старих версіях браузерів.

Сучасні веб-сторінки рідко обмежуються лише використанням HTML. Для їхньої повноцінної роботи та привабливого оформлення застосовують засоби динамічного HTML, серед яких скрипти на JavaScript, каскадні таблиці стилів (CSS), а також іноді Java-аплети.

Структурною основою HTML є елементи, які складаються з тегів, атрибутів і вмісту (останній може бути відсутнім).

- Теги – це спеціальні службові позначення, які вказують браузеру, як саме потрібно відображати певний елемент.

- ✓ Відкриваючий тег визначає початок елемента і записується у кутових дужках. Наприклад, тег абзацу має вигляд `<p>`, а для зображення використовується тег `<img>`.
- ✓ Закриваючий тег сигналізує про завершення елемента і має аналогічний запис, але з косою рисою перед назвою, наприклад `</p>`. Слід зауважити, що окремі елементи, такі як `<img>`, не потребують закриваючого тегу.
- Атрибути застосовуються для уточнення параметрів елементів. Завдяки їм можна визначати характеристики, наприклад висоту й ширину зображень. Це реалізується у вигляді коду, подібного до наступного прикладу:  
`<img width="100" height="100">`.

### 2.2.2 Каскадні таблиці стилів CSS

Візуальне оформлення веб-сторінки, тобто вибір шрифту, розташування текстових блоків, позиціонування різних об'єктів та загальний вигляд інтерфейсу, забезпечується за допомогою каскадних таблиць стилів – CSS (Cascading Style Sheets). Основне призначення CSS полягає у розширенні можливостей HTML, надаючи веб-розробникам гнучкі засоби для якісного дизайну веб-сторінок.

Поява CSS суттєво змінила підхід до веб-розробки. Тепер створювачі сайтів можуть формувати зовнішній вигляд сторінки відповідно до власного бачення. Це відкриває широкі можливості для творчого оформлення, хоча водночас створює і певні труднощі, пов'язані з неоднорідністю стилів та можливими конфліктами.

Існує кілька способів додавання CSS-стилів у структуру веб-сторінки:

- Внутрішні стилі. Вони задаються безпосередньо всередині конкретного тегу й впливають лише на нього. Такий метод є найпростішим у застосуванні, проте не відзначається високою гнучкістю. Наприклад:  
`<h1 style="font-size: 20px;"></h1>`.
- Стилі в блоці `<style>`. У цьому випадку стилі описуються у спеціальному блоці коду, що розміщується всередині секції `<head>` HTML-документа. Цей

підхід дозволяє одночасно застосовувати стилі до кількох елементів сторінки, проте втрачає ефективність у масштабних веб-проєктах. Приклад оголошення стилю:

```
<head>
  <style>
    h1 { color: blue; }
  </style>
</head>
```

- Зовнішній файл. Найбільш ефективний метод, що передбачає створення окремого файлу зі стилями з розширенням .css. Один файл може керувати відображенням великої кількості сторінок, що робить його використання зручним і масштабованим. Для підключення достатньо додати у код посилання на цей файл, і всі внесені до нього зміни автоматично впливатимуть на вигляд сайту.

При роботі з HTML і CSS необхідно враховувати, що ці дві технології виконують різні функції:

- HTML відповідає за логічну структуру сторінки: визначає порядок абзаців, вставку зображень, побудову таблиць і семантичне значення окремих елементів.
- CSS керує фізичним відображенням сторінки: визначає шрифт тексту, кольори заголовків, стилі таблиць, рамки тощо.

Варто зауважити, що CSS у своїй основі є статичною мовою стилізації: він визначає вигляд сторінки під час її завантаження. Проте, використовуючи JavaScript, можна динамічно змінювати властивості CSS, роблячи сторінку інтерактивною.

Для підвищення гнучкості стилізації веб-сторінок у розробці використовуються препроцесори, серед яких найбільш відомим є *LESS*. Це інструмент, що трансформує код із власного синтаксису у стандартний CSS. LESS може як розширювати можливості CSS (додаючи нові конструкції), так і заміщати

деякі його елементи. На виході завжди отримується звичайний CSS-код, який легко інтерпретується браузерами.

Перевага LESS полягає у підтримці змінних, математичних операцій, модульності стилів, функцій та міксинів, що робить роботу з великими таблицями стилів набагато зручнішою.

Використовувати LESS можна двома основними способами:

1. Підключити файл із розширенням .less до HTML-сторінки та здійснити його компіляцію «на льоту» за допомогою JavaScript. Для цього необхідно завантажити скрипт less.js і підключити його, як звичайний JS-файл:

```
<script src="less.js" type="text/javascript"></script>
```

Після цього створений файл .less можна підключити таким чином:

```
<link rel="stylesheet/less" type="text/css" href="style.less">
```

2. Скомпілювати .less-код заздалегідь у стандартний .css-файл і підключати його, як звичайний файл стилів.

У результаті використання LESS веб-розробник отримує більш гнучкий і структурований інструмент для організації стилів, що значно полегшує підтримку та масштабування веб-проектів.

### **2.2.3 Динамічна мова програмування JavaScript та бібліотека JQuery**

JavaScript (JS) – це динамічна повноцінна мова програмування, яка використовується у поєднанні з HTML-документами для надання веб-сторінкам інтерактивності. Автором мови є Брендан Айх (Brendan Eich), співзасновник проєктів Mozilla, Mozilla Foundation і Mozilla Corporation.

На відміну від багатьох інших мов, JavaScript із самого початку розроблявся з метою інтеграції інтерактивних можливостей у веб-додатки. Код JavaScript виконується безпосередньо у браузері користувача, що зменшує навантаження на сервер і підвищує швидкість обробки подій.

Основна роль JavaScript полягає у забезпеченні взаємодії веб-ресурсу з користувачем:

- перевірка та аналіз введених даних;

- валідація форм;
- динамічна зміна контенту сторінки;
- відображення повідомлень, діалогових та модальних вікон;
- реагування на дії користувача у режимі реального часу.

Щоб інтегрувати JavaScript у HTML-документ, необхідно додати два ключові компоненти: сам скрипт і процедуру його запуску. Тіло скрипта зазвичай розташовується в контейнері `<head>`. Наприклад:

```
<head>  
  <script>  
    function what() { }  
  </script>  
</head>
```

Активація скрипта відбувається через події, що визначаються у блоці `<body>`. Скрипт може запускатися у відповідь на взаємодію з елементами інтерфейсу: наприклад, наведення курсора миші чи натискання кнопки.

Популярність JavaScript пояснюється широкими можливостями роботи з елементами сторінки без необхідності її повторного завантаження. Це дозволяє реалізовувати приховування або відображення фрагментів дизайну, переміщення об'єктів, зміну їхнього зовнішнього вигляду. Завдяки цьому створюються ефекти презентаційного типу, інтерактивні меню, ігрові елементи, а також забезпечується зручна обробка даних у формах.

Мова також підтримує роботу з cookies – невеликими текстовими файлами, що зберігаються на комп'ютері користувача. За їх допомогою можна фіксувати дату останнього відвідування, зберігати паролі чи інші дані про взаємодію користувача із сайтом. Це дозволяє персоналізувати веб-ресурс і зробити його комфортнішим для відвідувача.

JavaScript містить усі базові арифметичні операції, підтримує стандартні математичні функції та може працювати як із цілими числами, так і з числами з плаваючою комою.

### Переваги JavaScript:

- підтримка усіма сучасними веб-браузерами;
- простота використання навіть для новачків завдяки готовим скриптам і плагінам;
- постійний розвиток та вдосконалення (наприклад, проєкт JavaScript 2.0);
- можливість взаємодії з різними програмами, включаючи текстові редактори Microsoft Office та OpenOffice.

### Недоліки JavaScript:

- знижений рівень безпеки через відкритість вихідного коду більшості скриптів;
- наявність численних дрібних помилок під час роботи, що зменшує професійність мови у порівнянні з іншими;
- залежність від JavaScript багатьох додатків: їхня робота стає неможливою без підтримки мови.

Для полегшення створення клієнтської частини сайту використовуються бібліотеки з готовими наборами функцій і компонентів. Найпоширенішою є jQuery – безкоштовна бібліотека, що містить набір зручних методів для вирішення стандартних завдань на JavaScript, значно прискорюючи процес розробки.

jQuery – це швидка, компактна та багатофункціональна бібліотека для мови JavaScript, що зазвичай розповсюджується у вигляді одного файлу з розширенням .js. Вона значно спрощує роботу веб-розробників, адже містить велику кількість готових функцій, за допомогою яких можна легко виконувати поширені завдання.

### *Ключові можливості jQuery:*

- Робота з DOM. Бібліотека надає гнучкі селектори, що дозволяють знаходити елементи DOM за різними критеріями: назвою тега, ідентифікатором, класом CSS, атрибутами, їхніми значеннями чи позицією в ієрархії.
- Маніпуляція DOM. Завдяки вбудованим методам можна змінювати вміст HTML-елементів, додавати або видаляти їх, редагувати класи CSS тощо.

- Анімації та спецефекти. jQuery забезпечує просту реалізацію візуальних ефектів: плавне показування та приховування елементів, зміна прозорості, слайд-ефекти, анімація та інші графічні трансформації.
- Обробка подій. Бібліотека має зручні методи для роботи з подіями, такими як click, dblclick, mouseenter, mouseleave, blur, keyup, keydown тощо. При цьому jQuery автоматично вирішує проблеми сумісності між різними браузерами.
- Робота з AJAX. jQuery містить прості у використанні інструменти для асинхронного завантаження даних із сервера без потреби оновлення всієї веб-сторінки.
- Кросбраузерність. Одна з найважливіших переваг jQuery – її здатність однаково працювати в різних браузерах (Internet Explorer 6.0+, Firefox 2.0+, Safari 3.0+, Chrome, Opera 9.0+), знімаючи з розробника необхідність у ручному налаштуванні сумісності.

Переваги використання jQuery:

- Простота у вивченні. jQuery базується на стандартному JavaScript-синтаксисі, тому її легко опанувати навіть початківцям.
- Висока продуктивність. Завдяки великій кількості готових функцій розробники пишуть менше коду, що прискорює процес створення проєкту.
- Якісна документація. Офіційний сайт jQuery містить детально описаний API з численними прикладами використання.
- Ненав'язливість. Використання jQuery не порушує структуру HTML-коду, дозволяючи чітко розділяти розмітку сторінки та програмну логіку.

#### **2.2.4 Технологія взаємодії з сервером без перезавантаження сторінки AJAX**

AJAX (Asynchronous JavaScript and XML) – це технологія, яка дозволяє організувати обмін даними з сервером без необхідності повного перезавантаження веб-сторінки. Завдяки цьому значно зростає швидкість роботи з веб-сайтами та підвищується комфорт користувача.

Основний принцип роботи AJAX зображено на рисунку 2.16.

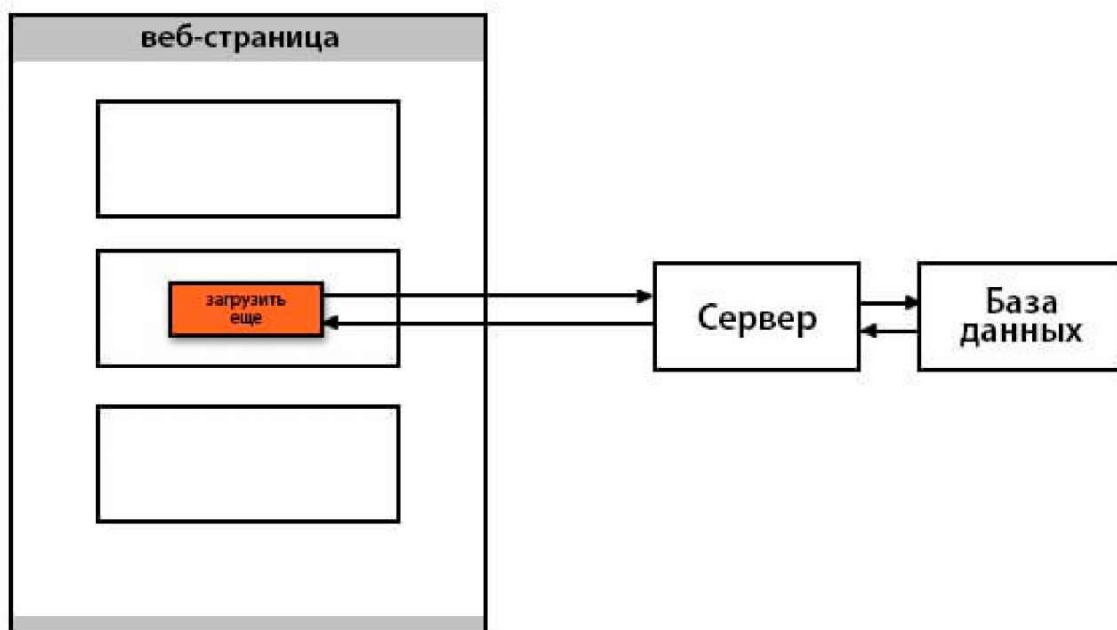


Рисунок 2.16 – Основний принцип роботи AJAX

Принцип дії AJAX можна описати поетапно.

1. Ініціація запиту. Користувач активує AJAX-запит, наприклад, натискаючи кнопку для завантаження додаткових даних.

2. Формування запиту. Система відправляє на сервер запит разом із необхідними параметрами (наприклад, запит на отримання певного файлу або витяг конкретної інформації з бази даних).

3. Обробка на сервері. Сервер, отримавши запит, опрацьовує його та формує відповідь, яка базується на даних із бази.

4. Обробка відповіді. Відповідь повертається у браузер, де JavaScript її розпізнає та відображає кінцевому користувачеві.

Для організації такого процесу використовується спеціальний об'єкт XMLHttpRequest, що виконує роль посередника між клієнтською частиною (браузером) та сервером. Запити можуть здійснюватися у двох основних формах:

- GET – застосовується для звернення до документа на сервері, де у параметрах передається URL-адреса ресурсу;

- POST – використовується для передачі великих обсягів даних, наприклад, при надсиланні форм.

Щоб уникнути помилок у процесі звернень, у JavaScript можна застосовувати спеціальні функції, наприклад ``escape()``, яка запобігає перериванню запиту.

Особливістю AJAX є асинхронність: користувач може продовжувати взаємодіяти зі сторінкою, поки в «фоновому режимі» триває обмін даними із сервером. При цьому користувачу зазвичай надається візуальне повідомлення (наприклад, індикатор завантаження), щоб уникнути враження, ніби сторінка «зависла».

Як відповідь сервер може надсилати:

- текстовий формат – відразу відображається на сторінці;
- XML – перед відображенням конвертується у HTML;
- JSON – обробляється як JavaScript-код, після чого формується об'єкт, зручний для подальшого використання у скриптах.

Таким чином, технологія AJAX забезпечує динамічну взаємодію між клієнтом і сервером, підвищуючи інтерактивність та ефективність роботи сучасних веб-додатків.

Переваги використання AJAX:

- Оптимізація трафіку. Обсяг даних, що передається між клієнтом і сервером, істотно зменшується, адже немає потреби щоразу завантажувати всю сторінку повністю. Достатньо отримати лише змінений фрагмент або необхідний набір даних, після чого JavaScript оновлює відповідну частину інтерфейсу в браузері.
- Зниження навантаження на сервер. Завдяки правильному застосуванню AJAX зменшується кількість однотипних звернень до сервера. Наприклад, постійні елементи сайту (шапка, логотип, меню) не перезавантажуються, а динамічно оновлюється лише змінний контент. При голосуванні на сайті користувач обирає варіант і натискає кнопку – інформація передається на сервер і відповідь повертається без перезавантаження сторінки.

- Підвищення швидкодії сервісу. Оскільки завантажується лише змістовна частина сторінки, користувач отримує результат своїх дій значно швидше.
- Розширені можливості інтерактивності. AJAX не обмежується обробкою форм. Він дозволяє, наприклад, перевіряти доступність логіна під час реєстрації або формувати підказки в пошукових системах (як це реалізовано в Google, коли після введення кожної букви з'являються варіанти пошукових запитів). Це підвищує зручність та ефективність взаємодії з веб-ресурсом.

Недоліки AJAX:

- Залежність від JavaScript. Якщо у браузері вимкнена підтримка JavaScript, сторінки з AJAX не зможуть функціонувати належним чином. Крім того, такі сторінки гірше індексуються пошуковими системами та не завжди дозволяють створювати закладки.
- Обмежена інтеграція з інструментами браузера. При динамічному оновленні вмісту браузер не фіксує ці зміни в історії відвідувань, тому кнопка «Назад» не завжди працює коректно. Подібні проблеми частково вирішуються додатковими скриптами. Також ускладнюється створення закладок для конкретних частин сторінки.
- Зниження рівня безпеки. Вихідний код AJAX-додатків можна переглянути у браузері, що потенційно створює додаткові вразливості.
- Проблеми зі статистикою відвідувань. Динамічне завантаження контенту ускладнює збір аналітичних даних: стандартні лічильники не завжди фіксують фактичні переходи між сторінками, що може призводити до заниження показників трафіку й відповідно – доходів від реклами.

Крім того, сайти, що активно використовують AJAX, іноді отримують нижчі позиції в пошуковій видачі. Це пов'язано з низкою причин:

- пошукові боти можуть не врахувати весь динамічний контент;
- всі підвантажені сторінки можуть мати однакову URL-адресу;
- індексація може відображати інший вміст, ніж бачить реальний користувач.

Проте таких негативних наслідків можна уникнути, якщо використовувати AJAX лише за призначенням – для реалізації динамічної взаємодії з сервером. Наприклад, основні текстові блоки з ключовими словами слід розташовувати статично на початку сторінки. Динамічні елементи можна кешувати, щоб відображати їх як статичні. Крім того, рекомендується запускати AJAX-запити через класичні посилання-якорі, а не виключно через події `onClick`.

Отже, грамотне застосування AJAX дозволяє суттєво підвищити швидкодію та зручність веб-сайтів, одночасно зберігаючи їхню індексацію в пошукових системах та оптимізуючи роботу серверів.

### 2.2.5 Фреймворк Bootstrap

Bootstrap – це безкоштовний і відкритий програмний фреймворк, призначений для створення веб-сайтів і веб-додатків. Його функціонал базується на технологіях HTML, CSS та JavaScript, що суттєво полегшує процес розробки адаптивних, орієнтованих насамперед на мобільні пристрої інтерфейсів.

Архітектура Bootstrap зводиться до трьох основних складових:

- bootstrap.css – каскадні таблиці стилів, що визначають зовнішнє оформлення сторінок;
- bootstrap.js – набір JavaScript-скриптів та інструментів для взаємодії з бібліотекою jQuery;
- набір шрифтів та іконок, що забезпечує графічні елементи інтерфейсу.

Оскільки Bootstrap використовує можливості jQuery, ця бібліотека JavaScript є обов'язковою для його коректної роботи.

Особливу увагу у фреймворку приділено адаптивному дизайну. Він дозволяє веб-сторінкам автоматично визначати параметри пристрою користувача (розмір і орієнтацію екрана) та підлаштовувати свій вигляд відповідно до них. Такий підхід, відомий як mobile-first («спочатку для мобільних»), передбачає, що смартфони, планшети та спеціалізовані мобільні додатки стають ключовими інструментами роботи, а отже, вимоги цих платформ повинні враховуватися у дизайні з самого початку.

Bootstrap включає в себе широкий набір готових компонентів користувацького інтерфейсу, шаблони макетів, інструменти для роботи з JavaScript, а також механізми для швидкої інтеграції у веб-проекти. Розробники можуть завантажити готові збірки або скористатися вихідним кодом для індивідуальної конфігурації.

Фреймворк був створений Марком Отто та Джейкобом Торнтоном у компанії Twitter. Спочатку він мав назву Twitter Blueprint і був внутрішнім інструментом для уніфікації стилів і зменшення витрат на технічне обслуговування. Пізніше проект отримав сучасну назву – Twitter Bootstrap (сьогодні відомий просто як Bootstrap) і став загальнодоступним.

Термін bootstrap у комп'ютерній термінології означає процес завантаження системи, коли запуск великої програми (наприклад, операційної системи) відбувається за допомогою невеликої початкової програми, що ініціює цей процес.

Переваги Bootstrap:

- Адаптивність. Завдяки вбудованій системі сіток і шаблонів дизайн автоматично підлаштовується під будь-які розміри екранів – від смартфонів до великих моніторів.
- Швидкість розробки. Bootstrap надає готові компоненти (кнопки, форми, меню, модальні вікна тощо), що дозволяє значно скоротити час створення інтерфейсу.
- Кросбраузерність. Усі елементи фреймворку коректно працюють у сучасних браузерах, що забезпечує однаковий вигляд і функціональність веб-сторінок для користувачів.
- Єдиний стиль. Використання уніфікованих компонентів дозволяє розробникам створювати візуально цілісні та гармонійні інтерфейси.
- Широка підтримка та документація. Bootstrap має велику спільноту користувачів, регулярні оновлення та докладну офіційну документацію. Це робить його доступним навіть для початківців.
- Розширюваність. При необхідності розробники можуть модифікувати стандартні стилі, додавати власні шаблони та інтегрувати інші бібліотеки.

### Недоліки Bootstrap:

- Типовість дизайну. Якщо не вносити зміни у стандартні стилі, сайти на Bootstrap можуть виглядати схожими між собою, що знижує унікальність проєкту.
- Надлишковий код. Багато вбудованих класів і стилів можуть залишатися невикористаними, що збільшує «вагу» веб-сторінки. Для невеликих проєктів це може стати зайвим навантаженням.
- Обмеження у кастомізації. Хоча Bootstrap підтримує зміни стилів, значне відхилення від стандартних шаблонів вимагає додаткових зусиль і знань CSS.
- Залежність від jQuery. Для коректної роботи деяких компонентів необхідно підключати бібліотеку jQuery, що не завжди зручно при переході на сучасні фреймворки.
- Можливі проблеми з продуктивністю. У великих проєктах з великою кількістю підключених компонентів Bootstrap може дещо уповільнювати завантаження сторінки.

### 2.2.6 Додаток Adobe Photoshop

Adobe Photoshop – це професійне програмне забезпечення, яке широко використовується для редагування растрових зображень, створення графічного дизайну та цифрового мистецтва. Його ключовою особливістю є робота з багатошаровістю, що надає гнучкість і глибину у процесі створення та обробки графіки. У поєднанні з потужними інструментами редагування програма дозволяє досягати практично необмежених результатів.

Історія Photoshop почалася у 1989 році, коли брати Томас і Джон Нолли розробили програму для обробки зображень. Того ж року Джон Нолл продав її компанії Adobe Systems, яка випустила продукт під назвою «Photoshop». Відтоді він став де-факто стандартом у сфері редагування растрової графіки. Сьогодні Photoshop розповсюджується для операційних систем macOS та Windows, однак офіційної підтримки Linux немає.

Основне призначення Photoshop – робота з цифровими фотографіями та растровими зображеннями. Серед його переваг варто виділити:

- багатий набір інструментів для створення та редагування графіки;
- високу якість обробки зображень;
- зручність використання та зрозумілий інтерфейс;
- широкі можливості автоматизації завдяки сценаріям та підтримці кольорових профілів;
- великий набір фільтрів для створення художніх ефектів та спецефектів.

Для роботи з окремими частинами зображення у програмі передбачено різні методи виділення: за фігурою, за діапазоном кольорів або у ручному режимі. Крім того, Photoshop пропонує численні фільтри для стилізації, деформації та імітації художніх технік (розмиття, текстуризація тощо).

Програма містить інструменти цифрового живопису – від стандартних пензлів до підтримки сторонніх бібліотек пензлів, стилів, палітр і шрифтів. Це робить її корисною не лише у поліграфії, для якої вона спершу створювалася, а й у веб-дизайні, анімації та цифровому мистецтві.

Photoshop також інтерпується з іншими продуктами Adobe – Illustrator, Premiere Pro, After Effects, що дозволяє застосовувати його для професійного відеомонтажу, створення спецефектів, текстур, фонів та анімації. Основний робочий формат файлів – PSD, підтримується експорт та імпорт у більшості програм сімейства Adobe.

Важливою особливістю є робота з шарами, які дають змогу поєднувати текст, графіку, фото та інші елементи у межах одного проєкту. Photoshop також має розвинені можливості роботи з текстами, фігурами, стилями та контурами, що робить його універсальним інструментом як для художників, так і для дизайнерів.

Переваги Adobe Photoshop:

- Широкий набір інструментів. Програма містить усе необхідне для редагування фотографій, створення графіки, цифрового живопису та дизайну.

- Підтримка шарів. Робота з багатошаровістю забезпечує гнучкість у редагуванні та дозволяє створювати складні композиції.
- Висока якість обробки. Photoshop застосовується в поліграфії, веб-дизайні, кінематографії та інших сферах, що потребують професійної точності.
- Автоматизація процесів. Можливість використання скриптів та дій (actions) значно прискорює виконання рутинних операцій.
- Розширюваність. Користувачі можуть встановлювати додаткові пензлі, стилі, фільтри та плагіни, що розширює стандартний функціонал.
- Інтеграція з іншими продуктами Adobe. Програма сумісна з Illustrator, Premiere Pro, After Effects та іншими, що робить її частиною єдиної екосистеми.
- Міжнародний стандарт. Photoshop визнано провідним інструментом у сфері професійної обробки зображень, тому його знання підвищує конкурентоспроможність спеціаліста.

#### Недоліки Adobe Photoshop:

- Висока вартість. Ліцензійна версія програми є дорогою, що може стати перешкодою для початківців або невеликих студій.
- Вимогливість до ресурсів. Для комфортної роботи необхідні потужні комп'ютери з достатнім обсягом оперативної пам'яті та продуктивною відеокартою.
- Крива навчання. Через багатофункціональність і складність інтерфейсу новачкам потрібно багато часу, щоб освоїти програму.
- Відсутність офіційної підтримки Linux. Photoshop доступний лише для Windows та macOS, що обмежує його використання.
- Надлишковість функцій для базових задач. У багатьох випадках для простого редагування фотографій можливостей Photoshop забагато, і більш легкі програми можуть бути ефективнішими.

## 2.3 Інструменти для розробки серверної частини додатку

Для розробки серверної частини додатку будуть використані мова програмування PHP, сервер OpenServer, СКБД MySQL та CMS BlizAnt.

### 2.3.1 Мова програмування PHP

*PHP* виявився оптимальним вибором серед мов програмування для реалізації серверної частини, адже найбільш точно відповідає економічним вимогам та критеріям трудомісткості.

PHP – це мова програмування, орієнтована на створення вебсайтів. Вона надає можливість автоматизувати процеси роботи ресурсу. PHP є скриптовою мовою, інтегрованою в HTML, код якої інтерпретується та виконується безпосередньо на сервері.

Основна відмінність між PHP та JavaScript полягає в тому, що сценарії PHP обробляються на сервері, після чого користувачу надсилається вже готовий результат. Натомість JavaScript-код передається повністю на клієнтський комп'ютер і виконується безпосередньо там.

Наприкінці минулого століття для створення вебресурсу «світового рівня» було достатньо володіння HTML та мінімальними дизайнерськими навичками. Згодом вимоги до сайтів значно ускладнилися: використання JavaScript і Dynamic HTML стало звичним явищем, а сторінки без подібних динамічних елементів виглядали застарілими.

Невдовзі «правилом хорошего тону» для корпоративних вебресурсів стало застосування складних програмних рішень на основі Perl або C++. Проте більшість вебдизайнерів не прагнули працювати з надмірно складними мовами. Саме в цей період з'явилася мова PHP – інструмент, який поєднав у собі можливості потужних скриптових технологій із простотою у вивченні та використанні.

PHP належить до мов сценаріїв, що інтегруються у вебсередовище. Її основна мета – надати розробникам можливість швидко створювати динамічно згенеровані сторінки. Сучасний вебресурс не обмежується лише текстом і зображеннями: він має забезпечувати певний рівень інтерактивності з користувачем – від пошуку

інформації та продажу товарів до організації конференцій і зворотного зв'язку. PHP функціонує як частина вебсервера, не потребує суворої типізації даних і звільняє програміста від ручного управління пам'яттю. Код на PHP є легким для читання й сприйняття навіть для тих, хто лише починає працювати з цією мовою.

На PHP можна реалізувати практично будь-які завдання: від обробки даних, отриманих із вебформ, до створення динамічних сторінок та організації роботи з сесіями й куками (cookies). Крім того, мова має вбудовану підтримку широкого спектра баз даних, що робить розробку вебзастосунків із використанням БД максимально зручною та простою.

Додатковою перевагою є підтримка протоколів IMAP, SNMP, NNTP, POP3, а також HTTP. PHP дозволяє працювати із сокетом (sockets) і взаємодіяти з іншими мережевими протоколами. Завдяки цьому мова значно полегшує роботу розробника. Наприклад, використання команди `'include()'` дає можливість автоматично додавати один і той самий HTML-фрагмент на численні сторінки сайту. У результаті достатньо змінити лише вихідний файл із загальною частиною, щоб оновлення відобразилися на всіх сторінках. Аналогічно, при редагуванні новин змінюється лише їхній текст, а не структура HTML-документа, що робить процес зручнішим як для самого автора, так і для менш досвідчених помічників.

Функціонал PHP не обмежується лише допомогою вебмайстру чи технічній підтримці ресурсу. Ця мова дозволяє значно розширити можливості сайту, забезпечуючи його інтеграцію з додатковими функціями. Зокрема, за допомогою функції `'mail()'` можна реалізувати відправлення електронних повідомлень безпосередньо зі сторінок сайту.

Серед основних переваг PHP варто виокремити:

- створення та налагодження скриптів значно простіше у порівнянні з іншими мовами програмування;
- завдяки тому, що PHP-код легко вбудовується в HTML-документ, немає необхідності у використанні спеціалізованих IDE;
- відпадає потреба у написанні численних CGI-програм, що скорочує час доступу до сторінок, а також зменшує загальну тривалість розробки ресурсу.

Сьогодні PHP вважається однією з найперспективніших мов програмування для мережі Інтернет. Його популярність постійно зростає завдяки широкій підтримці різних технологій, сумісності з більшістю серверів і баз даних, простоті використання та відкритій ліцензії.

### 2.3.2 Сервер OpenServer

OpenServer – це безкоштовний програмний комплекс для веброзробників, що містить набір серверних компонентів. Його основне призначення полягає у створенні та тестуванні повноцінних сайтів на локальному комп'ютері. Інакше кажучи, за допомогою OpenServer можна розгорнути середовище, подібне до Linux-сервера, але під операційною системою Windows, та безперешкодно запускати вебресурси, зокрема написані на PHP.

Програма вирізняється простотою у використанні. Для її запуску немає потреби у ручному налаштуванні Apache чи Nginx або в конфігурації MySQL – усі ці дії система виконує автоматично. До того ж OpenServer реалізовано у вигляді портативного застосунку, що не потребує інсталяції. Він може запускатися безпосередньо з флеш-накопичувача, жорсткого диска чи навіть CD/DVD. У випадку використання з оптичних дисків програма автоматично створює тимчасовий каталог із даними на диску або у віртуальному пристрої. Також передбачено консольне керування та можливість створення власних збірок середовища.

При необхідності OpenServer може працювати як сервер у локальній або глобальній мережі. Якщо ж він використовується як інтернет-сервер, потрібен статичний IP-адрес, а також варто подбати про безпеку комп'ютера та захист від шкідливих скриптів.

Базовий набір компонентів OpenServer включає:

- сервер Apache;
- DNS-сервер Bind;
- HTTP-сервер Nginx;
- системи керування базами даних MySQL, MariaDB, MongoDB, PostgreSQL;

- сховище даних із журналюванням Redis;
- систему кешування Memcached;
- FTP-сервер FileZilla;
- середовище для виконання PHP-скриптів (PHP 5.x);
- інтерпретатор PostScript Ghostscript;
- поштовий сервіс Sendmail;
- допоміжні утиліти: HeidiSQL, Adminer, RockMongo, phpMyAdmin та інші.

Програмний комплекс представлений у трьох версіях: Basic, Premium та Ultimate. Основна відмінність між ними полягає в кількості додаткових утиліт. У базовій редакції доступний лише мінімально необхідний набір для запуску сервера. Версія Premium, крім основних серверних програм, містить додаткові засоби: Git, ImageMagick, MongoDB, Rockmongo, PostgreSQL та PhpPgAdmin. У версії Ultimate передбачено значно розширений набір вільнопоширюваного програмного забезпечення, зокрема інструменти для роботи з графікою, текстові та HTML-редактори, утиліти для роботи з інтернетом, менеджери завантажень, SCP-клієнти тощо.

OpenServer також може функціонувати в обмеженому режимі у випадках, коли користувач не має можливості змінювати файл HOSTS через обмеження прав доступу або блокування з боку фаєрвола. У такій ситуації вебсайти будуть доступні лише за локальною адресою 127.0.0.1 (localhost).

### 2.3.3 СКБД MySQL

Система управління базами даних MySQL – це програмний комплекс, призначений для створення структури нової бази даних, її заповнення, редагування інформаційного вмісту та подальшого відображення даних.

Основні переваги системи управління базами даних MySQL:

- Відкритий програмний код. Кожен користувач має можливість безкоштовно завантажити систему з офіційного ресурсу розробника. Водночас для використання системи у комерційних програмних продуктах необхідно придбати ліцензійну версію у компанії MySQL AB.

- Підтримка роботи на різних операційних системах, що робить її універсальною та зручною для застосування у різних середовищах.
- Наявність численних програмних інтерфейсів, завдяки яким до бази даних можуть підключатися додатки, розроблені з використанням багатьох мов програмування.
- Високі технічні характеристики, серед яких багатопоточність, орієнтація на роботу з великою кількістю користувачів, швидкодія та ефективність обробки запитів.
- Розвинена система безпеки та механізми розмежування доступу до даних на основі привілеїв користувачів.

Реляційна база даних зберігається у вигляді таблиць, кожна з яких має власну назву. На перетині рядка та стовпця розташоване окреме значення. Рядки можуть зберігатися у довільному порядку, але не допускається їх повторення. Кожен стовпець таблиці має визначену назву та тип даних, відповідно до якого задаються всі значення у стовпці.

Для зручності адміністрування та роботи з базами даних розроблені спеціальні веб-додатки з графічним інтерфейсом. Вони дозволяють керувати сервером баз даних, виконувати адміністративні команди, а також працювати з вмістом таблиць і самих баз даних. За відсутності таких інструментів адміністрування здійснюється через консольні засоби. Прикладами подібних програм є phpMyAdmin (для роботи з MySQL) та pgAdmin (для PostgreSQL). Ці програмні засоби управління базами даних інтегровані також у панель керування cPanel, яка застосовується у більшості сучасних систем віртуального хостингу.

#### **2.3.4 Система управління контентом BlizAnt**

BlizAnt – це програмне забезпечення з інтуїтивно зрозумілою оболонкою, призначене для створення вебсайтів та управління їхнім інформаційним наповненням.

Ця система є сучасною платформою з відкритим вихідним кодом, яка поширюється на безоплатній основі та дозволяє вільне використання. Програмне

ядро BlizAnt підтримує підключення зовнішніх модулів, що значно розширює його функціональні можливості. Завдяки цьому платформа може застосовуватися для створення інтернет-ресурсів найрізноманітніших типів – від особистих блогів до масштабних інформаційних порталів зі складною структурою.

Ключова перевага BlizAnt полягає у високій гнучкості. Архітектурна концепція системи передбачає легке й швидкодієне ядро, до якого може підключатися необмежена кількість додаткових модулів (плагінів) залежно від завдань розробника. Крім того, існує можливість використання готових дизайнерських рішень, що дозволяє створювати вебсайти з унікальним оформленням.

Основні особливості CMS BlizAnt:

- Простота інсталяції та початкового налаштування вебсайту.
- Підтримка трансляції RSS-каналів, які забезпечують короткий опис нової інформації на ресурсі. У сучасних умовах наявність RSS-підписки вважається стандартом для будь-якого вебсайту, особливо для блогових платформ.
- Реєстрація відвідувачів та підтримка багатокористувацького режиму, що надає кожному зареєстрованому користувачу можливість вести власний блог.
- Наявність системи тегів, які є невід’ємним елементом сучасних вебресурсів для структурування контенту.
- Можливість коментування матеріалів користувачами та адміністрування цих коментарів (включно з фільтрацією небажаних повідомлень). Вбудованого форуму система не має, проте існує можливість підключення відповідного модуля.
- Підтримка мультимедійних форматів (зображень, аудіо- та відеофайлів) з можливістю їх завантаження та коректного відображення на сторінках сайту.
- Організація системи поштової підписки, яка дозволяє відвідувачам залишати свою електронну адресу для отримання нових матеріалів.

- Розширення функціоналу шляхом встановлення додаткових модулів (плагінів).
- Використання різноманітних дизайн-тем, що забезпечують швидку зміну зовнішнього вигляду сайту та способів подання інформації. Також існує можливість замовлення індивідуального дизайну у професійних вебдизайнерів.

Таким чином, BlizAnt є багатофункціональною та масштабованою системою управління контентом, можливості якої завдяки підтримці плагінів практично не мають обмежень.

## 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ

### 3.1 Елементи керування додатку

Під час розробки додатку були застосовані такі об'єкти та елементи інтерфейсу:

- **Form**

Основне вікно програми, що виконує роль контейнера для розміщення інших елементів та об'єктів.

- **Label**

Компонент, призначений для відображення статичного тексту, який користувач не може редагувати.

- **TextBox**

Поле введення, за допомогою якого користувач має можливість вносити необхідні дані.

- **PictureBox**

Елемент, що дозволяє відображати зображення, а також може використовуватися для створення анімацій чи графічних побудов.

- **GroupBox**

Служить для логічного об'єднання кількох елементів керування в одну групу.

- **WebBrowser**

Забезпечує перегляд веб-сторінок безпосередньо у вікні програми.

- **TreeView**

Компонент для представлення ієрархічних структур із можливістю додавання піктограм та текстових позначок.

У процесі функціонування програми використовуються такі події:

- **Load**

Ініціюється під час першого відображення форми та використовується для виконання початкових налаштувань.

- **MouseClicked**

Виникає в результаті натискання кнопки миші на елементі керування.

- **MouseHover**

Активується у випадку, коли курсор миші затримується над певним елементом керування.

- **MouseLeave**

Спрацьовує під час виходу курсора миші за межі елемента керування.

### 3.2 Архітектура додатку

Структурна схема додатку зображена на рисунку 3.1.

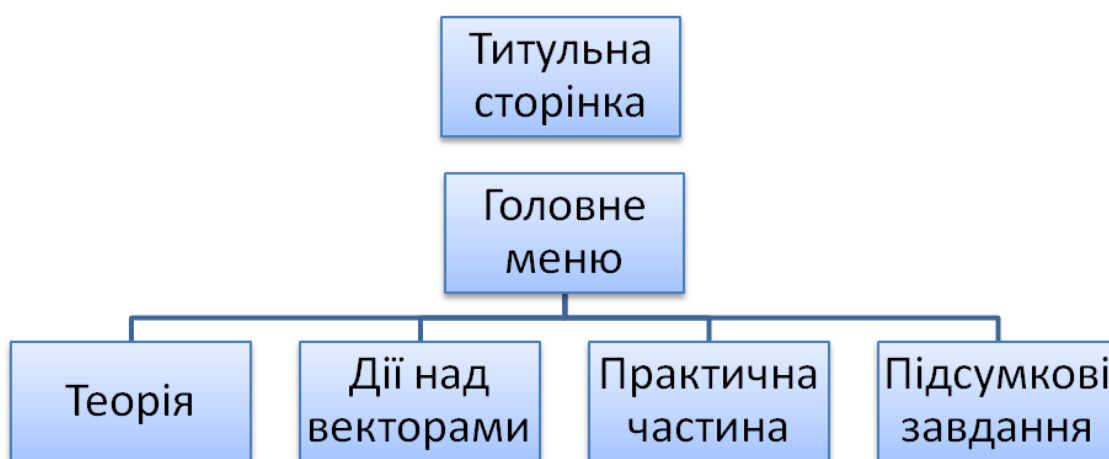


Рисунок 3.1 – Структурна схема додатку

Даний додаток має таку структуру (див. рис. 3.2):

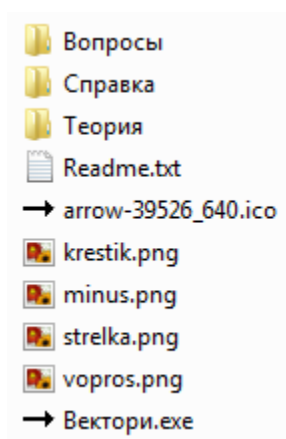


Рисунок 3.2 – Структура додатку

- Вектори.exe – виконуваний файл, що забезпечує запуск системи.
- Каталог «Питання» – містить набір завдань для розділу «Підсумкові завдання» (див. рис. 3.3)

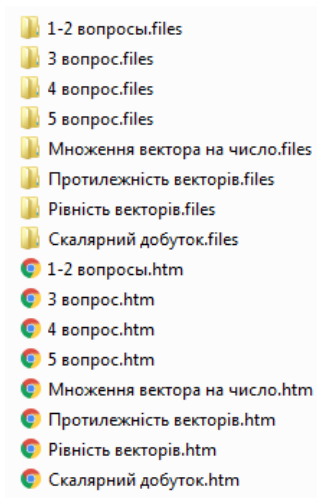


Рисунок 3.3 – Вміст каталогу «Питання»

- Каталог «Теорія» – призначений для зберігання матеріалів розділу «Теорія» (див. рис. 3.4)

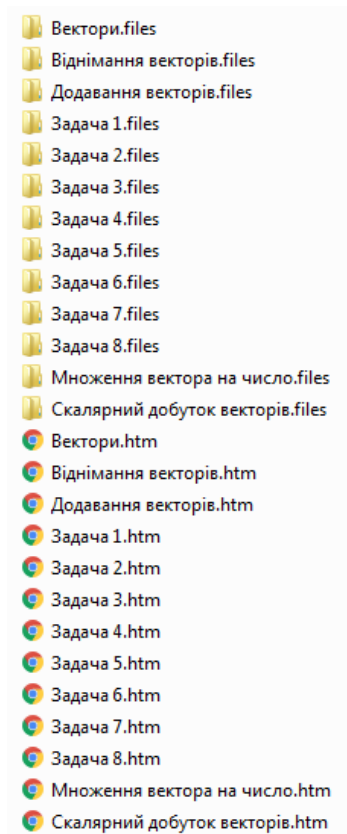


Рисунок 3.4 – Вміст каталогу «Теорія»

- Каталог «Довідка» – використовується для розміщення матеріалів розділу «Дії над векторами» (див. рис. 3.5)

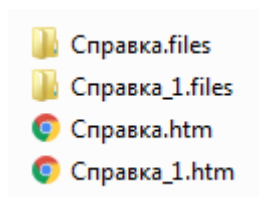


Рисунок 3.5 – Вміст каталогу «Довідка»

- Файл «readme.txt» – текстовий документ, у якому подано опис системи, а також наведено вимоги до апаратного забезпечення, операційної системи та програмного середовища (див. рис. 3.6)

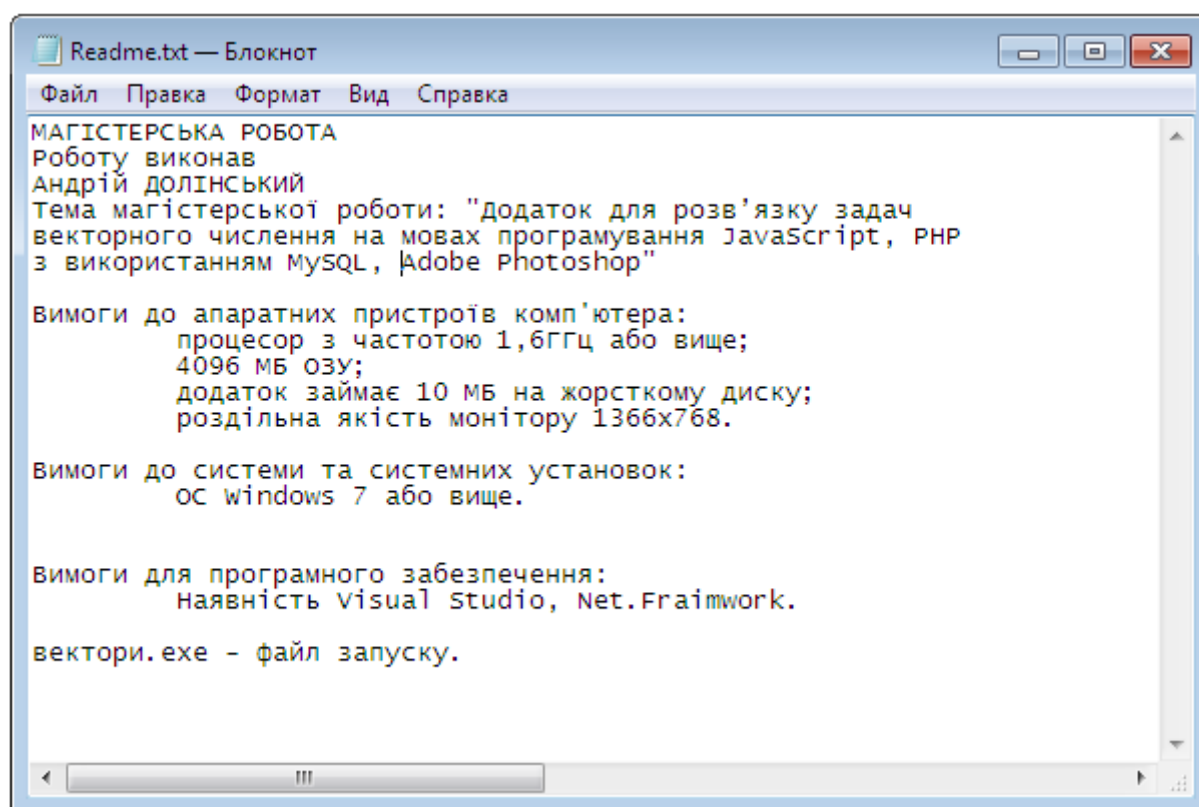


Рисунок 3.6 – Вміст файлу «readme.txt»

- Графічні файли «hrestik.png», «minus.png», «strilka.png» та «pytannia.png» – застосовуються для відображення елементів інтерфейсу (кнопок керування системою) (див. рис. 3.7)

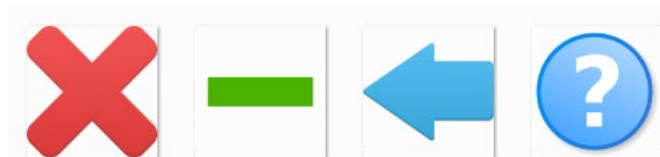


Рисунок 3.7 – Файли «hrestik.png», «minus.png», «strilka.png» та «pytannia.png»

### 3.3 Екранні форми додатку та його робота

Головна форма додатку показана на рисунку 3.8.

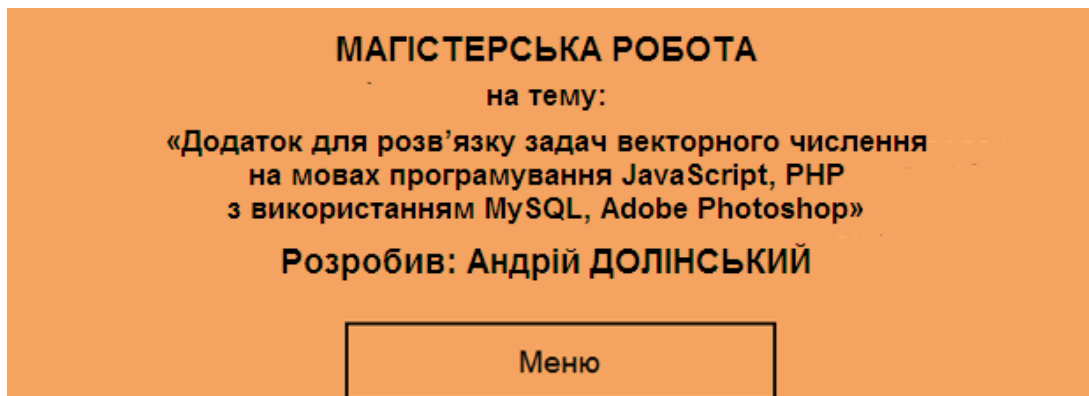


Рисунок 3.8 – Головна форма додатку

Додаток структурований на чотири основні розділи: «Теорія», «Дії над векторами», «Практична частина», «Підсумкові завдання» (див. рис. 3.9).

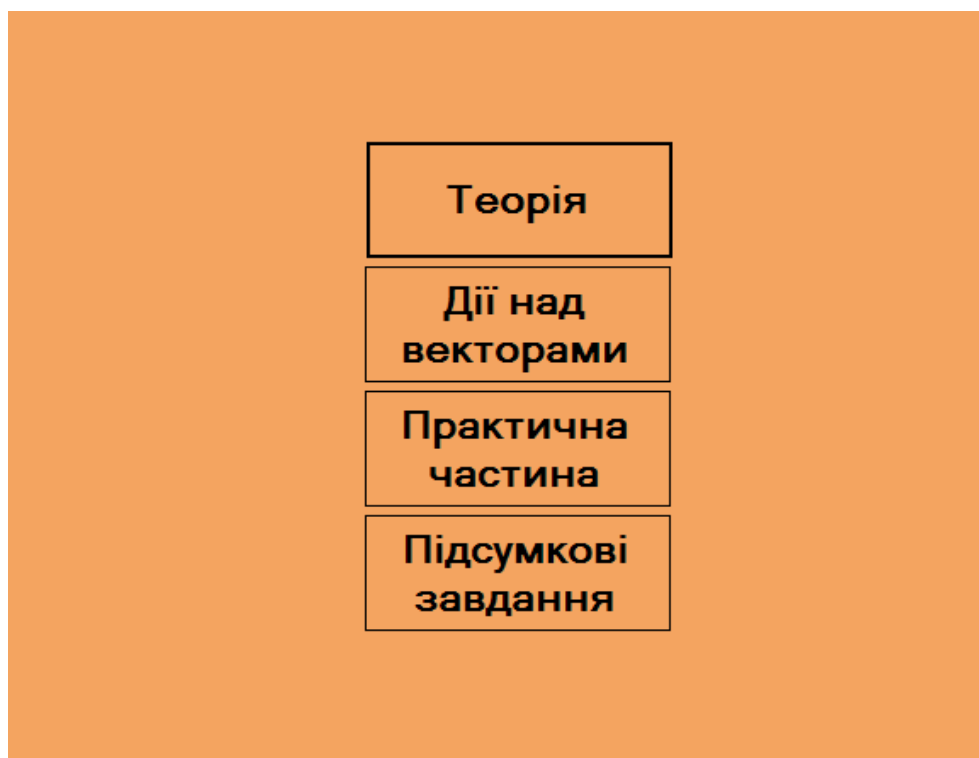


Рисунок 3.9 – Основні розділи додатку

У розділі «Теорія» подано теоретичні відомості щодо векторів на площині, які є необхідними для повноцінної роботи користувача з програмою. Також у цьому розділі наведено приклади розв’язання типових задач за відповідними темами (див. рис. 3.10). Для реалізації функціоналу використано об’єкти TreeView та WebBrowser. Об’єкт TreeView забезпечує відображення структури заголовків, а при виборі певного елемента в даному компоненті, у вікні WebBrowser автоматично відкривається відповідний документ із теоретичним матеріалом.

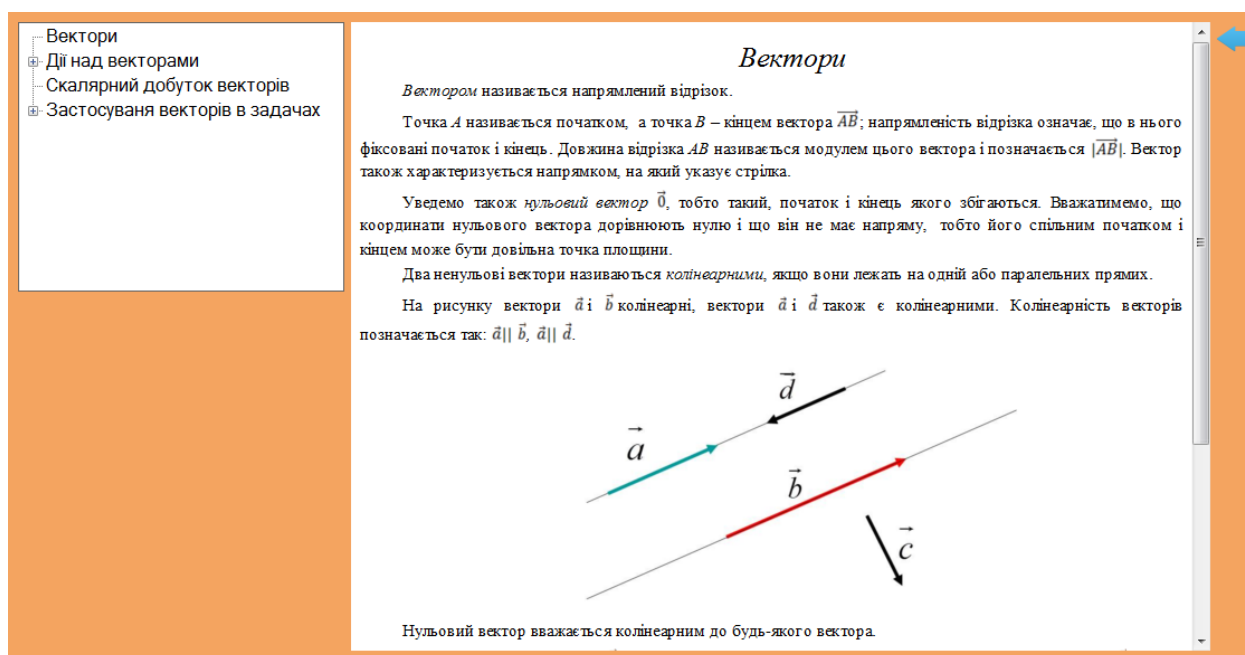


Рисунок 3.10 – Розділ «Теорія» додатку

Користувач також має можливість переглядати та опрацьовувати теоретичний матеріал, розкривши відповідну гілку дерева «Дії над векторами» (див. рис. 3.11). Змістові сторінки даної гілки відображені на рисунках 3.12–3.15.

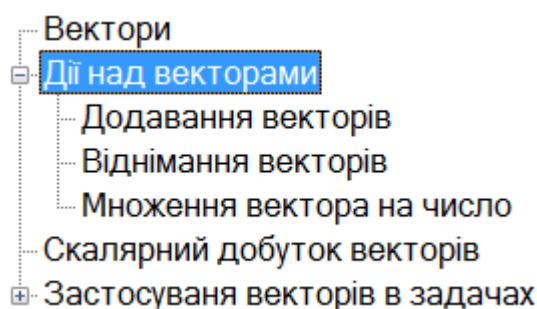


Рисунок 3.11 – Структура гілки дерева «Дії над векторами»

Вектори

- Дії над векторами
  - Додавання векторів**
  - Віднімання векторів
  - Множення вектора на число
- Скалярний добуток векторів
- Застосування векторів в задачах

### Додавання векторів

Поняття рівності векторів дає змогу «відкласти» вектор, що дорівнює даному, від будь-якої точки площини.

Розглянемо спочатку два ненульові вектори  $\vec{a}$  і  $\vec{b}$ . Відкладемо від довільної точки  $A$  вектор  $\overrightarrow{AB} = \vec{a}$ , а від точки  $B$  – вектор  $\overrightarrow{BC} = \vec{b}$ . Вектор  $\overrightarrow{AC}$  називатимемо *сумою векторів  $\vec{a}$  і  $\vec{b}$* . У цьому випадку записуватимемо

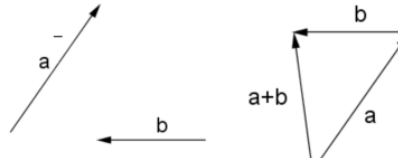
$$\overrightarrow{AC} = \overrightarrow{AB} + \overrightarrow{BC} = \vec{a} + \vec{b}.$$


Рисунок 3.12 – Сторінка «Додавання векторів»

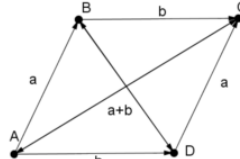
Вектори

- Дії над векторами
  - Додавання векторів
  - Віднімання векторів**
  - Множення вектора на число
- Скалярний добуток векторів
- Застосування векторів в задачах

### Віднімання векторів

Вектор  $\vec{x}$  називається *різницею векторів  $\vec{a}$  і  $\vec{b}$* , якщо  $\vec{a} = \vec{b} + \vec{x}$ , і позначається  $\vec{x} = \vec{a} - \vec{b}$ .

З правила трикутника випливає, що сума двох протилежних векторів дорівнює  $\vec{0}$ , тому для кожного вектора  $\vec{a}$  різниця  $\vec{0} - \vec{a}$  є вектор, протилежний вектору  $\vec{a}$ .



На рисунку  $\overrightarrow{DB} = \vec{a} - \vec{b}$ ,  $\overrightarrow{BD} = \vec{b} - \vec{a}$ .

Рисунок 3.13 – Сторінка «Віднімання векторів»

Вектори

- Дії над векторами
  - Додавання векторів
  - Віднімання векторів
  - Множення вектора на число**
- Скалярний добуток векторів
- Застосування векторів в задачах

### Множення вектора на число

Нехай вектор  $\vec{a} \neq \vec{0}$  і  $\lambda$  – ненульове число. Вважатимемо, при  $\lambda > 0$  вектор  $\lambda\vec{a}$  однаково напрямлений з вектором  $\vec{a}$  причому в обох випадках

$$|\lambda\vec{a}| = |\lambda| \cdot |\vec{a}|.$$

Крім того, домовимось, що  $0 \cdot \vec{a} = \vec{0}$  і  $\lambda \cdot \vec{0} = \vec{0}$ .

Очевидно за допомогою множення ненульового вектора  $\vec{a}$  на число можемо дістати будь-який вектор, колінеарний вектору  $\vec{a}$ . Крім того,  $|\lambda| = \frac{|\lambda\vec{a}|}{|\vec{a}|}$  при  $\vec{a} \neq \vec{0}$ , тобто модуль числа  $\lambda$  вказує на відношення модулів векторів, а знак числа  $\lambda$  – на взаємне напрямленість векторів  $\vec{a}$  і  $\lambda\vec{a}$ .

Вектор, протилежний вектору  $\vec{a}$ , позначається:  $-\vec{a}$ .

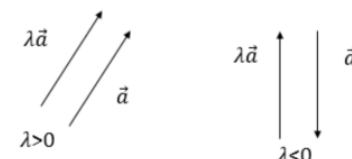


Рисунок 3.14 – Сторінка «Множення вектора на число»

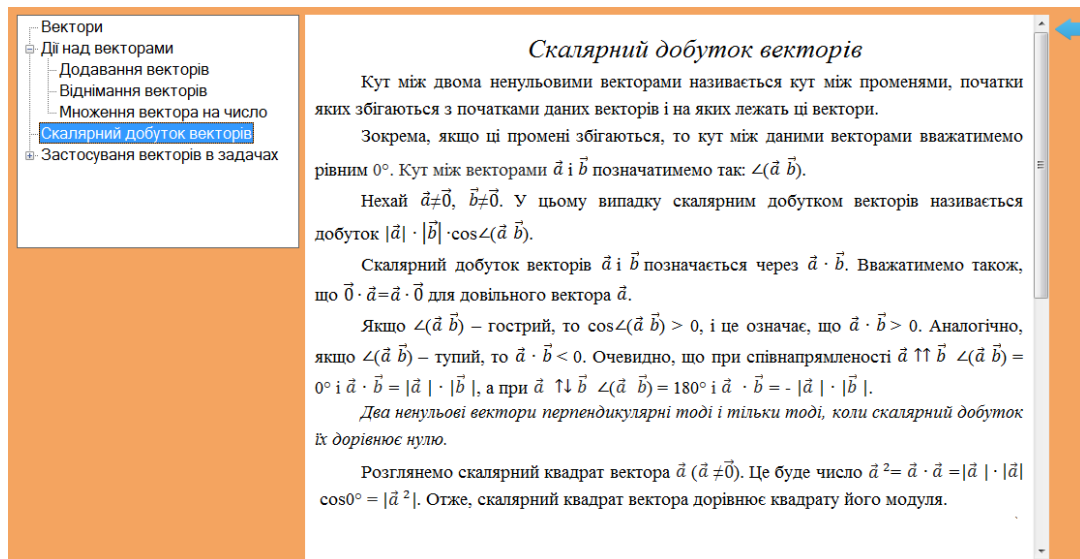


Рисунок 3.15 – Сторінка «Скалярний добуток векторів»

Крім того, користувач має змогу ознайомитися з прикладами розв'язання задач, розкривши гілку дерева «Застосування векторів у задачах» (див. рис. 3.16). Приклад відображення однієї з таких сторінок наведено на рисунку 3.17.

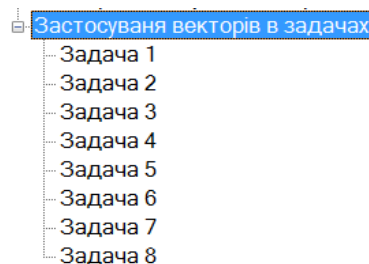


Рисунок 3.16 – Структура гілки дерева «Застосування векторів в задачах»

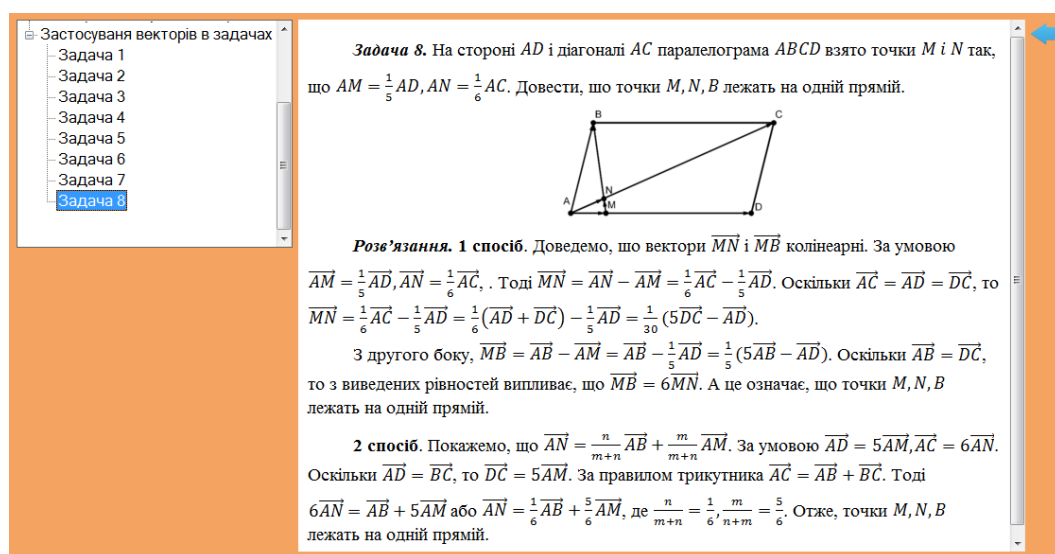


Рисунок 3.17 – Приклад сторінки із задачею

У розділ «Дії над векторами» користувач може практично ознайомитися з правилами побудови суми двох векторів методом трикутника та паралелограма, різниці методом трикутника та множення вектора на число (рис. 3.18).

The interface is divided into several sections:

- Побудувати систему координат**: A button at the top right of the grid area.
- Побудова векторів**: A section containing:
  - Координати першого вектора**:
 

| Початок | Кінець |
|---------|--------|
| x -2    | x 0    |
| y 1     | y -2   |
  - Координати другого вектора**:
 

| Початок | Кінець |
|---------|--------|
| x 2     | x 4    |
| y 3     | y 2    |
  - Побудувати вектори по координатах**: A button below the first vector's coordinates.
  - Побудувати вектори по вказаних точках**: A button below the second vector's coordinates.
- Дії над векторами**: A section on the right containing:
  - Помножити перший вектор на ціле число**: Input field with value 0.
  - Помножити другий вектор на ціле число**: Input field with value 0.
  - Побудувати суму векторів методом**:
    - Паралелограма**: A button.
    - Трикутника**: A button.
  - Побудувати різницю векторів методом трикутника**: A button.
- Очистити систему координат**: A button at the bottom left of the grid area.

Рисунок 3.18 – Сторінка задачі з розділу «Дії над векторами»

Вектори у додатку можуть будуватися кількома способами: шляхом задання координат початкової та кінцевої точки (див. рис. 3.19) або через визначення двох пар точок на координатній площині (див. рис. 3.20). У випадку, коли під час побудови суми чи різниці двох векторів початок або кінець результуючого вектора виходить за межі поточної системи координат, додаток автоматично виконує перебудову координатної сітки з урахуванням більших граничних значень (максимальних та мінімальних). Крім того, розділ містить довідкові матеріали, у яких наведено інформацію щодо особливостей відображення векторів на координатній площині (див. рис. 3.21).

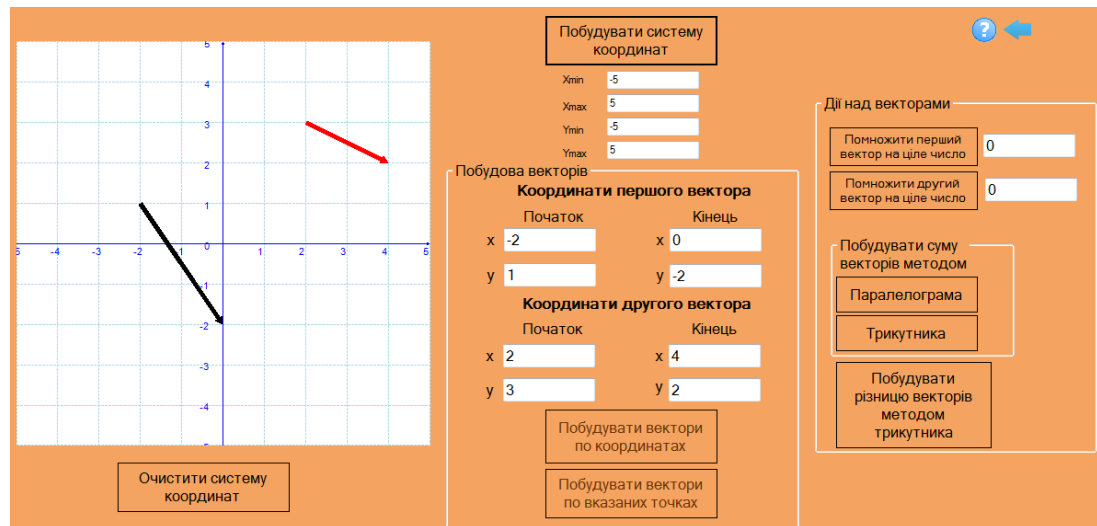
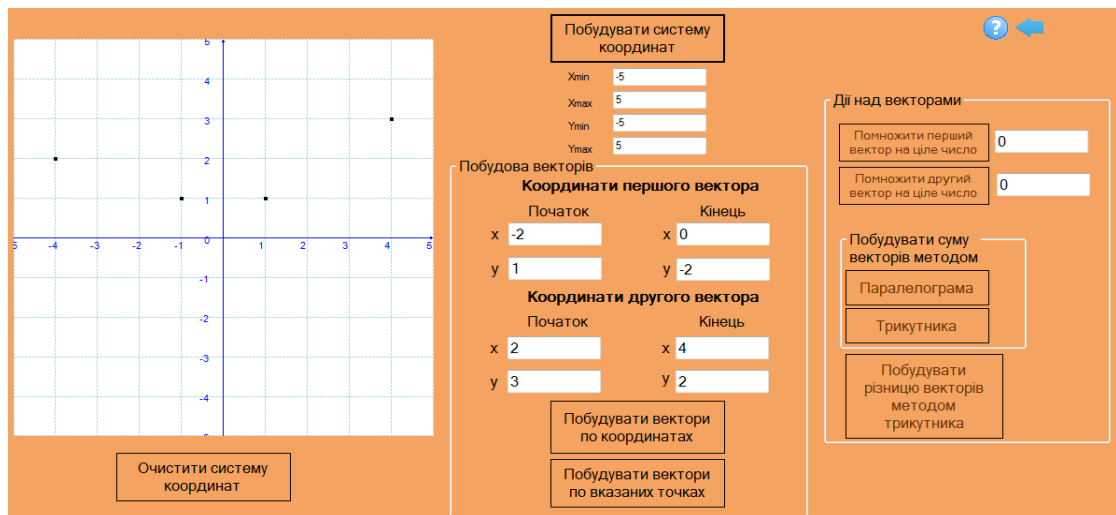
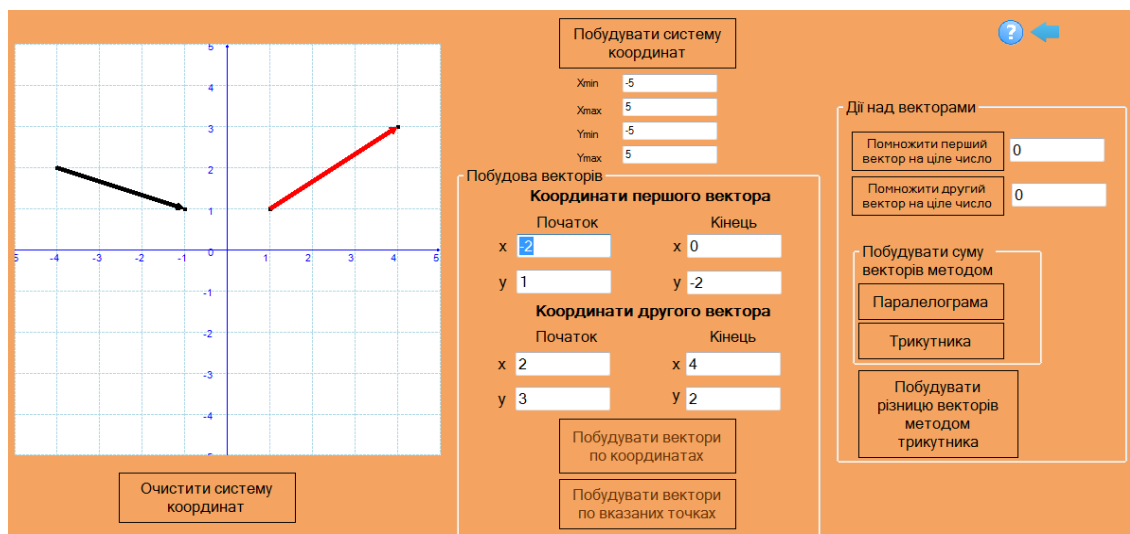


Рисунок 3.19 – Побудова вектору по координатах



а) задання точок



б) побудова векторів

Рисунок 3.20 – Побудова вектору по заданим точкам

- Перший вектор позначається чорним кольором
- Другий вектор позначається червоним кольором
- Сума векторів позначається зеленим кольором
- Різниця векторів позначається синім кольором

Рисунок 3.21 – Довідка

У розділі «Практична частина» користувач має можливість перевірити рівень засвоєння навичок побудови суми та різниці векторів (див. рис. 3.22). У випадках, коли результуючий вектор не збігається зі сторонами чи діагоналями паралелограма, навколо фігури автоматично формуються точки, у яких може розташовуватися його кінець. Для виконання завдання користувач у відповідні поля вводить назви двох векторів, обирає дію, яку необхідно здійснити, та вводить власний варіант відповіді (див. рис. 3.23). Після цього додаток здійснює перевірку введених даних і виводить повідомлення про правильність або неправильність розв'язку (див. рис. 3.24).

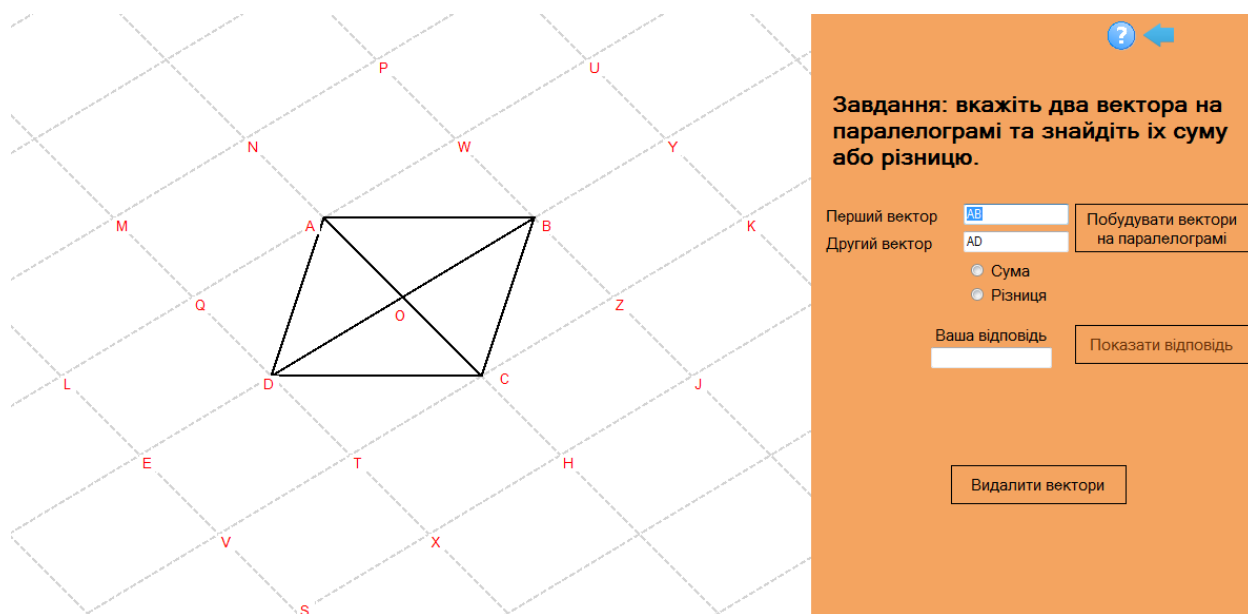


Рисунок 3.22 – Сторінка «Практична частина»

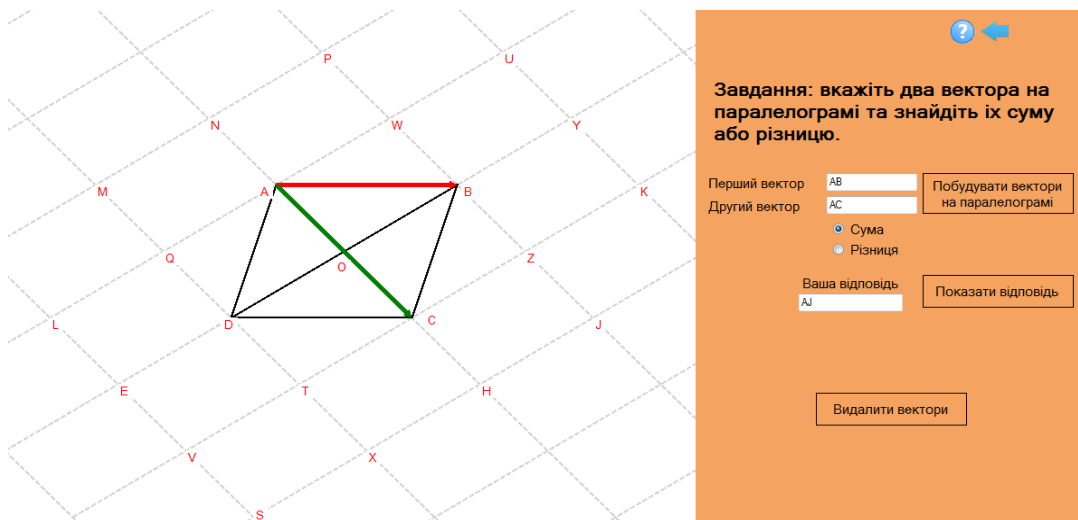
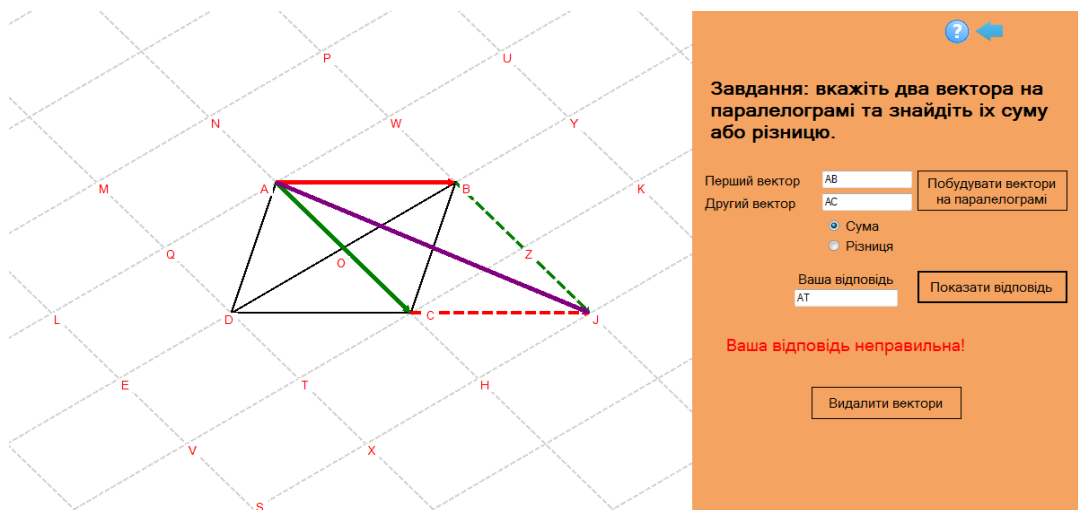
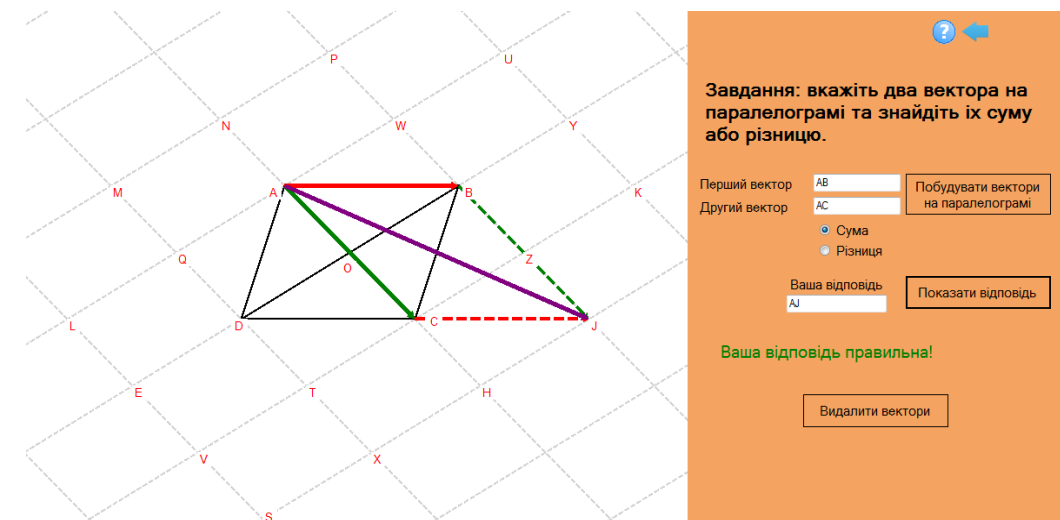


Рисунок 3.23 – Введення назви двох векторів, обрання дії та введення відповіді



а) відповідь не вірна



б) відповідь вірна

Рисунок 3.24 – Перевірка додатком відповіді користувача

Розділ «Підсумкові завдання» виконує функцію своєрідної контрольної роботи, що дозволяє перевірити рівень засвоєння користувачем знань і навичок, отриманих під час роботи з програмою (див. рис. 3.25).

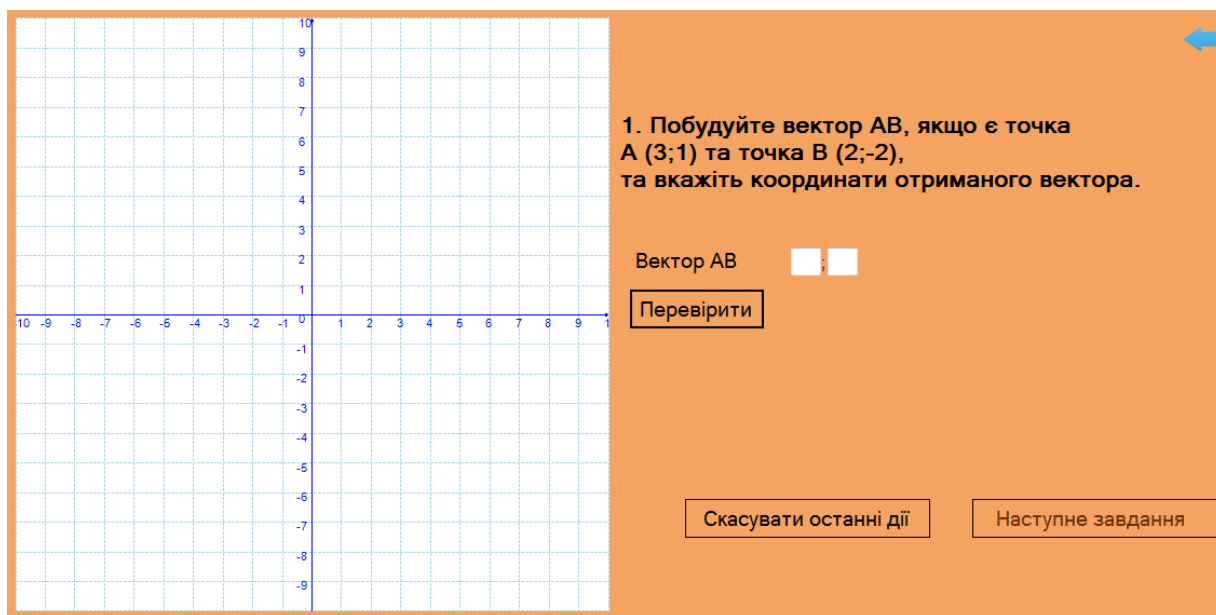


Рисунок 3.25 – Сторінка «Підсумкові завдання»

Розділ містить десять завдань, які охоплюють весь матеріал, представлений у роботі. Якщо користувач відповідає на питання правильно, виводиться відповідне повідомлення (див. рис. 3.26). Якщо користувач відповідає на питання неправильно (див. рис. 3.27), програма пропонує звернутися до відповідної теорії (див. рис. 3.28).

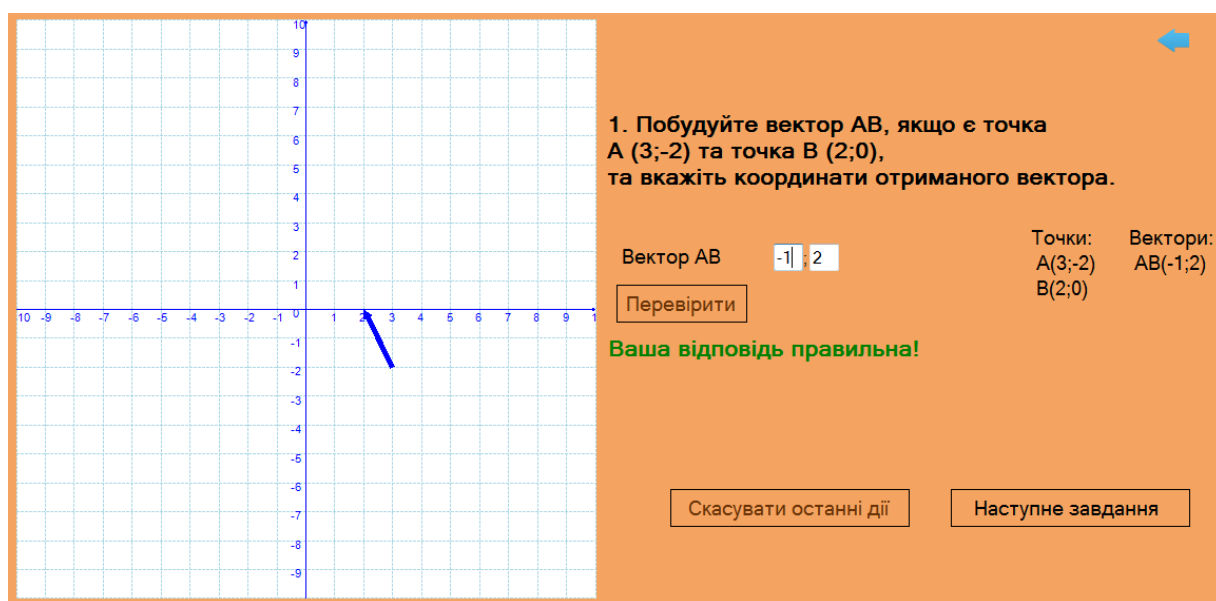


Рисунок 3.26 – Приклад сторінки із вірною відповіддю на завдання

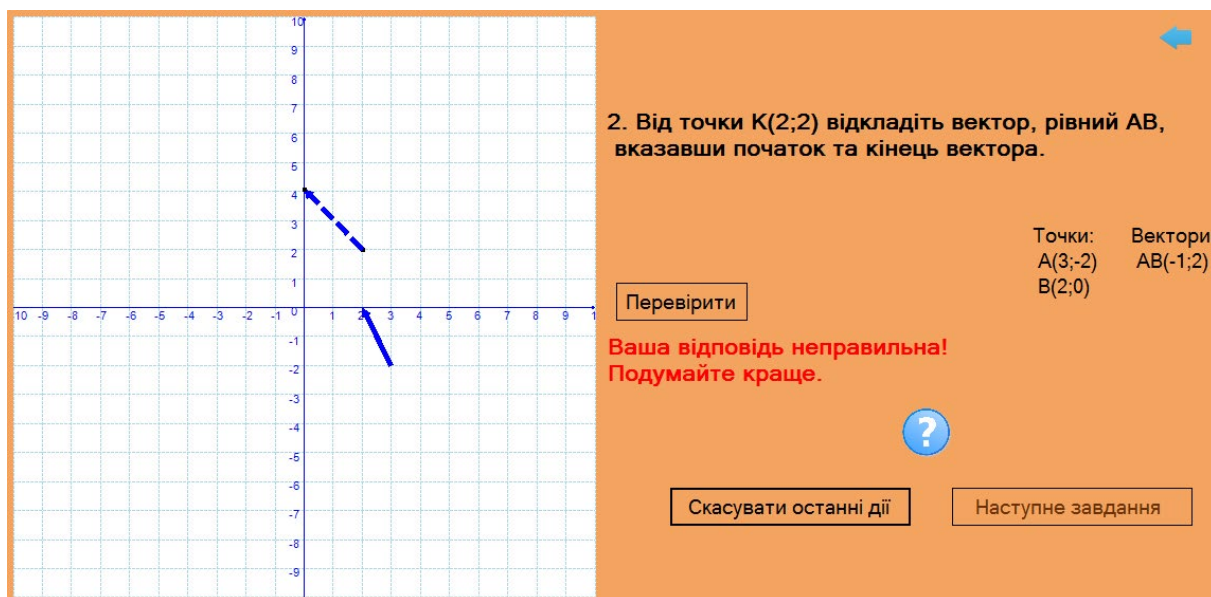


Рисунок 3.27 – Приклад сторінки із невірною відповіддю на завдання

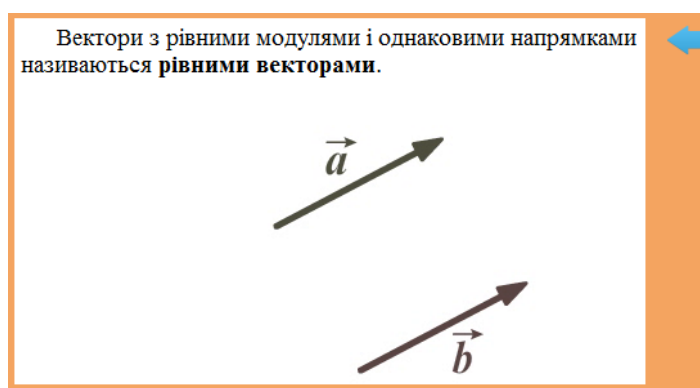


Рисунок 3.28 – Сторінка «Теорія»

Після завершення виконання всіх завдань користувач отримує інформацію про кількість допущених помилок (рис. 3.29). Додатково передбачена можливість скасування останньої виконаної дії. Водночас перехід до наступного завдання стає доступним лише після правильного виконання поточного.

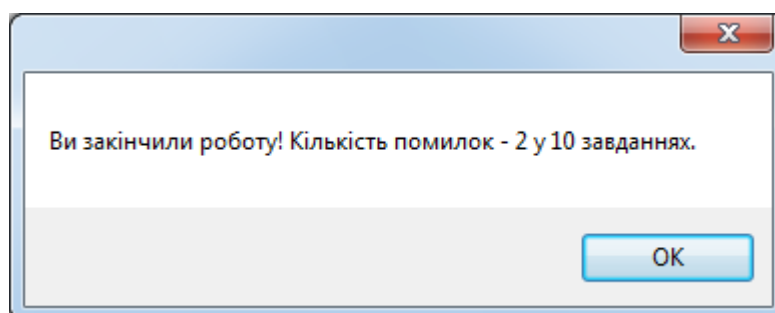


Рисунок 3.29 – Інформація про кількість помилок

Таким чином, програмний додаток структуровано у чотири основні розділи («Теорія», «Практична частина», «Дії над векторами» та «Підсумкові завдання»), кожен із яких виконує свою функцію: ознайомлення з теоретичними відомостями, відпрацювання практичних навичок, виконання операцій над векторами та перевірку рівня засвоєння матеріалу. Така організація забезпечує комплексність навчального процесу, послідовність опанування матеріалу й ефективний контроль результатів користувача.

## ВИСНОВКИ

Мета атестаційної роботи магістра полягала у створенні програмного додатку, що забезпечуватиме розв'язування задач з векторного числення і сприятиме кращому засвоєнню матеріалу учнями та студентами.

Для її реалізації було зроблено:

- Вивчено призначення та класифікація програмних засобів комп'ютерної математики, розглянуто наявні програмні засоби, визначено їх функціонал, переваги та недоліки.
- За допомогою HTML, CSS, JavaScript, JQuery, AJAX, Bootstrap, Adobe Photoshop для клієнтської частини та PHP, OpenServer, MySQL для серверної частини було розроблено додаток для розв'язування задач векторного числення із зручним та зрозумілим користувачеві інтерфейсом. В додатку користувач може ознайомитися з теоретичними відомостями про вектори, засвоїти правила виконання дій з ними, а також перевірити отримані знання на практиці при розв'язанні задач векторного числення: тут представлені завдання з перевіркою та підсумкові завдання. Завдяки зручному інтерфейсу та інтерактивним елементам такий додаток може слугувати ефективним посібником для опанування та закріплення навичок роботи з векторами.

Практичне значення додатку полягає в тому, що він може використовуватися як учнями у позашкільний час для опанування матеріалом та вдосконалення набутих навичок, так і вчителями для більш наочного пояснення нового матеріалу учням.

У перспективі вдосконалення додатку можна зазначити створення його адаптивних версій для смартфонів та планшетів.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Жалдак М. І. Педагогічний потенціал комп'ютерно-орієнтованих систем комп'ютерної математики / М. І. Жалдак // Комп'ютерно-орієнтовані системи навчання : зб. наук праць / Редкол. – К. : НПУ ім. М. П. Драгоманова. – [Випуск 7]. – 2003. – С. 3–16.
2. Саранцев Г.І. Методична система навчання предмета як об'єкт дослідження / Г. І. Саранцев // Педагогіка. – 2005. – № 2. – С. 14-19.
3. Триус Ю. В. Комп'ютерно-орієнтовані методичні системи комп'ютерної математики : [монографія] / Ю. В. Триус. – Черкаси: Брама-Україна, 2005. – 400 с.
4. Ципкін А.Г, Пінській А.І. Справочник по методам решения задач по математике. – Москва: Просвещение, 1989. – 268 с.
5. Бурда М.І, Савченко Л.М. Геометрія. – К.: Наук.думка, 1998. – 289 с.
6. Нікулін О.В, Кукуш О.Г. Геометрія. Поглиблений курс. – К.: Наук.думка, 1999. – 324 с.
7. Кушнір І. Трикутник та тетраедр в задачах. – К.: Наук.думка, 2004. – 186 с.
8. Єршов А.П., Голобородько В.В., Крижановський О.Ф., Єршов С.В. Геометрія 9 клас. – Харків: Просвіта, 2009. – 316 с.
9. Базылев В.Т., Дуничев К.И., Иваницкая В.П. Геометрия, Ч.І. – М: Просвещение, 1974. – 351 с.
10. Атанасян Л.С., Базылев В.Т. Геометрия, Ч.І – М: Просвещение, 1986. – 336 с.
11. Атанасян Л.С. Геометрия, Ч.І – М: Просвещение, 1967. – 300 с.
12. Атанасян Л.С., Атанасян В.А. Сборник задач по геометрии, Ч.І. – М: Просвещение, 1973. – 256 с.
13. Яковець, Боровик, Коваленко. Аналітична геометрія: навч. пос. – Суми: Університетська книга, 2004. – 295 с.
14. Цубербиллер О.Н. Задачи и упражнения по аналитической геометрии, М: Наука, 1970. – 335 с.

15. Клетенник Д.В. Сборник задач по аналитической геометрии, М: Наука, 1972. – 240 с.
16. Панішева О.В. Векторний метод: Інтегрований урок геометрії та фізики // Математика. – 2000. – №14. – с. 4-5.
17. Єгорова Г.О. Векторний і координатний методи розв'язування задач // Математика. – 2001. – №5. – с. 5 – 11.
18. Гончаров А. Самоучитель HTML. – СПб.: Питер, 2002. – 240с.
19. Шафран Э. Создание Web-страниц: самоучитель. – СПб.: Питер, 2001. – 320 с.
20. Фролов А.В., Фролов Г.В. Создание Web-приложений: Практическое руководство. – М.: Издательско-торговый дом "Русская редакция", 2001. – 1040 с.
21. Федорчук А. Как создаются Web-сайты. Краткий курс. – СПб.: Питер, 2000. – 224с.
22. Нельсен Якоб. Веб дизайн. – СПб.: Питер, 2013. – 504 с.
23. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных [Электронный ресурс] – URL: <http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/>
24. Пауэлл Т.А. Полное руководство по HTML. – Мн.: ООО "Попурри", 2011. – 912 с.
25. Кастро Э. Создание Web-страниц с помощью HTML. М.: НТ Пресс, 2015. – 144 с.
26. HTML[Электронный ресурс]. – URL: <https://uk.wikipedia.org/wiki/HTML>
27. Web-додаток [Электронный ресурс]. – URL: <https://uk.wikipedia.org/wiki/Веб-застосунок>
28. Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. – М.: Наука, 2015. – 464 с.
29. Молли Э. Хольцшлаг. Использование HTML и XHTML. Специальное издание. – М.: Издательский дом "Вильямс" – 2004. – 736с.

30. Мержевич Влад. HTML и CSS на примерах. – Спб.: Издательство “БХВ-Петербург” – 2005. – 448с.
31. Мальчук Е.В. HTML и CSS. Самоучитель. – М.: Издательский дом “Вильямс”. – 2018. – 416 с.
32. Дубаков М.А. Веб-мастеринг средствами CSS. – СПб.: БХВ-Петербург, 2012. – 544 с.
33. Дронов, В. JavaScript в Web-дизайне / В. Дронов. – М.: СПб: БХВ, 2014. – 880 с.
34. Дунаев В. HTML, скрипты и стили. Спб.: БХВ- Петербург, 2008. – 1024 с.
35. Кингсли-Хью, К.Э. JavaScript 1.5: учебный курс / К.Э. Кингсли-Хью. – М.: СПб: Питер, 2013. – 272 с.
36. Негрино JavaScript для начинающих / Негрино, Том. – М.: Огни, 2013. – 544 с.
37. Титтел Эд, Бурмейстер Мэри. HTML 4 для "чайников". 5-е издание. – М.: Диалектика-Вильямс, 2007. – 368с.
38. Коржинский С.Н. Настольная книга Web-мастера: эффективное применение HTML, CSS и Javascript. – М.: Издательский торговый дом "Кнорус", 2000. – 320с.
39. Федоров, А.Г. JavaScript для всех / А.Г. Федоров. – М.: Машиностроение, 2012. – 384 с.
40. Флэнаган Д. JavaScript: Посібник. – СПб: Символ Плюс, 2008. – 992 с.
41. Исси Коэн Л. Полный справочник по HTML, CSS и JavaScript / Лазаро Исси Коэн, Джозеф Исси Коэн. – М.: ЭКОМ Паблишерз, 2019. – 938 с.
42. Росс В.С. Створення сайтів: HTML, CSS, PHP, MySQL: підручник. Ч. 1. – М.: МГДД(Ю)Т, 2020. – 107 с.
43. Bootstrap Documentation – Офіційна документація Bootstrap 5. URL: <https://getbootstrap.com/docs/5.0/getting-started/introduction/> Чавез Э.Ф.К. Adobe Photoshop CC. Официальный учебный курс. – М.: Эксмо, 2022. – 448 с.

44. Bradley G. Mastering Adobe Photoshop 2024: Discover the smart way to polish your digital imagery skills by editing professional looking photos. – Packt Publishing, 2024. – 458 p.

45. Долінський А.А., Шикула О.М. Розробка додатку для розв'язку задач векторного числення за допомогою JavaScript, PHP, MySQL, Adobe Photoshop // V всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУІКТ, 2025. – С. 197-199.

## ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

```
<html xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:w="urn:schemas-microsoft-com:office:word"
xmlns:m="http://schemas.microsoft.com/office/2004/12/omml"
xmlns="http://www.w3.org/TR/REC-html40">

<head>
<meta http-equiv=Content-Type content="text/html; charset=windows-
1251">
<meta name=ProgId content=Word.Document>
<meta name=Generator content="Microsoft Word 14">
<meta name=Originator content="Microsoft Word 14">
<link rel=File-List href="Північть%20векторпів.files/filelist.xml">
<link rel=Edit-Time-Data
href="Північть%20векторпів.files/editdata.mso">
<!--[if !mso]>
<style>
v\:* {behavior:url(#default#VML);}
o\:* {behavior:url(#default#VML);}
w\:* {behavior:url(#default#VML);}
.shape {behavior:url(#default#VML);}
</style>
<![endif]--><!--[if gte mso 9]><xml>
  <o:DocumentProperties>
    <o:Author>ivanov</o:Author>
    <o:Template>Normal</o:Template>
    <o:LastAuthor>ivanov</o:LastAuthor>
    <o:Revision>2</o:Revision>
    <o:TotalTime>2</o:TotalTime>
    <o:Created>2017-01-12T07:39:00Z</o:Created>
    <o:LastSaved>2017-01-12T07:39:00Z</o:LastSaved>
    <o:Pages>1</o:Pages>
    <o:Words>12</o:Words>
    <o:Characters>73</o:Characters>
    <o:Lines>1</o:Lines>
    <o:Paragraphs>1</o:Paragraphs>
    <o:CharactersWithSpaces>84</o:CharactersWithSpaces>
    <o:Version>14.00</o:Version>
  </o:DocumentProperties>
  <o:OfficeDocumentSettings>
    <o:AllowPNG/>
  </o:OfficeDocumentSettings>
</xml><![endif]-->
<link rel=themeData href="Північть%20векторпів.files/themedata.thmx">
<link rel=colorSchemeMapping
href="Північть%20векторпів.files/colorscchemmapping.xml">
<!--[if gte mso 9]><xml>
  <w:WordDocument>
    <w:TrackMoves>>false</w:TrackMoves>
```

```

<w:TrackFormatting/>
<w:PunctuationKerning/>
<w:ValidateAgainstSchemas/>
<w:SaveIfXMLInvalid>>false</w:SaveIfXMLInvalid>
<w:IgnoreMixedContent>>false</w:IgnoreMixedContent>
<w:AlwaysShowPlaceholderText>>false</w:AlwaysShowPlaceholderText>
<w:DoNotPromoteQF/>
<w:LidThemeOther>RU</w:LidThemeOther>
<w:LidThemeAsian>X-NONE</w:LidThemeAsian>
<w:LidThemeComplexScript>X-NONE</w:LidThemeComplexScript>
<w:Compatibility>
  <w:BreakWrappedTables/>
  <w:SnapToGridInCell/>
  <w:WrapTextWithPunct/>
  <w:UseAsianBreakRules/>
  <w:DontGrowAutofit/>
  <w:SplitPgBreakAndParaMark/>
  <w:EnableOpenTypeKerning/>
  <w:DontFlipMirrorIndents/>
  <w:OverrideTableStyleHps/>
</w:Compatibility>
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="&#45;-"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
  <m:lMargin m:val="0"/>
  <m:rMargin m:val="0"/>
  <m:defJc m:val="centerGroup"/>
  <m:wrapIndent m:val="1440"/>
  <m:intLim m:val="subSup"/>
  <m:naryLim m:val="undOvr"/>
</m:mathPr></w:WordDocument>
</xml><![endif]--><!--[if gte mso 9]><xml>
  <w:LatentStyles DefLockedState="false" DefUnhideWhenUsed="true"
    DefSemiHidden="true" DefQFormat="false" DefPriority="99"
    LatentStyleCount="267">
    <w:LsdException Locked="false" Priority="0" SemiHidden="false"
      UnhideWhenUsed="false" QFormat="true" Name="Normal"/>
    <w:LsdException Locked="false" Priority="9" SemiHidden="false"
      UnhideWhenUsed="false" QFormat="true" Name="heading 1"/>
    <w:LsdException Locked="false" Priority="9" QFormat="true"
      Name="heading 2"/>
    <w:LsdException Locked="false" Priority="9" QFormat="true"
      Name="heading 3"/>
    <w:LsdException Locked="false" Priority="9" QFormat="true"
      Name="heading 4"/>
    <w:LsdException Locked="false" Priority="9" QFormat="true"
      Name="heading 5"/>
    <w:LsdException Locked="false" Priority="9" QFormat="true"
      Name="heading 6"/>

```

```

    <w:LsdException Locked="false" Priority="9" QFormat="true"
Name="heading 7"/>
    <w:LsdException Locked="false" Priority="9" QFormat="true"
Name="heading 8"/>
    <w:LsdException Locked="false" Priority="9" QFormat="true"
Name="heading 9"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 1"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 2"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 3"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 4"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 5"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 6"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 7"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 8"/>
    <w:LsdException Locked="false" Priority="39" Name="toc 9"/>
    <w:LsdException Locked="false" Priority="35" QFormat="true"
Name="caption"/>
    <w:LsdException Locked="false" Priority="10" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Title"/>
    <w:LsdException Locked="false" Priority="1" Name="Default
Paragraph Font"/>
    <w:LsdException Locked="false" Priority="11" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Subtitle"/>
    <w:LsdException Locked="false" Priority="22" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Strong"/>
    <w:LsdException Locked="false" Priority="20" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Emphasis"/>
    <w:LsdException Locked="false" Priority="59" SemiHidden="false"
    UnhideWhenUsed="false" Name="Table Grid"/>
    <w:LsdException Locked="false" UnhideWhenUsed="false"
Name="Placeholder Text"/>
    <w:LsdException Locked="false" Priority="1" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="No Spacing"/>
    <w:LsdException Locked="false" Priority="60" SemiHidden="false"
    UnhideWhenUsed="false" Name="Light Shading"/>
    <w:LsdException Locked="false" Priority="61" SemiHidden="false"
    UnhideWhenUsed="false" Name="Light List"/>
    <w:LsdException Locked="false" Priority="62" SemiHidden="false"
    UnhideWhenUsed="false" Name="Light Grid"/>
    <w:LsdException Locked="false" Priority="63" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Shading 1"/>
    <w:LsdException Locked="false" Priority="64" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Shading 2"/>
    <w:LsdException Locked="false" Priority="65" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium List 1"/>
    <w:LsdException Locked="false" Priority="66" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium List 2"/>
    <w:LsdException Locked="false" Priority="67" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Grid 1"/>
    <w:LsdException Locked="false" Priority="68" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Grid 2"/>
    <w:LsdException Locked="false" Priority="69" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Grid 3"/>

```

```

<w:LsdException Locked="false" Priority="70" SemiHidden="false"
  UnhideWhenUsed="false" Name="Dark List"/>
<w:LsdException Locked="false" Priority="71" SemiHidden="false"
  UnhideWhenUsed="false" Name="Colorful Shading"/>
<w:LsdException Locked="false" Priority="72" SemiHidden="false"
  UnhideWhenUsed="false" Name="Colorful List"/>
<w:LsdException Locked="false" Priority="73" SemiHidden="false"
  UnhideWhenUsed="false" Name="Colorful Grid"/>
<w:LsdException Locked="false" Priority="60" SemiHidden="false"
  UnhideWhenUsed="false" Name="Light Shading Accent 1"/>
<w:LsdException Locked="false" Priority="61" SemiHidden="false"
  UnhideWhenUsed="false" Name="Light List Accent 1"/>
<w:LsdException Locked="false" Priority="62" SemiHidden="false"
  UnhideWhenUsed="false" Name="Light Grid Accent 1"/>
<w:LsdException Locked="false" Priority="63" SemiHidden="false"
  UnhideWhenUsed="false" Name="Medium Shading 1 Accent 1"/>
<w:LsdException Locked="false" Priority="64" SemiHidden="false"
  UnhideWhenUsed="false" Name="Medium Shading 2 Accent 1"/>
<w:LsdException Locked="false" Priority="65" SemiHidden="false"
  UnhideWhenUsed="false" Name="Medium List 1 Accent 1"/>
<w:LsdException Locked="false" UnhideWhenUsed="false"
Name="Revision"/>
<w:LsdException Locked="false" Priority="34" SemiHidden="false"
  UnhideWhenUsed="false" QFormat="true" Name="List Paragraph"/>
<w:LsdException Locked="false" Priority="29" SemiHidden="false"
  UnhideWhenUsed="false" QFormat="true" Name="Quote"/>
<w:LsdException Locked="false" Priority="30" SemiHidden="false"
  UnhideWhenUsed="false" QFormat="true" Name="Intense Quote"/>
<w:LsdException Locked="false" Priority="66" SemiHidden="false"
  UnhideWhenUsed="false" Name="Medium List 2 Accent 1"/>
<w:LsdException Locked="false" Priority="67" SemiHidden="false"
  UnhideWhenUsed="false" Name="Medium Grid 1 Accent 1"/>
<w:LsdException Locked="false" Priority="68" SemiHidden="false"
  UnhideWhenUsed="false" Name="Medium Grid 2 Accent 1"/>
<w:LsdException Locked="false" Priority="69" SemiHidden="false"
  UnhideWhenUsed="false" Name="Medium Grid 3 Accent 1"/>
<w:LsdException Locked="false" Priority="70" SemiHidden="false"
  UnhideWhenUsed="false" Name="Dark List Accent 1"/>
<w:LsdException Locked="false" Priority="71" SemiHidden="false"
  UnhideWhenUsed="false" Name="Colorful Shading Accent 1"/>
<w:LsdException Locked="false" Priority="72" SemiHidden="false"
  UnhideWhenUsed="false" Name="Colorful List Accent 1"/>
<w:LsdException Locked="false" Priority="73" SemiHidden="false"
  UnhideWhenUsed="false" Name="Colorful Grid Accent 1"/>
<w:LsdException Locked="false" Priority="60" SemiHidden="false"
  UnhideWhenUsed="false" Name="Light Shading Accent 2"/>
<w:LsdException Locked="false" Priority="61" SemiHidden="false"
  UnhideWhenUsed="false" Name="Light List Accent 2"/>
<w:LsdException Locked="false" Priority="62" SemiHidden="false"
  UnhideWhenUsed="false" Name="Light Grid Accent 2"/>
<w:LsdException Locked="false" Priority="63" SemiHidden="false"
  UnhideWhenUsed="false" Name="Medium Shading 1 Accent 2"/>
<w:LsdException Locked="false" Priority="64" SemiHidden="false"

```

[illegible]

[illegible]

```

    UnhideWhenUsed="false" Name="Light List Accent 6"/>
<w:LsdException Locked="false" Priority="62" SemiHidden="false"
    UnhideWhenUsed="false" Name="Light Grid Accent 6"/>
<w:LsdException Locked="false" Priority="63" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Shading 1 Accent 6"/>
<w:LsdException Locked="false" Priority="64" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Shading 2 Accent 6"/>
<w:LsdException Locked="false" Priority="65" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium List 1 Accent 6"/>
<w:LsdException Locked="false" Priority="66" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium List 2 Accent 6"/>
<w:LsdException Locked="false" Priority="67" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Grid 1 Accent 6"/>
<w:LsdException Locked="false" Priority="68" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Grid 2 Accent 6"/>
<w:LsdException Locked="false" Priority="69" SemiHidden="false"
    UnhideWhenUsed="false" Name="Medium Grid 3 Accent 6"/>
<w:LsdException Locked="false" Priority="70" SemiHidden="false"
    UnhideWhenUsed="false" Name="Dark List Accent 6"/>
<w:LsdException Locked="false" Priority="71" SemiHidden="false"
    UnhideWhenUsed="false" Name="Colorful Shading Accent 6"/>
<w:LsdException Locked="false" Priority="72" SemiHidden="false"
    UnhideWhenUsed="false" Name="Colorful List Accent 6"/>
<w:LsdException Locked="false" Priority="73" SemiHidden="false"
    UnhideWhenUsed="false" Name="Colorful Grid Accent 6"/>
<w:LsdException Locked="false" Priority="19" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Subtle Emphasis"/>
<w:LsdException Locked="false" Priority="21" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Intense Emphasis"/>
<w:LsdException Locked="false" Priority="31" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Subtle Reference"/>
<w:LsdException Locked="false" Priority="32" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Intense Reference"/>
<w:LsdException Locked="false" Priority="33" SemiHidden="false"
    UnhideWhenUsed="false" QFormat="true" Name="Book Title"/>
<w:LsdException Locked="false" Priority="37" Name="Bibliography"/>
<w:LsdException Locked="false" Priority="39" QFormat="true"
Name="TOC Heading"/>
</w:LatentStyles>
</xml><![endif]-->
<style>
<!--
/* Font Definitions */
@font-face
    {font-family:Calibri;
    panose-1:2 15 5 2 2 2 4 3 2 4;
    mso-font-charset:204;
    mso-generic-font-family:swiss;
    mso-font-pitch:variable;
    mso-font-signature:-536859905 -1073732485 9 0 511 0;}
@font-face
    {font-family:Tahoma;
    panose-1:2 11 6 4 3 5 4 4 2 4;

```

```

mso-font-charset:0;
mso-generic-font-family:swiss;
mso-font-format:other;
mso-font-pitch:variable;
mso-font-signature:3 0 0 0 1 0;}
/* Style Definitions */
p.MsoNormal, li.MsoNormal, div.MsoNormal
    {mso-style-unhide:no;
mso-style-qformat:yes;
mso-style-parent:"";
margin-top:0cm;
margin-right:0cm;
margin-bottom:10.0pt;
margin-left:0cm;
line-height:115%;
mso-pagination:widow-orphan;
font-size:11.0pt;
font-family:"Calibri","sans-serif";
mso-ascii-font-family:Calibri;
mso-ascii-theme-font:minor-latin;
mso-fareast-font-family:Calibri;
mso-fareast-theme-font:minor-latin;
mso-hansi-font-family:Calibri;
mso-hansi-theme-font:minor-latin;
mso-bidi-font-family:"Times New Roman";
mso-bidi-theme-font:minor-bidi;
mso-fareast-language:EN-US;}
p.MsoAcetate, li.MsoAcetate, div.MsoAcetate
    {mso-style-noshow:yes;
mso-style-priority:99;
mso-style-link:"Текст выноски Знак";
margin:0cm;
margin-bottom:.0001pt;
mso-pagination:widow-orphan;
font-size:8.0pt;
font-family:"Tahoma","sans-serif";
mso-fareast-font-family:Calibri;
mso-fareast-theme-font:minor-latin;
mso-bidi-font-family:Tahoma;
mso-fareast-language:EN-US;}
span.a
    {mso-style-name:"Текст выноски Знак";
mso-style-noshow:yes;
mso-style-priority:99;
mso-style-unhide:no;
mso-style-locked:yes;
mso-style-link:"Текст выноски";
mso-ansi-font-size:8.0pt;
mso-bidi-font-size:8.0pt;
font-family:"Tahoma","sans-serif";
mso-ascii-font-family:Tahoma;
mso-hansi-font-family:Tahoma;
mso-bidi-font-family:Tahoma;}

```

```

.MsoChpDefault
    {mso-style-type:export-only;
    mso-default-props:yes;
    font-family:"Calibri","sans-serif";
    mso-ascii-font-family:Calibri;
    mso-ascii-theme-font:minor-latin;
    mso-fareast-font-family:Calibri;
    mso-fareast-theme-font:minor-latin;
    mso-hansi-font-family:Calibri;
    mso-hansi-theme-font:minor-latin;
    mso-bidi-font-family:"Times New Roman";
    mso-bidi-theme-font:minor-bidi;
    mso-fareast-language:EN-US;}

.MsoPapDefault
    {mso-style-type:export-only;
    margin-bottom:10.0pt;
    line-height:115%;}

@page WordSection1
    {size:595.3pt 841.9pt;
    margin:2.0cm 42.5pt 2.0cm 3.0cm;
    mso-header-margin:35.4pt;
    mso-footer-margin:35.4pt;
    mso-paper-source:0;}

div.WordSection1
    {page:WordSection1;}

-->
</style>
<!--[if gte mso 10]>
<style>
    /* Style Definitions */
    table.MsoNormalTable
        {mso-style-name:"Обычная таблица";
        mso-tstyle-rowband-size:0;
        mso-tstyle-colband-size:0;
        mso-style-noshow:yes;
        mso-style-priority:99;
        mso-style-parent:"";
        mso-padding-alt:0cm 5.4pt 0cm 5.4pt;
        mso-para-margin-top:0cm;
        mso-para-margin-right:0cm;
        mso-para-margin-bottom:10.0pt;
        mso-para-margin-left:0cm;
        line-height:115%;
        mso-pagination:widow-orphan;
        font-size:11.0pt;
        font-family:"Calibri","sans-serif";
        mso-ascii-font-family:Calibri;
        mso-ascii-theme-font:minor-latin;
        mso-hansi-font-family:Calibri;
        mso-hansi-theme-font:minor-latin;
        mso-bidi-font-family:"Times New Roman";
        mso-bidi-theme-font:minor-bidi;
        mso-fareast-language:EN-US;}

```

```

</style>
<![endif]--><!--[if gte mso 9]><xml>
  <o:shapedefaults v:ext="edit" spidmax="1026"/>
</xml><![endif]--><!--[if gte mso 9]><xml>
  <o:shapelayout v:ext="edit">
    <o:idmap v:ext="edit" data="1"/>
  </o:shapelayout></xml><![endif]-->
</head>

<body lang=RU style='tab-interval:35.4pt'>

<div class=WordSection1>

<p class=MsoNormal style='text-align:justify;text-indent:21.25pt'><span
lang=UK style='font-size:14.0pt;line-height:115%;font-family:"Times
New Roman","serif";
background:white;mso-ansi-language:UK;mso-bidi-font-weight:bold'>Вектори з
рівними модулями і однаковими напрямками називаються&nbsp;<b>рівними
векторами</b></span><span
style='font-size:14.0pt;line-height:115%;font-family:"Times New
Roman","serif";
background:white;mso-bidi-font-weight:bold'>.&nbsp;</span><span
lang=UK
style='font-size:14.0pt;line-height:115%;font-family:"Times New
Roman","serif";
background:white;mso-ansi-language:UK;mso-bidi-font-weight:bold'><o:p></o:p></span></p>

<p class=MsoNormal align=center style='text-align:center;text-indent:21.25pt'><span
style='font-size:14.0pt;line-height:115%;font-family:"Times New
Roman","serif";
mso-fareast-language:RU;mso-no-proof:yes'><!--[if gte vml
1]><v:shapetype id="_x0000_t75"
  coordsize="21600,21600" o:spt="75" o:preferrelative="t"
  path="m@4@5l@4@11@9@11@9@5xe"
  filled="f" stroked="f">
  <v:stroke joinstyle="miter"/>
  <v:formulas>
    <v:f eqn="if lineDrawn pixelLineWidth 0"/>
    <v:f eqn="sum @0 1 0"/>
    <v:f eqn="sum 0 0 @1"/>
    <v:f eqn="prod @2 1 2"/>
    <v:f eqn="prod @3 21600 pixelWidth"/>
    <v:f eqn="prod @3 21600 pixelHeight"/>
    <v:f eqn="sum @0 0 1"/>
    <v:f eqn="prod @6 1 2"/>
    <v:f eqn="prod @7 21600 pixelWidth"/>
    <v:f eqn="sum @8 21600 0"/>
    <v:f eqn="prod @7 21600 pixelHeight"/>
    <v:f eqn="sum @10 21600 0"/>

```

```
</v:formulas>
<v:path o:extrusionok="f" gradientshapeok="t"
o:connecttype="rect"/>
<o:lock v:ext="edit" aspectratio="t"/>
</v:shapetype><v:shape id="Рисунок_x0020_1" o:spid="_x0000_i1025"
type="#_x0000_t75"
style='width:234pt;height:195.75pt;visibility:visible;mso-wrap-
style:square'>
<v:imagedata src="Рівність%20векторів.files/image001.png"
o:title=""/>
</v:shape><![endif]--><![if !vml]><![endif]></span><span
lang=UK style='font-size:14.0pt;line-height:115%;font-family:"Times
New Roman","serif";
mso-ansi-language:UK'><o:p></o:p></span></p>

</div>

</body>

</html>
```

## ДОДАТОК Б. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ  
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ  
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ  
ТЕХНОЛОГІЙ  
Кафедра Комп'ютерних наук

**ДИПЛОМНА РОБОТА**  
на на ступінь вищої освіти магістр  
із спеціальності F3 Комп'ютерні науки  
**Додаток для розв'язку задач векторного числення**  
**на мовах програмування JavaScript, PHP**  
**з використанням MySQL, Adobe Photoshop**

Виконав: студент 6 курсу, групи КНДМ-62

Долінський Андрій Андрійович

Керівник: професор кафедри Комп'ютерних наук  
д.ф.-м.н., професор Шикуча О.М.

Київ - 2025

1

Слайд 1

### Актуальність дослідження

Більшість завдань з векторного числення, які часто створюють значні труднощі для школярів і студентів, підпорядковуються певному алгоритму, що відкриває можливість розробити спеціальний програмний додаток, призначений для виконання вправ з векторного числення. Використання такого додатку дозволить у позааудиторний та позааудиторний час самостійно вдосконалювати вміння та навички роботи з векторами. Крім того, цей інструмент стане у пригоді викладачам, оскільки може слугувати наочним матеріалом для пояснення теми. З огляду на це, проведене дослідження має беззаперечну *актуальність*.

2

Слайд 2

## МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

**Об'єкт дослідження** – процес створення додатку для розв'язку задач з векторного числення.

**Предмет дослідження** – додаток для розв'язку задач з векторного числення..

**Мета дослідження** – є створення програмного додатку, що забезпечуватиме розв'язування задач з векторного числення і сприятиме кращому засвоєнню матеріалу учнями та студентами.

**Наукове завдання :**

- вивчити призначення та класифікацію програмних засобів комп'ютерної математики, розглянути наявні програмні засоби, визначити їх функціонал, переваги та недоліки.
- підібрати інструменти для розробки клієнтської та серверної частин додатку;
- розробити додаток для розв'язування задач з векторного числення зі зручним інтуїтивно-зрозумілим інтерфейсом.

**Методи дослідження** – HTML, CSS, JavaScript, JQuery, AJAX, Bootstrap, Adobe Photoshop для клієнтської частини та PHP, OpenServer, MySQL для серверної частини.

3

Слайд 3

## 1. ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ

Комп'ютерна математика - комплекс теоретичних основ, методичних підходів, алгоритмічних прийомів, а також апаратних і програмних засобів, призначених для ефективного розв'язування за допомогою комп'ютера широкого спектра математичних задач. Важливою особливістю цього напрямку є високий рівень візуалізації кожного етапу обчислювального процесу.

Програмні засоби комп'ютерної математики – це спеціалізовані програмні продукти, які дозволяють автоматизувати виконання найрізноманітніших математичних операцій: від чисельних розрахунків до аналітичних перетворень і побудови графічних моделей.

4

Слайд 4

# 1. ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ

**Класифікація програмних засобів комп'ютерної математики:**

- Системи для чисельних розрахунків.
- Табличні процесори .
- Матричні системи.
- Системи для статистичних розрахунків.
- Системи для спеціалізованих розрахунків.
- Системи для аналітичних розрахунків (системи комп'ютерної алгебри).
- Універсальні системи.

5

Слайд 5

# 1. ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ

**Розглянуто характеристики деяких найбільш відомих систем, їх функціонал, переваги та недоліки:**

Macsyma, Maple, Mathematica, MathCad, Axiom.

Після проведеного детального огляд та аналізу тематичної області, огляду наявних програмних засобів, визначення їх функціоналу, переваг та недоліків було поставлено завдання розробити додаток для розв'язку задач векторного числення зі зручним інтуїтивно-зрозумілим інтерфейсом.

6

Слайда 6

## 2. АНАЛІЗ ТА ШЛЯХИ РОЗВ'ЯЗКУ ПРОБЛЕМИ

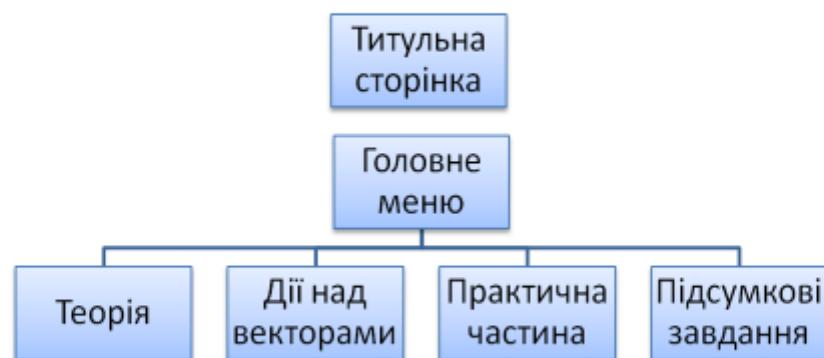
Для реалізації проекту було обрано наступні технології:

- HTML, CSS, JavaScript, Bootstrap, Adobe Photoshop для створення клієнтської частини
- PHP, OpenServer, MySQL для створення серверної частини

7

Слайд 7

## 3. ОПИС РОБОТИ ДОДАТКУ



Структурна схема додатку

8

Слайд 8

### 3. ОПИС РОБОТИ ДОДАТКУ

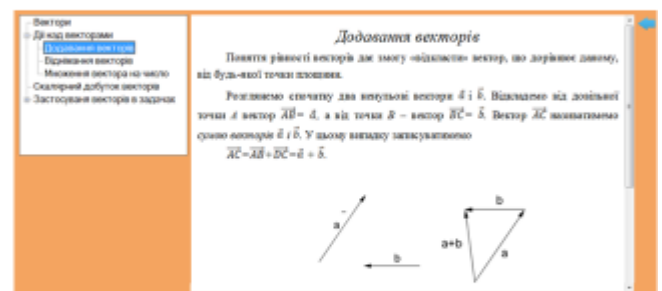


Основні розділи системи

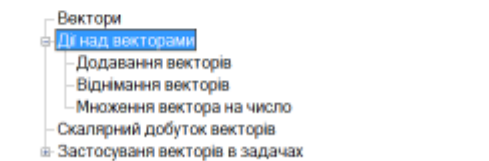
9

Слайд 9

### 3. ОПИС РОБОТИ ДОДАТКУ



Вікна розділу «Теорія»



Структура гілки дерева «Дії над векторами»

10

Слайд 10

### 3. ОПИС РОБОТИ ДОДАТКУ

Вектори

Ді над векторами

Додавання векторів

Віднімання векторів

Висновок вектора на число

Скалярний добуток векторів

Застосування векторів в задачі

Віднімання векторів

Вектор  $\vec{d}$  називається різницею векторів  $\vec{a}$  і  $\vec{b}$ , якщо  $\vec{a} = \vec{b} + \vec{d}$ , і позначається  $\vec{d} = \vec{a} - \vec{b}$ .

З правил трикутника випливає, що сума двох протилежних векторів дорівнює  $\vec{0}$ , тому для кожного вектора  $\vec{a}$  рівняє  $\vec{b} - \vec{a}$  є вектор, протилежний вектору  $\vec{a}$ .

На рисунку  $\vec{CB} = \vec{a} - \vec{b}$ ,  $\vec{CB} = \vec{b} - \vec{a}$ .

Вектори

Ді над векторами

Додавання векторів

Віднімання векторів

Висновок вектора на число

Скалярний добуток векторів

Застосування векторів в задачі

Висновок вектора на число

Новий вектор  $\lambda \vec{a}$  і  $\lambda$  – чисельне число. Визначимо, при  $\lambda > 0$  вектор  $\lambda \vec{a}$  однаково напрямлений з вектором  $\vec{a}$  причому в обидві сторони  $|\lambda \vec{a}| = |\lambda| \cdot |\vec{a}|$ . Крім того, доведемо, що  $\vec{0} \cdot \vec{a} = \vec{a} \cdot \vec{0} = \vec{0}$ .

Очевидно за допомогою множення нульового вектора  $\vec{a}$  на число можна дістати будь-який вектор, колінеарний вектору  $\vec{a}$ . Крім того,  $|\lambda| = \frac{|\lambda \vec{a}|}{|\vec{a}|}$  при  $\vec{a} \neq \vec{0}$ , тобто модуль числа  $\lambda$  вказує на відношення модулів векторів, а знак числа  $\lambda$  – на взаємну напрямленість векторів  $\vec{a}$  і  $\lambda \vec{a}$ .

Вектор, протилежний вектору  $\vec{a}$ , позначається  $-\vec{a}$ .

$\lambda > 0$   $\lambda < 0$

Вікна розділу «Теорія»

11

Слайд 11

### 3. ОПИС РОБОТИ ДОДАТКУ

Вектори

Ді над векторами

Додавання векторів

Віднімання векторів

Висновок вектора на число

Скалярний добуток векторів

Застосування векторів в задачі

Скалярний добуток векторів

Кут між двома ненульовими векторами називається кут між променями, початки яких збігаються з початками двох векторів і на них лежать ці вектори.

Зокрема, якщо ці промені збігаються, то кут між двома векторами зникатиме рівняє  $0^\circ$ . Кут між векторами  $\vec{a}$  і  $\vec{b}$  позначається так:  $\angle(\vec{a}, \vec{b})$ .

Новий  $\vec{a} \neq \vec{0}$ ,  $\vec{b} \neq \vec{0}$ . У цьому випадку скалярний добуток векторів визначається добутком  $|\vec{a}| \cdot |\vec{b}| \cdot \cos \angle(\vec{a}, \vec{b})$ .

Скалярний добуток векторів  $\vec{a}$  і  $\vec{b}$  позначається через  $\vec{a} \cdot \vec{b}$ . Визначимо також, що  $\vec{b} \cdot \vec{a} = \vec{a} \cdot \vec{b}$  для довільного вектора  $\vec{a}$ .

Якщо  $\angle(\vec{a}, \vec{b})$  – тупий, то  $\cos \angle(\vec{a}, \vec{b}) < 0$ , і це означає, що  $\vec{a} \cdot \vec{b} < 0$ . Аналогічно, якщо  $\angle(\vec{a}, \vec{b})$  – гострий, то  $\vec{a} \cdot \vec{b} > 0$ . Очевидно, що при співнаправленості  $\vec{a}$  і  $\vec{b}$   $\angle(\vec{a}, \vec{b}) = 0^\circ$  і  $\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}|$ , а при  $\vec{a} \perp \vec{b}$   $\angle(\vec{a}, \vec{b}) = 90^\circ$  і  $\vec{a} \cdot \vec{b} = 0$ .

Для ненульових векторів перпендикулярність модулів і нульовий добуток є еквівалентними.

Розглянемо скалярний квадрат вектора  $\vec{a}$  ( $\vec{a} \neq \vec{0}$ ). Це буде число  $\vec{a} \cdot \vec{a} = \vec{a} \cdot \vec{a} = |\vec{a}| \cdot |\vec{a}| \cdot \cos 0^\circ = |\vec{a}|^2$ . Отже, скалярний квадрат вектора дорівнює квадрату його модуля.

Вікна розділу «Теорія»

12

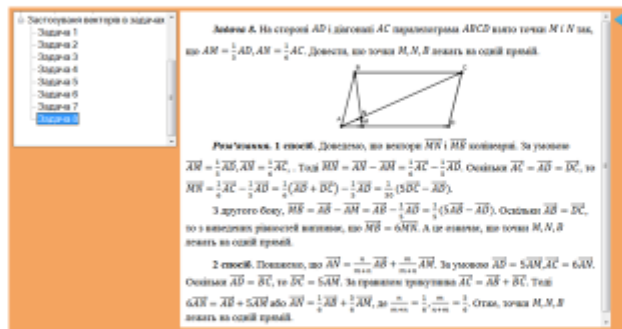
Слайд 12

### 3. ОПИС РОБОТИ ДОДАТКУ

#### Застосування векторів в задачах

- Задача 1
- Задача 2
- Задача 3
- Задача 4
- Задача 5
- Задача 6
- Задача 7
- Задача 8

Гілка дерева «Застосування векторів в задачах»

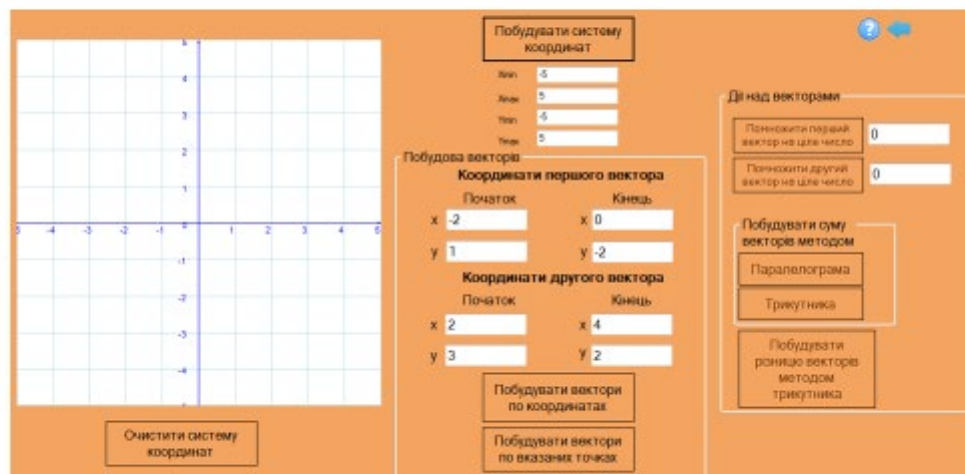


Приклад сторінки із задачею

13

Слайд 13

### 3. ОПИС РОБОТИ ДОДАТКУ

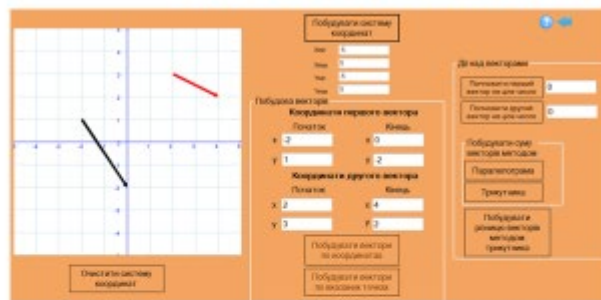


Вікно задачі з розділу «Дії над векторами»

14

Слайд 14

### 3. ОПИС РОБОТИ ДОДАТКУ

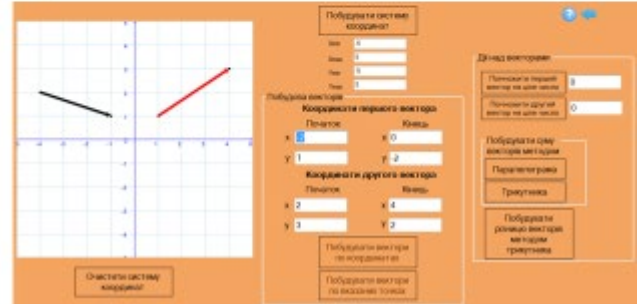
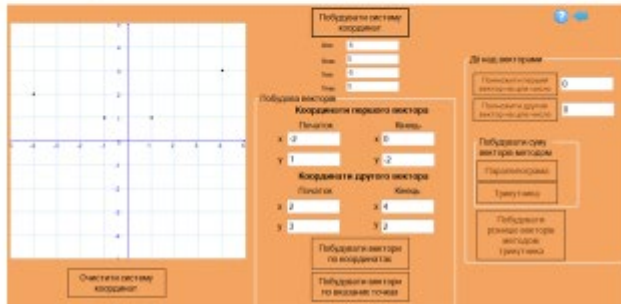


Побудова вектору по координатах

- Перший вектор позначається чорним кольором
- Другий вектор позначається червоним кольором
- Сума векторів позначається зеленим кольором
- Різниця векторів позначається синім кольором

Довідка

Побудова вектору по заданим точкам



Слайд 15

### 3. ОПИС РОБОТИ ДОДАТКУ



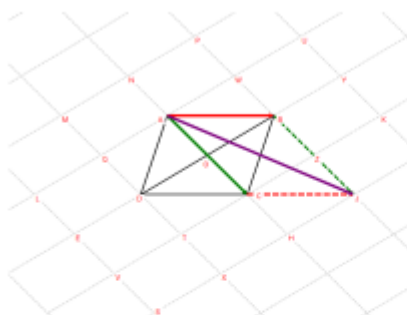
Вікно розділу «Практична частина»

Введення назви двох векторів, обрання дії та введення відповіді



Слайд 16

### 3. ОПИС РОБОТИ ДОДАТКУ



відповідь не вірна

Завдання: вкажіть два вектори на паралелограмі та знайдіть їх суму або різницю.

Перший вектор:  Побудуйте вектори на паралелограмі

Другий вектор:  Сума Різниця

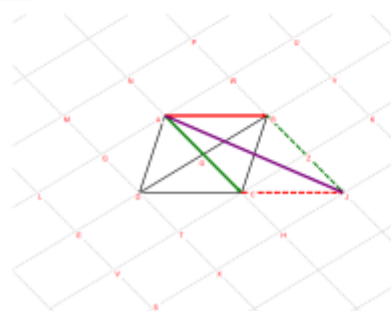
Ваша відповідь:  Покажіть відповідь

Ваша відповідь неправильна!

Відкажіть вектори

Перевірка додатком відповіді користувача

відповідь вірна



Завдання: вкажіть два вектори на паралелограмі та знайдіть їх суму або різницю.

Перший вектор:  Побудуйте вектори на паралелограмі

Другий вектор:  Сума Різниця

Ваша відповідь:  Покажіть відповідь

Ваша відповідь правильна!

Відкажіть вектори

Слайд 17

### 3. ОПИС РОБОТИ ДОДАТКУ

1. Побудуйте вектор  $\overrightarrow{AB}$ , якщо є точка  $A(3;1)$  та точка  $B(2;-2)$ , та вкажіть координати отриманого вектора.

Вектор  $\overrightarrow{AB}$

Перевірити

Скасувати останні дії

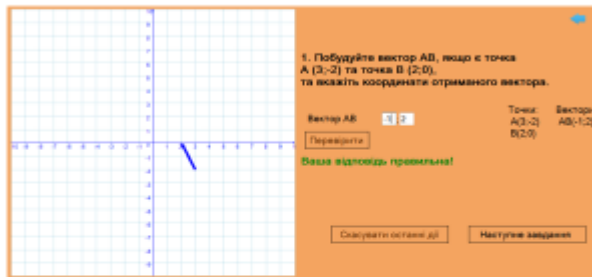
Наступне завдання

Вікно розділу «Підсумкові завдання» (контрольна робота)

18

Слайд 18

### 3. ОПИС РОБОТИ ДОДАТКУ

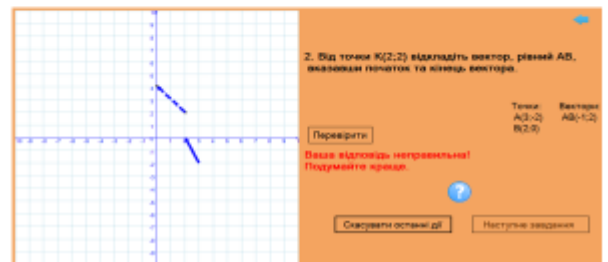


Приклад сторінки із вірною відповіддю на завдання

Приклад сторінки із невірною відповіддю на завдання

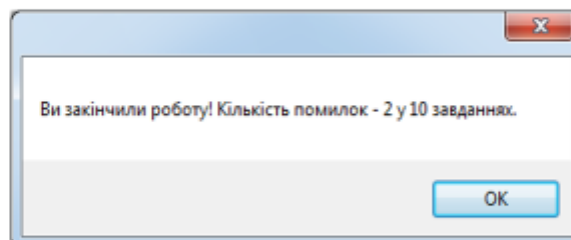


Відсилання до теорії



Слайд 19

### 3. ОПИС РОБОТИ ДОДАТКУ



Інформація про кількість помилок

Слайд 20

# ВИСНОВКИ

Мета атестаційної роботи магістра полягала у створенні програмного додатку, що забезпечуватиме розв'язування задач з векторного числення і сприятиме кращому засвоєнню матеріалу учнями та студентами.

Для її реалізації було зроблено:

- Вивчено призначення та класифікація програмних засобів комп'ютерної математики, розглянуто наявні програмні засоби, визначено їх функціонал, переваги та недоліки.

21

Слайд 21

# ВИСНОВКИ

- За допомогою HTML, CSS, JavaScript, JQuery, AJAX, Bootstrap, Adobe Photoshop для клієнтської частини та PHP, OpenServer, MySQL для серверної частини було розроблено додаток для розв'язування задач векторного числення із зручним та зрозумілим користувачеві інтерфейсом. В додатку користувач може ознайомитися з теоретичними відомостями про вектори, засвоїти правила виконання дій з ними, а також перевірити отримані знання на практиці при розв'язанні задач векторного числення: тут представлені завдання з перевіркою та підсумкові завдання. Завдяки зручному інтерфейсу та інтерактивним елементам такий додаток може слугувати ефективним посібником для опанування та закріплення навичок роботи з векторами.

22

Слайд 22

# ВИСНОВКИ

Практичне значення додатку полягає в тому, що він може використовуватися як учнями у позашкільний час для опанування матеріалом та вдосконалення набутих навичок, так і вчителями для більш наочного пояснення нового матеріалу учням.

У перспективі вдосконалення додатку можна зазначити створення його адаптивних версій для смартфонів та планшетів.