**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**ВСП «РІВНЕНСЬКИЙ ФАХОВИЙ КОЛЕДЖ НУБіП УКРАЇНИ»**

*Циклова комісія програмування та інформаційних дисциплін*

**КУРСОВА РОБОТА**

**з дисципліни «Інструментальні засоби візуального програмування»**

**на тему**

**Програма розв’язку «Задач лінійного програмування»**

Студента IV курсу 41- ІП групи

Галузь знань 12 Інформаційні

технології

Спеціальність 121 Інженерія

програмного забезпечення

Мельника Владислава Олександровича

Керівник: викладач комп’ютерно-

інформаційних дисциплін,   
Крупеня В. О.

Національна шкала  
Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Члени комісії**

(підпис)  (прізвище та ініціали)

(підпис)  (прізвище та ініціали)

(підпис)  (прізвище та ініціали)

м. Рівне- 2024

Циклова комісія програмування та інформаційних дисциплін

Дисципліна «Інструментальні засоби візуального програмування»

Галузь знань 12 Інформаційні технології

Спеціальність 121 Інженерія програмного забезпечення

курс ІV, група 41-ІП, семестр І

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| ***Мельник Владислав Олександрович*** |

(прізвище та ініціали студента)

Тема. **Програма розв’язку «Задач лінійного програмування»**

|  |  |
| --- | --- |
| **Вихідні дані:** | Програма містить такі функції як збереження інформації у бінарний файл |

**Зміст завдання та календарний план його виконання**

|  |  |  |
| --- | --- | --- |
| ***№***  ***з/п*** | ***Зміст завдання*** | ***Дата виконання*** |
| **1** | Здійснити аналітичний огляд літературних джерел за обраною темою | **05.09.24** |
| **2** | Дослідити які основні завдання повинна виконувати програма та провести аналіз засобів вирішення цих завдань | **20.09.24** |
| **3** | Здійснити проектування програмного продукту шляхом формування загальної архітектури та прив’язки її до конкретного середовища функціонування. | **29.09.24** |
| **4** | Запрограмувати функції об’єктів програмного забезпечення | **05.10.24** |
| **5** | Розробити графічний інтерфейс та навести ілюстрації роботи | **11.10.24** |
| **6** | Розробка форм вихідних документів, довідкової інформації. | **17.10.24** |
| **7** | Тестування програмного продукту. | **22.10.24** |
| **8** | Оформити пояснювальну записку до курсової роботи згідно вимог державних стандартів, дотримуючись такого змісту:   * вступ; * аналіз предметної області та вимог до програмного продукту; * спеціальна частина; * функціональний опис та ілюстрація роботи програмного продукту; * висновки; * використані літературні джерела; * додатки | ***Впродовж семестру*** |
| **9** | Попередній захист | **28.10.24** |

Завдання прийнято до виконання «*04*» *вересня* 2024 р.

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | *Мельник В. О.* |
|  | (підпис студента) |  | (прізвище та ініціали студента) |
|  |  |  |  |
| Керівник роботи |  |  | *Крупеня В. О.* |
|  | (підпис керівника) |  | (прізвище та ініціали керівника) |

**Зміст**

[**Вступ** 4](#_Toc181209212)

[**РОЗДІЛ I. Аналіз предметної області та вимог до програмного продукту** 5](#_Toc181209213)

[**РОЗДІЛ II. Спеціальна частина** 8](#_Toc181209214)

[**РОЗДІЛ III. Функціональний опис та ілюстрація роботи програмного продукту** 16](#_Toc181209215)

[**Висновок** 21](#_Toc181209216)

[ЛІТЕРАТУРА 23](#_Toc181209217)

[ДОДАТОК А 24](#_Toc181209218)

# **Вступ**

Задачі лінійного програмування є важливою частиною сучасної математики та інформатики, які мають велике значення в оптимізації процесів, аналізі ресурсів і прийнятті управлінських рішень. Вони широко застосовуються в економіці, логістиці, виробничому плануванні, управлінні запасами та інших сферах, де необхідно максимізувати прибуток або мінімізувати витрати за умов обмежених ресурсів. Одним з найбільш відомих і ефективних методів розв’язання таких задач є симплекс-метод, який дозволяє знаходити оптимальні рішення для великих систем лінійних нерівностей.

Мета нашої програми — надати користувачеві зручний інструмент для введення даних, розв’язання задач лінійного програмування методом симплекс-таблиць, а також відображення процесу розв’язання з детальними проміжними результатами. Програма дозволяє працювати з бінарними файлами для зберігання та завантаження симплекс-таблиць, що значно полегшує процес роботи з великими обсягами даних.

Даний програмний продукт орієнтований на студентів, науковців і фахівців, які мають потребу у швидкому розв’язанні задач оптимізації та візуалізації результатів. Завдяки зручному інтерфейсу та можливостям збереження й завантаження результатів, наша програма стане незамінним інструментом для розуміння та реалізації лінійного програмування в різних галузях.

# **РОЗДІЛ I. Аналіз предметної області та вимог до програмного продукту**

Лінійне програмування (ЛП) — це метод математичної оптимізації, який дозволяє знаходити оптимальні рішення у задачах, де цільова функція і обмеження є лінійними. ЛП широко застосовується у різних галузях, зокрема в економіці, логістиці, виробництві та управлінні ресурсами, щоб оптимізувати витрати, максимізувати прибутки чи мінімізувати ризики. Основний метод розв’язання задач лінійного програмування — симплекс-метод, який і буде реалізований у даному програмному продукті.

Мета програми — надати користувачеві інструмент для розв’язання задач лінійного програмування із застосуванням симплекс-методу. Програма повинна бути простою у використанні, мати графічний інтерфейс, дозволяти вводити симплекс-таблиці та автоматизувати обчислення, надаючи користувачеві проміжні результати.

Задачі лінійного програмування, як правило, мають структуру, що включає:

* **Цільову функцію** — функцію, яку потрібно максимізувати або мінімізувати (наприклад, прибуток або витрати).
* **Систему обмежень** — лінійні рівняння чи нерівності, які обмежують можливі значення змінних (наприклад, обмеження на ресурси).
* **Невід’ємність змінних** — обмеження, яке вимагає, щоб змінні набували лише додатних або нульових значень, що важливо для багатьох практичних завдань.

Задачі лінійного програмування можуть бути вирішені шляхом побудови симплекс-таблиці та виконання симплекс-методу для переходу від початкового розв'язку до оптимального.

**1. Основні вимоги до програмного продукту**

**1.1 Функціональні вимоги**

* **Введення даних**:
  + Користувач повинен мати можливість вводити початкові дані задачі, зокрема коефіцієнти цільової функції та обмежень.
  + Підтримка багаторядкового введення для зручного редагування симплекс-таблиці.
* **Обробка та збереження файлів**:
  + Збереження введених даних у бінарний файл, який можна пізніше завантажити, щоб продовжити розв’язання задачі.
  + Завантаження бінарних файлів із даними задачі.
* **Розв’язання задачі**:
  + Реалізація симплекс-методу для розв’язання задачі.
  + Автоматичне виконання симплекс-ітерацій із збереженням проміжних результатів, що дозволяє простежити етапи виконання.
  + Відображення кінцевого розв'язку, а також попередніх кроків симплекс-методу (таблиць), щоб користувач міг розібратися у ході обчислень.
* **Інтерфейс користувача**:
  + Наявність зрозумілого графічного інтерфейсу (на базі WPF), який забезпечує простоту введення та редагування даних, управління функціями та перегляд результатів.
  + Меню з пунктами для відкриття, збереження файлів, початку обчислень і перегляду інформації про програму.

**1.2 Нефункціональні вимоги**

* **Простота використання**: Інтерфейс має бути інтуїтивно зрозумілим та зручним для користувача.
* **Продуктивність**: Обробка даних повинна бути швидкою та не перевантажувати систему навіть для великих обсягів даних.
* **Надійність**: Програма повинна обробляти помилки введення даних і некоректне використання.
* **Розширюваність**: Код програми має бути структуруваним і піддаватися розширенню, наприклад, для реалізації інших методів розв’язання задач оптимізації.
* **Сумісність**: Програма має працювати на Windows з підтримкою .NET Framework або .NET Core.

**2. Сценарії використання**

1. **Створення задачі**: Користувач запускає програму, вводить початкові дані задачі у відповідні поля для коефіцієнтів та обмежень.
2. **Збереження та завантаження задачі**: Користувач може зберегти введені дані задачі у бінарний файл, а пізніше завантажити його для подальшого розв’язання.
3. **Запуск симплекс-методу**: Користувач натискає кнопку для запуску симплекс-методу, програма автоматично виконує всі ітерації та показує проміжні та кінцеві результати.
4. **Перегляд результатів**: Користувач може бачити кінцевий результат та кроки, які виконувалися, щоб досягти оптимального розв’язку.
5. **Скидання введених даних**: Користувач має можливість очистити всі введені дані та почати нову задачу.

# **РОЗДІЛ II. Спеціальна частина**

Програмний продукт, розроблений у рамках цієї курсової роботи, призначений для автоматизації процесу розв’язання задач лінійного програмування. Основна мета програми — забезпечити зручний інструмент для виконання обчислень за допомогою симплекс-методу, що дозволяє швидко знайти оптимальні значення змінних для мінімізації чи максимізації цільової функції. Програма спрямована на спрощення роботи користувача шляхом автоматичного проведення симплекс-ітерацій, збереження даних і зниження ймовірності помилок у розрахунках.

Програма дозволяє користувачам вводити коефіцієнти цільової функції та обмеження, після чого здійснює автоматичні обчислення за допомогою симплекс-методу. Окрім обчислень, продукт дозволяє зберігати задачі у бінарному форматі для подальшого завантаження, що є зручним для роботи з великою кількістю даних.

Ключовою функцією програми є покрокове розв’язання задачі: від побудови початкової симплекс-таблиці до отримання оптимального розв’язку, з можливістю перегляду проміжних кроків. Такий підхід допомагає користувачам краще розуміти хід обчислень і відслідковувати результати на кожному етапі.

**Функціональні можливості**

* **Введення даних задачі**: Інтерфейс програми дозволяє зручно вводити дані, включаючи коефіцієнти цільової функції та обмеження.
* **Збереження та завантаження задач**: Програма підтримує збереження задач у бінарний файл, що дозволяє зберігати прогрес роботи для подальшого розв’язання або аналізу.
* **Розв’язання задачі симплекс-методом**: Програма виконує симплекс-ітерації та виводить проміжні результати, що дозволяє користувачу бачити всі етапи розрахунків.
* **Відображення результатів**: Після виконання всіх ітерацій програма надає оптимальний розв’язок задачі, а також надає можливість перегляду всіх проміжних таблиць.

Ця програма надає можливість швидко й ефективно розв'язувати задачі лінійного програмування, знижуючи ризик помилок, пов’язаних із ручними обчисленнями, а також спрощує процес навчання та розуміння симплекс-методу завдяки покроковому відображенню результатів. Збереження задач у файлі дозволяє користувачу зберігати прогрес і повертатися до задачі в будь-який час.

Розроблений програмний продукт є ефективним інструментом для автоматизації процесу розв’язання задач лінійного програмування, який спрощує навчання та роботу з оптимізаційними моделями. Подальші розділи курсової роботи охоплять реалізацію основних функцій програми, методи обробки даних і структуру збереження інформації.

**Збереження даних у бінарному форматі**

Програмний продукт включає методи для збереження та завантаження симплекс-таблиці у бінарному форматі. Завдяки використанню бінарних файлів програма дозволяє зберігати симплекс-таблиці компактно та надійно, що полегшує їхнє подальше завантаження і використання.

**Метод Save\_Click для збереження симплекс-таблиці у бінарний файл**

private void Save\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "Binary Files (\*.bin)|\*.bin",

Title = "Save Simplex Table as Binary File"

};

if (saveFileDialog.ShowDialog() == true)

{

try

{

using (BinaryWriter writer = new BinaryWriter(File.Open(saveFileDialog.FileName, FileMode.Create)))

{

// Записуємо дані з текстового поля у файл

writer.Write(InputTextBox.Text);

}

MessageBox.Show("File saved successfully!", "Success", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Error saving file: {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

**Опис методу Save\_Click:**

* **Діалог збереження файлу**: Метод відкриває діалог збереження, де користувач обирає місце та ім'я файлу для симплекс-таблиці.
* **Запис даних**: У разі підтвердження, використовуючи BinaryWriter, програма записує текст із текстового поля InputTextBox у бінарний файл.
* **Повідомлення**: Якщо файл збережено успішно, програма повідомляє користувача про успіх, а у разі виникнення помилки — виводить повідомлення про неї.

**Метод Open\_Click для завантаження симплекс-таблиці з бінарного файлу**

private void Open\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "Binary Files (\*.bin)|\*.bin",

Title = "Open Simplex Table Binary File"

};

if (openFileDialog.ShowDialog() == true)

{

try

{

using (BinaryReader reader = new BinaryReader(File.Open(openFileDialog.FileName, FileMode.Open)))

{

// Читаємо дані з файлу і завантажуємо їх у текстове поле

InputTextBox.Text = reader.ReadString();

}

MessageBox.Show("File loaded successfully!", "Success", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Error loading file: {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

**Опис методу Open\_Click:**

* **Діалог відкриття файлу**: Метод відкриває діалог, де користувач обирає бінарний файл для завантаження симплекс-таблиці.
* **Читання даних**: Використовуючи BinaryReader, метод зчитує дані з обраного файлу та завантажує їх у текстове поле InputTextBox.
* **Повідомлення**: Якщо завантаження успішне, програма інформує про це користувача, а у разі помилки виводить відповідне повідомлення.

**Переваги використання бінарного формату**

* **Ефективне зберігання**: Бінарні файли займають менше місця, ніж текстові, що є корисним при частому збереженні великих симплекс-таблиць.
* **Швидке читання та запис**: Використання BinaryWriter та BinaryReader забезпечує швидкі операції з даними.
* **Мінімізація помилок формату**: Дані зберігаються та завантажуються без необхідності конвертації, що зменшує ймовірність помилок під час операцій.

Цей підхід надає програмі зручності у роботі з симплекс-таблицями та забезпечує ефективність обробки даних для задач лінійного програмування.

**Розв’язування задач за допомогою симплекс-методу**

Розв'язок задачі за допомогою симплекс-методу складається з кількох ключових етапів, які дозволяють зчитувати симплекс-таблицю, обробляти її, знаходити оптимальне рішення, а потім виводити результати. Давайте розглянемо детально кожен із цих етапів.

**1. RunSimplexMethod\_Click – Обробка події запуску симплекс-методу**

Цей метод запускається, коли користувач натискає на кнопку "Run Simplex Method". Він приймає вхідні дані з InputTextBox та ініціює їх обробку.

* **Вхідні дані**: Користувач вводить симплекс-таблицю в текстове поле, використовуючи формат, де значення в кожному рядку розділені комами.
* **Парсинг таблиці**: Метод викликає функцію ParseTableau для перетворення тексту на числовий масив.
* **Розв'язання симплекс-методом**: Отримана таблиця передається до SolveSimplex для обробки симплекс-алгоритмом.
* **Вивід результату**: Якщо немає помилок, результат відображається у ResultTextBlock.

private void RunSimplexMethod\_Click(object sender, RoutedEventArgs e)

{

try

{

// Введіть симплекс-таблицю в наступному форматі:

// Спочатку кожен рядок матриці, розділений комами і новими рядками.

var tableau = ParseTableau(InputTextBox.Text);

var result = SolveSimplex(tableau);

ResultTextBlock.Text = result;

}

catch (Exception ex)

{

ResultTextBlock.Text = "Error: " + ex.Message;

}

}

**2. ParseTableau – Перетворення вхідних даних у симплекс-таблицю**

Цей метод обробляє вхідні дані, перетворюючи їх на двовимірний масив типу double, що представляє симплекс-таблицю.

* **Розбиття рядків**: Вхідні рядки розбиваються за символом \n, щоб отримати кожен рядок таблиці.
* **Розбиття стовпців**: Кожен рядок розбивається за комами для отримання окремих значень.
* **Заповнення таблиці**: Отримані значення додаються до масиву tableau, який повертається для подальшої обробки.

private double[,] ParseTableau(string input)

{

var rows = input.Trim().Split('\n');

var tableau = new double[rows.Length, rows[0].Split(',').Length];

for (int i = 0; i < rows.Length; i++)

{

var columns = rows[i].Trim().Split(',');

for (int j = 0; j < columns.Length; j++)

{

tableau[i, j] = double.Parse(columns[j]);

}

}

return tableau;

}

**3. SolveSimplex – Реалізація симплекс-алгоритму**

Цей метод обчислює оптимальне рішення симплекс-задачі шляхом ітеративного пошуку опорного рядка та стовпця.

**Ключові етапи роботи симплекс-методу:**

* **Вибір ведучого стовпця**: Перевіряється останній рядок таблиці (рядок Z), щоб знайти найбільш негативний коефіцієнт. Цей стовпець вибирається як ведучий стовпець (pivotCol), що вказує на змінну, яку потрібно ввести до базису.
* **Вибір ведучого рядка**: Для кожного рядка обчислюється співвідношення значення вільного члена до коефіцієнта у ведучому стовпці. Рядок із найменшим співвідношенням обирається як ведучий рядок (pivotRow), який визначає змінну, що буде виведена з базису.
* **Перетворення рядка**: Елементи ведучого рядка нормалізуються до одиниці. Це досягається діленням кожного елемента ведучого рядка на ведучий елемент (pivot).
* **Обнулення стовпця**: Для всіх інших рядків ведучий стовпець обнуляється шляхом віднімання добутку ведучого рядка і відповідного коефіцієнта.
* **Перевірка на оптимальність**: Якщо в останньому рядку (Z) більше немає від'ємних елементів, алгоритм завершує роботу, і таблиця містить оптимальне рішення.

private string SolveSimplex(double[,] tableau)

{

int rows = tableau.GetLength(0);

int cols = tableau.GetLength(1);

while (true)

{

// Шукаємо ведучий стовпець (стовпець з найменшим коефіцієнтом у рядку Z)

int pivotCol = -1;

double minVal = 0;

for (int j = 0; j < cols - 1; j++)

{

if (tableau[rows - 1, j] < minVal)

{

minVal = tableau[rows - 1, j];

pivotCol = j;

}

}

// Якщо немає від'ємних елементів, отримуємо оптимальний розв'язок

if (pivotCol == -1)

break;

// Шукаємо ведучий рядок (рядок з найменшим значенням відношення результату до коефіцієнта)

int pivotRow = -1;

double minRatio = double.PositiveInfinity;

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

{

if (tableau[i, pivotCol] > 0)

{

double ratio = tableau[i, cols - 1] / tableau[i, pivotCol];

if (ratio < minRatio)

{

minRatio = ratio;

pivotRow = i;

}

}

}

if (pivotRow == -1)

throw new Exception("Unbounded solution");

// Нормалізація ведучого елементу до 1

double pivot = tableau[pivotRow, pivotCol];

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

tableau[pivotRow, j] /= pivot;

// Обнулення інших елементів у ведучому стовпці

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

{

if (i != pivotRow)

{

double factor = tableau[i, pivotCol];

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

{

tableau[i, j] -= factor \* tableau[pivotRow, j];

}

}

}

}

// Отримуємо результат

return FormatSolution(tableau);

}

**4. FormatSolution – Форматування результату**

Метод форматує та повертає оптимальне рішення для виведення користувачеві.

* **Перевірка базисних змінних**: Метод перевіряє кожен стовпець таблиці, щоб визначити, чи є змінна базисною. Це робиться шляхом пошуку стовпців, які мають лише один елемент, рівний одиниці, а інші — нулі.
* **Виведення значень змінних**: Якщо змінна є базисною, її значення отримується з вільного члена рядка, у якому вона знаходиться. Небазисним змінним призначається значення 0.
* **Оптимальне значення**: Остаточне значення цільової функції (Z) відображається в останній комірці таблиці.

private string FormatSolution(double[,] tableau)

{

int rows = tableau.GetLength(0);

int cols = tableau.GetLength(1);

string result = "Optimal Solution:\n";

for (int j = 0; j < cols - 1; j++)

{

bool isBasic = true;

double value = 0;

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

{

if (tableau[i, j] == 1)

{

value = tableau[i, cols - 1];

}

else if (tableau[i, j] != 0)

{

isBasic = false;

break;

}

}

if (isBasic)

result += $"x{j + 1} = {value}\n";

else

result += $"x{j + 1} = 0\n";

}

result += $"Optimal Value Z = {tableau[rows - 1, cols - 1]}";

return result;

}

# **РОЗДІЛ III. Функціональний опис та ілюстрація роботи програмного продукту**

Програма "Simplex Method Solver" має інтуїтивно зрозумілий інтерфейс для обробки задач симплекс-методом. Основне завдання інтерфейсу – надати користувачу легкий доступ до всіх основних функцій, таких як введення симплекс-таблиці, запуск симплекс-методу та відображення розв'язку.

**Основні елементи інтерфейсу:**

* **Поле введення симплекс-таблиці**: У верхній частині вікна розташоване велике текстове поле, де користувач може ввести симплекс-таблицю у зручному форматі. Це поле дозволяє вводити дані у форматі, що використовує коми для розділення значень та нові рядки для позначення окремих рядків таблиці.
* **Кнопка запуску симплекс-методу**: Під полем введення знаходиться кнопка з написом "Run Simplex Method", яка активує розрахунок симплекс-методом. Кнопка підкреслена яскравим кольором, що робить її помітною та привертає увагу користувача до основної дії.
* **Область для виведення результату**: У нижній частині вікна розміщене текстове поле для виведення результатів. Після виконання обчислень тут відображається оптимальне рішення або повідомлення про помилку, якщо щось пішло не так.

**Додаткові елементи інтерфейсу:**

* **Меню**: У верхній частині вікна знаходиться меню з пунктами "File", "Task", та "Help". Це дозволяє користувачу отримати доступ до додаткових функцій або інформації, таких як збереження та завантаження симплекс-таблиць, а також доступ до довідкових матеріалів щодо роботи з програмою.

Для наочності наведено скріншот загального вигляду інтерфейсу програми, де можна побачити всі основні елементи.

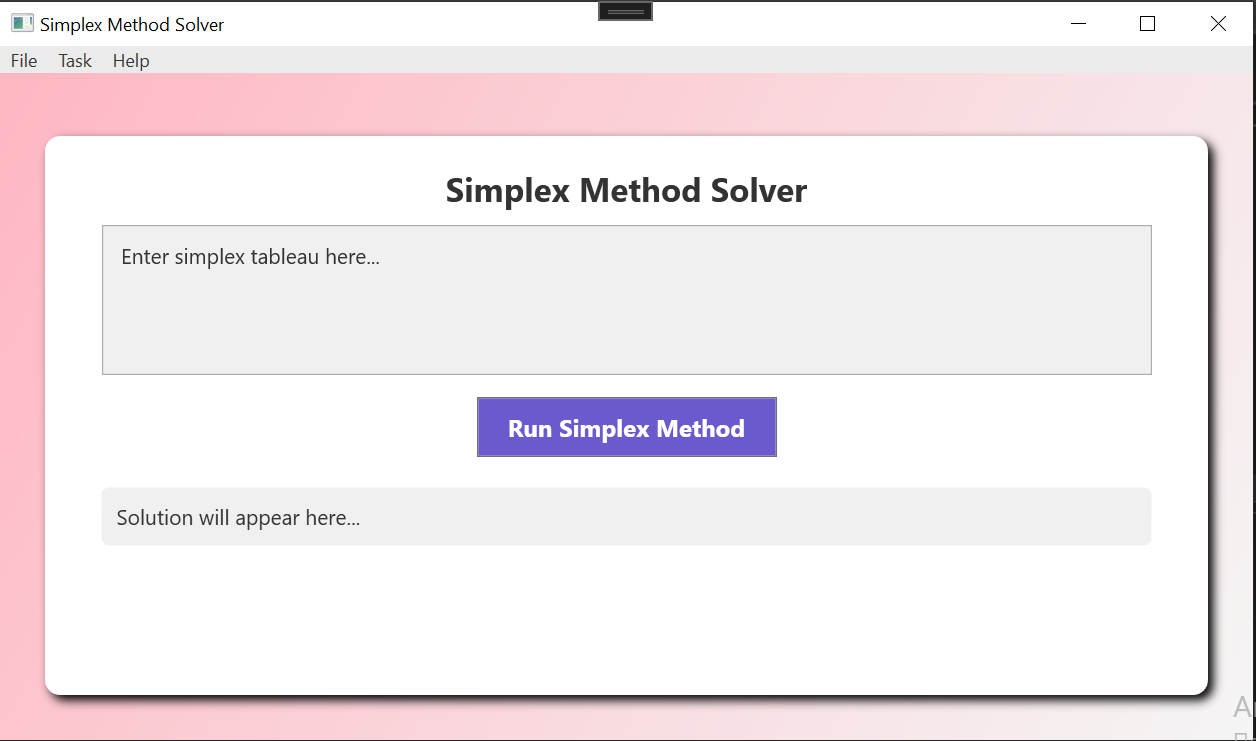


Рис. 1 Інтерфейс програми

**Введення задачі**

Для вирішення задачі симплекс-методом користувач має ввести симплекс-таблицю, де кожен рядок представляє один обмежувальний рядок, а значення розділені комами. Це дозволяє легко та швидко вводити умову задачі.

Користувач заповнює поле введення у відповідному форматі: кожен рядок має містити числові значення, розділені комами, а новий рядок представляє окремий рядок симплекс-таблиці. Після заповнення всіх даних користувач натискає кнопку "Run Simplex Method". Програма автоматично перевіряє коректність введених даних — допускаються лише числові значення, що запобігає можливим помилкам у розрахунках.

Нижче наведено фрагмент коду, який відповідає за зчитування введеної симплекс-таблиці та її обробку:

var tableau = ParseTableau(InputTextBox.Text);

var result = SolveSimplex(tableau);

ResultTextBlock.Text = result;

Код виконує обробку введених даних та передає їх у функцію розв'язку симплекс-методом. Програма аналізує симплекс-таблицю та виводить оптимальне рішення, яке з'являється в області результатів під кнопкою запуску.

Після успішного розрахунку користувач отримує повідомлення з оптимальним розв'язком, що дозволяє переглянути результат розрахунків або змінити умови задачі за потреби. Інтерфейс спрощує процес введення та забезпечує зручність використання навіть для користувачів, які не знайомі з технічними деталями симплекс-методу.

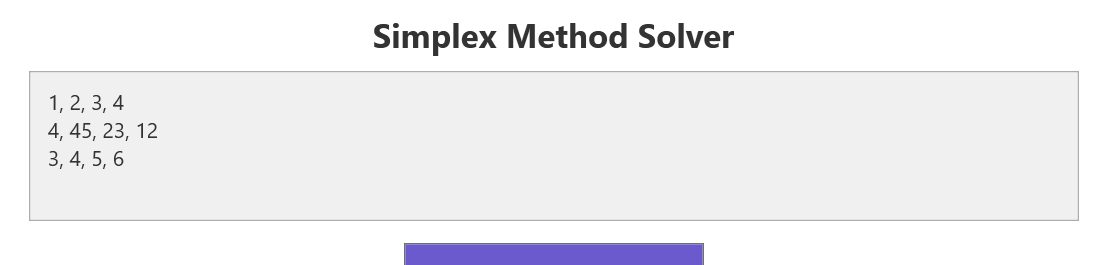


Рис. 1.1 Блок для введеня умови задачі

**Розв’язок задачі**

Після того, як умова задачі успішно введена та розв'язана симплекс-методом, користувач має можливість переглянути результат обчислень. Для цього в інтерфейсі програми передбачена область виведення результату, яка розташована під кнопкою "Run Simplex Method" і відображає оптимальний розв'язок задачі.

Після натискання кнопки "Run Simplex Method" програма обробляє введену симплекс-таблицю, виконуючи необхідні обчислення для пошуку оптимального розв'язку. В кінці обчислень програма виводить результат в область для відображення тексту, де користувач може переглянути оптимальне значення цільової функції та значення змінних.

Фрагмент коду, який відповідає за обчислення та відображення розв'язку, виглядає так:

var tableau = ParseTableau(InputTextBox.Text);

var result = SolveSimplex(tableau);

ResultTextBlock.Text = result;

Програма отримує дані з текстового поля, де введена симплекс-таблиця, виконує обчислення в методі SolveSimplex, а потім виводить результат в ResultTextBlock. Оптимальний розв'язок включає значення змінних та оптимальне значення цільової функції, що дозволяє користувачеві бачити підсумковий результат обчислень.

Для зручності роботи з програмою інтерфейс дозволяє переглядати розв'язок безпосередньо після виконання обчислень, що спрощує процес аналізу результатів.

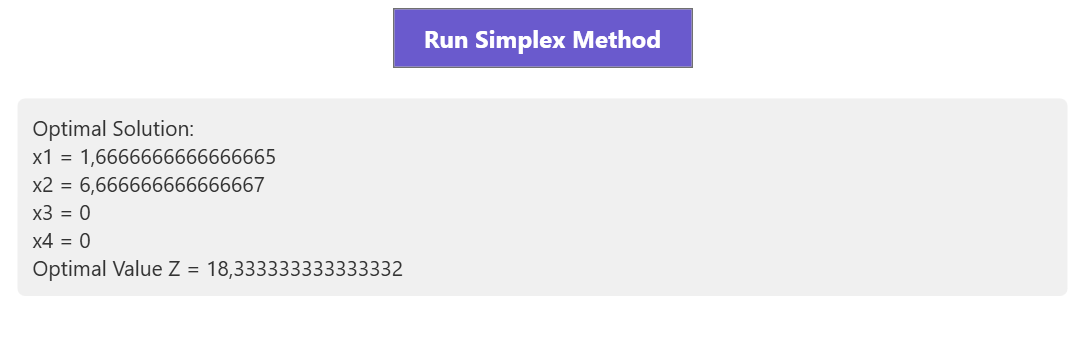


Рис 1.2 Блок виведення результату

**Обробка помилок**

Однією з важливих функцій програми є обробка помилок, які можуть виникати під час розв'язання задачі симплекс-методом. Програма реалізує кілька механізмів для забезпечення надійності та коректності обробки вхідних даних.

**Перевірка коректності введення даних:**

Програма забезпечує валідацію введеної симплекс-таблиці. Якщо користувач вводить щось, окрім числових значень або неправильний формат таблиці, програма зупиняє обчислення і виводить повідомлення про помилку. Це дозволяє уникнути помилок через некоректне введення. Код для перевірки може виглядати наступним чином:

private bool IsValidTableau(string input)

{

// Розділимо введені дані на рядки

var rows = input.Split(new[] { '\n', '\r' }, StringSplitOptions.RemoveEmptyEntries);

if (rows.Length == 0)

{

MessageBox.Show("Помилка: Введіть дані для симплекс-таблиці.");

return false;

}

// Перевіримо, чи кожен рядок містить лише числа і має однакову кількість елементів

int columnCount = rows[0].Split(' ', StringSplitOptions.RemoveEmptyEntries).Length;

foreach (var row in rows)

{

var elements = row.Split(' ', StringSplitOptions.RemoveEmptyEntries);

if (elements.Length != columnCount)

{

MessageBox.Show("Помилка: Усі рядки повинні мати однакову кількість елементів.");

return false;

}

foreach (var element in elements)

{

if (!double.TryParse(element, out \_))

{

MessageBox.Show("Помилка: Усі елементи мають бути числовими значеннями.");

return false;

}

}

}

return true;

}

**Повідомлення про помилку під час розв'язання задачі:**

Якщо під час обчислень програма виявляє, що задача не має розв'язку (наприклад, невиконання умов оптимальності), вона повідомляє користувача про проблему, щоб уникнути неправильних результатів. Це допомагає користувачу зрозуміти, що умови задачі, можливо, слід переглянути.

**Обробка помилок під час обчислення:**

Якщо виникає внутрішня помилка під час розрахунків (наприклад, ділення на нуль у таблиці або інша математична помилка), програма виводить повідомлення про технічну помилку і зупиняє обчислення. Це дозволяє уникнути некоректних результатів через несправності під час виконання.

# **Висновок**

У процесі виконання курсової роботи було розроблено та реалізовано програму для розв’язку задач лінійного програмування за допомогою симплекс-методу. Основною метою проекту було створення зручного, інтуїтивно зрозумілого інтерфейсу, який дозволяє користувачам вводити симплекс-таблицю, перевіряти правильність введених даних та автоматично знаходити оптимальне розв’язання задачі.

Під час роботи було досягнуто таких результатів:

1. Розроблено інтерфейс програми на основі технологій WPF, який забезпечує зручний доступ до всіх необхідних функцій для роботи з симплекс-таблицею.
2. Реалізовано механізм обробки та валідації введених даних, який забезпечує коректне введення та запобігає помилкам у форматі вхідної інформації.
3. Створено основні алгоритми для обчислення симплекс-методу, включно з обробкою початкового введення, визначенням базисних змінних, виконанням кроків симплекс-методу та обробкою отриманих результатів.
4. Проведено тестування програми, яке продемонструвало її працездатність та ефективність при розв’язанні задач різного рівня складності.

Розроблена програма може бути корисною для студентів, викладачів та спеціалістів, які вивчають або використовують методи лінійного програмування. Вона дозволяє не лише автоматизувати процес розв’язання задачі, а й надає можливість краще зрозуміти сам алгоритм симплекс-методу через покроковий розв’язок.

У майбутньому можливим напрямком розвитку даного проекту є інтеграція додаткових методів лінійного програмування, а також удосконалення інтерфейсу для покращення користувацького досвіду. Додатково можна розглянути можливість візуалізації проміжних етапів симплекс-методу, що допоможе зробити програму більш інформативною та корисною для навчальних цілей.

Загалом, курсова робота виконана успішно, оскільки поставлені цілі були досягнуті, а реалізована програма відповідає функціональним вимогам та дозволяє ефективно розв'язувати задачі лінійного програмування за допомогою симплекс-методу.

# **ЛІТЕРАТУРА**

1. **Глушков В.М., Яблонський С.В.** Теоретичні основи лінійного програмування. – Київ: Наукова думка, 1973. – 256 с.
2. **Гаврилюк В.В.** Методи оптимізації: Навчальний посібник. – Київ: Видавництво НТУУ «КПІ», 2007. – 168 с.
3. **Корнейчук Б.Б.** Математичне програмування: підручник. – Київ: КНТЕУ, 2009. – 304 с.
4. **Winston W.L.** Operations Research: Applications and Algorithms. – Belmont, CA: Duxbury Press, 2004. – 1440 p.
5. **Максименко Г.М.** Лінійне програмування та економічна оптимізація. – Львів: ЛНУ імені Івана Франка, 2010. – 295 с.
6. **Richter J.** Windows Presentation Foundation Unleashed. – Sams Publishing, 2006. – 576 p.
7. **Troelsen A.** Pro C# 7: With .NET and .NET Core. – Apress, 2017. – 1372 p.
8. **Алгоритми симплекс-методу** [Електронний ресурс]. – Режим доступу: https://example.com/simplex-algorithm.
9. **Офіційна документація WPF** [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/>.

# **ДОДАТОК А**

КАЛЕНДАРНИЙ ПЛАН

на курсову роботу з дисципліни «Інструментальні засоби візуального програмування» студента групи 41-іП

*Мельник Владислав Олександрович*

(прізвище, ім’я, по батькові студента)

**Тема: Програма розв’язку «Задач лінійного програмування»**

**Завдання:** *розробити програму для розв’язку задач лінійного програмування.*

**Зміст завдання та календарний план його виконання**

|  |  |  |
| --- | --- | --- |
| № з/п | Зміст завдання | Дата |
| 1 | Постановка задачі, визначення вимог, структури даних, методів рішення тощо | 05.09.24-20.09.24 |
| 2 | Розробка алгоритму, визначення форми представлення даних, структури програми | 29.09.24-05.10.24 |
| 3 | Програмування. Розробка документації. Ілюстрація роботи програми | 11.10.24-17.10.24 |
| 4 | Оформлення пояснювальної записки | 21.10.24-28.10.24 |
| 5 | Представлення роботи до попереднього розгляду керівником | 28.10.24 |

Завдання прийнято до виконання «03» вересня 2024 р.

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

(підпис та ПІБ студента)

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Крупеня В. О.

(Прізвище І.П.)

# **ДОДАТОК Б**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**ВІДОКРЕМНЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ**

**« РІВНЕНСЬКИЙ ФАХОВИЙ КОЛЕДЖ НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ БІОРЕСУРСІВ ТА ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ »**

**ЗАТВЕРДЖУЮ**

Керівник курсової роботи

                             Крупеня В. О.

«      »                             20          р.

**ТЕХНІЧНЕ ЗАВДАННЯ**

на розробку програмного продукту «Simplex Method Solver»

до курсової роботи на тему **Програма розв’язку «Задач лінійного програмування»**

Спеціальність 121 Інженерія програмного забезпечення

Галузь знань 12 Інформаційні технології

Виконав

Студент 41-ІП групи

На здобуття освітнього-

кваліфікаційного рівня

«молодший спеціаліст»

Мельник Владислав Олександрович

м. Рівне- 20 24рік

**ЗМІСТ РОЗДІЛІВ**

**ВСТУП**

**РОЗДІЛ І.** **НАЙМЕНУВАННЯ Й ОБЛАСТЬ ЗАСТОСУВАННЯ.**

Програма розв’язку «Задач лінійного програмування»

**РОЗДІЛ ІІ. ПІДСТАВА ДЛЯ РОЗРОБКИ.**

Розробка виконана на основі індивідуального завдання отриманого на курсову роботу з дисципліни “Інструментальні засоби візуального програмування”

**РОЗДІЛ ІІІ. ПРИЗНАЧЕННЯ РОЗРОБКИ.**

Програма призначена для розв’язування задач лінійного програмування та забезпечення зручного доступу до даних.

**РОЗДІЛ IV. ТЕХНІЧНІ ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ.**

**4.1. Вимоги до функціональних характеристик**

Програма має забезпечувати введення, редагування, перегляд і видалення умови задачі, а також її розв’язок.

**4.2.** **Вимоги до надійності**

Забезпечується стійке функціонування з обробкою помилок введення та повідомленнями користувачу.

**4.3. Умови експлуатації**

Програма призначена для роботи на персональних комп'ютерах під управлінням операційної системи Windows

**4.4. Вимоги до складу й параметрів технічних засобів**

|  |  |
| --- | --- |
| **Найменування** | **Характеристики** |
| Процесок | *Intel Core i3 або еквівален* |
| Оперативна пам’ять | *4 ГБ* |
| Операційна система | *Windows 7 або новіша версія* |
| Монітор | *Мінімальна роздільна здатність 1024x768* |
| Місткість жорсткого диска | *Не менш як 500 МБ* |

**4.5. Вимоги до інформаційної й програмної сумісності** зазначають вимоги до інформаційних структур на вході й виході, до методів рішення, кодів програм, мов програмування. При необхідності повинен забезпечуватися захист інформації й програм.

**РОЗДІЛ V. СТАДІЇ Й ЕТАПИ РОЗРОБКИ.**

|  |  |  |
| --- | --- | --- |
| Стадії розробки | Етапи робіт | Зміст робіт |
| Технічне завдання  04.10.24 | Обґрунтування необхідності | Визначення цілей програми та її основних функцій |
| Затвердження технічного завдання  06.10.24 |  | Погодження основних вимог до програмного продукту |
| Тестовий проєкт №1  10.10.24 - 12.10.24 | Розробка | Створення першого прототипу програми |
| Затвердження тестового проєкту №1 |  | Перевірка та затвердження першої версії |
| Тестовий проєкт №2  13.10.24 - 16.10 .24 | Розробка | Внесення правок та покращень |
| Затвердження тестового проєкту №2 |  | Погодження остаточної тестової версії |
| Робочий проєкт  17.10.24 - 26.10.24 | Розробка програми | Створення остаточної версії програми |
| Розробка програмної документації |  | Підготовка супровідної документації |
| Випробування програми |  | Тестування та усунення помилок |
| Впровадження  27.10.24 - 30. 10. 24 |  | Запуск програми |

**РОЗДІЛ VІ. ПОРЯДОК КОНТРОЛЮ Й ПРИЙМАННЯ.**

Програма підлягає функціональному та тестовому випробуванню для перевірки коректності роботи.

# **ДОДАТОК В**