НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра автоматизованих систем обробки інформації та управління

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з                                           «Основи програмування»

(назва дисципліни)

на тему:                «Розвязання СЛАР наближеними методами»

Студента I курсу ІП-51 групи

напряму підготовки 6.050103 «Програмна інженерія»

спеціальності «Програмне забезпечення систем»

Зарічкового О. А.

(прізвище та ініціали)

Керівник Муха І.П

Доцент кафедри АСОІУ

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

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

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

Київ ‑ 2016 рік

Національний технічний університет України “КПІ”

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Програмна інженерія"

Курс І Група ІП-51 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Зарічкового Олександра Анатолійовича |

(прізвище, ім’я, по батькові)

|  |  |
| --- | --- |
| 1. Тема роботи | Розв'язання СЛАР наближеними методами |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи |  |

|  |  |
| --- | --- |
| 3. Вихідні дані до роботи | Технічне завдання (додаток А) |
|  | |
|  | |
|  | |

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 | 14.03.16-20.03.16 |  |
| 2. | Підготовка ТЗ | 21.03.16-03.04.16 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 04.04.16-17.04.16 |  |
| 4. | Розробка алгоритму вирішення задачі | 18.04.16-24.04.16 |  |
| 6. | Узгодження алгоритму з керівником | 18.04.16-24.04.16 |  |
| 5. | Розробка сценарію роботи програми | 25.04.16-01.05.16 |  |
| 6. | Узгодження сценарію роботи програми з керівником | 25.04.16-01.05.16 |  |
| 7. | Узгодження з керівником інтерфейсу користувача | 25.04.16-01.05.16 |  |
| 8. | Розробка програмного забезпечення | 02.05.16-22.05.16 |  |
| 9. | Налагодження розрахункової частини програми | 02.05.16-22.05.16 |  |
| 10. | Розробка та налагодження інтерфейсної частини програми | 23.05.16-29.05.16 |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу | 23.05.16-29.05.16 |  |
| 12. | Тестування програми | 30.05.16-05.06.16 |  |
| 13. | Підготовка пояснювальної записки | 06.06.16-12.06.16 |  |
| 14. | Здача курсової роботи на перевірку | 06.06.16-12.06.16 |  |
| 15. | Захист курсової роботи |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Зарічковий О. А. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | Муха І. П. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 80 сторінок, 22 рисунка,   
8 таблиць, 10 посилань.

Об’єкт дослідження: електронні таблиці.

Мета роботи: створення програмного забезпечення для спрощення процесу громіздких обчислень та представлення інформації у зручному для подальшого аналізу вигляді (таблиця).

Вивчено методи реалізації електронних таблиць, особливості реалізаї динамічного інтерфейсу, можливості метапрограмування в мові Visual С++, робота з потоками та класами платформи Microsoft .NET. Приведені змістовні постановки задач, їх індивідуальні математичні моделі, а також описано детальний процес розв’язання кожної з них.

Виконана програмна реалізація електронної таблиці.

ЕЛЕТРОННІ ТАБЛИЦІ, ГРАФИ, ТОПОЛОГІЧНЕ СОРТУВАННЯ, АЛГОРИТМ СОРТУВАЛЬНОЇ СТАНЦІЇ, ОБЕРНЕНА ПОЛЬСЬКА НОТАЦІЯ

Зміст

[Вступ 6](#_Toc451632576)

[1 Постановка задачі 7](#_Toc451632577)

[2 Теоретичні відомості 8](#_Toc451632578)

[2.1. Метод Якобі 8](#_Toc451632579)

[2.2. Метод Гауса-Зейделя 9](#_Toc451632580)

[2.3. Метод градієнтного спуску (спряжених градієнтів) 10](#_Toc451632581)

[3 Опис алгоритмів 13](#_Toc451632582)

[3.1. Загальний алгоритм 13](#_Toc451632583)

[3.2. Алгоритм методу Якобі 14](#_Toc451632584)

[3.3. Алгоритм методу Гауса-Зейделя 17](#_Toc451632585)

[3.4. Алгоритм методу градієнтного спуску 19](#_Toc451632586)

[4 Опис програмного забезпечення 21](#_Toc451632587)

[4.1. Функціональна структура програмного забезпечення 21](#_Toc451632588)

[4.2. Опис функцій частин програмного забезпечення 21](#_Toc451632589)

[4.2.1. Користувацькі функції 22](#_Toc451632590)

[4.2.2. Стандартні функції 31](#_Toc451632591)

[5 Тестування програмного забезпечення 33](#_Toc451632592)

[5.1. План тестування 33](#_Toc451632593)

[5.2. Приклади тестування 33](#_Toc451632594)

[5.2.1. Виконання програми, коли у матрицю системи невірно введено число 33](#_Toc451632595)

[5.2.2. Виконання програми, коли у стовпець вільних членів невірно введено число 34](#_Toc451632596)

[5.2.3. Виконання програми, коли у матрицю системи введено число з точністю більшою, ніж 3 знаки після коми 35](#_Toc451632597)

[5.2.4. Перевірка вірності роботи програми, коли користувачем вірно введені всі дані та для обраного методу рішення сходиться 36](#_Toc451632598)

[6 Інструкція користувача 38](#_Toc451632599)

[6.1. Робота з програмою 38](#_Toc451632600)

[6.2. Формат вхідних та вихідних даних 43](#_Toc451632601)

[6.3. Системні вимоги 43](#_Toc451632602)

[7 Аналіз і узагальнення результатів 45](#_Toc451632603)

[Висновки 50](#_Toc451632604)

[Перелік посилань 51](#_Toc451632605)

[**Додаток А Технічне завдання** 52](#_Toc451632606)

[Додаток Б Тексти програмного коду 55](#_Toc451632607)

Вступ

Електронна таблиця являє собою інтерактивну комп'ютерну програму для організації, аналізу і зберігання даних в табличній формі. [1] [2] [3] Електронні таблиці розроблені як комп'ютеризованих моделювання бухгалтерського обліку паперу робочих листів. [4] Програма працює на даних, введених в комірки таблиці. Кожна клітинка може містити або числові або текстові дані або результати формул, які автоматично обчислювати і відображати значення, засноване на утриманні інших клітин. Електронна таблиця може також стосуватися одного такого електронного документа. [5] [6] [7] користувачі електронних таблиць може налаштувати будь-яку збережене значення і спостерігати ефекти на розрахункових значень. Це робить таблицю корисною для "що-якщо" аналіз, так як багатьох випадках може бути швидко досліджена без ручного перерахунку.

Дана курсова робота призначена для використання в області бугалтерського обліку та в усіх розділах аналізу інформації. Вона значно спростить процес створення власних таблиць, спростить користувачам процес громіздких обчислень та надасть можливість представлення інформації у зручному для подальшого обробки та аналізу вигляді.

Дана курсова робоба є актуальною оскільки…

Для вирішення даної задачі необхідно розробити зручний для користувача інтерфейс, механізми обробки введених користувачем формул з можливістю посилатися на інші комірки таблиці. Передбачити динамічну змінну вмісту комірок якщо користувач змінив дані від яких залежатить вміст цих комірок. Необхідно добавити можливість збереження та зчитування таблиці з файла.

Для реалізації обробки формул ми будем використовувати алгоритм сорутвальної станції, який розроблений нідерландським науковцем у галузі комп'ютерних наук – Едгерсом Дейкстрою, який преводить вираз заданий в інфіксній нотації в обернену польску нотацію. В такому вигляді вираз можна обчислити за допомогою відомого алгоритма з використанням стеку.

Для реалізації механізму пререрахунку формул ми будемо використовувати підхід перерахунку по графу, що попередньо топологічно відсортований.

# Постановка задачі

Розробити програмне забезпечення, що буде реалізовувати електорнні таблиці.

Програма повинна забезпечувати можливість виконання нижче наведених функцій:

1. Створення таблиці довільного розміру;
2. Введення та обчислення алгебраїчних виразів
3. Обчислення форму, які посилаються на інші елементи таблиці
4. Обчислення деяких елементарних функцій
5. Переобчислення значень залежних комірок таблиці
6. Виявлення циклічних посилань у комірках та блокування можливості їх обчислення
7. Збереження та відкриття таблиці з файлу (\*.adc)

Вхідні дані до програми повинні бути організовані у вигляді файлів, відповідної специфікації та з розширенням (\*.adc).

Вихідними даними для програми організовані у вигляді файлів, відповідної специфікації та з розширенням (\*.adc).

# Теоретичні відомості

Для вирішення поставленої задачі необхідно розробити механізми обробки введених користувачем формул з можливістю посилатися на інші комірки таблиці. Передбачити динамічну змінну вмісту комірок якщо користувач змінив дані від яких залежатить вміст цих комірок.

Для реалізації обробки формул ми будем використовувати алгоритм сорутвальної станції. Алгоритм сортувальної станції — метод аналізу математичних виразів, які представлені в інфіксной нотації. Він може бути використаний для отримання математичного виразу у вигляді зворотної польської нотації або у вигляді абстрактного синтаксичного дерева. Алгоритм винайдений Едсгерсом Дейкстрою і названий ним «алгоритм сортувальної станції», оскільки нагадує дію залізничної сортувальної станції.

Так само, як і обчислення значень виразів в зворотної польської записи, алгоритм працює за допомогою стека. Інфіксной запис математичних виразів найчастіше використовується людьми, її приклади: 2 + 4 і 3 + 6 \* (3-2). Для перетворення в зворотний польський нотацію використовується 2 рядки: вхідний і вихідний, і стек для зберігання операторів, ще не доданих в вихідну чергу. При перетворенні алгоритм зчитує 1 символ і виробляє дії, що залежать від даного символу.

Кожен токен-число, функція або оператор виводиться тільки один раз, а також кожен токен-функція, оператор або кругла дужка буде додано і видалений з стека по одному разу. Постійна кількість операцій на токен дає в результаті лінійну складність алгоритму сортувальної станції O (n) [8].

На рисунку 2.1 зображено реалізацію алгоритму , що використовує тригілкове залізничний вузол. Початкові дані опрацьовуються по одному символу за раз, якщо отримана змінна або номер вона копіюється прямо на вихід b), d), f), h). Якщо ж це оператор, він заштовхується в стек операторів c), e), однак, якщо старшинство менше ніж у оператора на верхівці стека або вони мають однакове старшинство й оператор лівоасоціативний, тоді той оператор виштовхується зі стека й записується на вихід g). Насамкінець оператор, що залишились у стеці, виштовхуються і додаються до виходу.

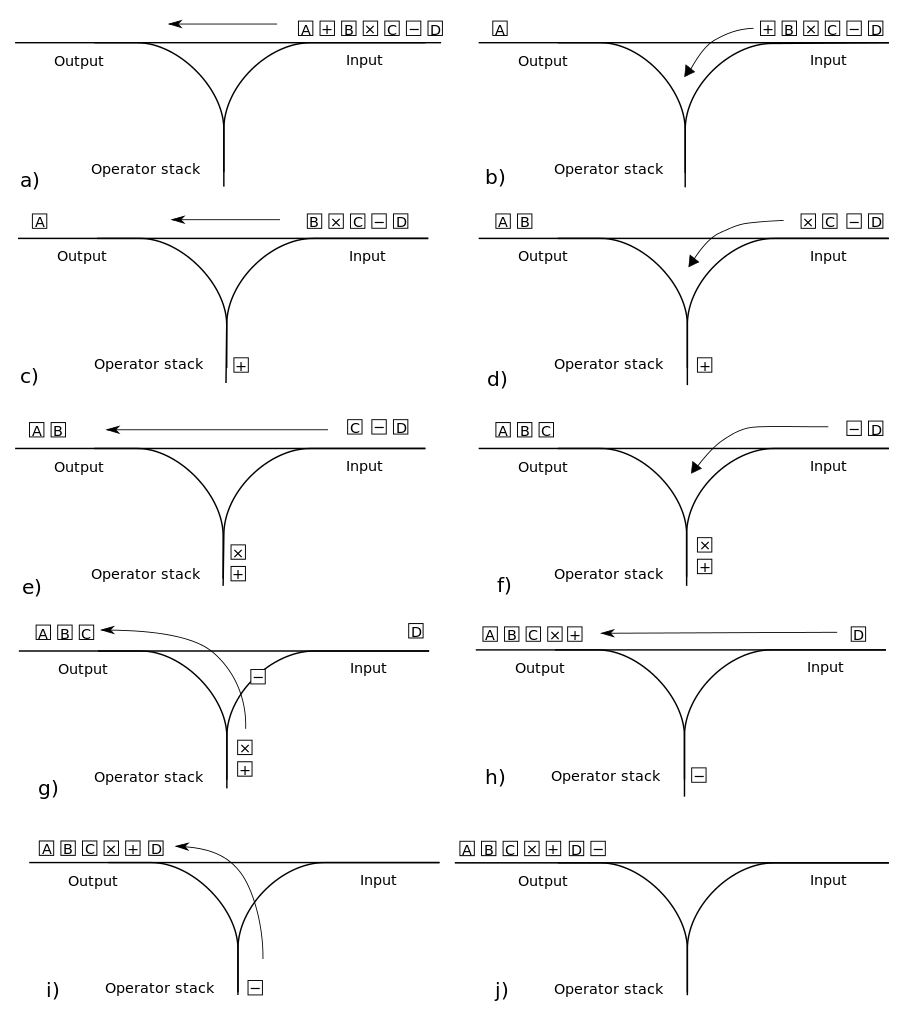


Рисунок 2.1 – Приклад роботи алгоритму сортвальної станції

Для реалізації механізму пререрахунку формул ми будемо використовувати підхід перерахунку по графу, що попередньо топологічно відсортований.

Для реалізації алгортму топологічного сортування необхідно представити табилицю у вигляді графу. Для цього представимо кожну комірку таблиці, що містить будь-яку інформацію як вершину графу та проведем ребра від даної вершини до тих вершин в яких вона використовується для обчислення формули. Приклад такого графу наведено на Рисунку 2.2.

C:\Users\Саша\Downloads\Untitled Diagram.png

Рисунок 2.2 – Приклад графу електронної таблиці з трьома комірками

Коли такий граф стоврений можна його топологічно відсортувати. Топологічне сортування — впорядковування вершин безконтурного орієнтованого графа згідно з частковим порядком, визначеним ребрами цього графу на множині його вершин.

Наприклад, для графу (рисунок 2.3)

,

існує декілька узгоджених послідовностей його вершин, які можуть бути отримані за допомогою топологічного сортування, наприклад:

* 7, 5, 11, 3, 8, 2, 9, 10
* 3, 7, 5, 8, 11, 10, 9, 2
* 7, 5, 11, 2, 3, 8, 9, 10

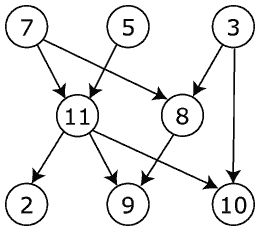


Рисунок 2.3 – Приклад граф, який можливо топологічно впорядкувати.

Час виконання для звичайного алгоритму топологічного сортування лінійний до кількості вершин плюс кількість ребер O(|V|+|E|) [9].

# Опис алгоритмів

# Опис програмного забезпечення

## Функціональна структура програмного забезпечення

На рисунку 4.1 Ви можете бачити загальну структуру додатку, де наявні усі класи програми та стрілками позначено наслідування класів (стрілка напрямлена до Базового класу).

## Опис функцій частин програмного забезпечення

В ході виконання поставленого завдання було створено наступні модулі та бібліотеки:

а) Home.h – реалізує інтерфейс користувача;

б) Graph.h – реалізує побудову та виведення графіка;

в) Jacobi.h – реалізує знаходження розв’язку СЛАР методом Якобі;

г) Gauss-Seidel.h – реалізує знаходження розв’язку СЛАР методом   
Гауса-Зейделя;

д) GradientDecent.h – реалізує знаходження розв’язку СЛАР методом градієнтного спуску;

е) Utilities.h – реалізує допоміжні функції (знаходження визначника матриці системи, перевірки матриці системи на симетричність і т.д.), які необхідні для розв’язання поставленої задачі;

### Користувацькі функції

Користувацькі функції, які використані в даній курсовій роботі, описані в таблиці 4.1.

Таблиця 4.1 – Опис користувацьких функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | Cell | Cell | Конструктор класу | - | - |
| 2 | Cell | setValue | Присвоєння нового занчення поля value | Нове значення поля value | - |
| 3 | Cell | getValue | Отримання поточного значення поля value | - | Поточне значення поля value |
| 4 | Cell | getResult | Отримання поточного значення поля result | - | Поточне значення поля result |
| 5 | Cell | setResult | Присвоєння нового занчення поля result | Нове значення поля result | - |
| 6 | Cell | getIsFormula | Отримання поточного значення поля isFormula | - | Поточне значення поля isFormula |
| 7 | Cell | setIsFormula | Присвоєння нового занчення поля isFormula | Нове значення поля isFormula | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 8 | Graph | Graph | Конструктор класу | Висота та ширина поточної таблиці | - |
| 9 | Graph | getListOfCells | Визначення переліку комірок, які входять до складу формули | Формула, розміри таблиці, послання на масив в яких необхідно зберегрти результат | - |
| 10 | Graph | topologicalSort | Виконує топологічне сортування графу, починаючи з поточної комірки | Коородинати поточної комірки, посилання на таблицю, посилання на масив результату сортування та посилання на масив  циклів | - |
| 11 | Graph | dfs | Виконує обхід графу у глубину | Номер поточної вершини, посилання на масив відвіданих вершин, посилання на масив результату обходу та посилання на масив циклів |  |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 12 | Graph | getY | Визначає Y-координату комірки у таблиці | Строка, що містить лінк (посилання), індекс з якої починати розбір Y-координату, ширина таблиці | Число, Y-координата лінку |
| 13 | Graph | getX | Визначає Y-координату комірки у таблиці | Строка, що містить лінк (посилання), індекс з якої починати розбір X-координату, висота таблиці | Число, X-координата лінку |
| 14 | Graph | changeGraph | Зміна графа при зміні вмісту комірки таблиці | Посилання на таблицю, посилання на поточне представлення таблиці, координати зміненої комірки та чи містила раніше поточна комірка формулу | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 15 | Graph | toStdWstring | Перетворює строку С# в строку С++ | Строка С# | Строка С++ |
| 16 | Parser | Parser | Розрахунок значення заданої формули | Формула, яку необхідно підрахувати та посилання на таблицю | Значення формули |
| 17 | Parser | shuntingYard | Перетворення ініксного формату запису в постфіксний | Формула в ініксному форматі | Формула в постфіксному форматі |
| 18 | Parser | opPrior | Визначення пріорітету елементарних операцій | Символ операції | Пріотрітет операції |
| 19 | Parser | isElemOper | Визначення чи є дана операція елементарною | Символ операції | True якщо задана операція елементарна, інакше False |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 20 | Parser | getX\_index | Повертає Х-координату заданого лінку (посилання) | Строка, що містить лінк, індекс звідки починається розбір, висота таблиці | Число, X-координата лінку |
| 21 | Parser | getY\_index | Повертає Y-координату заданого лінку (посилання) | Строка, що містить лінк, індекс звідки починається розбір, ширина таблиці | Число, Y-координата лінку |
| 22 | Parser | strcmp | Порівнянн назви поточної функції з назвою функції з бібліотеки доступних | Строка, що містить назву функції та індекс її першого символу назви, індекс назви функції з бібліотеки | True, якщо назви співпадають, інашкше False |
| 23 | Parser | processNumbers | Обробка чисел | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 24 | Parser | processCloseBracket | Обробка закриваючих дужок | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці, стек операцій | - |
| 25 | Parser | processElemOperations | Обробка елементарних операцій | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці, стек операцій | - |
| 26 | Parser | processLink | Обробка посилань | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці | - |
| 27 | Parser | processFunctions | Обробка функцій | Вхідна строка, вихідна строка, індекс початку розбору в вхідній строці, стек операцій | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 28 | Parser | calculateNumber | Обчислення значення числа | Вхідна строка, стек чисел, індекс значення в вхідній строці | - |
| 29 | Parser | calculateLink | Обчислення значення посилання | Посилання на таблицю, вхідна строка, стек чисел, індекс значення в вхідній строці | - |
| 30 | Parser | calculateElemOperations | Обчислення елементарних операцій | Вхідна строка, стек чисел, індекс значення в вхідній строці | - |
| 31 | Parser | calculateFunctions | Обчислення функції | Вхідна строка, стек чисел, індекс значення в вхідній строці | - |
| 32 | Parser | searchAssigmentSymbol | Пошук символа присвоєння | Вхідна строка | Індекс символу присвоєння |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 33 | Parser | caseFuction | Обчислення значення функцій | Індекс функції у бібліотеці та її аргумент | Значення функції |
| 34 | Parser | isOnlyOneDigit | Обробка строки, що складається з одного числа | Вхідна строка та індекс з якого починаєтся пошук | Значення єдиного числа |
| 35 | Table | Table | Конструктор класу | Розміри таблиці | - |
| 36 | Table | getHeight | Отримання поточного значення поля height | - | Поточне значення поля height |
| 37 | Table | getWidth | Отримання поточного значення поля width | - | Поточне значення поля width |
| 38 | Table | changeHeight | Зміна значення поля height на певна значення | Значення на яке треба змінити поле height | - |
| 39 | Table | changeWidth | Зміна значення поля width на певна значення | Значення на яке треба змінити поле width | - |

### Стандартні функції

Стандартні функції, які використані в даній курсовій роботі, описані в   
таблиці 4.2.

Таблиця 4.2 – Опис стандартних функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | round | Округлення числа до найближчого цілого | Число, що необхідно округлити | Округлене число | сmath |
| 2 | abs | Знаходження абсолютного значення числа | Число для якого необхідно знайти абсолютне його значення | Абсолютне значення числа | cmath |
| 3 | min | Знаходження мінімуму двох чисел | Два числа з яких необхідно обрати мінімум | Мінімум з двох чисел | Algorithm |
| 4 | ToDouble | Приводить до типу double певну змінну | Змінна простого типу | Число з плаваючою крапкою | - |
| 5 | sin | Обчислення функції sin | Параметр функції | Значення функції для заданого параметру | сmath |

# 

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 6 | cos | Обчислення функції cos | Параметр функції | Значення функції для заданого параметру | сmath |
| 7 | tan | Обчислення функції tan | Параметр функції | Значення функції для заданого параметру | сmath |
| 8 | ctg | Обчислення функції ctg | Параметр функції | Значення функції для заданого параметру | сmath |
| 9 | ln | Обчислення функції ln | Параметр функції | Значення функції для заданого параметру | сmath |
| 10 | exp | Обчислення функції exp | Параметр функції | Значення функції для заданого параметру | сmath |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 11 | asin | Обчислення функції asin | Параметр функції | Значення функції для заданого параметру | сmath |
| 12 | acos | Обчислення функції acos | Параметр функції | Значення функції для заданого параметру | сmath |
| 13 | atan | Обчислення функції atan | Параметр функції | Значення функції для заданого параметру | сmath |

# Тестування програмного забезпечення

## План тестування

Усі можливі випадки виникнення помилок у програмі залежать від вхідних даних, тобто тої інформації, що отримується від користувача. Тому тестування програми полягає у виявленні правильності та коректності обробки програмою різних вхідних даних.

Задля виявлення усіх помилок у роботі програми потрібно запустити її на виконання в таких умовах:

а) у матрицю системи або стовпець вільних членів невірно введено число (або введена інформація не являється числом);

б) у матрицю системи або стовпець вільних членів введено число з точністю більшою, ніж 3 знаки після коми;

Для вирішення відповідних помилкових ситуацій слід вдосконалити алгоритми роботи програми та обробити всі виключні ситуації.

Далі слід упевнитись, що усі методи програми коректно працюють на усіх наборах вхідних значень.

## Приклади тестування

### Виконання програми, коли у матрицю системи невірно введено число

Якщо під час введення інформації в матрицю системи користувач введе недозволений символ, то програма виведе у спеціально відведене місце на графічному інтерфейсі (далі: поле системної інформації) відповідне повідомлення, зафарбує в червоний колір комірку таблиці в якій було зроблено помилку та заблокує можливість натискання кнопки «Знайти рішення» (рисунок 5.1) до тих пір, поки користувач не виправить вхідні дані.

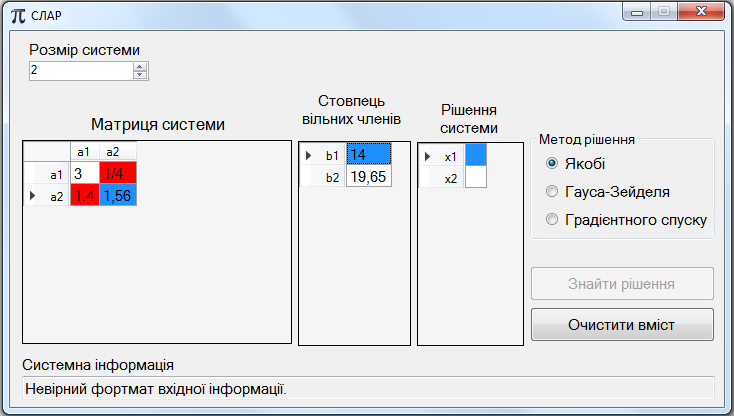


Рисунок 5.1 – Вікно програми, після введення некоректної інформації в матрицю системи

### Виконання програми, коли у стовпець вільних членів невірно введено число

Якщо під час введення інформації в стовпець вільних членів користувач введе недозволений символ, то програма виведе у поле системної інформації відповідне повідомлення, зафарбує в червоний колір комірку таблиці в якій було зроблено помилку та заблокує можливість натискання кнопки «Знайти рішення»   
(рисунок 5.2) до тих пір, поки користувач не виправить вхідні дані.

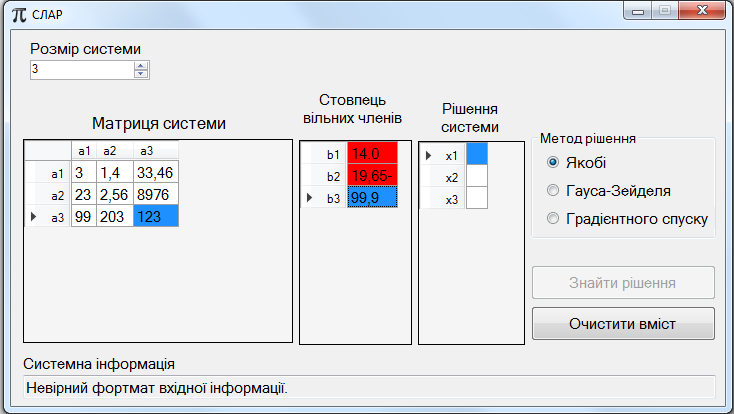


Рисунок 5.2 – Вікно програми, після введення некоректної інформації в стовпець вільних членів

### Виконання програми, коли у матрицю системи введено число з точністю більшою, ніж 3 знаки після коми

Якщо під час введення інформації в матрицю системи користувач введе число з точністю більшою, ніж 3 знаки після коми (рисунок 5.3), то програма автоматично округлить його до 3-х знаків за математичними правилами (рисунок 5.4).

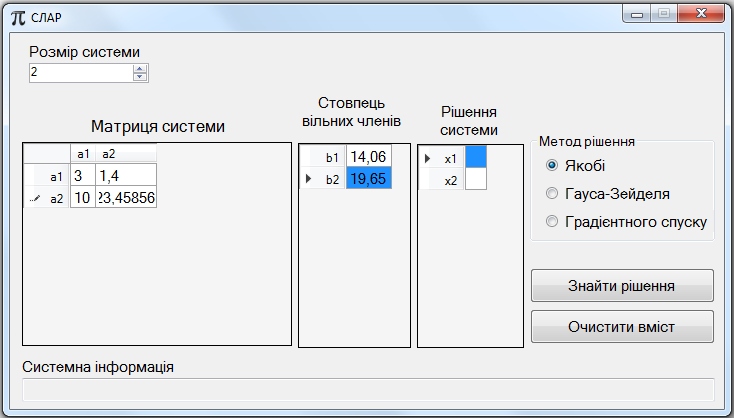


Рисунок 5.3 – Вікно програми під час введення числа з точністю більшою,   
ніж 3 знаки після коми

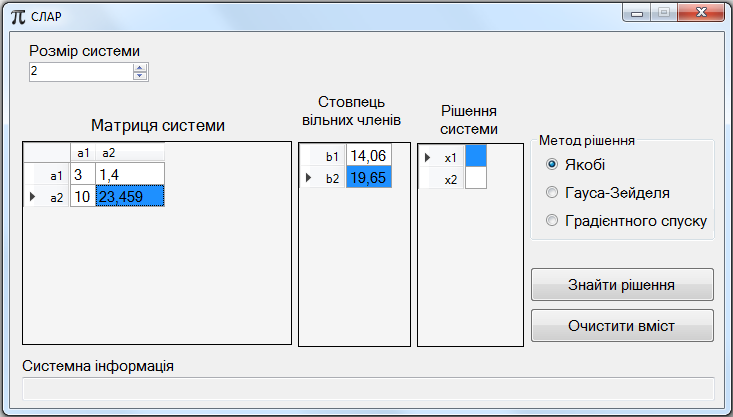


Рисунок 5.4 – Вікно програми після завершення введення числа з точністю  
 більшою, ніж 3 знаки після коми

### Перевірка вірності роботи програми, коли користувачем вірно введені всі дані та для обраного методу рішення сходиться

Результати тестування програми на коретних даних наведена у   
таблицях 5.1 - 5.3.

Таблиця 5.1 – Тестування методу Якобі

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані методу Якобі на коректних даних |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір: 2; Метод Якобі |
| Схема проведення тесту | Введення даних та натискання кнопки «Знайти рішення» |
| Очікуваний результат | Виведення рішення системи та побудовано графік системи |
| Стан програми після проведення випробувань | Виведено рішення системи та відкрито вікно з графіком системи |

Таблиця 5.2 – Тестування методу Гауса-Зейделя

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані методу Гауса-Зейделя на коректних даних |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір: 3; Метод Гауса-Зейделя |
| Схема проведення тесту | Введення даних та натискання кнопки «Знайти рішення» |
| Очікуваний результат | Виведення рішення системи |
| Стан програми після проведення випробувань | Виведено рішення системи |

Таблиця 5.3 – Тестування методу градієнтного спуску

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані методу градієнтного спуску на коректних даних |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір: 5; Метод градієнтного спуску |
| Схема проведення тесту | Введення даних та натискання кнопки «Знайти рішення» |
| Очікуваний результат | Виведення рішення системи |
| Стан програми після проведення випробувань | Виведено рішення системи |

# Інструкція користувача

## Робота з програмою

Після запуску виконавчого файлу з розширенням \*.exe, відкривається головне вікно програми (Рисунок 6.1).

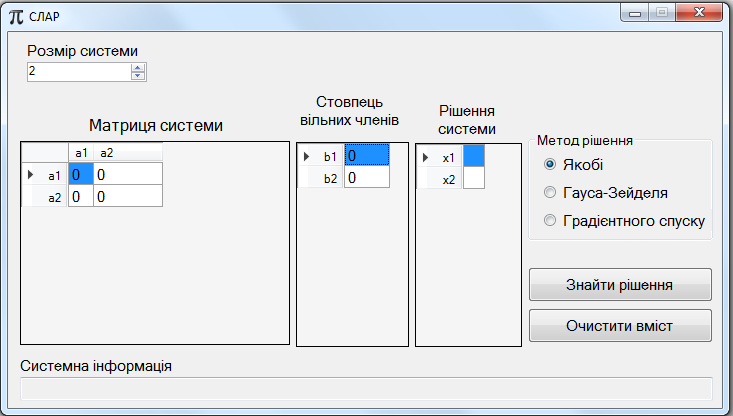


Рисунок 6.1 – Головне вікно програми

Далі за допомогою лічильника з назвою «Розмір системи» шляхом натиску на стрілки або введенням числа з клавіатури необхідно виставити розмір системи, що буде оброблятися програмою (рисунок 6.2):

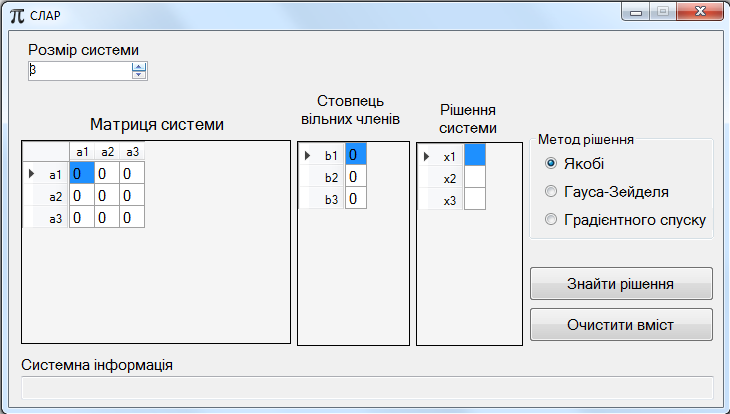


Рисунок 6.2 – Вибір необхідного розміру системи

Далі необхідно записати матрицю системи та стовпець вільних членів в матриці з відповідними назвами (Рисунок 6.3):

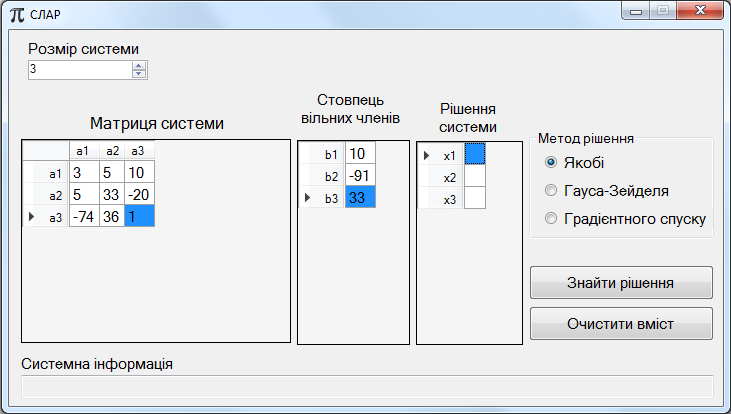


Рисунок 6.3 – Введення вхідних даних

Якщо якісь вхідні дані не є числами, то програма видасть відповідне повідомлення у полі «Системна інформація» та зафарбує комірки з некоректними даними в червоний колір (Рисунок 6.4). Для того щоб можна було продовжити роботу з програмою користувач повинен виправити вхідні дані.

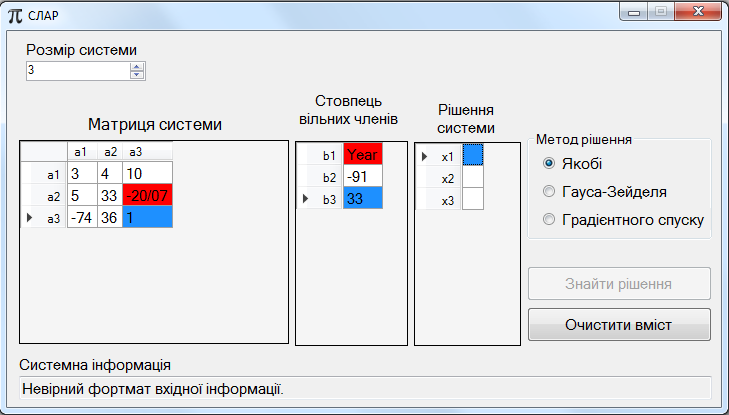


Рисунок 6.4 – Приклад некоректного введення даних користувачем

Після того як дані введені коректно в полі «Метод рішення» необхідно обрати метод, яким користувач бажає знайти розв’язки введеної системи та натиснути кнопку «Знайти рішення». Якщо для даної системи існує розв’язок та обраний метод сходить, то для неї буде виведено рішення у матрицю «Рішення системи» та буде записано у текстовий файл саму систему та її рішення (Рисунок 6.5):

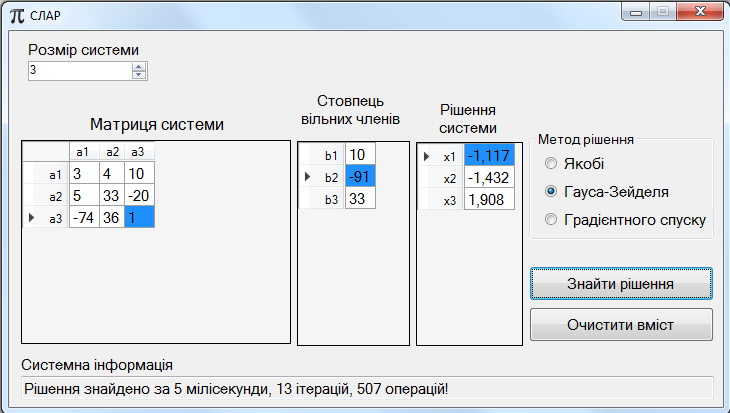


Рисунок 6.5 – Приклад результату роботи програми

Якщо для даної системи не існує розв’язку або їх безліч, то програмою в поле системної інформації буде виведене відповідне повідомлення (Рисунок 6.6):

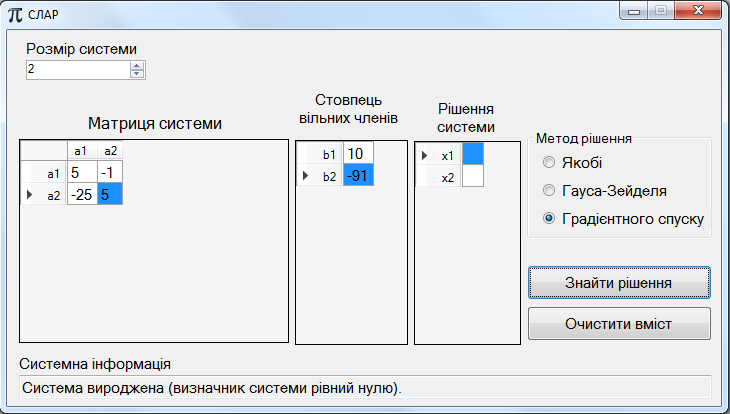


Рисунок 6.6 – Приклад введення системи для якої не існує розв’язку

Якщо для даної системи обраний метод не сходиться, то програмою в поле системної інформації буде виведене відповідне повідомлення (Рисунок 6.7):

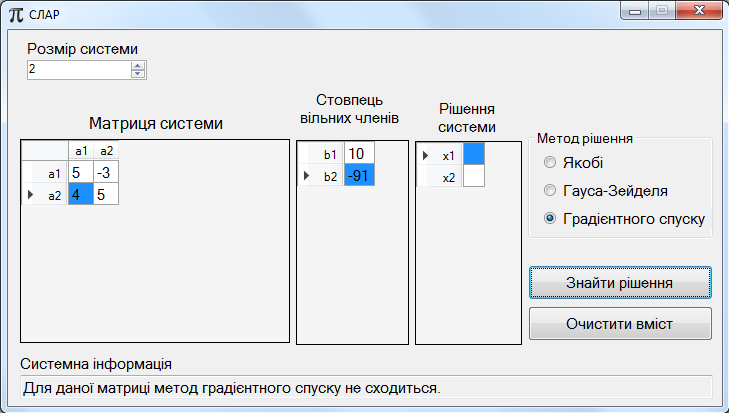


Рисунок 6.7 – Приклад введення системи для якої обраний метод   
не сходиться

Якщо під час виконання програми було обрано розмірність матриці «2» та записано систему для якої обраний метод сходиться, то програмою задля візуалізації рішення буде виведено графік системи (Рисунок 6.8):

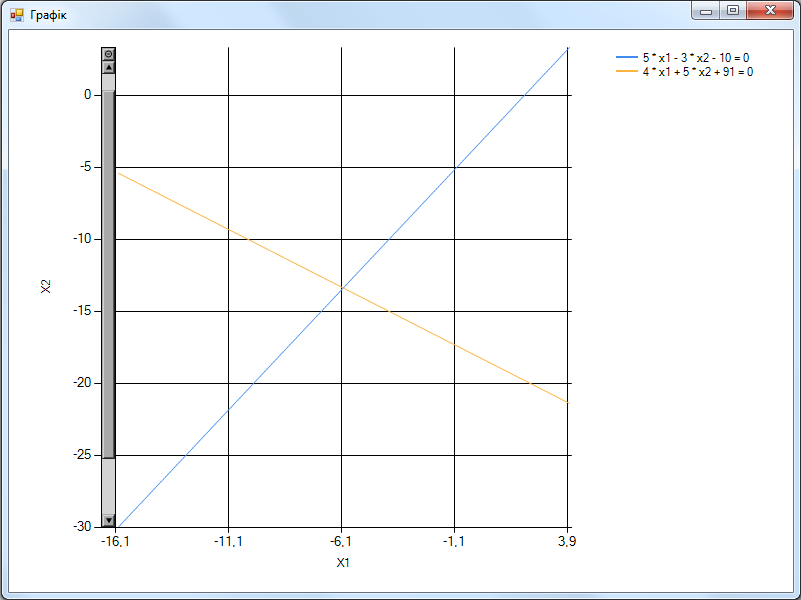


Рисунок 6.8 – Приклад введення графіку системи

Даний графік можна масштабувати виділивши необхідну область в прямокутник шляхом клацання лівою кнопкою та протягування миші   
(Рисунок 6.9 та Рисунок 6.10):

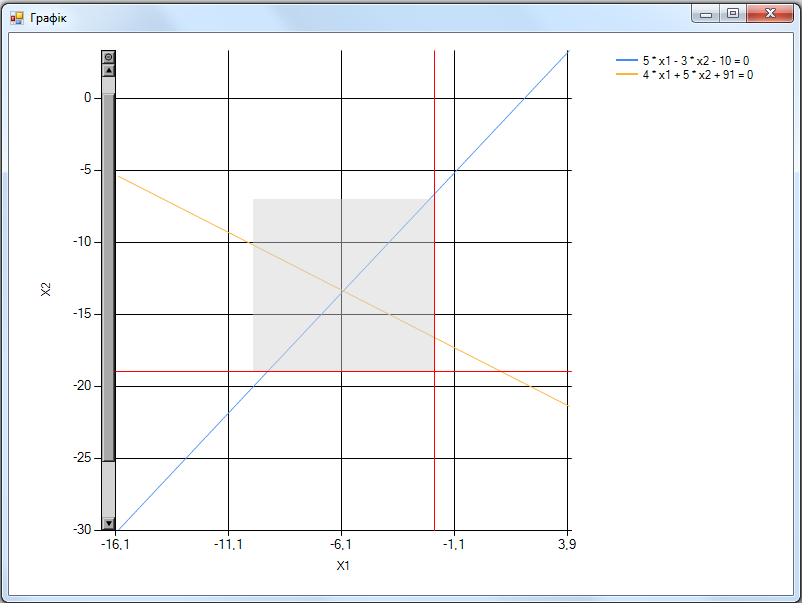


Рисунок 6.9 – Приклад виділення області графіку для масштабування

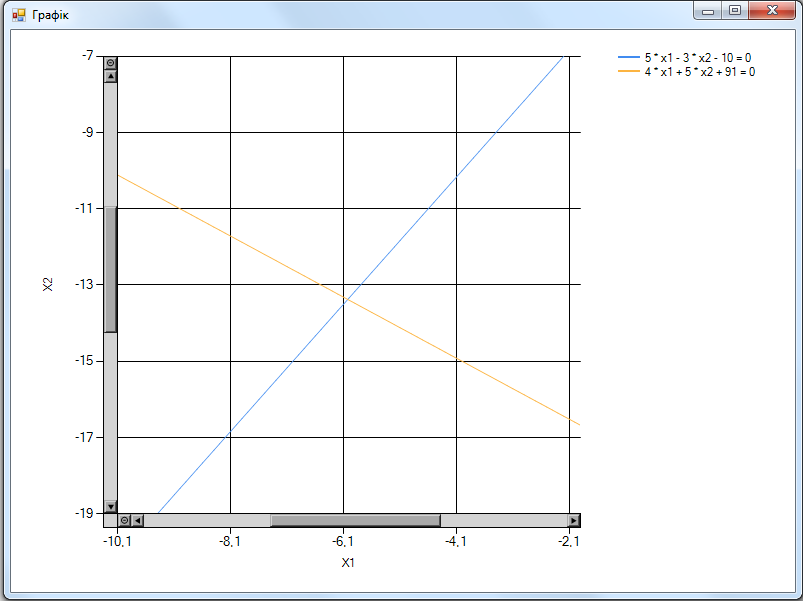


Рисунок 6.10 – Рисунок 6.9 після масштабування

Також можливо рухати графік системи за допомогою полос прокручування, що знаходяться знизу та зліва графіка. Якщо користувачеві необхідно зменшити масштаб по якійсь осі то, необхідно натиснути на кнопку на відповідні полосі прокручування.

Якщо користувачеві необхідно очисти всі введені вхідні дані, то це можна зробити за допомогою кнопки «Очистити вміст». Після її натискання матриця системи та стовпець вільних члені заповняться нулями, а вся інформація в таблиці «Рішення системи» буде стерта.

## Формат вхідних та вихідних даних

Користувачем на вхід програми подається СЛАР у матричному вигляді, тобто задається за допомогою матриці системи та стовпця вільних членів, числа яких дійсні, з точністю не більше, ніж 3 знака після коми (якщо точність більша, то програма автоматично округлить їх за математичними правилами до 3-х знаків після коми).

Результатом виконання програми є розв’язок зданої СЛАР, який видається у вигляді таблиці кожне число якої записане з точністю до 3-х знаків після коми та текствий файл, у який записано вхідну систему та її рішення, або повідомлення, що дана система не має розв’язків або не сходиться для обраного методу.

## 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 (з останніми обновленнями) | Windows 7/ Windows 8/Windows 10  (з останніми обновленнями) |
| Процесор | Intel® Pentium® ІІІ  1.0 GHz або  AMD Athlon™ 1.0 GHz | Intel® Pentium® D або AMD Athlon™ 64 X2 |

Продовження таблиці 6.1

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| 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 GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог) | |
| Дисплей | 800х600 | 1024х768 або краще |
| Прилади введення | Клавіатура, комп’ютерна миша | |
| Додаткове програмне забезпечення | Microsoft .Net Framework 4.5.2 або вище | |

# Аналіз і узагальнення результатів

Головною задачею курсової роботи була реалізація програми для розв’язання СЛАР наступними методами: Якобі, Гауса-Зейделя, градієнтного спуску.

Критичні ситуації у роботі програми виявлені не були. Під час тестування було виявлено, що більшість помилок виникало тоді, коли користувачем вводилися не числові вхідні дані. Тому всі дані, які вводить користувач, ретельно провіряються на валідність і лише потім подаються на обробку програмі.

Для перевірки та доведення достовірності результатів виконання створеного програмного забезпечення скористаюся MS Excel:

а) метод Якобі:

Результат виконання методу Якобі наведено на рисунку 7.1:

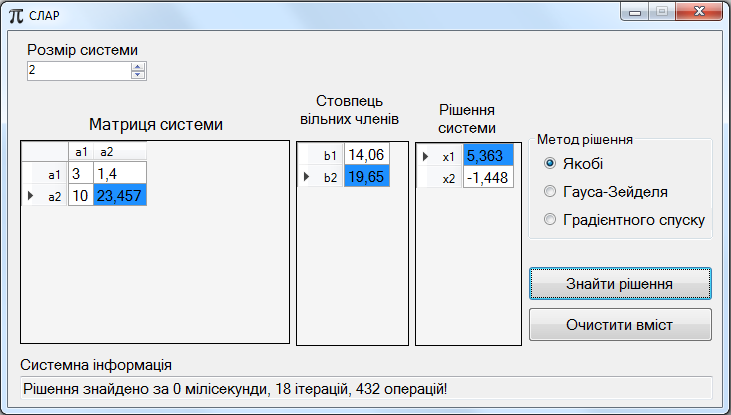


Рисунок 7.1 – Результат виконання методу Якобі

Оскільки результат виконання збігається з результатом в MS Excel   
(рисунок 7.2), то даний метод працює вірно.

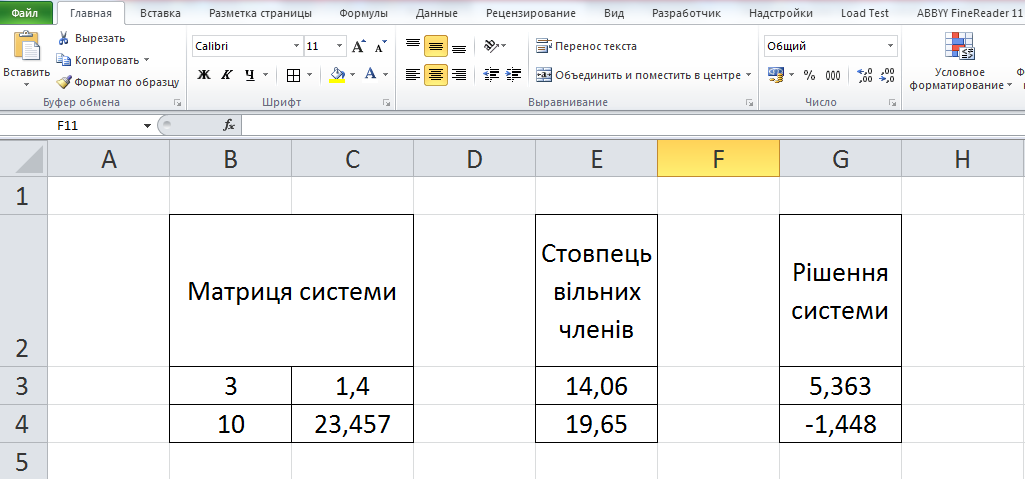


Рисунок 7.2 – Перевірка методу Якобі в MS Excel 2010

б) метод Гауса-Зейделя:

Результат виконання методу Гауса-Зейделя наведено на рисунку 7.3:

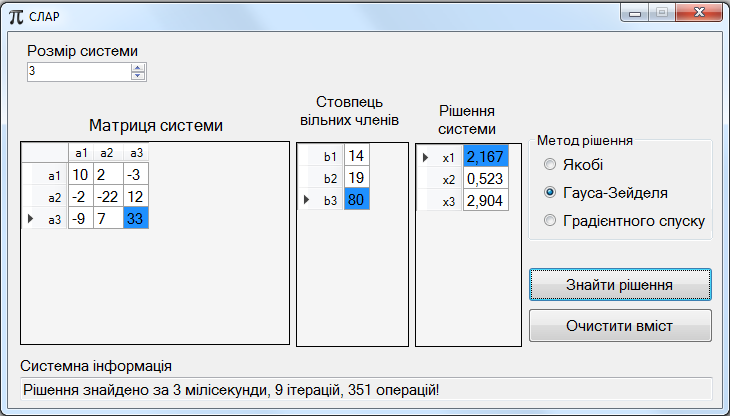


Рисунок 7.3 – Результат виконання методу Гауса-Зейделя

Оскільки результат виконання збігається з результатом в MS Excel   
(рисунок 7.4), то даний метод працює вірно.

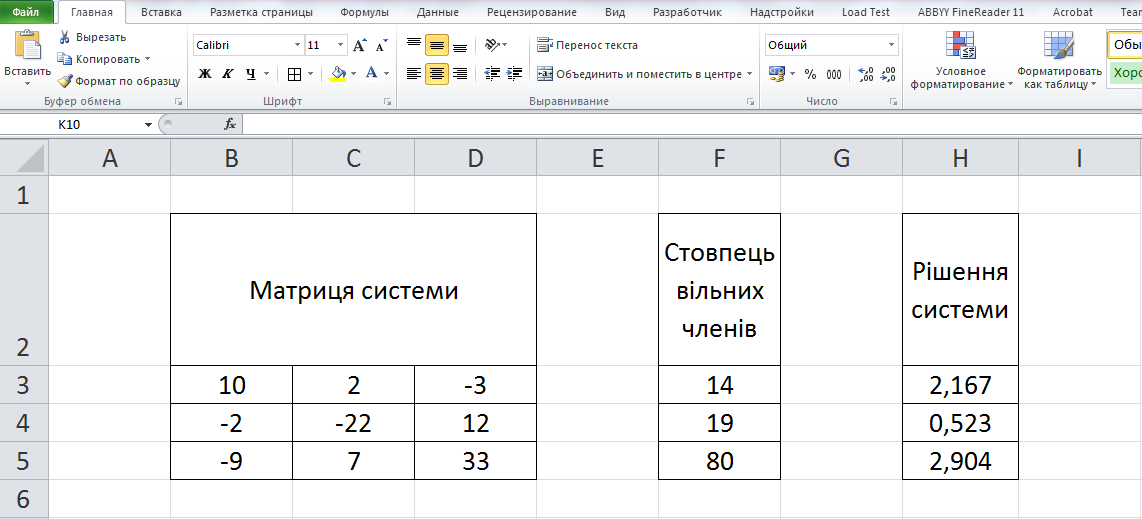


Рисунок 7.5 – Перевірка методу Гауса-Зейделя в MS Excel 2010

в) метод градієнтного спуску:

Результат виконання методу градієнтного спуску наведено на рисунку 7.5:

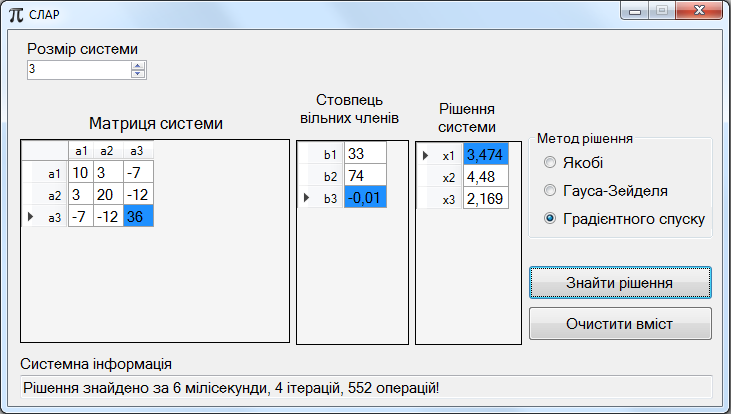


Рисунок 7.5 – Результат виконання методу градієнтного спуску

Оскільки результат виконання збігається з результатом в MS Excel   
(рисунок 7.6), то даний метод працює вірно.

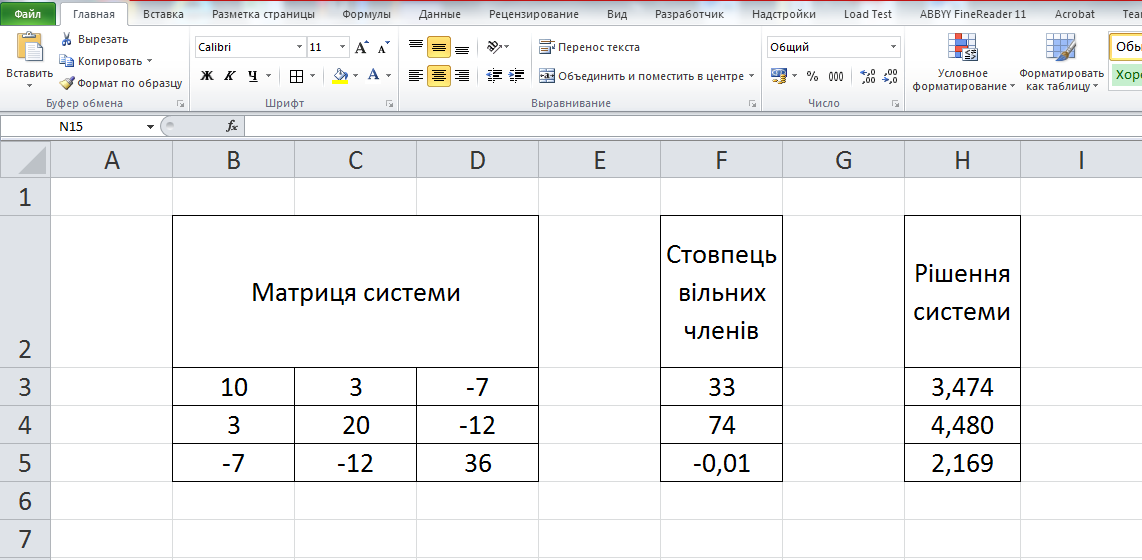


Рисунок 7.6 – Перевірка методу градієнтного спуску в MS Excel 2010

Для проведення тестування ефективності програми було створено матриці наступного вигляду:

(7.1),

де – розмірність системи.

Матриця (7.1) для для довільного додатного є симетричною, додатньо визначеною та має домінантну головну діагональ.

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

Таблиця 7.1 – Тестування ефективності методів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кількість невідомих | Параметри тестування | Метод | | |
| Якобі | Гауса-Зейделя | Градієнтного спуску |
| 1000 | Кількість ітерацій | 7 | 5 | 3 |
| Кількість елементарних операцій (млн.) | 7,68 | 4,80 | 7,20 |
| Час виконання методу (мс) | 16 | 10 | 15 |
| 2500 | Кількість ітерацій | 7 | 5 | 3 |
| Кількість елементарних операцій (млн.) | 37,44 | 14,88 | 29,76 |
| Час виконання методу (мс) | 78 | 31 | 62 |
| 5000 | Кількість ітерацій | 8 | 6 | 3 |
| Кількість елементарних операцій (млн.) | 134,88 | 89,76 | 89,76 |
| Час виконання методу (мс) | 281 | 187 | 187 |
| 10000 | Кількість ітерацій | 7 | 6 | 3 |
| Кількість елементарних операцій (млн.) | 449,28 | 367,20 | 366,72 |
| Час виконання методу (мс) | 936 | 765 | 764 |
| 15000 | Кількість ітерацій | 7 | 5 | 3 |
| Кількість елементарних операцій (млн.) | 995,52 | 681,60 | 831,36 |
| Час виконання методу (мс) | 2074 | 1420 | 1732 |

Візуалізація результатів табилиці 7.1 наведено на рисунку 7.1.

Рисунок 7.1 – Графік залежності часу виконання методу від розміру   
вхідної системи

За результатами тестування можна зробити наступні висновки:

а) всі розглянуті методи дозволяю знаходити розвязки великих та надвеликих СЛАР;

б) cкладність всіх розглянутих методів є квадратичною, тобто –  
 , де – кількість ітерацій виконаних методом, – розмір СЛАР;

в) з розглянутих методів найоптимальнішим для практичного використання є метод Гауса-Зейделя, оскільки він виконується найшвидше та має такі умови сходимості, що охоплюють найширший спектр СЛАР;

Висновки

На етапі ознайомлення з теоретичними аспектами завдання даної курсової роботи було досліджено алгоритми ітераційних методів розв’язання СЛАР, що використовуються для знаходження розв’язків великих та надвеликих СЛАР, проаналізовано їх особливості та доцільність використання в межах реальних проектів, що було доведено реалізацією програмного забезпечення, яке розглядається в данні курсовій роботі.

На етапі проектування програмного забезпечення реалізовано наступні ітераційні методи: Якобі, Гауса-Зейделя, градієнтного спуску. Результатом розробки програми є набуття досвіду роботи з великими та надвеликими СЛАР, закріплення, узагальнення та поглиблення знань і навичок програмування на мовах С\С++, набуття практичних навичок алгоритмізації прикладних задач з лінійної алгебри.

Для спрощення процесу взаємодії з користувачем та розширення кола можливих користувачів програмного забезпечення був розроблений графічний інтерфейс. Також була розроблена інструкція користувача, що дозволяє ознайомитися зі всіма можливостями та ключовим моментами використання програмного забезпечення.

На етапі тестування було визначено, що програма вірно обробляє всі вхідні дані та видає очікуваний результат.

Результатом курсової роботи є створена програма, яку можна використовувати у навчальних та/або інженерних цілях з метою розв’язання задач, які зводяться до знаходження рішення СЛАР або містять процес знаходження рішення СЛАР як підзадачу.

Перелік посилань

1. Иванов А.П. Практикум по численным методам решение систем линейных алгебраических уравнений (Методические указания). – С.-Пб., 2013 – 19 с.
2. В. М. Вержбицкий. Численные методы. Линейная алгебра и нелинейные уравнения. – М.: Изд. Высшая школа, 2000. – 274 с.
3. Баркалов К.А. Методы параллельных вычислений. – Н. Новгород: Изд-во Нижегородского госуниверситета им. Н.И. Лобачевского, 2011. – 124 с.
4. Н. Бахвалов, Н. Жидков, Г. Кобельков. Численные методы. – М.: Изд. Физматлит, 2006. – 432 с.
5. Фельдман Л.П., Петренко А.І., Дмитрієва О.А. Чисельні методи в інформатиці, BHV, 2006 – 480 с.
6. Калиткин Н.Н. Численные методы. – М.: Наука, 1978. – 512 с.
7. Страуструп. Б. Программирование: принципы и практика использования С++.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2011. – 1248 с.
8. Введение в вычислительную математику: для студентов задачи с решениями [Електронний ресурс] // Exponenta.ru – Режим доступу:

<http://www.exponenta.ru/educat/class/courses/student/vvm/examples.asp>

1. Особенности и преимущества итерационных методов решения систем линейных алгебраических уравнений [Електронний ресурс] //   
   studopedia.ru – Режим доступу:

<http://studopedia.ru/3_53319_osobennosti-i-preimushchestva-iteratsionnih-metodov-resheniya-sistem-lineynih-algebraicheskih-uravneniy.html>

1. Классификация итерационных методов. Иследование сходимости стационарных итерационных методов. Метод Якоби [Електронний ресурс] // old.math.tsu.ru – Режим доступу:

<http://old.math.tsu.ru/EEResources/cm/text/5_2.htm>

**Додаток А Технічне завдання**

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Муха І. П.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016р.

Виконавець:

Студент Зарічковий О. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: «Розв'язання СЛАР

наближеними методами»

з дисципліни:

«Основи програмування»

Київ 2016

1.1 Мета: Метою курсової роботи є розробка комплексу програм для розв’язання систем лінійних алгебраїчних рівнянь.

1.2 Найменування та галузь застосування об'єкта розробки:Дана робота присвячена розробці програмного забезпечення для розв’язання систем лінійних алгебраїчних рівнянь методами простої ітерації, Гаусса-Зейделя та методом найшвидшого спуску.

1.3 Підстава для проведення робіт:Підставою для розробки програмного забезпечення є навчальний план спеціальності 6.050301 «Програмна інженерія», робоча програма дисципліни „Основи програмування”, індивідуальне завдання.

1.4 Дата початку роботи: «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2016 р.

1.5 Дата закінчення роботи: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

1.6 Призначення розробки: Функціональним призначенням програми є автоматизація розв’язання СЛАР, для підвищення точності та швидкості обчислень. Програма може експлуатуватися користувачем, який стикається з науковими та інженерними розрахунками для обчислення коренів розв’язання СЛАР, які могли б з високою точністю збігатися з реальними значеннями.

1.7 Вимоги до програми та програмної документації: Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ИСО 5807 - 85 ГОСТ на розробку програмних документів, схем алгоритмів програм, даних та систем.

ГОСТ 19.781 - 74 - Вимоги до розробки програмного забезпечення.

ГОСТ 19.101-77 (СТ СЭВ 1626 - 79) - Держстандарт на розробку програмної документації, видів програм та програмних документів.

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.

1.8 Стадії та етапи розробки:

1. Аналіз методів вирішення поставленої задачі (до 17.04.2016 р.)

2. Розробка алгоритмічного забезпечення (до 24.04.2016 р.)

3. Розробка сценарію роботи програми (до 01.05.2016 р.)

4. Розробка програмного забезпечення (до 22.05.2016 р.)

5. Розробка інтерфейсу, планування, тестування розробленої програми (до 05.06.2016 р.)

6. Розробка пояснювальної записки (до 12.06.2016 р.).

7. Захист курсової роботи (до \_\_.\_\_.2016 р.).

1.9 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 її оцінювання.

Додаток Б Тексти програмного коду

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення*

Розв’язання СЛАР наближеними методами

(Вид носія даних)

*CD-RW*

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

*арк, Кб*

*студента групи ІП-51 І курсу*

Зарічкового Олександра Анатолійовича

Файл «Utilities.h»

#pragma once

namespace utilities {

const double eps = 1e-5; // Точність обчислення розвяку

// Структура, що представляє систему

struct system {

int size; // Розмір системи

double\*\* matrix; // Матриця системи

double\* free; // Стовпець вільних членів

};

double det(double\*\*, const int&); // Прототип функції обчислення визначника

double\*\* matrixCut(double\*\*, const int&, const int&); // Прототип функції уворення матриці(N - 1) - го порядку

double AlgebralAdditionCalc(double\*\*, const int&, const int&); // Прототип функції обчислення алгебраїного доповнення заданого елементу

double normCalc(double\*, double\*, const int&); // Прототип функції обчислення норми поточного рішення

bool isSemetric(const system&); // Прототип функції превірки матриці системи на симетричність

bool isPositive(const system&); // Прототип функції превірки матриці системи на додатньовизначеність

double\*\* cornerMinor(const system&, int); // Прототип функції утворення кутового мінору

double\* matrixMult(const system&, double\*); // Прототип функції множення матриці системи на вектор

double scalarMult(double\*, double\*, int&); // Прототип функції скалярного добутку векторів

void stableSystem(system&); // Прототип функції приведння матриці системи до канонічного вигляду

int indexOfMaxElement(double\*, const int&); // Прототип функції знаходження індекса максимального елементу в рядку

double lineSumElement(double\*, const int&); // Прототип функції знаходження суми елементів в рядку

bool isDiagDominate(const system&); // Прототип функції перевірки матриці системи на діагональний перевіс

double\*\* CreateMatrix(int); // Прототип функції виділення памяті під матрицю

void freeMatrix(double\*\* &, int); // Прототип функції видалення матриці з памяті

double\* CreateMas(int); // Прототип функції виділення памяті під масив

void freeMas(double\* &); // Прототип функції видалення масиву з памяті

void masCopy(double\*, double\*, int&); // Прототип функції копіювання масиву

};

Файл «Utilities.сpp»

#include "Utilities.h"

#include <cmath>

using namespace utilities;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обчислення визначника \*

\* матриці \*

\* Параметри: \*

\* matrix - матриця для якої потірбно \*

\* порахувати визначник \*

\* size - розмірність матриці \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double utilities::det(double\*\* matrix, const int& size) {

//ЯКЩО розмірність матриці одинична, ТО повернути визначник матриці 1х1

if (size == 1) return \*\*matrix;

else {

// Обчислення визначника матриці розкладом за першим стовпцем

double detRes = 0; // Визначник поточної матриці

for (int i = 0; i < size; i++) detRes += AlgebralAdditionCalc(matrix, size, i) \* matrix[i][0];

return detRes;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція уворення матриці \*

\* (N-1)-го порядку \*

\* Параметри: \*

\* matrix - матриця для якої потірбно \*

\* знайт мінор \*

\* size - розмірність матриці \*

\* y - номер рядка елемента, для \*

\* якого потрібно порахувати \*

\* мінор \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double\*\* utilities::matrixCut(double\*\* matrix, const int& size, const int& y) {

int elementCount = 0; // Кількість елементів в утвореній матриці

// Cтворюємо нову матрицю

double\*\* newMatrix = CreateMatrix(size - 1);

// Заповнюємо матрицю

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

for (int z = 1; z < size; z++)

if (i != y) {

newMatrix[elementCount / (size - 1)][elementCount % (size - 1)] = matrix[i][z];

elementCount++;

}

return newMatrix;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обчислення алгебраїчного \*

\* доповнення заданого елементу \*

\* Параметри: \*

\* matrix - матриця для якої потірбно \*

\* знайти алгебраїчне доповнення \*

\* size - розмірність матриці \*

\* y - номер рядка елемента, для \*

\* якого потрібно порахувати \*

\* алгебраїний мінор \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double utilities::AlgebralAdditionCalc(double\*\* matrix, const int& size, const int& y) {

double\*\* newMatrix = matrixCut(matrix, size, y);

double result = det(newMatrix, size-1) \* (y & 1 ? -1 : 1); // Алгебраїчне доповнення

freeMatrix(newMatrix, size - 1);

return result;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обчислення норми \*

\* поточного наближення \*

\* Параметри: \*

\* х - поперднє наближення \*

\* хk - поточне наближення \*

\* size - розмірність масивів \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double utilities::normCalc(double\* x, double\* xk, const int& size) {

double norm = abs(\*x - \*xk);

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

if (abs(x[i] - xk[i]) > norm) norm = abs(x[i] - xk[i]);

return norm;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція перевірки матриці \*

\* системи на симетричність \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool utilities::isSemetric(const system& sols) {

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

for (int z = i; z < sols.size; z++)

if (sols.matrix[i][z] != sols.matrix[z][i]) return false;

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція перевірки матриці \*

\* системи на додатньовизначеність \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool utilities::isPositive(const system& sols) {

// Користуємося критерієм Сильвестра

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

double\*\* newMatrix = cornerMinor(sols, i);

// Якщо визначник кутового мінору не є додатньою, то матриця не є додатньовизначеною

if (det(newMatrix, i) <= 0) return false;

freeMatrix(newMatrix, i);

}

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція утворення кутового \*

\* мінору матриці \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\* size - розмірність кутового мінору \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double\*\* utilities::cornerMinor(const system& sols, int size) {

double\*\* newMatrix = CreateMatrix(size);

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

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

newMatrix[i][z] = sols.matrix[i][z];

return newMatrix;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція множення матриці на \*

\* вектор \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\* mas - вектор, на який множиться СЛАР \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double\* utilities::matrixMult(const system& sols, double\* mas) {

double\* newVector = CreateMas(sols.size);

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

newVector[i] = 0;

for (int z = 0; z < sols.size; z++) newVector[i] += mas[z] \* sols.matrix[i][z];

}

return newVector;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція знаходження скалярного \*

\* добутку векторів \*

\* Параметри: \*

\* vector1 - перший векртор \*

\* vector2 - другий вектор \*

\* size - розмірність векторі \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double utilities::scalarMult(double\* vector1, double\* vector2, int& size) {

double scalar = 0;

for (int i = 0; i < size; i++) scalar += vector1[i] \* vector2[i];

return scalar;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція стабілізації матриці \*

\* відносно головної діагоналі \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void utilities::stableSystem(system& sols) {

// Масив який вказує на якому рядку було замічено діагональний перевіс

int\* mas = new int[sols.size];

for (int i = 0; i < sols.size; i++) mas[i] = -1;

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

int index = indexOfMaxElement(sols.matrix[i], sols.size);

double sum = lineSumElement(sols.matrix[i], sols.size);

if (sum >= abs(2 \* sols.matrix[i][index])) {

delete[] mas;

return;

}

if(mas[index] != -1){

delete[] mas;

return;

}

mas[index] = i;

}

system newSols;

newSols.size = sols.size;

newSols.matrix = CreateMatrix(sols.size);

newSols.free = CreateMas(sols.size);

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

newSols.free[i] = sols.free[mas[i]];

for (int z = 0; z < sols.size; z++) newSols.matrix[i][z] = sols.matrix[mas[i]][z];

}

freeMatrix(sols.matrix, sols.size);

freeMas(sols.free);

sols = newSols;

delete[] mas;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція знаходження \*

\* максимального за абсолютним \*

\* значення елемента в поточному \*

\* рядку \*

\* Параметри: \*

\* mas - поточний рядок \*

\* size - кількість елементів в рядку \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int utilities::indexOfMaxElement(double\* mas, const int& size) {

int index = 0; // Індекс максимального елементу

for (int i = 1; i < size; i++) if (abs(mas[i]) > abs(mas[index])) index = i;

return index;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція знаходження \*

\* суми абсолютних значеннь \*

\* елементів в поточному \*

\* рядку \*

\* Параметри: \*

\* mas - поточний рядок \*

\* size - кількість елементів в рядку \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double utilities::lineSumElement(double\* mas, const int& size) {

double sum = 0; // Сума елементів

for (int i = 0; i < size; i++) sum += abs(mas[i]);

return sum;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція перевірки матриці системи \*

\* на діагональний перевіс \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool utilities::isDiagDominate(const system& sols) {

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

double sum = lineSumElement(sols.matrix[i], sols.size);

if (sum >= abs(2 \* sols.matrix[i][i])) return false;

}

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція виділення динамічної \*

\* памяті під матрицю \*

\* Параметри: \*

\* size - розмірність матриці \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double\*\* utilities::CreateMatrix(int size) {

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

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

matrix[i] = new double[size];

return matrix;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція видалення матриці \*

\* Параметри: \*

\* matrix - матриця, яку необхідно \*

\* видалити \*

\* size - розмірність матриці \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void utilities::freeMatrix(double\*\* & matrix, int size) {

for (int i = 0; i < size; i++) delete[] matrix[i];

delete[] matrix;

matrix = nullptr;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція виділення динамічної \*

\* памяті під масив \*

\* Параметри: \*

\* size - розмірність масиву \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double\* utilities::CreateMas(int size) {

double\* mas = new double [size];

return mas;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція видалення масиву \*

\* Параметри: \*

\* mas - масив, який необхідно \*

\* видалити \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void utilities::freeMas(double\* & mas) {

delete[] mas;

mas = nullptr;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція копіювання масиву \*

\* Параметри: \*

\* scr - масив, який потрібно копіювати \*

\* dist - масив, в який потірбно \*

\* копіювати \*

\* size - розмірність масиву \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void utilities::masCopy(double\* src, double\* dist, int& size) {

for (int i = 0; i < size; i++) dist[i] = src[i];

}

Файл «Jacobi.h»

#pragma once

#include "Utilities.h"

using utilities::system;

namespace Jacobi {

double\* Jacobi(utilities::system&); // Прототип функції рішення системи методом Якобі

bool isSolved(utilities::system&); // Прототип функції перевірки на сходимість метода Якобі

void nextSolution(utilities::system&, double\*, double \*); // Прототип функції знаходження наступного наближення

};

Файл «Jacobi.cpp»

#include "Jacobi.h"

using namespace Jacobi;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція рішення СЛАР методом \*

\* Якобі \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double\* Jacobi::Jacobi(system& sols) {

double\* x = utilities::CreateMas(sols.size);

double\* xk = utilities::CreateMas(sols.size);

for (int i = 0; i < sols.size; i++) xk[i] = 0;

double norm = 2 \* utilities::eps;

while (norm >= utilities::eps) {

utilities::masCopy(xk, x, sols.size);

nextSolution(sols, x, xk);

norm = utilities::normCalc(x, xk, sols.size);

}

utilities::freeMas(x);

return xk;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція перевірки на сходимість \*

\* методу Якобі \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool Jacobi::isSolved(system& sols) {

utilities::stableSystem(sols);

if (utilities::isDiagDominate(sols)) return true;

else return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція знаходження наступного \*

\* наближення для методу Якобі \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\* x - поперднє наближення \*

\* xk - поточне наближення \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Jacobi::nextSolution(system& sols, double \*x, double\* xk) {

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

xk[i] = sols.free[i];

for (int z = 0; z < sols.size; z++)

if (i != z) xk[i] -= sols.matrix[i][z] \* x[z];

xk[i] /= sols.matrix[i][i];

}

}

Файл «Gauss-Seidel.h»

#pragma once

#include "Utilities.h"

namespace Seidel {

double\* Seidel(utilities::system&); // Прототип функції рішення системи методом Гауса-Зейделя

bool isSolved(utilities::system&); // Прототип функції перевірки на сходимість метода Гауса-Зейделя

void nextSolution(utilities::system&, double\*, double \*); // Прототип функції знаходження наступного наближення

};

Файл «Gauss-Seidel.cpp»

#include "Gauss–Seidel.h"

using namespace Seidel;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція рішення СЛАР методом \*

\* Гауса-Зейделя \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double\* Seidel::Seidel(utilities::system& sols) {

double\* x = utilities::CreateMas(sols.size);

double\* xk = utilities::CreateMas(sols.size);

for (int i = 0; i < sols.size; i++) xk[i] = 0;

double norm = 2 \* utilities::eps;

while (norm >= utilities::eps) {

utilities::masCopy(xk, x, sols.size);

nextSolution(sols, x, xk);

norm = utilities::normCalc(x, xk, sols.size);

}

delete[] x;

return xk;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція перевірки на сходимість \*

\* методу Гауса-Зейделя \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool Seidel::isSolved(utilities::system& sols) {

utilities::stableSystem(sols);

if (utilities::isDiagDominate(sols) || (utilities::isPositive(sols) && utilities::isSemetric(sols))) return true;

else return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція знаходження наступного \*

\* наближення для методу \*

\* Гауса-Зейделя \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\* x - поперднє наближення \*

\* xk - поточне наближення \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Seidel::nextSolution(utilities::system& sols, double\* x, double \* xk) {

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

xk[i] = sols.free[i];

for (int z = 0; z < i; z++) xk[i] -= sols.matrix[i][z] \* xk[z];

for (int z = i + 1; z < sols.size; z++) xk[i] -= sols.matrix[i][z] \* xk[z];

xk[i] /= sols.matrix[i][i];

}

}

Файл «GradientDescent.h»

#pragma once

#include "Utilities.h"

namespace GradientDescent {

bool isSolved(utilities::system&); // Прототип функції перевірки на сходимість метода градієнтоного спуску

double\* GradientDescent(utilities::system&); // Прототип функції рішення системи методом градієнтоного спуску

};

Файл «GradientDescent.cpp»

#include "GradientDescent.h"

using utilities::scalarMult;

using utilities::matrixMult;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція рішення СЛАР методом \*

\* градієнтного спуску \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double\* GradientDescent::GradientDescent(utilities::system& sols) {

double\* x = utilities::CreateMas(sols.size); // Поточне рішення системи

double\* xk = utilities::CreateMas(sols.size); // Наступне наближення рішення

double\* r = utilities::CreateMas(sols.size); // Поточне значення незвязка

double\* rk = utilities::CreateMas(sols.size); // Наступне значення незвязки

double\* z = utilities::CreateMas(sols.size); // Поточне значення вектору напряку

double\* zk = utilities::CreateMas(sols.size);// Наступне значення вектору напрямку

// Задання початкових уточнень

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

xk[i] = 0;

rk[i] = sols.free[i];

zk[i] = sols.free[i];

}

double norm = 2 \* utilities::eps;

while (norm >= utilities::eps) {

utilities::masCopy(xk, x, sols.size);

utilities::masCopy(rk, r, sols.size);

utilities::masCopy(zk, z, sols.size);

double\* Az = matrixMult(sols, z);

int j;

for (j = 0; j < sols.size; j++) if (Az[j]) break;

if (j == sols.size) break;

double alpha = scalarMult(r, r, sols.size) / scalarMult(Az, z, sols.size) ; // Скалярний крок градієнту

for (int i = 0; i < sols.size; i++) xk[i] += alpha \* z[i];

for (int i = 0; i < sols.size; i++) rk[i] -= alpha \* Az[i];

double beta = scalarMult(rk, rk, sols.size) / scalarMult(r, r, sols.size); // Скалярна корекція напрямку

utilities::masCopy(rk, zk, sols.size);

for (int i = 0; i < sols.size; i++) zk[i] += beta \* z[i];

utilities::freeMas(Az);

norm = utilities::normCalc(x, xk, sols.size);

}

utilities::freeMas(x);

utilities::freeMas(r);

utilities::freeMas(rk);

utilities::freeMas(z);

utilities::freeMas(zk);

return xk;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція перевірки на сходимість \*

\* методу градієнтного спуску \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool GradientDescent::isSolved(utilities::system& sols) {

return utilities::isPositive(sols) && utilities::isSemetric(sols);

}

Файл «Graph.h»

#pragma once

#include "Utilities.h"

#include <algorithm>

#include <math.h>

namespace CourseWork {

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>

/// Summary for Graph

/// </summary>

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

{

public:

Graph(utilities::system& sols, double\* x)

{

InitializeComponent(sols, x);

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~Graph()

{

if (components)

{

delete components;

}

}

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

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(utilities::system& sols, double\* x)

{

System::Windows::Forms::DataVisualization::Charting::ChartArea^ chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());

System::Windows::Forms::DataVisualization::Charting::Legend^ legend1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Legend());

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());

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

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

this->SuspendLayout();

//

// CGraph

//

chartArea1->Name = L"ChartArea1";

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

legend1->Name = L"Legend1";

this->CGraph->Legends->Add(legend1);

this->CGraph->Location = System::Drawing::Point(0, 0);

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

series1->ChartArea = L"ChartArea1";

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

series1->Legend = L"Legend1";

series1->LegendText = L"Рівняння-1";

series1->LegendText = sols.matrix[0][0] + " \* x1 " + (sols.matrix[0][1] >= 0 ? "+ " : "- ") + abs(sols.matrix[0][1]) + " \* x2 " + (sols.free[0] >= 0 ? "- " : "+ ") + abs(sols.free[0]) + " = 0";

series1->Name = L"Series1";

series2->ChartArea = L"ChartArea1";

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

series2->Legend = L"Legend1";

series2->LegendText = L"Рівняння-2";

series2->LegendText = sols.matrix[1][0] + " \* x1 " + (sols.matrix[1][1] >= 0 ? "+ " : "- ") + abs(sols.matrix[1][1]) + " \* x2 " + (sols.free[1] >= 0 ? "- " : "+ ") + abs(sols.free[1]) + " = 0";

series2->Name = L"Series2";

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

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

this->CGraph->Size = System::Drawing::Size(800, 600);

this->CGraph->TabIndex = 0;

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

//

// Graph

//

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

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

this->ClientSize = System::Drawing::Size(800, 600);

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

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

this->Name = L"Graph";

this->Text = L"Графік";

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

this->ResumeLayout(false);

this->PerformLayout();

this->MaximizeBox = false;

double xl = round(x[0] - 10);

double xr = round(x[0] + 10);

double yl = round(x[1] - 10);

double yr = round(x[1] + 10);

if (sols.matrix[0][1] != 0) {

yl = std::min(yl, (sols.free[0] - sols.matrix[0][0] \* xl) / sols.matrix[0][1]);

yr = std::max(yr, (sols.free[0] - sols.matrix[0][0] \* xr) / sols.matrix[0][1]);

}

if (sols.matrix[1][1] != 0) {

yl = std::min(yl, (sols.free[1] - sols.matrix[1][0] \* xl) / sols.matrix[1][1]);

yr = std::max(yr, (sols.free[1] - sols.matrix[1][0] \* xr) / sols.matrix[1][1]);

}

chartArea1->AxisX->ScaleView->Zoom(xl, xr);

chartArea1->CursorX->IsUserEnabled = true;

chartArea1->CursorX->IsUserSelectionEnabled = true;

chartArea1->AxisX->ScaleView->Zoomable = true;

chartArea1->AxisX->Title = "X1";

chartArea1->AxisY->ScaleView->Zoom(yl, yr);

chartArea1->CursorY->IsUserEnabled = true;

chartArea1->CursorY->IsUserSelectionEnabled = true;

chartArea1->AxisY->ScaleView->Zoomable = true;

chartArea1->AxisY->Title = "X2";

if(sols.matrix[0][1] != 0)

for (double i = xl; i <= xr; i += 0.1) series1->Points->AddXY(i, (sols.free[0] - sols.matrix[0][0] \* i) / sols.matrix[0][1]);

else for (double i = yl; i <= yr; i += 0.1) series1->Points->AddXY(sols.free[0] / sols.matrix[0][0], i);

if (sols.matrix[1][1] != 0)

for (double i = xl; i <= xr; i += 0.1) series2->Points->AddXY(i, (sols.free[1] - sols.matrix[1][0] \* i) / sols.matrix[1][1]);

else for (double i = yl; i <= yr; i += 0.1) series2->Points->AddXY(sols.free[1] / sols.matrix[1][0], i);

}

#pragma endregion

};

}

Файл «Home.h»

#pragma once

#include <cmath>

#include "Jacobi.h"

#include "Gauss–Seidel.h"

#include "GradientDescent.h"

#include "Utilities.h"

#include "Graph.h"

#include <time.h>

#define SIZE 300

#define MSIZE 5

#define presision 1000

namespace CourseWork {

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>

/// Summary for Home

/// </summary>

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

{

public:

Home(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~Home()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::NumericUpDown^ solsSize;

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

private: System::Windows::Forms::DataGridView^ dataFree;

private: System::Windows::Forms::DataGridView^ dataRes;

private: System::Windows::Forms::Label^ label2;

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

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::GroupBox^ groupBox1;

private: System::Windows::Forms::RadioButton^ rDesent;

private: System::Windows::Forms::RadioButton^ RSeidel;

private: System::Windows::Forms::RadioButton^ RJacobi;

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

private: System::Windows::Forms::Label^ label5;

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

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

private: System::Windows::Forms::DataGridView^ dataMatrix;

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </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::DataGridViewCellStyle^ dataGridViewCellStyle3 = (gcnew System::Windows::Forms::DataGridViewCellStyle());

System::Windows::Forms::DataGridViewCellStyle^ dataGridViewCellStyle4 = (gcnew System::Windows::Forms::DataGridViewCellStyle());

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

this->solsSize = (gcnew System::Windows::Forms::NumericUpDown());

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

this->dataMatrix = (gcnew System::Windows::Forms::DataGridView());

this->dataFree = (gcnew System::Windows::Forms::DataGridView());

this->dataRes = (gcnew System::Windows::Forms::DataGridView());

this->label2 = (gcnew System::Windows::Forms::Label());

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

this->label4 = (gcnew System::Windows::Forms::Label());

this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());

this->rDesent = (gcnew System::Windows::Forms::RadioButton());

this->RSeidel = (gcnew System::Windows::Forms::RadioButton());

this->RJacobi = (gcnew System::Windows::Forms::RadioButton());

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

this->label5 = (gcnew System::Windows::Forms::Label());

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

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

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

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

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

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

this->groupBox1->SuspendLayout();

this->SuspendLayout();

//

// solsSize

//

this->solsSize->InterceptArrowKeys = false;

this->solsSize->Location = System::Drawing::Point(19, 30);

this->solsSize->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 8, 0, 0, 0 });

this->solsSize->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 2, 0, 0, 0 });

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

this->solsSize->Size = System::Drawing::Size(120, 20);

this->solsSize->TabIndex = 0;

this->solsSize->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 2, 0, 0, 0 });

this->solsSize->ValueChanged += gcnew System::EventHandler(this, &Home::solsSize\_ValueChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label1->Location = System::Drawing::Point(16, 9);

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

this->label1->Size = System::Drawing::Size(119, 18);

this->label1->TabIndex = 1;

this->label1->Text = L"Розмір системи";

//

// dataMatrix

//

this->dataMatrix->AllowUserToAddRows = false;

this->dataMatrix->AllowUserToDeleteRows = false;

this->dataMatrix->AllowUserToResizeColumns = false;

this->dataMatrix->AllowUserToResizeRows = false;

this->dataMatrix->BackgroundColor = System::Drawing::Color::WhiteSmoke;

this->dataMatrix->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

dataGridViewCellStyle1->Alignment = System::Windows::Forms::DataGridViewContentAlignment::MiddleLeft;

dataGridViewCellStyle1->BackColor = System::Drawing::SystemColors::Window;

dataGridViewCellStyle1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

dataGridViewCellStyle1->ForeColor = System::Drawing::SystemColors::ControlText;

dataGridViewCellStyle1->SelectionBackColor = System::Drawing::SystemColors::Highlight;

dataGridViewCellStyle1->SelectionForeColor = System::Drawing::Color::Black;

dataGridViewCellStyle1->WrapMode = System::Windows::Forms::DataGridViewTriState::False;

this->dataMatrix->DefaultCellStyle = dataGridViewCellStyle1;

this->dataMatrix->Location = System::Drawing::Point(12, 109);

this->dataMatrix->MultiSelect = false;

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

dataGridViewCellStyle2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

dataGridViewCellStyle2->ForeColor = System::Drawing::Color::Black;

dataGridViewCellStyle2->SelectionBackColor = System::Drawing::Color::DodgerBlue;

dataGridViewCellStyle2->SelectionForeColor = System::Drawing::Color::Black;

this->dataMatrix->RowsDefaultCellStyle = dataGridViewCellStyle2;

this->dataMatrix->Size = System::Drawing::Size(270, 204);

this->dataMatrix->TabIndex = 2;

this->dataMatrix->CellEndEdit += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &Home::dataMatrix\_CellEndEdit);

//

// dataFree

//

this->dataFree->AllowUserToAddRows = false;

this->dataFree->AllowUserToDeleteRows = false;

this->dataFree->AllowUserToResizeColumns = false;

this->dataFree->AllowUserToResizeRows = false;

this->dataFree->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom)

| System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

this->dataFree->BackgroundColor = System::Drawing::Color::WhiteSmoke;

this->dataFree->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataFree->ColumnHeadersVisible = false;

this->dataFree->Location = System::Drawing::Point(288, 110);

this->dataFree->MultiSelect = false;

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

dataGridViewCellStyle3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

dataGridViewCellStyle3->ForeColor = System::Drawing::Color::Black;

dataGridViewCellStyle3->SelectionBackColor = System::Drawing::Color::DodgerBlue;

dataGridViewCellStyle3->SelectionForeColor = System::Drawing::Color::Black;

this->dataFree->RowsDefaultCellStyle = dataGridViewCellStyle3;

this->dataFree->Size = System::Drawing::Size(113, 205);

this->dataFree->TabIndex = 3;

this->dataFree->CellEndEdit += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &Home::dataFree\_CellEndEdit);

//

// dataRes

//

this->dataRes->AllowUserToAddRows = false;

this->dataRes->AllowUserToDeleteRows = false;

this->dataRes->AllowUserToResizeColumns = false;

this->dataRes->AllowUserToResizeRows = false;

this->dataRes->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom)

| System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

this->dataRes->BackgroundColor = System::Drawing::Color::WhiteSmoke;

this->dataRes->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataRes->ColumnHeadersVisible = false;

this->dataRes->Location = System::Drawing::Point(407, 111);

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

this->dataRes->ReadOnly = true;

dataGridViewCellStyle4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

dataGridViewCellStyle4->ForeColor = System::Drawing::Color::Black;

dataGridViewCellStyle4->SelectionBackColor = System::Drawing::Color::DodgerBlue;

dataGridViewCellStyle4->SelectionForeColor = System::Drawing::Color::Black;

this->dataRes->RowsDefaultCellStyle = dataGridViewCellStyle4;

this->dataRes->Size = System::Drawing::Size(107, 204);

this->dataRes->TabIndex = 4;

//

// label2

//

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label2->Location = System::Drawing::Point(15, 81);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(267, 25);

this->label2->TabIndex = 5;

this->label2->Text = L"Матриця системи";

this->label2->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// label3

//

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label3->Location = System::Drawing::Point(285, 50);

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

this->label3->Size = System::Drawing::Size(116, 57);

this->label3->TabIndex = 6;

this->label3->Text = L"Стовпець вільних членів";

this->label3->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// label4

//

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label4->Location = System::Drawing::Point(407, 68);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(107, 38);

this->label4->TabIndex = 7;

this->label4->Text = L"Рішення системи";

this->label4->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// groupBox1

//

this->groupBox1->Controls->Add(this->rDesent);

this->groupBox1->Controls->Add(this->RSeidel);

this->groupBox1->Controls->Add(this->RJacobi);

this->groupBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->groupBox1->Location = System::Drawing::Point(520, 100);

this->groupBox1->Name = L"groupBox1";

this->groupBox1->Size = System::Drawing::Size(185, 109);

this->groupBox1->TabIndex = 8;

this->groupBox1->TabStop = false;

this->groupBox1->Text = L"Метод рішення";

//

// rDesent

//

this->rDesent->AutoSize = true;

this->rDesent->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->rDesent->Location = System::Drawing::Point(16, 77);

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

this->rDesent->Size = System::Drawing::Size(168, 22);

this->rDesent->TabIndex = 2;

this->rDesent->TabStop = true;

this->rDesent->Text = L"Градієнтного спуску";

this->rDesent->UseVisualStyleBackColor = true;

//

// RSeidel

//

this->RSeidel->AutoSize = true;

this->RSeidel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->RSeidel->Location = System::Drawing::Point(16, 49);

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

this->RSeidel->Size = System::Drawing::Size(131, 22);

this->RSeidel->TabIndex = 1;

this->RSeidel->TabStop = true;

this->RSeidel->Text = L"Гауса-Зейделя";

this->RSeidel->UseVisualStyleBackColor = true;

//

// RJacobi

//

this->RJacobi->AutoSize = true;

this->RJacobi->Checked = true;

this->RJacobi->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->RJacobi->Location = System::Drawing::Point(16, 21);

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

this->RJacobi->Size = System::Drawing::Size(66, 22);

this->RJacobi->TabIndex = 0;

this->RJacobi->TabStop = true;

this->RJacobi->Text = L"Якобі";

this->RJacobi->UseVisualStyleBackColor = true;

//

// Info

//

this->Info->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->Info->Location = System::Drawing::Point(12, 345);

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

this->Info->ReadOnly = true;

this->Info->Size = System::Drawing::Size(693, 24);

this->Info->TabIndex = 9;

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label5->Location = System::Drawing::Point(9, 324);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(160, 18);

this->label5->TabIndex = 10;

this->label5->Text = L"Системна інформація";

//

// BSolve

//

this->BSolve->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->BSolve->Location = System::Drawing::Point(520, 235);

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

this->BSolve->Size = System::Drawing::Size(185, 35);

this->BSolve->TabIndex = 11;

this->BSolve->Text = L"Знайти рішення";

this->BSolve->UseVisualStyleBackColor = true;

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

//

// BClear

//

this->BClear->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->BClear->Location = System::Drawing::Point(520, 276);

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

this->BClear->Size = System::Drawing::Size(185, 35);

this->BClear->TabIndex = 12;

this->BClear->Text = L"Очистити вміст";

this->BClear->UseVisualStyleBackColor = true;

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

//

// Home

//

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

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

this->ClientSize = System::Drawing::Size(714, 376);

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

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

this->Controls->Add(this->label5);

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

this->Controls->Add(this->groupBox1);

this->Controls->Add(this->label4);

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

this->Controls->Add(this->label2);

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

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

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

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

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

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

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

this->MaximizeBox = false;

this->Name = L"Home";

this->Text = L"СЛАР";

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

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

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

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

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

this->groupBox1->ResumeLayout(false);

this->groupBox1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція виведення результату \*

\* роботи програми у поле результату \*

\* Параметри: \*

\* х - результат виконання програми \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private: void print(double\* x) {

int size = Convert::ToInt32(solsSize->Value);

for (int i = 0; i < size; i++) dataRes->Rows[i]->Cells[0]->Value = Convert::ToString(round(x[i]\*presision)/ presision);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція читання СЛАР \*

\* Параметри: \*

\* sols - система лінійних алгебраїчних \*

\* рівнянь \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private: void DataRead(utilities::system& sols) {

bool flag = false;

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

for (int z = 0; z < sols.size; z++) sols.matrix[i][z] = Convert::ToDouble(dataMatrix->Rows[i]->Cells[z]->Value);

for (int i = 0; i < sols.size; i++) sols.free[i] = Convert::ToDouble(dataFree->Rows[i]->Cells[0]->Value);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція перевірки вхідних \*

\* даних на валідність \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private: bool DataRead() {

int size = Convert::ToInt32(solsSize->Value);

bool flag = false;

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

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

try {

double x = Convert::ToDouble(dataMatrix->Rows[i]->Cells[z]->Value);

x = round(x \* presision) / presision;

dataMatrix->Rows[i]->Cells[z]->Value = Convert::ToString(x);

dataMatrix->Rows[i]->Cells[z]->Style->BackColor = System::Drawing::Color::White;

}

catch (System::FormatException^ ex) {

flag = true;

dataMatrix->Rows[i]->Cells[z]->Style->BackColor = System::Drawing::Color::Red;

}

}

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

try {

double x = Convert::ToDouble(dataFree->Rows[i]->Cells[0]->Value);

x = round(x \* presision) / presision;

dataFree->Rows[i]->Cells[0]->Value = Convert::ToString(x);

dataFree->Rows[i]->Cells[0]->Style->BackColor = System::Drawing::Color::White;

}

catch (System::FormatException^ ex) {

flag = true;

dataFree->Rows[i]->Cells[0]->Style->BackColor = System::Drawing::Color::Red;

}

}

return flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція зміни розмірів матриці \*

\* системи,стовпця вільних членів \*

\* та вектора відповіді \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private: System::Void Resize() {

int size = Convert::ToInt32(solsSize->Value);

dataMatrix->ColumnCount = size;

dataMatrix->RowCount = size;

dataFree->ColumnCount = 1;

dataFree->RowCount = size;

dataRes->ColumnCount = 1;

dataRes->RowCount = size;

dataMatrix->AutoResizeColumns();

dataFree->AutoResizeColumns();

dataRes->AutoResizeColumns();

for (int i = 0; i < size; i++) dataMatrix->Columns[i]->HeaderText = "a" + (i + 1);

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

dataMatrix->Columns[i]->SortMode = DataGridViewColumnSortMode::NotSortable;

dataMatrix->Rows[i]->HeaderCell->Value = "a" + (i + 1);

dataFree->Rows[i]->HeaderCell->Value = "b" + (i + 1);

dataRes->Rows[i]->HeaderCell->Value = "x" + (i + 1);

dataMatrix->RowHeadersWidth = 48;

dataRes->RowHeadersWidth = 48;

dataFree->RowHeadersWidth = 48;

}

dataMatrix->RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode::DisableResizing;

dataFree->RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode::DisableResizing;

dataRes->RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode::DisableResizing;

dataMatrix->AutoResizeColumns();

dataFree->AutoResizeColumns();

dataRes->AutoResizeColumns();

/\*int t = SIZE / size;

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

dataFree->Rows[i]->Height = t;

dataRes->Rows[i]->Height = t;

dataMatrix->Columns[i]->Width = t;

dataMatrix->Rows[i]->Height = t;

}

dataFree->Columns[0]->Width = SIZE / MSIZE;

dataRes->Columns[0]->Width = 90;\*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція очищастки вмісту матриці \*

\* системи, стовпця вільних членів \*

\* та вектора відповіді \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private: System::Void Clear() {

int size = Convert::ToInt32(solsSize->Value);

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

for (int z = 0; z < size; z++) dataMatrix->Rows[i]->Cells[z]->Value = "0";

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

dataFree->Rows[i]->Cells[0]->Value = "0";

dataRes->Rows[i]->Cells[0]->Value = "";

}

Info->Text = "";

utilities::system sols;

sols.size = size;

sols.matrix = utilities::CreateMatrix(size);

sols.free = utilities::CreateMas(size);

DataRead();

BSolve->Enabled = true;

DataRead(sols);

utilities::freeMatrix(sols.matrix, size);

utilities::freeMas(sols.free);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обробкт події загрузки \*

\* вікна програми \*

\* Параметри: \*

\* Посилання на відправника, додаткові \*

\* аргументи виклику \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

Resize();

Clear();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обробки події змінення \*

\* розміру системи \*

\* Параметри: \*

\* Посилання на відправника, додаткові \*

\* аргументи виклику \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private: System::Void solsSize\_ValueChanged(System::Object^ sender, System::EventArgs^ e) {

int size = Convert::ToInt32(solsSize->Value);

int old\_size = dataFree->RowCount;

bool flag = size > dataMatrix->ColumnCount;

Resize();

if (flag) {

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

for (int z = old\_size; z < size; z++) dataMatrix->Rows[i]->Cells[z]->Value = "0";

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

for (int z = old\_size; z < size; z++) dataMatrix->Rows[z]->Cells[i]->Value = "0";

for (int z = old\_size; z < size; z++) dataFree->Rows[z]->Cells[0]->Value = "0";

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обробка події натискання \*

\* кнопки «Очистити вміст» \*

\* Параметри: \*

\* Посилання на відправника, додаткові \*

\* аргументи виклику \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

Clear();

Resize();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обробка події зміни вмісту \*

\* матриці системи \*

\* Параметри: \*

\* Посилання на відправника, додаткові \*

\* аргументи виклику \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

Resize();

int size = Convert::ToInt32(solsSize->Value);

Info->Text = "";

if (DataRead()) {

Info->Text = "Невірний фортмат вхідної інформації.";

BSolve->Enabled = false;

}

else BSolve->Enabled = true;

for (int i = 0; i < size; i++) dataRes->Rows[i]->Cells[0]->Value = "";

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обробка події зміни вмісту \*

\* стовпця вільних членів \*

\* Параметри: \*

\* Посилання на відправника, додаткові \*

\* аргументи виклику \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

Resize();

int size = Convert::ToInt32(solsSize->Value);

Info->Text = "";

if (DataRead()) {

Info->Text = "Невірний фортмат вхідної інформації.";

BSolve->Enabled = false;

}

else BSolve->Enabled = true;

for (int i = 0; i < size; i++) dataRes->Rows[i]->Cells[0]->Value = "";

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Функція обробки події натискання \*

\* кнопки «Знайти рішення» \*

\* Параметри: \*

\* Посилання на відправника, додаткові \*

\* аргументи виклику \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

Info->Text = "";

int size = Convert::ToInt32(solsSize->Value);

for (int i = 0; i < size; i++) dataRes->Rows[i]->Cells[0]->Value = "";

utilities::system sols;

sols.size = size;

sols.matrix = utilities::CreateMatrix(size);

sols.free = utilities::CreateMas(size);

DataRead(sols);

int start = clock();

if (utilities::det(sols.matrix, sols.size) == 0) {

Info->Text = "Система вироджена (визначник системи рівний нулю).";

utilities::freeMatrix(sols.matrix, size);

utilities::freeMas(sols.free);

return;

}

int oper;

double\* x;

if (RJacobi->Checked) {

if (!Jacobi::isSolved(sols)) {

Info->Text = "Для даної матриці метод Якобі не сходиться.";

utilities::freeMatrix(sols.matrix, size);

utilities::freeMas(sols.free);

return;

}

x = Jacobi::Jacobi(sols);

oper = x[sols.size] \* (4 \* (sols.size \* sols.size + sols.size));

}

else if (RSeidel->Checked) {

if (!Seidel::isSolved(sols)) {

Info->Text = "Для даної матриці метод Гауса-Зейделя не сходиться.";

utilities::freeMatrix(sols.matrix, size);

utilities::freeMas(sols.free);

return;

}

x = Seidel::Seidel(sols);

oper = x[sols.size] \* (3 \* sols.size \* sols.size + 4\*sols.size);

}

else {

if (!GradientDescent::isSolved(sols)) {

Info->Text = "Для даної матриці метод градієнтного спуску не сходиться.";

utilities::freeMatrix(sols.matrix, size);

utilities::freeMas(sols.free);

return;

}

x = GradientDescent::GradientDescent(sols);

oper = x[sols.size] \* (3 \* sols.size \* sols.size + 37 \* sols.size);

}

Info->Text = "Рішення знайдено за " + (clock() - start) + " мілісекунди, " + (int) x[sols.size] + " ітерацій, " + oper + " операцій" +"!";

print(x);

if (sols.size == 2) {

Graph^ form = gcnew Graph(sols, x);

form->Show();

}

time\_t rawtime;

struct tm \* timeinfo;

char buffer[200];

time(&rawtime);

timeinfo = localtime(&rawtime);

strftime(buffer, 200, "%d.%m.%Y\_%H.%M.%S", timeinfo);

if (RJacobi->Checked) strcat(buffer, "\_Jacobi.txt");

else if (RSeidel->Checked) strcat(buffer, "\_Seidel.txt");

else strcat(buffer, "\_GradientDescent.txt");

FILE\* file = fopen(buffer, "wt");

if (file == nullptr) Info->Text = "Неможливо створити вихідний файл.";

else {

fprintf(file, "Size: %d\n", sols.size);

fprintf(file, "\nMatrix:\n");

for (int i = 0; i < sols.size; i++, fprintf(file, "\n"))

for (int z = 0; z < sols.size; z++)

fprintf(file, "%Lf ", sols.matrix[i][z]);

fprintf(file, "\nFree:\n");

for (int i = 0; i < sols.size; i++) fprintf(file, "%Lf ", sols.free[i]);

fprintf(file, "\n\nSolution:\n");

for (int i = 0; i < size; i++) fprintf(file, "%Lf ", x[i]);

}

fclose(file);

Resize();

utilities::freeMatrix(sols.matrix, size);

utilities::freeMas(sols.free);

utilities::freeMas(x);

}

};

}

Файл «Home.сpp»

#include "Home.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(array<String^>^ arg) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

CourseWork::Home form;

Application::Run(%form);

}