МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

інформатики та програмної інженерії

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з дисципліни «Основи програмування - 2. Методології програмування»

(назва дисципліни)

на тему: Обернення матриці методами Жордана-Гауса та LUP-розкладу

Студента 1-го курсу, групи ІП-22

Нижника Дмитра Сергійовича

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник

ст.вик. Головченко М.М.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Члени комісії |  |  | |  | | --- | | к.т.н. доц. Муха І.П. | |
|  | (підпис) |  | (посада, вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  | |  | | --- | | ас. Вовк Є.А. | |
|  | (підпис) |  | (посада, вчене звання, науковий ступінь, прізвище та ініціали) |

Київ – 2023 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрям "ІПЗ"

Курс 1 Група ІП-22 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Нижника Дмитра Сергійовича

(прізвище, ім’я, по батькові)

1. Тема роботи обернення матриці методами Жордана-Гауса та LUP-розкладу

2. Строк здачі студентом закінченої роботи 31.05.2023

3. Вихідні дані до роботи Додаток “А” Технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

Постановка задачі, теоретичні відомості, опис алгоритмів, опис програмного забезпечення, тестування програмного забезпечення, інструкція користувача, аналіз і узагальнення результатів

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )

6. Дата видачі завдання 12.02.2023

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 12.02.2023 |  |
| 2. | Підготовка ТЗ | 20.02.2023 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 23.03.2023 |  |
| 4. | Розробка сценарію роботи програми | 08.04.2023 |  |
| 6. | Узгодження сценарію роботи програми з керівником | 11.04.2023 |  |
| 5. | Розробка (вибір) алгоритму рішення задачі | 15.04.2023 |  |
| 6. | Узгодження алгоритму з керівником | 25.04.2023 |  |
| 7. | Узгодження з керівником інтерфейсу користувача | 25.04.2023 |  |
| 8. | Розробка програмного забезпечення | 29.04.2023 |  |
| 9. | Налагодження розрахункової частини програми | 29.04.2023 |  |
| 10. | Розробка та налагодження інтерфейсної частини програми | 29.04.2023 |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу | 09.05.2023 |  |
| 12. | Тестування програми | 13.05.2023 |  |
| 13. | Підготовка пояснювальної записки | 19.05.2023 |  |
| 14. | Здача курсової роботи на перевірку | 31.05.2023 |  |
| 15. | Захист курсової роботи | 06.06.2023 |  |

Студент

(підпис)

Керівник Головченко М.М.

(підпис) (прізвище, ім’я, по батькові)

"12" лютого 2023 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 81 сторінка, 17 рисунків, 13 таблиць, 6 посилань.

Мета роботи: розробка якісного та надійного ПЗ для виконання обернення матриці за допомогою методів Жордана-Гауса та LUP-розкладу.

В ході виконання роботи вивчено методи Жордана-Гауса та LUP-декомпозиції для обернення матриці.

Виконана програмна реалізація алгоритмів обернення матриці Жордана-Гауса та LUP-декомпозиції. Розроблено застосунок з графічним інтерфейсом для обернення матриці одним із заданих двох методів. Роботу застосунку протестовано на різних значеннях та у різних умовах.

КВАДРАТНА МАТРИЦЯ, АЛГОРИТМ ЖОРДАНА-ГАУСА, АЛГОРИТМ LUP-ДЕКОМПОЗИЦІЇ, АЛГОРИТМ LUP-РОЗКЛАДУ, ВИЗНАЧНИК МАТРИЦІ, ОБЕРНЕНА МАТРИЦЯ, НЕВИРОДЖЕНА МАТРИЦЯ, ОБЕРНЕННЯ МАТРИЦІ.

Зміст

[Вступ 6](#_Toc136719098)

[1 Постановка задачі 7](#_Toc136719099)

[2 Теоретичні відомості 8](#_Toc136719100)

[2.1. Метод Жордана-Гауса: 8](#_Toc136719101)

[2.2. Метод LUP-декомпозиції: 9](#_Toc136719102)

[3 Опис алгоритмів 11](#_Toc136719103)

[3.1.Загальний алгоритм 12](#_Toc136719104)

[3.2. Алгоритм знаходження визначника методом Гауса 13](#_Toc136719105)

[3.3. Алгоритм LUP-декомпозиції 13](#_Toc136719106)

[3.4. Алгоритм обернення матриці Жордана-Гауса 15](#_Toc136719107)

[4 Опис програмного забезпечення 16](#_Toc136719108)

[4.1. Діаграма класів програмного забезпечення 16](#_Toc136719109)

[4.2. Опис методів частин програмного забезпечення 17](#_Toc136719110)

[4.2.1. Стандартні методи 17](#_Toc136719111)

[4.2.2. Користувацькі методи 27](#_Toc136719112)

[5 Тестування програмного забезпечення 37](#_Toc136719113)

[5.1. План тестування 37](#_Toc136719114)

[5.2. Приклади тестування 38](#_Toc136719115)

[6 Інструкція користувача 42](#_Toc136719116)

[6.1. Робота з програмою 42](#_Toc136719117)

[6.2. Формат вхідних та вихідних даних 48](#_Toc136719118)

[6.3. Системні вимоги 48](#_Toc136719119)

[7 Аналіз і узагальнення результатів 49](#_Toc136719120)

[Висновки 53](#_Toc136719121)

[Перелік посилань 54](#_Toc136719122)

[Додаток А Технічне завдання 55](#_Toc136719123)

[Додаток Б Тексти програмного коду 58](#_Toc136719124)

Вступ

Сьогодні математика займає особливе становище серед усіх точних наук. Затребуваність математики проявляється у наукових дослідженнях, інженерії, фінансах, комп'ютерних науках, штучному інтелекті та багатьох інших сферах. Математика залишається фундаментальною наукою, яка робить істотний внесок у розвиток сучасного світу і надає нам потужний інструментарій для розуміння та управління складними системами та процесами.

Обернення матриць є фундаментальною операцією в лінійній алгебрі і знаходить застосування у багатьох галузях, включаючи фізику, інженерію, економіку, комп'ютерні науки та багато інших. Цей процес дозволяє з вирішувати найрізноманітніші завдання, пов'язані із моделюванням, оптимізацією та обробкою даних, роботою з комп’ютерною графікою, вирішенням систем лінійних алгебраїчних рівнянь, криптографією тощо. Обернення матриць є актуальною та невід'ємною частиною багатьох наукових та інженерних завдань.

Ефективними алгоритмами обернення матриць є алгоритм Жордана-Гауса та метод LUP-розкладу. Алгоритм Жордана-Гауса є класичним та ефективним способом обернення матриць, що дозволяє отримувати обернені матриці шляхом елементарних перетворень над вихідною матрицею. Він широко використовується в наукових та інженерних розрахунках.

LUP-розклад, у свою чергу, дозволяє розкласти вихідну матрицю на добуток трьох матриць: нижньотрикутної, верхньотрикутної та матриці перестановок. Цей метод має переваги, такі як більш ефективні обчислення та більшу стійкість до помилок.

На завершення, обернення матриць методом Жордана-Гауса та LUP-розкладом є актуальною та затребуваною темою у різноманітних науках. Ці алгоритми забезпечують ефективність та точність у вирішенні різних завдань. Зі зростанням значущості матричних операцій у світі, обернення матриць стало невід'ємною частиною наукових та інженерних досліджень.

# Постановка задачі

Розробити програмне забезпечення, що буде знаходити обернену матрицю наступними методами:

а) метод Жодана-Гауса;

б) метод LUP-розкладу;

Вхідними даними для даної роботи є розмірність матриці та значення елементів матриці. Програма має працювати з квадратної матрицею, розмірність якої знаходиться в межах від 2 до 10 включно.

Вихідними даними для даної роботи є матриця, така, що:

*,*

де A – вихідна матриця, B – результуюча обернена матриця, а E – одинична матриця, тобто така, у якої всі елементі головної діагоналі = 1, а всі інші = 0. Програмне забезпечення повинно видавати розв’язок тільки за умови, що для матрицю можна обернути, тобто матриця не містить нульових елементів на головній діагоналі і її визначник не дорівнює 0. Якщо це не так, то програма повинна вивести повідомлення помилки. Окрім цього, програма повинна підраховувати кількість кроків, витрачених на вирішення задачі.

# Теоретичні відомості

Обернену матрицю можна представити наступним чином:

|  |
| --- |
| (2.1) |

де:

, ,

Тоді якщо , то обернена матриця B (2.1) існує та вона єдина. Якщо матриця оборотна, то обернену матрицю можна знайти одним із наступних методів.

## Метод Жордана-Гауса:

Сутність методу Жордана-Гауса полягає в наступному: вихідну матрицю треба доповнити одиничною матрицею:

,

далі необхідно звести вихідну матрицю до східчастого вигляду за такими формулами, виконуючи ті самі дії з доповненням матриці:

*,*

*,*

(за умови, що )

Таким чином ми отримали наступну матрицю:

Далі необхідно провести зворотній хід Гауса за такими формулами:

(за умови, що )

Результатом цього алгоритму отримано таку матрицю:

Причому доповнення матриці – це і є обернена матриця.

Алгоритмічна складність методу Жордана-Гауса O(n3).

## Метод LUP-декомпозиції:

Сутність цього методу полягає в декомпозиції матриці. Результатом LUP-розкладу є рівність P\*A=L\*U. Таким чином, B = L-1 \* U-1 \* P.

Алгоритм складається з декількох етапів. Перший – ініціалізація. На цьому кроці створюються матриці L, U і масив перестановок P. В матрицю U копіюються елементи матриці A, L і U залишаються порожніми.

Наступним етапом є отримання матриць L та U, а також масиву перестановок P. На кожному кроці ітерації знаходиться опорний елемент (pivot) – це найбільший за значенням в стовпці елемент, рядок опорного елемента стає опорним рядком (pivotRow). Якщо опорний елемент не знаходиться на діагоналі поточного стовпця, змінюємо рядки у матриці U та відповідні елементи у масиві P для забезпечення знаходження провідного елемента на діагоналі. Робимо те саме для матриці L, щоб зберегти відповідну перестановку. Після цього застосовуємо операцію елімінації Гауса до поточного стовпця матриці U, щоб звести елементи нижче провідного елемента до 0. Результат цієї операції записується у матрицю L. Таку послідовність дій повторюємо для кожного стовпця матриці A.

Після завершення таких ітерацій ми отримуємо матрицю U, яка є верхньотрикутною матрицею, матрицю L, яка є нижньотрикутною матрицею, та масив перестановок P. Тепер ми можемо використовувати ці матриці для обернення матриці.

Щоб обернути матрицю A, ми вирішуємо СЛАР P\*L\*U\*X=I, де I – одинична матриця. Ми починаємо з розв'язання системи L\*Y=P^-1\*I методом прямого ходу для нижньотрикутних матриць, а потім вирішуємо систему U\*X=Y методом зворотного ходу для верхньотрикутних матриць. Отримане рішення X і буде оберненою матрицею B.

# Опис алгоритмів

Перелік основних змінних та їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| dimension | Розмірність матриці |
| method | Номер методу обернення матриці |
| matrix | Початкова квадратна матриця |
| determinant | Визначник матриці |
| markerInv | Маркер перевірки результату обернення. Якщо отримано NaN (внаслідок, наприклад, ділення на 0 або інших неправильних операцій), то маркер = false |
| tempMatrix | Тимчасова матриця |
| inversedmatrix | Обернена матриця |
| factor | Коефіцієнт для перетворення матриці |
| L | Нижньотрикутна матриця |
| U | Верхньотрикутна матриця |
| P | Масив перестановок |
| pivotRow | Опорний ряд |
| pivot | Опорний елемент |
| e | Вектор з одиничним елементом |
| x | Вектор рішень |
| y | Вектор проміжних даних |
| i | Лічильник |
| j | Лічильник |
| k | Лічильник |

## 3.1.Загальний алгоритм

1. ПОЧАТОК
2. Зчитати розмірність системи *dimension*.
3. Зчитати номер обраного методу *method*.
4. Створити квадратну матрицю *matrix* розмірності *dimension*
5. Зчитати матрицю системи з інтерфейсу:
   1. Цикл проходу по всіх рядках матриці інтерфейсу від 1 до *dimension* (ai – поточна строка):
      1. Цикл проходу по всіх стовпцях матриці інтерфейсу від 1 до *dimension* (aij – поточний елемент):
         1. ЯКЩО поточний елемент матриці – правильно записане число, ТО записати його в відповідну комірку *matrix*. ІНАКШЕ записати в комірку елемент NaN.
6. Знайти визначник *determinant* методом Гауса (підрозділ 3.2)
7. ЯКЩО *determinant* 0 і *method =* 1, ТО отримати обернену матрицю *invmatrix* згідно алгоритму методу LUP-декомпозиції (підрозділ 3.3). ІНАКШЕ вивести повідомлення помилки.
8. ЯКЩО *determinant* 0 і *method =* 2, ТО отримати обернену матрицю *invmatrix* згідно алгоритму методу Жордана-Гауса (підрозділ 3.4) ІНАКШЕ вивести повідомлення помилки.
9. Перевірити результат обернення:
   1. *markerInv* := true
   2. Цикл проходу по всіх рядках матриці інтерфейсу від 1 до *dimension* (ai – поточна строка):
      1. Цикл проходу по всіх стовпцях матриці інтерфейсу від 1 до *dimension* (aij – поточний елемент):
         1. ЯКЩО поточний елемент матриці = NaN (Not a Number), ТО *markerInv* := false
10. ЯКЩО *markerInv* = true, ТО вивести обернену матрицю *inversedmatrix* на головне вікно. ІНАКШЕ вивести повідомлення помилки.
11. КІНЕЦЬ

## 3.2. Алгоритм знаходження визначника методом Гауса

1. ПОЧАТОК
2. Створити копію матриці matrix - матрицю *tempMatrix* розмірності *dimension*
3. Процес виключення Гауса:
   1. Цикл *i* від 1 до *dimension-1*:
      1. Цикл *j* від 1 до *dimension*:
         1. *factor* := *tempMatrixji* / *tempMatrixii*
         2. Цикл *k* від 1 до dimension :
            1. *tempMatrixjk* = *tempMatrixjk*– *factor* \* *tempMatrixii*
4. Цикл *i* від 1 до *dimension-1*:
   1. *determinant* = *determinant* \* *tempMatrixii*
5. КІНЕЦЬ

## 3.3. Алгоритм LUP-декомпозиції

1. ПОЧАТОК
2. Створити матрицю *L* розмірності *dimension*
3. Створити матрицю *U* розмірності *dimension*
4. Створити матрицю *invmatrix* розмірності *dimension*
5. Створити масив *P* розмірності *dimension*
6. Заповнити матрицю *L* 0
7. Заповнити матрицю *invmatrix* 0
8. Скопіювати матрицю *matrix* у матрицю *U*
9. Цикл *і* від 1 до *dimension*:

9.1. *Pi* := *i* - 1

1. Цикл *k* від 1 до *dimension:*

10.1. *pivotRow* := *k*

10.2. *pivot* :=

10.3. Цикл *i* від *k* до *dimension*:

10.3.1. ЯКЩО :

10.3.1.1. *pivotRow* := i;

10.3.1.2. *pivot* :=

10.4. Поміняти місцями *Pk* і *PpivotRow*

10.5. Поміняти місцями *Uk* і *UpivotRow*

10.6. Поміняти місцями *Lk* і *LpivotRow*

10.7. Цикл *i* від *k+1* до *dimension*:

10.7.1. *Lik* := *Uik*/*Ukk*

10.7.2. Цикл *j* від *k*+1 до *dimension*:

10.7.2.1. *Uij := Uij - Lik\* Uki*

1. Цикл *i* від 0 до *dimension:*
   1. Створити масив *e* розмірності *dimension* та заповнити його 0
   2. ei = 0
   3. Створити масив *x* розмірності *dimension*
   4. Створити масив *y* розмірності *dimension*
   5. Цикл *j* від 0 до *dimension*:

11.5.1. *yj* :=

11.5.2. Цикл *k* від 0 до *dimension*:

11.5.2.1. *yj* := *yj – Ljk\*yk*

* 1. Цикл *j* від *dimension*-1 до 0:

11.6.1. *xj := yj*

11.6.2. Цикл *k* від *j+1* до *dimension*:

11.6.2.1. *xj := xj – Ujk\*xk*

11.6.3. *xj := xj / Ujj*

* 1. Цикл *j* від 0 до *dimension*:

11.7.1. *invmatrixji* := *xj*

12. КІНЕЦЬ

## 3.4. Алгоритм обернення матриці Жордана-Гауса

1. ПОЧАТОК
2. Створити матрицю *invmatrix* розмірності *dimension* і заповнити діагональні елементи 1, а недіагональні – 0
3. Процес виключення Гауса:
   1. Цикл *i* від 1 до *dimension-1*:
      1. Цикл *j* від 1 до *dimension*:
         1. *factor* := *invmatrix ji* / *invmatrix ii*
         2. Цикл *k* від 1 до *dimension*:
            1. *invmatrix jk* = *invmatrix jk*– *factor* \* *invmatrix ii*
            2. *matrix jk* = *matrix jk*– *factor* \* *matrix ii*
4. Цикл *i* від 1 до *dimension*:
   1. Цикл i від 1 до *dimension*:
      1. *invmatrix ij* = *invmatrix ij*– *matrixii*
5. КІНЕЦЬ

# Опис програмного забезпечення

## Діаграма класів програмного забезпечення

У програмі представлено 4 класи: Matrices, MainWindow, StartWindow, SquareMatrix (Рисунок 4.1), кожний з яких виконує свою власну функцію.

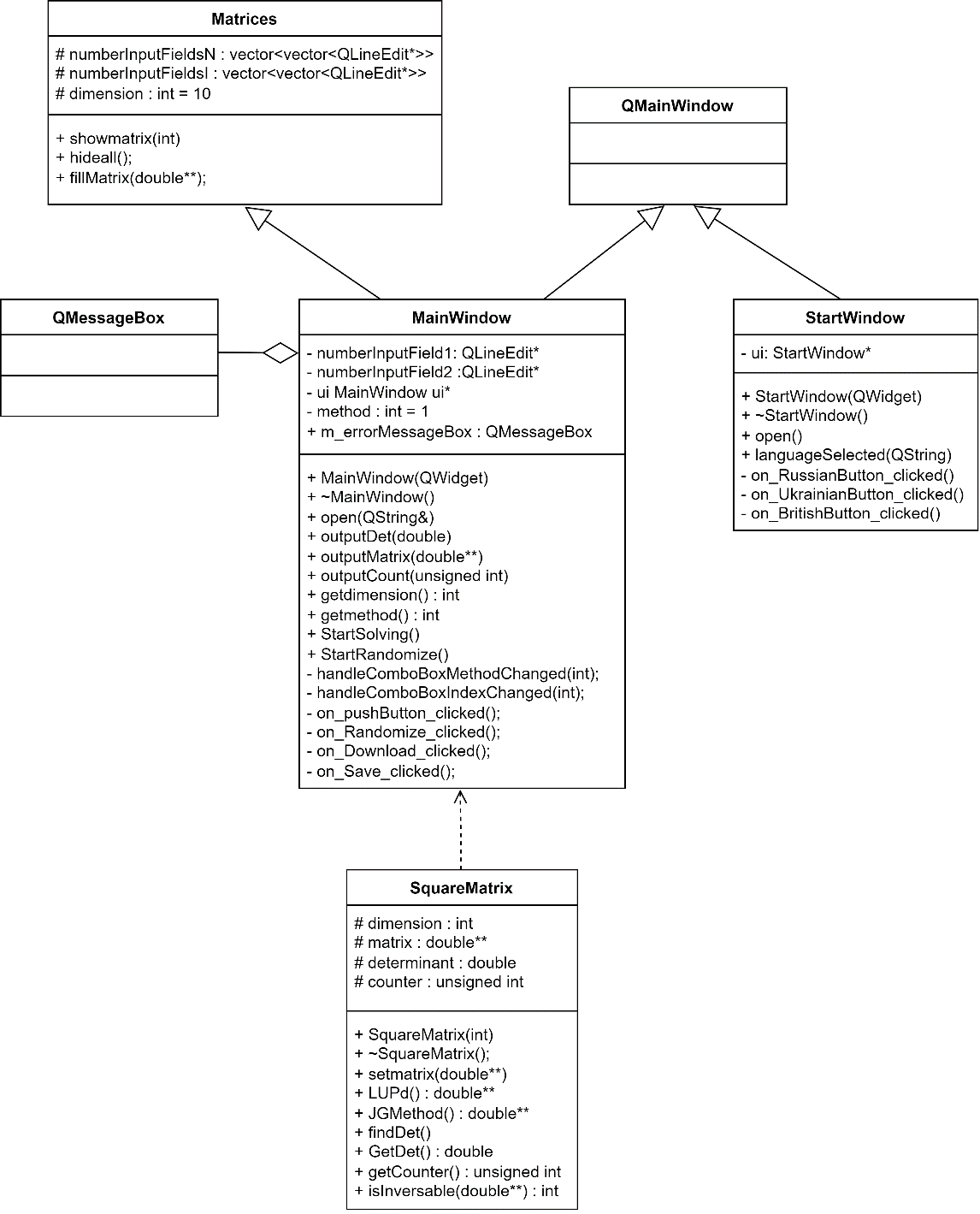


Рисунок 4.1 – Діаграма класів

Matrices - це клас, що відповідає за матриці графічного інтерфейсу. Він зв’язує поля для введення та управляє ними.

Клас MainWindow – це клас головного вікна, що є нащадком класу Matrices та класу QMainWindow зі стандартної бібліотеки QMainWindow та пов’язаний зв’язком агрегації із класом QMessageBox. Він управляє відображенням цього вікна і відповідає за взаємодію користувача із основними елементами програми.

Клас StartWindow – це клас початкового вікна вибору мови, є нащадком класу QMainWindow зі стандартної бібліотеки QMainWindow. Він реалізує можливість вибору мови інтерфейсу користувачем, а також відповідає за взаємодію користувача і графічного інтерфейсу вікна.

Клас SquareMatrix – це клас робочої квадратної матриці, є дружнім класом для MainWindow. Цей клас реалізує основні алгоритми матриці: обернення, пошук визначника, перевірка на символи тощо.

## Опис методів частин програмного забезпечення

### Стандартні методи

У таблиці 4.1 наведено використані у програмі стандартні методи.

Таблиця 4.1 – Стандартніметоди

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | QCoreApplication | applicationDirPath | Повертає шлях до каталогу, в якому знаходиться файл програми, що виконується | Не має вхідних параметрів | Шлях до каталогу типу QString | QCoreApplication |
| 2 | QApplication | exec | Запускає головний цикл подій Qt | Не має вхідних параметрів | Повертає код виходу із застосунку типу int | QApplication |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 3 | QWidget | setGeometry | Встановлює положення та розміри віджету на екрані | Приймає значення x та y типу int – координати, а також width та height – відповідно ширину та висоту віджету | Не має вихідних параметрів | QWidget |
| 4 | QComboBox | addItem | Використовується для додавання нового елемента у QComboBox | Приймає аргумент типу QString – строка, яку буде додано в список | Не має вихідних параметрів | QComboBox |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 5 | QMetaObject | connect | Використовується для встановлення зв'язку між сигналом та слотом | Приймає покажчик на відправника сигналу типу QObject\*, сигнал типу const char\*, покажчик на об'єкт-одержувач слота типу QObject\*, слот типу const char\* | Не має вихідних параметрів | QMetaObject |
| 6 | QLineEdit | setReadOnly | Використовується для встановлення текстового поля у стан "тільки для читання" | Приймає один аргумент типу bool, який визначає, чи буде текстове поле тільки для читання | Не має вихідних параметрів | QLineEdit |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 7 | QLineEdit | setAlignment | Використовується для встановлення вирівнювання тексту всередині віджету | Приймає один аргумент – прапор вирівнювання тексту | Не має вихідних параметрів | QLineEdit |
| 8 | QTextStream | readLine | Використовується для читання рядка з текстового потоку | Не має вхідних параметрів | Повертає рядок, прочитаний типу з текстового потоку QString | QTextStream |
| 9 | QIODevice | close() | Припиняє операції читання або запису та звільняє пов'язані ресурси | Не має вхідних параметрів | Не має вихідних параметрів | QIODevice |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 10 | QIODevice | open() | Використовується для відкриття файлу | Не має вхідних параметрів | Не має вихідних параметрів | QIODevice |
| 11 | QWidget | setStyleSheet | Використовується для застосування стилів на віджет за допомогою CSS | Приймає рядок типу QString, що містить таблицю стилів CSS | Не має вихідних параметрів | QWidget |
| 12 | QWidget | setWindowTitle | Використовується для встановлення тексту, який буде відображатися в заголовку вікна віджету | Приймає рядок типу QString, що містить потрібний заголовок | Не має вихідних параметрів | QWidget |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 13 | QWidget | resize | Використовується для зміни ширини та висоти віджету на вказані значення | Приймає ширину int w та висоту int h, що представляють нові розміри віджету. | Не має вихідних параметрів | QWidget |
| 14 | QWidget | setWindowIcon | Використовується для встановлення значка, який відображатиметься у заголовку вікна віджету | Праймає іконку типу QIcon | Не має вихідних параметрів | QWidget |
| 15 | QAbstractButton | setIcon | Використовується для встановлення іконки на кнопку або інший віджет | Праймає іконку типу QIcon | Не має вихідних параметрів | QAbstractButton |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 16 | QObject | findChild | Використовується для пошуку дочірнього об'єкта, який є нащадком даного об'єкта | Приймає ім'я дочірнього об'єкта типу QSting, опціонально тип дочірнього об'єкта типу QObject\* | Повертає покажчик на знайдений дочірній об'єкт типу QObject\* або нульовий покажчик | QObject |
| 17 | QLabel | setText | Використовується для встановлення тексту, який відображатиметься на віджеті | Приймає текстовий рядок типу QString | Не має вихідних параметрів | QLabel |
| 18 | QWidget | show | Використовується для відображення віджету на екрані, роблячи його видимим для користувача | Не має вхідних параметрів | Не має вихідних параметрів | QWidget |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 19 | QString | number | Використовується для перетворення числового значення в рядок | Приймає один аргумент – число, яке необхідно перетворити у строку | Повертає рядок - об’єкт класу QString | QString |
| 20 | QComboBox | currentText | Використовується для отримання поточного текстового значення вибраного елемента QComboBox | Не має вхідних параметрів | Повертає рядок - об’єкт класу QString | QComboBox |
| 21 | QObject | qobject\_cast | Використовується для безпечного приведення типів об'єктів, що успадковують від класу QObject | Приймає покажчик на об’єкт та тип, до якого потрібно привести | Повертає покажчик на об'єкт зазначеного типу, але якщо приведення типів не успішне, то повертається нульовий покажчик (nullptr) | QObject |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 22 | QString | number | Використовується для перетворення числового значення в рядок | Приймає один аргумент – число, яке необхідно перетворити у строку | Повертає рядок - об’єкт класу QString | QString |
| 23 | string | stoi | Використовується для перетворення рядка на ціле значення | Приймає рядок типу string, який потрібно перетворити на ціле число | Повертає ціле значення, отримане з рядка | string |
| 24 | QString | toStdString | Використовується для перетворення об'єкта QString на рядок string | Не має вхідних параметрів | Повертає об'єкт string, що містить еквівалентне рядкове представлення QString | QString |
| 25 | QString | toDouble | Використовується для перетворення об'єкта QString на число типу double | Не має вхідних параметрів | Повертає значення типу double, що еквівалентне числу із об’єкта QString | QString |

Продовження таблиці 4.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 26 | QLabel | text | Використовується для отримання текстового вмісту, що відображається в об’єкті типу QLabel | Не має вхідних параметрів | Повертає об'єкт класу QString, що містить текстовий вміст мітки | QLabelset |
| 27 | QWidget | setVisible | Використовується для встановлення видимості віджету | Приймає аргумент типу bool, що вказує на видимість віджету | Не має вихідних параметрів | QWidget |
| 28 | QLabel | setPixmap | Використовується для встановлення зображення у мітку QLabel | Приймає аргумент типу QPixmap, що представляє зображення, яке потрібно встановити в мітку. | Не має вихідних параметрів | QLabel |

### Користувацькі методи

У таблиці 4.2 наведено всі користувацькі методі програми.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | Matrices | showmatrix | Відобразити поля для вводу, що представляють собою матрицю заданого розміру | int amount - це кількість полів для введення та розмірність матриці | Не має вихідних параметрів | mainwindow.h |
| 2 | Matrices | hideall | Приховати всі поля для введення з ціллю подальшого відображення потрібної кількості полів для введення | Не має вхідних параметрів | Не має вихідних параметрів | mainwindow.h |

– Користувацькі методи

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 3 | Matrices | fillMatrix | Зчитати дані з полів для введення, записати їх в матрицю | double\*\* matrix – покажчик на матрицю, в яку будуть записані зчитані дані | Не має вихідних параметрів | mainwindow.h |
| 4 | MainWindow | MainWindow | Конструктор об’єкта головного вікна | QWidget \*parent –покажчик на батьківський віджет. За замовченням вказаний нульовий покажчик (nullptr) | Не має вихідних параметрів | mainwindow.h |
| 5 | MainWindow | ~MainWindow | Деструктор об’єкта головного вікна | Не має вхідних параметрів | Не має вихідних параметрів | mainwindow.h |

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 6 | MainWindow | open | Метод відкриття головного вікна з інтерфейсом на обраній користувачем мові | QString& LanguagePath – посилання на строку з шляхом до файлу тексту інтерфейсу | Не має вихідних параметрів | mainwindow.h |
| 7 | MainWindow | outputDet | Вивести значення визначника на головний екран | double det – значення визначника матриці | Не має вихідних параметрів | mainwindow.h |
| 8 | MainWindow | outputMatrix | Вивести обернену матрицю на головний екран | double\*\* matrix – обернена матриця, яку потрібно вивести | Не має вихідних параметрів | mainwindow.h |
| 9 | MainWindow | outputCount | Вивести кількість кроків алгоритму на головний екран | unsigned int steps – кількість кроків, яку потрібно вивести | Не має вихідних параметрів | mainwindow.h |

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 10 | MainWindow | getdimension | Повернути розмірність матриці, яку обрав користувач | Не має вхідних параметрів | Повертає значення типу int – розмірність матриці | mainwindow.h |
| 11 | MainWindow | getmethod | Повернути номер обраного методу розв’язання завдання | Не має вхідних параметрів | Повертає значення типу int – номер обраного методу | mainwindow.h |
| 12 | MainWindow | handleComboBoxMethodChanged | Відреагувати на зміну обраного методу в списку | int method – номер обраного методу | Не має вихідних параметрів | mainwindow.h |
| 13 | MainWindow | handleComboBoxIndexChanged | Відреагувати на зміну обраної розмірності матриці в списку | int index – обрана розмірність матриці | Не має вихідних параметрів | mainwindow.h |

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 14 | MainWindow | on\_pushButton\_clicked | Відреагувати на натискання кнопки початку роботи алгоритму | Не має вхідних параметрів | Не має вихідних параметрів | mainwindow.h |
| 15 | MainWindow | on\_Randomize\_clicked | Відреагувати на натискання кнопки заповнення матриці випадковим чином | Не має вхідних параметрів | Не має вихідних параметрів | mainwindow.h |
| 16 | MainWindow | on\_Save\_clicked | Відреагувати на натискання кнопки збереження матриці | Не має вхідних параметрів | Не має вихідних параметрів | mainwindow.h |

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 17 | MainWindow | StartSolving | Сигнал початку роботи програми після натискання кнопки початку розв’язку | Не має вхідних параметрів | Не має вихідних параметрів | mainwindow.h |
| 18 | MainWindow | StartRandomize | Сигнал початку заповнення матриці випадковим чином після натискання кнопки заповнення випадковим чином | Не має вхідних параметрів | Не має вихідних параметрів | mainwindow.h |

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 19 | StartWindow | StartWindow | Конструктор об’єкта стартового вікна | QWidget \*parent –покажчик на батьківський віджет. За замовченням вказаний нульовий покажчик (nullptr) | Не має вихідних параметрів | startwindow.h |
| 20 | StartWindow | ~StartWindow | Деструктор об’єкта стартового вікна | Не має вхідних параметрів | Не має вихідних параметрів | startwindow.h |
| 21 | StartWindow | on\_RussianButton\_clicked | Відреагувати на вибір користувачем російської мови | Не має вхідних параметрів | Не має вихідних параметрів | startwindow.h |
| 22 | StartWindow | on\_UkrainianButton\_clicked | Відреагувати на вибір користувачем української мови | Не має вхідних параметрів | Не має вихідних параметрів | startwindow.h |

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 23 | StartWindow | on\_BritishButton\_clicked | Відреагувати на вибір користувачем англійської мови | Не має вхідних параметрів | Не має вихідних параметрів | startwindow.h |
| 24 | StartWindow | open | Відобразити стартове вікно вибору мови для взаємодії користувача з ним | Не має вхідних параметрів | Не має вихідних параметрів | startwindow.h |
| 25 | SquareMatrix | SquareMatrix | Конструктор об’єкта квадратної матриці | int d – розмірність матриці | Не має вихідних параметрів | squarematrix.h |
| 26 | SquareMatrix | ~SquareMatrix | Деструктор об’єкта квадратної матриці | Не має вхідних параметрів | Не має вихідних параметрів | squarematrix.h |
| 27 | SquareMatrix | setmatrix | Модифікатор матриці matrix | double\*\* temp – матриця, яку треба присвоїти атрибуту matrix | Не має вихідних параметрів | squarematrix.h |

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 28 | SquareMatrix | LUPd | Метод, що реалізує алгоритм LUP-декомпозиції | Не має вхідних параметрів | Обернена матриця – результат типу double\*\* | squarematrix.h |
| 29 | SquareMatrix | LUPd | Метод, що реалізує алгоритм LUP-декомпозиції | Не має вхідних параметрів | Обернена матриця – результат типу double\*\* | squarematrix.h |
| 30 | SquareMatrix | JGMethod | Метод, що реалізує алгоритм Жордана-Гауса | Не має вхідних параметрів | Обернена матриця типу double\*\* | squarematrix.h |
| 31 | SquareMatrix | findDet | Метод, що реалізує пошук визначника матриці | Не має вхідних параметрів | Не має вихідних параметрів | squarematrix.h |
| 32 | SquareMatrix | getDet | Селектор визначника матриці determinant | Не має вхідних параметрів | Повертає значення determinant типу double | squarematrix.h |

Продовження таблиці 4.2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 33 | SquareMatrix | getCounter | Селектор кількості кроків counter | Не має вхідних параметрів | Повертає значення counter типу unsigned int | squarematrix.h |
| 34 | SquareMatrix | isInversable | Метод перевірки результату обернення матриці на помилки | double\*\* inversed | Повертає маркер  правильності обернення типу bool | squarematrix.h |

# Тестування програмного забезпечення

## План тестування

1. Тестування правильності введених значень.
   1. Тестування при введенні в матрицю некоректних символів.
   2. Тестування при неповному заповненні матриці.
2. Тестування коректної роботи при введенні матриць, що не можуть бути оберненими.
   1. Тестування роботи програми при нульовому значенні визначника.
3. Тестування коректності роботи методів 1, 2
   1. Перевірка коректності роботи методу 1.
   2. Перевірка коректності роботи методу 2.
4. Тестування коректності роботи методів 1, 2 з дробовими значеннями.
   1. Перевірка коректності роботи методу 1.
   2. Перевірка коректності роботи методу 2.

## Приклади тестування

Для того, щоб впевнитись в коректній роботі програми, проведемо ряд тестів, порядок виконання яких наведено у плані тестування (пункт 5.1)

У таблицях 5.1, 5.2 наведено результати тестування програми при введенні у матрицю некоректних символів (букв або некоректних знаків пунктуації).

Таблиця 5.1 – Тестування при введенні в матрицю некоректних символів (букв).

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення некоректних даних |
| Початковий стан програми | Відкрите головне вікно програми |
| Вхідні дані | 1 2 3 4 5 6 7 h 9 |
| Схема проведення тесту | Поелементне ручне заповнення матриці користувачем |
| Очікуваний результат | Повідомлення про помилку і неможливість обернення такої матриці |
| Стан програми після проведення випробувань | Видано помилку «Помилка! Ця матриця необоротна!» |

Таблиця 5.2 – Тестування при введенні некоректних знаків пунктуації.

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення некоректних даних |
| Початковий стан програми | Відкрите головне вікно програми |
| Вхідні дані | 1.4 2 3?1 4.3 5.12 6 7.01 4.34 9 |
| Схема проведення тесту | Поелементне заповнення матриці |
| Очікуваний результат | Повідомлення про помилку і неможливість обернення такої матриці |
| Стан програми після проведення випробувань | Видано помилку «Помилка! Ця матриця необоротна!» |

У таблиці 5.3 наведено результати неповного заповнення матриці. Передбачено, що незаповнені поля сприймаються програмою як 0.

Таблиця 5.3 – Тестування при неповному заповненні матриці.

|  |  |
| --- | --- |
| Мета тесту | Перевірити реакцію програми на неповне заповнення матриці |
| Початковий стан програми | Відкрите головне вікно програми |
| Вхідні дані | 1 2 3 4 5 6 7 8 |
| Схема проведення тесту | Поелементне ручне заповнення матриці користувачем, ігнорування заповнення останнього поля |
| Очікуваний результат | Сприйняття програмою порожнього поля як елемент, що = 0 |
| Стан програми після проведення випробувань | Виведено матрицю, що обернена до матриці 1 2 3 4 5 6 7 8 0 |

У таблиці 5.4 наведено результати обернення матриці, визначник якої дорівнює 0. Важливо, що оберненими можуть бути тільки квадратні невироджені матриці.

Таблиця 5.4 – Тестування при введенні матриці, визначник якої = 0.

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість обернення необоротної матриці |
| Початковий стан програми | Відкрите головне вікно програми |
| Вхідні дані | -1 3 4 -2 5 7 -1 1 2 |
| Схема проведення тесту | Поелементне заповнення матриці |
| Очікуваний результат | Повідомлення про помилку і неможливість обернення такої матриці |
| Стан програми після проведення випробувань | Видано помилку «Помилка! Ця матриця необоротна!» |

У таблиці 5.5 наведено результати обернення заданої оборотної матриці методом Жордана-Гауса. У таблиці 5.6 наведено результат роботи методу LUP-декомпозиції на такому ж наборі значень.

Таблиця 5.5 – Тестування роботи методу Жодана-Гауса при введенні оборотної матриці розмірності 3 на 3.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи алгоритму Жордана-Гауса |
| Початковий стан програми | Відкрите головне вікно програми |
| Вхідні дані | 2 5 7 6 3 4 5 -2 -3 |
| Схема проведення тесту | Поелементне ручне заповнення матриці користувачем |
| Очікуваний результат | Виведення матриці  1 -1 1 -38 41 -34 27 -29 24 |
| Стан програми після проведення випробувань | У поле для виводу оберненої матриці виведено таку матрицю:  1 -1 1 -38 41 -34 27 -29 24 |

Таблиця 5.6 – Тестування роботи методу LUP-декомпозиції при введенні оборотної матриці розмірності 3 на 3.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи алгоритму LUP-декомпозиції |
| Початковий стан програми | Відкрите головне вікно програми |
| Вхідні дані | 2 5 7 6 3 4 5 -2 -3 |
| Схема проведення тесту | Поелементне заповнення матриці |
| Очікуваний результат | Виведення матриці  1 -1 1 -38 41 -34 27 -29 24 |
| Стан програми після проведення випробувань | У поле для виводу оберненої матриці виведено таку матрицю:  1 -1 1 -38 41 -34 27 -29 24 |

У таблиці 5.7 наведено результати обернення заданої оборотної матриці з дробовими елементами методом Жордана-Гауса. У таблиці 5.8 наведено результат роботи методу LUP-декомпозиції на такому ж наборі значень.

Таблиця 5.7 – Тестування роботи методу Жодана-Гауса при введенні оборотної матриці розмірності 3 на 3 з дробовими значеннями.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи алгоритму 1 з дробовими значеннями |
| Початковий стан програми | Відкрите головне вікно програми |
| Вхідні дані | -0.2 -0.8 1.2 -0.7 0.7 -0.3 0.8 0.2 -0.8 |
| Схема проведення тесту | Поелементне ручне заповнення матриці користувачем |
| Очікуваний результат | Виведення матриці  5 4 6 8 8 9 7 6 7 |
| Стан програми після проведення випробувань | У поле для виводу оберненої матриці виведено таку матрицю:  5 4 6 8 8 9 7 6 7 |

Таблиця 5.8 – Тестування роботи методу LUP-декомпозиції при введенні оборотної матриці розмірності 3 на 3 з дробовими значеннями.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи алгоритму 2 з дробовими значеннями |
| Початковий стан програми | Відкрите головне вікно програми |
| Вхідні дані | -0.2 -0.8 1.2 -0.7 0.7 -0.3 0.8 0.2 -0.8 |
| Схема проведення тесту | Поелементне заповнення матриці |
| Очікуваний результат | Виведення матриці  5 4 6 8 8 9 7 6 7 |
| Стан програми після проведення випробувань | У поле для виводу оберненої матриці виведено таку матрицю:  5 4 6 8 8 9 7 6 7 |

# Інструкція користувача

## Робота з програмою

Для встановлення програми на свій ПК достатньо запустити файл встановлення MI\_Setup.exe, після чого користувач побачить вікно вибору мови, яка буде використовуватись при встановленні (Рисунок 6.1). Після вибору мови слід натиснути «ОК».

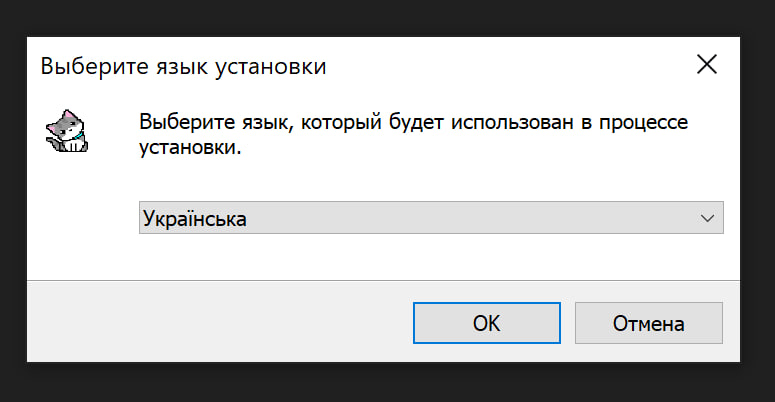


Рисунок 6.1 – Вікно вибору мови встановлення

Далі користувач побачить вікно вибору додаткових завдань. Тут варто поставити прапорець навпроти «Створити ярлики на Робочому столі», якщо такий прапорець відсутній (Рисунок 6.2). Після цього слід натиснути «Далі».

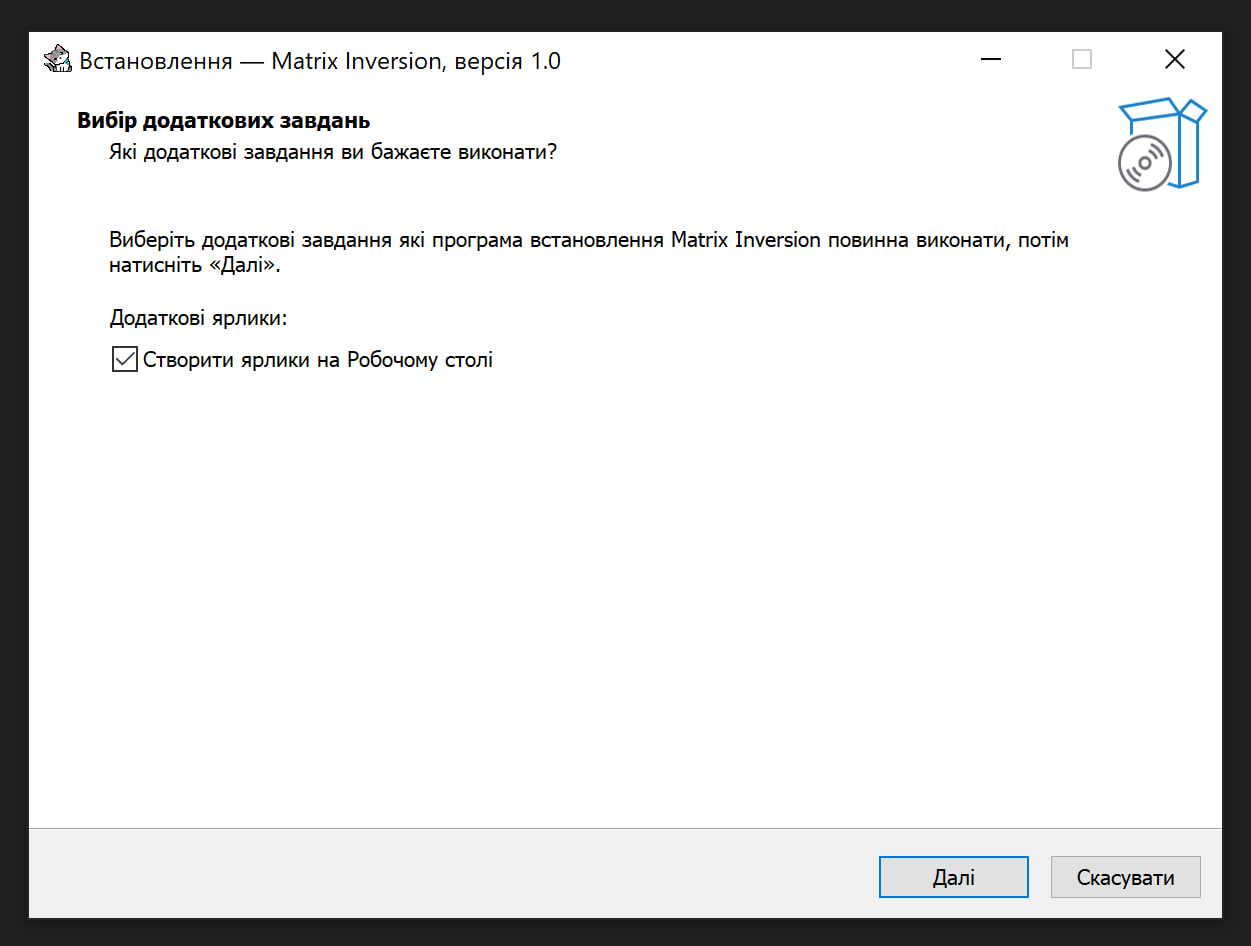


Рисунок 6.2 – Створення ярлику програми на Робочому столі

Після цього відкриється вікно готовності до встановлення (Рисунок 6.3). Тут треба натиснути «Встановити». Після цього програма встановиться за такою адресою C:\Program Files (x86)\Matrix Inversion. Окрім цього, ярлик буде встановлено на Робочому столі.

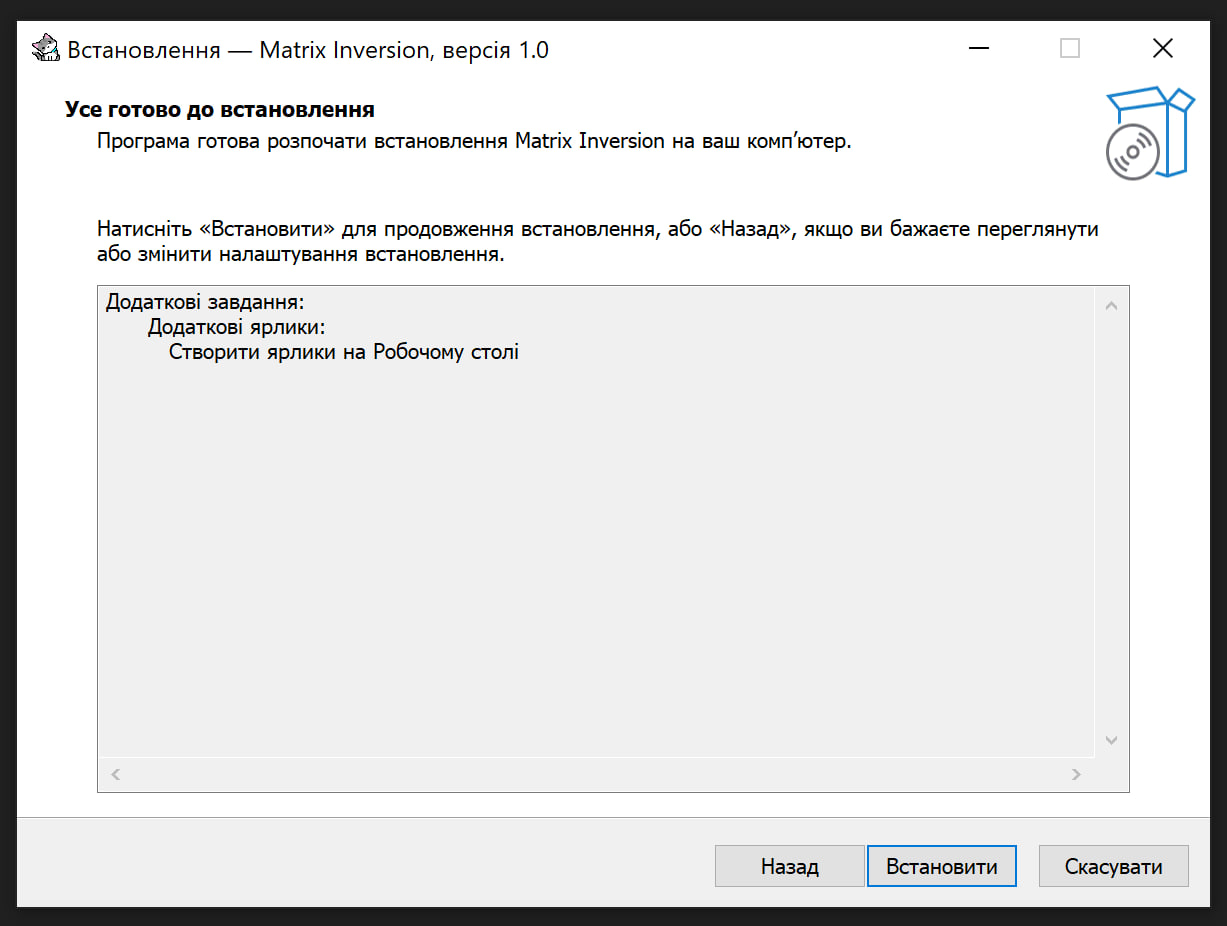


Рисунок 6.3 – Вікно готовності до встановлення

Після встановлення відкриється вікно завершення встановлення, де слід натиснути «Готово» (Рисунок 6.4). Якщо користувач не хоче одразу запустити програму, слід прибрати прапорець навпроти «Відкрити Matrix Inversion».

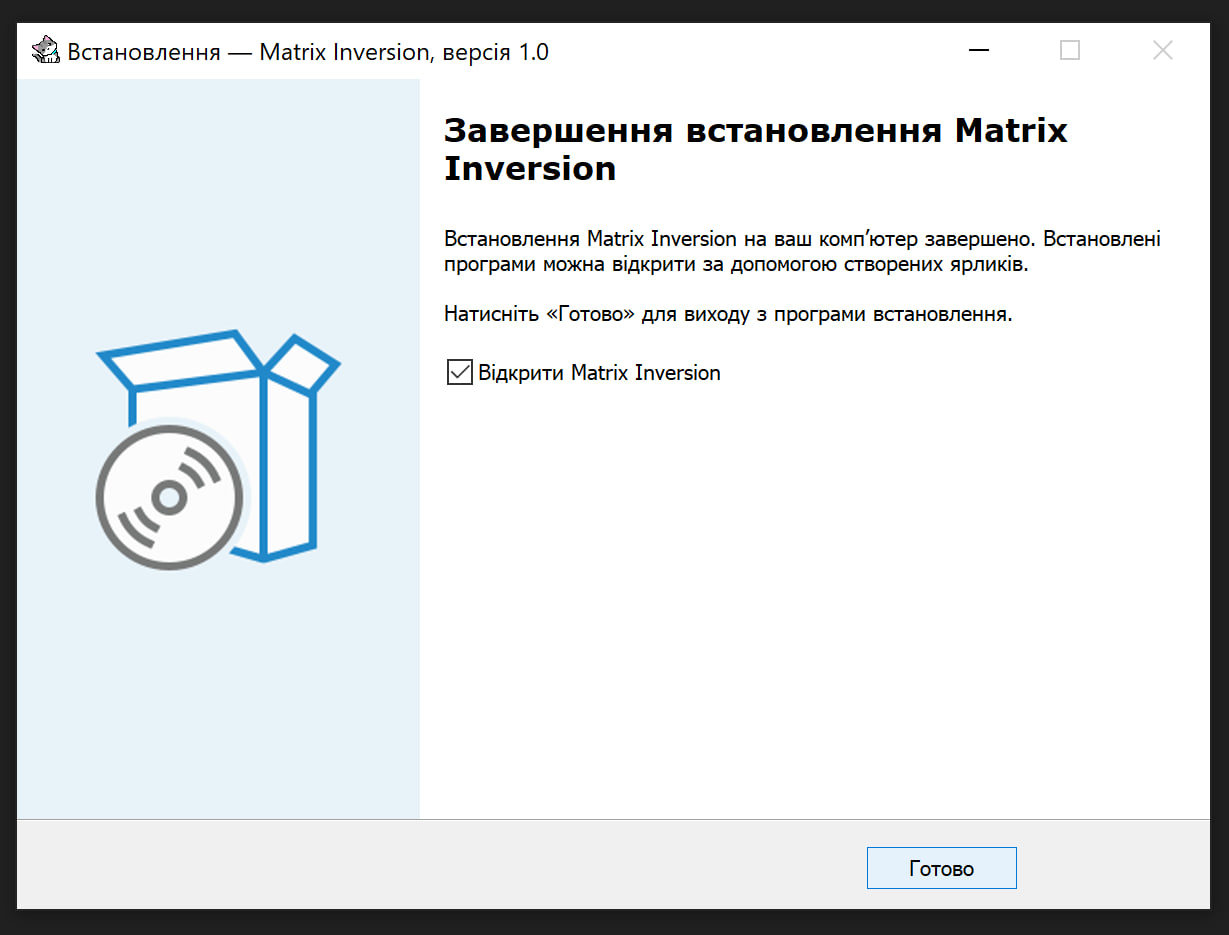


Рисунок 6.4 – Завершення встановлення програми

Після успішного встановлення програма відкриється сама (якщо прапорець навпроти «Відкрити Matrix Inversion» не було прибрано) або може бути відкрита через ярлик на Робочому столі. Тут користувача зустріне вікно вибору мови (Рисунок 6.5). Варто обрати зручну для себе мову роботи програми серед англійської, української та російської.

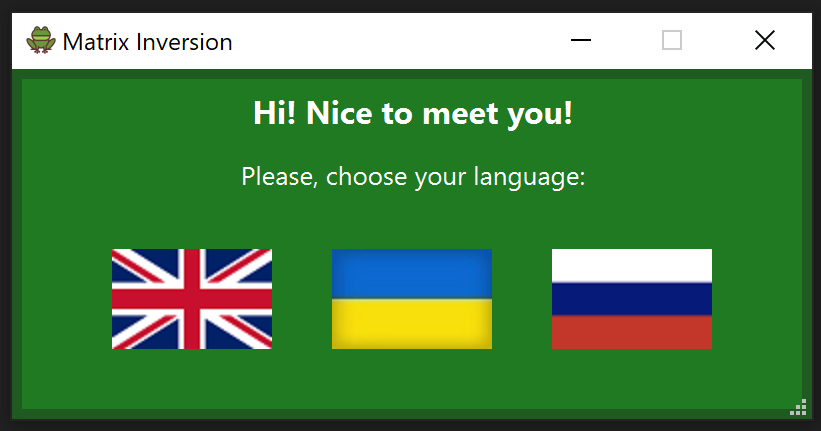


Рисунок 6.5 – Вікно вибору мови роботи програми

Після обрання мови роботи користувач побачить головне вікно роботи програми (Рисунок 6.6).

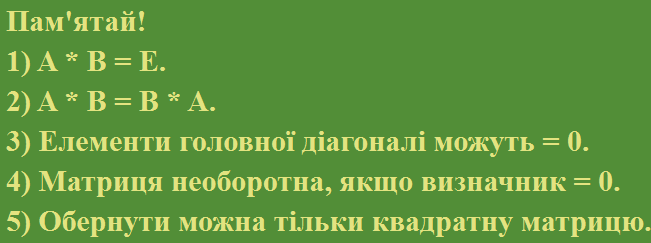
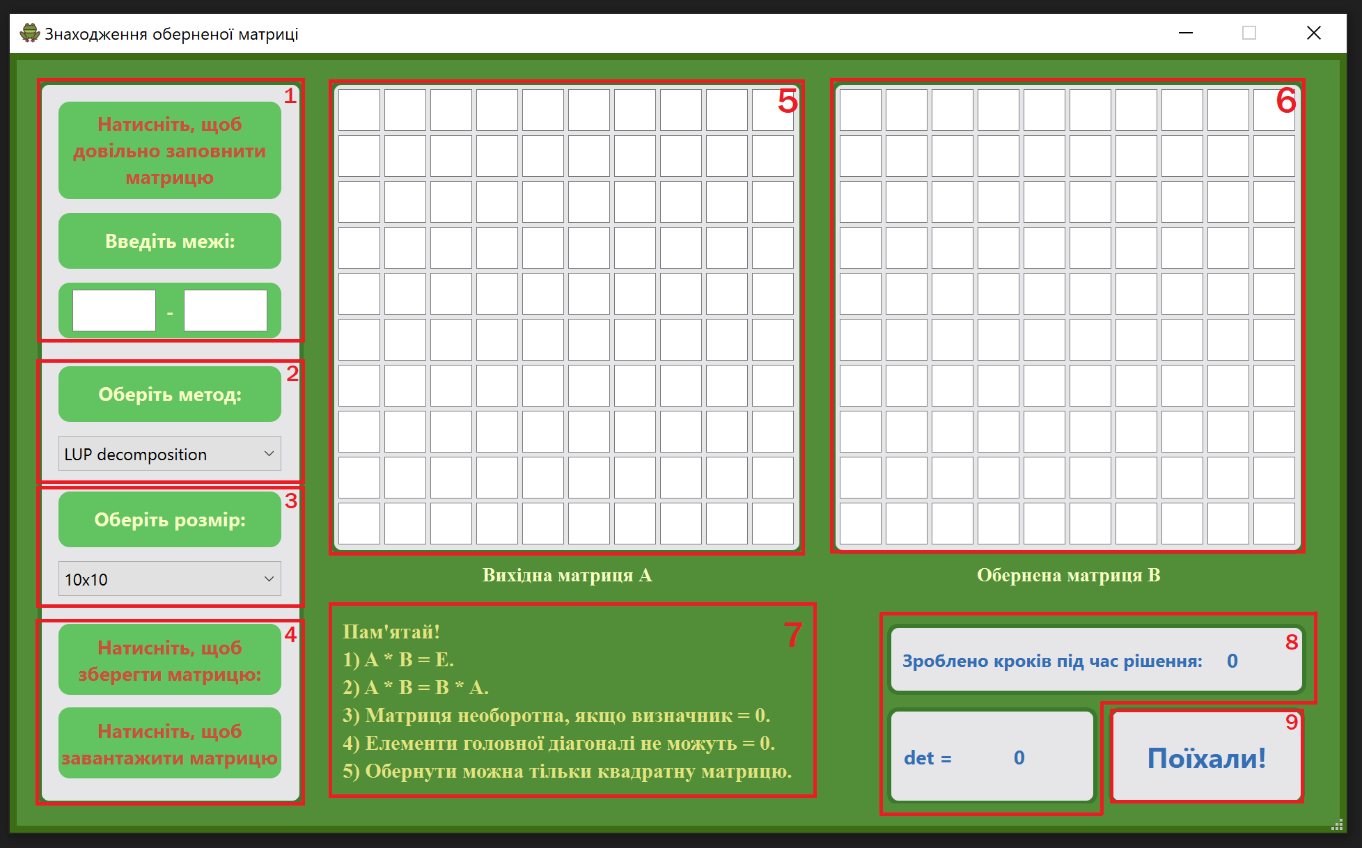


Рисунок 6.6 – Головне вікно програми

Тут наявні такі елементи: блок заповнення матриці випадковим чином (1), блок вибору методу розв’язання серед LUP-розкладу та методу Жордана-Гауса (2), блок вибору розмірності матриці від 2\*2 до 10\*10 (3), блок роботи з файлами – збереження у текстовий файл (4), вихідна матриця (5), обернена матриця (6), блок теорії (7), блок результатів роботи програми – кількості кроків та визначника матриці (8), кнопка запуску алгоритму обернення матриці (9).

Натискання на кнопку «Натисніть, щоб довільно заповнити матрицю» призведе до заповнення вихідної матриці елементами, згенерованими програмою у заданому користувачем діапазоні (Рисунок 6.7). Неважливо, у якому порядку – спадному чи зростаючому – будуть записані межі. Якщо межі будуть рівними, вся матриця буде заповнена однаковими елементами.

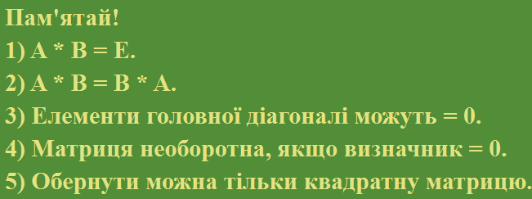
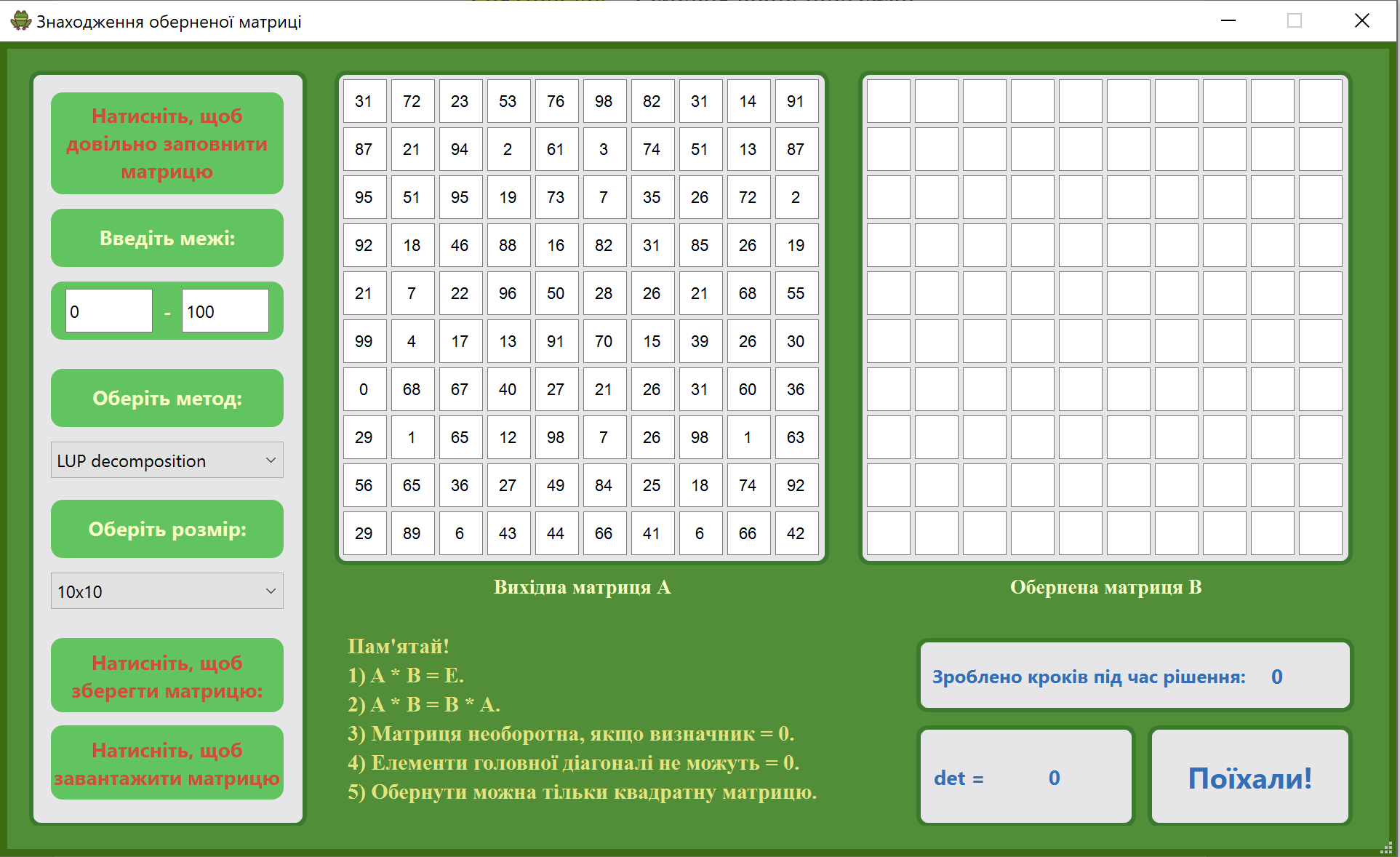


Рисунок 6.7 – Заповнення матриці випадковим чином

Далі користувач може обрати метод знаходження оберненої матриці. Це може бути або метод LUP-розкладу, або метод Жордана-Гауса (Рисунок 6.8).

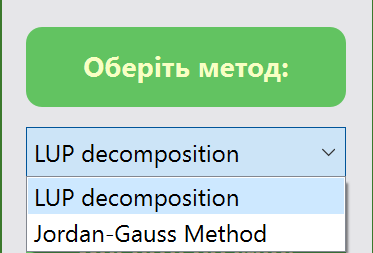


Рисунок 6.8 – Опція вибору методу обернення матриці

Далі користувач може змінити розмірність матриці. Зміна розмірності матриці змінить кількість і розмір доступних полів для вводу (Рисунок 6.9).

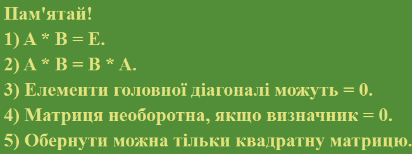
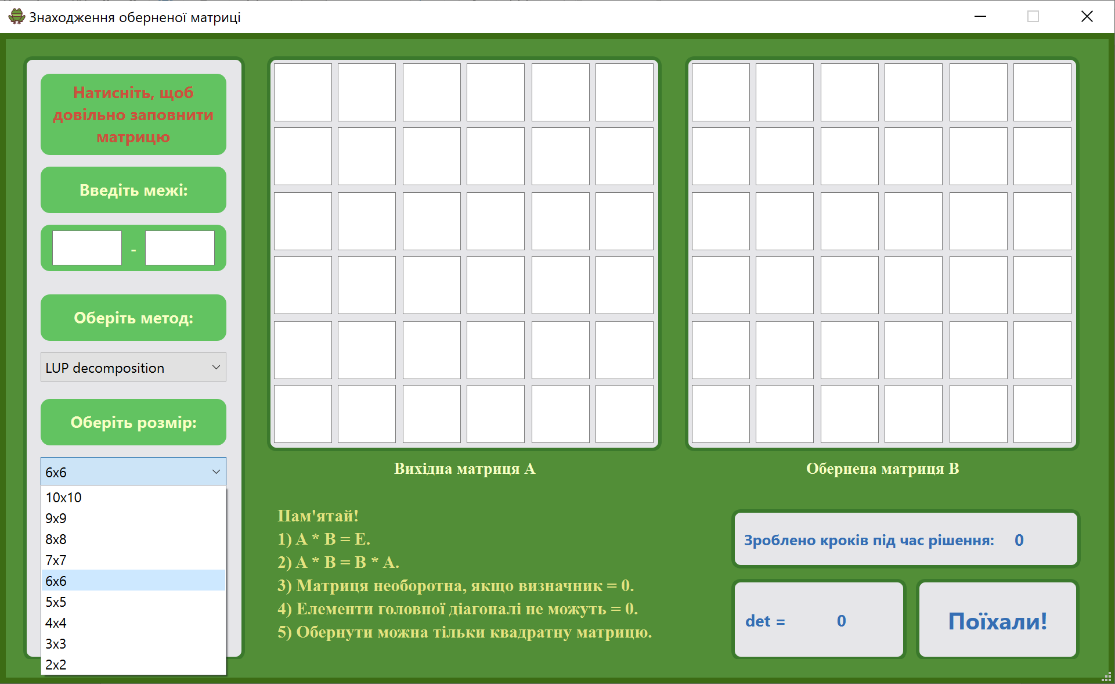


Рисунок 6.9 – Вибір розмірності матриці

За бажанням, користувач може ввести елементи матриці самостійно (Рисунок 6.10). Введені дані можуть бути як цілими значеннями, так і десятковими дробами.

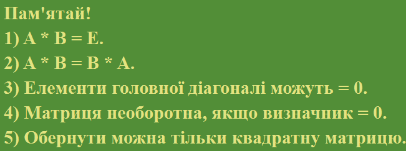
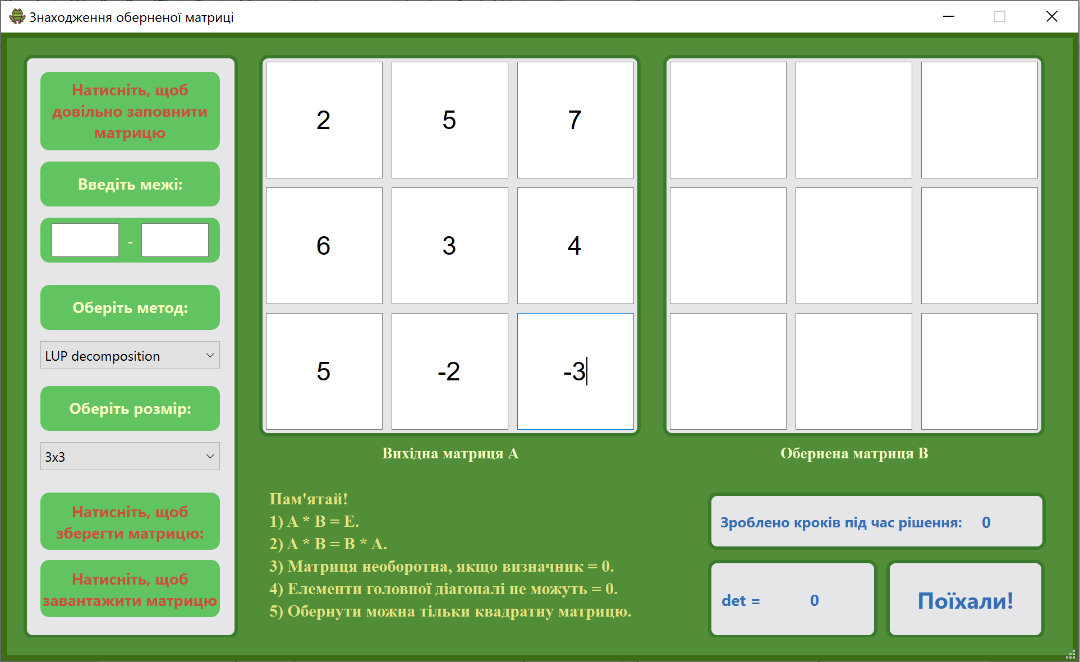


Рисунок 6.10 – Ручне введення елементів вихідної матриці користувачем

Далі користувачу слід натиснути кнопку «Поїхали!». Це запустить обраний користувачем алгоритм обернення матриці і виведе результат у блок оберненої матриці (Рисунок 6.11). Окрім цього, буде виведено кількість кроків, що знадобилася для вирішення даної задачі, і визначник вихідної матриці.

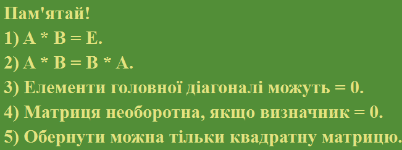
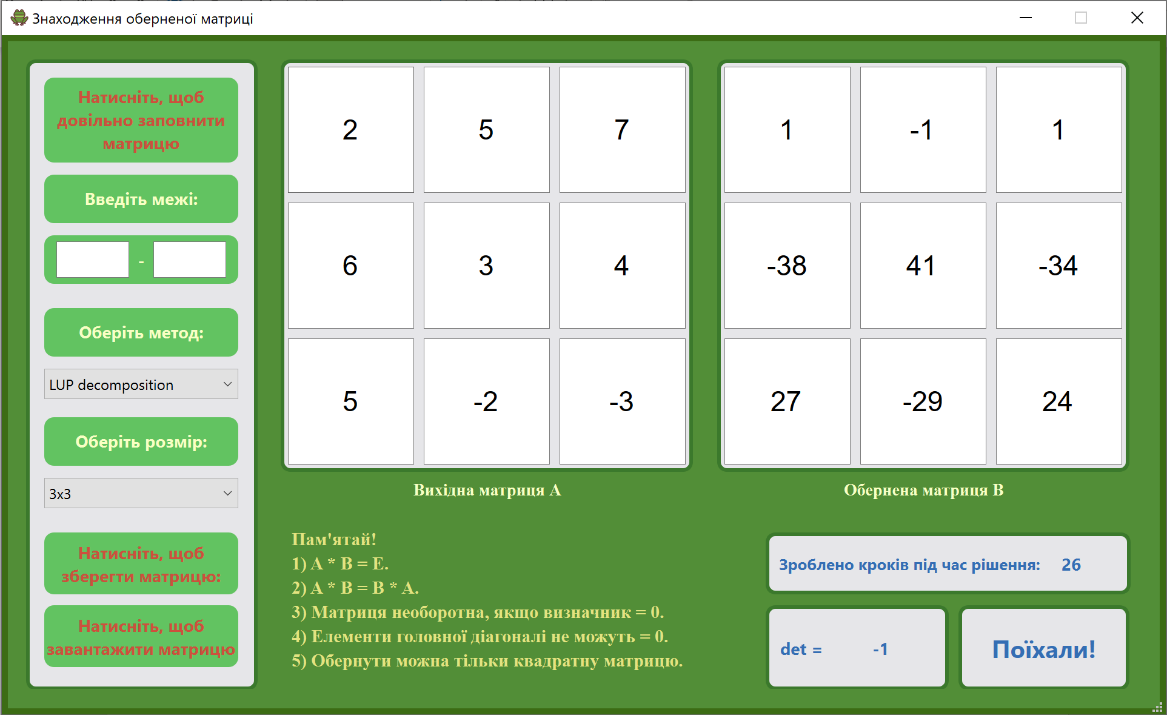


Рисунок 6.11 – Результат роботи програми

За бажанням, користувач може зберегти матриці. Для цього слід натиснути на «Натисніть, щоб зберегти матрицю». Важливо, що кожного разу файл с матрицями перезаписується, тобто запис може бути тільки один.

## Формат вхідних та вихідних даних

На вхід програми користувачем подається квадратна матриця заданої розмірності. Кожний елемент матриці - це число, яке може бути як цілим, так і десятковим дробом з точністю до 15 знаків після коми.

Результатом виконання програми є обернена матриця, визначник початкової матриці та кількість кроків, витрачених на розв’язання задачі. Елементи такої матриці – числа, які можуть бути як цілими, так і десятковими дробами з точністю до 15 знаків після коми.

## Системні вимоги

Системні вимоги до програмного забезпечення наведені в Таблиці 6.1.

Таблиця 6.1 – Системні вимоги програмного забезпечення

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows 10 | Windows 10 (з останніми оновленнями) або новіші системи Windows |
| Процесор | Intel Pentium ІІІ  1.0 GHz або  AMD Athlon 1.0 GHz | Intel i3-2100 3.1 GHz або AMD Athlon X2 245 2.9 GHz |
| Оперативна пам'ять | 1 GB RAM | 2 GB RAM |
| Відеоадаптер | Сумісний з DirectX 9 | Сумісний з DirectX 12 |
| Дисплей | 800х600 | 1024х768 або краще |
| Прилади введення | Клавіатура та комп’ютерна миша | |
| Додаткове ПЗ | - | |

# Аналіз і узагальнення результатів

Головною задачею курсової роботи була реалізація програми для обернення матриці наступними методами: метод Жордана-Гауса, метод LUP-декомпозиції.

Жодних критичних ситуацій в роботі програми виявлено не було. Помилки виникають лише тоді, коли користувачем були введені або некоректні дані (зайві букви, некоректні пунктуаційні символи), або матриці, які неможливо обернути (вироджені).

Для перевірки та доведення достовірності результатів виконання програмного забезпечення скористаюся веб-сайтом OnlineMSchool:

а) Метод Жордана-Гауса.

Результат виконання методу Жордана-Гауса наведено на рисунку 7.1:

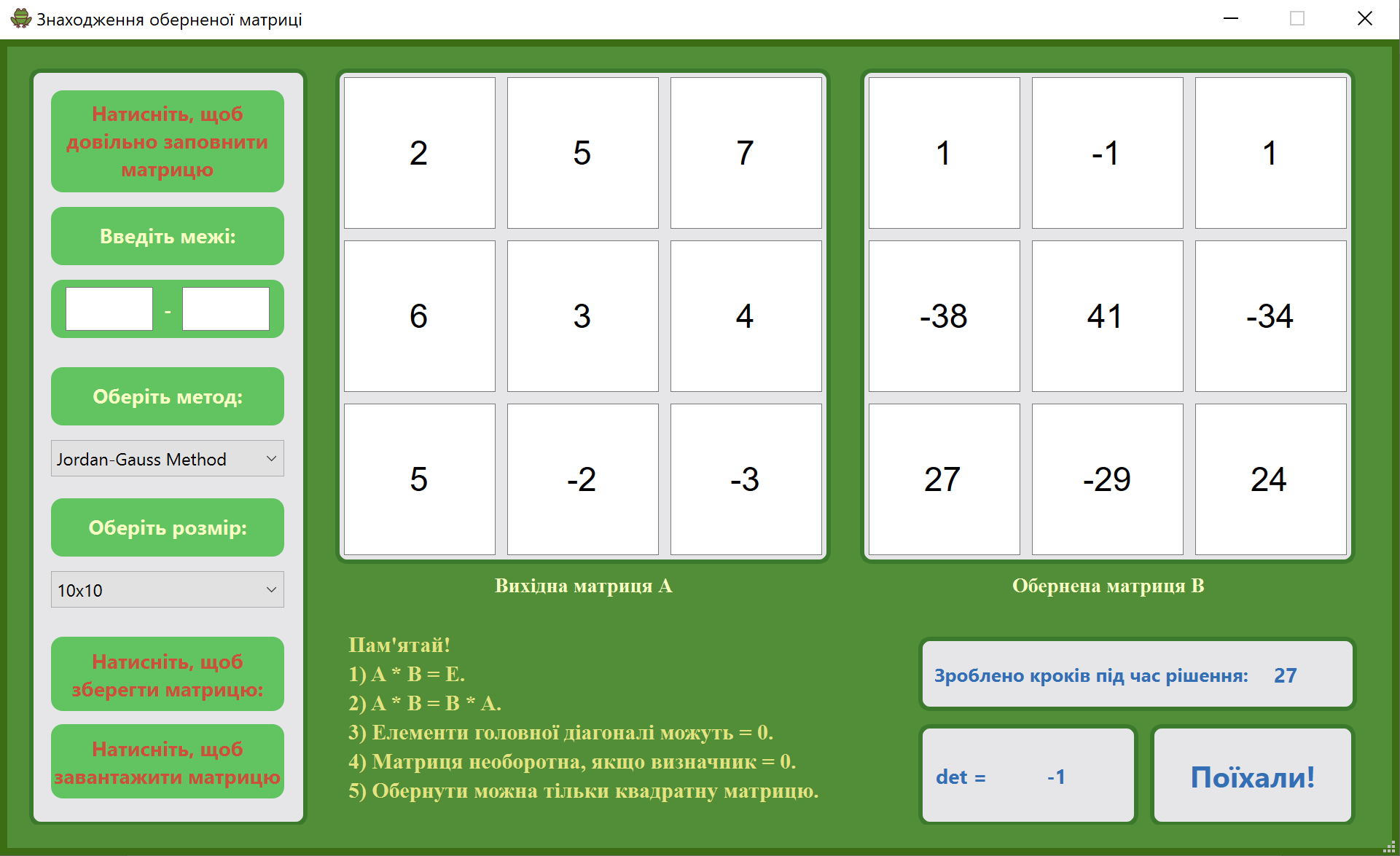


Рисунок 7.1 – Результат виконання методу Жордана-Гауса

б) Метод LUP-розкладу.

Результат виконання методу LUP-розкладу наведено на рисунку 7.2:

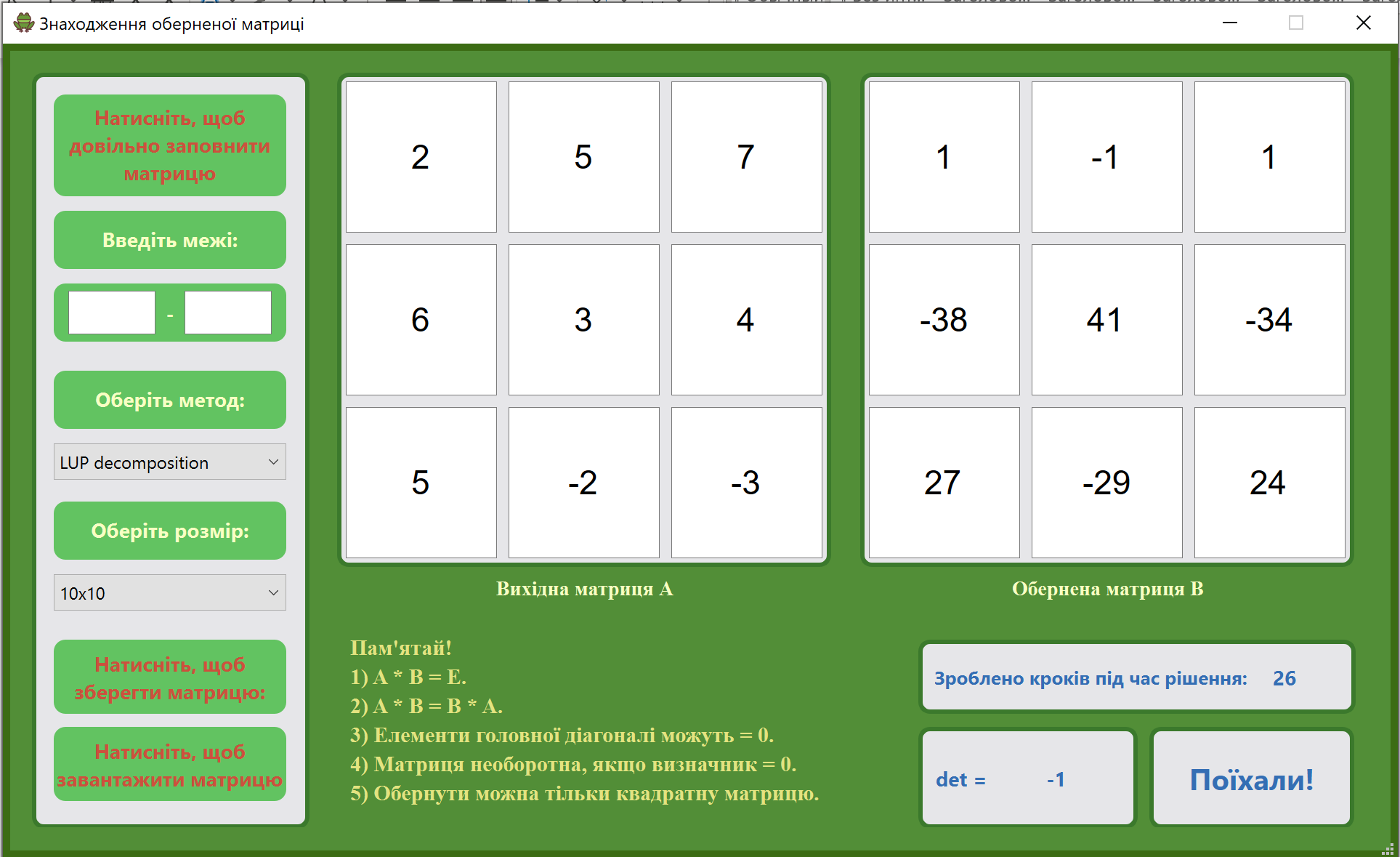


Рисунок 7.2 – Результат виконання методу LUP-розкладу

Оскільки результат виконання збігається з результатом на веб-сайті OnlineMSchool (рисунок 7.3), то дані методи працюють правильно.

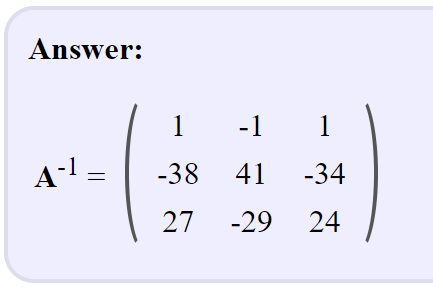


Рисунок 7.3 – Перевірка методу Жордана-Гауса на веб-сайті OnlineMSchool

Для проведення тестування ефективності програми було створено матриці наступного вигляду:

(7.1),

Матриця (7.1) є квадратною та невиродженою, і, завдяки цьому, може бути оберненою.

Результати тестування ефективності алгоритмів розв’язання СЛАР наведено в таблиці 7.1:

Таблиця 7.1 – Тестування ефективності методів

|  |  |  |  |
| --- | --- | --- | --- |
| Розмірність системи | Параметри тестування | Метод | |
| Жордана-Гауса | LUP-розкладу |
| 2 | Кількість ітерацій алгоритму | 8 | 5 |
| 3 | Кількість ітерацій алгоритму | 27 | 23 |
| 4 | Кількість ітерацій алгоритму | 64 | 62 |
| 5 | Кількість ітерацій алгоритму | 125 | 130 |
| 6 | Кількість ітерацій алгоритму | 216 | 235 |
| 7 | Кількість ітерацій алгоритму | 343 | 385 |
| 8 | Кількість ітерацій алгоритму | 512 | 588 |
| 9 | Кількість ітерацій алгоритму | 729 | 852 |
| 10 | Кількість ітерацій алгоритму | 1000 | 1185 |

Візуалізація результатів таблиці 7.1 наведено на рисунку 7.1:

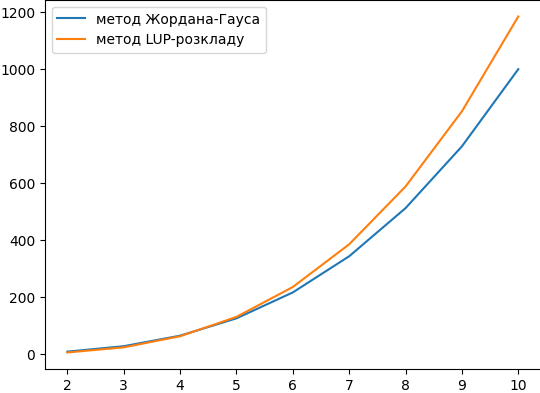


Рисунок 7.1 – Графік залежності кількості ітерацій алгоритму обернення від розміру вхідної системи

За результатами тестування можна зробити такі висновки:

1. Обидва алгоритми можуть ефективно обертати квадратні матриці.
2. Алгоритмічна складність обох методів O(n3), де n – розмірність матриці. Варто зазначити, що практична складність може дещо відрізнятися від теоретичної.
3. На невеликих матрицях (розмірність яких не перевищує 4) метод LUP-розкладу виявився кращим за Жордана-Гауса, але при більших розмірах матриці алгоритм Жордана-Гауса дещо кращий за метод LUP-декомпозиції.

Висновки

Метою виконання даної курсової роботи було створення програмного забезпечення зі зручним та інтуїтивно зрозумілим графічним інтерфейсом для обернення квадратних матриць із розмірністю від 2 до 10 методами Жордана-Гауса та LUP-декомпозиції.

Основне завдання додатку було описано у розділі «Постановка задачі». У розділі «Теоретичні відомості» було чітко описано принцип роботи обох алгоритмів. У розділі «Опис алгоритмів» було розписано загальний принцип роботи програми та наведено псевдокод обох алгоритмів. Діаграма класів, їх опис та опис їхніх методів були представлені у розділі «Опис програмного забезпечення». У 5 розділі програму було протестовано на різних значеннях, а також були досліджені реакції програми на ймовірні помилкові ситуації, спричинені неправильнім введенням або неможливістю обернути матрицю. У 6 розділі було наведено детальну інструкцію користувача, в тому числі й інструкцію щодо встановлення програми на свій ПК. Аналіз і узагальнення результатів було проведено у 7 розділі курсової роботи.

Результатом виконання курсової роботи є якісне програмне забезпечення для обернення матриці. Інтерфейс програми доступний на 3 мовах: англійській, українській та російській. Програма вміє працювати з різними матрицями розмірності від 2x2 до 10x10. Реалізовано обидва методи обертання матриці, збереження матриць у текстовий файл.

У результаті тестування програми помилок у реалізації не виявлено, програма чітко та коректно реагує на будь-які помилкові ситуації. Обернення матриці виконується правильно.

Перелік посилань

* + - 1. Jaan Kiusalaas. Numerical Methods in Engineering with Python 3. New York, NY : Cambridge University Press, 2013. 421 с.
      2. Scratchapixel. Matrix Inverse: Gauss-Jordan Method. URL:

<https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/matrix-inverse/matrix-inverse.html>.

(дата звернення: 23.03.2023).

* + - 1. QT Documentation. Creating a Qt Widget Based Application. URL:

<https://doc.qt.io/qtcreator/creator-writing-program.html>.

(дата звернення: 29.04.2023).

* + - 1. OnlineMSchool. Inverse matrix calculator (Gaussian Elimination). URL:

<https://onlinemschool.com/math/assistance/matrix/inverse/>.

(дата звернення: 19.05.2023).

* + - 1. Википедия. Метод Гаусса – Жордана. URL:

<https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0_%E2%80%94_%D0%96%D0%BE%D1%80%D0%B4%D0%B0%D0%BD%D0%B0>. (дата звернення: 23.03.2023).

* + - 1. Википедия. Обратная матрица. URL:

<https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0>. (дата звернення: 23.03.2023).

Додаток А Технічне завдання

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

інформатики та програмної інженерії

Затвердив

Керівник Головченко М.М. ㅤ

«20» лютого 2023 р.

Виконавець:

Студент Нижник Д.С. ㅤ

«06» червня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: «Обернення матриці»

з дисципліни:

«Основи програмування»

Київ 2023

* 1. *Мета*: Метою курсової роботи є розробка якісного та надійного ПЗ для виконання обернення матриці за допомогою методів Жордана-Гауса та LUP-розкладу.
  2. *Дата початку роботи: «12» лютого 2023 р.*
  3. *Дата закінчення роботи: «31» травня 2023 р.*
  4. *Вимоги до програмного забезпечення.*

1. Функціональні вимоги:

− Можливість обрання користувачем розмірності матриці (до 10)

− Можливість заповнення матриці користувачем вручну

− Можливість генерації матриці випадковим чином

− Реалізація зручного та інтуїтивно зрозумілого інтерфейсу

− Реалізація опції вибору мови інтерфейсу

(українська/англійська/російська мови)

− Можливість вибору користувачем методу розв’язання задачі

− Можливість збереження результатів виконання завдання у файл

− Виведення аналітичної оцінки складності обраного алгоритму

1. Нефункціональні вимоги:

− Можливість роботи програми на ОС Windows версії 10 та вище

* Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

* 1. *Стадії та етапи розробки*:

1. Об'єктно-орієнтований аналіз предметної області задачі (до 08.04.2023 р.)
2. Об'єктно-орієнтоване проектування архітектури програмної системи (до 25.04.2023 р.)
3. Розробка програмного забезпечення (до 09.05.2023 р.)
4. Тестування розробленої програми (до 19.05.2023 р.)
5. Розробка пояснювальної записки (до 31.05.2023 р.).
6. Захист курсової роботи (до 06.06.2023 р.).
   1. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

Додаток Б Тексти програмного коду

*студента групи ІП-22 І курсу*

*Нижника Д.С.*

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення вирішення задачі обернення матриць “Matrix Inversion”*

(Вид носія даних)

*Інсталяційний файл*

(Обсяг програми (документа), арк., Кб)

*21 арк, 46,2 Кб*

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include "ui\_mainwindow.h"

#include "squarematrix.h"

using namespace std;

namespace Ui

{

class MainWindow;

class Matrices;

}

class Matrices

{

protected:

vector<vector<QLineEdit\*>> numberInputFieldsN;

vector<vector<QLineEdit\*>> numberInputFieldsI;

int dimension = 10;

public:

void showmatrix(int);

void hideall();

void fillMatrix(double\*\*);

};

class MainWindow : public QMainWindow, public Matrices

{

Q\_OBJECT

public:

// Constructors and destructors

explicit MainWindow(QWidget \*parent = nullptr);

~MainWindow();

// visual changes

void open(QString&);

void outputDet(double);

void outputMatrix(double\*\*);

void outputCount(unsigned int);

// getters

int getdimension();

int getmethod();

// Error message

QMessageBox m\_errorMessageBox;

signals:

void StartSolving();

void StartRandomize();

private slots:

void handleComboBoxMethodChanged(int);

void handleComboBoxIndexChanged(int);

void on\_pushButton\_clicked();

void on\_Randomize\_clicked();

void on\_Download\_clicked();

void on\_Save\_clicked();

private:

QLineEdit \*numberInputField1;

QLineEdit \*numberInputField2;

friend SquareMatrix;

Ui::MainWindow \*ui;

int method = 1;

};

#endif

**squarematrix.h**

#pragma once

#include <QApplication>

#include <QMainWindow>

#include <QVBoxLayout>

#include <QPushButton>

#include <QMessageBox>

#include <QComboBox>

#include <QLineEdit>

#include <QObject>

#include <QFile>

#include <fstream>

#include <string>

#include <random>

#include <ctime>

using namespace std;

class SquareMatrix {

protected:

int dimension;

double \*\*matrix;

double determinant;

unsigned int counter;

public:

// Constructors and destructors

SquareMatrix(int);

~SquareMatrix();

// Matrix setter

void setmatrix(double\*\* temp);

// Matrix Inversion Methods

double\*\* LUPd();

double\*\* JGMethod();

// Determinant methods

void findDet();

double GetDet();

// Steps getter

unsigned int getCounter();

// Reversibility test

bool isInversable(double\*\*);

};

**startwindow.h**

#ifndef STARTWINDOW\_H

#define STARTWINDOW\_H

#include "./ui\_startwindow.h"

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui { class StartWindow; }

QT\_END\_NAMESPACE

class StartWindow : public QMainWindow

{

Q\_OBJECT

public:

StartWindow(QWidget \*parent = nullptr);

~StartWindow();

void open();

signals:

void languageSelected(QString language);

private slots:

void on\_RussianButton\_clicked();

void on\_UkrainianButton\_clicked();

void on\_BritishButton\_clicked();

private:

Ui::StartWindow \*ui;

};

#endif

**main.cpp**

#include "startwindow.h"

#include "mainwindow.h"

#include "squarematrix.h"

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

// Application language selection window

StartWindow ChooseLanguageWindow;

ChooseLanguageWindow.open();

// Application main window

MainWindow InversionMatrixWindow;

// Language chosen

QObject::connect(&ChooseLanguageWindow, &StartWindow::languageSelected,

[&ChooseLanguageWindow, &InversionMatrixWindow](QString language) {

// Files

QString projectPath = QCoreApplication::applicationDirPath();

QString LanguagePath = projectPath + "/" + language + ".txt";

// Open Matrix Inversion window

InversionMatrixWindow.open(LanguagePath);

InversionMatrixWindow.showmatrix(10);

InversionMatrixWindow.outputCount(0);

InversionMatrixWindow.outputDet(0);

// Close Language Choose window

ChooseLanguageWindow.close();

});

// Initial matrix

SquareMatrix\* Matrix = nullptr;

// Start button

QObject::connect(&InversionMatrixWindow, &MainWindow::StartSolving,

[&InversionMatrixWindow, &Matrix]() {

// info about matrix

int dimension = InversionMatrixWindow.getdimension();

int method = InversionMatrixWindow.getmethod();

// Initialize matrix

Matrix = new SquareMatrix(dimension);

// Create initial and inversed matrices

double\*\* inversedmatrix;

double\*\* matrix = new double\*[dimension];

for (int i = 0; i < dimension; i++) {

matrix[i] = new double[dimension];

}

// Fill initial matrix

InversionMatrixWindow.fillMatrix(matrix);

Matrix->setmatrix(matrix);

// Find determinant

Matrix->findDet();

double determinant = Matrix->GetDet();

// Is matrix reversable?

if ((determinant != 0) && (!isnan(determinant))) {

// Use chosen method to inverse

if (method == 1) inversedmatrix = Matrix->LUPd();

else inversedmatrix = Matrix->JGMethod();

// isnan + 0 on the main diagonal?

bool markerInv = Matrix->isInversable(inversedmatrix);

if (markerInv) {

// Print determinant

InversionMatrixWindow.outputDet(determinant);

// Print inversed matrix

InversionMatrixWindow.outputMatrix(inversedmatrix);

// Print counter

InversionMatrixWindow.outputCount(Matrix->getCounter());

}

else {

// Error

InversionMatrixWindow.m\_errorMessageBox.exec();

}

}

else {

// Error

InversionMatrixWindow.m\_errorMessageBox.exec();

}

});

// End

return a.exec();

}

**mainwindow.cpp**

#include "mainwindow.h"

// Constructors and destructors

MainWindow::MainWindow(QWidget \*parent) : QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

// Border of random 1

numberInputField1 = new QLineEdit(this);

numberInputField1->setGeometry(45, 170, 60, 30);

// Border of random 2

numberInputField2 = new QLineEdit(this);

numberInputField2->setGeometry(125, 170, 60, 30);

// Combo Box 1

QComboBox \*comboBox = new QComboBox(this);

comboBox->addItem("10x10");

comboBox->addItem("9x9");

comboBox->addItem("8x8");

comboBox->addItem("7x7");

comboBox->addItem("6x6");

comboBox->addItem("5x5");

comboBox->addItem("4x4");

comboBox->addItem("3x3");

comboBox->addItem("2x2");

comboBox->setGeometry(35, 365, 160, 25);

connect(comboBox, QOverload<int>::of(&QComboBox::currentIndexChanged), this, &MainWindow::handleComboBoxIndexChanged);

// Combo Box 2

QComboBox \*comboBox2 = new QComboBox(this);

comboBox2->addItem("LUP decomposition");

comboBox2->addItem("Jordan-Gauss Method");

comboBox2->setGeometry(35, 275, 160, 25);

connect(comboBox2, QOverload<int>::of(&QComboBox::currentIndexChanged), this, &MainWindow::handleComboBoxMethodChanged);

// Matrixes

int startX = 236;

int startY = 27;

int fieldWidth = 30;

int fieldHeight = 30;

numberInputFieldsN.resize(10);

numberInputFieldsI.resize(10);

for (int i = 0; i < 10; i++) {

numberInputFieldsN[i].resize(10);

numberInputFieldsI[i].resize(10);

for (int j = 0; j < 10; j++) {

int x = startX + i \* 33;

int y = startY + j \* 33;

numberInputFieldsN[i][j] = new QLineEdit(this);

numberInputFieldsI[i][j] = new QLineEdit(this);

numberInputFieldsN[i][j]->setGeometry(x, y, fieldWidth, fieldHeight);

numberInputFieldsI[i][j]->setGeometry(x+360, y, fieldWidth, fieldHeight);

numberInputFieldsI[i][j]->setReadOnly(true);

numberInputFieldsN[i][j]->setAlignment(Qt::AlignCenter);

numberInputFieldsI[i][j]->setAlignment(Qt::AlignCenter);

}

}

}

MainWindow::~MainWindow()

{

delete ui;

delete numberInputField1;

delete numberInputField2;

for (int i = 0; i < 10; ++i)

{

for (int j = 0; j < 10; ++j)

{

delete numberInputFieldsN[i][j];

delete numberInputFieldsI[i][j];

}

}

}

// Visual changes

void MainWindow::open(QString& LanguagePath)

{

// Files

QFile file(LanguagePath);

QTextStream languagefile(&file);

QString projectPath = QCoreApplication::applicationDirPath();

QIcon logo(projectPath + "/Frog-logo.png");

// QStrings

QString title, choosemethod, random, chooserandom, asymptotic, savematrix, start, openmatrix,

choosedimension, matrix1, matrix2, errormessage,

Theory1, Theory2, Theory3, Theory4, Theory5, Theory6;

// Open the file

file.open(QIODevice::ReadOnly | QIODevice::Text);

// Read the file

title = languagefile.readLine();

choosemethod = languagefile.readLine();

random = languagefile.readLine();

chooserandom = languagefile.readLine();

asymptotic = languagefile.readLine();

savematrix = languagefile.readLine();

start = languagefile.readLine();

openmatrix = languagefile.readLine();

choosedimension = languagefile.readLine();

matrix1 = languagefile.readLine();

matrix2 = languagefile.readLine();

Theory1 = languagefile.readLine();

Theory2 = languagefile.readLine();

Theory3 = languagefile.readLine();

Theory4 = languagefile.readLine();

Theory5 = languagefile.readLine();

Theory6 = languagefile.readLine();

errormessage = languagefile.readLine();

// Close the file

file.close();

// Application main window settings

this->setStyleSheet("QMainWindow { border: 5px solid #3c6c14; background-color: #528e37; }");

this->setWindowTitle(title);

this->setWindowIcon(logo);

this->resize(960, 560);

//

this->m\_errorMessageBox.setText(errormessage);

m\_errorMessageBox.setWindowIcon(logo);

m\_errorMessageBox.setIcon(QMessageBox::Critical);

m\_errorMessageBox.setWindowFlags(Qt::Window | Qt::WindowStaysOnTopHint);

// InfoWindow + Matrices

QLabel \*InfoWindow = this->findChild<QLabel\*>("InfoWindow");

InfoWindow->setStyleSheet("QLabel { border-radius: 8px; border: 3px solid #3b792c; background-color: #e6e6e9; }");

QLabel \*Matrix1 = this->findChild<QLabel\*>("Matrix1");

Matrix1->setStyleSheet("QLabel { border-radius: 8px; border: 3px solid #3b792c; background-color: #e6e6e9; }");

QLabel \*Matrix2 = this->findChild<QLabel\*>("Matrix2");

Matrix2->setStyleSheet("QLabel { border-radius: 8px; border: 3px solid #3b792c; background-color: #e6e6e9; }");

// text1

QLabel \*text1 = this->findChild<QLabel\*>("text1");

text1->setStyleSheet("QLabel { background-color: #62c361; border-radius: 8px; text-align: center;"

"color: #f8ffc3; font-size: 14px; font-weight:bold; }");

text1->setText("<center>" + choosemethod + "</center>");

// text2

QLabel \*text2 = this->findChild<QLabel\*>("text2");

text2->setStyleSheet("QLabel { background-color: #62c361; border-radius: 8px; text-align: center;"

"color: #cd503d; font-size: 14px; font-weight:bold; }");

text2->setText("<center>" + random + "</center>");

text2->setWordWrap(true);

// text3

QLabel \*text3 = this->findChild<QLabel\*>("text3");

text3->setStyleSheet("QLabel { background-color: #62c361; border-radius: 8px; text-align: center;"

"color: #f8ffc3; font-size: 14px; font-weight:bold; }");

text3->setText("<center>" + chooserandom + "</center>");

text3->setWordWrap(true);

// text4

QLabel \*text4 = this->findChild<QLabel\*>("text4");

text4->setStyleSheet("QLabel { border-radius: 8px; border: 3px solid #3b792c; background-color: #e6e6e9; text-align: center;"

"color: #346eb4; font-size: 20px; font-weight:bold; }");

text4->setAlignment(Qt::AlignCenter);

text4->setWordWrap(true);

text4->setText(start);

// text5

QLabel \*text5 = this->findChild<QLabel\*>("text5");

text5->setStyleSheet("QLabel { background-color: #62c361; border-radius: 8px; text-align: center;"

"color: #cd503d; font-size: 14px; font-weight:bold; }");

text5->setAlignment(Qt::AlignCenter);

text5->setWordWrap(true);

text5->setText(savematrix);

// text6

QLabel \*text6 = this->findChild<QLabel\*>("text6");

text6->setStyleSheet("QLabel { background-color: #62c361; border-radius: 8px; text-align: center;"

"color: #f8ffc3; font-size: 14px; font-weight:bold; }");

text6->setAlignment(Qt::AlignCenter);

text6->setWordWrap(true);

text6->setText("-");

// text8

QLabel \*text8 = this->findChild<QLabel\*>("text8");

text8->setStyleSheet("QLabel { background-color: #62c361; border-radius: 8px; text-align: center;"

"color: #f8ffc3; font-size: 14px; font-weight:bold; }");

text8->setText("<center>" + choosedimension + "</center>");

// Asymptotic

QLabel \*Asymptotic = this->findChild<QLabel\*>("Asymptotic");

Asymptotic->setStyleSheet("QLabel { border-radius: 8px; border: 3px solid #3b792c; background-color: #e6e6e9;"

"text-align: center; color: #346eb4; font-size: 13px; font-weight:bold; padding-left: 5px;}");

Asymptotic->setText(asymptotic);

// MatrixName1

QLabel \*MatrixName1 = this->findChild<QLabel\*>("MatrixName1");

MatrixName1->setStyleSheet("QLabel { background-color: rgba(255, 255, 255, 0); text-align: center; color: #f8ffc3; "

"font-size: 14px; font-weight: bold; font-family: Times New Roman; }");

MatrixName1->setText("<center>" + matrix1 + "</center>");

// MatrixName2

QLabel \*MatrixName2 = this->findChild<QLabel\*>("MatrixName2");

MatrixName2->setStyleSheet("QLabel { background-color: rgba(255, 255, 255, 0); text-align: center; color: #f8ffc3; "

"font-size: 14px; font-weight: bold; font-family: Times New Roman; }");

MatrixName2->setText("<center>" + matrix2 + "</center>");

// Theory labels

QLabel \*Theory1l = this->findChild<QLabel\*>("Theory1");

Theory1l->setStyleSheet("QLabel { background-color: rgba(255, 255, 255, 0); text-align: center; color: #e3e380; "

"font-size: 15px; font-weight: bold; font-family: Times New Roman; padding-left: 5px;}");

Theory1l->setText(Theory1);

QLabel \*Theory2l = this->findChild<QLabel\*>("Theory2");

Theory2l->setStyleSheet("QLabel { background-color: rgba(255, 255, 255, 0); text-align: center; color: #e3e380; "

"font-size: 15px; font-weight: bold; font-family: Times New Roman; padding-left: 5px;}");

Theory2l->setText(Theory2);

QLabel \*Theory3l = this->findChild<QLabel\*>("Theory3");

Theory3l->setStyleSheet("QLabel { background-color: rgba(255, 255, 255, 0); text-align: center; color: #e3e380; "

"font-size: 15px; font-weight: bold; font-family: Times New Roman; padding-left: 5px;}");

Theory3l->setText(Theory3);

QLabel \*Theory4l = this->findChild<QLabel\*>("Theory4");

Theory4l->setStyleSheet("QLabel { background-color: rgba(255, 255, 255, 0); text-align: center; color: #e3e380; "

"font-size: 15px; font-weight: bold; font-family: Times New Roman; padding-left: 5px;}");

Theory4l->setText(Theory4);

QLabel \*Theory5l = this->findChild<QLabel\*>("Theory5");

Theory5l->setStyleSheet("QLabel { background-color: rgba(255, 255, 255, 0); text-align: center; color: #e3e380; "

"font-size: 15px; font-weight: bold; font-family: Times New Roman; padding-left: 5px;}");

Theory5l->setText(Theory5);

QLabel \*Theory6l = this->findChild<QLabel\*>("Theory6");

Theory6l->setStyleSheet("QLabel { background-color: rgba(255, 255, 255, 0); text-align: center; color: #e3e380; "

"font-size: 15px; font-weight: bold; font-family: Times New Roman; padding-left: 5px;}");

Theory6l->setText(Theory6);

// determinant

QLabel \*determinant = this->findChild<QLabel\*>("determinant");

determinant->setStyleSheet("QLabel { border-radius: 8px; border: 3px solid #3b792c; background-color: #e6e6e9; text-align: center;"

"color: #346eb4; font-size: 14px; font-weight:bold; padding-left: 5px; }");

determinant->setText("det = ");

// Show the window

this->show();

}

void MainWindow::outputDet(double det)

{

QLabel \*Det = this->findChild<QLabel\*>("Det");

Det->setStyleSheet("QLabel { text-align: center; color: #346eb4; font-size: 14px; font-weight:bold; padding-left: 7px;}");

Det->setText("<center>" + QString::number(det) + "</center>");

}

void MainWindow::outputMatrix(double\*\* matrix)

{

for (int i = 0; i < dimension; ++i)

{

for (int j = 0; j < dimension; ++j)

{

numberInputFieldsI[i][j]->setText(QString::number(matrix[i][j]));

}

}

}

void MainWindow::outputCount(unsigned int steps)

{

QLabel \*counter = this->findChild<QLabel\*>("counter");

counter->setStyleSheet("QLabel { text-align: center; color: #346eb4; font-size: 14px; font-weight:bold; padding-left: 10px;}");

counter->setText(QString::number(steps));

}

// Getters

int MainWindow::getdimension()

{

return dimension;

}

int MainWindow::getmethod()

{

return method;

}

// Private slots

void MainWindow::handleComboBoxMethodChanged(int method)

{

QComboBox\* comboBox = qobject\_cast<QComboBox\*>(sender());

if (comboBox) {

QString selectedMethod = comboBox->currentText();

if (selectedMethod == "LUP decomposition") { this->method = 1; }

else if (selectedMethod == "Jordan-Gauss Method") { this->method = 2; }

}

}

void MainWindow::handleComboBoxIndexChanged(int index)

{

QComboBox \*comboBox = qobject\_cast<QComboBox\*>(sender());

std::string selectedText = std::string(comboBox->currentText().toUtf8().constData());

this->dimension = std::stoi(selectedText.substr(0, selectedText.find('x')));

this->showmatrix(dimension);

}

void MainWindow::on\_pushButton\_clicked()

{

emit StartSolving();

}

void MainWindow::on\_Randomize\_clicked()

{

srand(time(nullptr));

double lower, upper;

lower = numberInputField1->text().toDouble();

upper = numberInputField2->text().toDouble();

if (lower > upper) {

lower += upper;

upper = lower - upper;

lower -= upper;

}

for (int i = 0; i < dimension; ++i)

{

for (int j = 0; j < dimension; ++j)

{

if (((int)upper == upper) && ((int)lower == lower)) {

numberInputFieldsN[i][j]->setText(QString::number(rand() % (int)(upper - lower + 1) + (int)lower));

}

else {

numberInputFieldsN[i][j]->setText(QString::number((double) rand() / RAND\_MAX \* (upper - lower + 1) + lower));

}

}

}

}

void MainWindow::on\_Save\_clicked()

{

ofstream localfile("saved.txt");

QLabel \*Det = this->findChild<QLabel\*>("Det");

QString det = (Det->text());

localfile << dimension << " \n";

localfile << det.toStdString() << " \n";

for (int i = 0; i < dimension; ++i)

{

for (int j = 0; j < dimension; ++j)

{

localfile << numberInputFieldsN[j][i]->text().toDouble() << " ";

}

localfile << " \n";

}

localfile << "\n";

for (int i = 0; i < dimension; ++i)

{

for (int j = 0; j < dimension; ++j)

{

localfile << numberInputFieldsI[j][i]->text().toDouble() << " ";

}

localfile << " \n";

}

}

// Matrices

void Matrices::fillMatrix(double\*\* matrix)

{

for (int i = 0; i < dimension; ++i)

{

for (int j = 0; j < dimension; ++j)

{

matrix[i][j] = numberInputFieldsN[i][j]->text().toDouble();

string str = numberInputFieldsN[i][j]->text().toStdString();

for (char c : str) {

if (!isdigit(c) && c != '.' && c != '-') {

matrix[i][j] = numeric\_limits<double>::quiet\_NaN();

}

}

}

}

}

void Matrices::showmatrix(int amount)

{

this->hideall();

int startX = 236;

int startY = 26;

double position, size, fontsize;

switch (amount) {

case 10:

position = 33;

size = 30;

fontsize = 8;

break;

case 9:

position = 36.8;

size = 33.8;

fontsize = 9.5;

break;

case 8:

position = 41.5;

size = 37.5;

fontsize = 11;

break;

case 7:

position = 47.4;

size = 43.4;

fontsize = 12;

break;

case 6:

position = 55.5;

size = 50;

fontsize = 13;

break;

case 5:

position = 66.5;

size = 62;

fontsize = 14;

break;

case 4:

position = 83;

size = 78;

fontsize = 15;

break;

case 3:

position = 112;

size = 104;

fontsize = 17;

break;

case 2:

position = 167;

size = 161;

fontsize = 18;

break;

}

QFont font("Arial", fontsize);

for (int i = 0; i < amount; ++i)

{

numberInputFieldsN.resize(10);

numberInputFieldsI.resize(10);

for (int j = 0; j < amount; ++j)

{

numberInputFieldsN[i].resize(10);

numberInputFieldsI[i].resize(10);

numberInputFieldsN[i][j]->setVisible(true);

numberInputFieldsI[i][j]->setVisible(true);

int x = startX + i \* position;

int y = startY + j \* position;

numberInputFieldsN[i][j]->setGeometry(x, y, size, size);

numberInputFieldsI[i][j]->setGeometry(x+360, y, size, size);

numberInputFieldsN[i][j]->setFont(font);

numberInputFieldsI[i][j]->setFont(font);

}

}

}

void Matrices::hideall()

{

for (int i = 0; i < 10; ++i)

{

for (int j = 0; j < 10; ++j)

{

numberInputFieldsN[i][j]->setVisible(false);

numberInputFieldsI[i][j]->setVisible(false);

}

}

}

**squarematrix.cpp**

#include "squarematrix.h"

// Constructors and destructors

SquareMatrix::SquareMatrix(int d) : dimension(d)

{

matrix = new double\*[dimension];

for (int i = 0; i < dimension; i++) {

matrix[i] = new double[dimension];

}

}

SquareMatrix::~SquareMatrix()

{

//

for (int i = 0; i < dimension; i++) {

delete[] matrix[i];

}

delete[] matrix;

}

// Matrix setter

void SquareMatrix::setmatrix(double\*\* temp)

{

for (int i = 0; i < dimension; i++) {

for (int j = 0; j < dimension; j++) {

matrix[i][j] = temp[i][j];

}

}

}

// Matrix Inversion Methods

double\*\* SquareMatrix::LUPd()

{

// LUP

double\*\* L = new double\*[dimension];

double\*\* U = new double\*[dimension];

int\* P = new int[dimension];

// Inverted matrix

double\*\* invMatrix = new double\*[dimension];

// Counter

counter = 0;

for (int i = 0; i < dimension; i++) {

L[i] = new double[dimension];

U[i] = new double[dimension];

invMatrix[i] = new double[dimension];

for (int j = 0; j < dimension; j++) {

L[i][j] = 0.0;

U[i][j] = matrix[i][j];

invMatrix[i][j] = 0.0;

}

}

for (int i = 0; i < dimension; i++) {

P[i] = i;

}

for (int k = 0; k < dimension; k++) {

int pivotRow = k;

double pivot = abs(U[k][k]);

for (int i = k + 1; i < dimension; i++) {

if (abs(U[i][k]) > pivot) {

pivotRow = i;

pivot = abs(U[i][k]);

counter++;

}

}

swap(P[k], P[pivotRow]);

swap(U[k], U[pivotRow]);

swap(L[k], L[pivotRow]);

for (int i = k + 1; i < dimension; i++) {

L[i][k] = U[i][k] / U[k][k];

for (int j = k + 1; j < dimension; j++) {

U[i][j] -= L[i][k] \* U[k][j];

counter++;

}

}

}

for (int i = 0; i < dimension; i++) {

double\* e = new double[dimension];

for (int j = 0; j < dimension; j++) {

e[j] = 0.0;

}

e[i] = 1.0;

double\* y = new double[dimension];

double\* x = new double[dimension];

for (int j = 0; j < dimension; j++) {

y[j] = e[P[j]];

for (int k = 0; k < j; k++) {

y[j] -= L[j][k] \* y[k];

counter += 1;

}

}

for (int j = dimension - 1; j >= 0; j--) {

x[j] = y[j];

for (int k = j + 1; k < dimension; k++) {

x[j] -= U[j][k] \* x[k];

counter += 1;

}

x[j] /= U[j][j];

}

for (int j = 0; j < dimension; j++) {

invMatrix[j][i] = x[j];

}

delete[] e;

delete[] y;

delete[] x;

}

// Free memory

for (int i = 0; i < dimension; i++) {

delete[] L[i];

delete[] U[i];

}

delete[] L;

delete[] U;

delete[] P;

return invMatrix;

}

double\*\* SquareMatrix::JGMethod() {

double ratio;

int i, j, k;

counter = 0;

// Inverted matrix

double\*\* invmatrix = new double\*[dimension];

for (i = 0; i < dimension; i++) {

invmatrix[i] = new double[dimension];

}

for (i = 0; i < dimension; i++) {

for (j = 0; j < dimension; j++) {

if (i == j) {

invmatrix[i][j] = 1;

}

else {

invmatrix[i][j] = 0;

}

}

}

// Jordan-Gauss Elimination

for (i = 0; i < dimension; i++) {

for (j = 0; j < dimension; j++) {

if (i != j) {

ratio = matrix[j][i] / matrix[i][i];

if (ratio != 1) {

for (k = 0; k < dimension; k++) {

invmatrix[j][k] = invmatrix[j][k] - ratio \* invmatrix[i][k];

matrix[j][k] = matrix[j][k] - ratio\*matrix[i][k];

counter++;

}

}

}

}

}

// Row Operation

for (i = 0; i < dimension; i++) {

for (j = 0; j < dimension; j++) {

invmatrix[i][j] = invmatrix[i][j] / matrix[i][i];

counter++;

}

}

// Free memory

for (i = 0; i < dimension; i++) {

delete[] matrix[i];

}

delete[] matrix;

return invmatrix;

}

// Determinant methods

void SquareMatrix::findDet()

{

// Gauss method

double \*\*tempMatrix = new double\*[dimension];

for (int i = 0; i < dimension; i++) {

tempMatrix[i] = new double[dimension];

for (int j = 0; j < dimension; j++) {

tempMatrix[i][j] = matrix[i][j];

}

}

for (int i = 0; i < dimension - 1; i++) {

for (int j = i + 1; j < dimension; j++) {

double factor = tempMatrix[j][i] / tempMatrix[i][i];

for (int k = i; k < dimension; k++) {

tempMatrix[j][k] -= factor \* tempMatrix[i][k];

}

}

}

determinant = 1;

for (int i = 0; i < dimension; i++) {

determinant \*= tempMatrix[i][i];

}

// Free memory

for (int i = 0; i < dimension; i++) {

delete[] tempMatrix[i];

}

delete[] tempMatrix;

}

double SquareMatrix::GetDet()

{ return determinant; }

// Steps getter

unsigned int SquareMatrix::getCounter()

{

return counter;

}

// Reversibility test

bool SquareMatrix::isInversable(double\*\* inversed) {

for (int i = 0; i < dimension; i++) {

for (int j = 0; j < dimension; j++) {

if (isnan(inversed[i][j])) return false;

}

}

return true;

}

**startwindow.cpp**

#include "startwindow.h"

// Constructors and destructors

StartWindow::StartWindow(QWidget \*parent) : QMainWindow(parent), ui(new Ui::StartWindow) {ui->setupUi(this);}

StartWindow::~StartWindow() { delete ui; }

// Private slots

void StartWindow::on\_RussianButton\_clicked() { emit languageSelected("ru"); }

void StartWindow::on\_UkrainianButton\_clicked() { emit languageSelected("ua"); }

void StartWindow::on\_BritishButton\_clicked() { emit languageSelected("uk"); }

// open

void StartWindow::open()

{

// QStrings

QString projectPath = QCoreApplication::applicationDirPath();

QString logoPath = projectPath + "/Frog-logo.png";

QString BritishImagePath = projectPath + "/British.png";

QString UkrainianImagePath = projectPath + "/Ukrainian.jpg";

QString RussianImagePath = projectPath + "/Russian.png";

// Images

QImage BritishImage(BritishImagePath);

QImage UkrainianImage(UkrainianImagePath);

QImage RussianImage(RussianImagePath);

// Logotype

QIcon logo(logoPath);

// Window settings

this->setStyleSheet("QMainWindow { border: 5px solid #1f5b21; background-color: #1f7a21 }");

this->setWindowTitle("Matrix Inversion");

this->setWindowIcon(logo);

this->resize(400, 175);

// Language labels

QLabel \*BritishLabel = this->findChild<QLabel\*>("British");

QPixmap pixmapBritish(BritishImagePath);

BritishLabel->setPixmap(pixmapBritish);

QLabel \*UkrainianLabel = this->findChild<QLabel\*>("Ukrainian");

QPixmap pixmapUkrainian(UkrainianImagePath);

UkrainianLabel->setPixmap(pixmapUkrainian);

QLabel \*RussianLabel = this->findChild<QLabel\*>("Russian");

QPixmap pixmapRussian(RussianImagePath);

RussianLabel->setPixmap(pixmapRussian);

// Show the interface

this->show(); }