**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»**

**ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Кафедра інформаційних технологій

**КУРСОВА РОБОТА  
з дисципліни «Об’єктно-орієнтоване програмування»**

**на тему: «Калькулятор матриць»**

студента 2 курсу групи ІПЗ-211

спеціальності 121 «Інженерія програмного забезпечення»

Мозгового Максима Вікторовича

Керівник доцент кафедри IT

к.т.н., доцент Щербина Ю. В.

Національна шкала

Кількість балів \_\_\_\_\_ Оцінка ECTS

Члени комісії:

підпис прізвище та ініціали

підпис прізвище та ініціали

м. Одеса – 2023 р.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»**

**ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інформаційних технологій**

# **ЗАВДАННЯ**

з дисципліни «Об’єктно-орієнтоване програмування»

студента Мозгового Максима Вікторовича

спеціальності 121 Інженерія програмного забезпечення

другого курсу групи ІПЗ-211

Тема «Калькулятор матриць»

Курсова робота виконується в такому обсязі:

− аналіз операцій з матрицями;

* аналіз завдань, розв'язуваних з урахуванням їх представлення у матричному вигляді;

− особливості обробки даних при роботі з матрицями;

− аналіз існуючих систем;

− обґрунтування вибору засобів розробки;

− ключові фічі;

− розробка алгоритмів;

− розробка інтерфейсу та зв’язування його з алгоритмами;

− опис процесу розгортання програми;

− тестування роботи програми.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| № | Завдання | Термін виконання |
| 1 | Ознайомлення з завданням до курсової роботи. | 27.02.2023-03.03.2023 |
| 2 | Частина 1. Аналіз предметної області | 06.03.2023-13.03.2023 |
| 3 | Частина 2. Аналоги та засоби розробки | 14.03.2023-31.03.2023 |
| 4 | Частина 3. Реалізація системи | 1.04.2023-30.04.2023 |
| 5 | Оформлення пояснювальної записки | 30.04.2023-08.05.2023 |

# **ЗМІСТ**

[ЗАВДАННЯ 1](#_Toc134107949)

[ВСТУП 4](#_Toc134107951)

[1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 5](#_Toc134107952)

[1.1 Аналіз операцій з матрицями 5](#_Toc134107953)

[1.2 Аналіз завдань, розв'язуваних з урахуванням їх представлення у матричному вигляді 8](#_Toc134107954)

[1.3 Особливості обробки даних при роботі з матрицями 10](#_Toc134107955)

[2 АНАЛОГИ ТА ЗАСОБИ РОЗРОБКИ 12](#_Toc134107956)

[2.1 Аналіз існуючих систем 12](#_Toc134107957)

[2.2 Обґрунтування вибору засобів та методів розробки 16](#_Toc134107958)

[2.3 Ключові фічі 18](#_Toc134107959)

[3 РЕАЛІЗАЦІЯ СИСТЕМИ 22](#_Toc134107960)

[3.1 Розробка алгоритмів 22](#_Toc134107961)

[3.1.1 Опис функції MinusMatrix () 22](#_Toc134107962)

[3.1.2 Опис функції UmnMatrixChislo() 22](#_Toc134107963)

[3.1.3 Опис функції pow() 23](#_Toc134107964)

[3.1.4 Опис функції det() 23](#_Toc134107965)

[3.1.5 Опис функції Rounding1() 26](#_Toc134107966)

[3.2 Розробка інтерфейсу та зв'язування його з алгоритмами 28](#_Toc134107967)

[3.3 Опис процесу розгортання програми 30](#_Toc134107968)

[3.4 Тестування роботи програми 33](#_Toc134107969)

[ВИСНОВОК 35](#_Toc134107970)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 36](#_Toc134107971)

[ДОДАТОК А. ЛІСТИНГ КОДУ 38](#_Toc134107972)

# **ВСТУП**

В більшості офісах, лабораторіях та інших робочих місцях, пов'язаних із різноманітними розрахунками, для більшості операцій потрібні якісь обчислення з нестандартними типами даних. Наприклад, це можуть бути комплексні числа, числа в будь-якій системі обчислення, вектора, матриці і так далі. Щоб швидко виконувати ці операції дуже зручно використовувати комп'ютер, якщо в ньому є придатна для конкретного завдання програма.

Метою даного курсового проєкту є створення саме такої програми, яка виконувала б основні операції над матрицями.

Майже у всіх на комп'ютері зараз встановлена програма Excel із популярного пакету Microsoft Office. У цій програмі можна виконувати всі основні операції над матрицями. Але для виконання всіх цих операцій потрібно мати досить велику математичну базу і вміти працювати з Excel.

Тобто передбачуваний проєкт повинен мати низку якостей, які виділили б його серед інших програм з тією ж функцією.

По-перше, функціональність, тобто програма повинна включати основний та додатковий набір операцій із матрицями.

По-друге, простота інтерфейсу, розрахованого практично на будь-якого користувача, що має хоча б початкове поняття про матриці.

По-третє, зручність у використанні програми.

Основне призначення цього курсового проекту – збільшення показника продуктивності працівників, чия робота пов'язані з обчисленнями різноманітних величин розмірністю матриць, використання нового програмного продукту з виробництва.

Курсова робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. Загальний обсяг курсової роботи: 61 с., 21 рис., 3 табл., 1 додаток, 17 джерел.

# **АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

## **Аналіз операцій з матрицями**

Матриця розмірністю m×n називається таблиця упорядкованих чисел розташованих у m рядках, n стовпців і позначається великими літерами *A, B,…..,Y, Z*. Вигляд матриці 4×4 наведено на рис. 1.1:

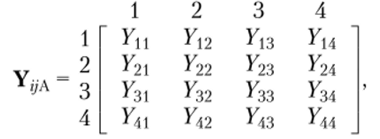


Рисунок 1.1 – Вигляд матриці 4×4

Основними операціями над матрицями є додавання, віднімання, множення матриць та множення матриць на число.

Додавання матриць (сума матриць) *A + B* є операція знаходження матриці *C*, всі елементи якої дорівнюють попарній сумі всіх відповідних елементів матриці *A* і *B*, тобто кожен елемент матриці *C* дорівнює:

= +

Додавання матриць А і В зображено на рис. 1.2:

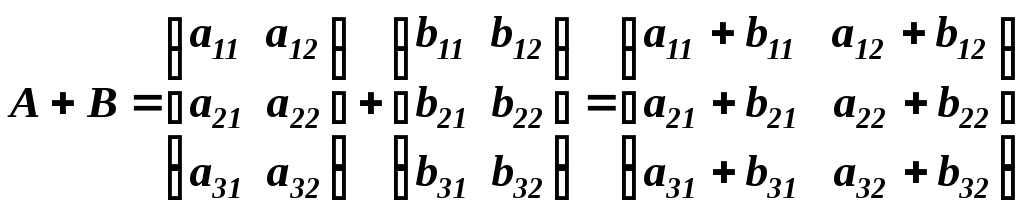


Рисунок 1.2 – Додавання матриць А і В

Віднімання матриць (різниця матриць) *A - B* є операція знаходження матриці C, всі елементи якої дорівнюють попарній різниці всіх відповідних елементів матриці A і B, тобто кожен елемент матриці C дорівнює:

= –

Результатом множення матриць та буде матриця така, що елемент матриці *C*, що знаходяться в *i*-тому рядку та *j*-тому стовпчику (), дорівнює сумі добутків елементів *i*-того рядку матриці *A* на відповідні елементи *j*-того стовпця матриці *B*:

Множення матриць зображено на рис. 1.3:

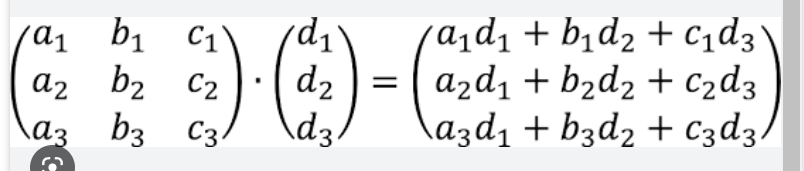


Рисунок 1.3 ­­­– Множення матриць

Добутком матриці *A* на число *k* називається матриця B = k · A того ж розміру, отримана з початкової множенням на задане число всіх її елементів:

Добуток матриці на число зображено на рис. 1.4:

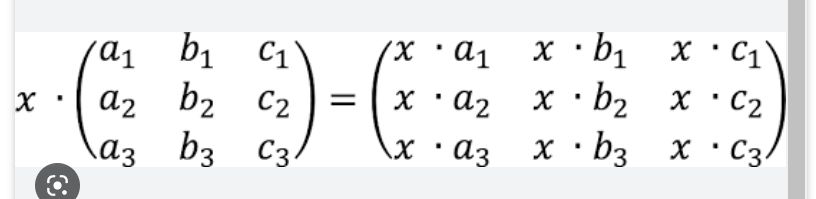


Рисунок 1.4 – Добуток матриці на число

Додавання та віднімання матриць може бути виконане лише над матрицями з однаковими розмірами, де кожен елемент однієї матриці додається або віднімається від відповідного елементу іншої матриці. Множення матриць може бути виконане над матрицями з різними розмірами, але з умовою, що кількість стовпців першої матриці дорівнює кількості рядків другої матриці. При множенні матриць створюється нова матриця, в якій кожен елемент є сумою добутків відповідних елементів рядка першої матриці та стовпця другої матриці [1].

Інші важливі операції над матрицями включають знаходження визначника матриці, транспонування, зведення до ступеня та знаходження оберненої матриці.

Визначником (детермінантом) будь-якої квадратної матриці *A* називається алгебраїчна сума всіх можливих добутків елементів матриці *A*, взятих по одному з кожного рядка і стовпця з певним знаком (Теорема Лапласа).

Властивості визначників:

* при транспонуванні визначник не змінюється;
* якщо у визначнику поміняти місцями будь-які два паралельні ряди, то знак визначника зміниться на протилежний;
* якщо визначник має двоє однакових парних рядів, то він дорівнює 0;
* визначник не зміниться, якщо до всіх елементів будь-якого ряду додати відповідний елемент паралельного ряду помноженого на одне й те саме число;
* для того, щоб визначник помножити на число, потрібно помножити на це число деякий його ряд [2].

Знаходження визначника матриці зображено на рис. 1.5:

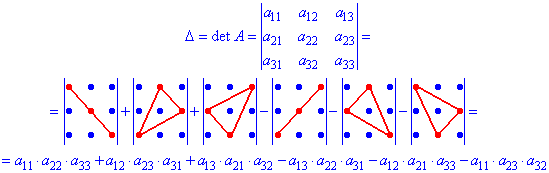


Рисунок 1.5 – Знаходження визначника матриці

Транспонована матриця є результатом переупорядкування вихідної матриці шляхом зміни рядків за стовпцями та стовпців за рядками в новій матриці [3]. Транспонування матриці зображено на рис. 1.6:

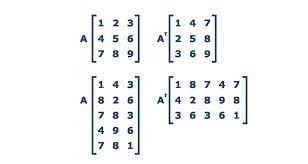


Рисунок 1.6 – Транспонування матриці

Обернена матриця – матриця, добуток якої на початкову матрицю *A* рівний одиничній матриці *E*:

Формула оберненої матриці зображена на рис. 1.7:

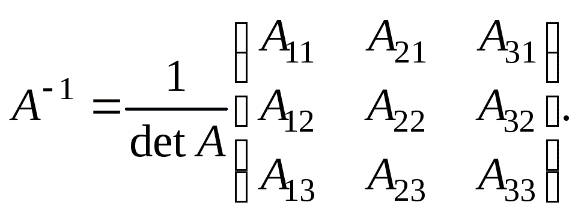


Рисунок 1.7 – Формула оберненої матриці

Піднесення матриці до степеню , буде матриця, яка дорівнює добутку самої на себе відповідну кількість разів.

Підносити до степеню можна тільки квадратні матриці (матриці у яких кількість рядків дорівнює кількості стовпчиків) [1].

## **Аналіз завдань, розв'язуваних з урахуванням їх представлення у матричному вигляді**

Матричне представлення даних для завдань є важливою складовою багатьох областей науки та техніки, таких як фізика, інженерія, комп'ютерна наука, економіка та статистика. Застосування матричного представлення дозволяє ефективно вирішувати багато завдань, що забезпечує підвищення продуктивності та точності обчислень.

Деякі завдання, які можна вирішити за допомогою матричного представлення, включають:

1. Вирішення систем лінійних рівнянь: система лінійних рівнянь може бути представлена в матричному вигляді, де коефіцієнти змінних записуються у вигляді матриці, а праві частини рівнянь записуються у вигляді вектора. Це дозволяє швидко та ефективно вирішувати системи лінійних рівнянь за допомогою методу Гауса чи інших методів [4].
2. Пошук власних значень та векторів: власні значення та вектори матриці можуть бути знайдені за допомогою матричних операцій. Це використовують у різних галузях, наприклад, в аналізі даних, теорії графів, машинному навчанні тощо [5].
3. Розв'язання задач оптимізації: матричне подання може бути використане для вирішення різних задач оптимізації, таких як пошук мінімуму функції або пошук найкращого наближення до деякого набору даних.
4. Розв'язання задачі знаходження зворотної матриці: зворотну матрицю можна знайти за допомогою матричних операцій. Це використовується в різних галузях, наприклад, теоретично ймовірностей, статистики і т.д..
5. Вирішення завдання стиснення даних: матричне подання може бути використане для стиснення даних, наприклад, за допомогою методу головних компонентів (PCA). Цей метод дозволяє скоротити розмірність даних, зберігаючи при цьому інформацію про найважливіші характеристики даних.
6. Розв'язання задачі багатовимірного шкалювання: багатомірне шкалювання використовується для візуалізації багатовимірних даних у дво- або тривимірному просторі. Матричне уявлення може бути використане для знаходження відстаней між об'єктами на основі їх характеристик, а потім для знаходження координат об'єктів у дво- чи тривимірному просторі.
7. Розв'язання задачі знаходження визначника матриці: визначник матриці може бути знайдений за допомогою матричних операцій, що може бути корисним у різних галузях, наприклад, у теорії ймовірностей, статистиці та теорії ігор.
8. Розв'язання задачі лінійного програмування: лінійне програмування – це метод оптимізації лінійної функції з обмеженнями. Матричне представлення може бути використане для запису обмежень та цільової функції, що дозволяє швидко та ефективно вирішувати задачі лінійного програмування.
9. Розв'язання задачі знаходження рангу матриці: ранг матриці – це число лінійно незалежних рядків або стовпців. Ранг матриці може бути знайдений за допомогою матричних операцій, що може бути корисним у різних областях, наприклад, у теорії графів, машинному навчанні і т.д..
10. Розв'язання задачі регресійного аналізу: регресійний аналіз – це метод аналізу даних, який використовується для побудови моделей залежності між змінними. Матричне уявлення може бути використане для запису даних та моделі, що дозволяє швидко та ефективно вирішувати завдання регресійного аналізу [6, 7, 8].

Тут наведено лише маленька частина того, щодо завдань, які можна вирішити за допомогою матричного представлення, адже вони включають ще дуже багато чого різного.

## **Особливості обробки даних при роботі з матрицями**

Матриці це одна з найбільш вживаних структур даних у чисельному аналізі, обробці зображень, машинному навчанні та інших областях. При роботі з матрицями важливо враховувати деякі особливості їх обробки даних, такі як:

1. Розмірність матриці: матриці можуть мати різні розмірності, і важливо враховувати це при роботі з ними. Наприклад, при додаванні двох матриць вони повинні мати однаковий розмір, інакше операція неможлива.
2. Операції з матрицями: існує багато операцій, які можна виконати з матрицями, такі як додавання, віднімання, множення, транспонування тощо. Важливо розуміти, які операції потрібно виконувати для досягнення бажаного результату.
3. Збереження даних: матриці можуть займати значну кількість пам'яті, тому важливо ефективно зберігати та управляти ними. Наприклад, можна використовувати розріджені матриці для збереження матриць з багатьма нулями.
4. Обчислювальна ефективність: деякі операції з матрицями можуть бути обчислювально витратними, особливо для великих матриць. Важливо розуміти, які операції можуть бути оптимізовані для забезпечення ефективної обробки даних.
5. Числова стійкість: матриці можуть мати поганий числовий стан, особливо для матриць з великою розмірністю або невизначеними елементами. Важливо враховувати це при виконанні операцій з матрицями та виборі методів обробки даних.
6. Доступ до елементів матриці: для роботи з елементами матриці можуть використовуватися різні методи, такі як індексація, ітерація, використання функцій для вибірки елементів тощо.
7. Матричні рівняння: матриці можуть бути використані для розв'язання різних матричних рівнянь, таких як системи лінійних алгебраїчних рівнянь. Важливо вивчити методи для розв'язання таких рівнянь.
8. Методи оптимізації: матриці можуть бути використані для розв'язання різних задач оптимізації, таких як мінімізація функцій витрат. Важливо вивчити методи оптимізації для роботи з матрицями.

Отже, обробка даних з використанням матриць може бути досить складною і вимагати глибоких знань та вмінь. Важливо розуміти особливості роботи з матрицями та мати достатній досвід, щоб ефективно виконувати різні операції та розв'язувати завдання з використанням цих структур [7, 8, 9].

# **АНАЛОГИ ТА ЗАСОБИ РОЗРОБКИ**

## **Аналіз існуючих систем**

**Mathway Matrix Calculator** - онлайн сервіс для роботи з матрицями, що дозволяє виконувати основні операції з матрицями, наприклад, додавання, віднімання, множення, обернена матриця та інші.

Mathway Matrix Calculator має більш-менш інтуїтивно зрозумілий інтерфейс, дозволяє користувачам виконувати основні операції з матрицями (додавання, віднімання, множення, обернення матриці). Найбільшим недоліком є те, що у цьому сервісі є обмеження на кількість елементів у матриці.

Сервіс платний, можна отримати підписку на місяць або рік, тому не має великої популярності серед користувачів. Крім того, сервіс не веде активної реклами на сайті та в соціальних мережах. Mathway Matrix Calculator зображений на рис. 2.1:

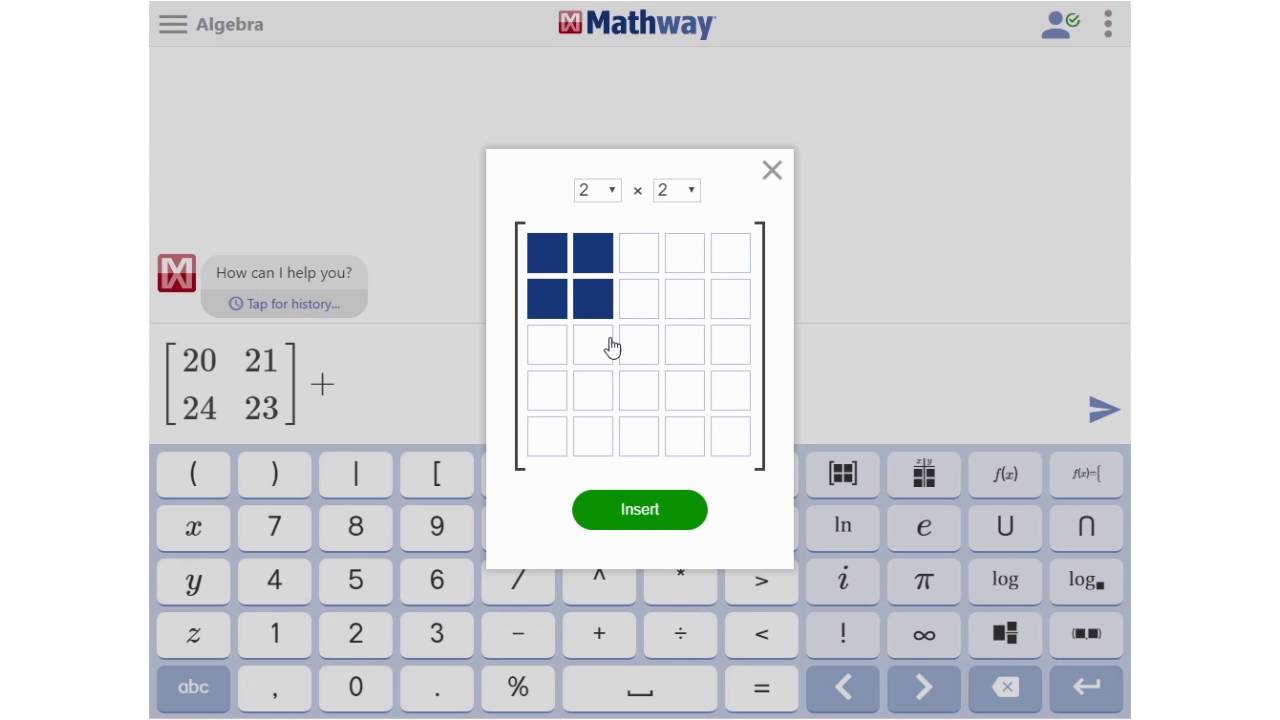


Рисунок 2.1 – Mathway Matrix Calculator

**Symbolab Matrix Calculator -** онлайн сервіс для роботи з матрицями, що дозволяє виконувати основні операції з матрицями, а також знаходити визначник та ранг матриці.

Якщо порівнювати з Mathway Matrix Calculator, то має більш розширений функціонал, але не може працювати з матрицями більшого розміру, а ось інтерфейс набагато складніший, може викликати деякі труднощі у користувачів.

Як і Mathway Matrix Calculator сервіс платний та має підписку на місяць або рік, але його перевагою є те що, має деякі доступні безкоштовні функції. Сервіс має активну рекламу на сайті та в соціальних мережах. Symbolab Matrix Calculator зображений на рис. 2.2:

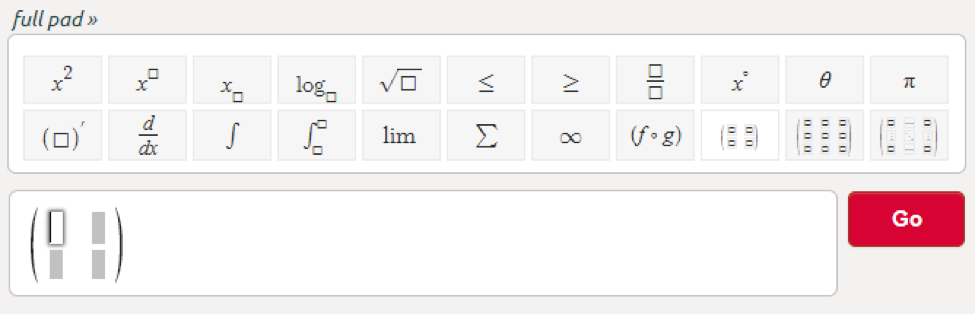


Рисунок 2.2 – Symbolab Matrix Calculator

**Desmos Matrix Calculator –** безкоштовний онлайн калькулятор матриць з можливістю виконання операцій з матрицями та використанням матриць у системах лінійних рівнянь.

Виконує основні операції з матрицями, як і інші конкуренти (додавання, віднімання, множення, обернення матриці), а також має додатковий функціонал, який включає обчислення детермінанту, знаходження відведення матриці до канонічної форми, розклад на добуток елементарних матриць та інші. Найбільшим недоліком є обмеження у розмірі матриць (найбільше це 6×6).

Інтерфейс трохи складний, та може викликати на початку використання труднощі у користувачів. Значною різницею між двома іншими калькуляторами є те, що сервіс повністю безкоштовний. Має малу активність в соціальних мережах, але користується популярністю серед викладачів та студентів з математики. Desmos Matrix Calculator зображений на рис. 2.3:

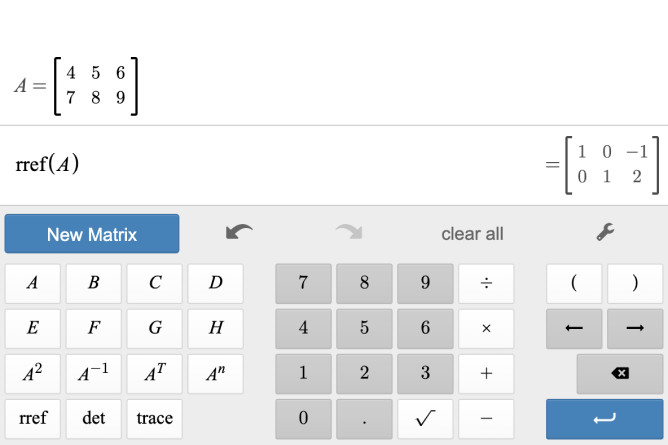


Рисунок 2.3 – Desmos Matrix Calculator

**Matrix Calculator** - додаток для Android, який дозволяє виконувати операції з матрицями на вашому мобільному пристрої.

Найбільшою відмінністю від усіх інших калькуляторів є те, що його потрібно завантажувати на свій телефон, крім того працювати він буде лише на Android. Функції включають додавання, віднімання, множення, транспонування, обернення та інші. Додаткові функції включають знаходження детермінанту, відведення до канонічної форми та знаходження рангу матриці. Недоліком є обмеження в розмірі матриць, які можна опрацьовувати.

Має доволі зручний інтерфейс, який має можливість зберігати результати та використовувати їх далі у інших додатках.

Сервіс безкоштовний, але має постійну рекламу. Доступна платна версія без реклами. Крім того, сервіс має рекламу на сторінці в Google Play. Matrix Calculator зображено на рис. 2.4:

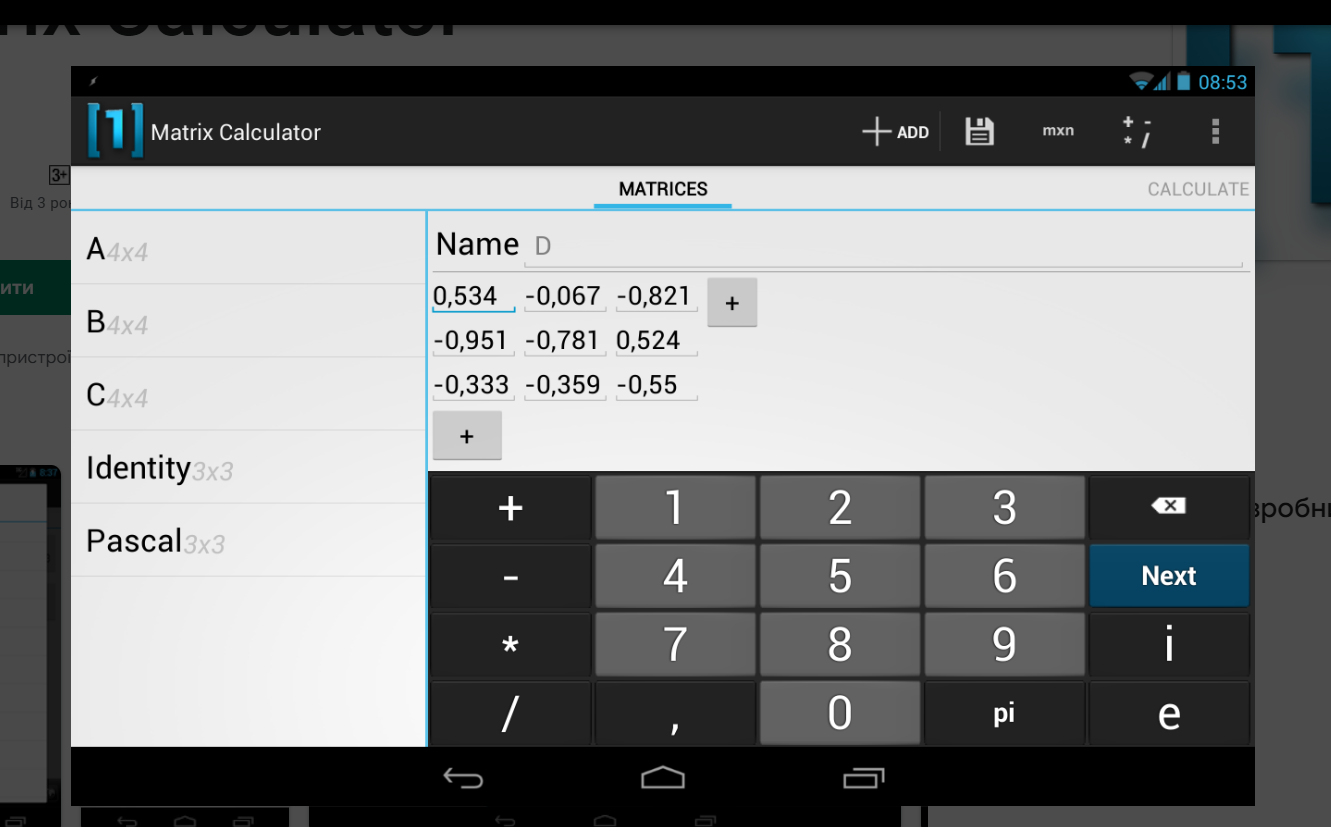


Рисунок 2.4 – Matrix Calculator

Альтернативою до попереднього додатку є **Matrix Calculator Pro -** додаток для iOS, який дозволяє виконувати операції з матрицями на вашому мобільному пристрої.

Має дуже велику схожість із Matrix Calculator, має той же функціонал, зручний інтерфейс, який має можливість зберігати результати та використовувати їх далі у інших додатках.

Найбільшою різницею є те, що версія платна, яка коштує $1,99 без реклами та з додатковими функціями. Рекламна стратегія майже відсутня, додаток доступний для завантаження в App Store.

Аналізуючи матричний калькулятор конкурентів, можна зробити ряд висновків:

* матричні калькулятори користуються попитом серед студентів, викладачів, інженерів та інших спеціалістів, які працюють у сферах математики та інженерії;
* можна виділити різні типи програм серед конкурентів. Наприклад: онлайн-сервіси, додатки для комп'ютерів і мобільних пристроїв з різними функціями та інтерфейсами;
* серед основних конкурентів виділяються Desmos Matrix Calculator, Mathway і Symbolab. Вони мають більші можливості, зручний інтерфейс і може використовуватися як на комп’ютері, так і на мобільних пристроях;
* є додатки, призначені для роботи з матрицями і зі зручними інтерфейсами для мобільних пристроїв. Ці програми дозволяють виконувати матричні операції та зберігати результати для подальшого використання.

Таким чином, висновок нашого аналізу конкурентів полягає в тому, що хоча ринок матричних калькуляторів досить насичений, але він зручний і потрібний користувачам. Створення конкурентоспроможного калькулятора вимагає не лише зосередження на функціях і недоліках існуючих програм і їх виправлення, але й додавання нових функцій та інтерфейсів, які відповідають потребам користувачів.

## **Обґрунтування вибору засобів та методів розробки**

Для написання нашого застосунку ми обрали середовище розробки Visual Studio 2019.

Visual Studio - це інтегроване середовище розробки, що дозволяє розробляти програми для різних платформ, включаючи Windows, Linux, Android та iOS.

Visual Studio має ряд бібліотек та інструментів для роботи з матрицями та векторами, зокрема бібліотеку Armadillo, яка є потужним інструментом для роботи з лінійною алгеброю. Крім того, Visual Studio має вбудовану можливість створювати графічний інтерфейс користувача для матричного калькулятору з використанням інструментів, таких як Windows Forms або WPF.

Також, Visual Studio підтримує популярні мови програмування, такі як C++, C#, а також мови скриптування, такі як Python, що можуть бути використані для розробки матричного калькулятору [11].

Мову програмування ми обрали С++.

C++ — це мова програмування загального призначення, яка підтримує підхід до об’єктно-орієнтованого програмування. У C++ є вбудовані функції для роботи з матрицями та векторами, а також бібліотеки для чисельних обчислень, які можна використовувати для розробки матричних калькуляторів.

Наприклад, бібліотека Armadillo є потужним інструментом для роботи з лінійною алгеброю на C++. Він містить багато функцій для роботи з матрицями та векторами, таких як додавання, віднімання, множення, вирішення систем лінійних рівнянь, знаходження власних значень і векторів.

Окрім Armadillo, у C++ є багато бібліотек для роботи з матрицями та векторами, таких як Eigen, Blitz++ і Boost. Ці бібліотеки можна використовувати для розробки матричних калькуляторів у C++ і надавати потужні функції для роботи з лінійною алгеброю.

Ви можете розробляти графічні інтерфейси користувача, використовуючи такі бібліотеки, як Qt і wxWidgets. Вони надають можливість створювати графічні інтерфейси користувача, які можна використовувати для введення та виведення матриць, виконання різноманітних операцій [12].

Для того, щоб мати змогу працювати з програмою необхідно підключити певні стандартні бібліотеки. Це можливо зробити за допомогою директиви #include:

1. iostream – бібліотека необхідна для введення і виведення даних в мові програмування С++;
2. ctime – бібліотека для роботи з часом, вона містить функції для роботи з часом та датами, такі як time() для отримання поточного часу та ctime() для конвертування часу в рядок символів;
3. string – бібліотека для роботи з рядками символів (стрічками), вона містить функції для роботи зі стрічками, такі як length() для отримання довжини стрічки та substr() для вирізання частини стрічки;
4. cmath – бібліотека математичних функцій, вона містить функції для математичних операцій, таких як sin(), cos(), sqrt(), log() та інші;
5. msclr\marshal\_cppstd.h - ця бібліотека дозволяє конвертувати рядки символів між .NET-кодом та нативним кодом. Вона забезпечує механізм конвертування рядків між .NET-кодом та нативним кодом, що дозволяє програмі взаємодіяти з кодом, написаним на мовах, що працюють в середовищі .NET, таких як C# та VB.NET [13].

Також необхідно підключити деякі директиви:

* Директива #pragma once повідомляє компілятору, що файл має бути включений лише один раз під час компіляції, незалежно від того, скільки різних файлів він містить. Це корисно, щоб уникнути проблем, які можуть виникнути з подвійним включенням файлів заголовків [14].
* Директива #pragma warning(disable :4244) вимикає попередження компілятора в коді 4244. Це призводить до можливої втрати точності під час відкидання чисел різних типів. Вимкнути це попередження корисно, якщо розробник впевнений, що втрата точності не є проблемою в цьому контексті, і попередження приховає більш важливе повідомлення про помилку [15].

Крім того необхідно підключити деякі простори імен:

* using namespace System – вказує на те, що всі елементи простору імен System будуть доступні безпосередньо в коді, без необхідності вказувати префікс System:: перед назвами класів і функцій;
* using namespace System::ComponentModel – додає в простір імен доступ до класів інтерфейсів компонентів, які використовуються віджетами Windows Forms, таких як BindingSource, Container і т.д.;
* using namespace System::Collections – включає в себе класи колекцій, такі як List, Dictionary, ArrayList та інші;
* using namespace System::Windows::Forms – надає доступ до класів, пов'язаних з розробкою інтерфейсу користувача з використанням Windows Forms, таких як Form, Button, Label, TextBox та інші;
* using namespace System::Data – дозволяє використовувати класи для роботи з базами даних, такі як DataSet, DataTable, SqlConnection та інші [16].

## **Ключові фічі**

Основною метою нашого калькулятора є обчислення математичних операцій з матрицями швидко та ефективно. У цьому розділі описані ключові фічі калькулятора матриць, які допомагають досягти цієї мети.

Ключові фічі:

1. Зрозумілий та простий інтерфейс

Калькулятор містить інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко вводити дані та опрацьовувати їх. Основні елементи інтерфейсу включають поле для введення матриці, різноманітні кнопки для виконання математичних операцій та виведення результатів обчислень, а також інформаційні повідомлення про помилки. Приклад інтерфейсу калькулятора матриці наведено на рис. 2.5:

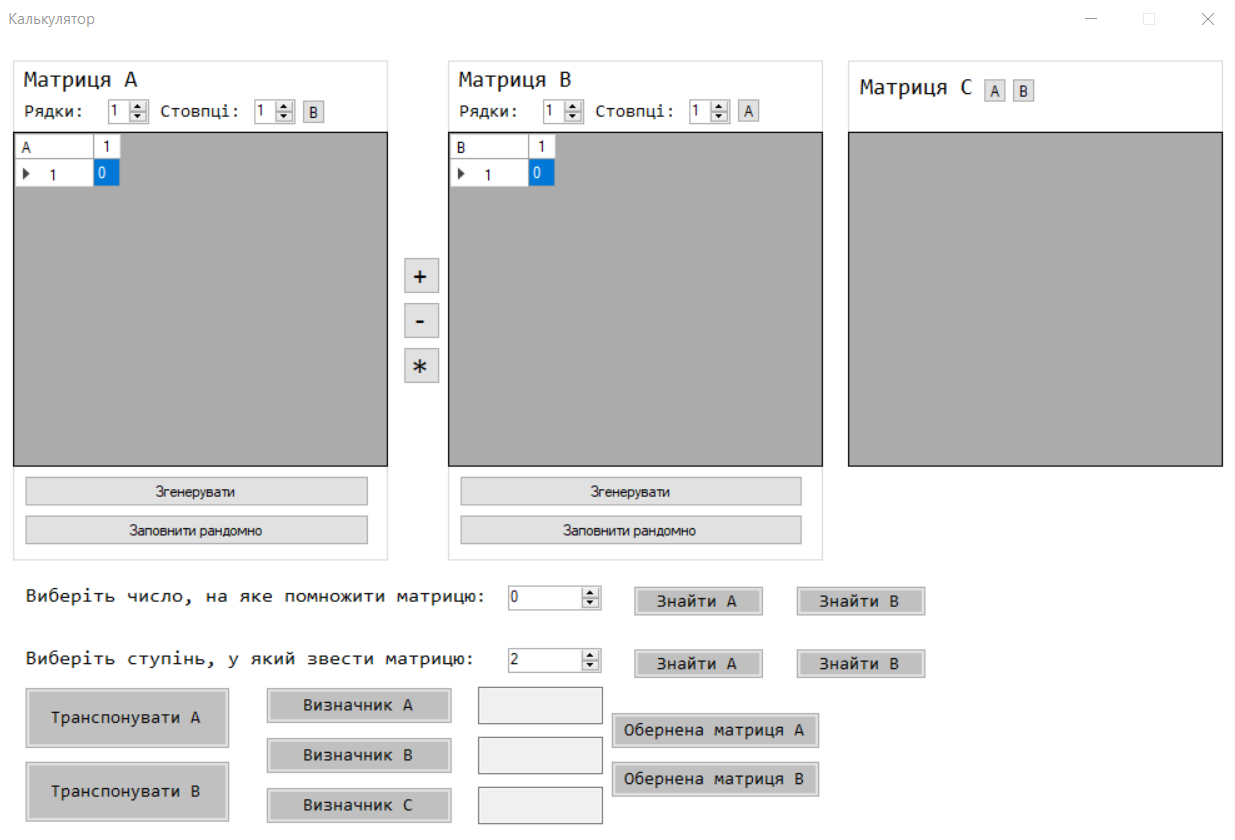


Рисунок 2.5 –Приклад інтерфейсу калькулятора матриці

1. Гнучкість

Калькулятор може опрацьовувати матриці в різних форматах. Користувач може вводити матрицю вручну, використовуючи клавіші на клавіатурі або заповнювати її автоматично. Крім того, калькулятор може опрацьовувати матриці різної розмірності, яку користувач вказує самостійно та різного рангу. Приклад рандомного заповнення матриці показано на рис. 2.6:

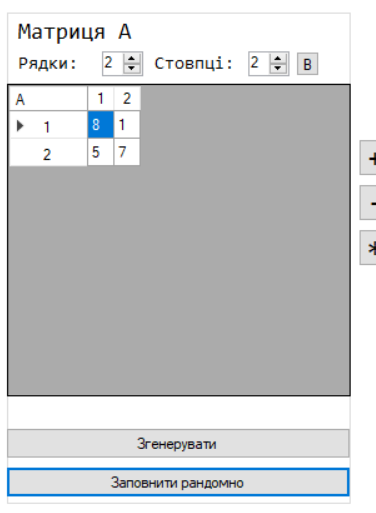


Рисунок 2.6 – Приклад рандомного заповнення матриці

1. Виведення результатів в зручному форматі

Виведення результатів відбувається у зручному та зрозумілому форматі для користувача. Приклад виведення результатів показано на рис. 2.7:

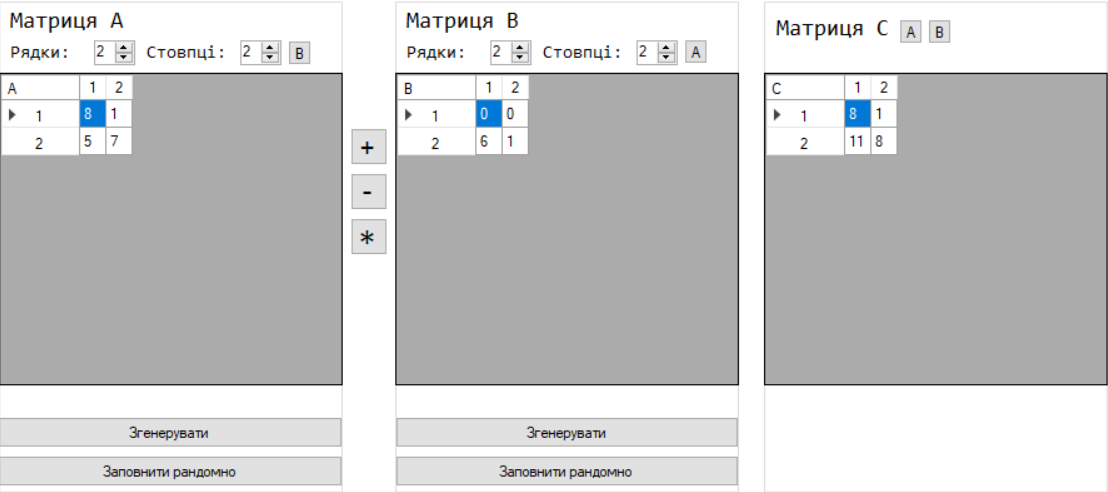


Рисунок 2.7 – Приклад виведення результату

1. Перенос матриці із одної в іншу

Калькулятор містить функцію для перенесення записів з однієї матриці в іншу. Користувач має можливість швидко взаємодіяти з матрицями та переносити результати в інше місце для подальшого користування. Для цього були створені відповідні кнопки, які зображені на рис. 2.8:

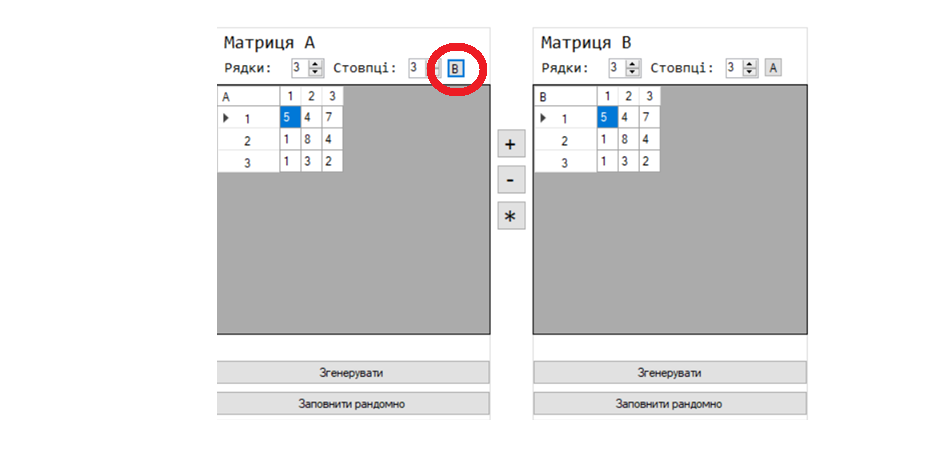


Рисунок 2.8 – Кнопка для перенесення матриці із А у В

# **РЕАЛІЗАЦІЯ СИСТЕМИ**

## **Розробка алгоритмів**

### **3.1.1 Опис функції MinusMatrix ()**

Функція *MinusMatrix()* здійснює відніманняматриць.

Аргументом функції є матриця *А* і матриця *В*, а також змінні *N* – кількість рядків, *M* – кількість стовпців.

Функція має тип double\*\* - повертає тип значення double\*\* засобами *return,* для створення динамічної матриці *С.*

Словесний опис алгоритму:

1. Спочатку потрібно виділити динамічну пам’ять для матриці С

*double\*\* C = new double\* [N];*

1. За допомогою циклу *for* виділяємо пам’ять для *С[i]*

*for (int i = 0; i < N; i++) { C[i] = new double[M]; }*

1. За допомогою вкладених циклів *for* відняти *A[i][j] і B[i][j]* та вивести C[i][j]

*for (int i = 0; i < N; i++) {*

*for (int j = 0; j < M; j++) {*

*C[i][j] = A[i][j] - B[i][j]; } }*

1. Повернути С

*return C;*

### **3.1.2 Опис функції UmnMatrixChislo()**

Функція *UmnMatrixChislo()* здійснює множенняматриць на число.

Аргументом функції є матриця *А* і матриця *В*, а також змінні *N* – кількість рядків, *M* – кількість стовпців та *num* – число на яке потрібно помножити матрицю.

Функція має тип double\*\* - повертає тип значення double\*\* засобами *return,* для створення динамічної матриці *С.*

Словесний опис алгоритму:

1. Спочатку потрібно виділити динамічну пам’ять для матриці С

*double\*\* C = new double\* [N];*

1. За допомогою циклу *for* виділяємо пам’ять для *С[i]*

*for (int i = 0; i < N; i++) { C[i] = new double[M]; }*

1. За допомогою вкладених циклів *for* помножити *A[i][j]* на число та вивести C[i][j]

*for (int i = 0; i < N; i++) {*

*for (int j = 0; j < M; j++) {*

*C[i][j] = A[i][j] \* num; } }*

1. Повернути С

*return C;*

### **3.1.3 Опис функції pow()**

Функція *pow()* здійснює зведенняматриць у степінь.

Аргументом функції є матриця *А*, а також змінні *N* – кількість рядків, *M* – кількість стовпців та *num* – степінь у яку треба звести матрицю.

Функція має тип double\*\* - повертає тип значення double\*\* засобами *return,* для створення динамічної матриці *С.*

Словесний опис алгоритму:

1. Спочатку потрібно виділити динамічну пам’ять для матриці С

*double\*\* C = new double\* [N];*

1. За допомогою циклу *for* виділяємо пам’ять для *С[i]*

*for (int i = 0; i < N; i++) { C[i] = new double[M]; }*

1. Присвоюємо *С* значення *А* та за допомогою циклу *for* зводимо матрицю у степінь

*for (int i = 0; i < num; i++) { C[i][j] =UmnMatrix(A, C, N, M, N, M; }*

1. Повернути С

*return C;*

### **3.1.4 Опис функції det()**

Функція *det()* здійснює знаходження визначникаматриці.

Аргументом функції є змінні *N* – кількість рядків, *M* – кількість стовпців.

Функція має тип double - повертає тип значення double засобами *return.*

Словесний опис алгоритму:

1. Оголошуємо змінну *Det(0)* та перевіряємо чи кількість рядків дорівнює 1, якщо так, то повертаємо перший елемент

*double Det(0);*

*if (N == 1) { return M[0][0]; }*

1. Якщо кількість рядків 2, то виконуємо наступні дії та повертаємо визначник

*else if (N == 2) {*

*Det = M[0][0] \* M[1][1] - M[0][1] \* M[1][0];*

*return Det; }*

1. Якщо кількість рядків 3, то виконуємо наступні дії та повертаємо визначник

*else if (N == 3) {*

*for (int i = 0; i < N; i++) {*

*Det += M[0][((i) % N)] \* M[1][((i + 1) % N)] \* M[2][((i + 2) % N)] - M[0][((i + 2) % N)] \* M[1][((i + 1) % N)] \* M[2][((i) % N)]; }*

*return Det; }*

1. Якщо кількість рядків 4, то виконуємо наступні дії та повертаємо визначник

*else if (N == 4) {*

*for (int i = 0; i < N; i++) {*

*Det += M[0][(i) % N] \* M[1][(i + 1) % N] \* M[2][(i + 2) % N] \* M[3][(i + 3) % N] - M[0][(i + 3) % N] \* M[1][(i + 2) % N] \* M[2][(i + 1) % N] \* M[3][(i) % N]; }*

*return Det; }*

1. Якщо кількість рядків 5, то виконуємо наступні дії та повертаємо визначник

*else if (N == 5) {*

*for (int i = 0; i < N; i++) {*

*Det += M[0][(i) % N] \* M[1][(i + 1) % N] \* M[2][(i + 2) % N] \* M[3][(i + 3) % N] \* M[4][(i + 4) % N] - M[0][(i + 4) % N] \* M[1][(i + 3) % N] \* M[2][(i + 2) % N] \* M[3][(i + 1) % N] \* M[4][(i) % N]; }*

*return Det; }*

1. Якщо кількість рядків 6, то виконуємо наступні дії та повертаємо визначник

*else if (N == 6) {*

*for (int i = 0; i < N; i++) {*

*Det += M[0][(i) % N] \* M[1][(i + 1) % N] \* M[2][(i + 2) % N] \* M[3][(i + 3) % N] \* M[4][(i + 4) % N] \* M[5][(i + 5) % N] - M[0][(i + 5) % N] \* M[1][(i + 4) % N] \* M[2][(i + 3) % N] \* M[3][(i + 2) % N] \* M[4][(i + 1) % N] \* M[5][(i) % N]; }*

*return Det; }*

1. Якщо кількість рядків 7, то виконуємо наступні дії та повертаємо визначник

*else if (N == 7) {*

*for (int i = 0; i < N; i++) {*

*Det += M[0][(i) % N] \* M[1][(i + 1) % N] \* M[2][(i + 2) % N] \* M[3][(i + 3) % N] \* M[4][(i + 4) % N] \* M[5][(i + 5) % N] \* M[6][(i + 6) % N] - M[0][(i + 6) % N] \* M[1][(i + 5) % N] \* M[2][(i + 4) % N] \* M[3][(i + 3) % N] \* M[4][(i + 2) % N] \* M[5][(i + 1) % N] \* M[6][(i) % N]; }*

*return Det; }*

1. Якщо кількість рядків 8, то виконуємо наступні дії та повертаємо визначник

*else if (N == 8) {*

*for (int i = 0; i < N; i++) {*

*Det += M[0][(i) % N] \* M[1][(i + 1) % N] \* M[2][(i + 2) % N] \* M[3][(i + 3) % N] \* M[4][(i + 4) % N] \* M[5][(i + 5) % N] \* M[6][(i + 6) % N] \* M[7][(i + 7) % N] - M[0][(i + 7) % N] \* M[1][(i + 6) % N] \* M[2][(i + 5) % N] \* M[3][(i + 4) % N] \* M[4][(i + 3) % N] \* M[5][(i + 2) % N] \* M[6][(i + 1) % N] \* M[7][(i) % N]; }*

*return Det; }*

1. Якщо кількість рядків 9, то виконуємо наступні дії та повертаємо визначник

*else if (N == 9) {*

*for (int i = 0; i < N; i++) {*

*Det += M[0][(i) % N] \* M[1][(i + 1) % N] \* M[2][(i + 2) % N] \* M[3][(i + 3) % N] \* M[4][(i + 4) % N] \* M[5][(i + 5) % N] \* M[6][(i + 6) % N] \* M[7][(i + 7) % N] \* M[8][(i + 8) % N] - M[0][(i + 8) % N] \* M[1][(i + 7) % N] \* M[2][(i + 6) % N] \* M[3][(i + 5) % N] \* M[4][(i + 4) % N] \* M[5][(i + 3) % N] \* M[6][(i + 2) % N] \* M[7][(i + 1) % N] \* M[8][(i) % N]; }*

*return Det; }*

1. Якщо кількість рядків 10, то виконуємо наступні дії та повертаємо визначник

*else if (N == 10) {*

*for (int i = 0; i < N; i++) {*

*Det += M[0][(i) % N] \* M[1][(i + 1) % N] \* M[2][(i + 2) % N] \* M[3][(i + 3) % N] \* M[4][(i + 4) % N] \* M[5][(i + 5) % N] \* M[6][(i + 6) % N] \* M[7][(i + 7) % N] \* M[8][(i + 8) % N] \* M[9][(i + 9) % N] - M[0][(i + 9) % N] \* M[1][(i + 8) % N] \* M[2][(i + 7) % N] \* M[3][(i + 6) % N] \* M[4][(i + 5) % N] \* M[5][(i + 4) % N] \* M[6][(i + 3) % N] \* M[7][(i + 2) % N] \* M[8][(i + 1) % N] \* M[9][(i) % N]; }*

*return Det; }*

### **3.1.5 Опис функції Rounding1()**

Функція *Rounding1()* здійснює округлення матриць.

Аргументом функції є змінна *Num*.

Функція має тип *double* - повертає тип значення double засобами *return*.

Словесний опис алгоритму:

1. Оголошуємо змінну *integer* та присвоюємо їй значення *Num*, а змінній *Num* присвоюємо значення *Num-integer*. Крім цього оголошуємо булеву змінну *zhak* та присвоюємо їй значення *false*. А також оголошуємо змінну *rere* та виконуємо наступні дії

*int integer = Num;*

*Num -= integer;*

*bool znak = false;*

*long long rere = (long long)(Num \* 1000000000000) % 1000000000000;*

1. Перевіряємо якого значення у нас змінна *rere* і в залежності від цього виконуємо дії

*if (rere < 0) { rere = -rere; znak = true; }*

*if (rere % 10 >= 5) { rere += 10; rere -= rere % 10; }*

*else if (rere % 10 < 5) { rere -= rere % 10; }*

*if ((rere % 100 / 10) >= 5) { rere += 100; rere -= (rere % 100); }*

*else if ((rere % 100 / 10) < 5) { rere -= (rere % 100); }*

*if ((rere % 1000 / 100) >= 5) { rere += 1000; rere -= ((rere % 1000)); }*

*else if ((rere % 1000 / 100) < 5) { rere -= ((rere % 1000)); }*

*if ((rere % 10000 / 1000) >= 5) { rere += 10000; rere -= (rere % 10000); }*

*else if ((rere % 10000 / 1000) < 5) { rere -= (rere % 10000); }*

*if ((rere % 100000 / 10000) >= 5) { rere += 100000; rere -= (rere % 100000); }*

*else if ((rere % 100000 / 10000) < 5) { rere -= (rere % 100000); }*

*if ((rere % 1000000 / 100000) >= 5) { rere += 1000000; rere -= (rere % 1000000); }*

*else if ((rere % 1000000 / 100000) < 5) { rere -= (rere % 1000000); }*

*if ((rere % 10000000 / 1000000) >= 5) { rere += 10000000; rere -= (rere % 10000000); }*

*else if ((rere % 10000000 / 1000000) < 5) { rere -= (rere % 10000000); }*

1. Присвоюємо змінній *Num* значення *rere/ 1000000000000* та додаємо integer. Здійснюємо перевірку над змінною *znak*

*Num = (double)rere / 1000000000000;*

*Num += integer;*

*if (znak) { Num = -Num; }*

1. Повертаємо *Num*

*return Num*

## **Розробка інтерфейсу та зв'язування його з алгоритмами**

Для створення інтерфейсу у нас є 3 файли: MyForm.h, MyForm.h[Конструктор], MyForm.cpp.

В MyForm.h[Конструктор] можна створювати інтерфейс програми та задавати властивості для кнопок. Приклад файлу MyForm.h[Конструктор] наведено на рис. 3.1:

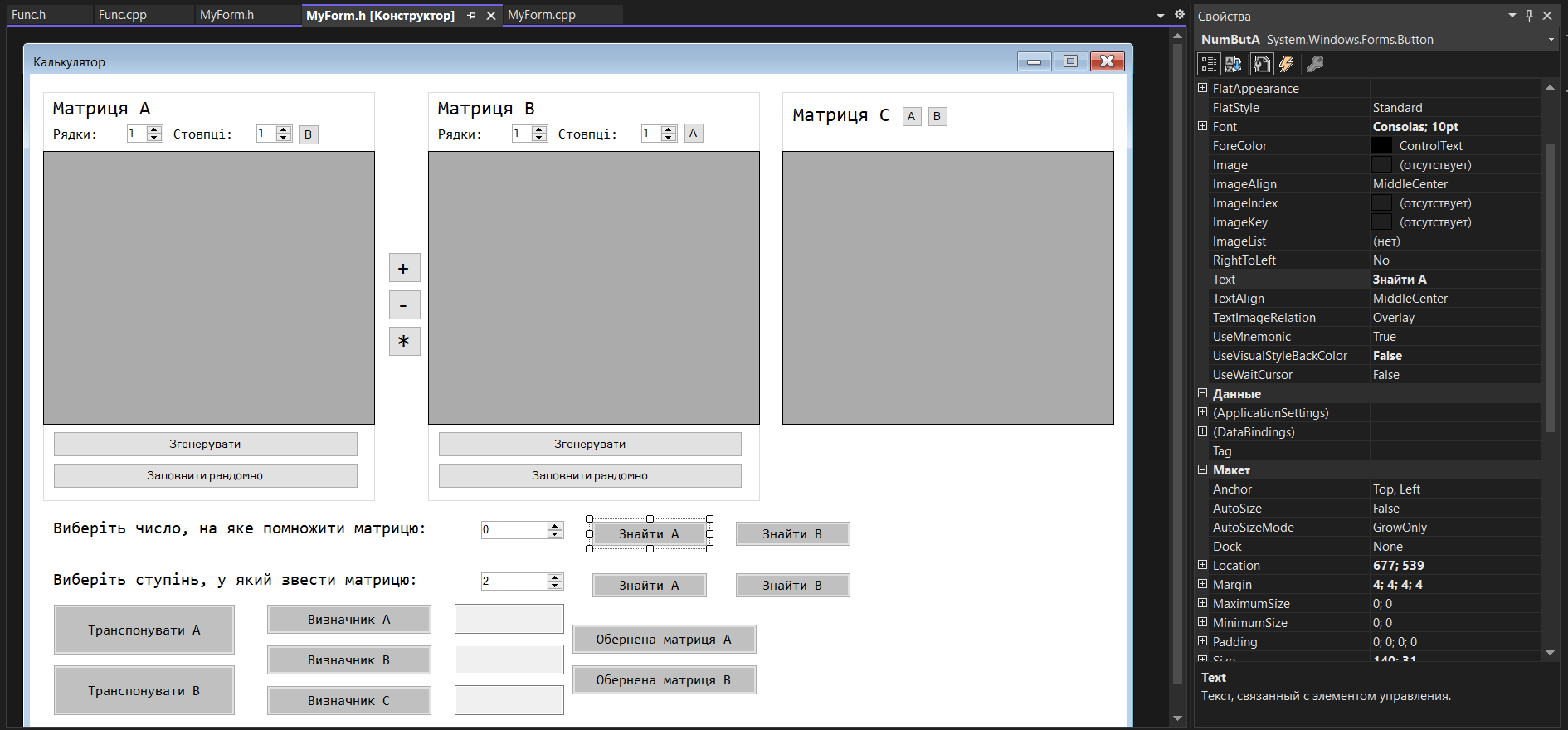


Рисунок 3.1 – Приклад файлу MyForm.h[Конструктор]

Властивості для кнопки “Знайти А” наведено на рис. 3.2:

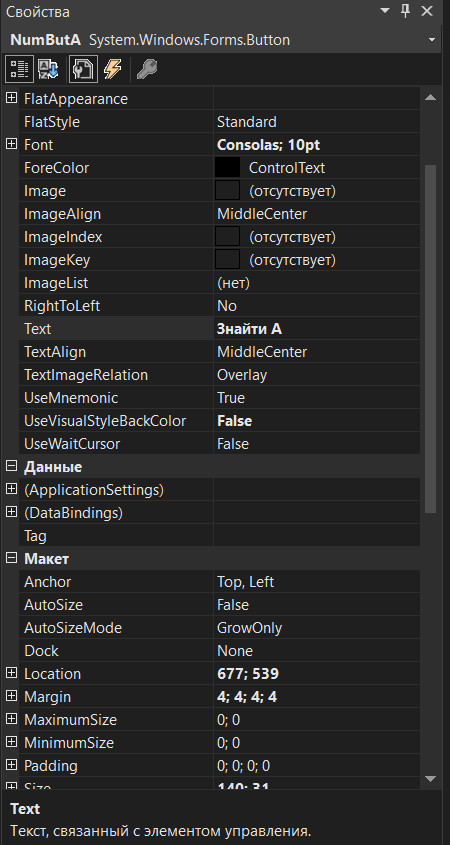


Рисунок 3.2 – Властивості для кнопки “Знайти А”

В MyForm.h містяться заголовки функцій для з’єднання інтерфейсу із ними. Приклад файлу MyForm.h показано на рис. 3.3-4:

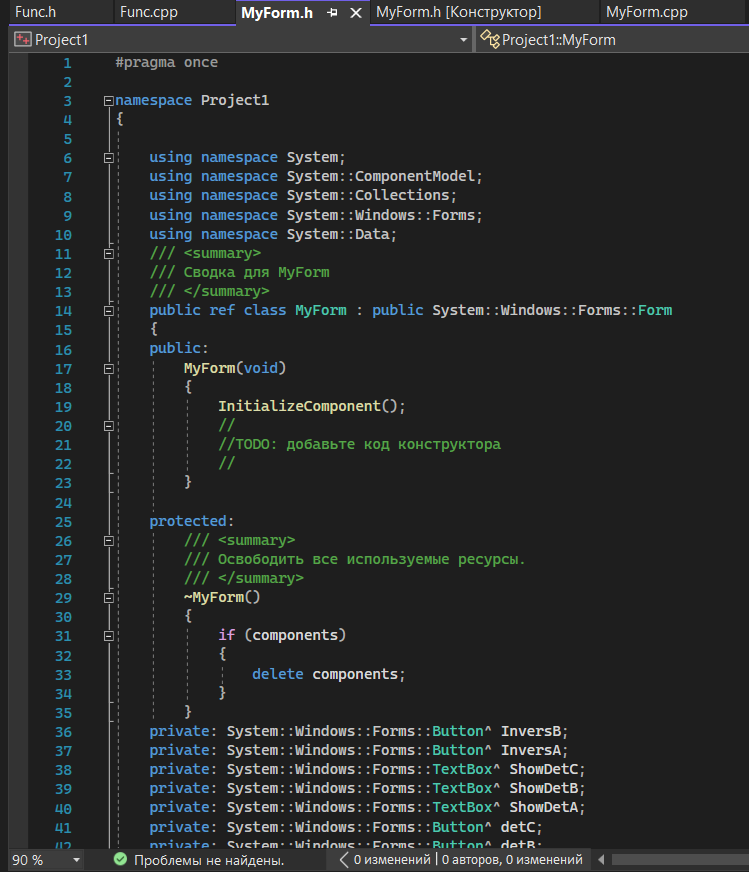


Рисунок 3.3 – Приклад файлу MyForm.h

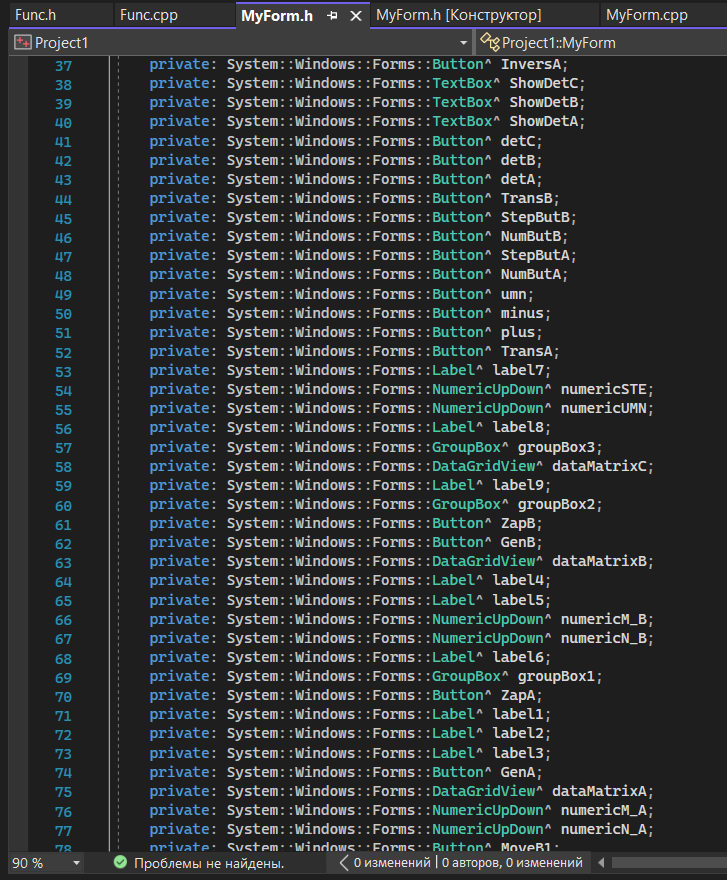


Рисунок 3.4 –Приклад файлу MyForm.h

В MyForm.cpp містяться функції з’єднання інтерфейсу із ними. Приклад функції з’єднання кнопки “Знайти А” із функцією NumButA\_Click наведено на рис. 3.5:

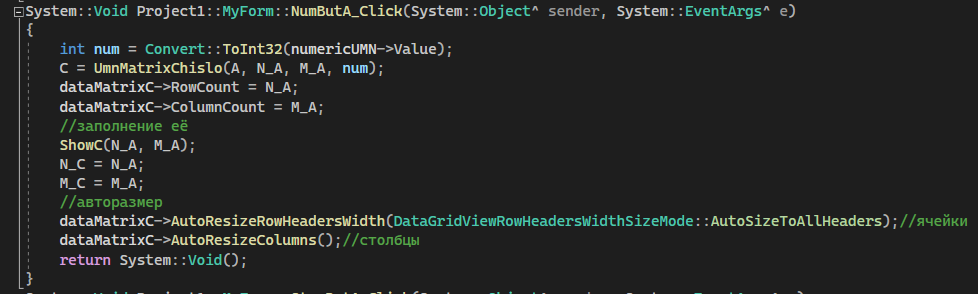


Рисунок 3.5 – Приклад функції з’єднання кнопки “Знайти А” із функцією NumButA\_Click

## **Опис процесу розгортання програми**

Перелік етапів розробки:

* створення проекту “Порожній проект CLR (.NET Framework)”;
* встановлення необхідних плагінів;
* створення інтерфейсу;
* створення логіки;
* компіляція проекту;
* запуск на комп’ютері \*.exe файла

Логіку програми зображено на рис. 3.6

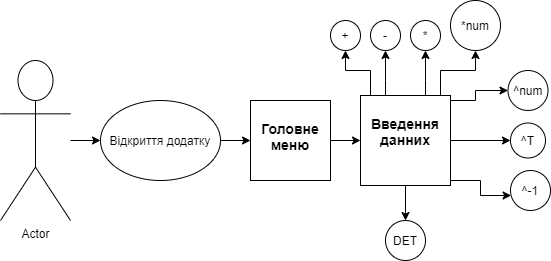
****

Рисунок 3.6 ­– Логіка програми

В програмі наявно 5 файлів:

* Func.h – файл заголовків функцій;
* Func.cpp – файл реалізації функцій;
* MyForm.h – файл для заголовків функцій для з’єднання інтерфейсу із функціями;
* MyForm.h[Конструктор] – файл для створення інтерфейсу;
* MyForm.cpp – файл функцій з’єднання інтерфейсу із функціями.

Опис функцій програми представлено у таблиці 1

Таблиця 1 Виконуючі функції

|  |  |  |  |
| --- | --- | --- | --- |
| Назва функції | Призначення | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | 2 | 3 | 4 |
| GenA\_Click() | Очищення матриці A | \* | \* |
| GenB\_Click() | Очищення матриці B | \* | \* |
| ZapA\_Click() | Заповнення рандомно матрицю A | \* | \* |
| ZapB\_Click() | Заповнення рандомно матрицю B | \* | \* |
| plus\_Click() | Додавання A і B | \* | \* |
| minus\_Click() | Різниця A і B | \* | \* |
| umn\_Click() | Множення A і B | \* | \* |
| NumButA\_Click() | Множення A і num | \* | \* |
| StepButA\_Click() | Зведення A до ступеню num | \* | \* |
| NumButB\_Click() | Множення B і num | \* | \* |
| StepButB\_Click() | Зведення B до ступеню num | \* | \* |
| TransA\_Click() | Транспонування A | \* | \* |
| TransB\_Click() | Транспонування B | \* | \* |
| detA\_Click() | Визначник A | \* | \* |
| detB\_Click() | Визначник B | \* | \* |
| detC\_Click() | Визначник C | \* | \* |
| InversA\_Click() | Обернена матриця A | \* | \* |
| InversB\_Click() | Обернена матриця B | \* | \* |
| dataMatrixA\_CellEndEdit() | Event на завершення редагування даних | \* | \* |

Продовження таблиці 1

|  |  |  |  |
| --- | --- | --- | --- |
| dataMatrixB\_CellEndEdit() | Event на завершення редагування даних | \* | \* |
| MyForm\_Load() | Початкова генерація матриці | \* | \* |
| ShowA() | Виводить матрицю A | \* | \* |
| ShowB() | Виводить матрицю B | \* | \* |
| ShowC(int N, int M) | Виводить матрицю C | N - кількість рядків M - кількість стовпців | \* |
| ShowCUMN() | Виводить матрицю C |  | \* |
| ShowCTrans(int N, int M) | Виводить транспоновану матрицю | N - кількість рядків M - кількість стовпців | \* |
| AddMatrix(double\*\* A, double\*\* B, int N, int M) | Сума матриць | A - Матриця A  B - Матриця B  N - Кількість рядків  M - Кількість стовпців | Матриця C |
| MinusMatrix(double\*\* A, double\*\* B, int N, int M) | Різниця матриць | A - Матриця A  B - Матриця B  N - Кількість рядків  M - Кількість стовпців | Матриця C |
| UmnMatrix(double\*\* A, double\*\* B, int N\_A, int M\_A, int N\_B, int M\_B) | Множення матриць | A - Матриця A B - Матриця B N\_A - Кількість рядків A M\_A - Кількість стовпців A N\_B - Кількість рядків B M\_B - Кількість стовпців B | Матриця C |
| UmnMatrixChislo(double\*\* A, int N, int M, int num) | Множення матриці на число | A - Матриця A  B - Матриця B  N - Кількість рядків  M - Кількість стовпців | Матриця C |
| Transp(double\*\* A, int N, int M) | Транспонування матриці | A - Матриця A  N - Кількість рядків  M - Кількість стовпців | Матриця C |
| Pow(double\*\* A, int N, int M, int num) | Зведення до ступеню | A - Матриця A  N - Кількість рядків  M - Кількість стовпців | Матриця C |
| Invers(double\*\* A, int N) | Обернена матриця | A - Матриця A  N - Кількість рядків | Матриця C |

Продовження таблиці 1

|  |  |  |  |
| --- | --- | --- | --- |
| Det(double\*\* Matrix, int N) | Визначник матриці | A - Матриця A  N - Кількість рядків | Double значення |
| Rounding(double Num) | 5 знаків після коми точність округлення | Num - значення яке треба округлити | Num |
| Rounding1(double Num) | 2 знака після коми точність округлення | Num - значення яке треба округлити | Num |

## **Тестування роботи програми**

Мета тестування: виявлення функціональних помилок, невідповідностей технічному завданню і очікуванням Замовника шляхом реалізації стандартних, а також нетривіальних тестових сценаріїв.

Класифікація функцій, які будуть протестовані:

1) Створення матриць A, B, C;

2) Тестування "+";

3) Тестування "-";

4) Тестування "\*";

5) Тестування "Згенерувати";

6) Тестування "Заповнити рандомно";

7) Тестування множення на число;

8) Тестування зведення до степеню;

9) Тестування транспонування;

10) Тестування визначників;

11) Тестування оберненої матриці;

Календарний план тестування наведено у таблиці 2. Та результати тестів у таблиці 3.

Таблиця 2 Календарний плані

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тести | Дата початку | Дата кінця | Дні | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 1 | 20.04.2023 | 30.04.2023 | х | x |  |  |  |  |  |  |  |  |  |
| 2 |  | x | x |  |  |  |  |  |  |  |  |
| 3 |  |  | x | x |  |  |  |  |  |  |  |
| 4 |  |  |  | x | x |  |  |  |  |  |  |
| 5 |  |  |  |  | x | x |  |  |  |  |  |

Продовження таблиці 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 |  |  |  |  |  |  |  | x | x |  |  |  |  |
| 7 |  |  |  |  |  |  | x | x |  |  |  |
| 8 |  |  |  |  |  |  |  | x | x |  |  |
| 9 |  |  |  |  |  |  |  |  | x | x |  |
| 10 |  |  |  |  |  |  |  |  |  | x | x |
| 11 |  |  |  |  |  |  |  |  |  |  | x |

Таблиця 3 Результат тестування

|  |  |  |
| --- | --- | --- |
| Відмітка | Види тестів | Примітка |
|  | Створення матриць A, B, C; | Тест пройдено успішно |
|  | Тестування "+" | Тест пройдено успішно |
|  | Тестування "-" | Тест пройдено успішно |
|  | Тестування "\*" | Тест пройдено успішно |
|  | Тестування "Згенерувати" | Тест пройдено успішно |
|  | Тестування "Заповнити рандомно" | Тест пройдено успішно |
|  | Тестування множення на число | Тест пройдено успішно |
|  | Тестування возведення до степеню | Тест пройдено успішно |
|  | Тестування транспонування | Тест пройдено успішно |
|  | Тестування визначників | Тест пройдено успішно |
|  | Тестування оберненої матриці | Тест пройдено успішно |

# **ВИСНОВОК**

Як висновок, курсова робота допомогла узагальнити, закріпити та розширити практичні і теоретичні знання, уміння і навички програмування мовою С++. Вона дала змогу укріпити вміння в об’єктно-орієнтованому програмуванні.

У даній курсовій роботі досягнуто поставленої мети – розроблено програму для збільшення показника продуктивності працівників, чия робота пов'язані з обчисленнями різноманітних величин розмірністю матриць.

У курсовій роботі було розглянуто функції додавання, віднімання, множення та множення матриць на число, транспонування, обернена матриця, знаходження визначника, зведення матриці до степеню, а також перенесення матриці із однієї в іншу. Крім цього, було проведено аналіз предметної області, а саме: аналіз операцій над матрицями, аналіз завдань розв’язуваних з урахуванням їх представлення у матричному вигляді, особливості обробки даних при роботі з матрицями. Здійснено огляд аналогів та засобів розробки, а саме: аналіз існуючих систем, обґрунтування вибору засобів розробки.

Під час роботи над проєктом я покращив уміння пошуку необхідної інформації, здобув досвід в дослідженні й експериментуванні під час розробки програми. Закріпив знання об’єктно-орієнтованого програмування та навичок програмування мовою С++.

Отже, розроблення програмного проєкту підтвердило знання, уміння і навички, здобуті протягом всього навчального року.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Теоретичний матеріал з математики. URL: https://ua.onlinemschool.com/math/library/ (дата звернення 07.04.2023).
2. Основні властивості визначників матриць. URL: <https://yukhym.com/uk/matritsi-ta-viznachniki/osnovni-vlastyvosti-vyznachnykiv.html> (дата звернення 07.04.2023).
3. Транспонована матриця. URL: <https://uk.economy-pedia.com/11030817-transposed-matrix>. (дата звернення 07.04.2023).
4. Комп'ютерна техніка та інформаційні технології. Електронний підручник. URL: <https://www.shevchenkove.org.ua/person_syte/Golub/КТ%20і%20ІТ/лаб21.html> (дата звернення 07.04.2023).
5. Власні значення та власні вектори матриці. URL: <https://ukrayinska.libretexts.org/Математика/Лінійна_алгебра/Перший_курс_з_лінійної_алгебри_(Kuttler)/07%3A_Спектральна_теорія/7.01%3A_Власні_значення_та_власні_вектори_матриці>. (дата звернення 07.04.2023).
6. Сборник задач по математике для вузов: в 3 ч./ В.А. Болгов, А.В. Ефимов, А.Ф. Каракулин и др. – М.:иНаука, 1986. – Линейная алгебра и основы математического анализа.
7. "Introduction to Matrices" від Math is Fun. URL: <https://www.mathsisfun.com/algebra/matrix-introduction.html> (дата звернення 07.04.2023).
8. "Matrix Computation" від MathWorks. URL: <https://www.mathworks.com/discovery/matrix-computation.html> (дата звернення 07.04.2023).
9. "Matrix Operations" від Wolfram MathWorld. URL: <https://mathworld.wolfram.com/MatrixOperations.html> (дата звернення 07.04.2023).
10. Методические указания по высшей математике Макушкина Марина Павловна (дата звернення 07.04.2023).
11. Создавайте код быстрее. Работайте эффективнее. URL: <https://visualstudio.microsoft.com/ru/vs/> (дата звернення 28.04.2023).
12. Справочник по языку C++. URL: <https://learn.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170> (дата звернення 28.04.2023).
13. Файлы заголовков стандартной библиотеки С и C++. URL: https://docs.microsoft.com/ru-ru/cpp/standard-library/cpp-standard-library-header-files?view=msvc-170 (дата звернення 28.04.2023).
14. Документація Microsoft про директиву #pragma once. URL: <https://docs.microsoft.com/en-us/cpp/preprocessor/once?view=msvc-160> (дата звернення 28.04.2023).
15. Документація Microsoft про директиву #pragma warning. URL: <https://docs.microsoft.com/en-us/cpp/preprocessor/warning?view=msvc-160> (дата звернення 28.04.2023).
16. System Namespace. URL: <https://docs.microsoft.com/en-us/dotnet/api/system?view=net-5.0> (дата звернення 28.04.2023).
17. Макушкина, М.П. Методичні вказівки / М. П. Макушкина. – Одеса: 2013 (дата звернення 07.04.2023 – 07.05.2023).

# **ДОДАТОК А. ЛІСТИНГ КОДУ**

**Func.h**

#pragma once

#pragma warning(disable : 4244)

#include <iostream>

#include <ctime>

#include <string>

#include <cmath>

#include <msclr\marshal\_cppstd.h>

double\*\* AddMatrix(double\*\* A, double\*\* B, int N, int M);

double\*\* MinusMatrix(double\*\* A, double\*\* B, int N, int M);

double\*\* UmnMatrix(double\*\* A, double\*\* B, int N\_A, int M\_A, int N\_B, int N\_C);

double\*\* UmnMatrixChislo(double\*\* A, int N, int M,int num);

double\*\* Transp(double\*\* A, int N, int M);

double\*\* pow(double\*\* A, int N, int M,int num);

double\*\* Invers(double\*\* A, int N);

double det(double\*\* Matrix, int N);

double Rounding(double Num);

double Rounding1(double Num);

**MyForm.h**

#pragma once

private: System::Void MoveA\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void MoveB\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void MoveA1\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void MoveB1\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void GenA\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void GenB\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void ZapA\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void ZapB\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void plus\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void minus\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void umn\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void NumButA\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void StepButA\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void NumButB\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void StepButB\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void TransA\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void TransB\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void detA\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void detB\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void detC\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void InversA\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void InversB\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void dataMatrixA\_CellEndEdit(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e);

private: System::Void dataMatrixB\_CellEndEdit(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e);

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e);

private: void ShowA();

private: void ShowB();

private: void ShowC(int N\_A, int M\_A);

private: void ShowCUMN();

private: void ShowCTrans(int N, int M);};}