

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 «Основ програмування»  
(назва дисципліни)  
на тему: Розв'язання систем нелінійних рівнянь

Студента (ки, ів) 1 курсу, групи ІІІ-12  
Кушнір Ганни Вікторівни

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

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2022 рік

# КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрямок "ІПЗ"

Курс 1 Група ІІ-12

Семестр 2

## ЗАВДАННЯ

на курсову роботу студента

**Кушнір Ганни Вікторівни**

(прізвище, ім'я, по батькові)

1. Тема роботи «Розв'язання систем нелінійних рівнянь»

2. Строк здачі студентом закінченої роботи 12.06.2022

3. Вихідні дані до роботи

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

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

6. Дата видачі завдання 22.02.2022

# КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	22.02.2022	
2.	Підготовка ТЗ	12.04.2022	
3.	Пошук та вивчення літератури з питань курсової роботи	18.04.2022	
4.	Розробка сценарію роботи програми	21.04.2022	
6.	Узгодження сценарію роботи програми з керівником	21.04.2022	
5.	Розробка (вибір) алгоритму рішення задачі	26.04.2022	
6.	Узгодження алгоритму з керівником	26.04.2022	
7.	Узгодження з керівником інтерфейсу користувача	26.04.2022	
8.	Розробка програмного забезпечення	17.05.2022	
9.	Налагодження розрахункової частини програми	17.05.2022	
10.	Розробка та налагодження інтерфейсної частини програми	24.05.2022	
11.	Узгодження з керівником набору тестів для контрольного прикладу	24.05.2022	
12.	Тестування програми	31.05.2022	
13.	Підготовка пояснювальної записки	10.06.2022	
14.	Здача курсової роботи на перевірку	12.06.2022	
15.	Захист курсової роботи	16.06.2022	

Студент \_\_\_\_\_  
(підпис)

Керівник \_\_\_\_\_  
(підпис)

\_\_\_\_\_  
Головченко М. М.  
(прізвище, ім'я, по батькові)

« 22 » лютого 2022 р.

## АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 57 сторінок, 29 рисунків, 21 таблиця, 3 посилання.

Об'єкт дослідження: задача пошуку точного розв'язку системи нелінійних рівнянь.

Мета роботи: дослідження методів розв'язання систем нелінійних рівнянь та методів розробки програмного забезпечення.

Вивчено метод розробки програмного забезпечення з використанням принципів об'єктно-орієнтованого програмування. Приведені змістовні постановки задач, їхні індивідуальні математичні моделі, а також описано детальний процес розв'язання кожної з них.

Виконана програмна реалізація алгоритмів методу Якобі та методу Гауса-Зейделя розв'язання систем нелінійних рівнянь.

СИСТЕМА НЕЛІНІЙНИХ РІВНЯНЬ, МЕТОД ЯКОБІ, МЕТОД ГАУСА-ЗЕЙДЕЛЯ, МАТРИЦЯ ЯКОБІ, ІТЕРАЦІЙНІ МЕТОДИ, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ.

## ЗМІСТ

ВСТУП.....	5
1 ПОСТАНОВКА ЗАДАЧІ.....	6
2 ТЕОРЕТИЧНІ ВІДОМОСТІ.....	7
2.1 Метод простих ітерацій (Якобі).....	7
2.2 Метод Гауса-Зейделя.....	9
3 ОПИС АЛГОРИТМІВ .....	10
3.1 Загальний алгоритм .....	10
3.2 Алгоритм методу Якобі.....	12
3.3 Алгоритм методу Гауса-Зейделя .....	13
4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	15
4.1 Діаграма класів програмного забезпечення .....	15
4.2 Опис методів частин програмного забезпечення.....	15
4.2.1 Стандартні методи.....	15
4.2.2 Користувацькі методи .....	19
5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	27
5.1 План тестування.....	27
5.2 Приклади тестування.....	28
6 ІНСТРУКЦІЯ КОРИСТУВАЧА .....	42
6.1 Робота з програмою .....	42
6.2 Формат вхідних та вихідних даних .....	45
6.3 Системні вимоги .....	45
7 АНАЛІЗ РЕЗУЛЬТАТІВ.....	46
ВИСНОВКИ .....	55
ПЕРЕЛІК ПОСИЛАНЬ .....	57
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ .....	58
ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ .....	61

## ВСТУП

Дана курсова робота призначена для вирішення задачі розв'язання систем нелінійних рівнянь наближеними (ітераційними) методами.

Актуальність даної роботи полягає в тому, що в наш час велику роль у розв'язанні математичних задач відіграє точність, а ітераційні методи дозволяють знайти розв'язки СНР з мінімальною похибкою, яку можна корегувати, зменшуючи її до несуттєвого розміру.

Завдання даної курсової роботи зводиться до розроблення програмного забезпечення для розв'язування систем нелінійних рівнянь методами простої ітерації (Якобі) та Гауса-Зейделя з використанням парадигми об'єктно-орієнтованого програмування.

Ціль даної роботи заключається у вивченні та застосуванні на практиці основних концепцій об'єктно-орієнтованого програмування, таких як інкапсуляція, абстрагування, успадкування та поліморфізм.

## 1 ПОСТАНОВКА ЗАДАЧІ

Розробити програмне забезпечення, що буде знаходити рішення для заданої системи нелінійних рівнянь наступними методами:

- а) метод простої ітерації (Якобі);
- б) метод Гауса-Зейделя.

Вхідними даними для даної роботи є система нелінійних рівнянь, яка задана в одному з наступних виглядів (за вибором користувача):

$$\text{а) } \begin{cases} a_{11}x^2 + a_{12}y^2 + a_{13} = 0, \\ a_{21}x^2 + a_{22}y^2 + a_{23} = 0, \end{cases}$$

$$\text{б) } \begin{cases} \sin(x + a_{11}) + a_{12}y + a_{13} = 0, \\ a_{21}x + \cos(y + a_{22}) + a_{23} = 0, \end{cases}$$

$$\text{в) } \begin{cases} e^{a_{11}x} + e^{a_{12}y} + a_{13} = 0, \\ e^{a_{21}x} + e^{a_{22}y} + a_{23} = 0, \end{cases}$$

де  $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$  – матриця коефіцієнтів обраного рівняння, яку задає

користувач.

Програмне забезпечення повинно обробляти матрицю коефіцієнтів для системи нелінійних рівнянь, кількість невідомих яких дорівнює 2.

Вихідними даними для даної роботи являється сукупність дійсних чисел, що є розв'язками даної системи, які виводяться на екран. Програмне забезпечення повинно видавати розв'язок за умови, що для вхідних даних обраний метод підходить. Якщо це не так, то програма повинна вивести відповідне повідомлення. Якщо система не має розв'язків або їх нескінченна кількість, то програма повинна видати відповідне повідомлення.

## 2 ТЕОРЕТИЧНІ ВІДОМОСТІ

Систему  $n$  нелінійних рівнянь з  $n$  невідомими можна представити наступним чином[2]:

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0, \\ f_2(x_1, x_2, \dots, x_n) = 0, \\ \dots \\ f_n(x_1, x_2, \dots, x_n) = 0, \end{cases} \quad (2.1)$$

що еквівалентно такому векторному запису[1]:

$$f(x) = 0,$$

де  $f = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$  – вектор-стовпець функцій,  $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$  – вектор-стовпець

змінних.

### 2.1 Метод простих ітерацій (Якобі)

Сутність методу Якобі полягає в тому, що система рівнянь (2.1) шляхом еквівалентних перетворень приводиться до системи виду[1][2]:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n), \\ x_2 = \varphi_2(x_1, x_2, \dots, x_n), \\ \dots \\ x_n = \varphi_n(x_1, x_2, \dots, x_n), \end{cases} \quad (2.2)$$

або у векторній формі[1]:

$$x = \Phi(x),$$

де  $\Phi(x) = \begin{pmatrix} \varphi_1(x) \\ \varphi_2(x) \\ \dots \\ \varphi_n(x) \end{pmatrix}$ , а  $\varphi_i(x)$  – визначені та неперервні в околі кореня  $x$ .

Далі обирається початкове наближення  $x^{(0)} = (x_1^{(0)}, \dots, x_n^{(0)})$ , де  $(0)$  – номер ітерації.

Тоді наступні наближення отримуються за формулою[2]:



$$\begin{cases} x_1^{(k+1)} = \Phi_1(x_1^{(k)}, \dots, x_n^{(k)}), \\ \vdots \\ x_n^{(k+1)} = \Phi_n(x_1^{(k)}, \dots, x_n^{(k)}), \end{cases} \quad k = 0, 1, \dots \quad (2.3)$$

При цьому ітерації закінчуються, якщо:  $\Delta^{(k+1)} = \max |x_i^{(k+1)} - x_i^{(k)}| \leq \varepsilon$ , де  $\varepsilon$  – задана точність[1].

Достатньою умовою збіжності ітераційного процесу (2.3) є виконання двох умов[2]:

$$\sum_{j=1}^n \left| \frac{\partial \Phi_i}{\partial x_j} \right| < 1, \forall i = \overline{1, n} \quad \text{та} \quad \sum_{i=1}^n \left| \frac{\partial \Phi_i}{\partial x_j} \right| < 1, \forall j = \overline{1, n}$$

Деталізована перша умова:

$$\left| \frac{\partial \Phi_1}{\partial x_1} \right| + \left| \frac{\partial \Phi_1}{\partial x_2} \right| + \dots + \left| \frac{\partial \Phi_1}{\partial x_n} \right| < 1 \text{ при } i = 1,$$

$$\left| \frac{\partial \Phi_n}{\partial x_1} \right| + \left| \frac{\partial \Phi_n}{\partial x_2} \right| + \dots + \left| \frac{\partial \Phi_n}{\partial x_n} \right| < 1 \text{ при } i = n.$$

Деталізована друга умова:

$$\left| \frac{\partial \Phi_1}{\partial x_1} \right| + \left| \frac{\partial \Phi_2}{\partial x_1} \right| + \dots + \left| \frac{\partial \Phi_n}{\partial x_1} \right| < 1 \text{ при } j = 1,$$

$$\left| \frac{\partial \Phi_1}{\partial x_n} \right| + \left| \frac{\partial \Phi_2}{\partial x_n} \right| + \dots + \left| \frac{\partial \Phi_n}{\partial x_n} \right| < 1 \text{ при } j = n.$$

Достатню умову збіжності ітераційного процесу (2.3) можна також описати у словесному вигляді: перша та друга норми матриці Якобі мають бути меншими за одиницю,

де матриця Якобі має вигляд[1]:

$$W(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{pmatrix}$$

## 2.2 Метод Гауса-Зейделя

Основна ідея методу Зейделя полягає в тому, що при обчисленні  $k + 1$ -го наближення невідомої  $x_i$  враховуються вже обчислені раніше  $k + 1$  наближення невідомих  $x_1, x_2, \dots, x_{i-1}$ , тобто виконуються послідовні ітерації. Схема методу Гауса-Зейделя для системи (2.2)[3]:

$$x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}),$$

$$x_2^{(k+1)} = \varphi_2(x_1^{(k+1)}, x_2^{(k)}, \dots, x_n^{(k)}),$$

...

$$x_n^{(k+1)} = \varphi_n(x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{n-1}^{(k+1)}, x_n^{(k)}).$$

Умова закінчення ітерацій[3]:  $\Delta^{(k+1)} = \max_i |x_i^{(k+1)} - x_i^{(k)}| \leq \varepsilon$ , де  $\varepsilon$  – задана точність.

Цей метод має кращу збіжність, ніж метод простих ітерацій, але призводить до більш об'ємних обчислень. Процес Зейделя може бути збіжним навіть у тому випадку, коли процес ітерацій Якобі розбіжний. Можливі також випадки, коли метод Зейделя збігається повільніше процесу простої ітерації, і навіть розбіжний за Зейделем[3].

### 3 ОПИС АЛГОРИТМІВ

Перелік усіх основних змінних та їхні призначення наведено у таблиці 3.1.

Таблиця 3.1 – Основні змінні та їхні призначення

Змінна	Призначення
$\epsilon$	Точність, з якою необхідно обчислити розв'язок системи
A	Двовимірний масив коефіцієнтів рівнянь системи ( $2 \times 3$ )
$x^{(0)}$	Масив початкових наближень ( $1 \times 2$ )
Result	Двовимірний масив результатів розв'язання системи
W	Матриця Якобі
M	Перший норм матриці Якобі
L	Другий норм матриці Якобі
$\Delta$	Різниця між наближеннями на $(k + 1)$ -ій та $k$ -ій ітераціях

#### 3.1 Загальний алгоритм

##### 1. ПОЧАТОК

##### 2. Зчитати точність обчислення результату:

2.1. ЯКЩО точність є додатнім дійсним числом, ТО записати її у змінну  $\epsilon$ , ІНАКШЕ ЯКЩО введена точність менше нуля або дорівнює нулю, ТО видати повідомлення про відповідну помилку та перейти до пункту 11, ІНАКШЕ вивести повідомлення про некоректність введених даних і перейти до пункту 11.

##### 3. Зчитати масив коефіцієнтів рівнянь системи та початкові наближення розв'язку:

##### 3.1. Зчитати масив коефіцієнтів рівнянь системи:

3.1.1. Цикл проходу по всіх рядках таблиці коефіцієнтів системи ( $a_i$  – поточний рядок):

3.1.1.1. Цикл проходу по всіх стовпцях таблиці коефіцієнтів системи ( $a_{ij}$  – поточний елемент):

3.1.1.1.1. ЯКЩО введений елемент є коректним дійсним числом, ТО записати його у відповідну комірку масиву А, ІНАКШЕ видати повідомлення про помилку та перейти до пункту 11.

3.2. Зчитати початкові наближення розв'язку:

3.2.1. Цикл проходу по всіх елементах таблиці початкових наближень ( $a_i$  – поточний елемент):

3.2.1.1. ЯКЩО введений елемент є коректним дійсним числом, ТО записати його у відповідну комірку масиву  $x^{(0)}$ , ІНАКШЕ видати повідомлення про помилку та перейти до пункту 11.

4. ЯКЩО обраний метод Якобі, ТО обробити дані згідно алгоритму методу Якобі (пункт 3.2).
5. ІНАКШЕ ЯКЩО обраний метод Гауса-Зейделя, ТО обробити дані згідно алгоритму методу Гауса-Зейделя (пункт 3.3).
6. ІНАКШЕ ЯКЩО не було обрано жодного методу, вивести повідомлення про помилку та перейти до пункту 11.
7. ЯКЩО система не має розв'язків, ТО:
  - 7.1. Вивести відповідне повідомлення.
  - 7.2. Побудувати та вивести графіки рівнянь системи.
  - 7.3. Перейти до пункту 11.
8. ЯКЩО система має незліченну кількість розв'язків, ТО вивести відповідне повідомлення і перейти до пункту 11.
9. ЯКЩО обраний метод розв'язання сходиться для вхідних даних, ТО:
  - 9.1. Вивести результати ітераційного процесу та кінцевий розв'язок.
  - 9.2. Побудувати та вивести графіки рівнянь системи.
  - 9.3. Перейти до пункту 11.
10. ЯКЩО обраний метод розв'язання розходиться для вхідних даних, ТО:
  - 10.1. Вивести відповідне повідомлення.

10.2. Побудувати та вивести графіки рівнянь системи.

## 11. КІНЕЦЬ

### 3.2 Алгоритм методу Якобі

#### 1. ПОЧАТОК

#### 2. Провести початковий аналіз масиву коефіцієнтів рівнянь системи:

2.1. ЯКЩО за результатами аналізу система має безліч розв'язків, ТО перейти до пункту 6.

2.2. ЯКЩО за результатами аналізу система не має розв'язків, ТО перейти до пункту 6.

2.3. ЯКЩО за результатами аналізу вдалося підібрати простий розв'язок, ТО записати його у масив *Result* і перейти до пункту 6.

#### 3. Перевести систему до виду $x = \Phi(x)$ .

#### 4. Перевірити ітераційний процес на збіжність:

##### 4.1. Цикл проходу по всіх рівняннях системи:

##### 4.1.1. Цикл проходу по кожній змінній даного рівняння:

4.1.1.1. Знайти часткову похідну даного рівняння за даною змінною у точці  $x^{(0)}$ .

4.1.1.2. Записати модуль знайденої похідної у відповідну комірку матриці  $W$ .

4.2. Присвоїти  $M = \max(W_{11} + W_{12}, W_{21} + W_{22})$ .

4.3. Присвоїти  $L = \max(W_{11} + W_{21}, W_{12} + W_{22})$ .

4.4. ЯКЩО значення змінних  $M$  та  $L$  менші за одиницю, ТО ітераційний процес збіжний, ІНАКШЕ ітераційний процес розбіжний.

#### 5. ЯКЩО ітераційний процес розбіжний, ТО перейти до пункту 6, ІНАКШЕ:

5.1. Записати початкові наближення  $x^{(0)}$  у масив *Result*, як нульову ітерацію.

##### 5.2. ВИКОНУВАТИ для $k = 0, 1, 2, \dots$ :

5.2.1. Обчислити наближення  $(k + 1)$ -ої ітерації:

5.2.1.1. Присвоїти  $x_1^{(k+1)} = \Phi_1(x_1^{(k)}, x_2^{(k)})$ .

5.2.1.2. Присвоїти  $x_2^{(k+1)} = \Phi_2(x_1^{(k)}, x_2^{(k)})$ .

5.2.2. Записати наближення  $x^{(k+1)}$  у масив *Result*.

5.2.3. Присвоїти  $\Delta = \max(x_1^{(k+1)} - x_1^{(k)}, x_2^{(k+1)} - x_2^{(k)})$ .

5.2.4. ЯКЩО  $\Delta \leq \varepsilon$ , ТО перейти до пункту 6, ІНАКШЕ перейти до пункту 5.2.

## 6. КІНЕЦЬ

### 3.3 Алгоритм методу Гауса-Зейделя

#### 1. ПОЧАТОК

2. Провести початковий аналіз масиву коефіцієнтів рівнянь системи:

2.1. ЯКЩО за результатами аналізу система має безліч розв'язків, ТО перейти до пункту 6.

2.2. ЯКЩО за результатами аналізу система не має розв'язків, ТО перейти до пункту 6.

2.3. ЯКЩО за результатами аналізу вдалося підібрати простий розв'язок, ТО записати його у масив *Result* і перейти до пункту 6.

3. Перевести систему до виду  $x = \Phi(x)$ .

4. Перевірити ітераційний процес на збіжність:

4.1. Цикл проходу по всіх рівняннях системи:

4.1.1. Цикл проходу по кожній змінній даного рівняння:

4.1.1.1. Знайти часткову похідну даного рівняння за даною змінною у точці  $x^{(0)}$ .

4.1.1.2. Записати модуль знайденої похідної у відповідну комірку матриці  $W$ .

4.2. Присвоїти  $M = \max(W_{11} + W_{12}, W_{21} + W_{22})$ .

4.3. Присвоїти  $L = \max(W_{11} + W_{21}, W_{12} + W_{22})$ .

- 4.4. ЯКЩО значення змінних  $M$  та  $L$  менші за одиницю, ТО ітераційний процес збіжний, ІНАКШЕ ітераційний процес розбіжний.
5. ЯКЩО ітераційний процес розбіжний, ТО перейти до пункту 6, ІНАКШЕ:
- 5.1. Записати початкові наближення  $x^{(0)}$  у масив *Result*, як нульову ітерацію.
- 5.2. ВИКОНУВАТИ для  $k = 0, 1, 2, \dots$  :
- 5.2.1. Обчислити наближення  $(k + 1)$ -ої ітерації:
- 5.2.1.1. Присвоїти  $x_1^{(k+1)} = \Phi_1(x_1^{(k)}, x_2^{(k)})$ .
- 5.2.1.2. Присвоїти  $x_2^{(k+1)} = \Phi_2(x_1^{(k+1)}, x_2^{(k)})$ .
- 5.2.2. Записати наближення  $x^{(k+1)}$  у масив *Result*.
- 5.2.3. Присвоїти  $\Delta = \max(x_1^{(k+1)} - x_1^{(k)}, x_2^{(k+1)} - x_2^{(k)})$ .
- 5.2.4. ЯКЩО  $\Delta \leq \varepsilon$ , ТО перейти до пункту 6, ІНАКШЕ перейти до пункту 5.2.
6. КІНЕЦЬ

## 4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1 Діаграма класів програмного забезпечення

Діаграма класів програмного забезпечення наведена на рисунку 4.1.

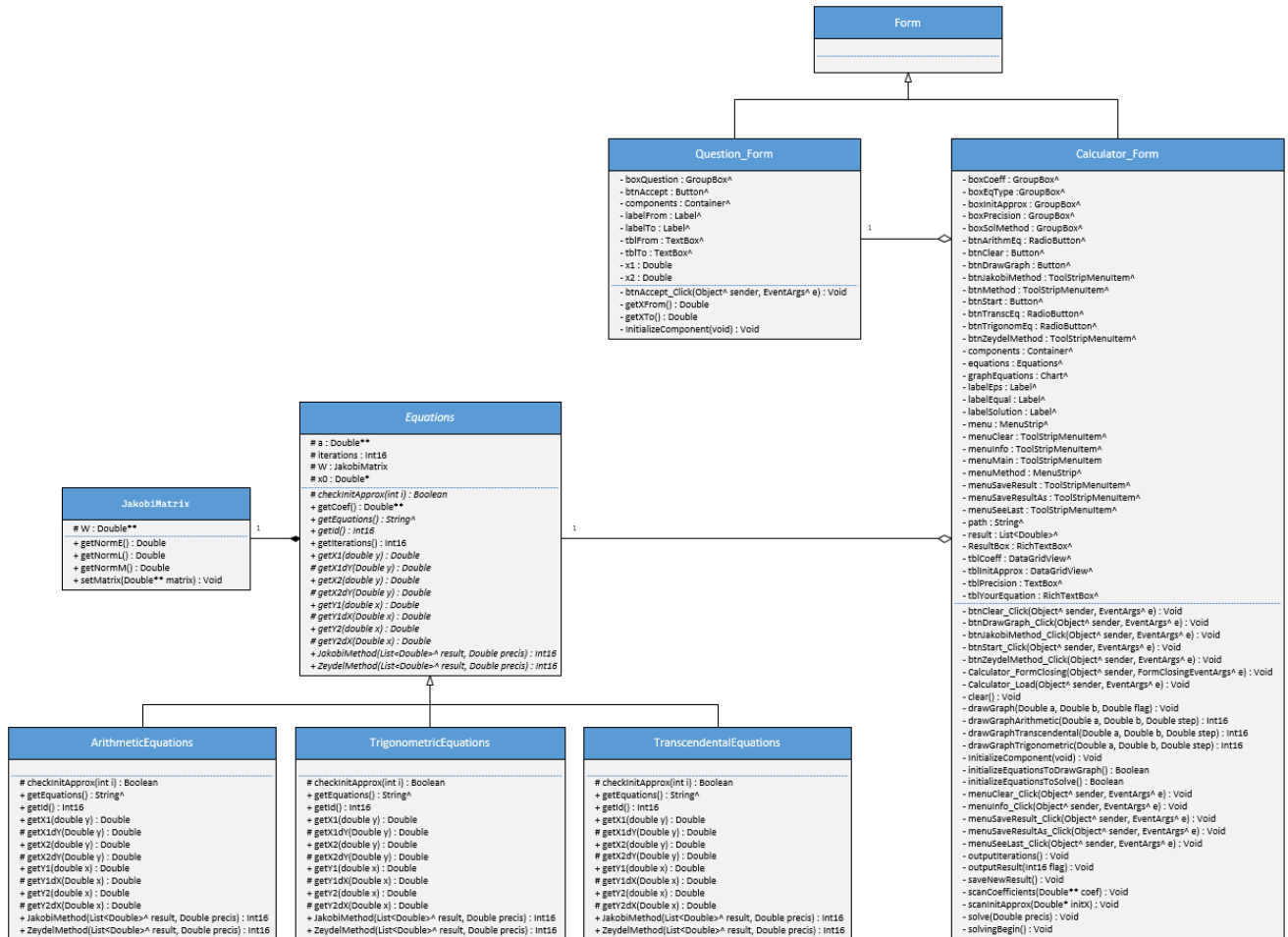


Рисунок 4.1 – Діаграма класів

### 4.2 Опис методів частин програмного забезпечення

#### 4.2.1 Стандартні методи

У таблиці 4.1 наведено стандартні методи, використані при розробці програмного забезпечення.

Таблиця 4.1 – Стандартні методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	Math	Abs	Повертає абсолютне значення заданого дійсного числа.	value – Число типу Double	Число x типу Double, таке що $0 \leq x \leq \text{Double}::\text{MaxValue}$ .



Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
2	Math	Acos	Повертає кут, косинус якого дорівнює заданому числу.	$d$ – Число, що являє собою косинус, причому $-1 \leq d \leq 1$ .	Кут $\theta$ виражений у радіанах, такий що $0 \leq \theta \leq \pi$ , або NaN, якщо $d < -1$ або $d > 1$ або $d = \text{NaN}$ .
3	List<T>	Add	Додає заданий об'єкт у кінець списку List<T>.	item – Об'єкт, який необхідно додати у кінець списку.	–
4	DataPointCollection	AddXY	Додає елемент типу DataPoint у кінець колекції із заданими значеннями X та Y.	xValue – Значення X точки, yValue – Значення Y точки.	Ціле число, що являє собою індекс, куди у колекцію була вставлена дана точка.
5	Math	Asin	Повертає кут, синус якого дорівнює заданому числу.	$d$ – Число, що являє собою синус, причому $-1 \leq d \leq 1$ .	Кут $\theta$ виражений у радіанах, такий що $-\pi/2 \leq \theta \leq \pi/2$ , або NaN, якщо $d < -1$ або $d > 1$ або $d = \text{NaN}$ .
6	DataPoint	Clear	Видаляє всі елементи з колекції Collection<T>.	–	–
7	List<T>	Clear	Видаляє всі елементи зі списку List<T>.	–	–
8	Form	Close	Закриває форму.	–	–
9	Math	Cos	Повертає косинус заданого числа.	$d$ – Кут, виражений у радіанах.	Косинус числа $d$ . Якщо $d$ дорівнює NaN, повертає NaN.
10	Math	Floor	Повертає найбільше ціле число, що не перевищує задане дійсне число.	$d$ – Дійсне число з плаваючою крапкою.	Найбільше ціле число, що не перевищує $d$ . Якщо $d$ дорівнює NaN, повертається NaN.

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
11	–	isfinite	Перевіряє, чи є задане число скінченним.	_X – Число типу double, яке необхідно перевірити.	Значення true, якщо число є скінченним, інакше – false.
12	Math	Log	Повертає натуральний логарифм заданого числа.	d – Число, логарифм якого потрібно знайти.	Натуральний логарифм числа d або нескінченність.
13	Math	Max	Повертає найбільше з двох дійсних чисел з плаваючою крапкою.	val1 – Число типу, val2 – Число типу double.	Більше серед значень val1 і val2. Якщо val1, val2 або вони обидва рівні NaN, повертається NaN.
14	Math	Pow	Повертає задане число, піднесене до заданого степеню.	x – Число, яке підноситься до степеню, y - Число, що являє собойо степінь.	Число x, піднесене до степеню y.
15	Message Box	Show	Виводить вікно повідомлення із заданими текстом, назвою, кнопками та значком для відображення.	text – Текст для виведення, caption – Текст заголовку вікна, buttons – Кнопки, які відображаються у вікні, icon – Значок, який відображається у вікні.	Одне зі значень типу DialogResult.
16	Common Dialog	ShowDi alog	Запускає загальне діалогове вікно із власником за замовчуванням.	–	DialogResult.OK, якщо користувач натиснув OK, інакше - DialogResult.Cancel.
17	Question Form	ShowDi alog	Виводить форму у якості модального діалогового вікна.	–	Одне зі значень типу DialogResult.

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
18	Math	Sin	Повертає синус заданого числа.	d – Кут, виражений у радіанах.	Синус числа d. Якщо d дорівнює нескінченності, повертає NaN.
19	Math	Sqrt	Повертає квадратний корінь заданого числа.	d – Число, корінь якого необхідно знайти.	Нуль або значення квадратного кореня числа d.
20	Process	Start	Запускає процес, специфікуючи ім'я додатка та набір команд, які потрібно виконати.	fileName – Ім'я додатка, який потрібно запустити, arguments – Аргументи командного рядка, які необхідно виконати.	Новий об'єкт класу Process.
21	Convert	ToDouble	Конвертує заданий рядок, що містить число, у еквівалентне дійсне число з плаваючою крапкою.	value – Рядок, що містить число, яке необхідно конвертувати.	Число типу double, яке еквівалентне числовому значенню value, або 0, якщо value = null.
22	Convert	ToString	Конвертує значення заданого дійсного числа у еквівалентне йому рядкове представлення.	value – Число типу double, яке необхідно конвертувати.	Рядкове представлення value.
23	File	WriteAll Text	Створює новий файл, записує заданий рядок у цей файл і закриває його. Якщо файл з вказаним ім'ям уже існує, перезаписує його вміст.	path – Файл, куди потрібно записати, contents – Текст, який потрібно записати.	—

#### 4.2.2 Користувацькі методи

У таблиці 4.2 наведено користувацькі методи, використані при розробці програмного забезпечення.

Таблиця 4.2 – Користувацькі методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
1	Question_Form	btnAccept_Click	Сканує введені користувачем межі для побудови графіків і закриває вікно форми.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Question_Form.h
2	Calculator_Form	btnClear_Click	Викликається при натисканні на кнопку btnClear. Викликає функцію для очищення форми.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calculator_Form.h
3	Calculator_Form	btnDrawGraph_Click	Викликається при натисканні на кнопку btnDrawGraph. Опрацьовує введені користувачем дані та викликає дочірню форму для введення меж побудови графіків, і на основі цих даних будує графіки.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calculator_Form.h
4	Calculator_Form	btnJakobiMethod_Click	Викликається при натисканні на елемент меню btnJakobiMethod. Знімає мітку з методу Зейделя і встановлює її на метод Якобі.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calculator_Form.h
5	Calculator_Form	btnStart_Click	Викликається при натисканні на кнопку btnStart. Опрацьовує введені користувачем дані та, якщо не виявлено помилок, розв'язує обрану систему рівнянь обраним методом.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calculator_Form.h

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Загол овний файл
6	Calcul ator_F orm	btnZeyde lMethod _Click	Викликається при натисканні на елемент меню btnZeydelMethod. Знімає мітку з методу Якобі і встановлює її на метод Зейделя.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calcul ator_F orm.h
7	Calcul ator_F orm	Calculato r_FormC losing	Викликається, коли користувач намагається закрити форму. Запитує у користувача, чи дійсно він бажає вийти, і якщо так - закриває форму.	sender – об'єкт класу Object^, e – об'єкт класу FormClosingEventArgs ^	–	Calcul ator_F orm.h
8	Calcul ator_F orm	Calculato r_Load	Викликається при завантаженні форми. Ініціалізує таблиці та користувацькі компоненти форми.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calcul ator_F orm.h
9	Equati ons	checkInit Approx	Перевіряє ітераційний процес на збіжність при заданому початковому наближенні.	i – Прапорець, який визначає, за якою формулою перетворити початкові рівняння.	true, якщо ітераційн ий процес збіжний.	Equati ons.h
10	Calcul ator_F orm	clear	Очищує форму та відновлює її до початкового стану.	–	–	Calcul ator_F orm.h
11	Calcul ator_F orm	drawGra ph	Будує графіки заданої в класі Calculator_Form системи нелінійних рівнянь на заданому проміжку.	a – Початок проміжку, b – Кінець проміжку, flag – Прапорець, який визначає, яким чином будувати графіки.	–	Calcul ator_F orm.h

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Загол овний файл
12	Calcu lator_F orm	drawGr aphArit hmetic	Будує графіки заданої в формі Calculator_Form системи арифметичних рівнянь на вказаному проміжку з вказаною точністю побудови.	a – Початок проміжку, b – Кінець проміжку, step – Крок побудови.	Прапорець, що визначає, чи вдалося побудувати графіки.	Calcu lator_F orm.h
13	Calcu lator_F orm	drawGr aphTra nscende ntal	Будує графіки заданої в формі Calculator_Form системи трансцендентних рівнянь на вказаному проміжку з вказаною точністю побудови.	a – Початок проміжку, b – Кінець проміжку, step – Крок побудови.	Прапорець, що визначає, чи вдалося побудувати графіки.	Calcu lator_F orm.h
14	Calcu lator_F orm	drawGr aphTrig onomet ric	Будує графіки заданої в формі Calculator_Form системи тригонометричних рівнянь на вказаному проміжку з вказаною точністю побудови.	a – Початок проміжку, b – Кінець проміжку, step – Крок побудови.	Прапорець, що визначає, чи вдалося побудувати графіки.	Calcu lator_F orm.h
15	Equati ons	getCoef	Повертає масив коефіцієнтів рівнянь даної системи.	—	Двовимірний динамічний масив типу Double.	Equati ons.h
16	Equati ons	getEqu ations	Повертає систему рівнянь у вигляді двох рядків, які являють собою загальний вид системи, але з вказаними коефіцієнтами.	—	Значення типу String <sup>^</sup>	Equati ons.h
17	Equati ons	getItera tions	Повертає кількість ітерацій, зроблених при обчисленні точного розв'язку даної системи рівнянь.	—	Значення типу Int16.	Equati ons.h

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголо вний файл
18	Equations	getId	Повертає номер системи у списку видів систем.	–	1, якщо система арифметична; 2, якщо система тригонометрична; 3, якщо система трансцендентна.	Equations.h
19	Jakobi Matrix	getNormL	Повертає другий норм (норм L) матриці Якобі.	–	–	Jakobi Matrix.h
20	Jakobi Matrix	getNormM	Повертає перший норм (норм M) матриці Якобі.	–	–	Jakobi Matrix.h
21	Equations	getX1	Повертає значення X з першого рівняння системи у вказаній точці у.	у – Дійсний аргумент функції.	Значення типу Double.	Equations.h
22	Equations	getX1dY	Повертає значення похідної по Y першого рівняння системи у точці у.	у – Дійсний аргумент функції.	Значення типу Double.	Equations.h
23	Equations	getX2	Повертає значення X з другого рівняння системи у вказаній точці у.	у – Дійсний аргумент функції.	Значення типу Double.	Equations.h
24	Equations	getX2dY	Повертає значення похідної по Y другого рівняння системи у точці у.	у – Дійсний аргумент функції.	Значення типу Double.	Equations.h
25	Question_Form	getXForm	Повертає значення лівої межі побудови графіків.	–	Значення типу Double.	Question_Form.h

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
26	Question_Form	getXTo	Повертає значення правої межі побудови графіків.	—	Значення типу Double.	Question_Form.h
27	Equations	getY1	Повертає значення Y з першого рівняння системи у вказаній точці x.	x – Дійсний аргумент функції.	Значення типу Double.	Equations.h
28	Equations	getY1dX	Повертає значення похідної по X першого рівняння системи у точці x.	x – Дійсний аргумент функції.	Значення типу Double.	Equations.h
29	Equations	getY2	Повертає значення Y з другого рівняння системи у вказаній точці x.	x – Дійсний аргумент функції.	Значення типу Double.	Equations.h
30	Equations	getY2dX	Повертає значення похідної по X другого рівняння системи у точці x.	x – Дійсний аргумент функції.	Значення типу Double.	Equations.h
31	Calculator_Form	initializeEquationsToDrawGraph	Ініціалізує об'єкт системи рівнянь обраного виду уведеними користувачем коефіцієнтами.	—	Значення типу Boolean, яке визначає, чи сталися якісь помилки при ініціалізації.	Calculator_Form.h
32	Calculator_Form	initializeEquationsToSolve	Ініціалізує об'єкт системи рівнянь обраного виду уведеними користувачем коефіцієнтами та початковим наближенням розв'язку.	—	Значення типу Boolean, яке визначає, чи сталися якісь помилки при ініціалізації.	Calculator_Form.h



Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
33	Equations	Jakobi Method	Розв'язує дану систему методом Якобі.	result – Масив для запису результатів ітераційного процесу, precis – Точність, з якою обчислюється результат.	Прапорець, який визначає чи вдалося розв'язати задану систему.	Equations.h
34	Calculator_Form	menuClear_Click	Викликається при натисканні на елемент меню menuClear. Викликає функцію для очищення форми.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calculator_Form.h
35	Calculator_Form	menuInfo_Click	Викликається при натисканні на елемент меню menuInfo. Відкриває вікно повідомлення, яке відображає інформацію про ПЗ.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calculator_Form.h
36	Calculator_Form	menuSaveResult_Click	Викликається при натисканні на елемент меню menuSaveResult. Зберігає результат у новий файл, якщо користувач зберігає результат вперше; інакше - оновлює вже створений файл.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calculator_Form.h
37	Calculator_Form	menuSaveResultAs_Click	Викликається при натисканні на елемент меню menuSaveResultAs. Зберігає результат у новий файл.	sender – об'єкт класу Object^, e – об'єкт класу EventArgs^	–	Calculator_Form.h

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Загол овний файл
38	Calcu ator_F orm	menuSe eLast_ Click	Викликається при натисканні на елемент меню menuSeeLast. Відкриває останній збережений результат.	sender – об’єкт класу Object^, e – об’єкт класу EventArgs^	–	Calcu ator_F orm.h
39	Calcu ator_F orm	outputIt erations	Виводить у текстове поле форми результати ітераційного процесу розв'язання введеної системи, а також виводить кількість цих ітерацій.	–	–	Calcu ator_F orm.h
40	Calcu ator_F orm	output Result	На основі отриманого параметра виводить результат розв'язання системи у форму.	flag – Прапорець, який визначає, чи вдалося розв'язати систему рівнянь.	–	Calcu ator_F orm.h
41	Calcu ator_F orm	saveNe wResul t	Зберігає результат розв'язання системи рівнянь у текстовий файл за допомогою діалогового вікна SaveFileDialog.	–	–	Calcu ator_F orm.h
42	Calcu ator_F orm	scanCo efficien ts	Сканує коефіцієнти рівнянь системи з форми Calculator_Form та записує їх у двовимірний масив.	coef – Двовимірний динамічний масив типу Double, у який зчитуються коефіцієнти.	–	Calcu ator_F orm.h
43	Calcu ator_F orm	scanInit Approx	Сканує початкові наближення з форми Calculator_Form та записує їх у одновимірний масив.	initX – Одновимірний динамічний масив типу Double, у який зчитуються початкові наближення.	–	Calcu ator_F orm.h

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
44	Jakobi Matrix	setMatrix	Формує матрицю Якобі із заданого двовимірного масиву.	matrix – Двовимірний масив типу Double, з якого буде створена матриця Якобі.	–	Jakobi Matrix.h
45	Calculator_Form	solve	У залежності від обраного методу розв'язання, викликає відповідні функції методів, або ж виводить повідомлення про помилку, якщо не було обрано жодного методу.	precis – Точність, з якою потрібно обчислити розв'язок системи.	–	Calculator_Form.h
46	Calculator_Form	solving Begin	"Готує" форму для виведення та подальшої обробки результату. Робить видимими поля розв'язку, графік, а також відкриває можливість зберегти подальший результат у файл. Виводить введені користувачем рівняння у форму.	–	–	Calculator_Form.h
47	Equations	Zeydel Method	Розв'язує дану систему методом Гауса-Зейделя.	result – Масив для запису результатів ітераційного процесу, precis – Точність, з якою обчислюється результат.	Прапорець, який визначає чи вдалося розв'язати задану систему.	Equations.h

## **5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

### **5.1 План тестування**

Тестування програмного забезпечення, яке включає тестування основного функціоналу програми та тестування реакцій на виключні ситуації, буде проведено за наступним планом:

- а) Тестування правильності введених значень.
  - 1) Тестування при введенні некоректних символів.
  - 2) Тестування при введенні недодатнього значення точності.
- б) Тестування коректного запуску розв'язання.
  - 1) Тестування при запуску розв'язання без обраного виду системи.
  - 2) Тестування при запуску розв'язання без обраного методу розв'язання.
- в) Тестування коректної роботи при введенні систем, що не мають коренів.
- г) Тестування коректної роботи при введенні початкових наближень, що призводять до розбіжного ітераційного процесу.
- д) Тестування коректності роботи методів Якобі та Гауса-Зейделя.
  - 1) Перевірка коректності роботи методу Якобі.
  - 2) Перевірка коректності роботи методу Гауса-Зейделя.
- е) Тестування коректності роботи методів Якобі та Гауса-Зейделя з дробовими коефіцієнтами.
  - 1) Перевірка коректності роботи методу Якобі.
  - 2) Перевірка коректності роботи методу Гауса-Зейделя.
- ж) Тестування побудови графіків.
  - 1) Тестування побудови графіків перед розв'язанням системи.
  - 2) Тестування побудови графіків розв'язаної системи.
- з) Тестування збереження результатів у файл.
  - 1) Тестування збереження результатів у новий файл.
  - 2) Тестування оновлення змісту раніше збереженого файлу.
- и) Тестування відкриття останнього збереженого результату.

## 5.2 Приклади тестування

Проведемо тестування програмного забезпечення за визначеним вище планом, фіксуючи при цьому мету тесту, початковий стан програми, вхідні дані, схему проведення тесту, очікуваний результат та стан програми після проведення випробувань у окремій таблиці для кожного тесту.

Результати тестування правильності введених значень наведено у таблицях 5.1 та 5.2. Результати тестування коректного запуску розв'язання наведено у таблицях 5.3 та 5.4. Результат тестування коректної роботи при введенні систем, що не мають коренів, наведено у таблиці 5.5. Результат тестування коректної роботи при введенні початкових наближень, що призводять до розбіжного ітераційного процесу, наведено у таблиці 5.6. Результати тестування коректності роботи методів Якобі та Гауса-Зейделя наведено у таблицях 5.7 та 5.8. Результати тестування коректності роботи методів Якобі та Гауса-Зейделя з дробовими коефіцієнтами наведено у таблицях 5.9 та 5.10. Результати тестування правильності побудови графіків наведено у таблицях 5.11 та 5.12. Результати тестування збереження результатів у файл наведено у таблицях 5.13 та 5.14. Результат тестування відкриття останнього збереженого результату наведено у таблиці 5.15.

Таблиця 5.1 – Приклад роботи програми при введенні некоректних символів

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрите вікно програми
Вхідні дані	Тест 1 - Коефіцієнти: 1 s 4 3 -2 9 Тест 2 - Коефіцієнти: F 1 -8 3d 0 h Тест 3 - Початкові наближення: 1 -k
Схема проведення тесту	Поелементно заповнити таблицю коефіцієнтів та початкових наближень і натиснути кнопку «Розв'язати»
Очікуваний результат	Повідомлення про помилку формату даних

Продовження таблиці 5.1

Стан програми після проведення випробувань	Видано помилку «Не коректний ввід числових даних!»
--	--

Таблиця 5.2 – Приклад роботи програми при введенні недодатнього значення точності

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрите вікно програми
Вхідні дані	Тест 1 - Точність: -0,0001 Тест 2 - Точність: -5 Тест 3 - Точність: 0
Схема проведення тесту	Ввести значення точності і натиснути кнопку «Розв’язати»
Очікуваний результат	Повідомлення про помилку формату даних
Стан програми після проведення випробувань	Видано помилку «Точність має бути додатнім числом!»

Таблиця 5.3 – Приклад роботи програми при запуску розв’язання без обраного виду системи

Мета тесту	Перевірити можливість запуску розв’язання без обраного виду системи
Початковий стан програми	Відкрите вікно програми
Вхідні дані	—
Схема проведення тесту	Натиснути кнопку «Розв’язати» без обраного виду системи
Очікуваний результат	Повідомлення про помилку
Стан програми після проведення випробувань	Видано помилку «Оберіть вид системи!»

Таблиця 5.4 – Приклад роботи програми при запуску розв’язання без обраного методу розв’язання

Мета тесту	Перевірити можливість запуску розв’язання без обраного методу розв’язання
Початковий стан програми	Відкрите вікно програми
Вхідні дані	–
Схема проведення тесту	Натиснути кнопку «Розв’язати» без обраного методу розв’язання системи
Очікуваний результат	Повідомлення про помилку
Стан програми після проведення випробувань	Видано помилку «Оберіть метод розв’язання!»

Таблиця 5.5 – Приклад роботи програми при введенні систем, що не мають коренів

Мета тесту	Перевірити реакцію програми на введення систем, що не мають коренів
Початковий стан програми	Відкрите вікно програми
Вхідні дані	Тест 1 - Вид системи: Арифметична Коефіцієнти: 1 2 3 -1 -4 -2 Тест 2 - Вид системи: Трансцендентна Коефіцієнти: 7 -5 3 4 5 -2 Тест 3 - Вид системи: Тригонометрична Коефіцієнти: 2 0 0 3 2 0
Схема проведення тесту	Обрати вид системи, метод розв’язання, ввести коефіцієнти та натиснути кнопку «Розв’язати»
Очікуваний результат	Повідомлення про відсутність розв’язків рівняння
Стан програми після проведення випробувань	Видано повідомлення «Дана система рівнянь не має розв’язків на множині дійсних чисел.»

Таблиця 5.6 – Приклад роботи програми при введенні початкових наближень, що призводять до розбіжного ітераційного процесу

Мета тесту	Перевірити реакцію програми на введення початкових наближень, що призводять до розбіжного ітераційного процесу
Початковий стан програми	Відкрите вікно програми
Вхідні дані	<p>Тест 1 - Вид системи: Арифметична  Коефіцієнти: 1 1 -2 4 1 -3  Початкові наближення: 10 10  Точність: 0,0001  Метод розв'язання: Якобі</p> <p>Тест 2 - Вид системи: Тригонометрична  Коефіцієнти: 0 1 0 0 1 0  Початкові наближення: -0,6 0,6  Точність: 0,0001  Метод розв'язання: Гауса-Зейделя</p> <p>Тест 3 - Вид системи: Трансцендентна  Коефіцієнти: -3 2 -5 2 1 -3  Початкові наближення: 2 1  Точність: 0,0001  Метод розв'язання: Якобі</p>
Схема проведення тесту	Ввести всі початкові дані та натиснути кнопку «Розв'язати»
Очікуваний результат	Повідомлення про розбіжність ітераційного процесу
Стан програми після проведення випробувань	Видано повідомлення «Введені вами початкові наближення призводять до розбіжного ітераційного процесу, або обраний метод не застосовний до введеної системи.»



Таблиця 5.7 – Приклад роботи методу Якобі

Мета тесту	Перевірити коректність роботи методу Якобі
Початковий стан програми	Відкрите вікно програми
Вхідні дані	<p>Метод розв'язання: Якобі</p> <p>Тест 1 - Вид системи: Арифметична  Коефіцієнти: 2 -2 4 3 -2 -2  Початкові наближення: -2,4 2,8  Точність: 0,00001</p> <p>Тест 2 - Вид системи: Тригонометрична  Коефіцієнти: 2 1 -2 1 2 -4  Початкові наближення: 5 1,3  Точність: 0,0001</p> <p>Тест 3 - Вид системи: Трансцендентна  Коефіцієнти: -3 1 -2 1 1 -4  Початкові наближення: 0,7 0,6  Точність: 0,0000001</p>
Схема проведення тесту	Ввести початкові дані, натиснути кнопку «Розв'язати» і перевірити отримані результати
Очікуваний результат	<p>Тест 1 - <math>x \approx -2,44948</math>, <math>y \approx 2,82842</math></p> <p>Тест 2 - <math>x \approx 4,9759</math>, <math>y \approx 1,3613</math></p> <p>Тест 3 - <math>x \approx 0,7452268</math>, <math>y \approx 0,6382054</math></p>
Стан програми після проведення випробувань	<p>Виведено результати з кожної ітерації та кінцевий розв'язок:</p> <p>Тест 1 - <math>x = -2,44948319493649</math>  <math>y = 2,82841436588093</math> (41 ітерація)</p> <p>Тест 2 - <math>x = 4,97597279239859</math>  <math>y = 1,36128859437952</math> (8 ітерацій)</p> <p>Тест 3 - <math>x = 0,745226838550095</math>  <math>y = 0,638205470523168</math> (16 ітерацій)</p>

Таблиця 5.8 – Приклад роботи методу Гауса-Зейделя

Мета тесту	Перевірити коректність роботи методу Гауса-Зейделя
Початковий стан програми	Відкрите вікно програми
Вхідні дані	<p>Метод розв'язання: Гауса-Зейделя</p> <p>Тест 1 - Вид системи: Арифметична</p> <p>Коефіцієнти: 1 1 -25 2 10 -64</p> <p>Початкові наближення: -4,8 1,3</p> <p>Точність: 0,00001</p> <p>Тест 2 - Вид системи: Тригонометрична</p> <p>Коефіцієнти: -1 4 2 5 5 5</p> <p>Початкові наближення: -1 -0,3</p> <p>Точність: 0,00000001</p> <p>Тест 3 - Вид системи: Трансцендентна</p> <p>Коефіцієнти: 3 3 -7 -2 11 -9</p> <p>Початкові наближення: 0,5 0,2</p> <p>Точність: 0,000001</p>
Схема проведення тесту	Ввести початкові дані, натиснути кнопку «Розв'язати» і перевірити отримані результати
Очікуваний результат	<p>Тест 1 - <math>x \approx -4,82182</math>, <math>y \approx 1,32287</math></p> <p>Тест 2 - <math>x \approx -1,00292589</math>, <math>y \approx -0,27298101</math></p> <p>Тест 3 - <math>x \approx 0,549418</math>, <math>y \approx 0,196318</math></p>
Стан програми після проведення випробувань	<p>Виведено результати з кожної ітерації та кінцевий розв'язок:</p> <p>Тест 1 - <math>x = -4,82182737144332</math></p> <p><math>y = 1,32287420414792</math> (6 ітерацій)</p> <p>Тест 2 - <math>x = -1,00292589626272</math></p> <p><math>y = -0,272981016524025</math> (6 ітерацій)</p> <p>Тест 3 - <math>x = 0,549418238461013</math></p> <p><math>y = 0,196317533724989</math> (4 ітерації)</p>

Таблиця 5.9 – Приклад роботи методу Якобі з дробовими коефіцієнтами

Мета тесту	Перевірити коректність роботи методу Якобі з дробовими коефіцієнтами
Початковий стан програми	Відкрите вікно програми
Вхідні дані	<p>Тест 1 - Вид системи: Арифметична</p> <p>Коефіцієнти: 1,5 -2,3 0,9 8,9 5,8 -9,2</p> <p>Початкові наближення: 0,7 0,8</p> <p>Точність: 0,000001</p> <p>Тест 2 - Вид системи: Тригонометрична</p> <p>Коефіцієнти: -3,7 2,1 -1,2 5,5 3,3 -9,7</p> <p>Початкові наближення: 1,8 1</p> <p>Точність: 0,000000001</p> <p>Тест 3 - Вид системи: Трансцендентна</p> <p>Коефіцієнти: -2,9 2,1 -3,5 7,7 1,9 -8,6</p> <p>Початкові наближення: 0,2 0,5</p> <p>Точність: 0,000000000001</p>
Схема проведення тесту	Ввести початкові дані, натиснути кнопку «Розв’язати» і перевірити отримані результати
Очікуваний результат	<p>Тест 1 - <math>x \approx 0,739223</math>, <math>y \approx 0,864688</math></p> <p>Тест 2 - <math>x \approx 1,832028116</math>, <math>y \approx 1,026746360</math></p> <p>Тест 3 - <math>x \approx 0,230688987216</math>, <math>y \approx 0,521204820532</math></p>
Стан програми після проведення випробувань	<p>Виведено результати з кожної ітерації та кінцевий розв’язок:</p> <p>Тест 1 - <math>x = 0,739223565530716</math></p> <p><math>y = 0,864687980144526</math> (28 ітерацій)</p> <p>Тест 2 - <math>x = 1,83202811599142</math></p> <p><math>y = 1,02674636067932</math> (11 ітерацій)</p> <p>Тест 3 - <math>x = 0,230688987216922</math></p> <p><math>y = 0,521204820532877</math> (15 ітерацій)</p>

Таблиця 5.10 – Приклад роботи методу Зейделя з дробовими коефіцієнтами

Мета тесту	Перевірити коректність роботи методу Гауса-Зейделя з дробовими коефіцієнтами
Початковий стан програми	Відкрите вікно програми
Вхідні дані	<p>Тест 1 - Вид системи: Арифметична</p> <p>Коефіцієнти: -3,7 2,1 -1,2 6,3 5,8 -9,7</p> <p>Початкові наближення: -0,6 -1,2</p> <p>Точність: 0,000001</p> <p>Тест 2 - Вид системи: Тригонометрична</p> <p>Коефіцієнти: 1,2 -2,1 4,3 -9,3 3,5 -8,6</p> <p>Початкові наближення: -0,8 2,2</p> <p>Точність: 0,000000001</p> <p>Тест 3 - Вид системи: Трансцендентна</p> <p>Коефіцієнти: 3,3 1,1 -2,2 -5,5 4,4 -9,9</p> <p>Початкові наближення: -0,2 0,4</p> <p>Точність: 0,00001</p>
Схема проведення тесту	Ввести початкові дані, натиснути кнопку «Розв’язати» і перевірити отримані результати
Очікуваний результат	<p>Тест 1 - <math>x \approx -0,621744</math>, <math>y \approx -1,119161</math></p> <p>Тест 2 - <math>x \approx -0,833921583</math>, <math>y \approx 2,218074505</math></p> <p>Тест 3 - <math>x \approx -0,17650</math>, <math>y \approx 0,45054</math></p>
Стан програми після проведення випробувань	<p>Виведено результати з кожної ітерації та кінцевий розв’язок:</p> <p>Тест 1 - <math>x = -0,621745206124332</math></p> <p><math>y = -1,11916126704308</math> (27 ітерацій)</p> <p>Тест 2 - <math>x = -0,833921583872804</math></p> <p><math>y = 2,21807450516563</math> (6 ітерацій)</p> <p>Тест 3 - <math>x = -0,176500199170395</math></p> <p><math>y = 0,450543169462334</math> (13 ітерацій)</p>

Таблиця 5.11 – Приклад роботи програми при побудові графіків перед розв’язанням системи

Мета тесту	Перевірити правильність побудови графіків окремо від розв’язання системи
Початковий стан програми	Відкрите вікно програми
Вхідні дані	Вид системи: Трансцендентна Коефіцієнти: -3 -7,7 -8,9 1,6 6,4 -8,5 Межі побудови графіків: Від -1 До 3
Схема проведення тесту	Обрати вид системи, ввести коефіцієнти та натиснути кнопку «Побудувати графік для визначення початкових наближень». У вікні, що з’явилося (рис. 5.1), ввести межі побудови графіків та натиснути кнопку «Підтвердити»
Очікуваний результат	Побудовані два графіки експоненти, що перетинаються у точці (1,33; -0,28)
Стан програми після проведення випробувань	Стан програми після побудови графіків наведений на рисунку 5.2

На рисунку 5.1 наведено приклад введення меж побудови графіків рівнянь системи для визначення початкових наближень розв’язку.

Рисунок 5.1 – Приклад введення меж побудови графіків

На рисунку 5.2 наведено стан програми після проведення випробування з таблиці 5.11.

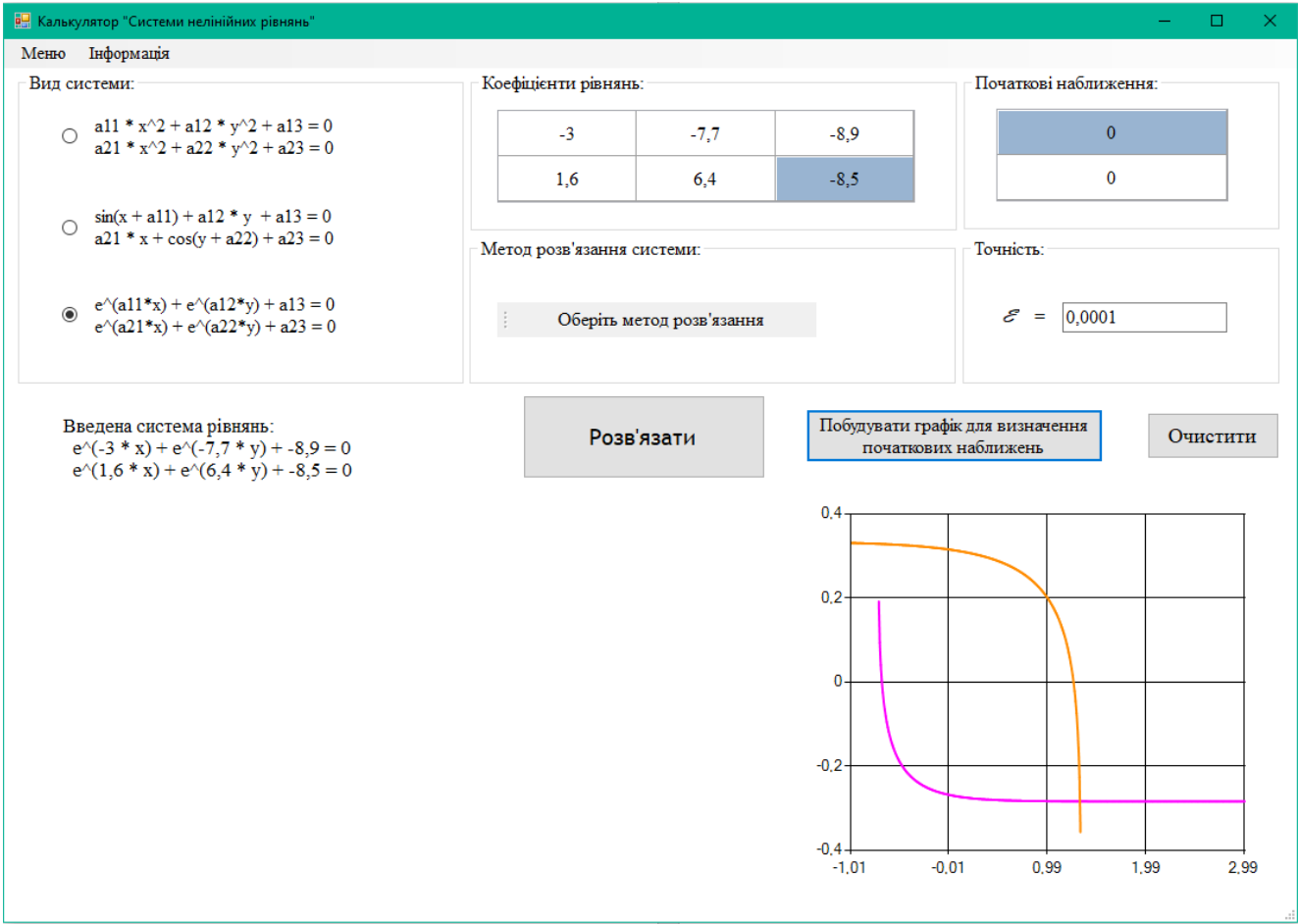


Рисунок 5.2 – Приклад роботи програми при побудові графіків для визначення початкових наближень

Таблиця 5.12 – Приклад роботи програми при побудові графіків під час розв'язання системи

Мета тесту	Перевірити правильність побудови графіків під час розв'язання системи
Початковий стан програми	Відкрите вікно програми
Вхідні дані	Вид системи: Арифметична Коефіцієнти: 1,2 5 -7,4 -2,9 8,6 5,6 Початкові наближення: 1,9 0,5 Точність: 0,0001 Метод розв'язання: Гауса-Зейделя

## Продовження таблиці 5.12

Схема проведення тесту	Ввести початкові дані та натиснути кнопку «Розв'язати»
Очікуваний результат	Побудовані графіки еліпса та гіперболи з центрами у точці (0;0)
Стан програми після проведення випробувань	Стан програми після побудови графіків наведений на рисунку 5.3

На рисунку 5.3 наведено стан програми після проведення випробування з таблиці 5.12.

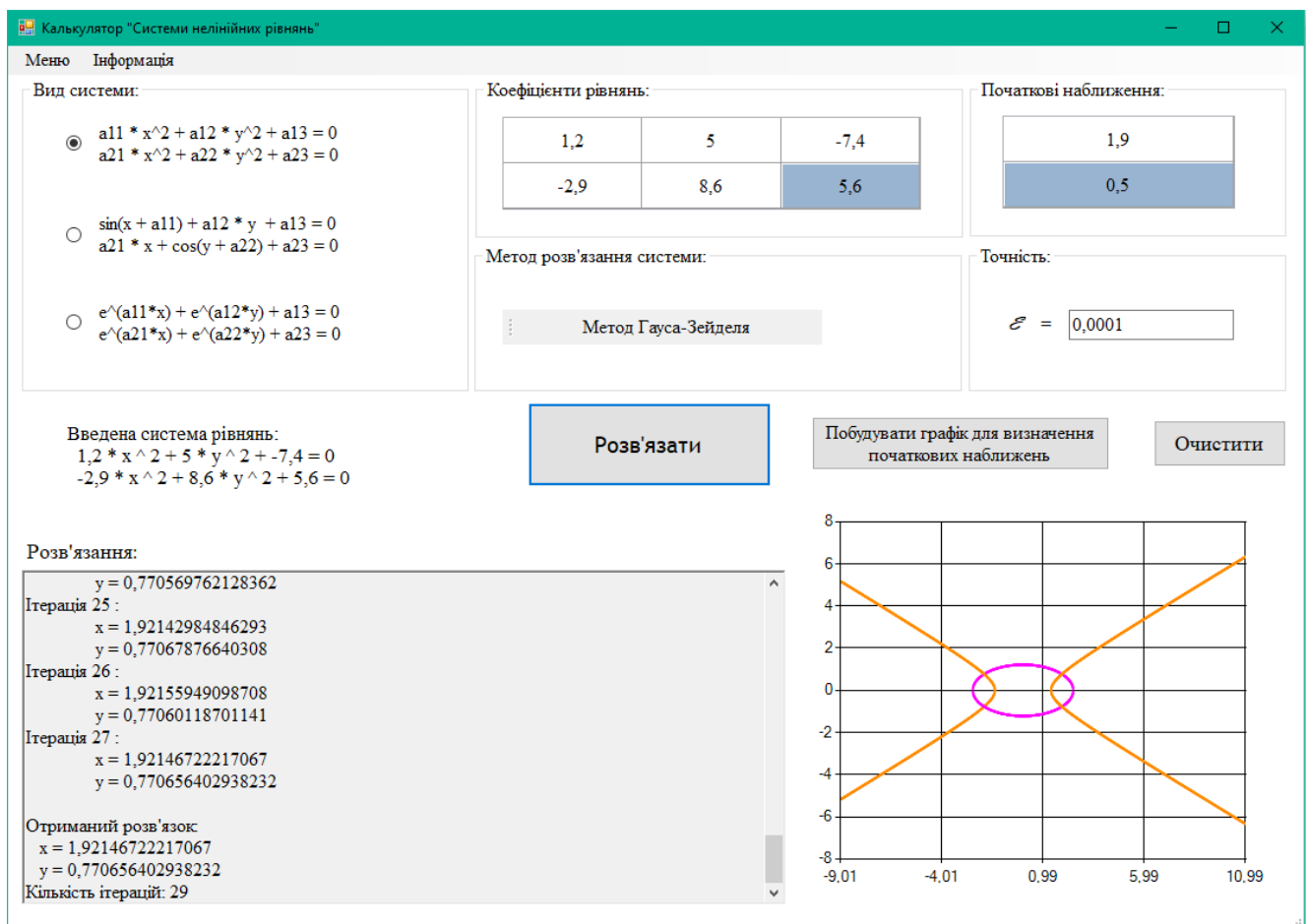


Рисунок 5.3 – Приклад роботи програми при побудові графіків під час розв'язання системи

На рисунку 5.4 наведено масштабований графік з рисунку 5.3. На ньому видно, що точка перетину графіків співпадає з обраним розв'язком системи.

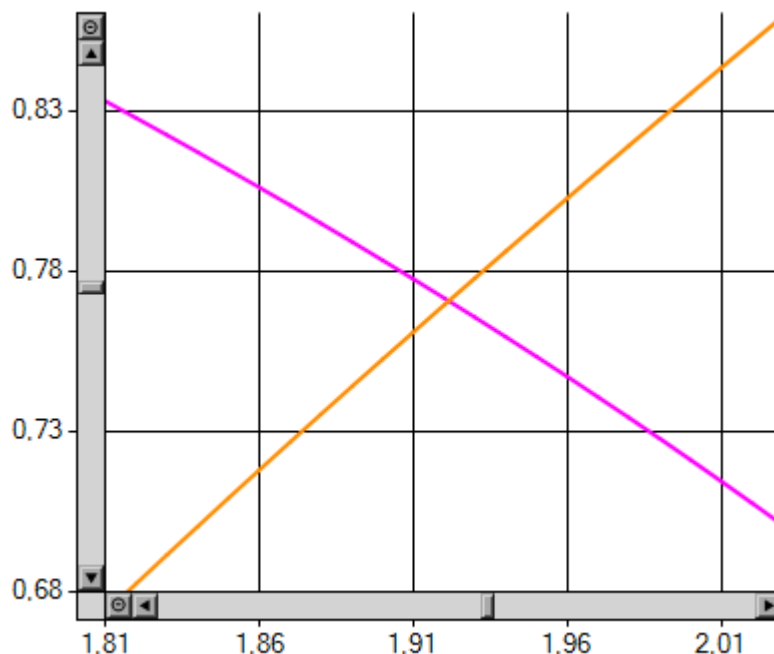


Рисунок 5.4 – Масштабований графік з рисунку 5.3

Таблиця 5.13 – Приклад роботи програми при збереженні результатів у файл

Мета тесту	Перевірити можливість збереження результату розв'язання системи у файл
Початковий стан програми	У спеціально відведене текстове поле вікна виведено результати розв'язання системи рівнянь
Вхідні дані	Введена система рівнянь та виведений результат її розв'язання (рис. 5.3)
Схема проведення тесту	Натиснути на «Меню» → «Зберегти результат як...». У вікні збереження обрати шлях до папки, де буде збережений файл, ввести ім'я файлу і натиснути кнопку «Зберегти» (рис. 5.5)
Очікуваний результат	У вказаному місці буде збережено файл з вказаним ім'ям. Вмістом файлу є введена система рівнянь та результат її розв'язання
Стан програми після проведення випробувань	Виведено повідомлення «Ваш результат успішно збережено!»



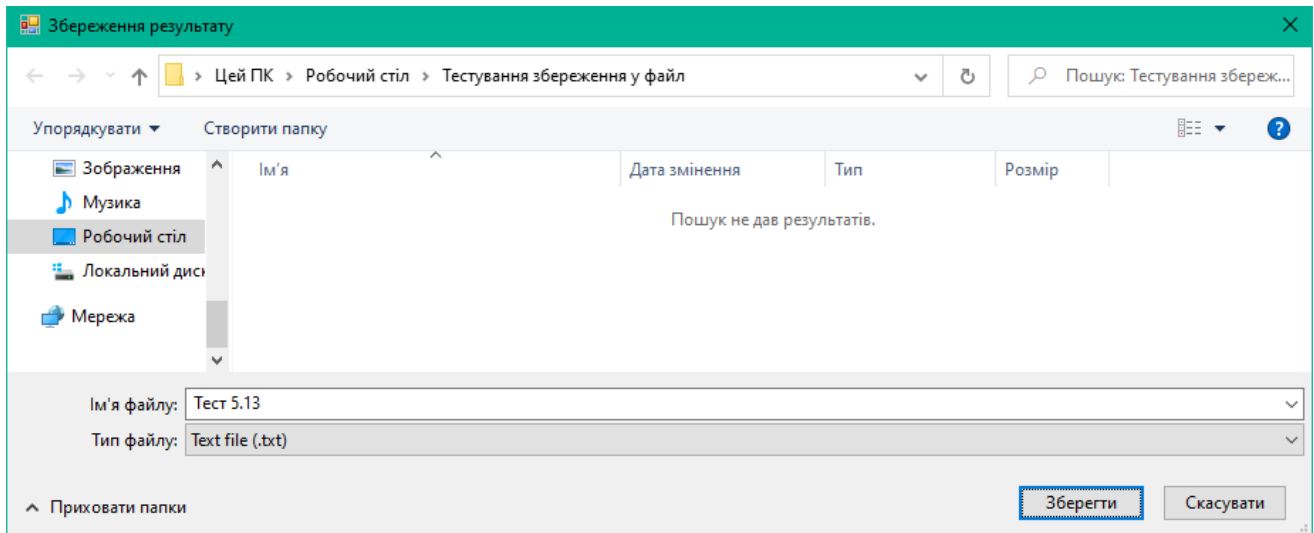


Рисунок 5.5 – Вікно збереження результату розв’язання системи у файл

На рисунках 5.6 та 5.7 наведено вміст збереженого файлу. Очевидно, що даний вміст повністю співпадає з очікуваннями.



Рисунок 5.6 – Початок вмісту збереженого файлу



Рисунок 5.7 – Кінець вмісту збереженого файлу

Таблиця 5.14 – Приклад роботи програми при оновленні вмісту раніше збереженого файлу

Мета тесту	Перевірити можливість оновлення вмісту раніше збереженого файлу новим результатом
Початковий стан програми	У спеціально відведене текстове поле вікна виведено результати розв’язання системи рівнянь і вже було збережено попередній результат
Вхідні дані	Введена нова система рівнянь та виведений результат її розв’язання
Схема проведення тесту	Натиснути на «Меню» → «Зберегти результат»
Очікуваний результат	Раніше збережений до файлу розв’язок буде замінено новим розв’язком
Стан програми після проведення випробувань	Замінено вміст останнього збереженого файлу новим результатом. Змін у програмі не відбулося

Таблиця 5.15 – Приклад роботи програми при відкритті останнього збереженого результату

Мета тесту	Перевірити можливість відкриття останнього збереженого результату через програму
Початковий стан програми	Відкрите вікно програми. Попередньо було розв’язано систему і збережено результат
Вхідні дані	Раніше збережений результат
Схема проведення тесту	Натиснути на «Меню» → «Відкрити останній збережений результат»
Очікуваний результат	Відкрито останній збережений у програмі файл з результатом розв’язку системи
Стан програми після проведення випробувань	У програмі “notepad.exe” відкрито останній збережений результат

## 6 ІНСТРУКЦІЯ КОРИСТУВАЧА

### 6.1 Робота з програмою

Після запуску виконавчого файлу з розширенням \*.exe відкривається привітальне вікно з коротким описом функціоналу програми (рис. 6.1).

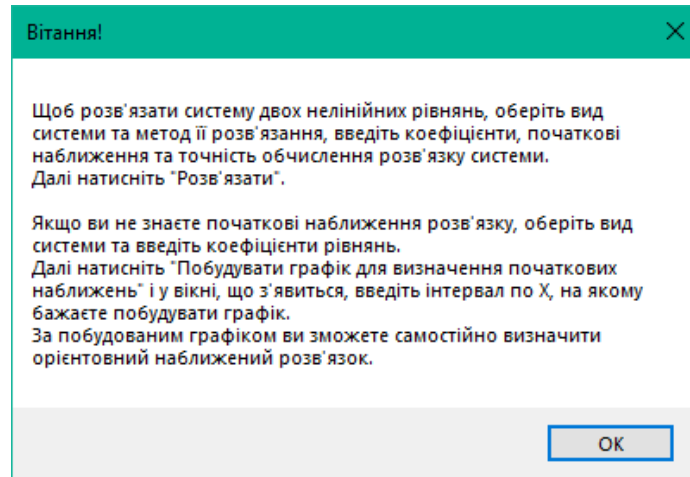


Рисунок 6.1 – Привітальне вікно

Після натиснення «OK» відкривається головне вікно програми (рис. 6.2).

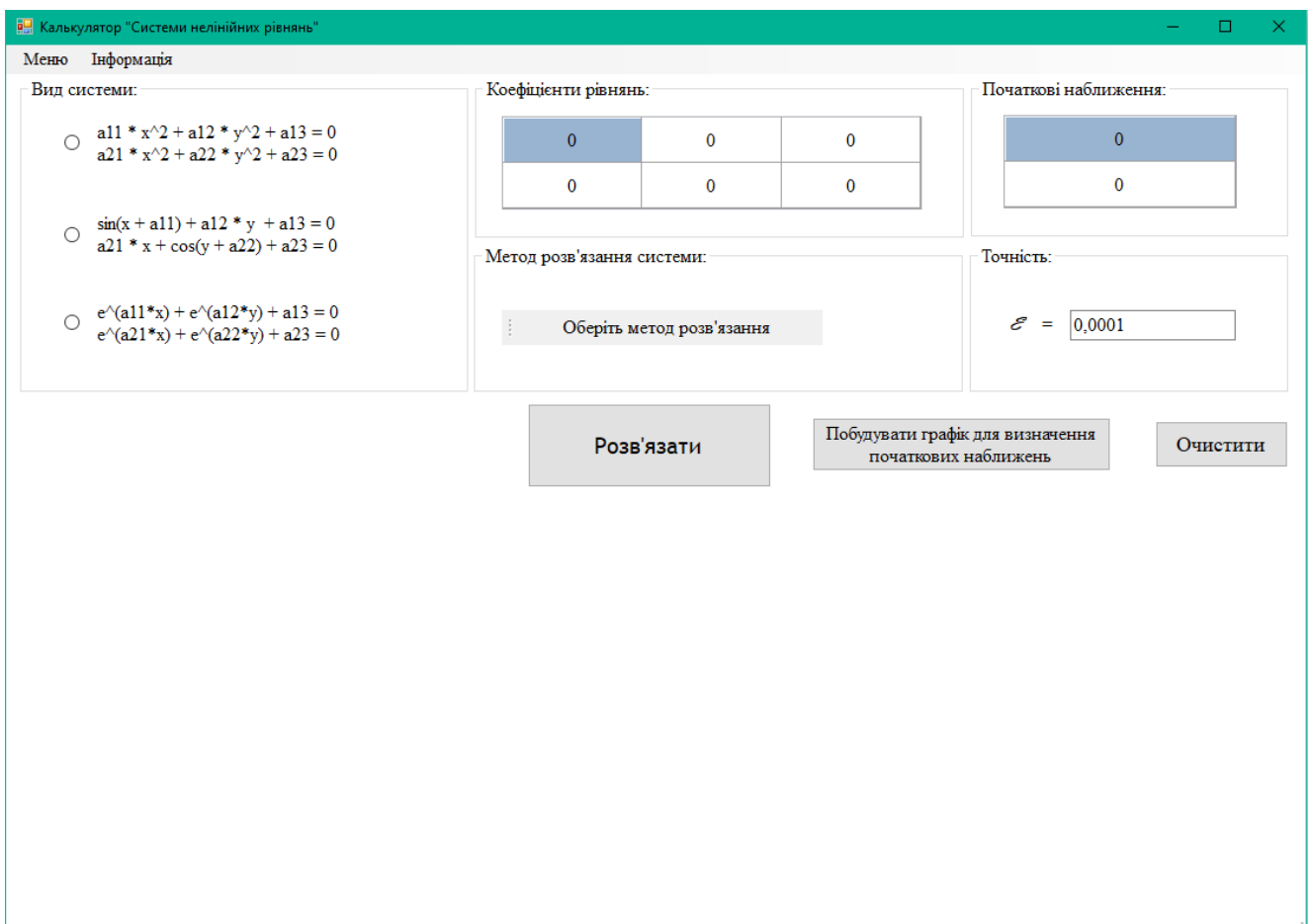


Рисунок 6.2 – Головне вікно програми

Далі, у будь-якому порядку, необхідно заповнити дані про систему рівнянь.

За допомогою радіокнопок, що знаходяться в області під назвою «Вид системи», шляхом натискання на одну з них обирається вид системи, яку необхідно розв'язати (рис. 6.3).

Вид системи:

☐  $a_{11} * x^2 + a_{12} * y^2 + a_{13} = 0$   
 $a_{21} * x^2 + a_{22} * y^2 + a_{23} = 0$

☒  $\sin(x + a_{11}) + a_{12} * y + a_{13} = 0$   
 $a_{21} * x + \cos(y + a_{22}) + a_{23} = 0$

☐  $e^{(a_{11}*x)} + e^{(a_{12}*y)} + a_{13} = 0$   
 $e^{(a_{21}*x)} + e^{(a_{22}*y)} + a_{23} = 0$

Рисунок 6.3 – Вибір виду системи

З використанням випадного списку, що розташований в області під назвою «Метод розв'язання системи», шляхом натискання на один із варіантів списку обирається відповідно метод розв'язання (рис. 6.4).

Метод розв'язання системи:

...  
 Оберіть метод розв'язання  
 Метод простої ітерації (Якобі)  
 Метод Гауса-Зейделя

Рисунок 6.4 – Вибір методу розв'язання системи

У таблиці з назвами «Коефіцієнти рівнянь» та «Початкові наближення» шляхом почергового заповнення полів вводяться з клавіатури відповідно коефіцієнти та початкові наближення шуканого розв'язку системи рівнянь (рис. 6.5). За замовчуванням всі поля таблиць рівні нулю.

Коефіцієнти рівнянь:

1,5	-2	1
6	0	-8,34

Початкові наближення:

1,2
0,7

Рисунок 6.5 – Введення коефіцієнтів та початкових наближень

У текстове поле з назвою «Точність» шляхом введення числа з клавіатури необхідно виставити значення, з точністю до якого потрібно обчислити розв’язок системи (рис. 6.6). За замовчуванням точність рівна 0,0001.

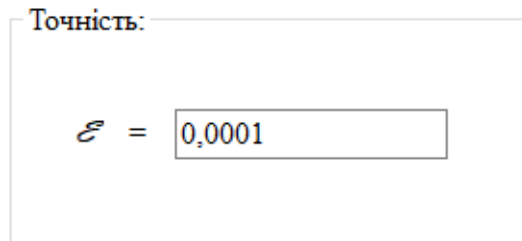


Рисунок 6.6 – Введення точності обчислень

Якщо заздалегідь невідомі початкові наближення розв’язку, є можливість побудувати графіки для їх визначення. Для цього необхідно лише обрати вид системи, ввести коефіцієнти рівнянь та натиснути відповідну кнопку.

Якщо ж ці дані вже визначені, необхідно заповнити всі поля головного вікна програми та натиснути кнопку «Розв’язати».

Щоб очистити форму від усіх введених даних, необхідно натиснути кнопку «Очистити» у головному вікні програми, або послідовно обрати «Меню» → «Очистити» (рис. 6.7), або натиснути сполучення клавіш «Ctrl+0» на клавіатурі.

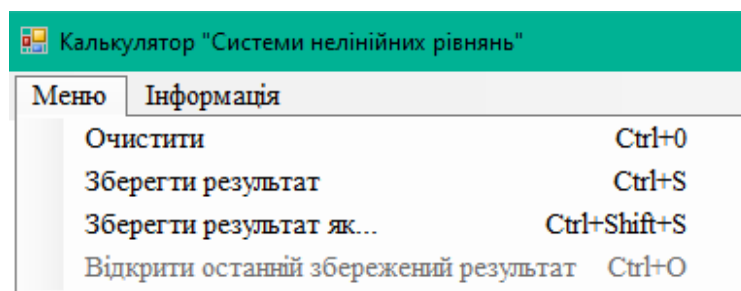


Рисунок 6.7 – Головне меню програми

Для збереження результатів розв’язку системи до файлу треба послідовно обрати «Меню» → «Зберегти результат» або «Меню» → «Зберегти результат як...» (рис. 6.7). Визначені вище пункти меню можна також викликати натисканням сполучень клавіш на клавіатурі «Ctrl+S» або «Ctrl+Shift+S» відповідно.

Щоб відкрити останній збережений результат необхідно послідовно обрати «Меню» → «Відкрити останній збережений результат» (рис. 6.7) або натиснути сполучення клавіш на клавіатурі «Ctrl+O». Дана команда відкриває збережений файл у програмі “notepad.exe”.

## 6.2 Формат вхідних та вихідних даних

Користувачем на вхід програми подається СНР у вигляді таблиці коефіцієнтів, яка має складатися з дійсних чисел, дробова частина яких відділяється від цілої частини комою. Також користувачем вводяться початкові наближення та точність обчислень, які теж є дійсними числами.

Результатом виконання програми є розв'язок даної СНР та отримані наближення розв'язків з кожної ітерації, які подаються без округлення для можливості подальшого аналізу правильності роботи алгоритму, або повідомлення, що дана система не має розв'язків або їх нескінченна кількість, або що ітераційний процес розходиться для обраного методу.

## 6.3 Системні вимоги

Системні вимоги до програмного забезпечення наведені у таблиці 6.1.

Таблиця 6.1 – Системні вимоги до програмного забезпечення

	Мінімальні	Рекомендовані
Операційна система	Windows® XP / Windows 7 / Windows 8 / Windows 10 (з останніми оновленнями)	Windows 8 / Windows 10 (з останніми оновленнями)
Процесор	Intel® Core™ i5 1.0 GHz або AMD Athlon™ 1.0 GHz	Intel® Core™ i5 або AMD Athlon™ 64 X2
Оперативна пам'ять	256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows 7 / Windows 8 / Windows 10)	2 GB RAM
Відеоадаптер	Intel Iris Xe Max Graphics з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог)	
Дисплей	1024x768	1920x1080 або краще
Прилади введення	Клавіатура, комп'ютерна миша	
Додаткове ПЗ	notepad.exe	

## 7 АНАЛІЗ РЕЗУЛЬТАТІВ

Головною задачею курсової роботи була реалізація програми для розв'язання СНР наступними методами: простої ітерації (Якобі) та Гауса-Зейделя.

Критичні ситуації у роботі програми виявлені не були. Під час тестування було з'ясовано, що більшість помилок виникало тоді, коли користувачем вводилися не числові вхідні дані або коли користувач намагався запустити розв'язування системи, не обравши її вид або метод розв'язання. Тому всі дані, які вводить користувач, ретельно перевіряються на валідність і лише потім подаються на обробку програмі.

Для перевірки та доведення достовірності результатів роботи програмного забезпечення буде використано MS Excel:

а) Метод Якобі:

1) Система арифметичних рівнянь.

Результат виконання методу Якобі для системи арифметичних рівнянь наведено на рисунку 7.1.

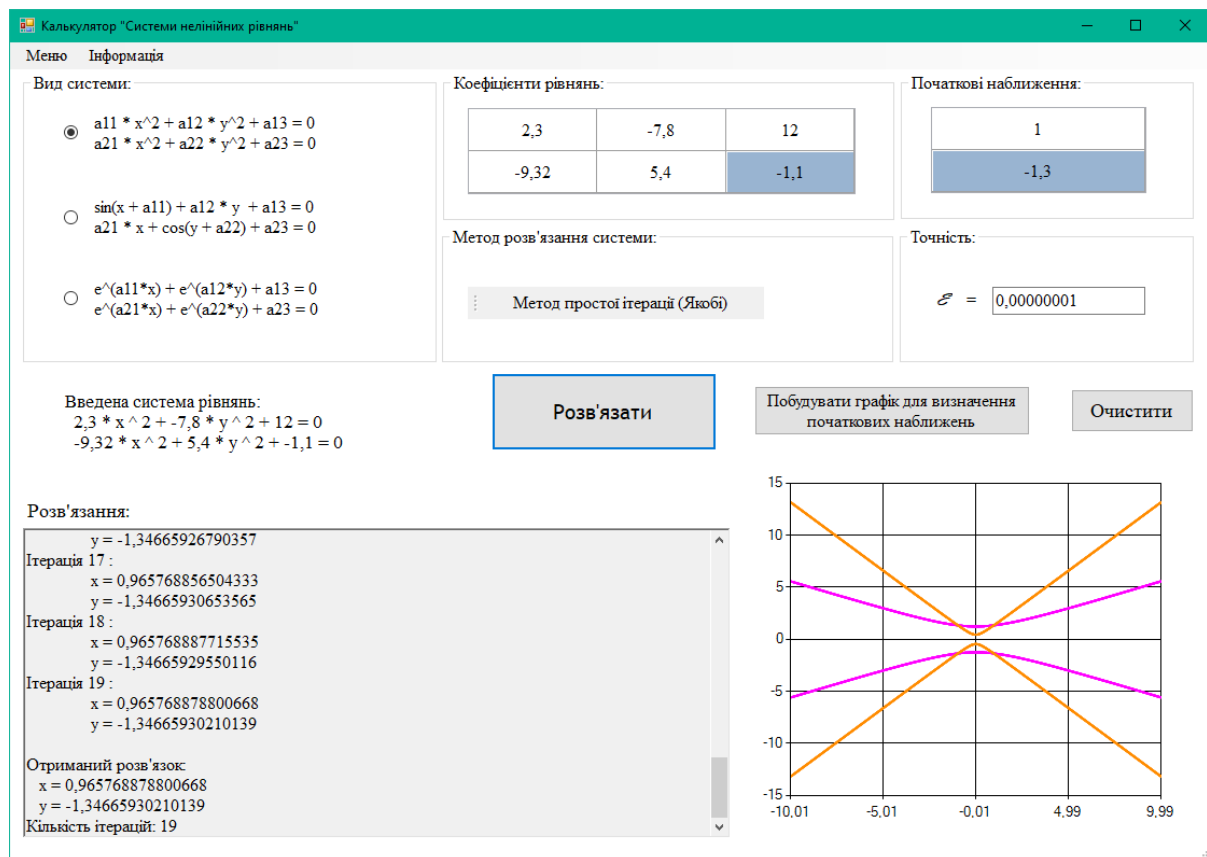
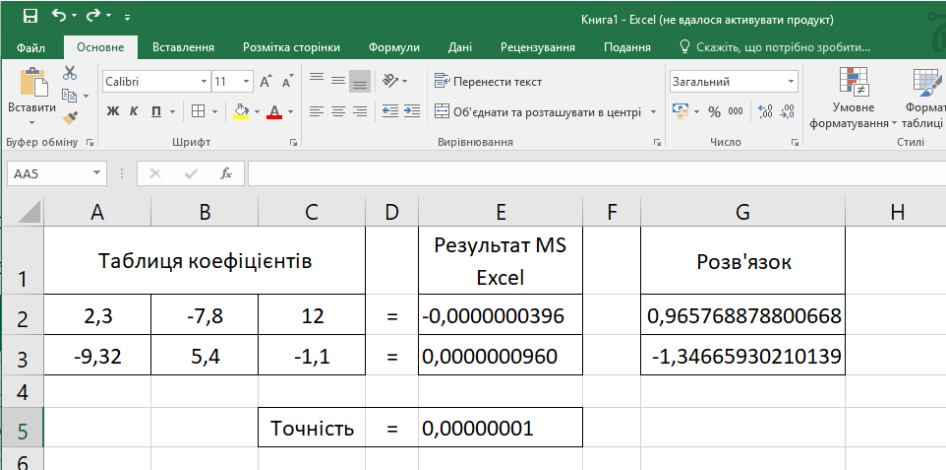


Рисунок 7.1 – Результат роботи методу Якобі для системи арифметичних рівнянь

Оскільки результат виконання збігається з результатом в MS Excel (рис. 7.2), то розроблений алгоритм методу Якобі для системи арифметичних рівнянь працює правильно.



	A	B	C	D	E	F	G	H
1	Таблиця коефіцієнтів				Результат MS Excel		Розв'язок	
2	2,3	-7,8	12	=	-0,0000000396		0,965768878800668	
3	-9,32	5,4	-1,1	=	0,0000000960		-1,34665930210139	
4								
5			Точність	=	0,00000001			
6								

Рисунок 7.2 – Перевірка методу Якобі в MS Excel

## 2) Система тригонометричних рівнянь.

Результат виконання методу Якобі для системи тригонометричних рівнянь наведено на рисунку 7.3.

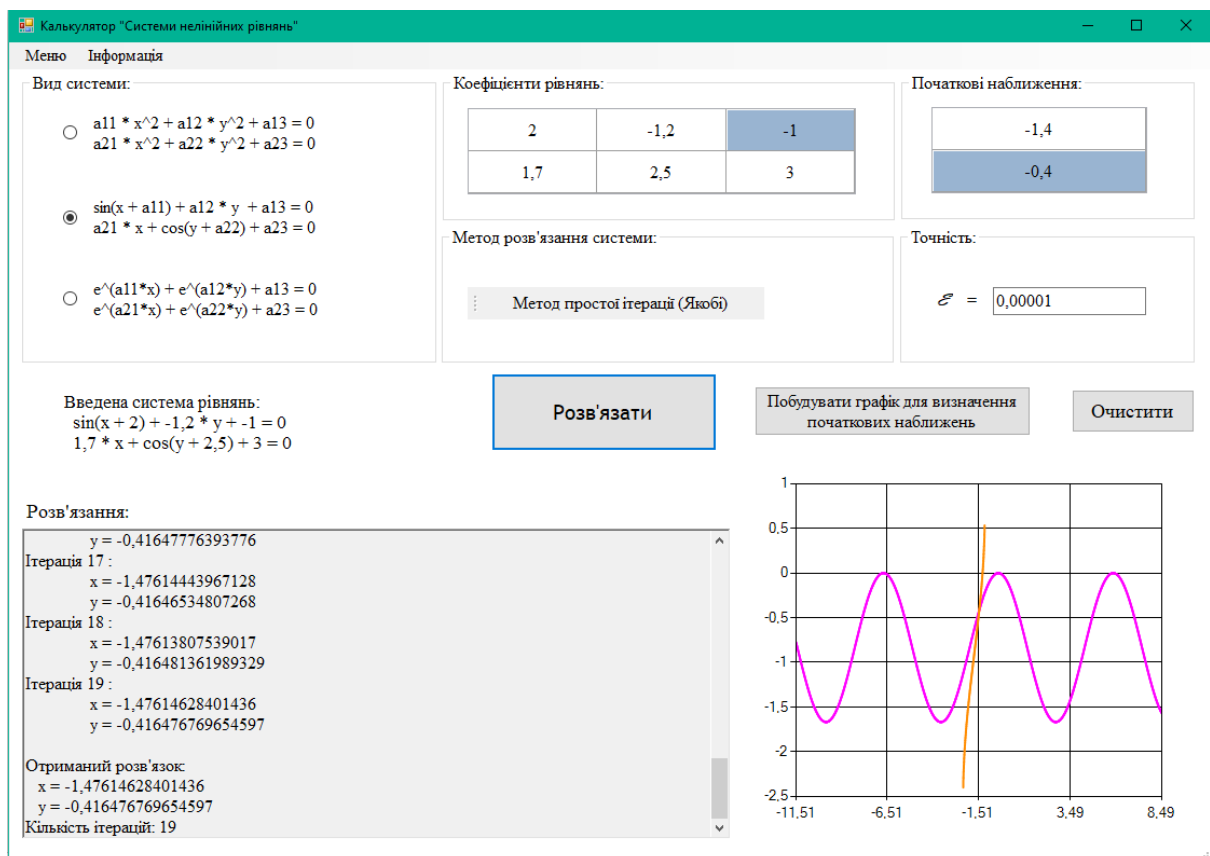
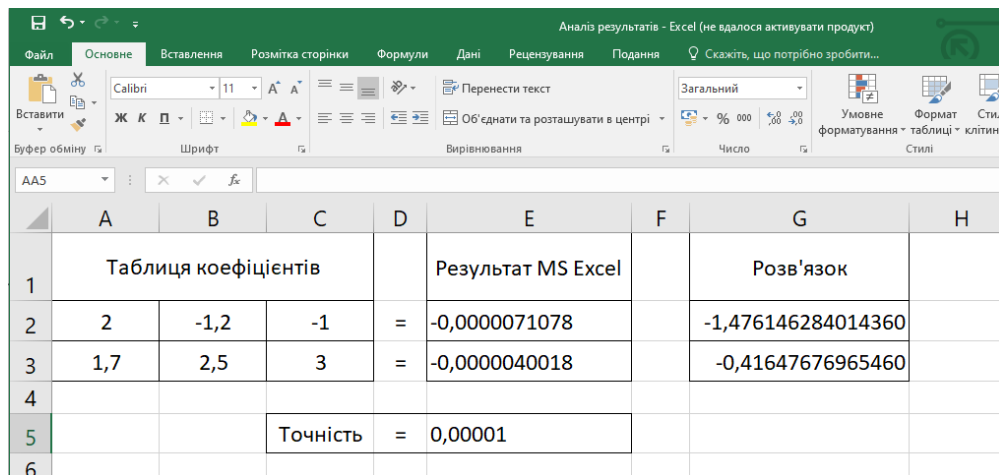


Рисунок 7.3 – Результат роботи методу Якобі для системи тригонометричних рівнянь



Оскільки результат виконання збігається з результатом в MS Excel (рис. 7.4), то розроблений алгоритм методу Якобі для системи тригонометричних рівнянь працює правильно.



	A	B	C	D	E	F	G	H
1	Таблиця коефіцієнтів				Результат MS Excel		Розв'язок	
2	2	-1,2	-1	=	-0,0000071078		-1,476146284014360	
3	1,7	2,5	3	=	-0,0000040018		-0,41647676965460	
4								
5			Точність	=	0,00001			
6								

Рисунок 7.4 – Перевірка методу Якобі в MS Excel

### 3) Система трансцендентних рівнянь.

Результат виконання методу Якобі для системи трансцендентних рівнянь наведено на рисунку 7.5.

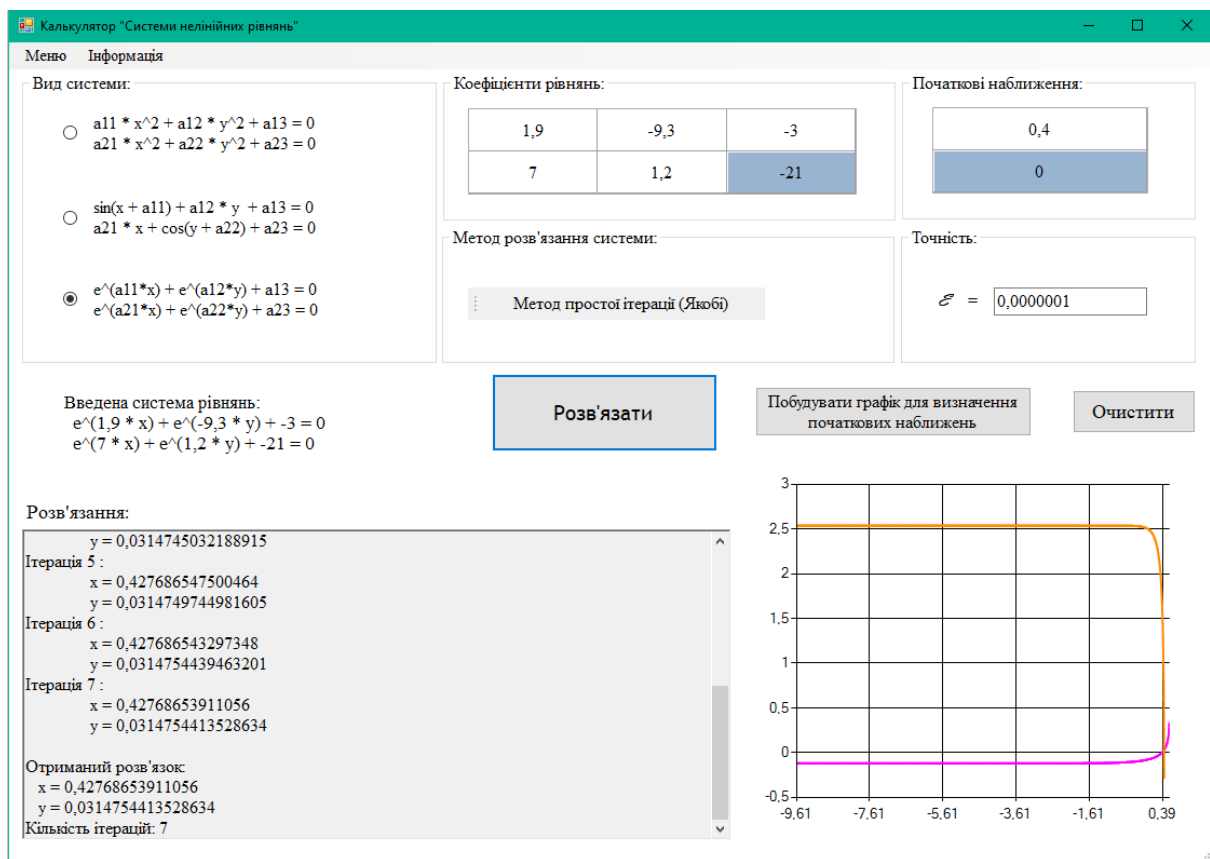
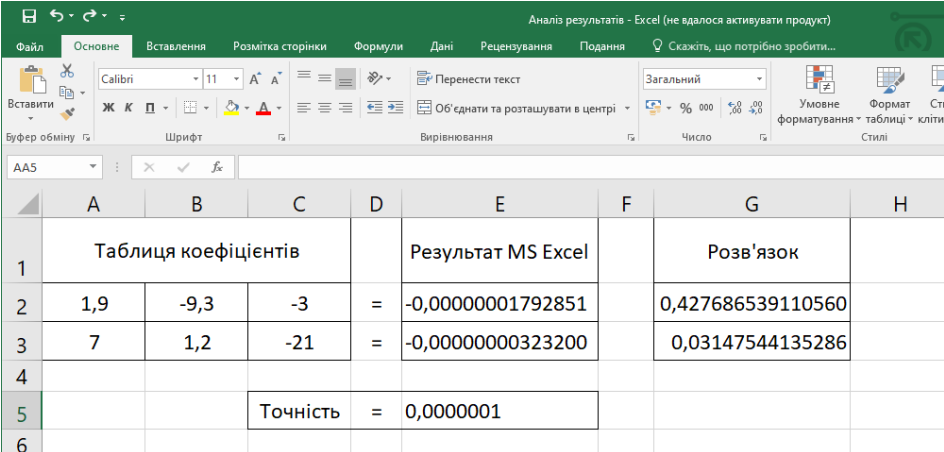


Рисунок 7.5 – Результат роботи методу Якобі для системи трансцендентних рівнянь

Оскільки результат виконання збігається з результатом в MS Excel (рис. 7.6), то розроблений алгоритм методу Якобі для системи трансцендентних рівнянь працює правильно.



	A	B	C	D	E	F	G	H
1	Таблиця коефіцієнтів				Результат MS Excel		Розв'язок	
2	1,9	-9,3	-3	=	-0,00000001792851		0,427686539110560	
3	7	1,2	-21	=	-0,00000000323200		0,03147544135286	
4								
5			Точність	=	0,00000001			
6								

Рисунок 7.6 – Перевірка методу Якобі в MS Excel

#### б) Метод Гауса-Зейделя:

##### 1) Система арифметичних рівнянь.

Результат виконання методу Гауса-Зейделя для системи арифметичних рівнянь наведено на рисунку 7.7.

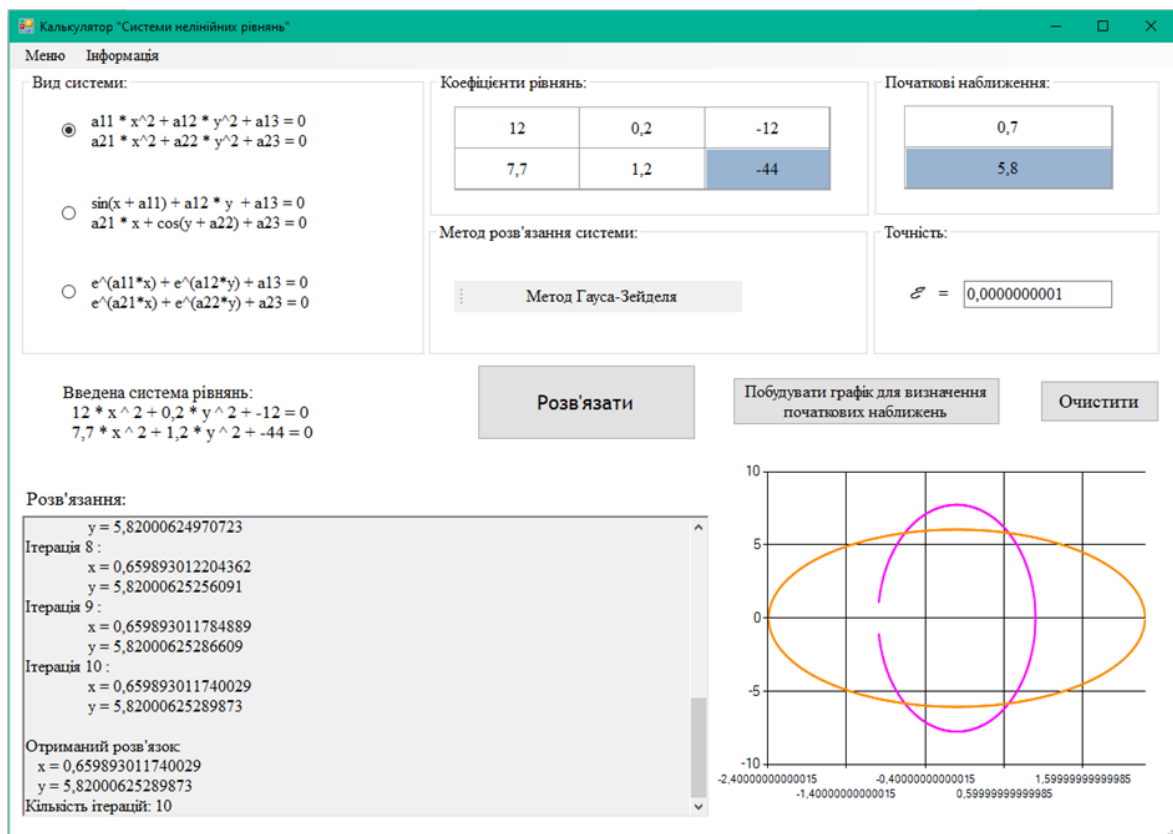


Рисунок 7.7 – Результат роботи методу Гауса-Зейделя для системи арифметичних рівнянь

Оскільки результат виконання збігається з результатом в MS Excel (рис. 7.8), то розроблений алгоритм методу Гауса-Зейделя для системи арифметичних рівнянь працює правильно.

	A	B	C	D	E	F	G	H
1	Таблиця коефіцієнтів				Результат MS Excel		Розв'язок	
2	12	0,2	-12	=	0,000000000076		0,659893011740029	
3	7,7	1,2	-44	=	0,000000000000		5,82000625289873	
4								
5			Точність	=	0,0000000001			
6								

Рисунок 7.8 – Перевірка методу Гауса-Зейделя в MS Excel

## 2) Система тригонометричних рівнянь.

Результат виконання методу Гауса-Зейделя для системи тригонометричних рівнянь наведено на рисунку 7.9.

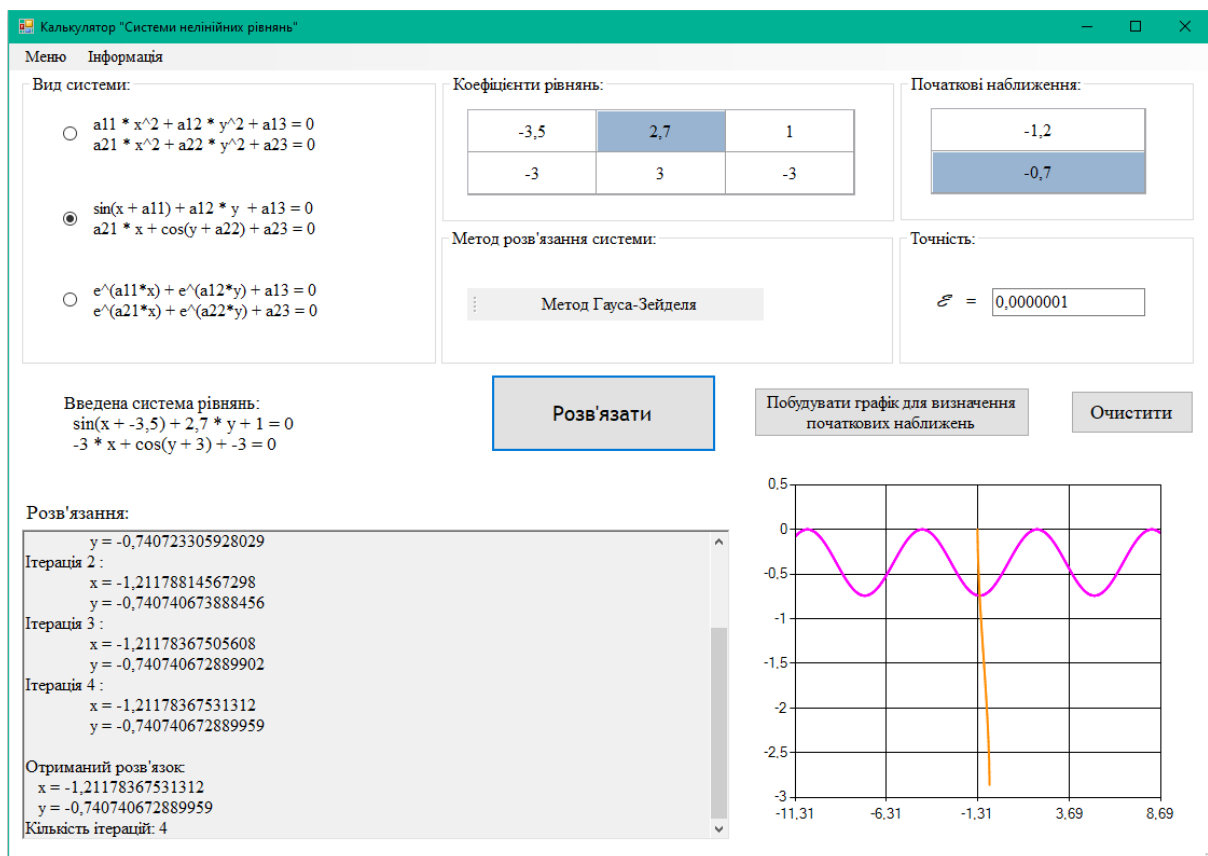


Рисунок 7.9 – Результат роботи методу Гауса-Зейделя для системи тригонометричних рівнянь

Оскільки результат виконання збігається з результатом в MS Excel (рис. 7.10), то розроблений алгоритм методу Гауса-Зейделя для системи тригонометричних рівнянь працює правильно.

	A	B	C	D	E	F	G	H
1	Таблиця коефіцієнтів				Результат MS Excel		Розв'язок	
2	-3,5	2,7	1	=	0,000000000000		-1,211783675313120	
3	-3	3	-3	=	0,000000000000		-0,74074067288996	
4								
5			Точність	=	0,00000001			
6								

Рисунок 7.10 – Перевірка методу Гауса-Зейделя в MS Excel

### 3) Система трансцендентних рівнянь.

Результат виконання методу Гауса-Зейделя для системи трансцендентних рівнянь наведено на рисунку 7.11.

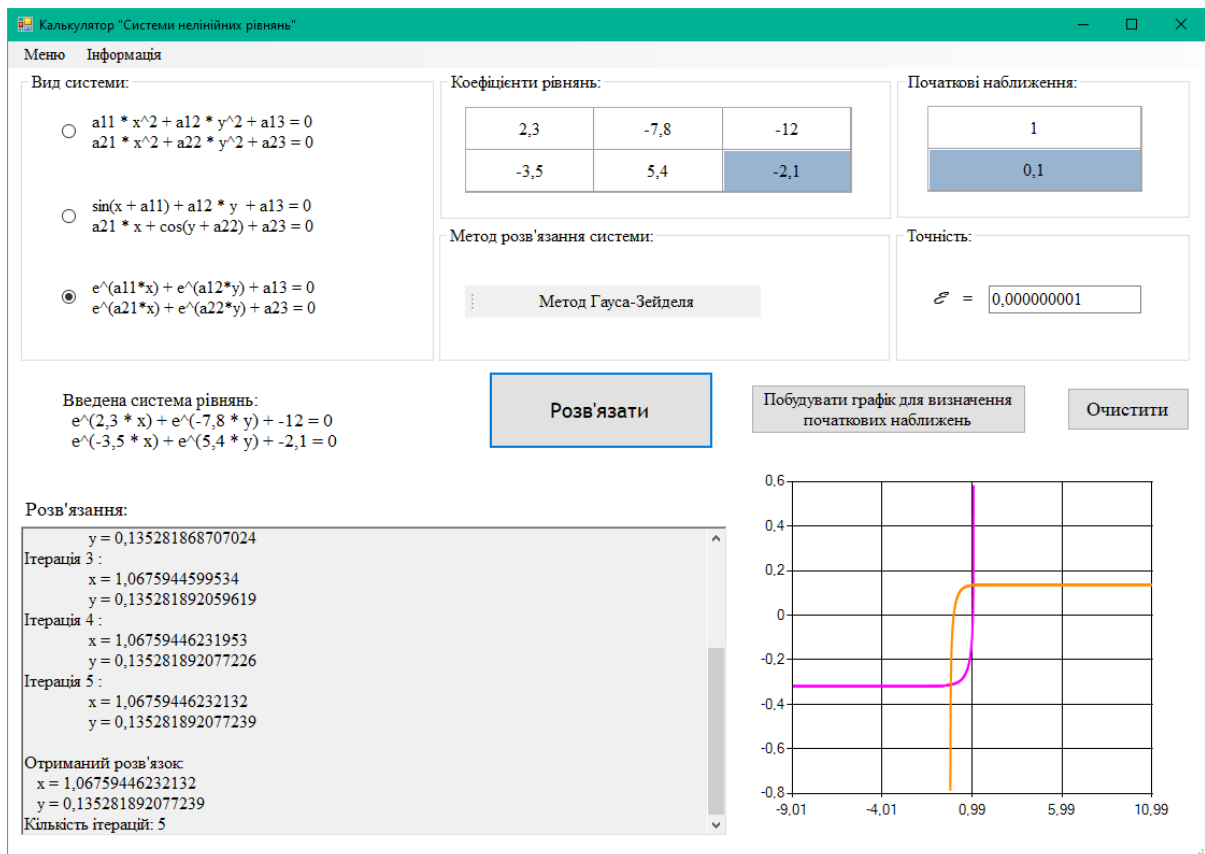


Рисунок 7.11 – Результат роботи методу Гауса-Зейделя для системи трансцендентних рівнянь

Оскільки результат виконання збігається з результатом в MS Excel (рис. 7.12), то розроблений алгоритм методу Гауса-Зейделя для системи трансцендентних рівнянь працює правильно.

	A	B	C	D	E	F	G	H
1	Таблиця коефіцієнтів				Результат MS Excel		Розв'язок	
2	2,3	-7,8	-12	=	0,000000000000005		1,067594462321320	
3	-3,5	5,4	-2,1	=	0,000000000000000		0,13528189207724	
4								
5			Точність	=	0,000000001			
6								

Рисунок 7.12 – Перевірка методу Гауса-Зейделя в MS Excel

Результати тестування ефективності алгоритмів розв'язання систем нелінійних рівнянь наведено у таблицях 7.1 та 7.2. Для точності проведення аналізу алгоритми тестуються на двох різних наборах вхідних даних, причому точність є змінною величиною, за якою буде зроблено аналіз.

Таблиця 7.1 – Тестування ефективності методів на I наборі вхідних даних

Точність	Параметри тестування	Метод	
		Якобі	Гауса-Зейделя
0,1	Кількість ітерацій	9	3
0,01	Кількість ітерацій	15	6
0,001	Кількість ітерацій	21	9
0,0001	Кількість ітерацій	27	12
0,00001	Кількість ітерацій	33	15
0,000001	Кількість ітерацій	41	19
0,0000001	Кількість ітерацій	47	22
0,00000001	Кількість ітерацій	53	25
0,000000001	Кількість ітерацій	59	28
0,0000000001	Кількість ітерацій	65	31

Візуалізація результатів таблиці 7.1 наведена на рисунку 7.13. Для більшої наочності на графіках показана залежність кількості ітерацій від логарифму з основою 0,1 значення точності, замість безпосередньо самої точності (тобто при точності 0,001 залежність буде від 3, при 0,0001 – від 4 і т.д.).

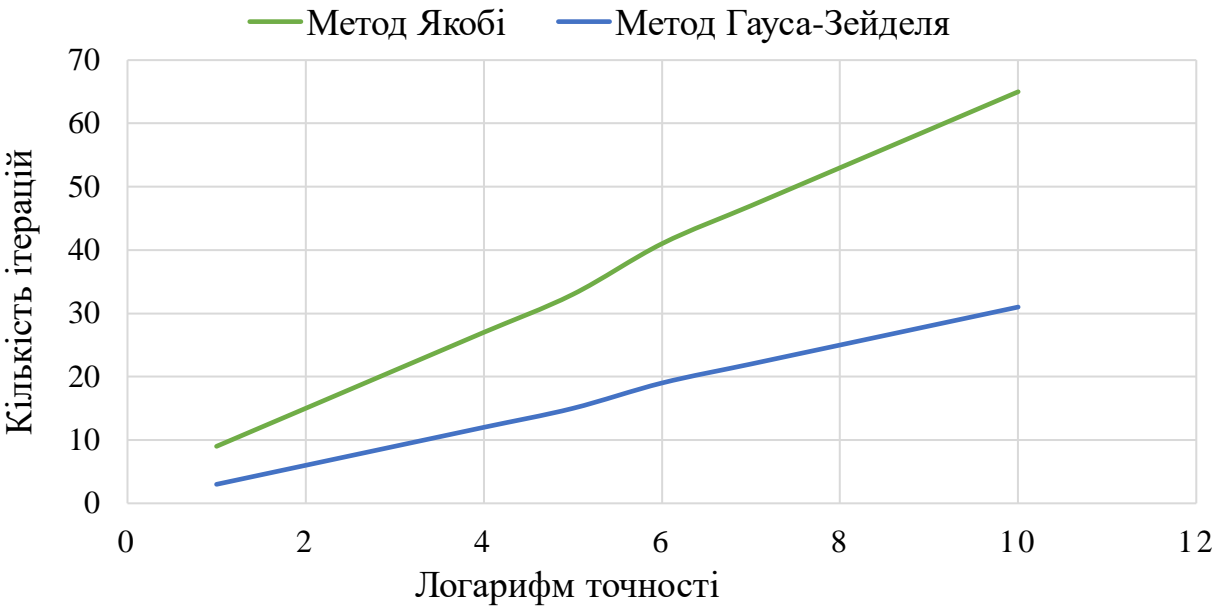


Рисунок 7.13 – Графік залежності кількості ітерацій методу від точності обчислень на I наборі вхідних даних

Таблиця 7.2 – Тестування ефективності методів на II наборі вхідних даних

Точність	Параметри тестування	Метод	
		Якобі	Гауса-Зейделя
0,1	Кількість ітерацій	1	1
0,01	Кількість ітерацій	3	2
0,001	Кількість ітерацій	3	2
0,0001	Кількість ітерацій	5	3
0,00001	Кількість ітерацій	5	3
0,000001	Кількість ітерацій	7	4
0,0000001	Кількість ітерацій	7	4
0,00000001	Кількість ітерацій	9	5
0,000000001	Кількість ітерацій	9	5
0,0000000001	Кількість ітерацій	11	6

Візуалізація результатів таблиці 7.2 наведена на рисунку 7.14.

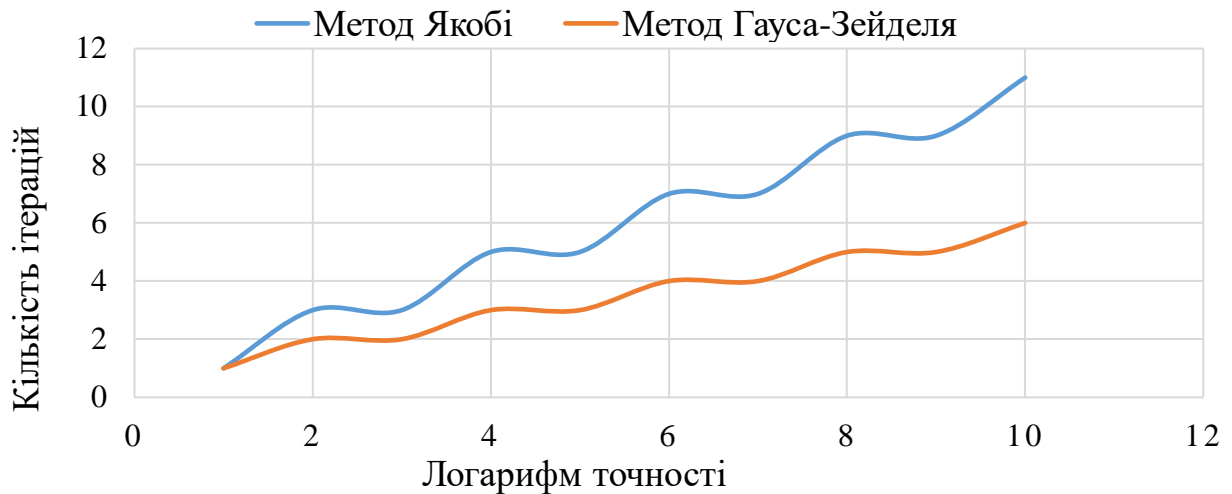


Рисунок 7.14 – Графік залежності кількості ітерацій методу від точності обчислень на II наборі вхідних даних

За результатами тестування можна зробити такі висновки:

- а) Кількість ітерацій для кожного з розглянутих методів сильно відрізняється для різних наборів вхідних даних, причому вона напряму залежить від обраної точності обчислення розв'язку.
- б) Асимптотична складність усіх розглянутих алгоритмів є логарифмічною, тобто  $\theta(\log n)$ , де  $n$  – точність обчислення розв'язку системи рівнянь.
- в) Серед розглянутих методів найоптимальнішим для практичного застосування є метод Гауса-Зейделя, оскільки на всіх наборах вхідних даних він виконується за меншу кількість ітерацій, тобто є швидшим.

## ВИСНОВКИ

У даній курсовій роботі було досліджено ітераційні методи розв'язання систем арифметичних, тригонометричних та трансцендентних рівнянь. Було виконано програмну реалізацію досліджених методів з використанням парадигми ООП засобами для графічного інтерфейсу користувача C++/CLI Windows Forms. Вивчено базові принципи об'єктно-орієнтованого програмування, такі як інкапсуляція, абстрагування, успадкування та поліморфізм.

Згідно з поставленою задачею було розроблено програмне забезпечення, що знаходить рішення заданої системи нелінійних рівнянь одним із наступних методів: методом Якобі, методом Гауса-Зейделя.

Під час виконання даної роботи було побудовано та описано алгоритми для кожного з розглянутих методів розв'язання СНР. Після цього було здійснено об'єктно-орієнтоване моделювання програмного забезпечення із застосуванням діаграми класів нотації мови UML (Unified Modeling Language) та зроблено детальний опис методів частин програмного забезпечення.

Після програмної реалізації сформованої моделі ПЗ було проведено всестороннє тестування роботоспроможності написаної програми і достовірності результатів, які вона видає. Автентичність отриманих коренів СНР було перевірено шляхом порівняння розв'язків, виведених програмою, і розв'язків, отриманих за допомогою загальновизнаних онлайн-калькуляторів. За підсумками тестування та аналізу можна зробити висновок, що програмне забезпечення коректно реагує на всі можливі виключні ситуації і належним чином обробляє їх, а також правильно знаходить корені систем нелінійних рівнянь із заданою похибкою.

Окрім цього було зроблено аналіз побудованого графічного інтерфейсу і виконано його оптимізацію для спрощення сприйняття користувачем функціоналу програми. На основі результуючого інтерфейсу було написано докладну інструкцію з експлуатації програмного забезпечення.

Наприкінці виконання даної курсової роботи було проведено детальний аналіз отриманих результатів, згідно з яким можна сказати, що розроблені алгоритми є досить ефективними і придатні для практичного застосування. При



цьому асимптотична складність усіх розроблених алгоритмів є логарифмічною і залежить від заданої точності обчислень. Серед побудованих алгоритмів найоптимальнішим виявився алгоритм методу Гауса-Зейделя, який показав найкращі результати на всіх тестових наборах вхідних даних.

Таким чином, можна зробити висновок, що розроблене програмне забезпечення відповідає всім поставленим функціональним та нефункціональним вимогам, а також виконує всі поставлені задачі.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Ігнатенко В. В. Методи розв'язування систем нелінійних рівнянь // *СумДУ Конопський інститут: Обчислювальна математика. Методичні вказівки до практичних занять*: навч.-метод. посіб. Суми, 2010. С. 19-22. URL: <https://studfile.net/preview/5581806/page:7/> (дата звернення: 20.05.2022)
2. Розв'язання систем нелінійних рівнянь // *ЛНУ: Фізичний факультет. Методичні вказівки до лабораторних робіт*. URL: <chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://physics.lnu.edu.ua/wp-content/uploads/Lab12.pdf> (дата звернення 20.05.2022)
3. Ігнатенко В. В. Методи розв'язування систем нелінійних рівнянь: метод Зейделя // *СумДУ Конопський інститут: Обчислювальна математика. Методичні вказівки до практичних занять*: навч.-метод. посіб. Суми, 2010. С. 22-23. URL: <https://studfile.net/preview/5581806/page:8/> (дата звернення: 20.05.2022)

**ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ**  
**КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського**

Кафедра  
інформатики та програмної інженерії

Затвердив

Керівник Головченко Максим Миколайович

« 12 » квітня 2022 р.

Виконавець:

Студентка Кушнір Ганна Вікторівна

« 12 » квітня 2022 р.

**ТЕХНІЧНЕ ЗАВДАННЯ**  
на виконання курсової роботи  
на тему: «Розв’язання систем нелінійних рівнянь»  
з дисципліни:  
«Основи програмування»

Київ 2022

1. *Мета:* Метою курсової роботи є розробка програмного забезпечення для розв'язання систем нелінійних рівнянь методом простої ітерації (Якобі) та методом Гауса-Зейделя.

2. *Дата початку роботи:* « 04 » квітня 2022 р.

3. *Дата закінчення роботи:* « 12 » червня 2022 р.

4. *Вимоги до програмного забезпечення.*

1) Функціональні вимоги:

- Можливість задати загальний вигляд системи нелінійних рівнянь (алгебраїчні, тригонометричні або трансцендентні)
- Можливість задати метод розв'язання системи (Якобі або Гауса-Зейделя)
- Можливість задати значення коефіцієнтів рівнянь системи
- Можливість розв'язати систему обраним методом
- Можливість збереження результатів у файл
- Можливість відображення статистичних даних для подальшого аналізу обчислювальної складності алгоритму

2) Нефункціональні вимоги:

- Можливість запуску програмного забезпечення на комп'ютерах із операційною системою Windows 7, Windows 8, Windows 10
- Можливість встановлення програмного забезпечення за допомогою інсталятора на операційні системи Windows 7, Windows 8, Windows 10
- 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.

5. *Стадії та етапи розробки:*

- 1) Об'єктно-орієнтований аналіз предметної області задачі (до 18.04.2022 р.)
  - 2) Об'єктно-орієнтоване проектування архітектури програмної системи (до 26.04.2022 р.)
  - 3) Розробка програмного забезпечення (до 17.05.2022 р.)
  - 4) Тестування розробленої програми (до 31.05.2022 р.)
  - 5) Розробка пояснювальної записки (до 12.06.2022 р.).
  - 6) Захист курсової роботи (до 16.06.2022 р.).
6. *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

*Тексти програмного коду програмного забезпечення  
вирішення задачі розв'язання систем нелінійних рівнянь*

---

(Найменування програми (документа))

*Електронний носій*

---

(Вид носія даних)

*71 арк.*

---

(Обсяг програми (документа), арк.)

*студентки групи ІП-12 І курсу*

*Кушнір Г. В.*

Файл «CppCLR\_WinFormsProject.cpp»

```
#include "pch.h"
#include "Calculator_Form.h"

using namespace System;
using namespace System::Windows::Forms;

[STAThread]
int main()
{
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew CalculatorNamespace::Calculator_Form());
    return 0;
}
```

Файл «Calculator\_Form.h»

```
#pragma once
#include <string>
#include "Equations.h"
#include "Question_Form.h"
using namespace System::Collections::Generic;

namespace CalculatorNamespace {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Клас для відображення форми, що містить засоби для розв'язання
    системи нелінійних рівнянь.
    /// </summary>
    public ref class Calculator_Form : public System::Windows::Forms::Form
    {
        /// <summary>
        /// Створює форму Calculator_Form та ініціалізує її компоненти.
        /// </summary>
        public: Calculator_Form(void);
        /// <summary>
        /// Видаляє всі використовувані формою Calculator_Form компоненти.
        /// </summary>
        protected: ~Calculator_Form();

        /// <summary>
        /// Шлях до файлу, куди користувач зберігає отриманий результат розв'язку
        системи.
        /// </summary>
        private: String^ path;
```

```

    /// <summary>
    /// Система нелінійних рівнянь обраного виду з уведеними коефіцієнтами,
а також,
    /// можливо, з вказаними початковими наближеннями розв'язку.
    /// </summary>
    private: Equations^ equations;
    /// <summary>
    /// Масив, що складається з отриманих результатів (x;y) на кожній
ітерації
    /// (якщо вдалося застосувати для розв'язання ітераційні методи), або з
одного
    /// розв'язку (x;y) (якщо не вдалося застосувати ітераційні методи або
розв'язок
    /// знайшовся аналітично).
    /// </summary>
    private: System::Collections::Generic::List<Double>^ result;

    /// <summary>
    /// Група, яка містить функціонал для вибору виду системи рівнянь.
    /// </summary>
    private: System::Windows::Forms::GroupBox^ boxEqType;
    /// <summary>
    /// Кнопка для вибору системи арифметичних рівнянь.
    /// </summary>
    private: System::Windows::Forms::RadioButton^ btnArithmEq;
    /// <summary>
    /// Кнопка для вибору системи тригонометричних рівнянь.
    /// </summary>
    private: System::Windows::Forms::RadioButton^ btnTrigonomeq;
    /// <summary>
    /// Кнопка для вибору системи трансцендентних рівнянь.
    /// </summary>
    private: System::Windows::Forms::RadioButton^ btnTranscEq;

    /// <summary>
    /// Група, яка містить таблицю для введення коефіцієнтів рівнянь.
    /// </summary>
    private: System::Windows::Forms::GroupBox^ boxCoeff;
    /// <summary>
    /// Таблиця коефіцієнтів рівнянь системи.
    /// </summary>
    private: System::Windows::Forms::DataGridView^ tblCoeff;

    /// <summary>
    /// Група, яка містить функціонал для вибору методу розв'язання системи
рівнянь.
    /// </summary>
    private: System::Windows::Forms::GroupBox^ boxSolMethod;
    /// <summary>
    /// Рядок меню для вибору методу розв'язання системи рівнянь.
    /// </summary>
    private: System::Windows::Forms::MenuStrip^ menuMethod;
    /// <summary>
    /// Меню для вибору методу розв'язання системи рівнянь.
    /// </summary>
    private: System::Windows::Forms::ToolStripMenuItem^ btnMethod;

```



```

/// <summary>
/// Елемент меню для вибору методу Якобі.
/// </summary>
private: System::Windows::Forms::ToolStripMenuItem^ btnJakobiMethod;
/// <summary>
/// Елемент меню для вибору методу Гауса-Зейделя.
/// </summary>
private: System::Windows::Forms::ToolStripMenuItem^ btnZeydelMethod;

/// <summary>
/// Група, яка містить таблицю для введення початкових наближень.
/// </summary>
private: System::Windows::Forms::GroupBox^ boxInitApprox;
/// <summary>
/// Таблиця початкових наближень розв'язку системи рівнянь.
/// </summary>
private: System::Windows::Forms::DataGridView^ tblInitApprox;

/// <summary>
/// Група, яка містить поле для введення точності обчислень.
/// </summary>
private: System::Windows::Forms::GroupBox^ boxPrecision;
/// <summary>
/// Текстове поле для введення точності обчислень.
/// </summary>
private: System::Windows::Forms::TextBox^ tblPrecision;
/// <summary>
/// Знак "=".
/// </summary>
private: System::Windows::Forms::Label^ labelEqual;
/// <summary>
/// Знак "Епсилон".
/// </summary>
private: System::Windows::Forms::Label^ labelEps;

/// <summary>
/// Кнопка для початку розв'язування системи рівнянь.
/// </summary>
private: System::Windows::Forms::Button^ btnStart;
/// <summary>
/// Кнопка для побудови графіків рівнянь системи.
/// </summary>
private: System::Windows::Forms::Button^ btnDrawGraph;
/// <summary>
/// Кнопка для очищення форми.
/// </summary>
private: System::Windows::Forms::Button^ btnClear;

/// <summary>
/// Напис "Розв'язання".
/// </summary>
private: System::Windows::Forms::Label^ labelSolution;
/// <summary>
/// Текстове поле для виведення результату розв'язання системи.
/// </summary>
private: System::Windows::Forms::RichTextBox^ ResultBox;

```

```

    /// <summary>
    /// Текстове поле для виведення обраного рівняння з введеними
коефіцієнтами.
    /// </summary>
    private: System::Windows::Forms::RichTextBox^ tblYourEquation;
    /// <summary>
    /// Графіки рівнянь системи.
    /// </summary>
    private: System::Windows::Forms::DataVisualization::Charting::Chart^
graphEquations;

    /// <summary>
    /// Рядок меню головного вікна.
    /// </summary>
    private: System::Windows::Forms::MenuStrip^ menu;
    /// <summary>
    /// Головне меню.
    /// </summary>
    private: System::Windows::Forms::ToolStripMenuItem^ menuMain;
    /// <summary>
    /// Елемент головного меню для очищення форми.
    /// </summary>
    private: System::Windows::Forms::ToolStripMenuItem^ menuClear;
    /// <summary>
    /// Елемент головного меню для збереження результату у раніше створений
    /// або новий файл.
    /// </summary>
    private: System::Windows::Forms::ToolStripMenuItem^ menuSaveResult;
    /// <summary>
    /// Елемент головного меню для збереження результату у новий файл.
    /// </summary>
    private: System::Windows::Forms::ToolStripMenuItem^ menuSaveResultAs;
    /// <summary>
    /// Елемент головного меню для відкриття останнього збереженого
результату.
    /// </summary>
    private: System::Windows::Forms::ToolStripMenuItem^ menuSeeLast;
    /// <summary>
    /// Меню для виведення інформації.
    /// </summary>
    private: System::Windows::Forms::ToolStripMenuItem^ menuInfo;

    /// <summary>
    /// Необхідна змінна для зберігання компонентів графічного дизайну.
    /// </summary>
    private: System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Ініціалізує всі графічні компоненти форми.
    /// </summary>
    void InitializeComponent(void)
    {
        System::Windows::Forms::DataGridViewCellStyle^
dataGridViewCellStyle1 = (gcnew
System::Windows::Forms::DataGridViewCellStyle());

```

```

        System::Windows::Forms::DataGridViewCellStyle^
dataGridViewCellStyle2 = (gcnew
System::Windows::Forms::DataGridViewCellStyle());

        System::Windows::Forms::DataVisualization::Charting::ChartArea^
chartArea1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::ChartArea());
        System::Windows::Forms::DataVisualization::Charting::Series^
series1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Series());
        System::Windows::Forms::DataVisualization::Charting::Series^
series2 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Series());
        System::Windows::Forms::DataVisualization::Charting::Series^
series3 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Series());
        System::Windows::Forms::DataVisualization::Charting::Series^
series4 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Series());
        this->boxEqType = (gcnew
System::Windows::Forms::GroupBox());
        this->btnTranscEq = (gcnew
System::Windows::Forms::RadioButton());
        this->btnTrigonometEq = (gcnew
System::Windows::Forms::RadioButton());
        this->btnArithmetEq = (gcnew
System::Windows::Forms::RadioButton());
        this->boxCoeff = (gcnew System::Windows::Forms::GroupBox());
        this->tblCoeff = (gcnew
System::Windows::Forms::DataGridView());
        this->boxInitApprox = (gcnew
System::Windows::Forms::GroupBox());
        this->tblInitApprox = (gcnew
System::Windows::Forms::DataGridView());
        this->boxSolMethod = (gcnew
System::Windows::Forms::GroupBox());
        this->menuMethod = (gcnew
System::Windows::Forms::MenuStrip());
        this->btnMethod = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->btnJakobiMethod = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->btnZeydelMethod = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->boxPrecision = (gcnew
System::Windows::Forms::GroupBox());
        this->tblPrecision = (gcnew
System::Windows::Forms::TextBox());
        this->labelEqual = (gcnew System::Windows::Forms::Label());
        this->labelEps = (gcnew System::Windows::Forms::Label());
        this->btnStart = (gcnew System::Windows::Forms::Button());
        this->btnClear = (gcnew System::Windows::Forms::Button());
        this->ResultBox = (gcnew
System::Windows::Forms::RichTextBox());
        this->labelSolution = (gcnew
System::Windows::Forms::Label());

```

```

        this->menuMain = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->menuClear = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->menuSaveResult = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->menuSaveResultAs = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->menuSeeLast = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->menuInfo = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->menu = (gcnew System::Windows::Forms::MenuStrip());
        this->graphEquations = (gcnew
System::Windows::Forms::DataVisualization::Charting::Chart());
        this->tblYourEquation = (gcnew
System::Windows::Forms::RichTextBox());
        this->btnDrawGraph = (gcnew
System::Windows::Forms::Button());
        this->boxEqType->SuspendLayout();
        this->boxCoeff->SuspendLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>tblCoeff))->BeginInit();
        this->boxInitApprox->SuspendLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>tblInitApprox))->BeginInit();
        this->boxSolMethod->SuspendLayout();
        this->menuMethod->SuspendLayout();
        this->boxPrecision->SuspendLayout();
        this->menu->SuspendLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>graphEquations))->BeginInit();
        this->SuspendLayout();
        //
        // boxEqType
        //
        this->boxEqType->AutoSize = true;
        this->boxEqType->Controls->Add(this->btnTranscEq);
        this->boxEqType->Controls->Add(this->btnTrigonometEq);
        this->boxEqType->Controls->Add(this->btnArithmetEq);
        this->boxEqType->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 12, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->boxEqType->Location = System::Drawing::Point(12, 28);
        this->boxEqType->Name = L"boxEqType";
        this->boxEqType->Size = System::Drawing::Size(384, 270);
        this->boxEqType->TabIndex = 0;
        this->boxEqType->TabStop = false;
        this->boxEqType->Text = L"Вид системы:";
        //
        // btnTranscEq
        //

```

```

        this->btnTranscEq->Cursor =
System::Windows::Forms::Cursors::Hand;
        this->btnTranscEq->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->btnTranscEq->Location = System::Drawing::Point(38,
187);

        this->btnTranscEq->Name = L"btnTranscEq";
        this->btnTranscEq->Size = System::Drawing::Size(269, 46);
        this->btnTranscEq->TabIndex = 2;
        this->btnTranscEq->Text = L"  $e^{(a_{11}x)} + e^{(a_{12}y)} + a_{13} =$ 
0\r\n  $e^{(a_{21}x)} + e^{(a_{22}y)} + a_{23} = 0$ ";
        this->btnTranscEq->UseVisualStyleBackColor = true;
        //
        // btnTrigonomeq
        //
        this->btnTrigonomeq->Cursor =
System::Windows::Forms::Cursors::Hand;
        this->btnTrigonomeq->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->btnTrigonomeq->Location = System::Drawing::Point(38,
113);

        this->btnTrigonomeq->Name = L"btnTrigonomeq";
        this->btnTrigonomeq->Size = System::Drawing::Size(269, 43);
        this->btnTrigonomeq->TabIndex = 1;
        this->btnTrigonomeq->Text = L"  $\sin(x + a_{11}) + a_{12} * y +$ 
a13 = 0\r\n  $a_{21} * x + \cos(y + a_{22}) + a_{23} = 0$ ";
        this->btnTrigonomeq->UseVisualStyleBackColor = true;
        //
        // btnArithmeq
        //
        this->btnArithmeq->Cursor =
System::Windows::Forms::Cursors::Hand;
        this->btnArithmeq->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->btnArithmeq->Location = System::Drawing::Point(38,
33);

        this->btnArithmeq->Name = L"btnArithmeq";
        this->btnArithmeq->Size = System::Drawing::Size(269, 46);
        this->btnArithmeq->TabIndex = 0;
        this->btnArithmeq->Text = L"  $a_{11} * x^2 + a_{12} * y^2 + a_{13} =$ 
0\r\n  $a_{21} * x^2 + a_{22} * y^2 + a_{23} = 0$ ";
        this->btnArithmeq->UseVisualStyleBackColor = true;
        //
        // boxCoeff
        //
        this->boxCoeff->AutoSize = true;
        this->boxCoeff->Controls->Add(this->tblCoeff);
        this->boxCoeff->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 12, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->boxCoeff->Location = System::Drawing::Point(402, 28);
        this->boxCoeff->Name = L"boxCoeff";
        this->boxCoeff->Size = System::Drawing::Size(419, 138);
        this->boxCoeff->TabIndex = 1;
        this->boxCoeff->TabStop = false;

```

```

        this->boxCoeff->Text = L"Коефіцієнти рівнянь:";
        //
        // tblCoeff
        //
        this->tblCoeff->AllowUserToAddRows = false;
        this->tblCoeff->AllowUserToDeleteRows = false;
        this->tblCoeff->AllowUserToResizeColumns = false;
        this->tblCoeff->AllowUserToResizeRows = false;
        this->tblCoeff->AutoSizeColumnsMode =
System::Windows::Forms::DataGridViewAutoSizeColumnsMode::Fill;
        this->tblCoeff->BackgroundColor =
System::Drawing::SystemColors::Window;
        this->tblCoeff->BorderStyle =
System::Windows::Forms::BorderStyle::Fixed3D;
        this->tblCoeff->CellBorderStyle =
System::Windows::Forms::DataGridViewCellBorderStyle::Raised;
        this->tblCoeff->ColumnHeadersVisible = false;
        this->tblCoeff->Cursor =
System::Windows::Forms::Cursors::IBeam;
        dataGridViewCellStyle1->Alignment =
System::Windows::Forms::DataGridViewContentAlignment::MiddleCenter;
        dataGridViewCellStyle1->BackColor =
System::Drawing::SystemColors::Window;
        dataGridViewCellStyle1->Font = (gcnew
System::Drawing::Font(L"Times New Roman", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        dataGridViewCellStyle1->ForeColor =
System::Drawing::SystemColors::ControlText;
        dataGridViewCellStyle1->NullValue = L"0";
        dataGridViewCellStyle1->SelectionBackColor =
System::Drawing::SystemColors::ActiveCaption;
        dataGridViewCellStyle1->SelectionForeColor =
System::Drawing::SystemColors::WindowText;
        dataGridViewCellStyle1->WrapMode =
System::Windows::Forms::DataGridViewTriState::False;
        this->tblCoeff->DefaultCellStyle = dataGridViewCellStyle1;
        this->tblCoeff->Location = System::Drawing::Point(23, 33);
        this->tblCoeff->Name = L"tblCoeff";
        this->tblCoeff->RowHeadersVisible = false;
        this->tblCoeff->RowTemplate->Height = 39;
        this->tblCoeff->ScrollBars =
System::Windows::Forms::ScrollBars::Horizontal;
        this->tblCoeff->Size = System::Drawing::Size(360, 80);
        this->tblCoeff->TabIndex = 0;
        //
        // boxInitApprox
        //
        this->boxInitApprox->AutoSize = true;
        this->boxInitApprox->Controls->Add(this->tblInitApprox);
        this->boxInitApprox->Font = (gcnew
System::Drawing::Font(L"Times New Roman", 12));
        this->boxInitApprox->Location = System::Drawing::Point(827,
28);

        this->boxInitApprox->Name = L"boxInitApprox";
        this->boxInitApprox->Size = System::Drawing::Size(272, 137);

```



```

this->boxInitApprox->TabIndex = 9;
this->boxInitApprox->TabStop = false;
this->boxInitApprox->Text = L"Початкові наближення:";
//
// tblInitApprox
//
this->tblInitApprox->AllowUserToAddRows = false;
this->tblInitApprox->AllowUserToDeleteRows = false;
this->tblInitApprox->AllowUserToResizeColumns = false;
this->tblInitApprox->AllowUserToResizeRows = false;
this->tblInitApprox->AutoSizeColumnsMode =
System::Windows::Forms::DataGridViewAutoSizeColumnsMode::Fill;
this->tblInitApprox->BackgroundColor =
System::Drawing::SystemColors::Window;
this->tblInitApprox->BorderStyle =
System::Windows::Forms::BorderStyle::Fixed3D;
this->tblInitApprox->CellBorderStyle =
System::Windows::Forms::DataGridViewCellBorderStyle::Raised;
this->tblInitApprox->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
this->tblInitApprox->ColumnHeadersVisible = false;
this->tblInitApprox->Cursor =
System::Windows::Forms::Cursors::IBeam;
dataGridViewCellStyle2->Alignment =
System::Windows::Forms::DataGridViewContentAlignment::MiddleCenter;
dataGridViewCellStyle2->BackColor =
System::Drawing::SystemColors::Window;
dataGridViewCellStyle2->Font = (gcnew
System::Drawing::Font(L"Times New Roman", 12));
dataGridViewCellStyle2->ForeColor =
System::Drawing::SystemColors::ControlText;
dataGridViewCellStyle2->NullValue = L"0";
dataGridViewCellStyle2->SelectionBackColor =
System::Drawing::SystemColors::ActiveCaption;
dataGridViewCellStyle2->SelectionForeColor =
System::Drawing::SystemColors::WindowText;
dataGridViewCellStyle2->WrapMode =
System::Windows::Forms::DataGridViewTriState::False;
this->tblInitApprox->DefaultCellStyle =
dataGridViewCellStyle2;
this->tblInitApprox->Location = System::Drawing::Point(28,
32);
this->tblInitApprox->Name = L"tblInitApprox";
this->tblInitApprox->RowHeadersVisible = false;
this->tblInitApprox->RowHeadersWidth = 100;
this->tblInitApprox->RowTemplate->Height = 39;
this->tblInitApprox->ScrollBars =
System::Windows::Forms::ScrollBars::Horizontal;
this->tblInitApprox->Size = System::Drawing::Size(200, 80);
this->tblInitApprox->TabIndex = 0;
//
// boxSolMethod
//
this->boxSolMethod->AutoSize = true;
this->boxSolMethod->Controls->Add(this->menuMethod);

```

```

        this->boxSolMethod->Font = (gcnew
System::Drawing::Font(L"Times New Roman", 12));
        this->boxSolMethod->Location = System::Drawing::Point(401,
171);

        this->boxSolMethod->Name = L"boxSolMethod";
        this->boxSolMethod->Size = System::Drawing::Size(419, 127);
        this->boxSolMethod->TabIndex = 3;
        this->boxSolMethod->TabStop = false;
        this->boxSolMethod->Text = L"Метод розв'язання системи:";
        //
        // menuMethod
        //
        this->menuMethod->AutoSize = false;
        this->menuMethod->BackColor =
System::Drawing::SystemColors::MenuBar;
        this->menuMethod->Dock =
System::Windows::Forms::DockStyle::None;
        this->menuMethod->Font = (gcnew
System::Drawing::Font(L"Times New Roman", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->menuMethod->GripMargin =
System::Windows::Forms::Padding(2, 0, 2, 0);
        this->menuMethod->GripStyle =
System::Windows::Forms::ToolStripGripStyle::Visible;
        this->menuMethod->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(1) { this->btnMethod });
        this->menuMethod->Location = System::Drawing::Point(24, 56);
        this->menuMethod->Name = L"menuMethod";
        this->menuMethod->Size = System::Drawing::Size(275, 30);
        this->menuMethod->TabIndex = 2;
        this->menuMethod->Text = L"menuMethod";
        //
        // btnMethod
        //
        this->btnMethod->AutoSize = false;
        this->btnMethod->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(2) {
            this->btnJakobiMethod,
            this->btnZeydelMethod
        });
        this->btnMethod->Name = L"btnMethod";
        this->btnMethod->Size = System::Drawing::Size(260, 25);
        this->btnMethod->Text = L"Оберіть метод розв'язання";
        //
        // btnJakobiMethod
        //
        this->btnJakobiMethod->AutoSize = false;
        this->btnJakobiMethod->BackColor =
System::Drawing::SystemColors::MenuBar;
        this->btnJakobiMethod->Name = L"btnJakobiMethod";
        this->btnJakobiMethod->Size = System::Drawing::Size(273,
24);
        this->btnJakobiMethod->Text = L"Метод простої ітерації
(Якобі)";

```



```

        this->btnJakobiMethod->Click += gcnnew
System::EventHandler(this, &Calculator_Form::btnJakobiMethod_Click);
        //
        // btnZeydelMethod
        //
        this->btnZeydelMethod->AutoSize = false;
        this->btnZeydelMethod->BackColor =
System::Drawing::SystemColors::MenuBar;
        this->btnZeydelMethod->Name = L"btnZeydelMethod";
        this->btnZeydelMethod->Size = System::Drawing::Size(273,
24);
        this->btnZeydelMethod->Text = L"Метод Гауса-Зейделя";
        this->btnZeydelMethod->Click += gcnnew
System::EventHandler(this, &Calculator_Form::btnZeydelMethod_Click);
        //
        // boxPrecision
        //
        this->boxPrecision->AutoSize = true;
        this->boxPrecision->Controls->Add(this->tblPrecision);
        this->boxPrecision->Controls->Add(this->labelEqual);
        this->boxPrecision->Controls->Add(this->labelEps);
        this->boxPrecision->Font = (gcnnew
System::Drawing::Font(L"Times New Roman", 12));
        this->boxPrecision->Location = System::Drawing::Point(826,
171);
        this->boxPrecision->Name = L"boxPrecision";
        this->boxPrecision->Size = System::Drawing::Size(273, 127);
        this->boxPrecision->TabIndex = 10;
        this->boxPrecision->TabStop = false;
        this->boxPrecision->Text = L"Точність:";
        //
        // tblPrecision
        //
        this->tblPrecision->Font = (gcnnew
System::Drawing::Font(L"Times New Roman", 12));
        this->tblPrecision->Location = System::Drawing::Point(86,
56);
        this->tblPrecision->Name = L"tblPrecision";
        this->tblPrecision->Size = System::Drawing::Size(142, 26);
        this->tblPrecision->TabIndex = 2;
        this->tblPrecision->Text = L"0,0001";
        //
        // labelEqual
        //
        this->labelEqual->AutoSize = true;
        this->labelEqual->Font = (gcnnew
System::Drawing::Font(L"Times New Roman", 12));
        this->labelEqual->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->labelEqual->Location = System::Drawing::Point(58, 58);
        this->labelEqual->Name = L"labelEqual";
        this->labelEqual->Size = System::Drawing::Size(18, 19);
        this->labelEqual->TabIndex = 1;
        this->labelEqual->Text = L"=";
        //
        // labelEps

```

```

        //
        this->labelEps->AutoSize = true;
        this->labelEps->Font = (gcnew
System::Drawing::Font(L"Vladimir Script", 12,
System::Drawing::FontStyle::Bold));
        this->labelEps->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->labelEps->Location = System::Drawing::Point(30, 59);
        this->labelEps->Name = L"labelEps";
        this->labelEps->Size = System::Drawing::Size(19, 19);
        this->labelEps->TabIndex = 0;
        this->labelEps->Text = L"E";
        //
        // btnStart
        //
        this->btnStart->Cursor =
System::Windows::Forms::Cursors::Hand;
        this->btnStart->Font = (gcnew
System::Drawing::Font(L"Trebuchet MS", 14));
        this->btnStart->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->btnStart->Location = System::Drawing::Point(447, 307);
        this->btnStart->Name = L"btnStart";
        this->btnStart->Size = System::Drawing::Size(209, 72);
        this->btnStart->TabIndex = 5;
        this->btnStart->Text = L"Розв'язати";
        this->btnStart->UseVisualStyleBackColor = true;
        this->btnStart->Click += gcnew System::EventHandler(this,
&Calculator_Form::btnStart_Click);
        //
        // btnClear
        //
        this->btnClear->Cursor =
System::Windows::Forms::Cursors::Hand;
        this->btnClear->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 13, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->btnClear->Location = System::Drawing::Point(985, 322);
        this->btnClear->Name = L"btnClear";
        this->btnClear->Size = System::Drawing::Size(114, 40);
        this->btnClear->TabIndex = 11;
        this->btnClear->Text = L"Очистити";
        this->btnClear->UseVisualStyleBackColor = true;
        this->btnClear->Click += gcnew System::EventHandler(this,
&Calculator_Form::btnClear_Click);
        //
        // ResultBox
        //
        this->ResultBox->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 12, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->ResultBox->Location = System::Drawing::Point(12, 451);
        this->ResultBox->Name = L"ResultBox";
        this->ResultBox->ReadOnly = true;

```

```

        this->ResultBox->Size = System::Drawing::Size(657, 288);
        this->ResultBox->TabIndex = 7;
        this->ResultBox->Text = L"";
        this->ResultBox->Visible = false;
        //
        // labelSolution
        //
        this->labelSolution->AutoSize = true;
        this->labelSolution->Font = (gcnew
System::Drawing::Font(L"Times New Roman", 13));
        this->labelSolution->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->labelSolution->Location = System::Drawing::Point(12,
425);

        this->labelSolution->Name = L"labelSolution";
        this->labelSolution->Size = System::Drawing::Size(104, 20);
        this->labelSolution->TabIndex = 8;
        this->labelSolution->Text = L"Розв'язання:";
        this->labelSolution->Visible = false;
        //
        // menuMain
        //
        this->menuMain->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(4) {
            this->menuClear,
            this->menuSaveResult, this->menuSaveResultAs,
this->menuSeeLast
        });
        this->menuMain->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 11, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->menuMain->Name = L"menuMain";
        this->menuMain->Size = System::Drawing::Size(58, 21);
        this->menuMain->Text = L"Меню";
        //
        // menuClear
        //
        this->menuClear->Name = L"menuClear";
        this->menuClear->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Con
trol | System::Windows::Forms::Keys::D0));
        this->menuClear->Size = System::Drawing::Size(378, 22);
        this->menuClear->Text = L"Очистити";
        this->menuClear->Click += gcnew System::EventHandler(this,
&Calculator_Form::menuClear_Click);
        //
        // menuSaveResult
        //
        this->menuSaveResult->Enabled = false;
        this->menuSaveResult->Name = L"menuSaveResult";
        this->menuSaveResult->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Con
trol | System::Windows::Forms::Keys::S));
        this->menuSaveResult->Size = System::Drawing::Size(378, 22);
        this->menuSaveResult->Text = L"Зберегти результат";

```

```

        this->menuSaveResult->Click += gcnew
System::EventHandler(this, &Calculator_Form::menuSaveResult_Click);
        //
        // menuSaveResultAs
        //
        this->menuSaveResultAs->Enabled = false;
        this->menuSaveResultAs->Name = L"menuSaveResultAs";
        this->menuSaveResultAs->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>(((System::Windows::Forms::Keys::Co
ntrol | System::Windows::Forms::Keys::Shift)
        | System::Windows::Forms::Keys::S));
        this->menuSaveResultAs->Size = System::Drawing::Size(378,
22);
        this->menuSaveResultAs->Text = L"Зберегти результат як...";
        this->menuSaveResultAs->Click += gcnew
System::EventHandler(this, &Calculator_Form::menuSaveResultAs_Click);
        //
        // menuSeeLast
        //
        this->menuSeeLast->Enabled = false;
        this->menuSeeLast->Name = L"menuSeeLast";
        this->menuSeeLast->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Con
trol | System::Windows::Forms::Keys::O));
        this->menuSeeLast->Size = System::Drawing::Size(378, 22);
        this->menuSeeLast->Text = L"Відкрити останній збережений
результат";
        this->menuSeeLast->Click += gcnew System::EventHandler(this,
&Calculator_Form::menuSeeLast_Click);
        //
        // menuInfo
        //
        this->menuInfo->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 11, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->menuInfo->Name = L"menuInfo";
        this->menuInfo->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Con
trol | System::Windows::Forms::Keys::I));
        this->menuInfo->Size = System::Drawing::Size(89, 21);
        this->menuInfo->Text = L"Інформація";
        this->menuInfo->Click += gcnew System::EventHandler(this,
&Calculator_Form::menuInfo_Click);
        //
        // menu
        //
        this->menu->Font = (gcnew System::Drawing::Font(L"Times New
Roman", 11, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->menu->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(2) { this->menuMain, this-
>menuInfo });
        this->menu->Location = System::Drawing::Point(0, 0);
        this->menu->Name = L"menu";

```

```

        this->menu->Size = System::Drawing::Size(1114, 25);
        this->menu->TabIndex = 13;
        this->menu->Text = L"Menu";
        //
        // graphEquations
        //
        this->graphEquations->BackColor =
System::Drawing::SystemColors::Window;
        chartArea1->AxisX->ScaleBreakStyle->Enabled = true;
        chartArea1->AxisX->ScaleView->SizeType =
System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::N
umber;
        chartArea1->CursorX->Interval = 0.01;
        chartArea1->CursorX->IsUserEnabled = true;
        chartArea1->CursorX->IsUserSelectionEnabled = true;
        chartArea1->CursorX->LineColor =
System::Drawing::Color::LimeGreen;
        chartArea1->CursorY->Interval = 0.01;
        chartArea1->CursorY->IsUserEnabled = true;
        chartArea1->CursorY->IsUserSelectionEnabled = true;
        chartArea1->CursorY->LineColor =
System::Drawing::Color::LimeGreen;
        chartArea1->Name = L"ChartArea1";
        this->graphEquations->ChartAreas->Add(chartArea1);
        this->graphEquations->Cursor =
System::Windows::Forms::Cursors::Cross;
        this->graphEquations->Location = System::Drawing::Point(676,
389);
        this->graphEquations->Name = L"graphEquations";
        series1->BorderWidth = 2;
        series1->ChartArea = L"ChartArea1";
        series1->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline
;
        series1->Color = System::Drawing::Color::Fuchsia;
        series1->CustomProperties = L"IsXAxisQuantitative=False";
        series1->Legend = L"Legend1";
        series1->Name = L"First";
        series1->YValuesPerPoint = 2;
        series2->BorderWidth = 2;
        series2->ChartArea = L"ChartArea1";
        series2->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline
;
        series2->Color = System::Drawing::Color::DarkOrange;
        series2->Legend = L"Legend1";
        series2->Name = L"Second";
        series2->YValuesPerPoint = 2;
        series3->BorderWidth = 2;
        series3->ChartArea = L"ChartArea1";
        series3->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline
;
        series3->Color = System::Drawing::Color::Fuchsia;
        series3->Name = L"First2";
        series4->BorderWidth = 2;

```

```

        series4->ChartArea = L"ChartArea1";
        series4->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline
;
        series4->Color = System::Drawing::Color::DarkOrange;
        series4->Name = L"Second2";
        this->graphEquations->Series->Add(series1);
        this->graphEquations->Series->Add(series2);
        this->graphEquations->Series->Add(series3);
        this->graphEquations->Series->Add(series4);
        this->graphEquations->Size = System::Drawing::Size(423,
350);
        this->graphEquations->TabIndex = 14;
        this->graphEquations->Visible = false;
        //
        // tblYourEquation
        //
        this->tblYourEquation->BackColor =
System::Drawing::SystemColors::Window;
        this->tblYourEquation->BorderStyle =
System::Windows::Forms::BorderStyle::None;
        this->tblYourEquation->Font = (gcnew
System::Drawing::Font(L"Times New Roman", 13,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->tblYourEquation->Location = System::Drawing::Point(50,
324);
        this->tblYourEquation->Name = L"tblYourEquation";
        this->tblYourEquation->ReadOnly = true;
        this->tblYourEquation->Size = System::Drawing::Size(346,
75);
        this->tblYourEquation->TabIndex = 15;
        this->tblYourEquation->Text = L"";
        this->tblYourEquation->Visible = false;
        //
        // btnDrawGraph
        //
        this->btnDrawGraph->Location = System::Drawing::Point(691,
319);
        this->btnDrawGraph->Name = L"btnDrawGraph";
        this->btnDrawGraph->Size = System::Drawing::Size(256, 46);
        this->btnDrawGraph->TabIndex = 16;
        this->btnDrawGraph->Text = L"Побудувати графік для
визначення початкових наближень";
        this->btnDrawGraph->UseVisualStyleBackColor = true;
        this->btnDrawGraph->Click += gcnew
System::EventHandler(this, &Calculator_Form::btnDrawGraph_Click);
        //
        // Calculator_Form
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(9, 19);
        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
        this->BackColor = System::Drawing::SystemColors::Window;
        this->ClientSize = System::Drawing::Size(1114, 761);
        this->Controls->Add(this->btnDrawGraph);

```



```

        this->Controls->Add(this->tblYourEquation);
        this->Controls->Add(this->graphEquations);
        this->Controls->Add(this->btnClear);
        this->Controls->Add(this->boxEqType);
        this->Controls->Add(this->boxCoeff);
        this->Controls->Add(this->boxInitApprox);
        this->Controls->Add(this->boxSolMethod);
        this->Controls->Add(this->boxPrecision);
        this->Controls->Add(this->btnStart);
        this->Controls->Add(this->ResultBox);
        this->Controls->Add(this->labelSolution);
        this->Controls->Add(this->menu);
        this->Font = (gcnew System::Drawing::Font(L"Times New Roman",
12,                                     System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->HelpButton = true;
        this->KeyPreview = true;
        this->MainMenuStrip = this->menu;
        this->MaximumSize = System::Drawing::Size(1130, 1100);
        this->MinimumSize = System::Drawing::Size(1130, 800);
        this->Name = L"Calculator_Form";
        this->SizeGripStyle =
System::Windows::Forms::SizeGripStyle::Show;
        this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
        this->Text = L"Калькулятор \"Системи нелінійних рівнянь\"";
        this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&Calculator_Form::Calculator_FormClosing);
        this->Load += gcnew System::EventHandler(this,
&Calculator_Form::Calculator_Load);
        this->boxEqType->ResumeLayout(false);
        this->boxCoeff->ResumeLayout(false);

        (cli::safe_cast<System::ComponentModel::ISupportInitialize>(this-
>tblCoeff))->EndInit();
        this->boxInitApprox->ResumeLayout(false);

        (cli::safe_cast<System::ComponentModel::ISupportInitialize>(this-
>tblInitApprox))->EndInit();
        this->boxSolMethod->ResumeLayout(false);
        this->menuMethod->ResumeLayout(false);
        this->menuMethod->PerformLayout();
        this->boxPrecision->ResumeLayout(false);
        this->boxPrecision->PerformLayout();
        this->menu->ResumeLayout(false);
        this->menu->PerformLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize>(this-
>graphEquations))->EndInit();
        this->ResumeLayout(false);
        this->PerformLayout();
    }
#pragma endregion

```

```

    /// <summary>
    /// Викликається при завантаженні форми. Ініціалізує таблиці та
    користувачі
    /// компоненти форми.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void Calculator_Load(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається, коли користувач намагається закрити форму. Запитує у
    користувача,
    /// чи дійсно він бажає вийти, і якщо так – закриває форму.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу FormClosingEventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void Calculator_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на кнопку btnStart. Опрацьовує введені
    користувачем
    /// дані та, якщо не виявлено помилок, розв'язує обрану систему рівнянь
    обраним методом.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void btnStart_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на кнопку btnDrawGraph. Опрацьовує
    введені користувачем
    /// дані та викликає дочірню форму для введення меж для побудови графіків,
    і на основі
    /// цих даних будує графіки.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void btnDrawGraph_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на кнопку btnClear. Викликає функцію для
    очищення форми.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void btnClear_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на елемент меню menuClear. Викликає
    функцію для очищення

```



```

    /// форми.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void menuClear_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на елемент меню menuSaveResult. Викликає
функцію для
    /// збереження результату у новий файл, якщо користувач зберігає
результат вперше;
    /// інакше – оновлює вже створений файл.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void menuSaveResult_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на елемент меню menuSaveResultAs.
Викликає функцію для
    /// збереження результату у новий файл.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void menuSaveResultAs_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на елемент меню menuSeeLast. Відкриває
останній збережений результат.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void menuSeeLast_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на елемент меню menuInfo. Відкриває вікно
повідомлення,
    /// яке відображає інформацію про програмне забезпечення.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void menuInfo_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на елемент меню btnJakobiMethod. Знімає
мітку з методу
    /// Зейделя і встановлює її на метод Якобі.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>

```

```

private: System::Void btnJakobiMethod_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Викликається при натисканні на елемент меню btnZeydelMethod. Знімає
мітку з методу
    /// Якобі і встановлює її на метод Зейделя.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
private: System::Void btnZeydelMethod_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Сканує коефіцієнти рівнянь системи з форми Calculator_Form та записує
їх у
    /// двовимірний масив coef.
    /// </summary>
    /// <param name = 'coef'>Двовимірний динамічний масив з елементами типу
Double,
    /// у який зчитуються коефіцієнти, введені користувачем.</param>
    /// <returns>Значення типу Void.</returns>
private: System::Void scanCoefficients(Double** coef);
    /// <summary>
    /// Сканує початкові наближення з форми Calculator_Form та записує їх у
одновимірний
    /// масив initX.
    /// </summary>
    /// <param name = 'initX'>Одновимірний динамічний масив з елементами типу
Double,
    /// у який зчитуються початкові наближення, введені користувачем.</param>
    /// <returns>Значення типу Void.</returns>
private: System::Void scanInitApprox(Double* initX);
    /// <summary>
    /// Очищує форму та відновлює її до початкового стану.
    /// </summary>
    /// <returns>Значення типу Void.</returns>
private: System::Void clear();
    /// <summary>
    /// Зберігає результат розв'язання системи рівнянь у текстовий файл за
допомогою
    /// діалогового вікна SaveFileDialog.
    /// </summary>
    /// <returns>Значення типу Void.</returns>
private: System::Void saveNewResult();
    /// <summary>
    /// Будує графіки заданої в класі Calculator_Form системи нелінійних
рівнянь на
    /// заданому проміжку.
    /// </summary>
    /// <param name = 'a'>Початок проміжку, на якому потрібно побудувати
графіки.</param>
    /// <param name = 'b'>Кінець проміжку, на якому потрібно побудувати
графіки.</param>
    /// <param name = 'flag'>Прапорець, який визначає, потрібно побудувати
графіки

```

```

    /// з ймовірно відомою точкою їх перетину (flag == 2) чи ні (flag ==
1).</param>
    /// <returns>Значення типу Void.</returns>
    private: System::Void drawGraph(Double a, Double b, Int16 flag);
    /// <summary>
    /// Будує графіки заданої в формі Calculator_Form системи арифметичних
рівнянь
    /// на вказаному проміжку з вказаною точністю побудови.
    /// </summary>
    /// <param name = 'a'>Початок проміжку, на якому потрібно побудувати
графіки.</param>
    /// <param name = 'b'>Кінець проміжку, на якому потрібно побудувати
графіки.</param>
    /// <param name = 'step'>Крок побудови графіків.</param>
    /// <returns>Повертає -3, якщо не вдалося побудувати графік жодного з
рівнянь системи;
    /// -1, якщо не вдалося побудувати лише перше рівняння системи;
    /// -2, якщо не вдалося побудувати лише друге рівняння системи;
    /// інакше - повертає 2.</returns>
    private: System::Int16 drawGraphArithmetic(Double a, Double b, Double
step);
    /// <summary>
    /// Будує графіки заданої в формі Calculator_Form системи
тригонометричних рівнянь
    /// на вказаному проміжку з вказаною точністю побудови.
    /// </summary>
    /// <param name = 'a'>Початок проміжку, на якому потрібно побудувати
графіки.</param>
    /// <param name = 'b'>Кінець проміжку, на якому потрібно побудувати
графіки.</param>
    /// <param name = 'step'>Крок побудови графіків.</param>
    /// <returns>Повертає -3, якщо не вдалося побудувати графік жодного з
рівнянь системи;
    /// -1, якщо не вдалося побудувати лише перше рівняння системи;
    /// -2, якщо не вдалося побудувати лише друге рівняння системи;
    /// інакше - повертає 2.</returns>
    private: System::Int16 drawGraphTrigonometric(Double a, Double b, Double
step);
    /// <summary>
    /// Будує графіки заданої в формі Calculator_Form системи трансцендентних
рівнянь
    /// на вказаному проміжку з вказаною точністю побудови.
    /// </summary>
    /// <param name = 'a'>Початок проміжку, на якому потрібно побудувати
графіки.</param>
    /// <param name = 'b'>Кінець проміжку, на якому потрібно побудувати
графіки.</param>
    /// <param name = 'step'>Крок побудови графіків.</param>
    /// <returns>Повертає -3, якщо не вдалося побудувати графік жодного з
рівнянь системи;
    /// -1, якщо не вдалося побудувати лише перше рівняння системи;
    /// -2, якщо не вдалося побудувати лише друге рівняння системи;
    /// інакше - повертає 2.</returns>
    private: System::Int16 drawGraphTranscendental(Double a, Double b,
Double step);
    /// <summary>

```

```

    /// "Готує" форму для виведення та подальшої обробки результату. Робить
видимими поля
    /// розв'язку, графік, а також відкриває можливість зберегти подальший
результат у файл.
    /// Виводить введене користувачем рівняння у форму.
    /// </summary>
    /// <returns>Значення типу Void</returns>
private: System::Void solvingBegin();
    /// <summary>
    /// На основі отриманого параметра flag виводить результат розв'язання
системи у форму,
    /// у тому числі викликає функцію для виведення результатів ітераційного
процесу.
    /// </summary>
    /// <param name = 'flag'>Прапорець, який визначає, вдалося розв'язати
систему чи ні.</param>
    /// <returns>Значення типу Void.</returns>
private: System::Void outputResult(Int16 flag);
    /// <summary>
    /// Виводить у текстове поле форми результату, отримані на кожній з
ітерацій процесу
    /// розв'язання введеної системи обраним методом, а також виводить
кількість цих ітерацій.
    /// </summary>
    /// <returns>Значення типу Void.</returns>
private: System::Void outputIterations();
    /// <summary>
    /// Ініціалізує об'єкт системи рівнянь обраного виду уведеними
користувачем коефіцієнтами
    /// та початковим наближенням розв'язку.
    /// </summary>
    /// <returns>Значення типу Boolean, яке визначає, чи сталися якісь
помилки при ініціалізації
    /// (повертає false), чи все пройшло успішно (повертає true).</returns>
private: System::Boolean initializeEquationsToSolve();
    /// <summary>
    /// Ініціалізує об'єкт системи рівнянь обраного виду уведеними
користувачем коефіцієнтами.
    /// </summary>
    /// <returns>Значення типу Boolean, яке визначає, чи сталися якісь
помилки при ініціалізації.
    /// (повертає false), чи все пройшло успішно (повертає true).</returns>
private: System::Boolean initializeEquationsToDrawGraph();
    /// <summary>
    /// У залежності від обраного методу розв'язання, викликає відповідні
функції методів, або ж виводить
    /// повідомлення про помилку, якщо не було обрано жодного методу.
    /// </summary>
    /// <param name = 'precis'>Точність, з якою потрібно обчислити розв'язок
системи.</param>
    /// <returns>Значення типу Void.</returns>
private: System::Void solve(Double precis);
};
}

```

Файл «Calculator\_Form.cpp»

```
#include "pch.h"
#include "Calculator_Form.h"
#include "ArithmeticEquations.h"
#include "TrigonometricEquations.h"
#include "TranscendentalEquations.h"

CalculatorNamespace::Calculator_Form::Calculator_Form(void)
{
    InitializeComponent();
    String^ output = ("Щоб розв'язати систему двох нелінійних рівнянь,
    оберіть вид системи та метод її розв'язання, " +
        "введіть коефіцієнти, початкові наближення та точність обчислення
    розв'язку системи.\nДалі натисніть \"Розв'язати\"." +
        "\n\nЯкщо ви не знаєте початкові наближення розв'язку, оберіть вид
    системи та введіть коефіцієнти рівнянь." +
        "\nДалі натисніть \"Побудувати графік для визначення початкових
    наближень\" і у вікні, що з'явиться, введіть інтервал по X, " +
        "на якому бажаєте побудувати графік.\nЗа побудованим графіком ви
    зможете самостійно визначити орієнтовний наближений розв'язок.");
    MessageBox::Show(output, "Вітання!", MessageBoxButtons::OK,
    MessageBoxIcon::None, MessageBoxDefaultButton::Button1);
}
CalculatorNamespace::Calculator_Form::~Calculator_Form()
{
    if (components)
    {
        delete components;
    }
}
System::Void
CalculatorNamespace::Calculator_Form::Calculator_Load(System::Object^ sender,
System::EventArgs^ e)
{
    tblCoeff->ColumnCount = 3;
    tblCoeff->RowCount = 2;
    tblInitApprox->ColumnCount = 1;
    tblInitApprox->RowCount = 2;
    result = gcnew List<Double>();
    path = "";
}
System::Void
CalculatorNamespace::Calculator_Form::Calculator_FormClosing(System::Object^
sender, System::Windows::Forms::FormClosingEventArgs^ e)
{
    String^ output = "Ви дійсно бажаєте вийти?\nПрограма втратить шлях до
останнього збереженого результату.";
    if (MessageBox::Show(output, "Вихід", MessageBoxButtons::YesNo,
    MessageBoxIcon::Question) == Windows::Forms::DialogResult::No) {
        e->Cancel = true;
    }
}
}
```

```

System::Void
CalculatorNamespace::Calculator_Form::btnStart_Click(System::Object^ sender,
System::EventArgs^ e)
{
    result->Clear();
    ResultBox->Text = "";
    Double precis;          // Змінна для зчитування точності.
    try {
        precis = Convert::ToDouble(tblPrecision->Text);
        if (precis <= 0) throw "Uncorrect";
    }
    catch (const char*) {
        MessageBox::Show("Точність має бути додатнім числом!", "Помилка!",
        MessageBoxButtons::OK, MessageBoxIcon::Warning);
        return;
    }
    catch (...) {
        if (MessageBox::Show("Не коректний ввід числових даних!\nОчистити
форму?", "Помилка!", MessageBoxButtons::YesNo, MessageBoxIcon::Warning) ==
        System::Windows::Forms::DialogResult::Yes)
        {
            clear();
        }
        return;
    }

    if (!initializeEquationsToSolve()) return;

    solve(precis);
}
System::Void
CalculatorNamespace::Calculator_Form::btnDrawGraph_Click(System::Object^
sender, System::EventArgs^ e)
{
    ResultBox->Text = "";
    tblYourEquation->Visible = true;
    graphEquations->Visible = true;
    menuSaveResult->Enabled = false;
    menuSaveResultAs->Enabled = false;
    ResultBox->Visible = false;
    labelSolution->Visible = false;

    if (!initializeEquationsToDrawGraph()) return;

    tblYourEquation->Text = "Введена система рівнянь:\n" + equations-
>getEquations();

    Question_FormNamespace::Question_Form^ childForm = gcnew
Question_FormNamespace::Question_Form;          // Форма для введення меж
побудови графіків.
    childForm->ShowDialog();
    Double from = childForm->getXFrom();          // Ліва межа побудови графіків.
    Double to = childForm->getXTo();              // Права межа побудови графіків.
    drawGraph(from, to, 1);
}

```

```

System::Void
CalculatorNamespace::Calculator_Form::btnClear_Click(System::Object^ sender,
System::EventArgs^ e)
{
    clear();
}

System::Void
CalculatorNamespace::Calculator_Form::menuClear_Click(System::Object^ sender,
System::EventArgs^ e)
{
    clear();
}

System::Void
CalculatorNamespace::Calculator_Form::menuSaveResult_Click(System::Object^
sender, System::EventArgs^ e)
{
    if (path == "") {
        saveNewResult();
    }
    else {
        try {
            System::IO::File::WriteAllText(path, tblYourEquation->Text +
"\n\n" + ResultBox->Text);
            menuSeeLast->Enabled = true;
        }
        catch (...) {
            MessageBox::Show("Не вдалося зберегти результат!",
"Помилка!", MessageBoxButtons::OK, MessageBoxIcon::Error);
            return;
        }
    }
}

System::Void
CalculatorNamespace::Calculator_Form::menuSaveResultAs_Click(System::Object^
sender, System::EventArgs^ e)
{
    saveNewResult();
}

System::Void
CalculatorNamespace::Calculator_Form::menuSeeLast_Click(System::Object^
sender, System::EventArgs^ e)
{
    try {
        System::Diagnostics::Process::Start("notepad.exe", path);
    }
    catch (...) {
        MessageBox::Show("Не вдалося відкрити файл!", "Помилка!",
MessageBoxButtons::OK, MessageBoxIcon::Warning);
        return;
    }
}

System::Void
CalculatorNamespace::Calculator_Form::menuInfo_Click(System::Object^ sender,
System::EventArgs^ e)
{

```



```

String^ output = ("Щоб розв'язати систему двох нелінійних рівнянь,
оберіть вид системи та метод її розв'язання, " +
    "введіть коефіцієнти, початкові наближення та точність обчислення
розв'язку системи.\nДалі натисніть \"Розв'язати\".\" +
    "\n\nЯкщо ви не знаєте початкові наближення розв'язку, оберіть вид
системи та введіть коефіцієнти рівнянь.\" +
    "\nДалі натисніть \"Побудувати графік для визначення початкових
наближень\" і у вікні, що з'явиться, введіть інтервал по X, " +
    "на якому бажаєте побудувати графік.\nЗа побудованим графіком ви
зможете самостійно визначити орієнтовний наближений розв'язок.\" +
    "\n\nEnvironment:\n                                Microsoft Visual Studio Community
2022\n                                Version 17.2.1\n" +
    "\nDeveloped by: \n                                Hanna Kushnir");
MessageBox::Show(output, "Інформація", MessageBoxButtons::OK,
MessageBoxIcon::None, MessageBoxDefaultButton::Button1);
}
System::Void
CalculatorNamespace::Calculator_Form::btnJakobiMethod_Click(System::Object^
sender, System::EventArgs^ e)
{
    btnJakobiMethod->Checked = true;
    btnZeydelMethod->Checked = false;
    btnMethod->Text = btnJakobiMethod->Text;
}
System::Void
CalculatorNamespace::Calculator_Form::btnZeydelMethod_Click(System::Object^
sender, System::EventArgs^ e)
{
    btnZeydelMethod->Checked = true;
    btnJakobiMethod->Checked = false;
    btnMethod->Text = btnZeydelMethod->Text;
}

System::Void CalculatorNamespace::Calculator_Form::scanCoefficients(Double**
coef)
{
    for (Int16 i = 0; i < 2; ++i) {
        for (Int16 j = 0; j < 3; ++j) {
            coef[i][j] = Convert::ToDouble(tblCoeff->Rows[i]->Cells[j]-
>Value);
        }
    }
}
System::Void CalculatorNamespace::Calculator_Form::scanInitApprox(Double*
initX)
{
    for (Int16 i = 0; i < 2; ++i) {
        initX[i] = Convert::ToDouble(tblInitApprox->Rows[i]->Cells[0]-
>Value);
    }
}
System::Void CalculatorNamespace::Calculator_Form::clear()
{
    btnArithmEq->Checked = false;
    btnTrigonometricEq->Checked = false;
    btnTranscEq->Checked = false;
}

```



```

    btnMethod->Text = "Оберіть метод розв'язання";
    btnJakobiMethod->Checked = false;
    btnZeydelMethod->Checked = false;
    for (Int16 i = 0; i < 2; ++i) {
        for (Int16 j = 0; j < 3; ++j) {
            tblCoeff->Rows[i]->Cells[j]->Value = nullptr;
        }
        tblInitApprox->Rows[i]->Cells[0]->Value = nullptr;
    }
    tblPrecision->Text = "0,0001";
    ResultBox->Text = "";
    tblYourEquation->Text = "";
    graphEquations->Series[0]->Points->Clear();
    graphEquations->Series[1]->Points->Clear();
    graphEquations->Series[2]->Points->Clear();
    graphEquations->Series[3]->Points->Clear();
    menuSaveResult->Enabled = false;
    menuSaveResultAs->Enabled = false;
    ResultBox->Visible = false;
    labelSolution->Visible = false;
    tblYourEquation->Visible = false;
    graphEquations->Visible = false;
}
System::Void CalculatorNamespace::Calculator_Form::saveNewResult()
{
    try {
        SaveFileDialog^ saveResultDialog = gcnew SaveFileDialog();
        // Вікно збереження результату.
        saveResultDialog->Filter = "Text file (.txt)|*.txt";
        saveResultDialog->Title = "Збереження результату";
        if (saveResultDialog->ShowDialog() ==
Windows::Forms::DialogResult::Cancel) {
            return;
        }
        else {
            path = saveResultDialog->FileName;
            System::IO::File::WriteAllText(path, tblYourEquation->Text +
"\n\n" + ResultBox->Text);
        }
    }
    catch (...) {
        MessageBox::Show("Не вдалося зберегти результат!", "Помилка!",
MessageBoxButtons::OK, MessageBoxIcon::Error);
        return;
    }
    MessageBox::Show("Ваш результат успішно збережено!", "Успішно!",
MessageBoxButtons::OK, MessageBoxIcon::None);
    menuSeeLast->Enabled = true;
}
System::Void CalculatorNamespace::Calculator_Form::drawGraph(Double a, Double
b, Int16 flag)
{
    if (flag == 2 && result->Count != 0) {
        a = Math::Floor(result[result->Count - 2] * 10) / 10 - 10;
        b = a + 20;
    }
}

```

```

graphEquations->Series[0]->Points->Clear();
graphEquations->Series[1]->Points->Clear();
graphEquations->Series[2]->Points->Clear();
graphEquations->Series[3]->Points->Clear();

Int16 flagGraph;      // Прапорець, який визначає чи вдалося побудувати
графіки.
Double step = 0.01;   // Крок побудови графіків.
switch (equations->getId()) {
case 1:
    flagGraph = drawGraphArithmetic(a, b, step);
    break;
case 2:
    flagGraph = drawGraphTrigonometric(a, b, step);
    break;
case 3:
    flagGraph = drawGraphTranscendental(a, b, step);
    break;
}

switch (flagGraph) {
case -3:
    MessageBox::Show("Не вдалося побудувати графіки для рівнянь
системи.", "Помилка", MessageBoxButtons::OK, MessageBoxIcon::None);
    graphEquations->Visible = false;
    break;
case -1:
    MessageBox::Show("Не вдалося побудувати графік для першого рівняння
системи.", "Помилка", MessageBoxButtons::OK, MessageBoxIcon::None);
    break;
case -2:
    MessageBox::Show("Не вдалося побудувати графік для другого рівняння
системи.", "Помилка", MessageBoxButtons::OK, MessageBoxIcon::None);
    break;
}
}
System::Int16
CalculatorNamespace::Calculator_Form::drawGraphArithmetic(Double a, Double b,
Double step)
{
    Double x, y1, y2;          // Змінні для збереження координат.
    Boolean flag1 = false,     // Прапорець, який визначає чи вдалося
побудувати графік першого рівняння системи.
    flag2 = false;             // Прапорець, який визначає чи вдалося
побудувати графік другого рівняння системи.
    if ((equations->getCoef()[0][0] > 0 && equations->getCoef()[0][1] < 0 &&
equations->getCoef()[0][2] < 0) ||
        (equations->getCoef()[0][0] < 0 && equations->getCoef()[0][1] > 0
&& equations->getCoef()[0][2] > 0)) {
        x = a;
        while (x <= (b - a) / 2 + a) {
            y1 = equations->getY1(x);
            if (isfinite(y1)) {
                graphEquations->Series[0]->Points->AddXY(x, y1);
                flag1 = true;
            }
        }
    }
}

```

```

        x += step;
    }
    while (x >= a) {
        y1 = -equations->getY1(x);
        if (isfinite(y1)) {
            graphEquations->Series[0]->Points->AddXY(x, y1);
            flag1 = true;
        }
        x -= step;
    }
    x = b;
    while (x >= (b - a) / 2 + a) {
        y1 = equations->getY1(x);
        if (isfinite(y1)) {
            graphEquations->Series[2]->Points->AddXY(x, y1);
            flag1 = true;
        }
        x -= step;
    }
    while (x <= b) {
        y1 = -equations->getY1(x);
        if (isfinite(y1)) {
            graphEquations->Series[2]->Points->AddXY(x, y1);
            flag1 = true;
        }
        x += step;
    }
}
else {
    x = a;
    while (x <= b) {
        y1 = equations->getY1(x);
        if (isfinite(y1)) {
            graphEquations->Series[0]->Points->AddXY(x, y1);
            flag1 = true;
        }
        x += step;
    }
    while (x >= a) {
        y1 = -equations->getY1(x);
        if (isfinite(y1)) {
            graphEquations->Series[2]->Points->AddXY(x, y1);
            flag1 = true;
        }
        x -= step;
    }
}
if ((equations->getCoef()[1][0] > 0 && equations->getCoef()[1][1] < 0 &&
equations->getCoef()[1][2] < 0) ||
    (equations->getCoef()[1][0] < 0 && equations->getCoef()[1][1] > 0
&& equations->getCoef()[1][2] > 0)) {
    x = a;
    while (x <= (b - a) / 2 + a) {
        y2 = equations->getY2(x);
        if (isfinite(y2)) {
            graphEquations->Series[1]->Points->AddXY(x, y2);

```

```

        flag2 = true;
    }
    x += step;
}
while (x >= a) {
    y2 = -equations->getY2(x);
    if (isfinite(y2)) {
        graphEquations->Series[1]->Points->AddXY(x, y2);
        flag2 = true;
    }
    x -= step;
}
x = b;
while (x >= (b - a) / 2 + a) {
    y2 = equations->getY2(x);
    if (isfinite(y2)) {
        graphEquations->Series[3]->Points->AddXY(x, y2);
        flag2 = true;
    }
    x -= step;
}
while (x <= b) {
    y2 = -equations->getY2(x);
    if (isfinite(y2)) {
        graphEquations->Series[3]->Points->AddXY(x, y2);
        flag2 = true;
    }
    x += step;
}
}
else {
    x = a;
    while (x <= b) {
        y2 = equations->getY2(x);
        if (isfinite(y2)) {
            graphEquations->Series[1]->Points->AddXY(x, y2);
            flag2 = true;
        }
        x += step;
    }
    while (x >= a) {
        y2 = -equations->getY2(x);
        if (isfinite(y2)) {
            graphEquations->Series[3]->Points->AddXY(x, y2);
            flag2 = true;
        }
        x -= step;
    }
}
if (!flag1 && !flag2) return -3;
if (!flag1) return -1;
if (!flag2) return -2;
return 2;
}

```

```

System::Int16
CalculatorNamespace::Calculator_Form::drawGraphTrigonometric(Double a, Double
b, Double step)
{
    Double x, y1, y2, x1;           // Змінні для збереження координат.
    Boolean flag1 = false;           // Прапорець, який визначає чи вдалося
побудувати графік першого рівняння системи.
    flag2 = false;                   // Прапорець, який визначає чи вдалося
побудувати графік другого рівняння системи.
    x = a;
    y1 = equations->getY1(x);
    if (isfinite(y1)) {
        while (x <= b) {
            y1 = equations->getY1(x);
            if (isfinite(y1)) {
                graphEquations->Series[0]->Points->AddXY(x, y1);
                flag1 = true;
            }
            y2 = equations->getY2(x);
            if (isfinite(y2)) {
                graphEquations->Series[1]->Points->AddXY(x, y2);
                flag2 = true;
            }
            x += step;
        }
        flag1 = true;
    }
    else {
        x1 = equations->getX1(0);
        if (isfinite(x1)) {
            y2 = equations->getY2(x1);
            if (isfinite(y2)) {
                graphEquations->Series[0]->Points->AddXY(x1, y2 - 5);
                graphEquations->Series[0]->Points->AddXY(x1, y2 + 5);
                flag1 = true;
            }
            else {
                graphEquations->Series[0]->Points->AddXY(x1, -5);
                graphEquations->Series[0]->Points->AddXY(x1, 5);
                flag1 = true;
            }
            x = x1 - 10;
            b = x1 + 10;
        }
        while (x <= b) {
            y2 = equations->getY2(x);
            if (isfinite(y2)) {
                graphEquations->Series[1]->Points->AddXY(x, y2);
                flag2 = true;
            }
            x += step;
        }
    }
    if (!flag1 && !flag2) return -3;
    if (!flag1) return -1;
    if (!flag2) return -2;
}

```

```

        return 2;
    }
    System::Int16
    CalculatorNamespace::Calculator_Form::drawGraphTranscendental(Double a,
    Double b, Double step)
    {
        Double x, y1, y2;           // Змінні для збереження координат.
        Boolean flag1 = false,       // Прапорець, який визначає чи вдалося
        побудувати графік першого рівняння системи.
        flag2 = false;               // Прапорець, який визначає чи вдалося
        побудувати графік другого рівняння системи.
        x = a;
        while (x <= b) {
            y1 = equations->getY1(x);
            if (isfinite(y1)) {
                graphEquations->Series[0]->Points->AddXY(x, y1);
                flag1 = true;
            }
            y2 = equations->getY2(x);
            if (isfinite(y2)) {
                graphEquations->Series[1]->Points->AddXY(x, y2);
                flag2 = true;
            }
            x += step;
        }
        if (!flag1 && !flag2) return -3;
        if (!flag1) return -1;
        if (!flag2) return -2;
        return 2;
    }
    System::Void CalculatorNamespace::Calculator_Form::solvingBegin()
    {
        ResultBox->Visible = true;
        labelSolution->Visible = true;
        tblYourEquation->Visible = true;
        menuSaveResult->Enabled = true;
        menuSaveResultAs->Enabled = true;
        graphEquations->Visible = true;
        tblYourEquation->Text = "Введена система рівнянь:\n" + equations-
        >getEquations();
    }
    System::Void CalculatorNamespace::Calculator_Form::outputResult(Int16 flag)
    {
        switch (flag)
        {
            case -3:
                ResultBox->Text = ("Обраний метод не застосовний для розв'язання
                введеної системи.");
                return;
            case -2:
                ResultBox->Text = "Дана система рівнянь не має розв'язків на множині
                дійсних чисел.";
                return;
            case -1:
                ResultBox->Text = ("Введені вами початкові наближення призводять до
                розбіжного ітераційного процесу, або обраний метод " +

```

```

        "не застосовний до введеної системи.");
        return;
    case 0:
        ResultBox->Text = "Дана система рівнянь має такий можливий
розв'язок:\n      x = " + result[0] + "\n      y = " + result[1];
        return;
    case 2:
        ResultBox->Text = "Дана система рівнянь визначена на всій множині
дійсних чисел.";
        return;
    default:
        outputIterations();
    }
}
System::Void CalculatorNamespace::Calculator_Form::outputIterations()
{
    ResultBox->Text = "Результати ітераційного процесу:";
    Int16 i = 1; // Лічильник циклу.
    for each (Double iter in result)
    {
        if (i % 2 == 1) {
            ResultBox->Text += "\nІтерація " + Math::Floor((Double)i / 2)
+ " :\n      x = " + iter;
        }
        else {
            ResultBox->Text += "\n      y = " + iter;
        }
        i++;
    }
    if (result->Count != 0) {
        ResultBox->Text += "\n\nОтриманий розв'язок:\n      x = "
+ result[result->Count - 2] + "\n      y = " + result[result-
>Count - 1] +
        "\nКількість ітерацій: " + equations->getIterations();
    }
}
System::Boolean
CalculatorNamespace::Calculator_Form::initializeEquationsToSolve()
{
    Double** coef = new Double * [2]; // Масив коефіцієнтів.
    Double* initX = new Double[2]; // Масив початкових наближень.
    for (Int16 i = 0; i < 2; ++i) {
        coef[i] = new Double[3];
    }
    try {
        scanCoefficients(coef);
        scanInitApprox(initX);
    }
    catch (...) {
        if (MessageBox::Show("Не коректний ввід числових даних!\nОчистити
форму?", "Помилка!", MessageBoxButtons::YesNo, MessageBoxIcon::Warning) ==
System::Windows::Forms::DialogResult::Yes)
        {
            clear();
        }
        return false;
    }
}

```

```

    }
    if (btnArithmEq->Checked) {
        equations = (gcnew ArithmeticEquations(coef, initX));
    }
    else if (btnTrigonomEq->Checked) {
        equations = (gcnew TrigonometricEquations(coef, initX));
    }
    else if (btnTranscEq->Checked) {
        equations = (gcnew TranscendentalEquations(coef, initX));
    }
    else {
        MessageBox::Show("Оберіть вид системи!", "Помилка!",
        MessageBoxButtons::OK, MessageBoxIcon::Warning);
        for (Int16 i = 0; i < 2; ++i) {
            delete[] coef[i];
        }
        delete[] coef, initX;
        return false;
    }
    for (Int16 i = 0; i < 2; ++i) {
        delete[] coef[i];
    }
    delete[] coef, initX;
    return true;
}
System::Boolean
CalculatorNamespace::Calculator_Form::initializeEquationsToDrawGraph()
{
    Double** coef = new Double * [2]; // Масив коефіцієнтів
    for (Int16 i = 0; i < 2; ++i) {
        coef[i] = new Double[3];
    }
    try {
        scanCoefficients(coef);
    }
    catch (...) {
        if (MessageBox::Show("Не коректний ввід числових даних!\nОчистити форму?", "Помилка!", MessageBoxButtons::YesNo, MessageBoxIcon::Warning) == System::Windows::Forms::DialogResult::Yes)
        {
            clear();
        }
        return false;
    }
    if (btnArithmEq->Checked) {
        equations = gcnew ArithmeticEquations(coef);
    }
    else if (btnTrigonomEq->Checked) {
        equations = gcnew TrigonometricEquations(coef);
    }
    else if (btnTranscEq->Checked) {
        equations = gcnew TranscendentalEquations(coef);
    }
    else {
        MessageBox::Show("Оберіть вид системи!", "Помилка!",
        MessageBoxButtons::OK, MessageBoxIcon::Warning);

```



```

        for (Int16 i = 0; i < 2; ++i) {
            delete[] coef[i];
        }
        delete[] coef;
        return false;
    }
    for (Int16 i = 0; i < 2; ++i) {
        delete[] coef[i];
    }
    delete[] coef;
    return true;
}
System::Void CalculatorNamespace::Calculator_Form::solve(Double precis)
{
    Int16 flag;           // Прапорець, який визначає, чи вдалося розв'язати
    систему рівнянь.
    if (btnJakobiMethod->Checked) {
        solvingBegin();
        try {
            flag = equations->JakobiMethod(result, precis);
        }
        catch (...) {
            MessageBox::Show("Не вдалося розв'язати вказану систему!",
"Помилка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);
            return;
        }
        outputResult(flag);
        drawGraph(-10, 10, 2);
    }
    else if (btnZeydelMethod->Checked) {
        solvingBegin();
        try {
            flag = equations->ZeydelMethod(result, precis);
        }
        catch (...) {
            MessageBox::Show("Не вдалося розв'язати вказану систему!",
"Помилка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);
            return;
        }
        outputResult(flag);
        drawGraph(-10, 10, 2);
    }
    else {
        MessageBox::Show("Оберіть метод розв'язання!", "Помилка!",
        MessageBoxButtons::OK, MessageBoxIcon::Warning);
        return;
    }
}

```

Файл «Question\_Form.h»

```

#pragma once
using namespace System::Collections::Generic;

namespace Question_FormNamespace {

```

```

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;

/// <summary>
/// Клас для відображення форми, створеної для введення меж побудови
графіків.
/// </summary>
public ref class Question_Form : public System::Windows::Forms::Form
{
    /// <summary>
    /// Створює форму Question_Form та ініціалізує її компоненти.
    /// </summary>
public: Question_Form(void);

    /// <summary>
    /// Видаляє всі використовувані формою Question_Form компоненти.
    /// </summary>
protected: ~Question_Form();

    /// <summary>
    /// Ліва межа побудови графіків (від якого значення координати x потрібно
будувати графік).
    /// </summary>
private: Double x1;
    /// <summary>
    /// Права межа побудови графіків (до якого значення координати x потрібно
будувати графік).
    /// </summary>
private: Double x2;
    /// <summary>
    /// Група, яка містить всі елементи форми Question_Form.
    /// </summary>
private: System::Windows::Forms::GroupBox^ boxQuestion;
    /// <summary>
    /// Поле для введення лівої межі побудови графіків.
    /// </summary>
private: System::Windows::Forms::TextBox^ tblFrom;
    /// <summary>
    /// Поле для введення правої межі побудови графіків.
    /// </summary>
private: System::Windows::Forms::TextBox^ tblTo;
    /// <summary>
    /// Кнопка для підтвердження введення даних та повернення до первинної
(батьківської) форми.
    /// </summary>
private: System::Windows::Forms::Button^ btnAccept;
    /// <summary>
    /// Напис "Від".
    /// </summary>
private: System::Windows::Forms::Label^ labelFrom;
    /// <summary>
    /// Напис "До".
    /// </summary>

```

```

private: System::Windows::Forms::Label^ labelTo;

/// <summary>
///  Необхідна змінна для зберігання компонентів графічного дизайну.
/// </summary>
private: System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
/// <summary>
///  Ініціалізує всі графічні компоненти форми.
/// </summary>
void InitializeComponent(void)
{
    this->boxQuestion = (gcnew
System::Windows::Forms::GroupBox());
    this->tblFrom = (gcnew System::Windows::Forms::TextBox());
    this->tblTo = (gcnew System::Windows::Forms::TextBox());
    this->btnAccept = (gcnew System::Windows::Forms::Button());
    this->labelFrom = (gcnew System::Windows::Forms::Label());
    this->labelTo = (gcnew System::Windows::Forms::Label());
    this->boxQuestion->SuspendLayout();
    this->SuspendLayout();
    //
    //  boxQuestion
    //
    this->boxQuestion->AutoSize = true;
    this->boxQuestion->Controls->Add(this->tblFrom);
    this->boxQuestion->Controls->Add(this->tblTo);
    this->boxQuestion->Controls->Add(this->btnAccept);
    this->boxQuestion->Controls->Add(this->labelFrom);
    this->boxQuestion->Controls->Add(this->labelTo);
    this->boxQuestion->Font = (gcnew
System::Drawing::Font(L"Times New Roman", 12));
    this->boxQuestion->Location = System::Drawing::Point(12, 12);
    this->boxQuestion->Name = L"boxQuestion";
    this->boxQuestion->Size = System::Drawing::Size(350, 197);
    this->boxQuestion->TabIndex = 10;
    this->boxQuestion->TabStop = false;
    this->boxQuestion->Text = L"Введіть межі для побудови
графіків";
    //
    //  tblFrom
    //
    this->tblFrom->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 12));
    this->tblFrom->Location = System::Drawing::Point(14, 66);
    this->tblFrom->Name = L"tblFrom";
    this->tblFrom->Size = System::Drawing::Size(120, 26);
    this->tblFrom->TabIndex = 2;
    this->tblFrom->Text = L"-10";
    //
    //  tblTo
    //
    this->tblTo->Font = (gcnew System::Drawing::Font(L"Times New
Roman", 12));
    this->tblTo->Location = System::Drawing::Point(194, 66);

```

```

        this->tblTo->Name = L"tblTo";
        this->tblTo->Size = System::Drawing::Size(120, 26);
        this->tblTo->TabIndex = 2;
        this->tblTo->Text = L"10";
        //
        // btnAccept
        //
        this->btnAccept->Cursor =
System::Windows::Forms::Cursors::Hand;
        this->btnAccept->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 13));
        this->btnAccept->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->btnAccept->Location = System::Drawing::Point(78, 119);
        this->btnAccept->Name = L"btnAccept";
        this->btnAccept->Size = System::Drawing::Size(170, 49);
        this->btnAccept->TabIndex = 5;
        this->btnAccept->Text = L"Підтвердити";
        this->btnAccept->UseVisualStyleBackColor = true;
        this->btnAccept->Click += gcnew System::EventHandler(this,
&Question_Form::btnAccept_Click);
        //
        // labelFrom
        //
        this->labelFrom->AutoSize = true;
        this->labelFrom->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 12));
        this->labelFrom->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->labelFrom->Location = System::Drawing::Point(10, 35);
        this->labelFrom->Name = L"labelFrom";
        this->labelFrom->Size = System::Drawing::Size(33, 19);
        this->labelFrom->TabIndex = 8;
        this->labelFrom->Text = L"Від:";
        //
        // labelTo
        //
        this->labelTo->AutoSize = true;
        this->labelTo->Font = (gcnew System::Drawing::Font(L"Times
New Roman", 12));
        this->labelTo->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
        this->labelTo->Location = System::Drawing::Point(190, 35);
        this->labelTo->Name = L"labelTo";
        this->labelTo->Size = System::Drawing::Size(31, 19);
        this->labelTo->TabIndex = 8;
        this->labelTo->Text = L"До:";
        //
        // Question_Form
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(9, 19);
        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
        this->BackColor = System::Drawing::SystemColors::Window;
        this->ClientSize = System::Drawing::Size(374, 221);
        this->Controls->Add(this->boxQuestion);

```

```

        this->Font = (gcnew System::Drawing::Font(L"Times New
Roman", 12, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->MaximumSize = System::Drawing::Size(390, 260);
        this->MinimumSize = System::Drawing::Size(390, 260);
        this->Name = L"Question_Form";
        this->SizeGripStyle =
System::Windows::Forms::SizeGripStyle::Hide;
        this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
        this->Text = L"Межі побудови";
        this->boxQuestion->ResumeLayout(false);
        this->boxQuestion->PerformLayout();
        this->ResumeLayout(false);
        this->PerformLayout();

    }
#pragma endregion
    /// <summary>
    /// Сканує введені користувачем межі для побудови графіків і закриває
вікно форми.
    /// </summary>
    /// <param name = 'sender'>Об'єкт класу Object^</param>
    /// <param name = 'e'>Об'єкт класу EventArgs^</param>
    /// <returns>Значення типу Void.</returns>
private: System::Void btnAccept_Click(System::Object^ sender,
System::EventArgs^ e);
    /// <summary>
    /// Повертає значення лівої межі побудови графіків.
    /// </summary>
    /// <returns>Значення типу Double.</returns>
public: Double getXFrom();
    /// <summary>
    /// Повертає значення правої межі побудови графіків.
    /// </summary>
    /// <returns>Значення типу Double.</returns>
public: Double getXTo();
};
}

```

Файл «Question\_Form.cpp»

```

#include "pch.h"
#include "Question_Form.h"
using namespace System;

Question_FormNamespace::Question_Form::Question_Form(void)
{
    InitializeComponent();
    x1 = -10;
    x2 = 10;
}
Question_FormNamespace::Question_Form::~~Question_Form()
{
}

```

```

        if (components)
        {
            delete components;
        }
    }

System::Void
Question_FormNamespace::Question_Form::btnAccept_Click(System::Object^
sender, System::EventArgs^ e)
{
    try {
        x1 = Convert::ToDouble(tblFrom->Text);
        x2 = Convert::ToDouble(tblTo->Text);
    }
    catch (...) {
        MessageBox::Show("Не коректний ввід числових даних!", "Помилка!",
MessageBoxButtons::OK, MessageBoxIcon::Warning);
        return;
    }
    Close();
}

Double Question_FormNamespace::Question_Form::getXFrom()
{
    return x1;
}

Double Question_FormNamespace::Question_Form::getXTo()
{
    return x2;
}

```

Файл «Equations.h»

```

#pragma once
#include "JakobiMatrix.h"
using namespace System;
using namespace System::Collections::Generic;
/// <summary>
/// Клас для відображення системи рівнянь.
/// </summary>
ref class Equations abstract
{
protected:

    /// <summary>
    /// Двовимірний масив коефіцієнтів рівнянь системи.
    /// </summary>
    Double** a;
    /// <summary>
    /// Одновимірний масив початкових наближень розв'язку системи.
    /// </summary>
    Double* x0;
    /// <summary>
    /// Матриця Якобі даної системи.
    /// </summary>

```

```

JakobiMatrix W;
/// <summary>
/// Кількість ітерацій.
/// </summary>
Int16 iterations;

```

```
public:
```

```

/// <summary>
/// Конструктор системи рівнянь за замовчуванням.
/// </summary>
Equations();
/// <summary>
/// Конструктор системи рівнянь із вказаними коефіцієнтами.
/// </summary>
/// <param name = 'coef'>Двовимірний масив коефіцієнтів рівнянь.</param>
Equations(Double** coef);
/// <summary>
/// Конструктор системи рівнянь з вказаними коефіцієнтами та
/// початковим наближенням розв'язку.
/// </summary>
/// <param name = 'coef'>Двовимірний масив коефіцієнтів рівнянь.</param>
/// <param name = 'initX'>Одновимірний масив початкових наближень
розв'язку
/// системи.</param>
Equations(Double** coef, Double* initX);
/// <summary>
/// Очищує динамічну пам'ять виділену на масиви коефіцієнтів та
початкових
/// наближень.
/// </summary>
~Equations();

/// <summary>
/// Розв'язує дану систему методом Якобі.
/// </summary>
/// <param name = 'result'>Масив для запису кінцевого розв'язку та
проміжних
/// результатів ітераційного процесу.</param>
/// <param name = 'precis'>Точність, з якою обчислюється
результат.</param>
/// <returns>-3, якщо розв'язання системи призвело до розбіжного
/// ітераційного процесу; -2, якщо система не має розв'язків; -1, якщо
/// ітераційний процес розбіжний; 0, якщо система має щонайменше один
розв'язок;
/// 1, якщо ітераційний процес збіжний; 2, якщо система має безліч
розв'язків.
/// </returns>
virtual Int16 JakobiMethod(List<Double>^ result, Double precis) = 0;
/// <summary>
/// Розв'язує дану систему методом Гауса-Зейделя.
/// </summary>
/// <param name = 'result'>Масив для запису кінцевого розв'язку та
проміжних
/// результатів ітераційного процесу.</param>

```

```

    /// <param name = 'precis'>Точність, з якою обчислюється
результат.</param>
    /// <returns>-3, якщо розв'язання системи призвело до розбіжного
    /// ітераційного процесу; -2, якщо система не має розв'язків; -1, якщо
    /// ітераційний процес розбіжний; 0, якщо система має щонайменше один
розв'язок;
    /// 1, якщо ітераційний процес збіжний; 2, якщо система має безліч
розв'язків.
    /// </returns>
    virtual Int16 ZeydelMethod(List<Double>^ result, Double precis) = 0;

    /// <summary>
    /// Повертає значення Y з першого рівняння системи у вказаній точці X.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    virtual Double getY1(Double x) = 0;
    /// <summary>
    /// Повертає значення Y з другого рівняння системи у вказаній точці X.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    virtual Double getY2(Double x) = 0;
    /// <summary>
    /// Повертає значення X з першого рівняння системи у вказаній точці Y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    virtual Double getX1(Double y) = 0;
    /// <summary>
    /// Повертає значення X з другого рівняння системи у вказаній точці Y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    virtual Double getX2(Double y) = 0;

    /// <summary>
    /// Повертає номер системи у списку видів систем.
    /// </summary>
    /// <returns>1, якщо система арифметична; 2, якщо система
тригонометрична; 3,
    /// якщо система трансцендентна.</returns>
    virtual Int16 getId() = 0;
    /// <summary>
    /// Повертає систему рівнянь у вигляді двох рядків, які являють собою
загальний
    /// вид системи, але з вказаними коефіцієнтами.
    /// </summary>
    /// <returns>Значення типу String^</returns>
    virtual String^ getEquations() = 0;
    /// <summary>
    /// Повертає кількість ітерацій, зроблених при обчисленні точного
розв'язку
    /// даної системи рівнянь.
    /// </summary>
    /// <returns>Значення типу Int16.</returns>

```



```

Int16 getIterations();
/// <summary>
/// Повертає масив коефіцієнтів рівнянь даної системи.
/// </summary>
/// <returns>Двовимірний динамічний масив типу Double.</returns>
Double** getCoef();

```

protected:

```

/// <summary>
/// Перевіряє ітераційний процес на збіжність при заданому початковому
наближенні.
/// </summary>
/// <param name = 'i'>Прапорець, який визначає, за якою формулою потрібно
перетворити
/// початкові рівняння для початку ітераційного процесу.</param>
/// <returns>false, якщо ітераційний процес розбіжний; true, якщо
ітераційний процес
/// збіжний.</returns>
virtual Boolean checkInitApprox(int i) = 0;
/// <summary>
/// Повертає значення похідної по X першого рівняння системи у точці x.
/// </summary>
/// <param name = 'x'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
virtual Double getY1dX(Double x) = 0;
/// <summary>
/// Повертає значення похідної по X другого рівняння системи у точці x.
/// </summary>
/// <param name = 'x'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
virtual Double getY2dX(Double x) = 0;
/// <summary>
/// Повертає значення похідної по Y першого рівняння системи у точці y.
/// </summary>
/// <param name = 'y'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
virtual Double getX1dY(Double y) = 0;
/// <summary>
/// Повертає значення похідної по Y другого рівняння системи у точці y.
/// </summary>
/// <param name = 'y'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
virtual Double getX2dY(Double y) = 0;
};

```

Файл «Equations.cpp»

```

#include "pch.h"
#include "Equations.h"

Equations::Equations()
{
    a = new Double * [2];

```

```

    x0 = new Double[2];
    for (Int16 i = 0; i < 2; ++i) {
        a[i] = new Double[3];
    }
}
Equations::Equations(Double** coef)
{
    a = new Double * [2];
    x0 = nullptr;
    for (Int16 i = 0; i < 2; ++i) {
        a[i] = new Double[3];
    }
    for (Int16 i = 0; i < 2; ++i) {
        for (Int16 j = 0; j < 3; ++j) {
            a[i][j] = coef[i][j];
        }
    }
}
Equations::Equations(Double** coef, Double* initX)
{
    a = new Double * [2];
    x0 = new Double[2];
    for (Int16 i = 0; i < 2; ++i) {
        a[i] = new Double[3];
    }
    for (Int16 i = 0; i < 2; ++i) {
        for (Int16 j = 0; j < 3; ++j) {
            a[i][j] = coef[i][j];
        }
        x0[i] = initX[i];
    }
}

Equations::~~Equations()
{
    for (Int16 i = 0; i < 2; ++i) {
        delete[] a[i];
    }
    delete[] a, x0;
}

Int16 Equations::getIterations()
{
    return iterations;
}
Double** Equations::getCoef()
{
    return a;
}

```

Файл «ArithmeticEquations.h»

```

#pragma once
#include "Equations.h"
/// <summary>

```

```

/// Клас для відображення системи арифметичних рівнянь:
///  $a_{11}x^2 + a_{12}y^2 + a_{13} = 0$ 
///  $a_{21}x^2 + a_{22}y^2 + a_{23} = 0$ 
/// </summary>
ref class ArithmeticEquations : public Equations
{
public:
    /// <summary>
    /// Конструктор системи арифметичних рівнянь за замовчуванням.
    /// </summary>
    ArithmeticEquations();
    /// <summary>
    /// Конструктор системи арифметичних рівнянь з вказаними коефіцієнтами.
    /// </summary>
    /// <param name = 'coef'>Двовимірний масив коефіцієнтів рівнянь.</param>
    ArithmeticEquations(Double** coef);
    /// <summary>
    /// Конструктор системи арифметичних рівнянь з вказаними коефіцієнтами
    /// та початковим наближенням розв'язку.
    /// </summary>
    /// <param name = 'coef'>Двовимірний масив коефіцієнтів рівнянь.</param>
    /// <param name = 'initX'>Одновимірний масив початкових наближень
розв'язку
    /// системи.</param>
    ArithmeticEquations(Double** coef, Double* initX);

    /// <summary>
    /// Розв'язує дану арифметичну систему методом Якобі.
    /// </summary>
    /// <param name = 'result'>Масив для запису кінцевого розв'язку та
проміжних
    /// результатів ітераційного процесу.</param>
    /// <param name = 'precis'>Точність, з якою обчислюється
результат.</param>
    /// <returns>-3, якщо обраний метод не застосовний для розв'язання
введеної системи;
    /// -2, якщо система не має розв'язків; -1, якщо ітераційний процес
розбіжний; 0,
    /// якщо система має щонайменше один розв'язок; 1, якщо ітераційний
процес збіжний;
    /// 2, якщо система має безліч розв'язків.</returns>
    Int16 JakobiMethod(List<Double>^ result, Double precis) override;
    /// <summary>
    /// Розв'язує дану арифметичну систему методом Гауса-Зейделя.
    /// </summary>
    /// <param name = 'result'>Масив для запису кінцевого розв'язку та
проміжних
    /// результатів ітераційного процесу.</param>
    /// <param name = 'precis'>Точність, з якою обчислюється
результат.</param>
    /// <returns>-3, якщо обраний метод не застосовний для розв'язання
введеної системи;
    /// -2, якщо система не має розв'язків; -1, якщо ітераційний процес
розбіжний; 0,

```

```

    /// якщо система має щонайменше один розв'язок; 1, якщо ітераційний
    процес збіжний;
    /// 2, якщо система має безліч розв'язків.</returns>
    Int16 ZeydelMethod(List<Double>^ result, Double precis) override;

    /// <summary>
    /// Повертає "номер" системи у списку видів систем.
    /// </summary>
    /// <returns>1, якщо система арифметична; 2, якщо система
    тригонометрична; 3, якщо система трансцендентна.</returns>
    Int16 getId() override;
    /// <summary>
    /// Повертає систему рівнянь у вигляді двох рядків, які являють собою
    загальний
    /// вид системи, але з вказаними коефіцієнтами.
    /// </summary>
    /// <returns>Значення типу String^</returns>
    String^ getEquations() override;

    /// <summary>
    /// Повертає значення Y з першого рівняння системи у вказаній точці X.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getY1(Double x) override;
    /// <summary>
    /// Повертає значення Y з другого рівняння системи у вказаній точці X.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getY2(Double x) override;
    /// <summary>
    /// Повертає значення X з першого рівняння системи у вказаній точці Y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getX1(Double y) override;
    /// <summary>
    /// Повертає значення X з другого рівняння системи у вказаній точці Y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getX2(Double y) override;

```

protected:

```

    /// <summary>
    /// Перевіряє ітераційний процес на збіжність при заданому початковому
    наближенні.
    /// </summary>
    /// <param name = 'i'>Прапорець, який визначає, за якою формулою потрібно
    перетворити
    /// початкові рівняння для початку ітераційного процесу.</param>
    /// <returns>false, якщо ітераційний процес розбіжний; true, якщо
    ітераційний процес
    /// збіжний.</returns>

```

```

Boolean checkInitApprox(int i) override;
/// <summary>
/// Повертає значення похідної по X першого рівняння системи у точці x.
/// </summary>
/// <param name = 'x'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
Double getY1dX(Double x) override;
/// <summary>
/// Повертає значення похідної по X другого рівняння системи у точці x.
/// </summary>
/// <param name = 'x'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
Double getY2dX(Double x) override;
/// <summary>
/// Повертає значення похідної по Y першого рівняння системи у точці y.
/// </summary>
/// <param name = 'y'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
Double getX1dY(Double y) override;
/// <summary>
/// Повертає значення похідної по Y другого рівняння системи у точці y.
/// </summary>
/// <param name = 'y'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
Double getX2dY(Double y) override;
};

```

Файл «ArithmeticEquations.cpp»

```

#include "pch.h"
#include "ArithmeticEquations.h"
#include <Math.h>
using namespace System;

ArithmeticEquations::ArithmeticEquations() : Equations() {}
ArithmeticEquations::ArithmeticEquations(Double** coef) : Equations(coef) {}
ArithmeticEquations::ArithmeticEquations(Double** coef, Double* initX) :
Equations(coef, initX) {}

Boolean ArithmeticEquations::checkInitApprox(int i)
{
    W(0, 0) = 0;
    W(1, 1) = 0;
    switch (i) {
    case 1:
        W(0, 1) = Math::Abs(getX1dY(x0[1]));
        W(1, 0) = Math::Abs(getY2dX(x0[0]));
        break;
    case 2:
        W(0, 1) = Math::Abs(getX2dY(x0[1]));
        W(1, 0) = Math::Abs(getY1dX(x0[0]));
        break;
    }
    if (!isfinite(W(0, 1)) || !isfinite(W(1, 0))) return false;
}

```

```

        if (W.getNormM() < 1 && W.getNormL() < 1) {
            return true;
        }
        return false;
    }

    Int16 ArithmeticEquations::JakobiMethod(List<Double>^ result, Double precis)
    {
        iterations = 0;
        result->Clear();
        if (a[0][0] == 0 && a[0][1] == 0 && a[0][2] == 0 && a[1][0] == 0 &&
a[1][1] == 0 && a[1][2] == 0) {
            return 2;
        }
        if ((a[0][0] * a[0][1] > 0 && a[0][0] * a[0][2] > 0 && a[0][1] * a[0][2]
> 0) ||
(a[1][0] * a[1][1] > 0 && a[1][0] * a[1][2] > 0 && a[1][1] * a[1][2]
> 0) ||
(a[0][0] == 0 && a[0][1] == 0 && a[0][2] != 0) ||
(a[1][0] == 0 && a[1][1] == 0 && a[1][2] != 0) ||
(a[0][0] * a[0][1] > 0 && a[0][2] == 0 && a[1][2] != 0) ||
(a[1][0] * a[1][1] > 0 && a[1][2] == 0 && a[0][2] != 0) ||
(a[0][0] * a[0][2] > 0 && a[0][1] == 0) ||
(a[0][1] * a[0][2] > 0 && a[0][0] == 0) ||
(a[1][0] * a[1][2] > 0 && a[1][1] == 0) ||
(a[1][1] * a[1][2] > 0 && a[1][0] == 0)) {
            return -2;
        }
        if (a[0][2] == 0 && a[1][2] == 0 && (a[0][0] * a[0][1] > 0 || a[1][0] *
a[1][1] > 0)) {
            result->Add(0);
            result->Add(0);
            return 0;
        }
        if (-(a[0][0] + a[0][1]) == a[0][2] && -(a[1][0] + a[1][1]) == a[1][2])
    {
        if (x0[0] >= 0) result->Add(1);
        else result->Add(-1);
        if (x0[1] >= 0) result->Add(1);
        else result->Add(-1);
        return 0;
    }
}

    Double num1, num2;           // Змінні для тимчасового збереження
результатів обчислення коренів.
    Boolean flag = true;         // Прапорець, який визначає, чи вдалося
розв'язати систему рівнянь.
    Double delta;                // Різниця між значеннями коренів на сусідніх
ітераціях.
    Int32 i;                     // Лічильник ітерацій.

/* 1 */
    if (a[0][0] != 0 && a[1][1] != 0 && checkInitApprox(1)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
    }

```

```

        while (true) {
            iterations++;
            num1 = getX1(result[i - 1]);
            num2 = getY2(result[i - 2]);
            if (!isfinite(num1) || !isfinite(num2)) {
                result->Clear();
                flag = false;
                break;
            }
            if (x0[0] >= 0) result->Add(num1);
            else result->Add(-num1);
            if (x0[1] >= 0) result->Add(num2);
            else result->Add(-num2);
            delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
            if (delta <= precis) {
                return 1;
            }
            i += 2;
        }
    }

/* 2 */
    if (a[1][0] != 0 && a[0][1] != 0 && checkInitApprox(2)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
        while (true) {
            iterations++;
            num1 = getX2(result[i - 1]);
            num2 = getY1(result[i - 2]);
            if (!isfinite(num1) || !isfinite(num2)) {
                result->Clear();
                return -3;
            }
            if (x0[0] >= 0) result->Add(num1);
            else result->Add(-num1);
            if (x0[1] >= 0) result->Add(num2);
            else result->Add(-num2);
            delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
            if (delta <= precis) {
                return 1;
            }
            i += 2;
        }
    }
    if (flag) return -1;
    return -3;
}

Int16 ArithmeticEquations::ZeydelMethod(List<Double>^ result, Double precis)
{
    iterations = 0;
    result->Clear();
    if (a[0][0] == 0 && a[0][1] == 0 && a[0][2] == 0 && a[1][0] == 0 &&
a[1][1] == 0 && a[1][2] == 0) {

```

```

        return 2;
    }
    if ((a[0][0] * a[0][1] > 0 && a[0][0] * a[0][2] > 0 && a[0][1] * a[0][2]
> 0) ||
        (a[1][0] * a[1][1] > 0 && a[1][0] * a[1][2] > 0 && a[1][1] * a[1][2]
> 0) ||
        (a[0][0] == 0 && a[0][1] == 0 && a[0][2] != 0) ||
        (a[1][0] == 0 && a[1][1] == 0 && a[1][2] != 0) ||
        (a[0][0] * a[0][1] > 0 && a[0][2] == 0 && a[1][2] != 0) ||
        (a[1][0] * a[1][1] > 0 && a[1][2] == 0 && a[0][2] != 0) ||
        (a[0][0] * a[0][2] > 0 && a[0][1] == 0) ||
        (a[0][1] * a[0][2] > 0 && a[0][0] == 0) ||
        (a[1][0] * a[1][2] > 0 && a[1][1] == 0) ||
        (a[1][1] * a[1][2] > 0 && a[1][0] == 0)) {
        return -2;
    }
    if (a[0][2] == 0 && a[1][2] == 0 && (a[0][0] * a[0][1] > 0 || a[1][0] *
a[1][1] > 0)) {
        result->Add(0);
        result->Add(0);
        return 0;
    }
    if (-(a[0][0] + a[0][1]) == a[0][2] && -(a[1][0] + a[1][1]) == a[1][2])
{
        if (x0[0] >= 0) result->Add(1);
        else result->Add(-1);
        if (x0[1] >= 0) result->Add(1);
        else result->Add(-1);
        return 0;
    }

    Double num1, num2;           // Змінні для тимчасового збереження
результатів обчислення коренів.
    Boolean flag = true;         // Прапорець, який визначає, чи вдалося
розв'язати систему рівнянь.
    Double delta;                // Різниця між значеннями коренів на сусідніх
ітераціях.
    Int32 i;                     // Лічильник ітерацій.

/* 1 */
    if (a[0][0] != 0 && a[1][1] != 0 && checkInitApprox(1)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
        while (true) {
            iterations++;
            num1 = getX1(result[i - 1]);
            if (!isfinite(num1)) {
                result->Clear();
                flag = false;
                break;
            }
            if (x0[0] >= 0) result->Add(num1);
            else result->Add(-num1);

            num2 = getY2(result[i]);

```



```

        if (!isfinite(num2)) {
            result->Clear();
            flag = false;
            break;
        }
        if (x0[1] >= 0) result->Add(num2);
        else result->Add(-num2);
        delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
        if (delta <= precis) {
            return 1;
        }
        i += 2;
    }
}

/* 2 */
if (a[1][0] != 0 && a[0][1] != 0 && checkInitApprox(2)) {
    result->Add(x0[0]);
    result->Add(x0[1]);
    i = 2;
    while (true) {
        iterations++;
        num1 = getX2(result[i - 1]);
        if (!isfinite(num1)) {
            result->Clear();
            return -3;
        }
        if (x0[0] >= 0) result->Add(num1);
        else result->Add(-num1);

        num2 = getY1(result[i]);
        if (!isfinite(num2)) {
            result->Clear();
            return -3;
        }
        if (x0[1] >= 0) result->Add(num2);
        else result->Add(-num2);
        delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
        if (delta <= precis) {
            return 1;
        }
        i += 2;
    }
}
if (flag) return -1;
return -3;
}

Int16 ArithmeticEquations::getId()
{
    return 1;
}
String^ ArithmeticEquations::getEquations()
{

```

```

        String^ eq = (" " + Convert::ToString(a[0][0]) + " * x ^ 2 + " +
Convert::ToString(a[0][1]) + " * y ^ 2 + " +
        Convert::ToString(a[0][2]) + " = 0\n " +
Convert::ToString(a[1][0]) + " * x ^ 2 + " + Convert::ToString(a[1][1]) +
        " * y ^ 2 + " + Convert::ToString(a[1][2]) + " = 0");
        return eq;
    }

    Double ArithmeticEquations::getY1(Double x)
    {
        if (a[0][1] == 0) return NAN;
        Double num = (-a[0][0] * Math::Pow(x, 2) - a[0][2]) / a[0][1];
        if (num < 0) return NAN;
        return Math::Sqrt(num);
    }

    Double ArithmeticEquations::getY2(Double x)
    {
        if (a[1][1] == 0) return NAN;
        Double num = (-a[1][0] * Math::Pow(x, 2) - a[1][2]) / a[1][1];
        if (num < 0) return NAN;
        return Math::Sqrt(num);
    }

    Double ArithmeticEquations::getX1(Double y)
    {
        if (a[0][0] == 0) return NAN;
        Double num = (-a[0][1] * Math::Pow(y, 2) - a[0][2]) / a[0][0];
        if (num < 0) return NAN;
        return Math::Sqrt(num);
    }

    Double ArithmeticEquations::getX2(Double y)
    {
        if (a[1][0] == 0) return NAN;
        Double num = (-a[1][1] * Math::Pow(y, 2) - a[1][2]) / a[1][0];
        if (num < 0) return NAN;
        return Math::Sqrt(num);
    }

    Double ArithmeticEquations::getY1dX(Double x)
    {
        Double num = a[0][1] * (-a[0][0] * Math::Pow(x, 2) - a[0][2]);
        if (num <= 0) return NAN;
        return -a[0][0] * x / Math::Sqrt(num);
    }

    Double ArithmeticEquations::getY2dX(Double x)
    {
        Double num = a[1][1] * (-a[1][0] * Math::Pow(x, 2) - a[1][2]);
        if (num <= 0) return NAN;
        return -a[1][0] * x / Math::Sqrt(num);
    }

    Double ArithmeticEquations::getX1dY(Double y)
    {
        Double num = a[0][0] * (-a[0][1] * Math::Pow(y, 2) - a[0][2]);
        if (num <= 0) return NAN;
        return -a[0][1] * y / Math::Sqrt(num);
    }

    Double ArithmeticEquations::getX2dY(Double y)

```

```

{
    Double num = a[1][0] * (-a[1][1] * Math::Pow(y, 2) - a[1][2]);
    if (num <= 0) return NAN;
    return -a[1][1] * y / Math::Sqrt(num);
}

```

Файл «TrigonometricEquations.h»

```

#pragma once
#include "Equations.h"
/// <summary>
/// Клас для відображення системи тригонометричних рівнянь:
/// sin(x+a11)+a12*y+a13=0
/// a21*x+cos(y+a22)+a23=0
/// </summary>
ref class TrigonometricEquations : public Equations
{
public:
    /// <summary>
    /// Конструктор системи тригонометричних рівнянь за замовчуванням.
    /// </summary>
    TrigonometricEquations();
    /// <summary>
    /// Конструктор системи тригонометричних рівнянь з вказаними
    коефіцієнтами.
    /// </summary>
    /// <param name = 'coef'>Двовимірний масив коефіцієнтів рівнянь.</param>
    TrigonometricEquations(Double** coef);
    /// <summary>
    /// Конструктор системи тригонометричних рівнянь з вказаними
    коефіцієнтами
    /// та початковим наближенням розв'язку.
    /// </summary>
    /// <param name = 'coef'>Двовимірний масив коефіцієнтів рівнянь.</param>
    /// <param name = 'initX'>Одновимірний масив початкових наближень
    розв'язку
    /// системи.</param>
    TrigonometricEquations(Double** coef, Double* initX);

    /// <summary>
    /// Розв'язує дану тригонометричну систему методом Якобі.
    /// </summary>
    /// <param name = 'result'>Масив для запису кінцевого розв'язку та
    проміжних
    /// результатів ітераційного процесу.</param>
    /// <param name = 'precis'>Точність, з якою обчислюється
    результат.</param>
    /// <returns>-3, якщо обраний метод не застосовний для розв'язання
    введеної системи;
    /// -2, якщо система не має розв'язків; -1, якщо ітераційний процес
    розбіжний; 0,
    /// якщо система має щонайменше один розв'язок; 1, якщо ітераційний
    процес збіжний;
    /// 2, якщо система має безліч розв'язків.</returns>

```

```

Int16 JakobiMethod(List<Double>^ result, Double precis) override;
/// <summary>
/// Розв'язує дану тригонометричну систему методом Гауса-Зейделя.
/// </summary>
/// <param name = 'result'>Масив для запису кінцевого розв'язку та
проміжних
/// результатів ітераційного процесу.</param>
/// <param name = 'precis'>Точність, з якою обчислюється
результат.</param>
/// <returns>-3, якщо обраний метод не застосовний для розв'язання
введеної системи;
/// -2, якщо система не має розв'язків; -1, якщо ітераційний процес
розбіжний; 0,
/// якщо система має щонайменше один розв'язок; 1, якщо ітераційний
процес збіжний;
/// 2, якщо система має безліч розв'язків.</returns>
Int16 ZeydelMethod(List<Double>^ result, Double precis) override;

/// <summary>
/// Повертає "номер" системи у списку видів систем.
/// </summary>
/// <returns>1, якщо система арифметична; 2, якщо система
тригонометрична; 3, якщо система трансцендентна.</returns>
Int16 getId() override;
/// <summary>
/// Повертає систему рівнянь у вигляді двох рядків, які являють собою
загальний
/// вид системи, але з вказаними коефіцієнтами.
/// </summary>
/// <returns>Значення типу String^</returns>
String^ getEquations() override;

/// <summary>
/// Повертає значення Y з першого рівняння системи у вказаній точці X.
/// </summary>
/// <param name = 'x'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
Double getY1(Double x) override;
/// <summary>
/// Повертає значення Y з другого рівняння системи у вказаній точці X.
/// </summary>
/// <param name = 'x'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
Double getY2(Double x) override;
/// <summary>
/// Повертає значення X з першого рівняння системи у вказаній точці Y.
/// </summary>
/// <param name = 'y'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
Double getX1(Double y) override;
/// <summary>
/// Повертає значення X з другого рівняння системи у вказаній точці Y.
/// </summary>
/// <param name = 'y'>Аргумент функції.</param>
/// <returns>Значення типу Double.</returns>
Double getX2(Double y) override;

```

protected:

```

    /// <summary>
    /// Перевіряє ітераційний процес на збіжність при заданому початковому
наближенні.
    /// </summary>
    /// <param name = 'i'>Прапорець, який визначає, за якою формулою потрібно
перетворити
    /// початкові рівняння для початку ітераційного процесу.</param>
    /// <returns>false, якщо ітераційний процес розбіжний; true, якщо
ітераційний процес
    /// збіжний.</returns>
    Boolean checkInitApprox(int i) override;
    /// <summary>
    /// Повертає значення похідної по X першого рівняння системи у точці x.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getY1dX(Double x) override;
    /// <summary>
    /// Повертає значення похідної по X другого рівняння системи у точці x.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getY2dX(Double x) override;
    /// <summary>
    /// Повертає значення похідної по Y першого рівняння системи у точці y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getX1dY(Double y) override;
    /// <summary>
    /// Повертає значення похідної по Y другого рівняння системи у точці y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getX2dY(Double y) override;
};

```

Файл «TrigonometricEquations.cpp»

```

#include "pch.h"
#include "TrigonometricEquations.h"
#include <Math.h>
using namespace System;

TrigonometricEquations::TrigonometricEquations() : Equations() {}
TrigonometricEquations::TrigonometricEquations(Double** coef) :
Equations(coef) {}
TrigonometricEquations::TrigonometricEquations(Double** coef, Double* initX)
: Equations(coef, initX) {}

Boolean TrigonometricEquations::checkInitApprox(int i)
{

```

```

W(0, 0) = 0;
W(1, 1) = 0;
switch (i) {
case 1:
    W(0, 1) = Math::Abs(getX2dY(x0[1]));
    W(1, 0) = Math::Abs(getY1dX(x0[0]));
    break;
case 2:
    W(0, 1) = Math::Abs(getX1dY(x0[1]));
    W(1, 0) = Math::Abs(getY2dX(x0[0]));
    break;
}
if (!isfinite(W(0, 1)) || !isfinite(W(1, 0))) return false;

if (W.getNormM() < 1 && W.getNormL() < 1) {
    return true;
}
return false;
}

Int16 TrigonometricEquations::JakobiMethod(List<Double>^ result, Double
precis)
{
    if ((a[0][1] == 0 && Math::Abs(a[0][2]) > 1) ||
        (a[1][0] == 0 && Math::Abs(a[1][2]) > 1)) {
        return -2;
    }
    if (a[0][1] == 0 && a[0][2] == 0 && a[1][0] == 0 && a[1][1] == 0 &&
a[1][2] == 0) {
        result->Add(-a[0][0]);
        result->Add(Math::Acos(0));
        return 0;
    }

    Double num1, num2;          // Змінні для тимчасового збереження
результатів обчислення коренів.
    Double delta;               // Різниця між значеннями коренів на сусідніх
ітераціях.
    Int32 i;                    // Лічильник ітерацій.

    /* 1 */
    if (a[1][0] != 0 && a[0][1] != 0 && checkInitApprox(1)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
        while (true) {
            iterations++;
            result->Add(getX2(result[i - 1]));
            result->Add(getY1(result[i - 2]));
            delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
            if (delta <= precis) {
                return 1;
            }
            i += 2;
        }
    }
}

```

```

    }

/* 2 */
    if (checkInitApprox(2)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
        while (true) {
            iterations++;
            num1 = getX1(result[i - 1]);
            num2 = getY2(result[i - 2]);
            if (!isfinite(num1) || !isfinite(num2)) {
                result->Clear();
                return -3;
            }
            result->Add(num1);
            result->Add(num2);
            delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
            if (delta <= precis) {
                return 1;
            }
            i += 2;
        }
    }
    return -1;
}

Int16 TrigonometricEquations::ZeydelMethod(List<Double>^ result, Double
precis)
{
    if ((a[0][1] == 0 && Math::Abs(a[0][2]) > 1) ||
        (a[1][0] == 0 && Math::Abs(a[1][2]) > 1)) {
        return -2;
    }
    if (a[0][1] == 0 && a[0][2] == 0 && a[1][0] == 0 && a[1][1] == 0 &&
a[1][2] == 0) {
        result->Add(-a[0][0]);
        result->Add(Math::Acos(0));
        return 0;
    }

    Double num1, num2;          // Змінні для тимчасового збереження
результатів обчислення коренів.
    Double delta;                // Різниця між значеннями коренів на сусідніх
ітераціях.
    Int32 i;                     // Лічильник ітерацій.

/* 1 */
    if (a[1][0] != 0 && a[0][1] != 0 && checkInitApprox(1)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
        while (true) {
            iterations++;
            result->Add(getX2(result[i - 1]));
            result->Add(getY1(result[i]));

```

```

        delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
        if (delta <= precis) {
            return 1;
        }
        i += 2;
    }
}

/* 2 */
if (checkInitApprox(2)) {
    result->Add(x0[0]);
    result->Add(x0[1]);
    i = 2;
    while (true) {
        iterations++;
        num1 = getX1(result[i - 1]);
        if (!isfinite(num1)) {
            result->Clear();
            return -3;
        }
        result->Add(num1);

        num2 = getY2(result[i]);
        if (!isfinite(num2)) {
            result->Clear();
            return -3;
        }
        result->Add(num2);
        delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
        if (delta <= precis) {
            return 1;
        }
        i += 2;
    }
}
return -1;
}

Int16 TrigonometricEquations::getId()
{
    return 2;
}

String^ TrigonometricEquations::getEquations()
{
    String^ eq = (" sin(x + " + Convert::ToString(a[0][0]) + ") + " +
Convert::ToString(a[0][1]) + " * y + " +
        Convert::ToString(a[0][2]) + " = 0\n " +
Convert::ToString(a[1][0]) + " * x + cos(y + " + Convert::ToString(a[1][1])
+
        ") + " + Convert::ToString(a[1][2]) + " = 0");
    return eq;
}

Double TrigonometricEquations::getY1(Double x)

```



```

{
    if (a[0][1] == 0) return NAN;
    return -(Math::Sin(x + a[0][0]) + a[0][2]) / a[0][1];
}
Double TrigonometricEquations::getY2(Double x)
{
    return Math::Acos(-a[1][0] * x - a[1][2]) - a[1][1];
}
Double TrigonometricEquations::getX1(Double y)
{
    return Math::Asin(-a[0][1] * y - a[0][2]) - a[0][0];
}
Double TrigonometricEquations::getX2(Double y)
{
    if (a[1][0] == 0) return NAN;
    return -(Math::Cos(y + a[1][1]) + a[1][2]) / a[1][0];
}

Double TrigonometricEquations::getY1dX(Double x)
{
    if (a[0][1] == 0) return NAN;
    return -Math::Cos(x + a[0][0]) / a[0][1];
}
Double TrigonometricEquations::getY2dX(Double x)
{
    Double num = 1 - Math::Pow(a[1][0] * x + a[1][2], 2);
    if (num <= 0) return NAN;
    return a[1][0] / Math::Sqrt(num);
}
Double TrigonometricEquations::getX1dY(Double y)
{
    Double num = 1 - Math::Pow(a[0][1] * y + a[0][2], 2);
    if (num <= 0) return NAN;
    return -a[0][1] / Math::Sqrt(num);
}
Double TrigonometricEquations::getX2dY(Double y)
{
    if (a[1][0] == 0) return NAN;
    return Math::Sin(y + a[1][1]) / a[1][0];
}

```

Файл «TranscendentalEquations.h»

```

#pragma once
#include "Equations.h"
/// <summary>
/// Клас для відображення системи трансцендентних рівнянь:
///  $e^{(a_{11}x)} + e^{(a_{12}y)} + a_{13} = 0$ 
///  $e^{(a_{21}x)} + e^{(a_{22}y)} + a_{23} = 0$ 
/// </summary>
ref class TranscendentalEquations : public Equations
{
public:
    /// <summary>

```

```

    /// Конструктор системи трансцендентних рівнянь за замовчуванням.
    /// </summary>
    TranscendentalEquations();
    /// <summary>
    /// Конструктор системи трансцендентних рівнянь з вказаними
коєфіцієнтами.
    /// </summary>
    /// <param name = 'coef'>Двовимірний масив коєфіцієнтів рівнянь.</param>
    TranscendentalEquations(Double** coef);
    /// <summary>
    /// Конструктор системи трансцендентних рівнянь з вказаними коєфіцієнтами
    /// та початковим наближенням розв'язку.
    /// </summary>
    /// <param name = 'coef'>Двовимірний масив коєфіцієнтів рівнянь.</param>
    /// <param name = 'initX'>Одновимірний масив початкових наближень
розв'язку
    /// системи.</param>
    TranscendentalEquations(Double** coef, Double* initX);

    /// <summary>
    /// Розв'язує дану трансцендентну систему методом Якобі.
    /// </summary>
    /// <param name = 'result'>Масив для запису кінцевого розв'язку та
проміжних
    /// результатів ітераційного процесу.</param>
    /// <param name = 'precis'>Точність, з якою обчислюється
результат.</param>
    /// <returns>-3, якщо обраний метод не застосовний для розв'язання
введеної системи;
    /// -2, якщо система не має розв'язків; -1, якщо ітераційний процес
розбіжний; 0,
    /// якщо система має щонайменше один розв'язок; 1, якщо ітераційний
процес збіжний;
    /// 2, якщо система має безліч розв'язків.</returns>
    Int16 JakobiMethod(List<Double>^ result, Double precis) override;
    /// <summary>
    /// Розв'язує дану трансцендентну систему методом Гауса-Зейделя.
    /// </summary>
    /// <param name = 'result'>Масив для запису кінцевого розв'язку та
проміжних
    /// результатів ітераційного процесу.</param>
    /// <param name = 'precis'>Точність, з якою обчислюється
результат.</param>
    /// <returns>-3, якщо обраний метод не застосовний для розв'язання
введеної системи;
    /// -2, якщо система не має розв'язків; -1, якщо ітераційний процес
розбіжний; 0,
    /// якщо система має щонайменше один розв'язок; 1, якщо ітераційний
процес збіжний;
    /// 2, якщо система має безліч розв'язків.</returns>
    Int16 ZeydelMethod(List<Double>^ result, Double precis) override;

    /// <summary>
    /// Повертає "номер" системи у списку видів систем.
    /// </summary>

```

```

    /// <returns>1, якщо система арифметична; 2, якщо система
тригонометрична; 3, якщо система трансцендентна.</returns>
    Int16 getId() override;
    /// <summary>
    /// Повертає систему рівнянь у вигляді двох рядків, які являють собою
загальний
    /// вид системи, але з вказаними коефіцієнтами.
    /// </summary>
    /// <returns>Значення типу String^</returns>
    String^ getEquations() override;

    /// <summary>
    /// Повертає значення Y з першого рівняння системи у вказаній точці X.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getY1(Double x) override;
    /// <summary>
    /// Повертає значення Y з другого рівняння системи у вказаній точці X.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getY2(Double x) override;
    /// <summary>
    /// Повертає значення X з першого рівняння системи у вказаній точці Y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getX1(Double y) override;
    /// <summary>
    /// Повертає значення X з другого рівняння системи у вказаній точці Y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getX2(Double y) override;

```

protected:

```

    /// <summary>
    /// Перевіряє ітераційний процес на збіжність при заданому початковому
наближенні.
    /// </summary>
    /// <param name = 'i'>Прапорець, який визначає, за якою формулою потрібно
перетворити
    /// початкові рівняння для початку ітераційного процесу.</param>
    /// <returns>false, якщо ітераційний процес розбіжний; true, якщо
ітераційний процес
    /// збіжний.</returns>
    Boolean checkInitApprox(int i) override;
    /// <summary>
    /// Повертає значення похідної по X першого рівняння системи у точці x.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getY1dX(Double x) override;
    /// <summary>

```

```

    /// Повертає значення похідної по X другого рівняння системи у точці x.
    /// </summary>
    /// <param name = 'x'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getY2dX(Double x) override;
    /// <summary>
    /// Повертає значення похідної по Y першого рівняння системи у точці y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getX1dY(Double y) override;
    /// <summary>
    /// Повертає значення похідної по Y другого рівняння системи у точці y.
    /// </summary>
    /// <param name = 'y'>Аргумент функції.</param>
    /// <returns>Значення типу Double.</returns>
    Double getX2dY(Double y) override;
};

```

Файл «TranscendentalEquations.cpp»

```

#include "pch.h"
#include "TranscendentalEquations.h"
#include <Math.h>
using namespace System;

TranscendentalEquations::TranscendentalEquations() : Equations() {}
TranscendentalEquations::TranscendentalEquations(Double** coef) :
Equations(coef) {}
TranscendentalEquations::TranscendentalEquations(Double** coef, Double*
initX) : Equations(coef, initX) {}

Boolean TranscendentalEquations::checkInitApprox(int i)
{
    W(0, 0) = 0;
    W(1, 1) = 0;
    switch (i) {
    case 1:
        W(0, 1) = Math::Abs(getX1dY(x0[1]));
        W(1, 0) = Math::Abs(getY2dX(x0[0]));
        break;
    case 2:
        W(0, 1) = Math::Abs(getX2dY(x0[1]));
        W(1, 0) = Math::Abs(getY1dX(x0[0]));
        break;
    }
    if (!isfinite(W(0, 1)) || !isfinite(W(1, 0))) return false;

    if (W.getNormM() < 1 && W.getNormL() < 1) {
        return true;
    }
    return false;
}

```

```

Int16 TranscendentalEquations::JakobiMethod(List<Double>^ result, Double
precis)
{
    if (a[0][0] == 0 && a[0][1] == 0 && a[1][0] == 0 && a[1][1] == 0 &&
a[0][2] == -2 && a[1][2] == -2) {
        return 2;
    }
    if ((a[0][2] >= 0 || a[1][2] >= 0) ||
(a[0][0] == 0 && a[0][1] == 0 && a[0][2] != -2) ||
(a[1][0] == 0 && a[1][1] == 0 && a[1][2] != -2) ||
(a[0][2] == -1 && (a[0][0] == 0 || a[0][1] == 0)) ||
(a[1][2] == -1 && (a[1][0] == 0 || a[1][1] == 0))) {
        return -2;
    }

    Double num1, num2;           // Змінні для тимчасового збереження
результатів обчислення коренів.
    Boolean flag = true;         // Прапорець, який визначає, чи вдалося
розв'язати систему рівнянь.
    Double delta;                // Різниця між значеннями коренів на сусідніх
ітераціях.
    Int32 i;                     // Лічильник ітерацій.

/* 1 */
    if (a[0][0] != 0 && a[1][1] != 0 && checkInitApprox(1)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
        while (true) {
            iterations++;
            num1 = getX1(result[i - 1]);
            num2 = getY2(result[i - 2]);
            if (!isfinite(num1) || !isfinite(num2)) {
                result->Clear();
                flag = false;
                break;
            }
            result->Add(num1);
            result->Add(num2);
            delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
            if (delta <= precis) {
                return 1;
            }
            i += 2;
        }
    }

/* 2 */
    if (a[1][0] != 0 && a[0][1] != 0 && checkInitApprox(2)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
        while (true) {
            iterations++;
            num1 = getX2(result[i - 1]);

```

```

        num2 = getY1(result[i - 2]);
        if (!isfinite(num1) || !isfinite(num2)) {
            result->Clear();
            return -3;
        }
        result->Add(num1);
        result->Add(num2);
        delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
        if (delta <= precis) {
            return 1;
        }
        i += 2;
    }
}
if (flag) return -1;
return -3;
}
Int16 TranscendentalEquations::ZeydelMethod(List<Double>^ result, Double
precis)
{
    if (a[0][0] == 0 && a[0][1] == 0 && a[1][0] == 0 && a[1][1] == 0 &&
a[0][2] == -2 && a[1][2] == -2) {
        return 2;
    }
    if ((a[0][2] >= 0 || a[1][2] >= 0) ||
        (a[0][0] == 0 && a[0][1] == 0 && a[0][2] != -2) ||
        (a[1][0] == 0 && a[1][1] == 0 && a[1][2] != -2) ||
        (a[0][2] == -1 && (a[0][0] == 0 || a[0][1] == 0)) ||
        (a[1][2] == -1 && (a[1][0] == 0 || a[1][1] == 0))) {
        return -2;
    }

    Double num1, num2;           // Змінні для тимчасового збереження
результатів обчислення коренів.
    Boolean flag = true;         // Прапорець, який визначає, чи вдалося
розв'язати систему рівнянь.
    Double delta;                // Різниця між значеннями коренів на сусідніх
ітераціях.
    Int32 i;                     // Лічильник ітерацій.

/* 1 */
    if (a[0][0] != 0 && a[1][1] != 0 && checkInitApprox(1)) {
        result->Add(x0[0]);
        result->Add(x0[1]);
        i = 2;
        while (true) {
            iterations++;
            num1 = getX1(result[i - 1]);
            if (!isfinite(num1)) {
                result->Clear();
                flag = false;
                break;
            }
            result->Add(num1);

```

```

        num2 = getY2(result[i]);
        if (!isfinite(num2)) {
            result->Clear();
            flag = false;
            break;
        }
        result->Add(num2);
        delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
        if (delta <= precis) {
            return 1;
        }
        i += 2;
    }
}

/* 2 */
if (a[1][0] != 0 && a[0][1] != 0 && checkInitApprox(2)) {
    result->Add(x0[0]);
    result->Add(x0[1]);
    i = 2;
    while (true) {
        iterations++;
        num1 = getX2(result[i - 1]);
        if (!isfinite(num1)) {
            result->Clear();
            return -3;
        }
        result->Add(num1);

        num2 = getY1(result[i]);
        if (!isfinite(num2)) {
            result->Clear();
            return -3;
        }
        result->Add(num2);
        delta = Math::Max(Math::Abs(result[i] - result[i - 2]),
Math::Abs(result[i + 1] - result[i - 1]));
        if (delta <= precis) {
            return 1;
        }
        i += 2;
    }
}
if (flag) return -1;
return -3;
}

Int16 TranscendentalEquations::getId()
{
    return 3;
}

String^ TranscendentalEquations::getEquations()
{
    String^ eq = (" e^(" + Convert::ToString(a[0][0]) + " * x) + e^(" +
Convert::ToString(a[0][1]) + " * y) + " +

```

```

        Convert::ToString(a[0][2]) + " = 0\n  e^(" +
Convert::ToString(a[1][0]) + " * x) + e^(" + Convert::ToString(a[1][1]) +
        " * y) + " + Convert::ToString(a[1][2]) + " = 0");
    return eq;
}

Double TranscendentalEquations::getY1(Double x)
{
    Double num = -Math::Pow(Math::E, a[0][0] * x) - a[0][2];
    if (num <= 0) return NAN;
    return Math::Log(num) / a[0][1];
}
Double TranscendentalEquations::getY2(Double x)
{
    Double num = -Math::Pow(Math::E, a[1][0] * x) - a[1][2];
    if (num <= 0) return NAN;
    return Math::Log(num) / a[1][1];
}
Double TranscendentalEquations::getX1(Double y)
{
    Double num = -Math::Pow(Math::E, a[0][1] * y) - a[0][2];
    if (num <= 0) return NAN;
    return Math::Log(num) / a[0][0];
}
Double TranscendentalEquations::getX2(Double y)
{
    Double num = -Math::Pow(Math::E, a[1][1] * y) - a[1][2];
    if (num <= 0) return NAN;
    return Math::Log(num) / a[1][0];
}

Double TranscendentalEquations::getY1dX(Double x)
{
    Double num = a[0][1] * (Math::Pow(Math::E, a[0][0] * x) + a[0][2]);
    if (num == 0) return NAN;
    return a[0][0] * Math::Pow(Math::E, a[0][0] * x) / num;
}
Double TranscendentalEquations::getY2dX(Double x)
{
    Double num = a[1][1] * (Math::Pow(Math::E, a[1][0] * x) + a[1][2]);
    if (num == 0) return NAN;
    return a[1][0] * Math::Pow(Math::E, a[1][0] * x) / num;
}
Double TranscendentalEquations::getX1dY(Double y)
{
    Double num = a[0][0] * (Math::Pow(Math::E, a[0][1] * y) + a[0][2]);
    if (num == 0) return NAN;
    return a[0][1] * Math::Pow(Math::E, a[0][1] * y) / num;
}
Double TranscendentalEquations::getX2dY(Double y)
{
    Double num = a[1][0] * (Math::Pow(Math::E, a[1][1] * y) + a[1][2]);
    if (num == 0) return NAN;
    return a[1][1] * Math::Pow(Math::E, a[1][1] * y) / num;
}

```



## Файл «JakobiMatrix.h»

```

#pragma once
#include <iostream>
using namespace System;
/// <summary>
/// Клас, що відображає матрицю Якобі.
/// </summary>
ref class JakobiMatrix
{
protected:

    /// <summary>
    /// Матриця Якобі.
    /// </summary>
    Double** W;

public:

    /// <summary>
    /// Конструктор матриці Якобі за замовчуванням.
    /// </summary>
    JakobiMatrix();
    /// <summary>
    /// Конструктор матриці Якобі із заданого двовимірного масиву.
    /// </summary>
    /// <param name = 'matrix'>Двовимірний масив, з якого буде створена
матриця Якобі.</param>
    JakobiMatrix(Double** matrix);
    /// <summary>
    /// Очищує динамічну пам'ять виділену на матрицю.
    /// </summary>
    ~JakobiMatrix();

    /// <summary>
    /// Формує матрицю Якобі із заданого двовимірного масиву.
    /// </summary>
    /// <param name = 'matrix'>Двовимірний масив, з якого буде створена
матриця Якобі.</param>
    Void setMatrix(Double** matrix);

    /// <summary>
    /// Повертає перший норм (норм M) матриці Якобі.
    /// </summary>
    /// <returns>Значення типу Double.</returns>
    Double getNormM();
    /// <summary>
    /// Повертає другий норм (норм L) матриці Якобі.
    /// </summary>
    /// <returns>Значення типу Double.</returns>
    Double getNormL();

    /// <summary>
    /// Повертає елемент матриці Якобі з індексами (row, col).
    /// </summary>
    /// <param name = 'row'>Номер рядка.</param>

```

```

    /// <param name = 'col'>Номер стовпця.</param>
    /// <returns>Посилання на змінну типу Double.</returns>
    Double& operator()(const Int16 row, const Int16 col);
};

```

Файл «JakobiMatrix.cpp»

```

#include "pch.h"
#include "JakobiMatrix.h"

JakobiMatrix::JakobiMatrix()
{
    W = new Double * [2];
    for (Int16 i = 0; i < 2; ++i) {
        W[i] = new Double[2];
    }
}

JakobiMatrix::JakobiMatrix(Double** matrix)
{
    W = new Double * [2];
    for (Int16 i = 0; i < 2; ++i) {
        W[i] = new Double[2];
    }
    for (Int16 i = 0; i < 2; ++i) {
        for (Int16 j = 0; j < 2; ++j) {
            W[i][j] = matrix[i][j];
        }
    }
}

JakobiMatrix::~JakobiMatrix()
{
    for (Int16 i = 0; i < 2; ++i) {
        delete[] W[i];
    }
    delete[] W;
}

Void JakobiMatrix::setMatrix(Double** matrix)
{
    for (Int16 i = 0; i < 2; ++i) {
        for (Int16 j = 0; j < 2; ++j) {
            W[i][j] = matrix[i][j];
        }
    }
}

Double JakobiMatrix::getNormM()
{
    return Math::Max(W[0][0] + W[0][1], W[1][0] + W[1][1]);
}

Double JakobiMatrix::getNormL()
{
    return Math::Max(W[0][0] + W[1][0], W[0][1] + W[1][1]);
}

```

```
Double& JakobiMatrix::operator()(const Int16 row, const Int16 col)
{
    return W[row][col];
}
```

Файл «pch.h»

```
// pch.h: This is a precompiled header file.
// Files listed below are compiled only once, improving build performance for
// future builds.
// This also affects IntelliSense performance, including code completion and
// many code browsing features.
// However, files listed here are ALL re-compiled if any one of them is updated
// between builds.
// Do not add files here that you will be updating frequently as this negates
// the performance advantage.

#ifndef PCH_H
#define PCH_H

// add headers that you want to pre-compile here

#endif //PCH_H
```

Файл «pch.cpp»

```
// pch.cpp: source file corresponding to the pre-compiled header

#include "pch.h"

// When you are using pre-compiled headers, this source file is necessary for
// compilation to succeed.
```

Файл «resource.h»

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Calculator.rc
//
#define IDI_ICON1 101

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE 105
#define _APS_NEXT_COMMAND_VALUE 40001
#define _APS_NEXT_CONTROL_VALUE 1001
#define _APS_NEXT_SYMED_VALUE 101
#endif
#endif
```



