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.

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. Інфіксний 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.

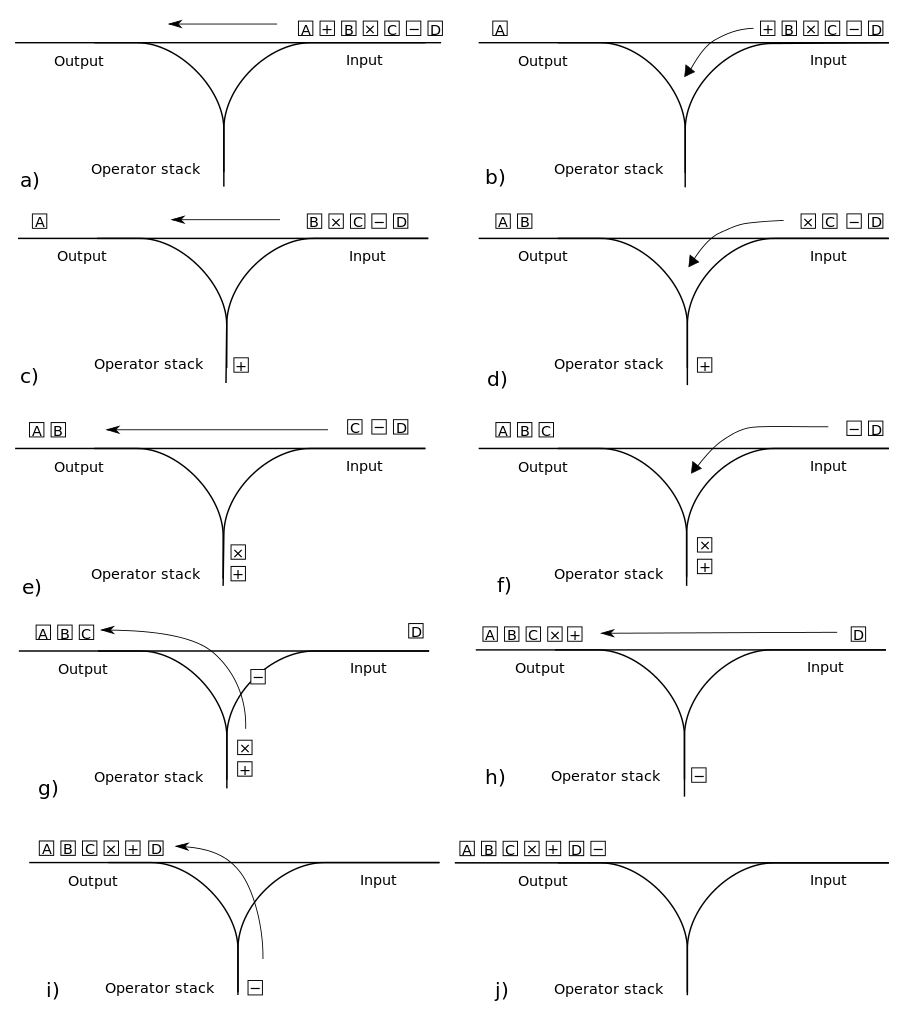


Рисунок 2.1 – Приклад роботи алгоритму сортвальної станції

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

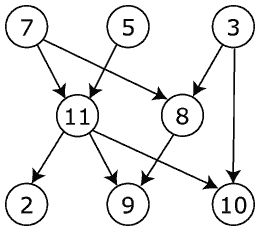


Рисунок 2.3 – Приклад граф, який можливо топологічно впорядкувати.

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

# Опис алгоритмів

## Алгоритм синтаксичного аналізу математичних виразів:

1. ПОЧАТОК
2. Перевести вхідну строку з інфіксної в постфіксну нотацію за допомогою алгоритму сортувальної станції (підрозділ 3.2).
3. ПОКИ не закінчилася вхідна строка:
   1. ЯКЩО поточний символ цифра АБО поточний символ знак мінус та наступний символ цифтра, ТО починаючи з поточної позиції перевести символи в число та помістити його в стек
   2. ЯКЩО поточний символ «$» АБО поточний символ знак мінус та наступний символ «$», ТО:
      1. Знайти координати на які вказує посилання.
      2. ЯКЩО комірка з заданими координатами існує ТА даній комірці зберігається число, ТО помістити дане число в стек, ІНАКШЕ видати повідомлення про помилку.
   3. ЯКЩО поточний символ є елементарною операцією, ТО:
      1. ЯКЩО стек містить менше 2-х значень, ТО видати повідомлення про помилку.
      2. Витягнути зі стеку два числа та виконати над ними задану операцію. Результат операції помістити в стек.
   4. ЯКЩО поточний символ є літерою, ТО:
      1. ЯКЩО стек пустий, то видати повідомлення про помилку.
      2. ЯКЩО існує задана в формулі функція ТА верхній елемент стеку належить області допустимих значень даної функції, ТО виконати дану функці з цим значенням та результат помістити в стек. ІНАКШЕ видати повідомлення про помилку.
4. КІНЕЦЬ

## Алгоритм сортувальної станції:

1. ПОЧАТОК
2. ПОКИ не кінець вхідної строки:
   1. ПОКИ поточний символ є пробілом, ТО перейти до наступного символа.
   2. ЯКЩО досягнути кінець вхідної строки, ТО перейти до пункту 3.
   3. ЯКЩО поточний символ цифра, ТО:
      1. ПОКИ поточний символ є цифрою, ТО вставити поточний символ в вихідну строку та перейти до наступного символу.
      2. ЯКЩО поточний символ є точкою, ТО помістити її вихідну строку та перейти до пункту 2.3.3. ІНАКШЕ перейти до пункту 2.3.4.
      3. ПОКИ поточний символ є цифрою, ТО вставити поточний символ в вихідну строку та перейти до наступного символу.
      4. ЯКЩО поточний символ не пробілом АБО кінцем строки, ТО видати повідомленянн про помилку.
      5. Вставити пробільний символ в вихідну строку.
      6. Перейти до пункту 2.
   4. ЯКЩО поточний символ відкриваюча дужека, ТО помістити її в стек та перейти до пункту 2.
   5. ЯКЩО поточний символ ТО:
      1. ПОКИ стек не пустий ТА поточний елемент стеку не відкриваюча дужка, ТО помістити верхній елемент стеку в вихідну строку та видалити даний елемент зі стеку.
      2. Вставити пробільний символ в вихідну строку.
      3. Перейти до пункту 2.
   6. ЯКЩО поточний символ є символом елементарної операції, ТО:
      1. ЯКЩО поточний символ є знаком мінус ТА вихідна строка пуста АБО верхній елемент стеку є відкриваючою дужкою, ТО помістити знак мінус як унарний операнд в вихідну строку ТА перейти до пункту 2.
      2. ПОКИ стек не пустий ТА на вершині стеку знаходиться операція з пріоритетом не нижчим, ніж дана операції, ТО помістити операцію зі стеку в вихідну строку та видалити її зі стеку.
      3. Помістити поточну операцію у стек.
      4. Встивити пробільний символ в вихідну строку.
      5. Перейти до пункту 2.
   7. ЯКЩО поточний символ «$», ТО:
      1. ПОКИ поточний символ літера, ТО помістити поточну літеру в вихідну строку та перейди до наступного симлову.
      2. ЯКЩО поточний символ не дорівнює «$», ТО видати повідомлення про помилку.
      3. ПОКИ поточний символ цифра, ТО помістити цю цифру в вихідну строку та перейти до наступного символу.
      4. ЯКЩО поточний символом не є пробілом АБО кінцем строки, ТО видати повідомлення про помилку.
      5. Помістити пробіл в вихідну строку.
      6. Перейти до пункту 2.
   8. ЯКЩО поточний символ є літерою, ТО:
      1. ПОКИ поточний символ є літерою, ТО помістити її вихідну строку та перейти до наступнього символу.
      2. ПОКИ поточний символ є пробілом, ТО перейти до наступного символу.
      3. ЯКЩО досягнуто кінець строки АБО поточний символ не є відкриваючою дужкою, ТО видати повідомлення про помилку.
      4. Добавити назву функції в стек.
      5. Добавити відкриваючу дужку в стек.
      6. Добавити пробільний символ в вихідну строку.
      7. Перейти до пункту 2.
   9. Вивести повідомлення про помилку.
3. ПОКИ стек не пустий, ТО помісти операцію з вершини стеку в вихідну строку та видалити її зі стеку.
4. КІНЕЦЬ

## Алгоритм зміни графа таблиці:

1. ПОЧАТОК
2. ЯКЩО змінена комірка містила вірну формулу, ТО:
   1. Знайти список всіх комірок на які посилається стара формула зміненої комірки.
   2. Видалити з графу відповідні ребра між зміненою коміркою та всіма комірками, що знайдені у пункті 2.1.
3. ЯКЩО змінена комірка у комірку введено вірно формулу, ТО:
   1. Знайти список всіх комірок на які посилається нова формула зміненої комірки.
   2. Добавити у граф відповідні ребра між зміненою коміркою та всіма комірками, що знайдені у пункті 3.1.
4. Виконати топологічне сортування графу, починаючи зі зміненої комірки (розділ 3.5) та помістити результат сортування у список.
5. Видалити змінену комірку зі списку знайденому у пункті 4.
6. ПОКИ список, що знадений у пункті 5, містить елементи, ТО:
   1. Обчислити значення комікри, що містить у хвості списку.
   2. Видалити комірку зі хвосту списку.
7. КІНЕЦЬ

## Алгоритм топологічного сортування

1. ПОЧАТОК
2. Стоворити масив відвіданих вершин.
3. Виконати обхідн в глубину графа (розділ 3.6) починаючи з комірнки, що була змінена.
4. КІНЕЦЬ

## Рекурсивний алгоритм обходу графу у глубину

1. ПОЧАТОК
2. Помітити поточну вершину, як ту, що знаходиться в процесі обробки.
3. ЦИКЛ по всім ребрам, що виходять з поточної вершини:
   1. ЯКЩО вершина в яку веде поточне ребро ще не відвідана, ТО рекурсивний виклик алгоритму обходу графу у глубину (розділ 3.6).
   2. ЯКЩО поточно вершина знаходить в процесі обробки, ТО помітити поточну вершинау як з’єднувачем графа.
4. Помітити поточну вершину як оброблену.
5. Помістити поточну вершину у чергу, що є топологічно відсортованою.
6. КІНЕЦЬ

# 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) Cell.h – реалізує комірку таблиці;

б) Graph.h – реалізує побудову та проведення дій з графом таблиці;

в) Number.h – реалізує тип число, що є основним контейнером результатів;

г) Parser.h – реалізує механізм синтаксичного розбору математичних виразів;

д) Table.h – реалізує талицю;

е) resource.h – містить унікальні індентифікатори ресурсів проекту;

є) Excel.h - реалізує головне вікно программи;

ж) New.h - реалізує вікно створення нової таблиці;

з) About.h - реалізує вікно About;

и) Help.h - реалізує вікно зі справкою;

і) Save.h - реалізує вікно запитання про збереження незбереженого файлу;

### 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 | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 40 | Excel | WantSave | Виводить повідомлення про те що файл не збережено | - | - |
| 41 | Excel | SaveLastFiles | Зберігає список останніх файлів | - | - |
| 42 | Excel | LoadLastFiles | Завантажує список останніх файлів | - | - |
| 43 | Excel | UpdateLastFiles | Додає файл до списку останніх використаних | Шлях до файлу | - |
| 44 | Excel | CollumnHeader | Підраховує заголовок i-го стовпця | Номер стовпця | Строка - заголовок |
| 45 | Excel | toStdWstring | Перетворює .net рядок у с++ рядок | Рядок | Рядок |
| 46 | Excel | DeleteTable | Очищує таблицю | - | - |
| 47 | Excel | CreateTable | Створює таблицю | Розмір нової таблиці | - |
| 48 | Excel | ReCreateTable | Заново створює таблицю | Розмір нової таблиці | - |
| 49 | Excel | UpdateText | Поновлює текст комірки таблиці | Рядок з новим текстом, координати комірки | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 50 | Excel | Excel\_Closed | Обробляє закриття вікна | Об’єкт що надсилає подію, аргументи події | - |
| 51 | Excel | Excel\_Load | Обробляє відкриття вікна | Об’єкт що надсилає подію, аргументи події | - |
| 52 | Excel | dataGridView1\_CurrentCellChanged | Обробляє зміну поточної комірки | Об’єкт що надсилає подію, аргументи події | - |
| 53 | Excel | dataGridView1\_CellValueChanged | Обробляє зміну данних у комірці | Об’єкт що надсилає подію, аргументи події | - |
| 54 | Excel | textBox1\_KeyPress | Обробляє нажаті клавіші у строці вводу | Об’єкт що надсилає подію, аргументи події | - |
| 55 | Excel | dataGridView1\_CellEndEdit | Обробляє завершення вводу данних до комірки користувачем | Об’єкт що надсилає подію, аргументи події | - |
| 56 | Excel | saveToolStripMenuItem\_Click | Обробляє нажимання на кнопку Save | Об’єкт що надсилає подію, аргументи події | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 58 | Excel | dataGridView1\_CellBeginEdit | Обробляє початок вводу данних до комірки користувачем | Об’єкт що надсилає подію, аргументи події | - |
| 59 | Excel | newToolStripMenuItem\_Click | Обробляє нажимання на кнопку New | Об’єкт що надсилає подію, аргументи події | - |
| 60 | Excel | aboutToolStripMenuItem\_Click | Обробляє нажимання на кнопку About | Об’єкт що надсилає подію, аргументи події | - |
| 61 | Excel | toolStripMenuItem2\_Click | Обробляє нажимання на будь який з останніх файлів | Об’єкт що надсилає подію, аргументи події | - |
| 62 | Excel | closeToolStripMenuItem\_Click | Обробляє нажимання на кнопку Close | Об’єкт що надсилає подію, аргументи події | - |
| 63 | Excel | functionsToolStripMenuItem\_Click | Обробляє нажимання на кнопку Docs | Об’єкт що надсилає подію, аргументи події | - |
| 64 | Excel | dataGridView2\_CurrentCellChanged | Обробляє зміну вибраної комірки в 2й таблиці | Об’єкт що надсилає подію, аргументи події | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 66 | New | getRows | Повертає введене число рядків | - | Число рядків |
| 67 | New | getCols | Повертає введене число стовпців | - | Число стовпців |
| 68 | New | button1\_Click | Обробляє нажимання на кнопку Create | Об’єкт що надсилає подію, аргументи події | - |
| 69 | New | New\_Load | Обробляє відкриття вікна | Об’єкт що надсилає подію, аргументи події | - |
| 70 | Save | button1\_Click | Обробляє нажимання на кнопку Yes | Об’єкт що надсилає подію, аргументи події | - |
| 71 | Save | button2\_Click | Обробляє нажимання на кнопку No | Об’єкт що надсилає подію, аргументи події | - |

### 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 |
| 14 | ShowDialog | Показує діалогове вікно | - | - | - |
| 15 | strcpy | Копіює рядок | 2 рядки | - | string.h |
| 16 | strcat | Зклеює рядки | 2 рядки | - | String.h |
| 17 | mkdir | Зтворює каталог | Рядок з адресою нового каталогу | Код помилки | direct.h |
| 18 | GetBytes | Перетворює вхідні данні на масив байтів | Данні | Масив байтів | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 19 | Write | Записує масив байтів у файл | Масив байтів, зміщення, розмір | - | - |
| 20 | WriteByte | Записує байт у файл | Байт данних | - | - |
| 21 | getenv | Повертає змінну системи | Ім’я змінної | Рядок зі змінною | stdlib.h |
| 22 | Read | Зчитує масив байт з файлу | зміщення, розмір | Масив байтів | - |
| 23 | ToInt32 | Перетворює масив байтів у int | Масив байтів | Число int | - |
| 24 | Resize | Змінює розмір массиву | Масив, новий розмів | - | - |
| 25 | ReadByte | Зчитує байт з файлу | - | Байт данних | - |
| 26 | Add | Додає елемент до коллекції | Елемент | - | - |
| 27 | Equals | Порівнює .net рядки | 2 рядки | Індикатор рівності рядків | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 28 | ToCharArray | Переводить рядок .net в масив символів | Рядок | Масив символів | - |
| 29 | Remove | Видаляє єлемент з коллекції | Елемент | - | - |
| 30 | Add | Додає стовпець | Ім’я стопвця, заголовок стовпця | - | - |
| 31 | Add | Додає рядки | Кількість рядків | - | - |
| 32 | clock | Повертає час | - | Час | time.h |
| 33 | OpenFile | Відкриває файл вибраний в діалозі вибору файла | - | - | - |
| 34 | ToString | Перетворює об’єкт у рядок | Об’єкт | Рядок | - |
| 35 | Close | Закриває вікно | - | - | - |

# Тестування програмного забезпечення

## Теорія

Тестування програмного забезпечення — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:

* відповідність вимогам, якими керувалися проектувальники та розробники
* правильна відповідь для усіх можливих вхідних даних
* виконання функцій за прийнятний час
* практичність
* сумісність з програмним забезпеченням та операційними системами
* відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення багів (помилок або інших дефектів).

Тестування ПЗ може надавати об'єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів так і для замовників.

Тестування може проводитись, як тільки створено виконуваний код (навіть частково завершено). Процес розробки зазвичай передбачає коли та як буде відбуватися тестування. Наприклад, при поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На противагу цьому, відповідно до вимог гнучкої розробки ПЗ, програмування і тестування часто відбувається одночасно.

## 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:

1. Тестування правильності введених значень.
   1. Тестування при введенні некоректних посилань
   2. Тестування при введенні некоретних чисел
   3. Тестування при введенні неіснуючої функції
2. Тестування коректності роботи
   1. Перевірка коректності збереження файлу
   2. Перевірка коректності загрузки зі збереженого файлу
   3. Перевірка коректності роботи при створенні таблиці малих розмірів
   4. Перевірка коректності роботи при стовренні таблиці великих розмірів
   5. Перевірка поведінки програми при закриті файлу, що містить не зберегжені зміни

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

### Тестування правильності введених значень

#### Тестування при введенні некоректних посилань

Якщо під час введення інформації в таблицю користувач введе посилання на комірку, що не існує (рисунок 5.1, рисунок 5.2, рисунок 5.3) або введе посилання у невірному форматі (рисунок 5.4), то в відповідну комірку програма видасть попередження.

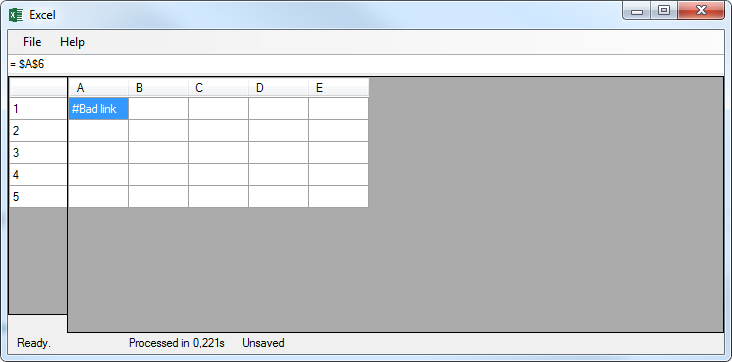


Рисунок 5.1 – Вікно програми, після введення посилання на неіснуючий рядок

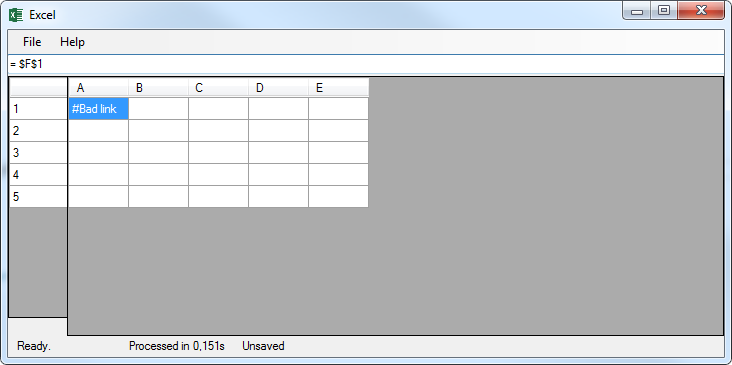


Рисунок 5.2 – Вікно програми, після введення посилання на неіснуючию колонку

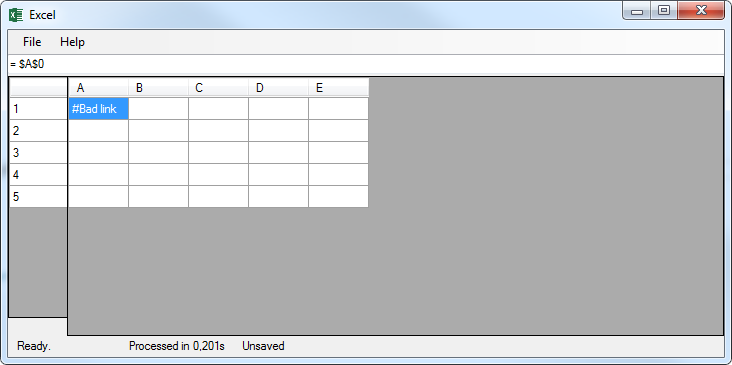


Рисунок 5.3 – Вікно програми, після введення посилання на неіснуючий рядок

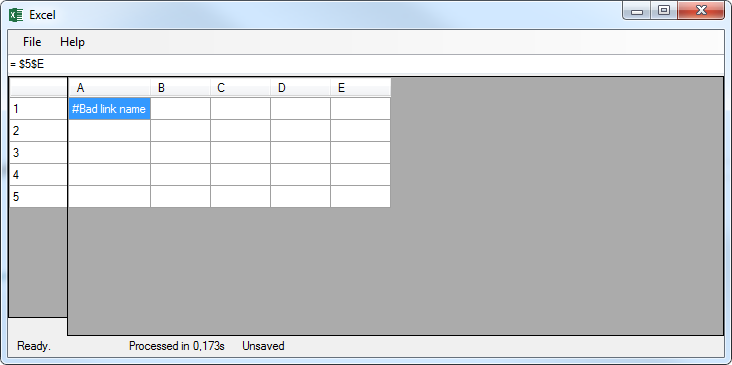


Рисунок 5.4 – Вікно програми, після введення посилання у некоректному форматі

#### Тестування при введенні некоретних чисел

Якщо під час введення інформації в таблицю користувач введе число у некоректному форматі (рисунок 5.5), то в відповідну комірку програма видасть попередження.

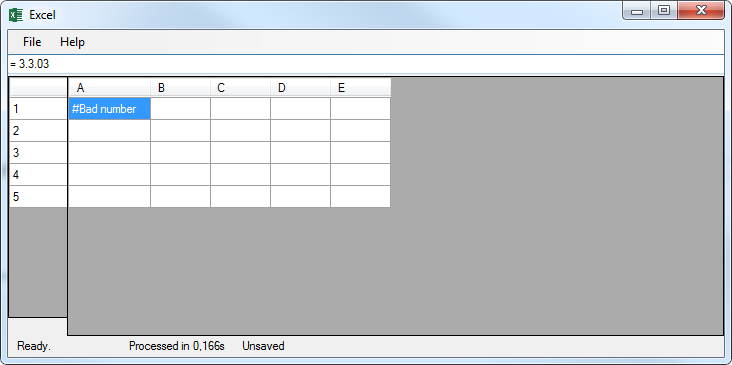


Рисунок 5.5 – Вікно програми, після введення числа у некоректному форматі

#### Тестування при введенні неіснуючої функції

Якщо під час введення інформації в таблицю користувач введе назву неісуючої функції (рисунок 5.6), то в відповідну комірку програма видасть попередження.

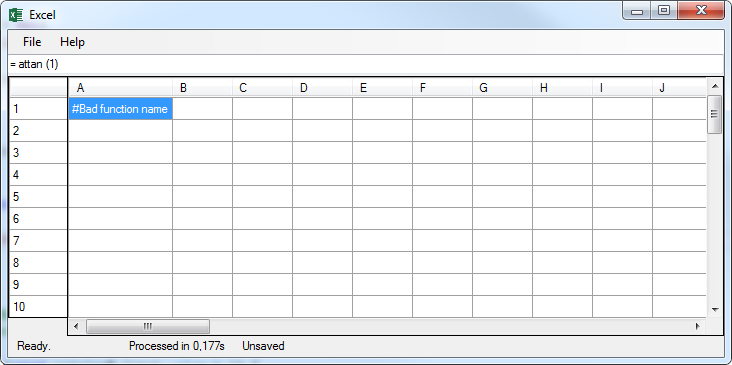


Рисунок 5.6 – Вікно програми, після введення неіснуючої функції

### Тестування коректності роботи

#### Перевірка коректності збереження файлу

Якщо під час користування програмним продуктом буде внесено якісь зміни у файл, то програма виведе відповідне повідомлення в полосці статусу (рисунок 5.7). Відповідні зміни можна зберегти у новий файл або поточний. Тоді відповідно дане повідомлення зникне (рисунок 5.8).

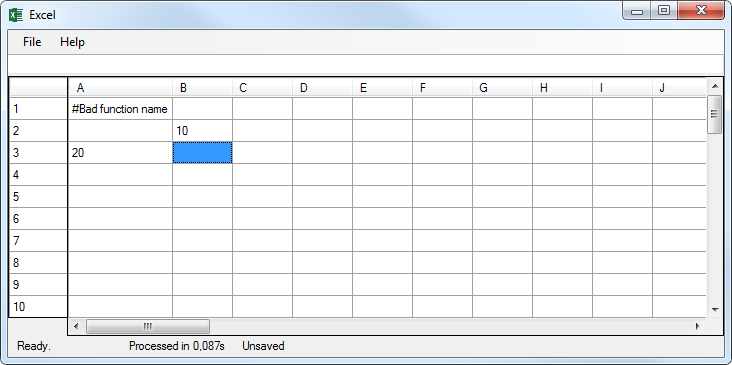


Рисунок 5.7 – Вікно програми, після внесення змін у файл

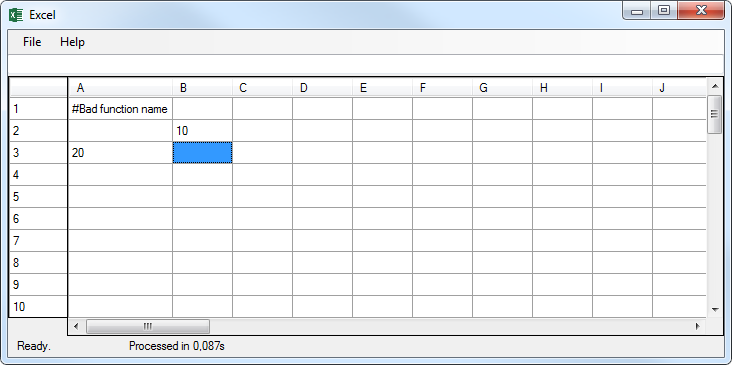


Рисунок 5.8 – Вікно програми, після збереження у файл

#### Перевірка коректності відкриття файлу

Відкриття збереженого файла в пункті 5.3.2.1 зображено на рисунок 5.9.

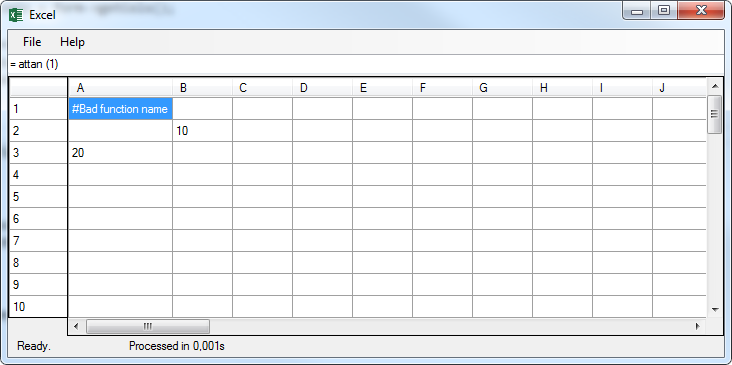


Рисунок 5.9 – Вікно програми, після відкриття таблиці з файлу

#### Перевірка коректності роботи при створенні таблиці малих розмірів

Якщо користувач створить таблицю мінімального можливого розміру (1х1) та введе в неї які-небудь дані, то програма спрацює коректно (рисунок 5.10).

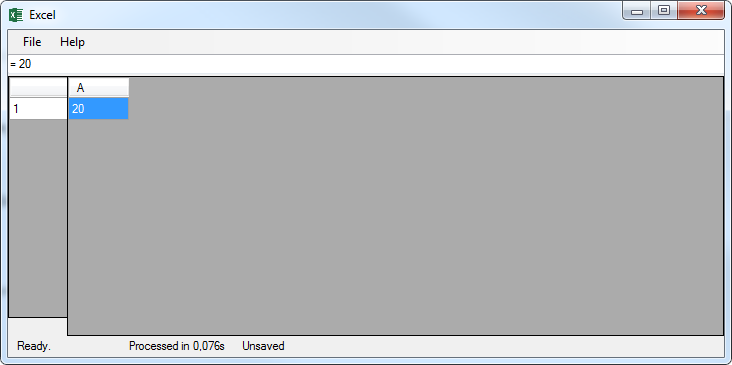


Рисунок 5.10 – Вікно програми, після створення таблиці мінімального розміру

#### Перевірка коректності роботи при стовренні таблиці великих розмірів

Якщо користувач створить таблицю максимально можливого розміру (500х500) та введе в неї які-небудь дані, то програма спрацює коректно   
(рисунок 5.1).

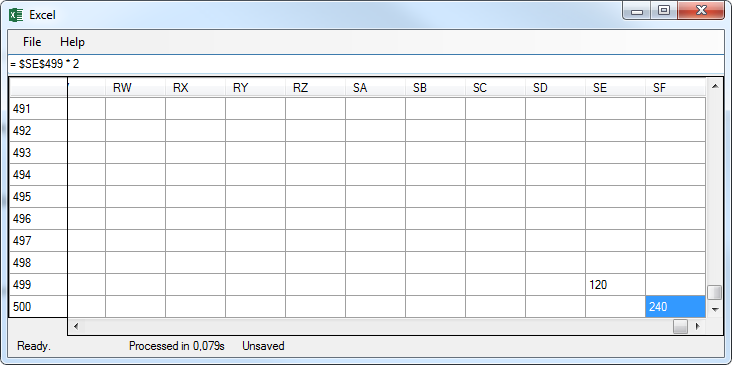


Рисунок 5.11 – Вікно програми, після створення таблиці максимального розміру

#### Перевірка поведінки програми при закриті файлу, що містить не зберегжені зміни

Якщо користувач внесе певні зміни у файл та спробує його зберегерти то він отримає повідомлення про те, що файл містить не збережені змінни (рисунок 5.12).

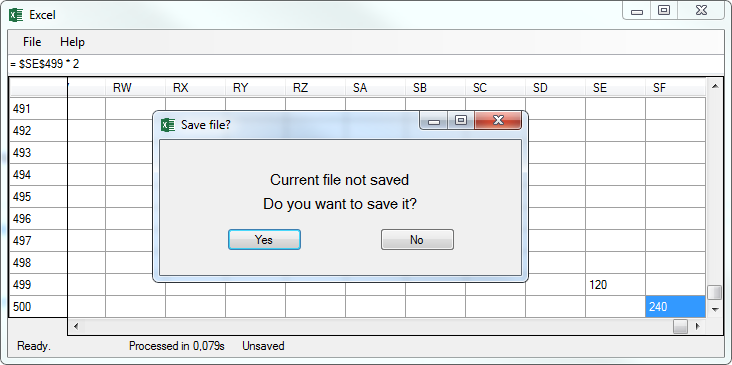


Рисунок 5.12 – Вікно програми, після спроби закрити файл, що містить незбрежені змінни

### 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 обробки посилань

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані посилань у формулі |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір таблиці: 50 на 50 |
| Схема проведення тесту | Введення даних та натискання кнопки «Enter» |
| Очікуваний результат | Виведення рішення формули |
| Стан програми після проведення випробувань | Виведено рішення формули |

Таблиця 5.2 – Тестування обробки функцій

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані функцій у формулі |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір таблиці: 50 на 50 |
| Схема проведення тесту | Введення даних та натискання кнопки «Enter» |
| Очікуваний результат | Виведення рішення формули |
| Стан програми після проведення випробувань | Виведено рішення формули |

Таблиця 5.3 – Тестування обробки складних формул з багатьма посиланнями та функціями

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректну роботу програми при використані обробки складних формул з багатьма посиланнями та функціями |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Розмір таблиці: 50 на 50 |
| Схема проведення тесту | Введення даних та натискання кнопки «Enter» |
| Очікуваний результат | Виведення рішення формули |
| Стан програми після проведення випробувань | Виведено рішення системи |

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

# Інструкція користувача

## Робота з програмою

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

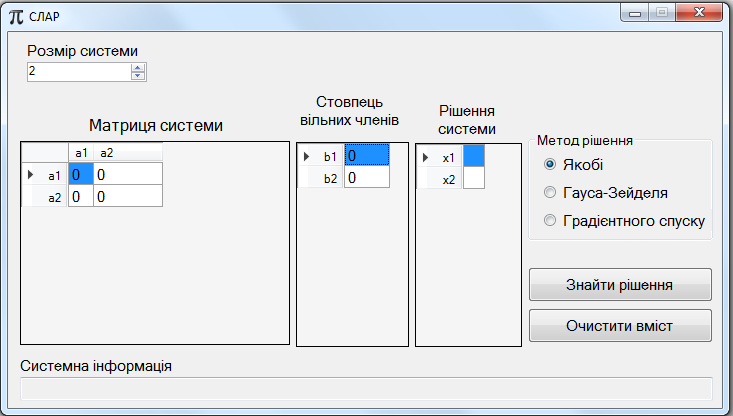


Рисунок 6.1 – Головне вікно програми

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):

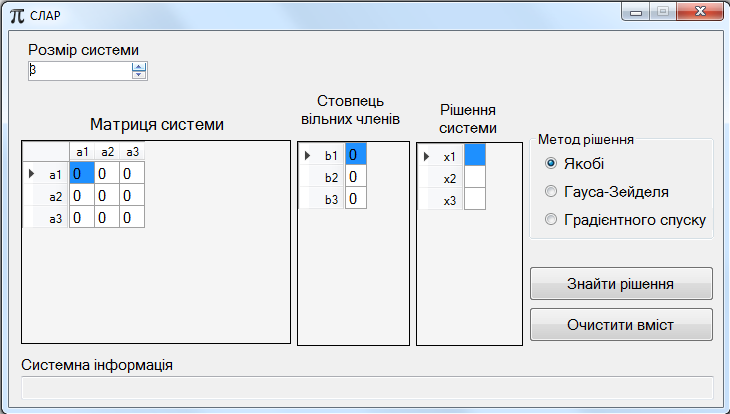


Рисунок 6.2 – Вибір необхідного розміру системи

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

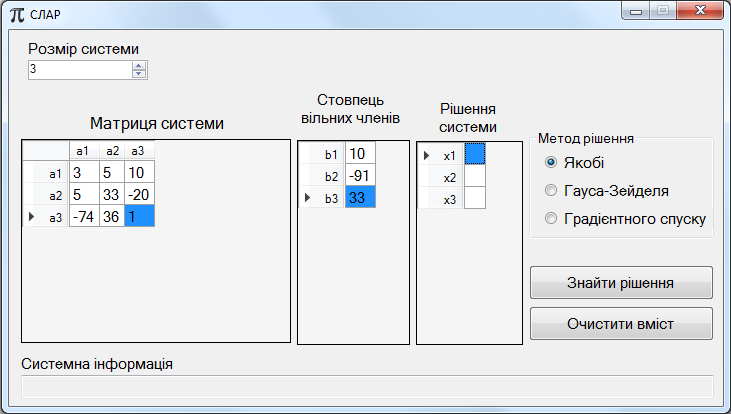


Рисунок 6.3 – Введення вхідних даних

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.

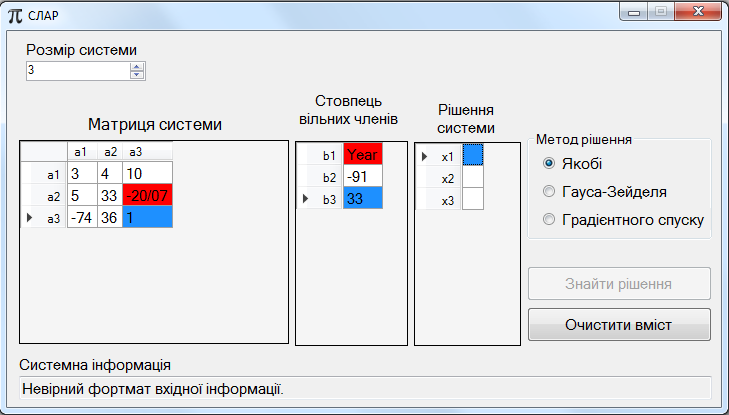


Рисунок 6.4 – Приклад некоректного введення даних користувачем

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):

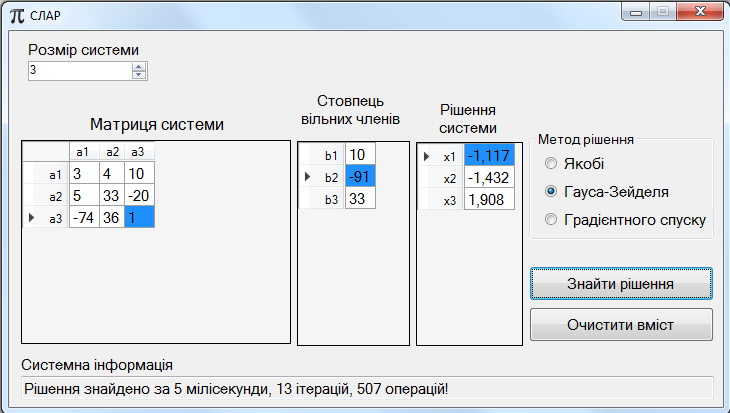


Рисунок 6.5 – Приклад результату роботи програми

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

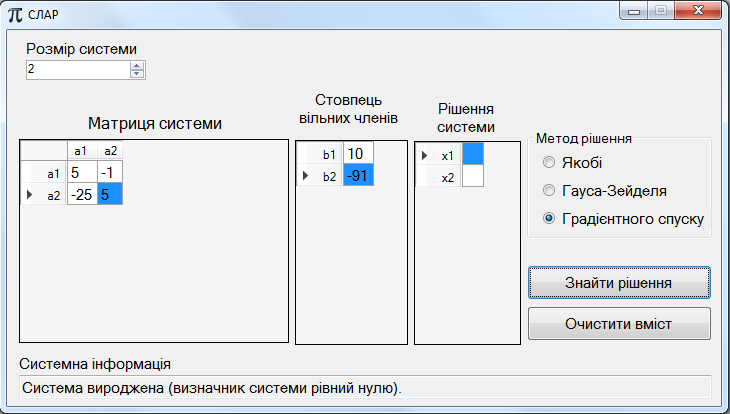


Рисунок 6.6 – Приклад введення системи для якої не існує розв’язку

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

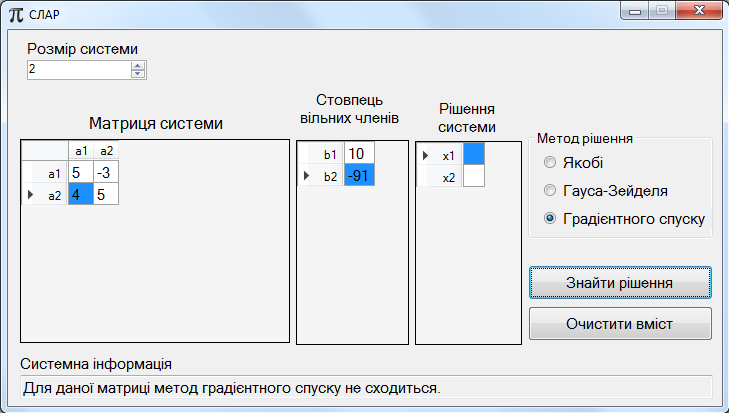


Рисунок 6.7 – Приклад введення системи для якої обраний метод   
не сходиться

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):

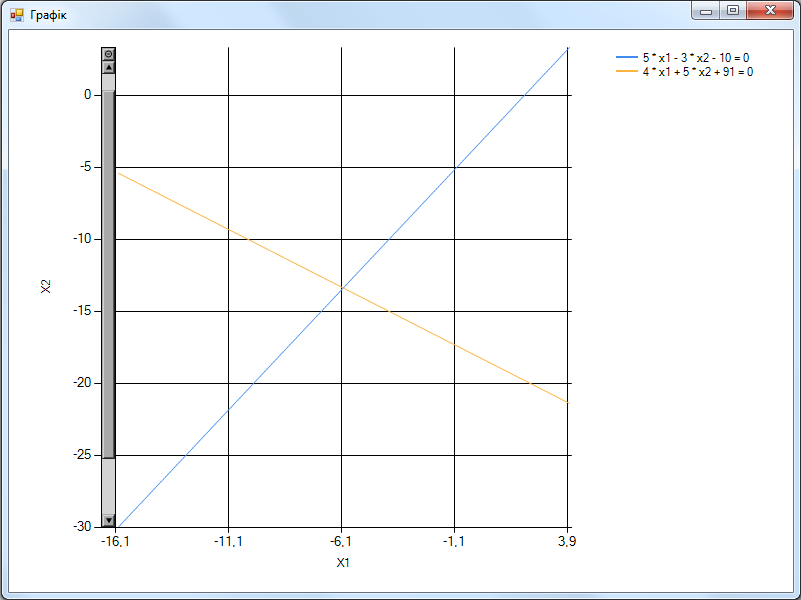


Рисунок 6.8 – Приклад введення графіку системи

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

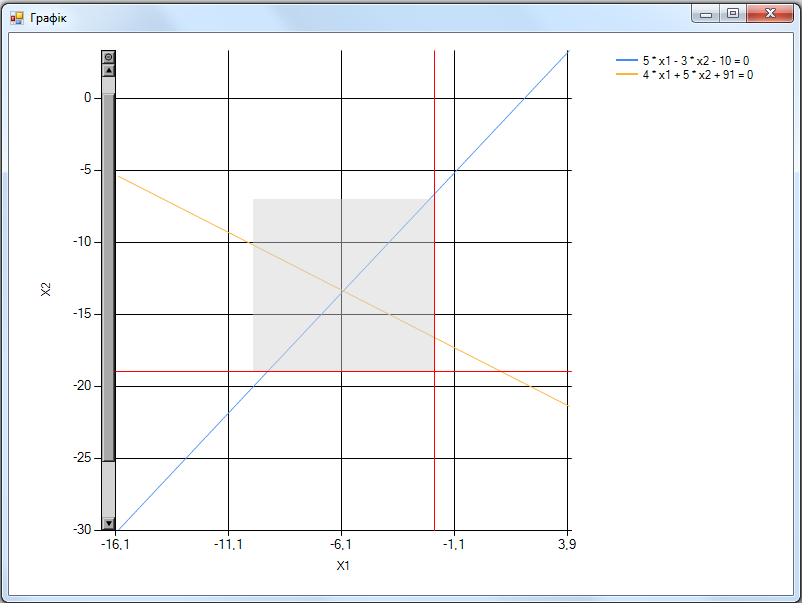


Рисунок 6.9 – Приклад виділення області графіку для масштабування

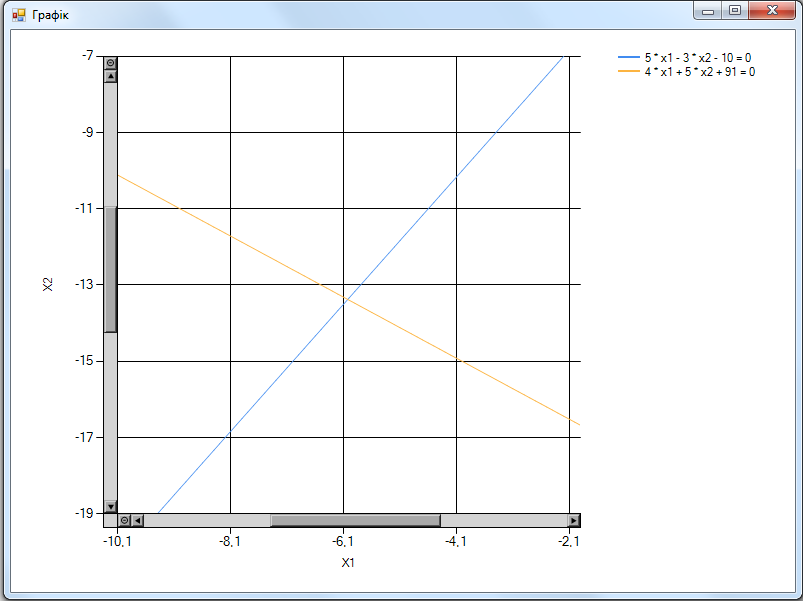


Рисунок 6.10 – Рисунок 6.9 після масштабування

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 | |

Висновки

На етапі ознайомлення з теоретичними аспектами завдання даної курсової роботи було виявлено необхідні механізми для вирішення поставленої задачі та алгоритми їх реалізації, проаналізовано їх складність виконання, 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 було реалізовано наступний функціонал: синтаксичний аналізатор, механізм перерахунку залежних формул за допомогою використання топологічного сортування, функції збереження та відкриття таблиці з провідника операційної системи Windows. Результатом розробки програми є набуття досвіду роботи з електронними таблицями, графами та структурами даних STL, закріплення, узагальнення та поглиблення знань і навичок програмування на мовах С\С++, набуття практичних навичок алгоритмізації прикладних задач.

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 навчально-інженерних цілях та/або бухгалтерському обліку з метою спрощення важких та об’ємних обчислень.

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