МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: Розв’язання систем лінійних рівнянь

Студента 1 курсу, групи ІП-81

Касьяненка М.В.

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник: ас. Головченко М.М

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ- 2019 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

## Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Програмна інженерія"

Курс 1 Група ІП-81 Семестр 2

### **ЗАВДАННЯ**

#### на курсову роботу студента

##### Касьяненка Максима Володимировича

(прізвище, ім’я, по батькові)

1. Тема роботи Розв’язання систем лінійних алгебраїчних рівнянь.

Розробити програму, яка дозволяє розв’язувати системи лінійних алгебраїчних рівнянь

2. Строк здачі студентом закінченої роботи

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

КАЛЕНДАРНИЙ ПЛАН

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

# Студент

(підпис)

Керівник Муха І. П.

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

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

# Анотація

Пояснювальна записка до курсової роботи: 56 сторінок, 12 рисунків, 8 таблиць, 5 посилань.

Об’єкт дослідження: ефективні методи розв’язання систем лінійних рівнянь.

Мета роботи: дослідження методів розв’язання систем лінійних рівнянь, а саме методу Гаусса, методу Жордана-Гаусса, Методу Обертання та створення програмного забезпечення для реалізації цих методів.

Дана курсова робота включає в себе: опис методів, застосування методу до конкретного завдання, код програми вирішення перерахованих вище методів на мові програмування С++, а також описання детального процесу розв’язання кожного з них.

Вступ

Життя будь-якої людини завжди неупорядковане та непередбачуване. Через це, люди завжди намагаються спростити своє існування. Одним із способів досягнення цієї мети є винайдення корисних для неї пристроїв. У наш час велику роль відіграють електронні обчислювальні машини, оскільки вони здатні дуже зручно спростити життя людини.

Електронні обчислювальні машини керують багатьма сферами життя людини. До того-ж люди постійнно пряцюють над їх вдосконаленням. Таким чином іх можливості постійно зростають. Постійно з’являються нові машини, деталі так само як і нові алгоритми. Нові алгоритми створюють з метою використовувати комп’ютерні ресурси більш ефективно. Також завдякиновим алгоритмам все більше видів задач можна виконати за допомогою комп’ютера.

Одним зі способів використання комп’ютера є розв’язок математичних задач, а саме розв’язок систем лінійних алгебраїчних рівнянь. Така можливість є дуже зручною, оскільки налагоджений алгоритм не допускає помилок при розв’язанні, що е властивим людям.

Саме тому у даній курсовій роботі мною було гозглянуто і проаналізовано алгоритми розв’язання систем лінійних рівнянь. Окрім цього було створено програму з графічним інтерфейсом, яка дозволяє розв’язіват системи, знаходити їх корені.

1. Постановка задачі

* Розробити прграму, завдякии якій користувач може отримувати розв’язок системи лінійних алгебраїчних рівнянь, введеної в прграму у текстовому вигляді.
* Реалізувати можливість вибору алгоритму за яким систем буде розв’язяна.
* Реалізувати два текстових поля, одне з яких необхідне для введеня користувачем системи рівнянь, а інше- для виведення на екран резуль тату роботи програми.
* Реалізувати можливість зберегти розв’язок системи в файл.
* Реалізувати можливість користувачу самостійно вводити ім’я файлу у який буде виконано збереження результатів роботи програми.
* Розробити три алгоритми для розв’вязку системи лінійних рівнянь.

1. Аналіз предметної області

2.1 ТЕОРЕТИЧНІ ВІДОМОСТІ

Система лінійних алгебраїчних рівнянь (СЛАР) в [лінійній алгебрі](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D0%BD%D1%96%D0%B9%D0%BD%D0%B0_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0) має вигляд:

|  |  |
| --- | --- |
| C:\Users\123User\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\85BD49F1.tmp | (2.1) |

де , i =1,2,…,m; j =1,2,…,n – відомі коефіцієнти; , i =1,2…,m – праві частини (чи вільні члени) також відомі; – невідомі, які слід визначити. Розв’язком системи (2.1) називається набір чисел, які при підстановці в кожне рівняння перетворюють його на тотожність. Система називається сумісною, якщо вона має хоча б один розв’язок. Система називається несумісною, якщо вона розв’язків не має.

**Методи розв’язування системи лінійних рівнянь.**

**Метод Гауса**

Запишемо систему Ax=f, в розгорнутому вигляді

|  |  |
| --- | --- |
|  | (2.2) |

Метод Гауса полягає в послідовному виключенні невідомих з даної системи [1]. Припустимо, що . Якщо це не так, то переставимо на перше місце рівняння з першим ненульовим доданком. Послідовно множачи перше рівняння на та додаючи його до i-го рівняння, виключимо з усіх рівнянь окрім першого. Отримаємо систему:

|  |  |
| --- | --- |
|  | (2.3) |

Аналогічним чином із отриманої системи виключимо . Послідовно вилючаючи усі невідомі, отримаємо систему трикутного вигляду

|  |  |
| --- | --- |
|  | (2.4) |

Описана процедура називається прямим ходом методу Гауса. Зауважимо, що її виконання можливе за умови, що усі не дорівнюють нулю.

Виконуючи послідовні підстановки у останній системі, починаючи із останнього рівняння, можна отримати усі значення невідомих

|  |  |
| --- | --- |
|  | (2.5) |

Ця процедура отримала назву зворотній хід методу Гауса.

Алгоритм методу Гауса можна легко реалізувати на комп’ютері. При виконанні обчислень, як правило, не цікавлять проміжні значення матриці А. Тому чисельна реалізація методу зводиться до перетворення елементів матриці (m×(m+1)), де m+1 стовпчик, який містить праву частину системи.

**Метод обертання розв’язку лінійних систем.**

Як і у методі Гауса, мета прямого ходу перетворень у цьому методі – приведення системи до трикутного виду послідовно отримуючи нульові значення елементів під діагоналлю спочатку першого стовпчика, потім другого і т.д.

|  |  |
| --- | --- |
|  | (2.6) |

Помножимо перше рівняння вихідної системи (2.6) на *с1* ,друге *s1* та додаємо їх, отриманим рівнянням замінимо перше рівняння системи [2]. Потім перше рівняння системи помножимо на –*s1* , друге на *с1* і результатом їх додавання замінимо друге рівняння системи. Таким чином перші два рівняння системи (2.6) замінюються рівняннями

|  |  |
| --- | --- |
|  | (2.7) |

На величини *с1* і *s1* накладемо дві умови:

|  |  |
| --- | --- |
|  | (2.8) |
|  | (2.9) |

Звідси

|  |  |
| --- | --- |
| . | (2.10) |

Ці числа можна інтерпретувати як косинус та синус деякого кута .

У результаті перетворень отримаємо систему

|  |  |
| --- | --- |
|  | (2.11) |

де

|  |  |
| --- | --- |
|  | (2.12) |

Далі перше рівняння системи замінюється новим, отриманим додаванням результатів множення першого і третього рівнянь відповідно на

|  |  |
| --- | --- |
|  | (2.13) |

а третє – рівнянням, яке отримано додаванням результатів множення тих самих рівнянь відповідно на –*s1* і *с1*. Отримаємо систему

|  |  |
| --- | --- |
|  | (2.14) |

де

|  |  |
| --- | --- |
|  | (2.15) |

Виконавши перетворення *m-1* раз, отримаємо систему

|  |  |
| --- | --- |
|  | (2.16) |

Отримана система має вигляд такий самий, як після першого етапу перетворення методу Гауса. Ця система має наступну властивість: довжина будь-якого вектору-стовпчика розширеної матриці залишається такою самою, як у вихідної матриці. Отже, при виконанні перетворень не спостерігається збільшення елементів.

Далі по такому самому алгоритму, виконуються перетворення матриці

|  |  |
| --- | --- |
|  | (2.17) |

і т.д.

У результаті *m*-1 етапів прямого ходу система буде приведена до трикутного виду.

|  |  |
| --- | --- |
|  | (2.18) |

Невідомі знаходяться за допомогою зворотнього ходу методу Гауса.

2.2 ДІАГРАМА ПРЕЦЕДЕНТІВ

На рисунку 2.1схему взаємодії користувача з програмою.

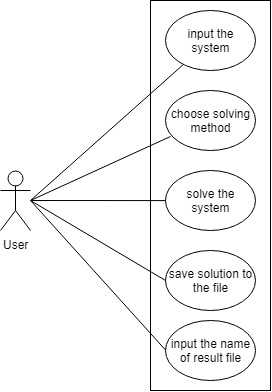


Рис 2.1 - Діаграма прецедентів.

* Актор User – це користувач програми. Він є єдиним актором у даній ситуації.
* Input the system – користувач має можливість вводити у врограму систему лінійних алгебраїчних рівнянь.
* Choose solving method – корстувач має можливість обирати метоз яким буде розв’язна введена ним система лінійних рівнянь.
* Solve the System – користувач може отримати розв’язок системи рівнянь отриманий у результаті роботи обраного ним алгоритму розв’язку системи.
* Save solution to the file – має можливість зберігати розв’язок системи у текстовий файл.
* Input the name of the file –користувач може вводити назву файлу, у який буде збережено систему.

2.3 ОПИС АЛГОРИТМІВ

Перелік всіх основних змінних та їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| The\_system | Система лінійних рівнянь |
| Matrix | Матриця системи |
| How\_many | Вказує на кількість змінних у системі |
| Variables | Список змінних присутніх у системі. |
| Plus | Вказує на знак що стоїть перед змінною |
| Cnumber | Число яке ми занесемо у матрицю |
| Row | Рядок матриці з яким ми працюємо в даний момент |
| Coma | Вказує на наявність десяткового дробу |

## Загальний алгоритм

1. ПОЧАТОК
2. Зчитати систему у текстовому вигляді.
3. Зберегти систему у вигляді матриці системи, та списку змінних:
   1. Зчитати матрицю системи:
      1. Цикл проходу по всіх символах введеного рядка :
         * 1. ЯКЩО данний елемент – літера, яка не зустрічалась у рядку до цього – додати її до списку змінних.
   2. Цикл проходу по всіх символах введеного рядка:
      1. ЯКЩО даний елемент рядка – операція зберегти знак коефіціенту у змінну
      2. ЯКЩО даний елемент кома зберегти це у змінній, з можливістю подальшої обробки десяткового дробу.
      3. ЯКЩО даний елемент – число, додати його до сnumber з урахуванням коми і знаку.
      4. ЯКЩО даний елемент – літера, зберегти cnumber у клітинці що відповідає даній змінній. Та обнулити змінні використані до цього.
4. ЯКЩО обраний метод Гауса, ТО обробити дані згідно алгоритму методу Гауса(пункт 3.2)
5. ЯКЩО обраний метод Жордана-Гауса, ТО обробити дані згідно алгоритму методу Жордана-Гауса (пункт 3.3)
6. ЯКЩО обраний метод Обертання, ТО обробити дані згідно алгоритму методу Обертання (пункт 3.4).
7. ЯКЩО обрано графічний метод:

7.1 ЯКЩО система має 2 змінні, ТО обробити дані згідно алгоритму Графічного методу

1. Вивести результат роботи на екран.
2. ЯКЩО вибрано збереження результату в файл, зберегти результат у файл.
3. КІНЕЦЬ

## Алгоритм методу Гауса

1. ПОЧАТОК
2. Цикл по всіх рядах матриці (*Matrix*):
   1. ЦИКЛ по всіх рядках матриці починаючи з поточного
      1. ЯКЩО початковий елементрядка не рівний нулю, поставити його на поточне місце розділити даний рядок матриці на початковий елемент, закінчити цикл. (*Matrix*)
   2. Цикл по всіх рядках матриці після відповідного:
      1. Відняти від поточного рядка даний помножений на початковий елемент поточного рядка.
3. Цикл по всіх рядках матриці починаючи з останнього.
   1. ЯКЩО рядок останній, розділити рядок на елемент arr[y][y].
      1. Занести значення даної змінної до множини розв’язків.
   2. ЯКЩО радок не останній, ТО відняти від останнього елементу рядку занчення усіх змінних помножені на коефіціени матриці.
      1. Занести значення змінної до множини розв’язку.
4. Записати результат у вигляді рядкової змінної
5. КІНЕЦЬ

**с. Метод Обертання**

1. ПОЧАТОК
2. Цикл повсіх рядках матриці:
   1. Цикл по всіх рядках матриці:
      1. Знайти значення змінних S, C, для даної пари рядків.
      2. Помножити перший рядок на S додати до другого помноженого на С, записати у перший рядок.
      3. Помножитидругий рядок на S додати відняти від першого помноженого на С, записати у другий рядок.
3. Цикл по всіх рядках матриці починаючи з останнього.
   1. ЯКЩО рядок останній, розділити рядок на елемент arr[y][y].
      1. Занести значення даної змінної до множини розв’язків.
   2. ЯКЩО радок не останній, ТО відняти від останнього елементу рядку занчення усіх змінних помножені на коефіціени матриці.
      1. Занести значення змінної до множини розв’язку.
4. Записати результат у вигляді рядкової змінної
5. КІНЕЦЬ
6. Опис архітектури програмної системи

На рисунку 3.1 зображена взаємодія між класами.

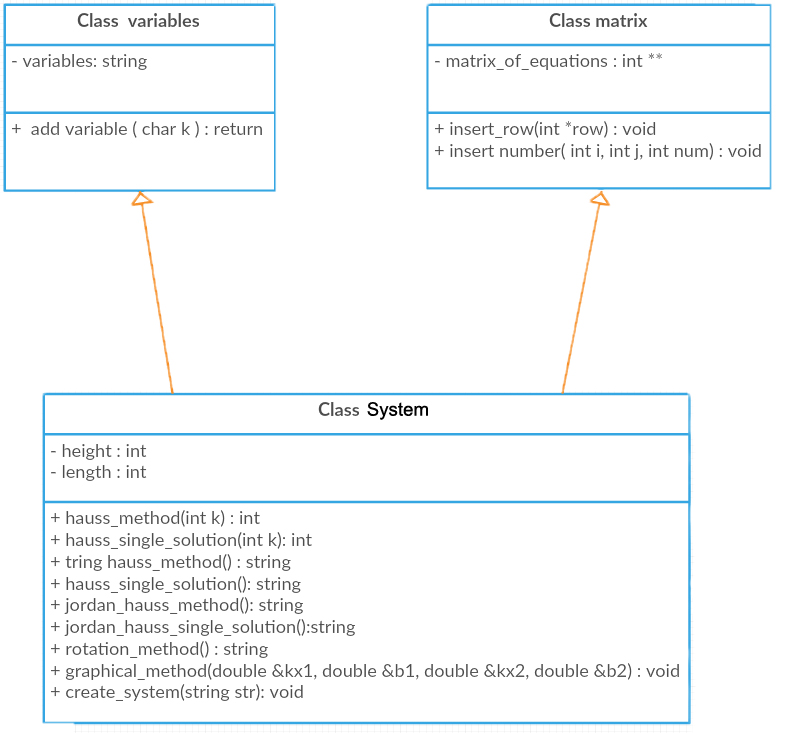


Рис 3.1 – Принцип взаємодії мі жкласами

За даною UML діаграмою видно, що клас System, що є інтерпретаціею системи лінйних алгебраїчних рівнянь, породжений від двох класів за допомогою множинного наслідування. При цьому батьківськими класами для нього є клас Variables I клас Matrix.

Variables – це клас який зберігає множину змінних що містяться у системі. Назви змінних – літери латинського алфавіту. Зберігаються вони у вигляді текстового рядка.

Matrix – це клас що зберігає матрицю системи та стовпець вільних членів. При цьому обидва вищезазначені елементи зберігаються у вигляді одного двовимірного масиву. Дана реалізація є доцільною з огляду на те, що алгоритми, які використовує дана програма виконують аналогічні дії як з матрицею системи так і з стовпцем вільних членів.

System – це власне клас, об’єкти якого зберігають у комп’ютерній пам’яті системи лінійних рівнянь. Він успадковує усі поля і методи батьківських класів, та має певну кількість своїх. Власні методи даного класу – це метод що будує систему алгебраїчних рівнянь з введеного користувачем рядка, та власне методи які перетворюють систему таким чином, щоб знайти розв’язок системи. Методи: hauss\_method, Jordan\_hauss\_method, rotation\_method – це методи, що розв’язують систему рівнянь. Вони повертають рядок, який на екран виводиться в текстове поле. Також його можливо зберегти в файл.

1. Опис програмного забезпечення

Опис специфікації функцій здійснюється у вигляді наведеному у таблиці 4.1

Таблиця 4.1 – Специфікація функцій

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | Variables | Add\_variable | Додає змінну до списку змінних | К – літера, що позначае змінну у системі рівнянь | - | System.h |
| 3 | Marix | Insert\_number | Вставляє в певну позицію чистеми певний індекс | Num – число яке необхідно встатити в систему. I номер рядка у який необхідно вставити систему. J – стовпчик в який необхідно вставити число | - | System.h |
| 2 | Matrix | Insert\_row | Додає до матриці рядок, що передається у функцію як покажчик | Row – покажчик на одновимірний цілочисельний масив, який деобхідно додати | - | System.h |
| 4 | - | Is\_letter | Перевіряє чи є символ літерою | Letter – символ який необхідно перевірти | Булеве значення правда чи ні | before\_solving.h |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 5 | - | How\_many\_wariables | Підраховує кількість змінних у рядку введеному користувачем | Raw\_system – текст введений користувачем, що має бути перетворений на систему рівнянь | Кількіть змінних у системі | before\_solving.h |
| 6 | - | How\_many\_equations | Підраховує кількість рядків у введеній системі, таким чином підраховує кількість рівнянь у системі | Raw\_system – текст введений користувачем, що має бути перетворений на систему рівнянь | Кількість рівнянь в системі | before\_solving.h |
| 7 | - | Is\_number | Перевіряє чи є даний символ числом | Letter – символ який еобхідно перевірити | Булеве значення чиє символ числом | before\_solving.h |
| 8 | Systema | Zero\_row | Перевіряє систему на наявність рядка що містить лише нулі(неврахоуючи стовпець вільних членів) | - | Булеве значення true, якщо такий рядок пристуній | system.h |
| 9 | Systema | Upper\_triangle\_matrix | Функція – перша частина методу Гауса(прямих хід методу Гауса) | - | - | system.h |
| 10 | Systema | Hauss\_method | Перевіряє чи можливе використання методу Гауса, у разі можливості, викликає функцію методу Гауса | - | Рядок з розв’язком системи | system.h |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 11 | Systema | Hauss\_method\_single\_solution | Функція, що розв’язує систему методом Гауса | - | Рядок з розв’язком системи | system.h |
| 12 | Systema | Jordan\_hauss\_method | Перевіряє чи можливе використання методу Жордана-Гауса, у разі можливості, викликає функцію методу Жордана-Гауса | - | Рядок з розв’язком системи | system.h |
| 13 | Systema | Jordan\_hauss\_single\_solution | Функція, що розв’язує систему методом Жордана-Гауса | - | Рядок з розв’язком системи | system.h |
| 14 | Systema | Rotation\_method | Розв’язує систему методом обертання | - | Рядок з розв’язком системи | system.h |
| 15 | Systema | Graphical\_method | Отримує коефіціенти рівняннь прямих для побудови їх у випадку вибору графічного методу | Kx1 – коефіцієнт при Х у першому рівнянні прямої b1- вільний член в першому рівнянні прямої kx2 – коефіцієнт при Х у другому рівнянні прямої b2 - вільний член в другому рівнянні прямої | - | system.h |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 16 | Systema | Create\_system | Будує систему лінійних рівнянь на основы тексту який введений користувачем з клавіатури. | Рядок введений користувачем з клавіатури у відповідне поле, на основі якого має бути побудовано системулінійних рівнянь | - | system.h |
| 17 | Systema | Matrix | Записує матрицю системи у рядкову змінну, для можливості подвльшого виведення її на екран | - | Рядкова змінна, що містить матрицю системи лінійних алгебраїчних рівнянь рівнянь | SelectionScreen.py |

1. Результати тестування програмного забезпечення

Тестування програмного забезпечення — це процес дослідження програми та її коду, з ціллю визначити якість продукту. Тестування включає в себе багато методів і процесів, від пошуку помилок логіки в коді програми до випробування програми непередбачуваними вхідними даними. Оцінюється:

* відповідність вимогам технічного завдання
* коректність результатів виконання обчислень
* час, витрачений на обчислення
* практичність
* сумісність з програмним забезпеченням та операційними системами

Тобто, тестування надає об'єктивну інформацію про якість програмного забезпечення і ризики його відмови.

Як правило, тестування проводиться протягом усього життєвого циклу програмного забезпечення – від початку його розробки до кінця підтримки.

Етапи тестування:

* Введення системи рівнянь не змінюючи порядок змінних у кожному рядку

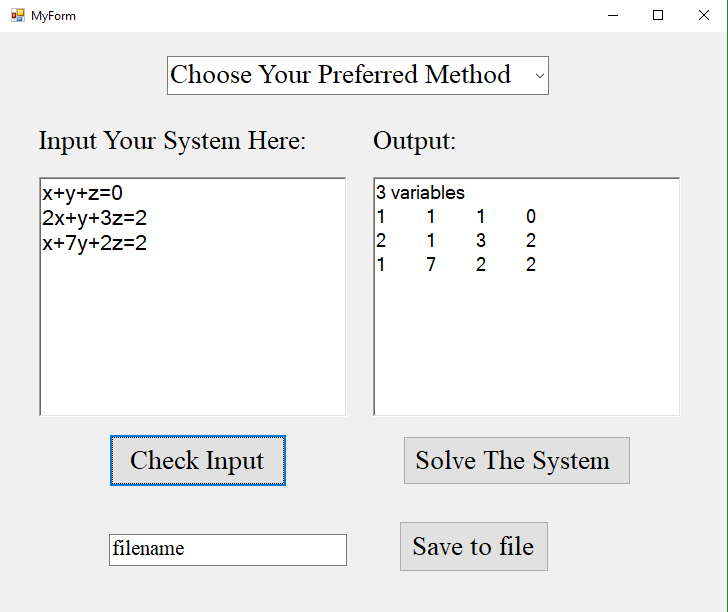


Рисунок 5.1 – Введення системи лінійних рівнянь, перевірка коректності зчитування

* Введення системи рівнянь змінюючи порядок слідування змінних

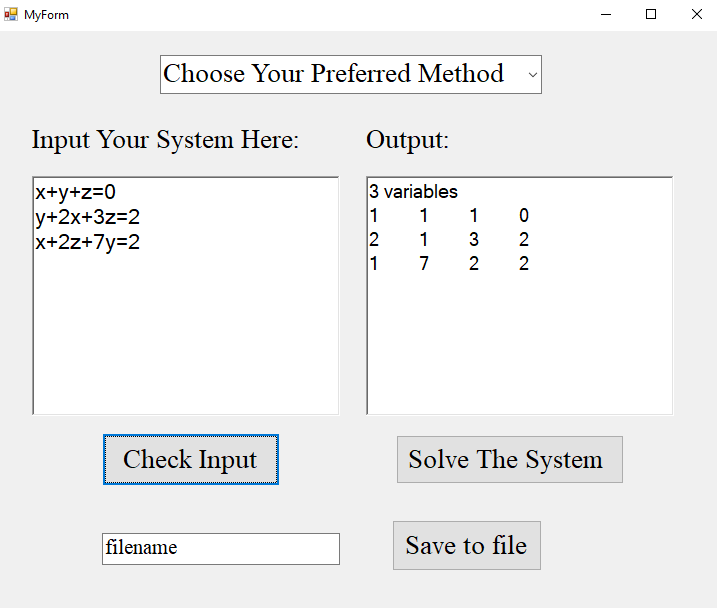


Рисунок 5.2 – Введення змінних у довільному поядку

* Перевірка опрацювання помилки введення – замало рівнянь.

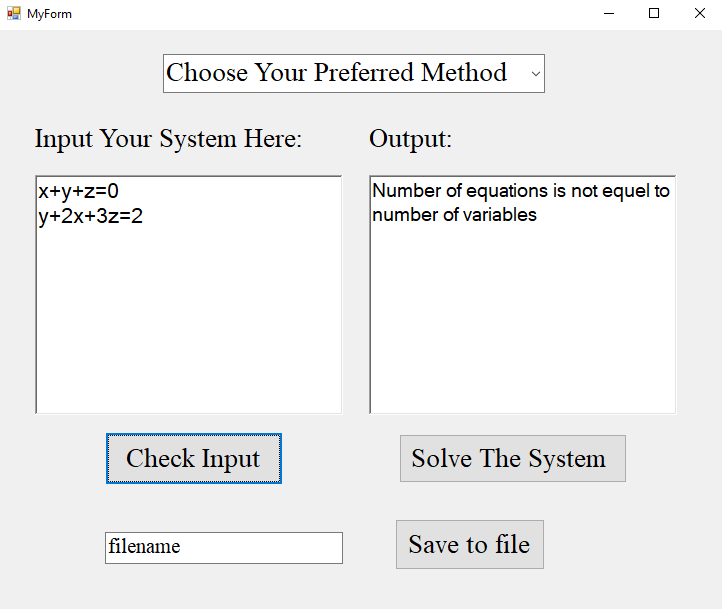


Рисунок 5.3 – повідомлення про помилку, введено недостатню кількість рівнянь

* Перевірка опрацювання помилки введення –забагато рівнянь.

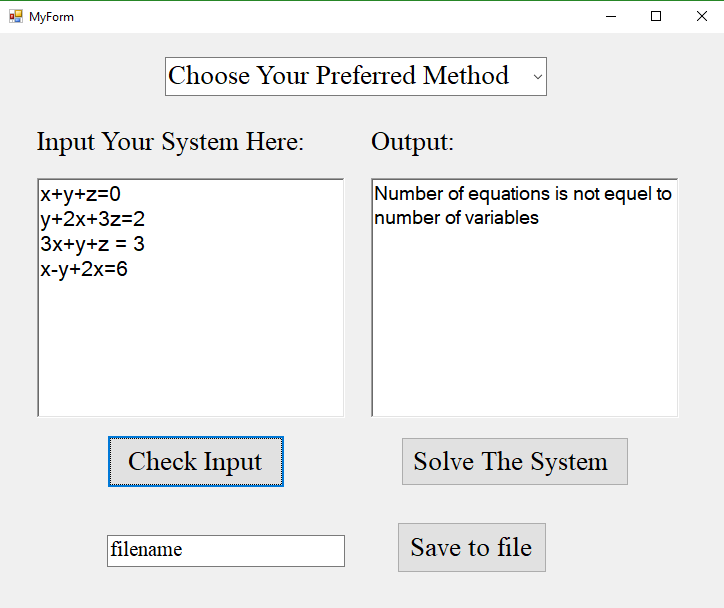


Рисунок 5.4 – повідомлення про помилку, введено завелику кількість рівнянь

* Перевірка роботи при лінійно залежній системі

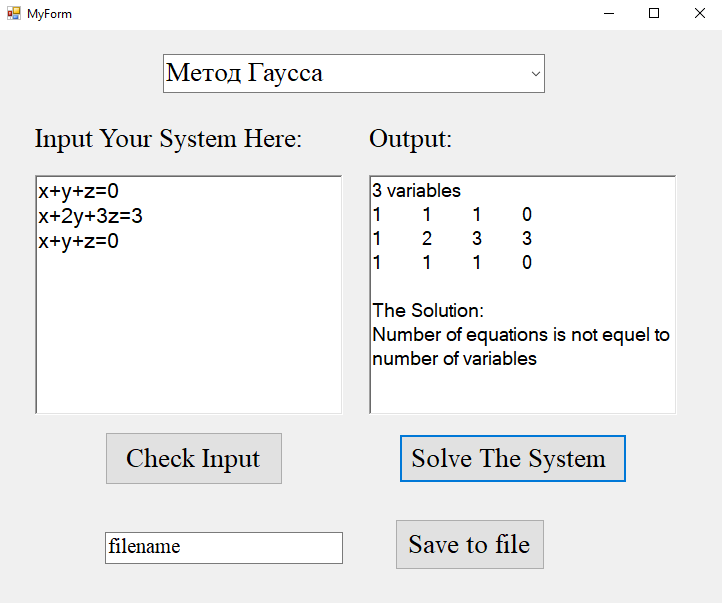


Рисунок 5.5 – повідомлення про лінійну залежність системи

* Обрання методу Гауса

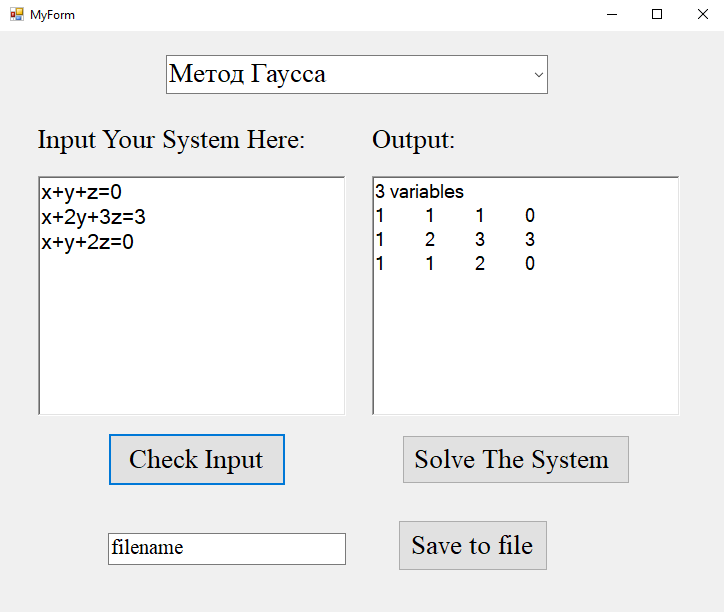


Рисунок 5.6 – обрання методу Гауса

* Обрання методу Жордана-Гауса

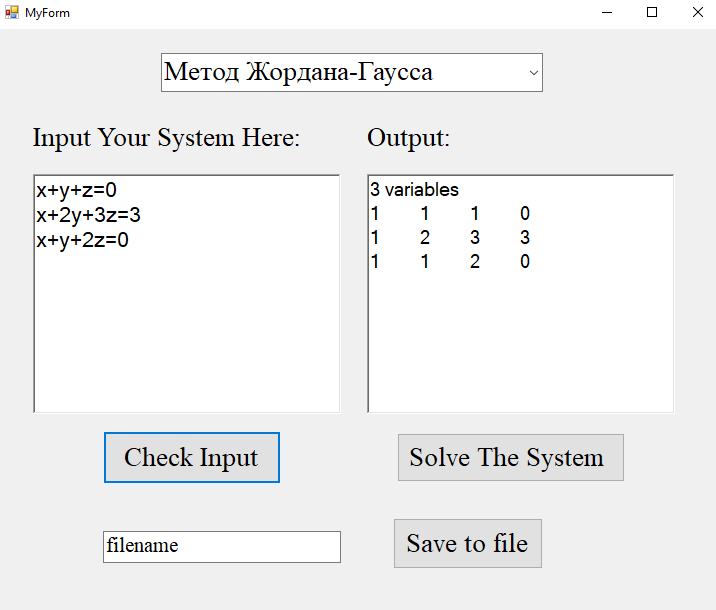


Рисунок 5.7 – Обрання методу Жордана-Гауса

* Обрання Методу Обертання

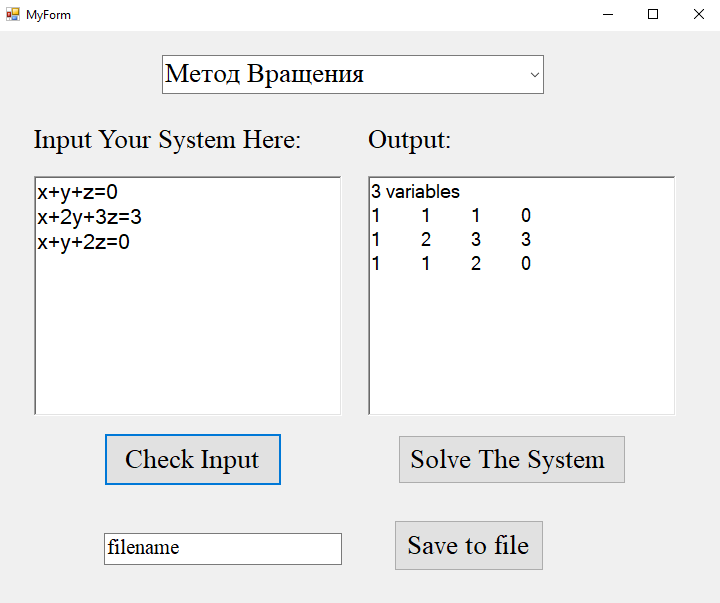


Рисунок 5.8 – обрання методу обертання

* Перевірка обрання графічного методу

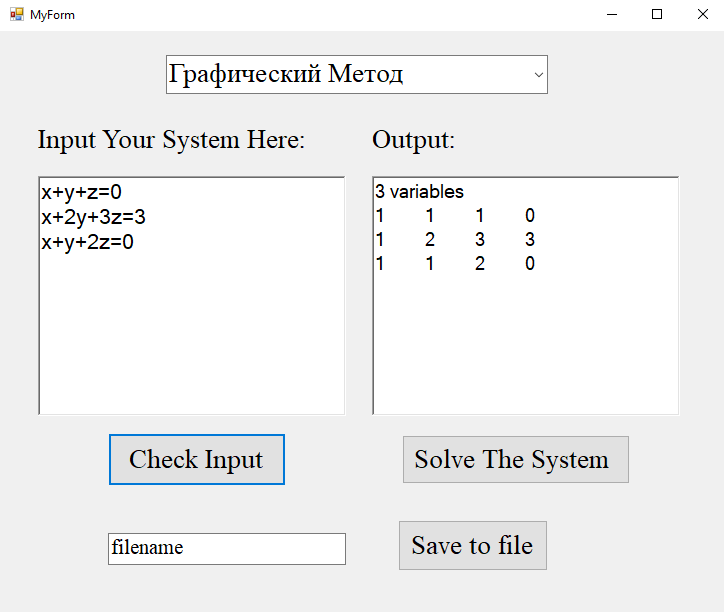


Рисунок 5.9 – обрання графічного методу

* Перевірка коректності роботи методу Гауса

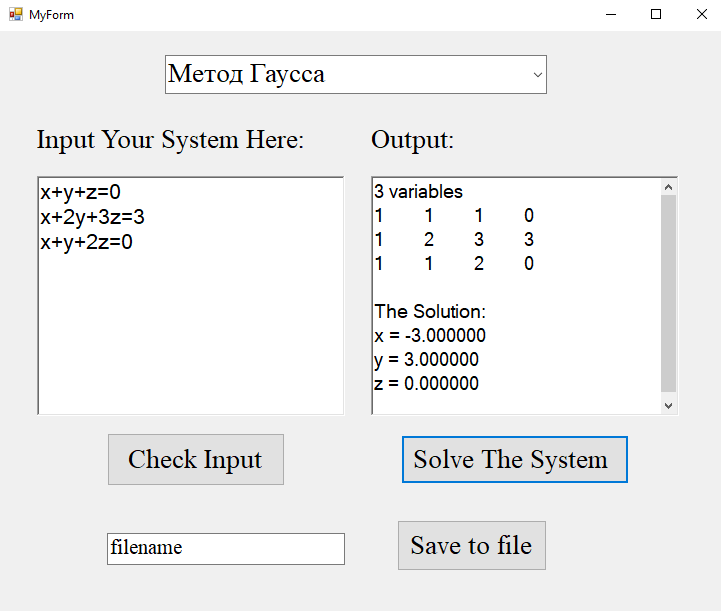


Рисунок 5.10 – Результат роботи Методу Гауса

* Перевірка коректності роботи методу Жордана-Гауса

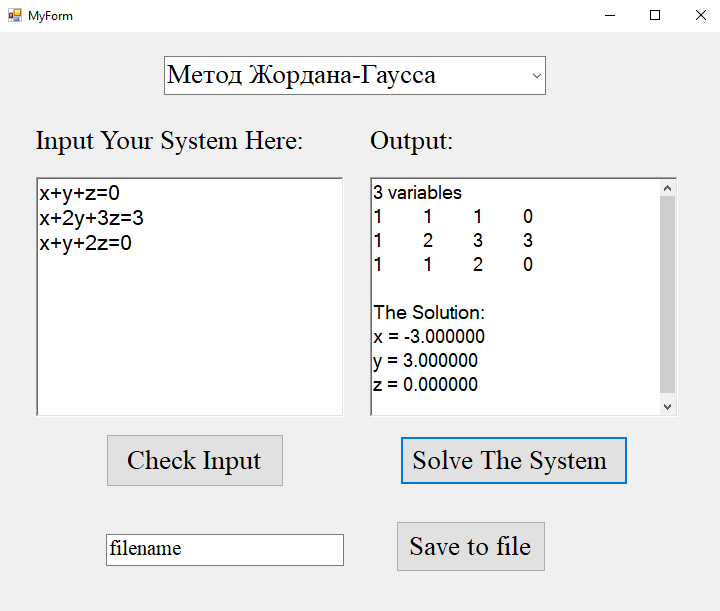


Рисунок 5.11 – результат роботи методу Жордана-Гауса

* Перевірка коректності роботи методу обертання

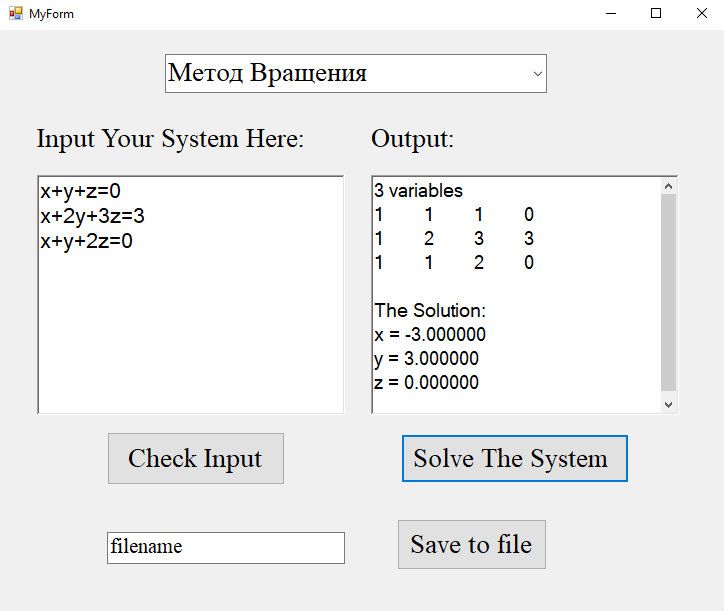


Рисунок 5.12 – результат роботи методу обертання

* Перевірка коректності роботи графічного методу

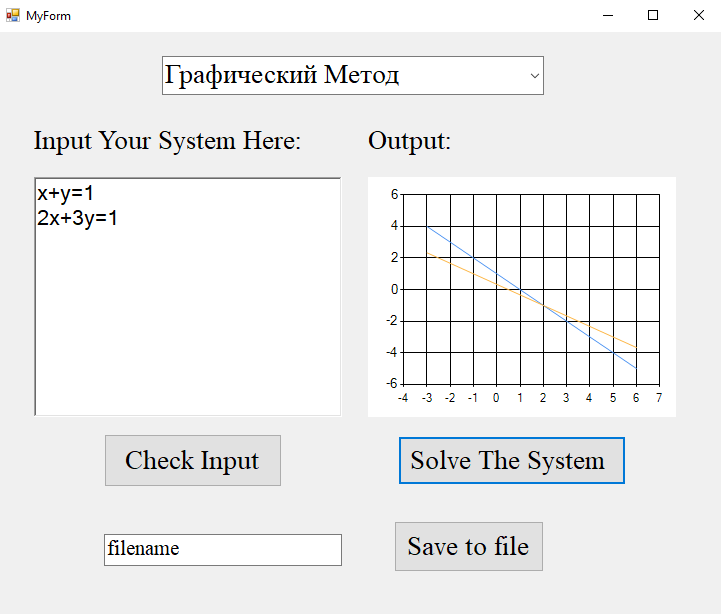


Рисунок 5.13 – результат роботи методу обертання

* Перевірка випадку, коли метод розв’язку не вибраний

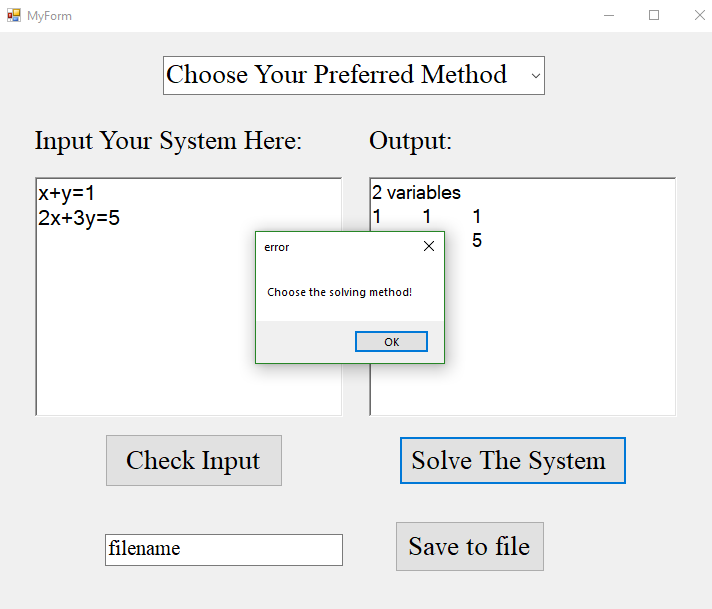


Рисунок 5.14 – повідомлення про необхідність вибрати метод

* Перевірка збереження результату роботи у файл

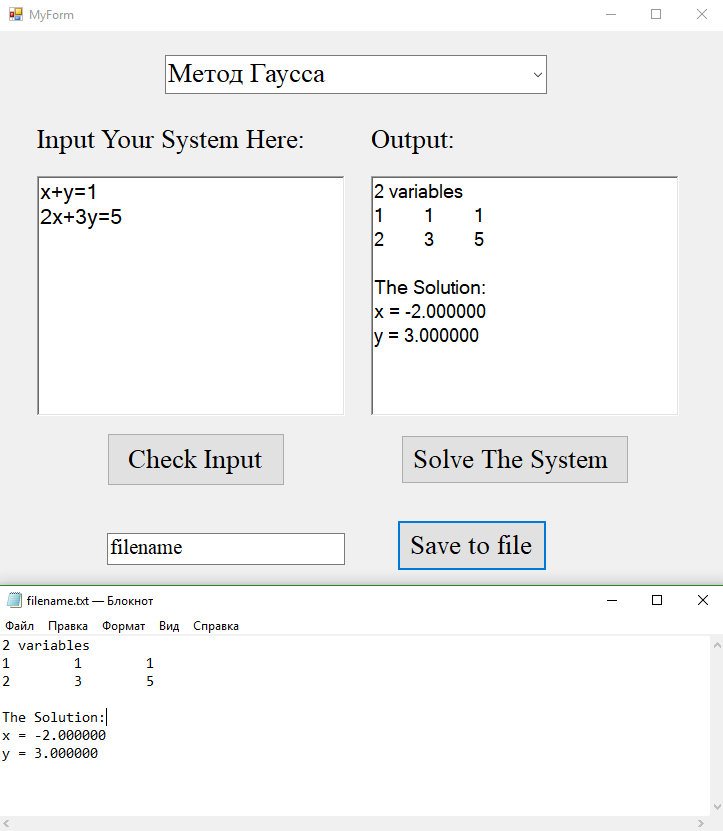


Рисунок 5.15 – результат роботи програми збережений у файл

1. Інструкція користувача
   1. Кроки виконання для успішної роботи програми:
2. Введіть систему лінійних рівнянь

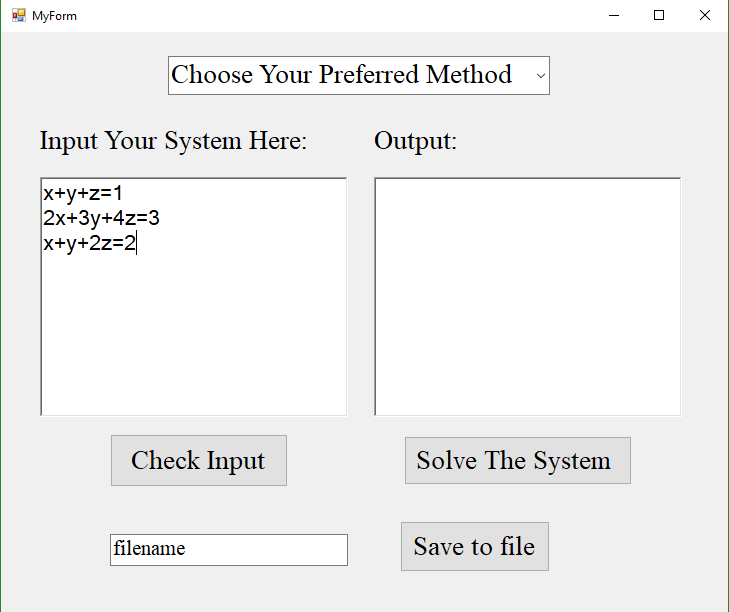


Рисунок 6.1 – введення системи

1. Оберіть метод яким хочете розв’язати систему

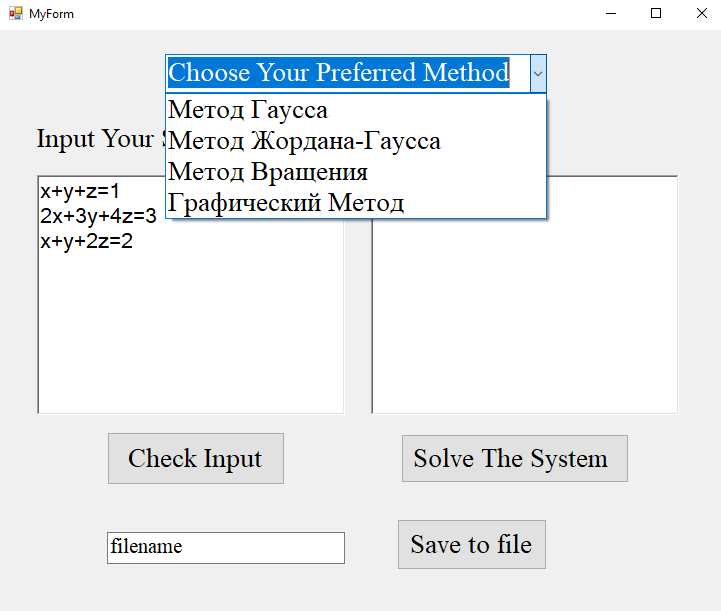


Рисунок 6.2 – обрання методу розв’язку системи

1. Натисніть Check Input, щоб побудувати систему у пам’яті комп’ютера та перевірити, чи правильно вона зчитана.

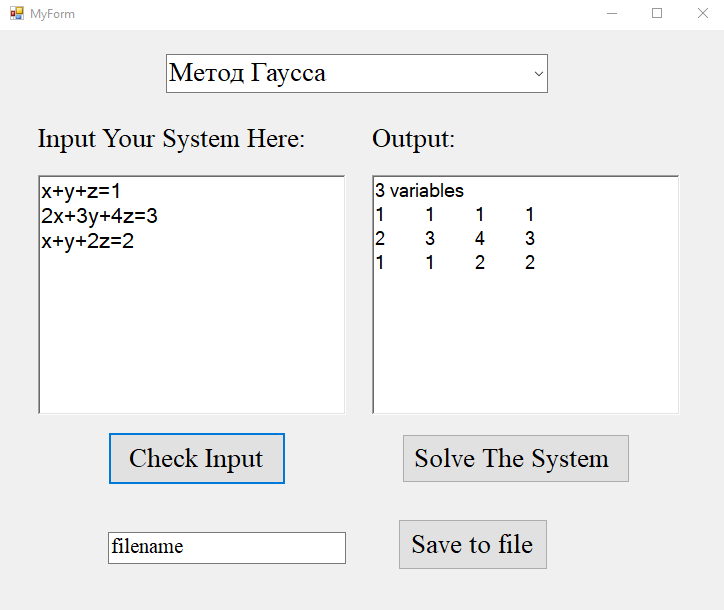


Рисунок 6.3 – вибрання клітинки для ходу

1. Натисніть Solve The System щоб розв’язати систему

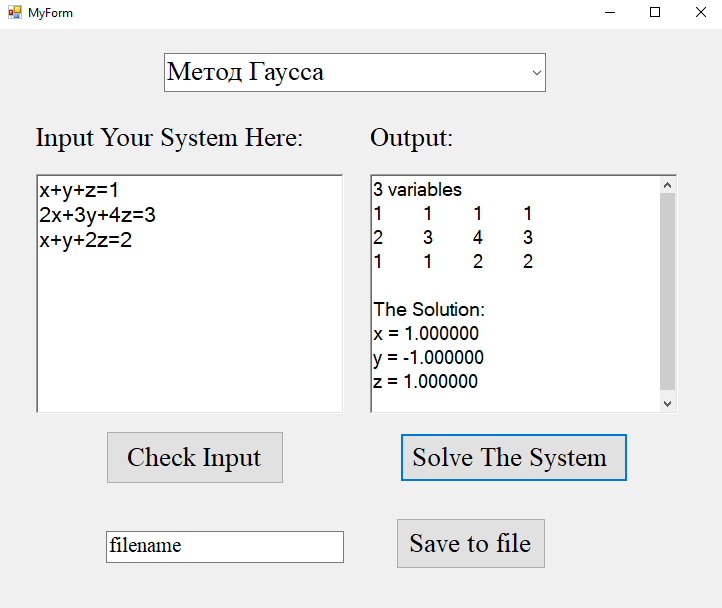


Рисунок 6.4 –отримання розв’язку системи

1. Натисніть Save to file щоб зберегти результат роботи програми у файл.

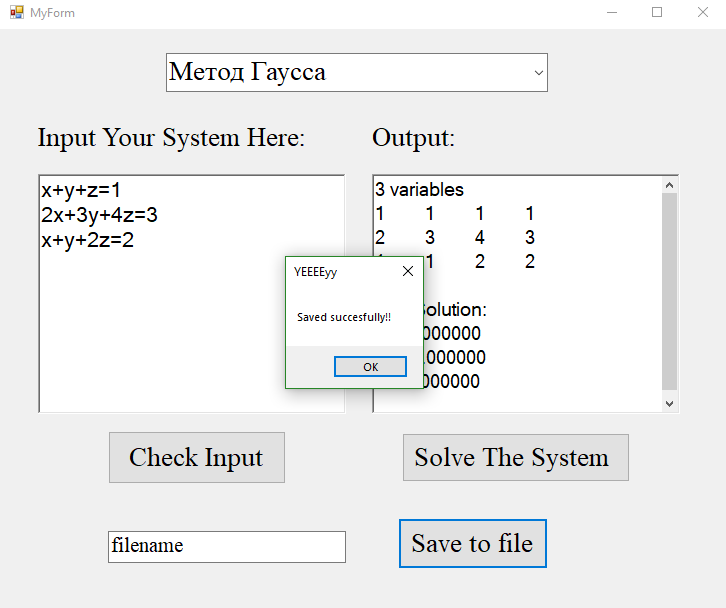


Рисунок 6.4 – повідомлення про успішне збереження у файл

1. Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1

Таблиця 6.1 – Системні вимоги програмного забезпечення

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows®XP/  Windows Vista/  Windows 7/  Windows 8/  Windows 10  MacOS 10.14.3 Mojave | Windows 7/Windows 8/  Windows 10  MacOS 10.14.3 Mojave |
| Процесор | Intel® Pentium® ІІІ 1.0 GHz або AMD Athlon™ 1.0 GHz | Intel® Core™ i3-3110M CPU @ 2.40 GHz або AMD Athlon™ 2.40 GHz |
| Оперативна пам'ять | 256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows Vista/Windows 7/  Windows 8/Windows 10) | 2 GB RAM |
| Відеоадаптер | Intel® HD Graphics 4000 або Radeon™ HD 8670M (або любий сумісний аналог) | |
| Дисплей | 1080х720 | 1600х900 або краще |
| Прилади введення | Клавіатура, комп’ютерна миша | |
| Додаткове програмне забезпечення | Python 3.4 та Pygame 1.9.2a0 або вище | |

Висновок

Під час розробки програми для розв’язку систем лінійних рівнянь були досліджені нові методи створення графічного інтерфейсу для зручнго подальшого використання програми. Були освоєні алгоритми призначені для розв’язку систем за допомогою комп’ютера. Були використані три основні принципи ООП: інкапсуляція, поліморфізм, та успадкування. Були реалізовані власні класи з перевизначеними методами для більш гнучкої обробки даних.

Також були розроблені та реалізовані 4 алгоритми пошуку розв’язків системи лінійних рівнянь. Для систем, що мають 2 змінні був розроблений графічний метод, який малює графіки для двох рівнянь системи і знаходить точку перетину. Та для систем будь-якої розмірності були розроблені 3 алгоритми розв’язку систем, які однаково правильно розв’язують введену користувачем систему рівнянь. Програма запобігає виникненню помилок пов’язаних з некоректним введенням користувачем початкових даних. Також передбачені повідомлення на випадок коли дана система не може бути розв’язана даним методом. Включена функція збереження результату роботи програм у файл, при чому ім’я файлу може бути задане користувачем, і не має обмежень у довжині.

Перелік посилань

1. Albert Sweigart, Making Games with Python & Pygame, — 2012 р.
2. Albert Sweigart, Invent Your Own Computer Games with Python 3rd Edition — 2011 р.
3. Томас Кормен, Алгори́тмы: построе́ние и ана́лиз, — 2010 p.
4. Стюарт Рассел, Питер Норвиг, Искусственный интеллект: современный подход (AIMA-2), 2-е издание, — 2011 p.
5. Эндрю Роллингз, Дэйв Моррис, Проектирование и архитектура компьютерных игр, — 2016 p.
6. Алекс Дж. Шампандар, Искусственный интеллект в компьютерных играх — 2015 p.

Додаток А Технічне завдання

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко М.М

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_ р.

Виконавець:

Студент Касьяненко М.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_ р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: Розв’язяння систем лінійних алгебраїчних рівнянь

точними методами

з дисципліни:

«Об’єктно-орієнтоване програмування»

Київ 2019

* 1. *Мета*: Метою курсової роботи є розробка програми, здатної знаходити розв’язки систем лінійних рівнянь за допомогою методу Гаусса, Жордана-Гаусса та методу обертання, та виводити на екран розв’язок даної системи у текстовому форматі.
  2. *Дата початку роботи*: «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_201\_ р.
  3. *Дата закінчення роботи*: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 201\_ р.
  4. *Вимоги до програмного забезпечення*.

1. Функціональні вимоги:

* Можливість розв’язувати систему лінійних рівнянь методом Гаусса.
* Можливість розв’язувати систему лінійних рівнянь методом Жордана-Гаусса.
* Можливість розв’язувати систему лінійних рівнянь методом Обертання.
* Виконання перевірки, чи можливе застосування даного методу до даної системи
* Виведення Результатів роботи на екран.
* Робота з графічним інтерфейсом.

1. Нефункціональні вимоги:

* На комп’ютері повинна бути встановлена операційна система родини Microsoft Windows.
* 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 - 95 - Розробка технічної документації.

* 1. *Стадії та етапи розробки*:

1. Об'єктно-орієнтований аналіз предметної області задачі (до\_\_.\_\_.201\_ р.)
2. Об'єктно-орієнтоване проектування архітектури програмної системи (до \_\_.\_\_.201\_р.)
3. Розробка програмного забезпечення (до \_\_.\_\_.201\_р.)
4. Тестування розробленої програми (до \_\_.\_\_.201\_р.)
5. Розробка пояснювальної записки (до \_\_.\_\_.201\_ р.).
6. Захист курсової роботи (до \_\_.\_\_.201\_ р.).
   1. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

Додаток Б Тексти програмного коду

*студента групи ІП-81 І курсу*

*Касьяненка М.В.*

(Вид носія даних)

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

*22 арк, 36 Кб*

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення розв’язку систем лінійних рівнянь*

MyForm.h

#pragma once

#include <string>

#include <fstream>

#include <msclr\marshal\_cppstd.h>;

#include "before\_solving.h"

#include "system.h"

//#include "stdafx.h"

systema the\_system;

namespace Project {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

//friend void parce\_system();

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::ComboBox^ comboBox1;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::RichTextBox^ richTextBox2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::DataVisualization::Charting::Chart^ chart1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Button^ button3;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

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::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->button2 = (gcnew System::Windows::Forms::Button());

this->richTextBox2 = (gcnew System::Windows::Forms::RichTextBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->chart1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->button3 = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->BeginInit();

this->SuspendLayout();

//

// richTextBox1

//

this->richTextBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->richTextBox1->Location = System::Drawing::Point(39, 145);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(308, 240);

this->richTextBox1->TabIndex = 0;

this->richTextBox1->Text = L"";

//

// label1

//

this->label1->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(33, 93);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(281, 31);

this->label1->TabIndex = 1;

this->label1->Text = L"Input Your System Here:";

//

// comboBox1

//

this->comboBox1->FlatStyle = System::Windows::Forms::FlatStyle::System;

this->comboBox1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(4) {

L"Метод Гаусса", L"Метод Жордана-Гаусса", L"Метод Вращения",

L"Графический Метод"

});

this->comboBox1->Location = System::Drawing::Point(167, 24);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(382, 39);

this->comboBox1->TabIndex = 2;

this->comboBox1->Text = L"Choose Your Preferred Method";

this->comboBox1->SelectedIndexChanged += gcnew System::EventHandler(this, &MyForm::comboBox1\_SelectedIndexChanged);

//

// button1

//

this->button1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->Location = System::Drawing::Point(109, 402);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(178, 53);

this->button1->TabIndex = 4;

this->button1->Text = L"Check Input";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// button2

//

this->button2->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button2->Location = System::Drawing::Point(403, 404);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(228, 49);

this->button2->TabIndex = 5;

this->button2->Text = L"Solve The System ";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// richTextBox2

//

this->richTextBox2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->richTextBox2->Location = System::Drawing::Point(373, 145);

this->richTextBox2->Name = L"richTextBox2";

this->richTextBox2->Size = System::Drawing::Size(308, 240);

this->richTextBox2->TabIndex = 6;

this->richTextBox2->Text = L"";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(367, 93);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(97, 31);

this->label3->TabIndex = 7;

this->label3->Text = L"Output:";

//

// chart1

//

chartArea1->Name = L"ChartArea1";

this->chart1->ChartAreas->Add(chartArea1);

this->chart1->Location = System::Drawing::Point(373, 145);

this->chart1->Name = L"chart1";

series1->ChartArea = L"ChartArea1";

series1->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;

series1->Name = L"line1";

series2->ChartArea = L"ChartArea1";

series2->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;

series2->Name = L"line2";

this->chart1->Series->Add(series1);

this->chart1->Series->Add(series2);

this->chart1->Size = System::Drawing::Size(308, 240);

this->chart1->TabIndex = 8;

this->chart1->Text = L"chart1";

this->chart1->Visible = false;

this->chart1->Click += gcnew System::EventHandler(this, &MyForm::chart1\_Click);

//

// textBox1

//

this->textBox1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 15.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->Location = System::Drawing::Point(109, 502);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(238, 32);

this->textBox1->TabIndex = 9;

this->textBox1->Text = L"filename";

//

// button3

//

this->button3->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button3->Location = System::Drawing::Point(399, 489);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(150, 51);

this->button3->TabIndex = 10;

this->button3->Text = L"Save to file";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->ClientSize = System::Drawing::Size(727, 583);

this->Controls->Add(this->button3);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->chart1);

this->Controls->Add(this->label3);

this->Controls->Add(this->richTextBox2);

this->Controls->Add(this->button2);

this->Controls->Add(this->button1);

this->Controls->Add(this->comboBox1);

this->Controls->Add(this->label1);

this->Controls->Add(this->richTextBox1);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::Fixed3D;

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->ImeMode = System::Windows::Forms::ImeMode::Disable;

this->Name = L"MyForm";

this->Text = L"MyForm";

this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

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

}

private: System::Void comboBox1\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

msclr::interop::marshal\_context context;

std::string str = context.marshal\_as<std::string>(richTextBox1->Text);

int how\_many = how\_many\_wariables(str);

int aquations = how\_many\_equations(str);

System::String^ managedString;

if (how\_many == aquations) {

managedString += how\_many;

managedString += " variables\n";

richTextBox2->Text = managedString;

the\_system.create\_system(str);

richTextBox2->Text += the\_system.matrix();

}

else {

richTextBox2->Text = "Number of equations is not equel to number of variables\n";

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

string res;

msclr::interop::marshal\_context context;

switch (comboBox1->SelectedIndex) {

case -1:

MessageBox::Show("Choose the solving method!", "error");

break;

case 0:

res = the\_system.hauss\_method();

richTextBox2->Text += "\nThe Solution:\n";

richTextBox2->Text += context.marshal\_as<System::String^>(res);

break;

case 1:

res = the\_system.jordan\_hauss\_method();

richTextBox2->Text += "\nThe Solution:\n";

richTextBox2->Text += context.marshal\_as<System::String^>(res);

case 2:

res = the\_system.rotation\_method();

richTextBox2->Text += "\nThe Solution:\n";

richTextBox2->Text += context.marshal\_as<System::String^>(res);

break;

case 3:

if (the\_system.number\_of\_variables() != 2) {

MessageBox::Show("This Method is Inapropriate", "error");

break;

}

else {

//strin

double kx1, b1, kx2, b2;

the\_system.graphical\_method(kx1, b1, kx2, b2);

int resultx = the\_system.hauss\_method(1);

chart1->Visible = true;

for (double i = resultx-5; i < resultx+5; i += (double)1) {

chart1->Series["line1"]->Points->AddXY(i, kx1\*i + b1);

chart1->Series["line2"]->Points->AddXY(i, kx2\*i + b2);

}

break;

}

};

}

private: System::Void chart1\_Click(System::Object^ sender, System::EventArgs^ e) {

chart1->Visible = false;

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

msclr::interop::marshal\_context context;

std::string fname = context.marshal\_as<std::string>(textBox1->Text);

if (fname.find(".txt") == string::npos) {

fname += ".txt";

}

std::string text = context.marshal\_as<std::string>(richTextBox2->Text);

ofstream the\_file(fname, ios::trunc);

the\_file << text;

MessageBox::Show("Saved succesfully!!", "YEEEEyy");

}

};

}

Before\_solving.h

#pragma once

#include <string>

using namespace std;

int how\_many\_equations(string raw\_system);

int how\_many\_wariables(string raw\_system);

bool is\_letter(char letter);

bool is\_number(char symbol);

void upper\_triangle\_matrix(double \*\*arr, int length, int height);

int zero\_row(double \*\*arr, int height);

Before\_solving.cpp

#include <string>

#include <iostream>

#include <iomanip>

using namespace std;

bool is\_letter(char letter) {

if (((letter >= 'a') && (letter <= 'z')) || ((letter >= 'A') && (letter <= 'Z'))) {

return true;

}

return false;

}

bool is\_number(char symbol) {

if ((symbol >= '0') && (symbol <= '9'))

return true;

return false;

}

int how\_many\_wariables(string raw\_system) {

string variables;

int how\_many = 0;

for (int i = 0; i < raw\_system.size(); i++) {

if ((variables.find(raw\_system[i]) == string::npos) && is\_letter(raw\_system[i])) {

variables += raw\_system[i];

how\_many++;

}

}

return how\_many;

}

void upper\_triangle\_matrix(double \*\*arr, int length, int height) {

double \*buff;

double multiplication;

for (int row = 0; row < height-1; row++) {

for (int i = row; i < height; i++) {

if (arr[i][row] != 0) {

buff = arr[row];

arr[row] = arr[i];

arr[i] = buff;

break;

}

}

multiplication = arr[row][row];

for (int i = row; i < (length); i++) {

arr[row][i] /= multiplication;

}

for (int i = row + 1; i < height; i++) {

multiplication = arr[i][row];

for (int j = row; j < length; j++) {

arr[i][j] -= (arr[row][j])\*multiplication;

}

}

}

//system("pause");

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

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

cout << setw(8) << arr[i][j];

}

cout << endl;

}

}

int zero\_row(double \*\*arr, int height) {

int j;

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

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

if (arr[i][j] != 0) break;

}

if (j == height) return i;

}

return -1;

}

int get\_determinant(double \*\*arr, int size) {

double \*\*copy = new double\*[size];

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

copy[i] = new double[size];

}

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

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

copy[i][j] = arr[i][j];

//coppied code fron upper triangle matrix;

double determinant = 1;

double \*buff;

double multiplication;

for (int row = 0; row < size - 1; row++) {

for (int i = row; i < size; i++) {

if (copy[i][row] != 0) {

buff = copy[row];

copy[row] = copy[i];

copy[i] = buff;

break;

}

}

multiplication = copy[row][row];

determinant \*= multiplication;

for (int i = row; i < (size); i++) {

copy[row][i] /= multiplication;

}

for (int i = row + 1; i < size; i++) {

multiplication = copy[i][row];

for (int j = row; j < size; j++) {

copy[i][j] -= (copy[row][j])\*multiplication;

}

}

}

multiplication \*= copy[size - 1][size - 1];

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

delete[] copy[i];

}

delete[]copy;

return determinant;

}

int how\_many\_equations(string raw\_system) {

int counter = 0;

if (raw\_system[raw\_system.size() - 1] != '\n') {

raw\_system += '\n';

}

for (int i = 0; i < raw\_system.size(); i++) {

if (raw\_system[i] == '\n') {

counter++;

}

}

return counter;

}

Systema.h

#include <string>

#pragma once

using namespace std;

class systema {

private:

string variables;

double \*\*arr;

int height;

int length;

public:

int hauss\_method(int k);

int hauss\_single\_solution(int k);

string hauss\_method();

string hauss\_single\_solution();

string jordan\_hauss\_method();

string jordan\_hauss\_single\_solution();

string rotation\_method();

void graphical\_method(double &kx1, double &b1, double &kx2, double &b2);

void create\_system(string str);

System::String^ matrix();

int number\_of\_variables();

};

System.cpp

#include "system.h"

#include "before\_solving.h"

#include <stdlib.h>

#include <string>

#include <msclr\marshal\_cppstd.h>

//#include "system.h"

//#include "MyForm.h"

using namespace std;

void systema::create\_system(string str) {

height = how\_many\_wariables(str);

length = height + 1;

// create the array of zeroes

if (arr != nullptr) delete[] arr;

arr = new double\*[height];

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

arr[i] = new double[length];

}

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

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

arr[i][j] = 0;

//getting the list of variables

variables = "";

for (int i = 0; i < str.size(); i++) {

if ((variables.find(str[i]) == string::npos) && is\_letter(str[i])) {

variables += str[i];

}

}

//making matrix

double cnumber = 0; //Number we are going to put into the matrix

int row = 0; // Number of te row in matrix;

//string help;

bool plus = true; // shows is number positive or negative

bool coma = false; // shows presence of coma

double multiply\_coma = 1;

str += "\n";

for (int i = 0; i < str.size(); i++) {

if (is\_number(str[i])) {

if (!coma) {

cnumber \*= 10;

cnumber += (int)str[i] - 48;

}

else{

multiply\_coma \*= 0.1;

cnumber += ((int)str[i] - 48) \* multiply\_coma;

}

}

else if ((is\_letter(str[i])) && (variables.find\_first\_of(str[i]) != string::npos)) {

if (cnumber == 0) {

cnumber = 1;

}

arr[row][variables.find\_first\_of(str[i])] = plus ? cnumber : cnumber \* (-1);

cnumber = 0;

plus = true;

coma = false;

multiply\_coma = 1;

}

else if (str[i] == '-') {

plus = false;

}

else if ((str[i] == '.')||(str[i] == ',')) {

coma = true;

}

else if (str[i] == '\n') {

arr[row][length - 1] = plus ? cnumber : cnumber \* (-1);

row++;

cnumber = 0;

plus = true;

coma = false;

multiply\_coma = 1;

}

}

}

System::String^ systema::matrix() {

System::String^ the\_str;

string help;

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

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

help = to\_string(arr[i][j]);

for (int i = help.size() - 1; i >= 0; i--) {

if (help[i] != '0') break;

else {

help.erase(i);

}

}

if(help[help.size()-1] == '.') help.erase(help.size() - 1);

while(help.size() <= 8){

help += " ";

}

msclr::interop::marshal\_context context;

the\_str += context.marshal\_as<System::String^>(help);

//the\_str += marshal\_as<String^>(help);

//the\_str += arr[i][j];

//the\_str += " ";

}

the\_str += "\n";

}

return the\_str;

}

int systema::number\_of\_variables() {

return variables.size();

}