ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» МОВОЮ ПРОГРАМУВАННЯ C#**»

Виконав: студентка 4 курсу, групи ПД–41
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Шилкіна А.О.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
"____" _____2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

ШИЛКІНІЙ АННІ ОЛЕКСІЇВНІ
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розроблення програмного забезпечення для вивчення дисципліни "Дослідження операцій" мовою програмування C#»

Керівник роботи: Жебка В.В., к.т.н., доцент кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «12» березня 2021 року № 65.
2. Строк подання студентом роботи « 01 » червня 2021 року
3. Вхідні дані до роботи
3.1 Методи лінійного програмування
3.2 Науково-технічна література з питань, пов'язаних щодо створення Windows Form (інтерфейс програмування додатків)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1 Вивчення дисципліни «Дослідження операцій» за допомогою методів лінійного програмування
4.3 Розробка програмного забезпечення для вивчення дисципліни «Дослідження

операцій»
5. Перелік демонстраційного матеріалу (назва основних слайдів)
5.1 Актуальність проблеми. Мета, об’єкт та предмет дослідження
5.2 Методи лінійного програмування
5.3 Технічні завдання
5.4 Програмні засоби реалізації
5.5 Методи та класи програми
5.6 Принцип роботи розробленого програмного забезпечення

Дата видачі завдання «19» квітня 2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Обговорення напрямку дослідження та обрання теми	26.02.21	Виконано
2	Підбір науково-технічної літератури	14.03.21	Виконано
3	Вимоги до системи	20.03.21	Виконано
4	Створення графічного методу лінійного програмування	26.03.21	Виконано
5	Створення симплекс методу лінійного програмування	01.04.21	Виконано
6	Створення двоїстої задачі лінійного програмування	16.04.21	Виконано
7	Створення двоїстого симплекс методу лінійного програмування	23.04.21	Виконано
8	Вступ, висновки, реферат	26.04.21	Виконано
9	Розробка обов’язкових демонстраційних матеріалів	26.04.21	Виконано
10	Попередній захист роботи		
11	Здача роботи		

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Загальний обсяг роботи : 87с., 24 рис., 13 табл., 22 джерел.

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» МОВОЮ ПРОГРАМУВАННЯ C#

Об'єктом досліджень - викладання дисципліни «Дослідження операцій» у вищій школі за допомогою математичних моделей виробничих систем та застосуванню практичних методів для їх оптимального управління.

Предметом дослідження – програмне забезпечення для вивчення дисципліни «Дослідження операцій».

Метою дослідження - визначення новітніх підходів до методів викладання «Дослідження операцій» для майбутніх програмістів, формування ролі знань і вмінь з дослідження операцій у підготовці фахівців з комп'ютерних наук за допомогою розробленого програмного забезпечення.

У роботі проведено огляд, таких *методів* лінійного програмування як графічний метод, симплекс метод, двоїста задача та двоїстий симплекс метод.

Особливістю програми є виконання алгоритму дій різних методів з можливістю поетапно здійснювати перегляд ходу рішень задач лінійного програмування. Загрузка задачі та зберігання файлу в текстовому форматі, а саму «.txt» та «.csv».

Додаток був реалізований в інтегрованому середовищі розробки Microsoft Visual Studio, використовуючи інтерфейсове програмування додатків Windows Form, мовою програмування C#.

Додаток працює в операційній системі Windows.

Було проведено опис архітектури та основні принципи розробки.

У якості вихідних даних є результат в додатку, а також файл з текстовими даними.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ДО - дослідження операцій

ВНЗ –вищий навчальний заклад

ЗЛП- задача лінійного програмування

ОДР – область допустимих рішень

ПЗ – програмне забезпечення

ДСМ - двоїстий симплекс-метод

МКЗЛП - майже канонічна задача лінійного програмування

МДБР - майже допустимий базисний розв'язок

UML - Unified Modeling Language

ВСТУП

Сьогодні українська освіта повинна відповідати сучасним запитам з боку особистості та суспільства, потребам економіки та світовим тенденціям. Перетворити українську освіту на інноваційне середовище є першочерговим завданням сьогодення.

Актуальність теми. Кардинальні, різнорівневі, глобальні, трансформаційні зміни у суспільстві, зумовлені швидкою еволюцією цифрових технологій, комп'ютерних мереж зумовлюють новий підхід галузі освіти й суттєве підвищення вимог до якості знань у молодих фахівців. Використання сучасних технологій в освіті має відігравати ключову роль у створенні необхідних умов для саморозвитку всіх суб'єктів навчальної діяльності, активізації когнітивних та творчих процесів, формуванню необхідних компетентностей, сприяти утвердженню гуманістичних ідей. Саме фахівці із інженерії програмного забезпечення та комп'ютерних наук, що мають знання та навички з проектування, моделювання та аналізу програмного забезпечення зможуть здійснити кардинальний переворот у розвитку економіки, науки, освіти та інших галузях життєдіяльності суспільства. Галузь комп'ютерних наук характеризується постійним розвитком та вдосконаленням багатьох методологій, що містять сукупність моделей, методів програмного забезпечення. А спрямованість сучасних розробок на підтримку інтернету речей потребує більш глибоких знань в галузі програмного забезпечення вбудованих систем. Вимоги ринку та особливості проектів з вбудованими системами обумовлюють використання відкритого програмного забезпечення, при цьому важливо розуміти як це програмне забезпечення можливо використовувати відповідно до ліцензій під якими воно розповсюджуються.

Значну роль у формуванні спеціалістів ІТ-технологій відіграє вивчення науки математики, а саме прикладної математики. Велику увагу приділяється новому класу задач оптимізації, суть якого полягає в знаходженні у деякої області

точок найбільшого або найменшого значення функціоналу, який залежить від великої кількості змінних. Це так звані задачі математичного програмування, які виникають в різних областях людської діяльності, зокрема в економічних дослідженнях, у практиці планування та організації виробництва, в багатьох технологічних проблемах.

Математичне програмування належить до числа дисциплін прикладної математики, що найбільш інтенсивно використовуються дисципліною « Дослідження операцій» вищих учбових закладів. В рамках методів оптимізації розглядаються лінійне програмування. Важливим розділом нелінійного програмування є опукле програмування. Багато проблем, пов'язаних з оптимізацією до задач оптимального керування.

Щоб підвищити рівень знань з напрямів «Програмна інженерія» та «Комп'ютерні науки та інформаційні технології» я хочу запропонувати нові підходи до вивчення дисципліни «Дослідження операцій». Будь – яке програмне забезпечення створюється за допомогою різних мов програмування. Одним з таких є C# (С Шарп) - це нова мова, розроблена в корпорації Microsoft в якості основного середовища розробки для . NET Framework і всіх майбутніх продуктів Microsoft. C # бере свій початок в інших мовах, в основному, C++, Delphi і Smalltalk.

Пошуку шляхів вирішення зазначених проблем приділяють значну увагу фахівці в галузі автоматизації, створення й впровадження інформаційно-комунікаційних технологій в освіті, педагогіки й психології, теорії і методики навчання інформатики тощо. Розгляд комплексу питань, пов'язаних з використанням сучасних інформаційно-комунікаційних технологій в освітньому процесі в вищій школі, започатковано в роботах Р. Вільямса, К. Макліна, А. П. Єршова, М. І. Жалдака, Е. І. Кузнєцова, О. А. Кузнєцова, В. М. Монахова, Ю. С. Рамського та інших дослідників. Проблеми створення і впровадження комп'ютерно орієнтованих методичних систем навчання математичних, інформатичних, фізичних дисциплін у закладах вищої освіти досліджували В. Ю.

Биков, Гриценко В. Г., Гладка Л. І., Кріковцов С. В. , В. І. Клочко, Н. В. Морзе, Ю. В. Горошко, Ю. Г. Лотюк, С. А. Раков, С. О. Семеріков, В. П. Сергієнко, О. В. Співаковський, О. О. Спірін, Ю. В. Триус, І. М. Цідило, Л. О. Черних, С. М. Яшанов та інші. Теоретико-методичні засади проектування, формування і використання освітніх середовищ розробляли О. М. Алєксєєв, Т. А. Вакалюк, О. Г. Глазунова, О. Г. Колгатін, К. Р. Колос, С. Г. Литвинова, В. В. Осадчий, Л. Ф. Панченко, М. П. Шишкіна та інші.

Метою дослідження є визначення новітніх підходів до методів викладання «Дослідження операцій» для майбутніх програмістів, формування ролі знань і вмінь з дослідження операцій у підготовці фахівців з комп'ютерних наук за допомогою розробленого програмного забезпечення.

Предметом дослідження є програмне забезпечення для вивчення дисципліни «Дослідження операцій».

Об'єктом досліджень є викладання дисципліни «Дослідження операцій» у вищій школі за допомогою математичних моделей виробничих систем та застосуванню практичних методів для їх оптимального управління.

Методи дослідження:

- теоретичні методи (аналіз психолого-педагогічної, методичної й спеціальної літератури; аналіз програм, підручників, навчальних посібників з вищої математики та інформатики для ВНЗ, ресурсів Інтернет – для визначення теоретичних засад дослідження);
- емпіричні методи (спостереження, бесіди з викладачами математичних дисциплін);
- статистичні методи (опрацювання й аналіз результатів проведеного експерименту);
- метод системного аналізу;
- методи евристичних алгоритмів;
- методи дискретної математики;
- методи дослідження операцій;

- методи комбінаторної оптимізації.

Завдання дослідження:

- проаналізувати предметну область дисципліни «Дослідження операцій»;
- здійснити огляд методів лінійного програмування;
- проаналізувати інтегроване середовище розробки Microsoft Visual Studio;
- розробити основні компоненти методики програмного забезпечення для вивчення дисципліни « Дослідження операцій»;
- здійснити тестування програмного забезпечення;
- скласти інструкцію роботи з програмою.

Наукова новизна в роботі є розробка концепції щодо викладення дисципліни «Дослідження операцій» за допомогою методів теорії дослідження операцій, методів розв'язування екстремальних задач функцій однієї та багатьох змінних, основ лінійного та нелінійного програмування на основі розробленого програмного забезпечення.

Практична значущість результатів отриманих результатів дослідження полягає в створені програмного забезпечення для використання методів лінійного програмування у вивчені дисципліни «Дослідження операцій».

Структура роботи складається із переліку умовних скорочень, вступу, трьох розділів з підрозділами, висновку, переліку посилань, додатків та демонстраційного матеріалу (презентація).

1 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» ЗА ДОПОМОГОЮ МЕТОДІВ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Прогресивний розвиток інформаційних технологій, сучасної техніки, впровадження автоматизації в різні галузі життя та діяльності суспільства – все це приводить до розробки та впровадження в наше життя новітніх програм та технологій з боку науковців та фахівців інформаційних технологій. Галузь комп'ютерних наук характеризується постійним розвитком та вдосконаленням багатьох методологій, що містять сукупність моделей, методів програмного забезпечення. Пошук найкращий рішень передбачає побудову математичної моделі і використання для її аналізу певного математичного апарату. Для цього застосовують спеціальні наукові методи, що об'єднані спільною назвою «дослідження операцій».

1.1 Основні поняття і визначення. Методика проведення дослідження операцій

Відомий науковець в галузі інформаційних технологій Лавров Є.А. дає визначення понятійному апарату « Дослідження операцій ».

Дослідження операцій (ДО) – це використання математичних, кількісних методів для обґрунтування рішень у всіх сферах цілеспрямованої людської діяльності [10].

Дослідження операцій – це застосування та використання різних математичних методів для моделювання систем та аналізу, надання характеристик.

Операція – це дія або сукупність дій, підпорядкованих єдиному задуму та спрямованих на досягнення певної мети, яка має характер повторюваності, тобто багаторазовості [10].

Операція складається з цілеспрямованості і повторюваності. Якщо мета не визначена, не буде існувати й операції. Визначеність мети дає можливість пошуку різних шляхів для її досягнення і можливість обрання найкращого. Повторюваність припускає встановлення закономірностей. Отже, виникає можливість проводити дослідження, які стосуються різних сторін операції. Операції можна вивчати в процесі дослідження як оригіналів, так і моделей операцій.

Отже, дослідити операцію – це знайти оптимальне, найкраще рішення за наявності обмежень технічного, економічного, характеру тощо. Оптимальними є ті рішення, які кращі за інші.

Отже, мета дослідження операцій – це є попереднє кількісне обґрунтування оптимальних та найкращих рішень.

Якщо на практиці застосовувати кількісні методи дослідження, то необхідно побудувати математичну модель операції.

Математична модель – система математичних виразів, що описують характеристики об’єкта моделювання та взаємозв’язки між ними [10].

Існують моделі **детерміновані, ймовірнісні та ігрові.**

Математичні моделі бувають складними, вони записуються формальною (математичною) мовою, їх досліджують за допомогою математичних методів та комп’ютерів.

Не є універсальними методика і провідні принципи дослідження операцій. Свої особливості має кожне дослідження і потребує від спеціаліста, який робить дослідження інтуїції та уявлення, щоб правильно визначити цілі та досягти відмінних результатів в дослідженні.

Складом методики дослідження операцій є такі заходи:

1) Визначення цілей.

2) Складання плану розробки проекту, операції.

3) Формулювання проблем.

4) Побудова моделі.

5) Розробка обчислювального методу.

6) Розробка технічного завдання на програмування, програмування та відлагодження програми.

7) Збір даних.

8) Перевірка моделі.

9) Реалізація результатів [12].

Етапи виконання розробки програмного забезпечення для вивчення дослідження операцій зображено на рис. 1.1.

Рисунок 1.1 – Етапи виконання ПЗ

1) Першочергова ціль будь-якого дослідження операцій полягає в тому, щоб з'ясувати, що очікує отримати керівник операції в результаті її проведення, тобто,

які передбачувані результати проведення операції можна очікувати. Цілі дослідження треба формулювати, виходячи з суті рішення або рішення, на яке орієнтована дана робота. Цілі не треба формулювати ані занадто вузько, ані занадто широко. Неправильне і неточне формулювання цілей може призвести дослідників до розв'язання невірно поставленої задачі.

2) Другий етап дослідження полягає у складанні плану виконання проекту операції, тобто установленню необхідних термінів завершення певних видів робіт. Це – одна з форм контролю за ходом розробки проекту. Як документ, план розробки проекту операції являє собою календарний графік виконання його етапів. Етапи можуть деталізуватися до рівня окремих завдань. Наприклад, етап розробки обчислювального методу може мати такі завдання: розробку методу розв'язання для кожної підмоделі задачі; опис і документальне оформлення методів розв'язання; перевірку запропонованих методів на вибраних задачах невеликої розмірності; внесення уточнень та змін щодо методів розв'язання на підставі результатів пробних розрахунків тощо. При складанні плану треба також приділяти увагу розподілу робіт між окремими виконавцями.

3) Формулювання проблеми – наступний етап дослідження. Він містить не тільки обговорення з керівником операції цілей дослідження, а й збір даних, які надають можливість уявити суть проблеми, що мало місце у минулому, чого треба очікувати в майбутньому, який характер співвідношень між змінними досліджуваної задачі. На основі цих результатів формулюється загальна схема побудови моделі і визначаються напрям усієї наступної роботи.

Одним із питань, які пов'язані з формулюванням проблеми, є визначення того, чи можна усю проблему представити у вигляді окремих підпроблем, щоб паралельно або послідовно дослідити їх незалежно одна від одної (декомпозиція). Друге питання пов'язане з визначенням ступені деталізації моделі, що розробляється. Останнє залежить від обсягів виділених коштів, календарного плану розробки проекту, цілі дослідження. Наступна фаза стосується галузі застосування та розмірності моделі, що розробляється, визначенню керованих і

некерованих змінних, технологічних параметрів операції, показників ефективності, які нададуть змогу оцінити конкретні розв'язки розглянутої проблеми.

4) Четвертий етап дослідження пов'язаний із побудовою моделі. Вона відображає взаємозв'язок між керованими змінними, некерованими змінними, технологічними параметрами і показниками ефективності. Правильно побудована модель – основна умова успішної розробки проекту операції. Приступаючи до розробки моделі, насамперед, треба з'ясувати питання про можливість використання тих чи інших показників і співвідношень у рамках моделі. Існує декілька різних типів співвідношень, які формують модель: співвідношення, які виходять із визначень, емпіричні співвідношення, нормативні співвідношення. Крім того, треба зібрати та ретельно проаналізувати великий об'єм даних. На кінцевому етапі побудови моделі досліднику треба дати точне аналітичне або алгоритмічне формулювання досліджуваної проблеми.

5) Разом з роботою з побудови моделі необхідно вибрати або розробити чисельний метод розв'язання. Для цього треба з'ясувати такі моменти: − чи треба використовувати імітаційне моделювання або будь-який із методів оптимізації: − чи повинна модель враховувати випадковий характер деяких змінних або ж достатньо використовувати детермінований підхід; − чи треба враховувати нелінійність певних співвідношень, чи достатньо обмежитися їх лінійною апроксимацією; − чи можна використовувати існуючі методи розв'язання, або треба розробити новий метод.

Отже, необхідно з'ясувати, які треба зробити припущення та який метод розробити, щоб застосування моделі було практично виправданим у відношенні до використовуваних обчислювальних процедур.

6) Створення програм для ЕОМ у багатьох випадках є складовою частиною дослідження. Розробка технічного завдання на програмування повинна виконуватися ретельно, що дасть змогу забезпечити більш якісне документальне оформлення програм, результатом чого дослідження стає значною мірою

орієнтованим на користувача, на задоволення його потреб. Треба звернути увагу на одну з робіт, які проводяться на цьому етапі, складання вхідних форм та вихідних (документів), їх обговорення та узгодження з керівництвом та виконуючим персоналом. Вхідні форми – бази даних, які надають можливість забезпечити користувача інформацією, що швидко підготовлюється і легко оновлюється. Наглядні вихідні форми повинні дати користувачу зрозумілу, добре підібрану й зручно розташовану інформацію. Що стосується саме програмування і відлагодження, то в багатьох випадках проект з дослідження операцій не потребує розробки машинних програм, а за потреби завжди можна використати існуючі програми.

7) На цьому етапі здійснюється збір та аналіз даних, які є необхідними для перевірки правильності моделі та практичного застосування результатів дослідження операцій, тоді як на попередніх етапах збір даних переслідував цілі, що були пов'язані, перш за все, з формулюванням проблеми та побудовою моделі. Тому проблема відсутності даних не є перешкодою до виконання продуктивних операційних досліджень, оскільки математична модель є засобом, який дозволяє обійти труднощі отримання відповідних оцінок шляхом зведення їх до більш простих вимірювань. Використання моделей, які розробляються при дослідженні операцій, допомагає в процесі прийняття рішень. Розв'язок проблеми зводиться до найпростіших вимірювань, встановленню вихідних змінних і показників ефективності, які є функціями цих змінних.

8) Етап перевірки моделі містить дві фази: визначення способів перевірки і здійснення самої перевірки. На першій – вибираються аналітичні й експериментальні методи перевірки несуперечності, чутливості, реалістичності та роботоздатності моделі. Для здійснення перевірки моделі будуть необхідні дані, які отримані на попередньому стані. Результати цієї роботи можуть призвести до необхідності перебудови моделі та, відповідно, до складання нових програм.

9) Отримані результати дослідження операцій треба представити рядом робочих процедур, які можна легко зрозуміти і застосувати діючій стороні. Це

найважливіший етап, яким завершується операційне дослідження, його можна розглядати як самостійну задачу [12].

1.2 Типові класи задач дослідження операцій

Дослідження операцій поділяється на типові ряд класів за змістом та постановками множини. Якщо застосувати задачі відповідних класів, то вони «виходять» одна з одної відповідно до того, як відбувається уявлення про досліджувану операцію (Рис. 1.2).

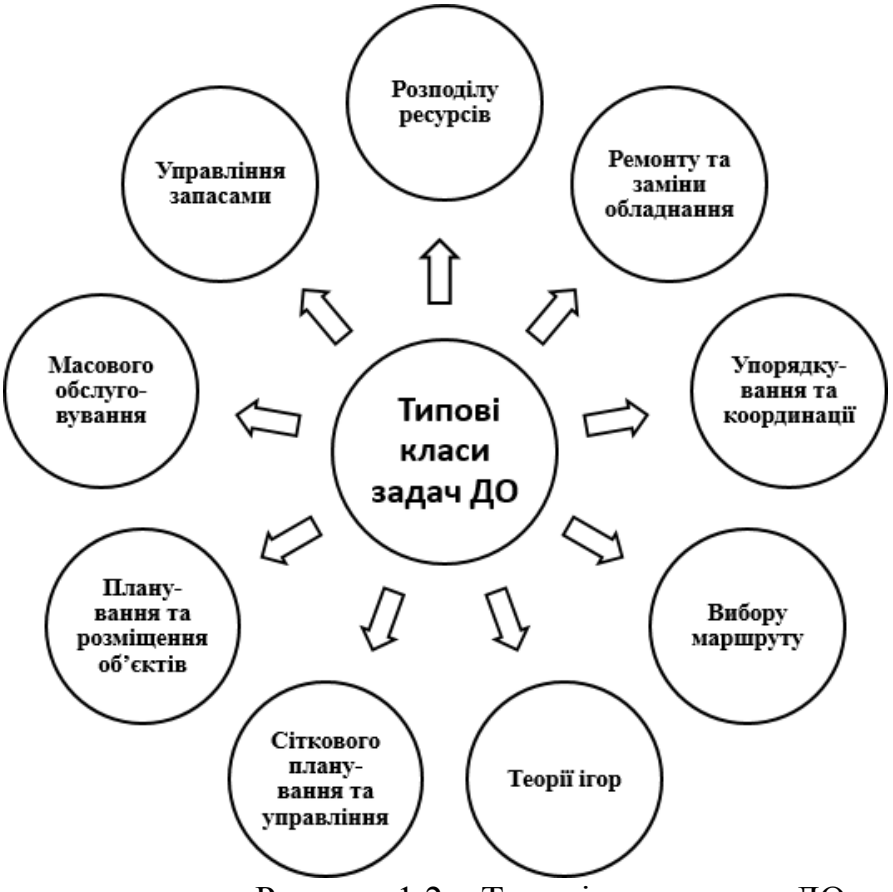


Рисунок 1.2 – Типові класи задач ДО

Задачі управління запасами – один із найпоширеніших і добре вивчених класів задач дослідження операцій. Вони виникають у випадках, коли необхідно

визначити оптимальну кількість запасів. Зі збільшенням запасів збільшуються і витрати на їх зберігання, але зменшуються втрати від їх можливої нестачі [12].

В функції величини наявні як запаси, так і витрати на зберігання, також наявні витрати на доставку ресурсів. Потрібно знайти оптимальне значення постачання, їх частоту і терміни надходження ресурсів, щоб витрати в цілому були мінімальними. Критерієм оптимальності (максимум чи мінімум) є сумарна сума зберігання та постачання ресурсів.

Класифікація задач управління запасами:

- за кількістю періодів управління (поповнення запасів) – на одно періодні та багато періодні;
- за характером поповнення запасів – із неперервною системою поповнення запасів (миттєве) і періодичне;
- за урахуванням попиту – на детерміновані та ймовірнісні (статичні);
- за кількістю типових ресурсів – на одно продуктові і багато продуктові;
- за видом цільової функції – на задачі з пропорційними та непропорційними витратами.

Задачі розподілу ресурсів виникають у тому разі, коли існує певний набір робіт (операцій), які необхідно виконати, а наявних ресурсів для виконання якнайкраще кожної роботи не вистачає.

Формулювання задачі розподілу ресурсів: можуть бути заданими як роботи, так і ресурси або лише роботи. Необхідно відшукати такий розподіл ресурсів, при якому максимізується спільний прибуток або результат, чи мінімізуються витрати.

Залежно від умов задачі розподілу ресурсів поділяють на групи:

1. Відомі і роботи, і ресурси. Необхідно розподілити ресурси між роботами таким чином, щоб максимізувати певну міру ефективності (прибуток) або мінімізувати очікувані витрати (витрати виробництва).

2. Відома лише кількість ресурсів. Визначити, який перелік робіт можна виконати з урахуванням цих ресурсів, щоб забезпечити максимум

певної міри ефективності.

3. Відомий лише перелік робіт. Визначити, які ресурси необхідні для того, щоб мінімізувати сумарні витрати виробництва.

Задачі ремонту та заміни обладнання виникають у тих випадках, коли обладнання зношується, застаріває і з часом підлягає заміні. Зношене обладнання ремонтують або повністю замінюють.

Формулювання задачі ремонту та заміни обладнання: визначити терміни поновлювального ремонту та момент заміни обладнання, за яких сумарні витрати на ремонт та заміну, а також витрати внаслідок погіршених характеристик мінімізуються.

Існує і таке обладнання, в якому деталі повністю виходять з ладу і не поновлюються (наприклад, електронні лампи). У цьому разі необхідно визначити терміни профілактичного контролю, за яких сумарні витрати на

проведення контролю та втрати від простою обладнання внаслідок тимчасово непрацюючого обладнання і заміни деталей мінімізуються.

Класифікація задач ремонту та заміни обладнання:

1) за характером зміни обладнання:

– задача зміни обладнання довгострокового використання;

– задача зміни обладнання з метою попередження відмов;

– задачі вибору оптимального плану попереджувального ремонту та профілактичного обслуговування;

2) за характером урахування витрат на обладнання – на дискретні та неперервні;

3) за виходом із ладу обладнання – на детерміновані та випадкові;

4) за стратегією зміни обладнання;

за часом урахування витрат на обладнання – із зведенням та без зведення витрат більш пізніх років до розрахункового [10].

Задачі масового обслуговування розглядають питання утворення та функціонування черг, що виникають у повсякденному житті. Наприклад, черги

літаків, що йдуть на посадку, покупці у супермаркеті тощо. Черги виникають унаслідок того, що потік вимог (клієнтів на обслуговування) є випадковим і ним не можна управляти. Якщо кількість пристроїв обслуговування досить велика, то черги виникають нечасто, однак при цьому неминучі довготривалі простої обладнання. З іншого боку, якщо недостатня кількість пристроїв обслуговування, створюються черги й можливі великі втрати майбутніх звернень унаслідок очікування.

Одне із можливих **формулювань задачі масового обслуговування:** визначити, яка кількість пристроїв масового обслуговування необхідна, щоб мінімізувати сумарні втрати, що очікуються від несвоєчасного обслуговування та простою обладнання.

Залежно від мети дослідження **задачі масового обслуговування** поділяють на такі:

– задачі аналізу – припускають оцінювання ефективності функціонування систем при незмінних, наперед відомих вихідних характеристиках;

– задачі синтезу – оптимізації – спрямовані на пошук оптимальних параметрів і характеристик функціонування.

Задачі упорядкування та координації пов'язані з визначенням оптимальної послідовності оброблення виробів, масивів інформації на різних приладах обслуговування або за певного способу обслуговування, що має кілька обов’язкових етапів. Іноді такі задачі називають задачами календарного планування або задачами складання розкладу.

Задачі планування та розміщення визначають оптимальну кількість і місця розміщення нових об’єктів з урахуванням їх взаємодії з існуючими об’єктами та між собою. Наприклад, на території нового мікрорайону відоме розміщення житлових будинків. Необхідно визначити кількість продуктових магазинів та місця їх розміщення таким чином, щоб населення могло одержувати необхідні послуги із заданим ступенем їх доступності одночасно із забезпеченням вимоги прибутковості торгових точок.

Задачі вибору маршруту найчастіше трапляються у дослідженні транспортних систем та систем зв'язку. До них відносять задачу вибору найкоротшого шляху, задачу комівояжера, задачу про максимальний потік.

Задачі сіткового планування та управління розглядають співвідношення між терміном закінчення великого комплексу операцій і моментом початку всіх операцій комплексу. Вони актуальні під час розроблення складних та високовартісних проектів. Можливі такі **постановки задач сіткового планування і управління:**

1) задана тривалість усього комплексу робіт; визначити терміни початку кожної операції, за яких мінімізується один з таких критеріїв:

– загальні витрати на виконання всього комплексу робіт;

– середньоквадратичний показник нерівномірності ресурсів, що використовуються;

– імовірність невиконання комплексу робіт у календарний термін;

– середньоквадратичне відхилення наявних ресурсів від необхідних;

2) відомі загальний перелік комплексу робіт та наявні ресурси для їх виконання; визначити терміни початку кожної операції, за яких мінімізується тривалість виконання всього комплексу робіт.

Задачі теорії ігор займаються дослідженням конфліктних ситуацій. Під конфліктною ситуацією розуміють ситуацію, у якій стикаються протилежні інтереси двох і більше сторін. У цих задачах необхідно визначити такі стратегії (оптимальні) поведінки кожної зі сторін, які б призвели якщо не до виграшу, то хоча б до мінімального програшу.

За різними ознаками виділяють такі **класи задач теорії ігор**:

– по можливості гравців об'єднуватись у групи з відповідною координацією власних дій щодо інтересів усієї групи: кооперативні, некооперативні, гібридні;

– за рівністю стратегій: симетричні (якщо гравці поміняються місцями, їх виграші за одні й ті самі ходи не зміняться) та асиметричні;

– за наявністю можливості змінювати ресурси або фонд гри: ігри з нульовою та ненульовою сумами; прикладом гри з нульовою сумою є гра у покер, де один виграє всі ставки інших;

– за наявністю інформації про ходи суперника: паралельні та послідовні ігри; у паралельних іграх гравці ходять одночасно або вони не мають інформації про вибір суперника, поки він його не зробить;

– за повнотою інформації: ігри з повною та ігри з неповною інформацією; у грі з повною інформацією для кожного гравця відомі всі ходи до поточного моменту і можливі стратегії супротивника;

– за терміном тривалості гри: ігри зі скінченною та ігри з нескінченною кількістю кроків; ігри реального світу, як правило, скінченні, дослідження ігор з нескінченною кількістю кроків – поки що «забавка» математиків;

– за типом компонентів гри: дискретні та неперервні; для дискретних ігор характерна скінченна кількість гравців, стратегій, результатів тощо;

– якщо значення хоча б одного з компонентів належить множині дійсних чисел, то це диференціальні числа.

Комбіновані задачі містять декілька типових задач одночасно [10].

1.3 Огляд методів лінійного програмування

1.3.1 Графічний метод розв'язування задач лінійного програмування

Для розв'язку двовимірних ЗЛП, тобто задач з двома змінними, а також деяких тривимірних задач застосовують графічний метод, що ґрунтується на геометричній інтерпретації та аналітичних властивостях ЗЛП. Розглянемо таку задачу. Знайти екстремум (мінімум, максимум) функції:

$$F = c_1 x_1 + c_2 x_2 \rightarrow \max(\min) \tag{1.1}$$

за умов

$$\begin{cases} a_{11} x_1 + a_{12} x_2 \{\leq, =, \geq\} b_1 \\ \dots\dots\dots\dots\dots \\ a_{m1} x_1 + a_{m2} x_2 \{\leq, =, \geq\} b_m \end{cases} \tag{1.2}$$

$$x_1 \geq 0, x_2 \geq 0 \quad (1.3)$$

Припустимо, що система (1.2) за умов (1.3) сумісна і многокутник її розв'язків обмежений.

Згідно з геометричною інтерпретацією ЗЛП (1.1) кожне *i*-те обмеження нерівність (1.2) визначає півплощину з граничною прямою $a_{i1}x_1 + a_{i2}x_2 = b_i \ (i = 1, 2, \dots, m)$. Системою обмежень (1.2) описується спільна частина, тобто множина точок, координати яких задовольняють всі обмеження задачі. Таку множину точок називають многокутником розв'язків, або областю допустимих розв'язків ЗЛП (ОДР).

Умова (1.3) невід'ємності змінних означає, що область допустимих розв'язків задачі належить першому квадранту системи координат двовимірного простору. Цільова функція ЗЛП геометрично інтерпретується як сім'я паралельних прямих с $c_1x_1 + c_2x_2 = const$ [14].

Деякі властивості ЗЛП (для графічного розв'язування):

1. Якщо ОДР – обмежена, то екстремум лінійної функції Z досягається принаймні в одній з вершин многокутника.

2. Якщо ОДР – необмежена, то екстремум функції Z або не існує, або досягається принаймні в одній з вершин ОДР.

3. Якщо оптимальне значення Z досягається одночасно у двох вершинах разом, то це ж саме екстремальне значення досягається в будь-якій точці відрізка, що з'єднує ці дві вершини (альтернативний оптимум).

Отже, розв'язати ЗЛП графічно означає знайти таку вершину многокутника розв'язків, у результаті підстановки координат якої в (2.1) лінійна цільова функція набуває найбільшого (найменшого) значення [14].

Алгоритм графічного методу

1. У системі координат будуємо прямі, рівняння яких отримуємо шляхом заміни обмежень знаків нерівності на знак рівності.

Цільова функція з системою обмежень має вигляд:

$$Z = c_1x_1 + c_2x_2 \rightarrow \max(\min) \quad (1.4)$$

$$a_{11}x_1 + a_{12} = b_1$$
$$\dots\ \dots\ \dots\ \dots\ \dots \quad (1.5)$$
$$a_{m1}x_1 + a_{m2}x_2 = b_m$$

$$x_1 \geq 0, x_2 \geq 0 \quad (1.6)$$

2. Знаходимо півплощини, що визначаються кожному обмеженню задачі.

Вибираємо будь-яку точку на площині та підставляємо в систему обмежень. Якщо нерівність виконується – беремо півплощину, де знаходиться обрана точка. Якщо нерівність не виконується – беремо протилежну півплощину.

3. Обираємо багатокутник розв’язків – це спільна частина знайдених півплощин.

4. Будуємо вектор $\overline{C} = (c_1; c_1)$, де точки дорівнюють коефіцієнтам цільової функції.

5. Будуємо пряму $c_1x_1 + c_2x_2 = const$, яка повинна бути перпендикулярна до вектора $\overline{C}$.

6. Уявно переміщуємо пряму $c_1x_1 + c_2x_2 = const$ в напрямку вектора $\overline{C}$, в залежності від обраного екстерума. Якщо задача максимізації, то в напрямку вектора $\overline{C}$. Якщо задача мінімізації, то в протилежну сторону.

7. Цільова функція отримує екстремального значення на вершині багатокутника. Визначаємо точки координат, точки перетину двох прямих, отриманих шляхом заміни знаків обмежень. Обрані точки підставляємо в цільову функцію для обчислення екстремального значення цільової функції [3].

1.3.2 Симплекс-метод розв’язання задач лінійного програмування

Графічний метод для визначення оптимального плану задачі лінійного програмування доцільно застосовувати лише для задач із двома змінними. За більшої кількості змінних вдаються до загального методу розв'язування задач лінійного програмування – так званого симплекс-методу. Процес розв'язування задачі симплекс-методом має ітераційний характер: обчислювальні процедури

(ітерації) одного й того самого типу повторюються у певній послідовності доти, доки не буде отримано оптимальний план задачі або з'ясовано, що його не існує

Симплекс-метод – це поетапна обчислювальна процедура, в основу якої покладено принцип послідовного поліпшення значень цільової функції переходом від одного опорного плану задачі лінійного програмування до іншого [3].

Цільова функція з системою обмежень має вигляд:

$$F = c_1x_1 + \cdots + c_nx_n \rightarrow \max(min) \tag{1.7}$$

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n\{\leq, =, \geq\}\, b_1 \\ \quad \ldots\ \ldots\ \ldots\ \ldots\ \ldots \\ a_{m1}x_1 + \cdots + a_{mn}x_n\{\leq, =, \geq\}\, b_m \end{cases} \tag{1.8}$$

$$x_i \geq 0, i = 1, \ldots, n, m \tag{1.9}$$

Алгоритм симплекс-методу:

1. Потрібно записати задачу лінійного програмування у вигляді канонічної форми.

Основні ознаки канонічної форми:

- Цільова функція прямує до максимізації;
- Всі значення b_i повинні бути більше дорівнює нуля;
- В заданій системі обмежень замінюємо знаки нерівності на знак рівності.

Якщо задана цільова функція прямує до мінімізації, то потрібно всі знаки змінити на протилежні.

Якщо b_i менше за 0, тоді в рядку нерівності всі знаки змінюємо на протилежні.

Якщо в системі нерівності знак менше або менше рівне, то вводимо додаткову змінну u_i з арифметичною дією додавання до лівої частини нерівності. При цьому змінюємо знак нерівності на знак рівності.

Якщо в системі нерівності знак більше або більше рівне, то вводимо додаткову змінну u_i з арифметичною дією віднімання до лівої частини нерівності. При цьому $u_i > 0$ та змінюємо знак нерівності на знак рівності.

Всі дії, задані вище, виконуються при умові, що $x_n > 0$.

Якщо на зміну не накладена умова, тобто вона довільна, тоді замість x вводимо $u_i - u_i$. При цьому накладаємо умову, що $u_i \geq 0$.

Канонічна форма цільової функції має вигляд:

$$F = c_1x_1 + \cdots + c_nx_n \rightarrow max \tag{2.0}$$

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n + u_1 = b_1 \\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n + u_m = b_m \end{cases} \tag{2.1}$$

$$x_i \geq 0, b_i \geq 0, u_i \geq 0 \tag{2.2}$$

2. Будуємо вектор-стовпчиків коефіцієнтів при невідомих в системі обмежень та створюємо базис-стовпчикі, які в поєднанні дають одиничну матрицю.

Кількість рядочків в отриманій одиничній матриці повинна дорівнювати кількість нерівності в обмежені.

$$p_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{m1} \end{pmatrix}, \ldots, p_m = \begin{pmatrix} a_{1n} \\ a_{2n} \\ a_{3n} \\ a_{mn} \end{pmatrix}, \ldots, p_{m+1} = \begin{pmatrix} 1 \\ 0 \\ \ldots \\ 0 \end{pmatrix}, \ldots, p_n = \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 1 \end{pmatrix}$$

3. Будуємо симплекс таблицю, яку оформляти у вигляді таблиці 1.1.

Якщо в цільовій функції немає певної змінної, то її коефіцієнт дорівнює нулю.

Таблиця 1.1 – Симплекс таблиця

№	Базис	C_b	P_0	c_1	c_2	…	c_k	…	c_m	c_{m+1}	c_{m+2}	…	c_n
				p_1	p_2	…	p_k	…	p_m	p_{m+1}	p_{m+2}	…	p_n
1	p_{m+1}	0	b_1	a_{11}	a_{12}	…	a_{1k}	…	a_{1m}	1	0	…	0
2	p_{m+2}	0	b_2	a_{21}	a_{22}	…	a_{2k}	…	a_{2m}	0	1	…	0
3	p_{m+3}	0	b_3	a_{31}	a_{32}	…	a_{3k}	…	a_{3m}	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…

Продовження таблиці 1.1 – Симплекс таблиця

n	p_n	0	b_n	a_{n1}	a_{n2}	…	a_{nk}	…	a_{nm}	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…
m	p_m	0	b_m	a_{m1}	a_{m2}	…	a_{mk}	…	a_{mm}	0	0	…	1
m+1			F_0	Δ_1	Δ_2	…	Δ_k	…	Δ_m	Δ_{m+1}	Δ_{m+2}	…	Δ_n

Симплекс таблиця, де значення F_0 дорівнює скалярному добутку вектора P_0 на вектор C_b:

$$F_0 = \Sigma c_{bi} b_i \tag{2.3}$$

Після заповнення таблиці, вихідний опорний план перевіряють на оптимальність. Для цього переглядають елементи (m+1)-го рядка таблиці. В результаті може мати місце один з наступних трьох випадків

1. Усі $\Delta_j \leq, (j = \overline{1,n})$.
2. Існує j, для якого $\Delta_j < 0$ і для кожного такого j симплекс таблиці, міститься принаймні одне додатнє число a_{ij} $(a_{ij} > 0)$.
3. Існує j, для якого $\Delta_j < 0$ і всі відповідні цьому індексом величини $a_{ij} \leq, (i = \overline{1,m})$.

В першому випадку ми отримали **оптимальний розв'язок задачі лінійного програмування**. Значення F_0 останньої симплекс таблиці буде містити максимальне значенням цільової функції.

В другому випадку є можливість покращити значення цільової функції за допомогою переходу до іншого опорного плану (перехід від одного опорного плану до іншого здійснюється заміною базису, тобто виключенням з нього якоїсь змінної та включенням замість неї нової, з числа вільних змінних).

Третій випадок свідчить про необмеженість цільової функції на множині розв'язків.

Далі розглянемо, яким чином здійснюється перехід до іншого опорного плану. Для цього серед елементів рядка оцінок останньої симплекс

таблиці вибираємо те значення Δ_j, яке по абсолютній величині приймає максимальне значення. Якщо їх є декілька, то вибираємо те, якому відповідає найбільше C_j. Після того, як ми вибрали k-й стовпець ($\Delta_k < 0$), вектор p_k потрібно ввести в базис.

Для того, щоб визначити на місце якого вектора базису вводити вектор p_k, визначаємо $\min(b_i/a_{ik}), (i = 1, m)$, для усіх $a_{jk} > 0$. Нехай це буде елемент, який міститься в n-му рядку, тобто елемент a_{nk}. Надалі даний елемент будемо називати розв'язуючим елементом. Стовпець k і рядок n, напрямляючими стовпцем і рядком відповідно. Наступний крок полягає у побудові нової симплекс таблиці, тобто визначення усіх її коефіцієнтів згідно нового базису, які обчислюються за наступними формулами:

$$b_i' = \begin{cases} b_i - \left(\frac{b_n}{a_{nk}}\right) a_{nk}, i \neq n \\ \frac{b_n}{a_{nk}}, i = n \end{cases}; \; a_{ij}' = \begin{cases} a_{ij} - \left(\frac{a_{nj}}{a_{nk}}\right) a_{jk}, i \neq n \\ \frac{a_{nj}}{a_{nk}}, i - n \end{cases} \tag{2.4}$$

1.3.3 Розв'язування двоїстої задачі лінійного програмування

Пряма задача лінійного програмування визначає скільки необхідно виготовити продукту із заданих ресурсів з метою максимізації загальної виручки.

Та має такий вигляд:

$$f = c_1, c_2, \dots, c_n \rightarrow max \tag{2.5}$$

$$m\left\{\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} * \begin{pmatrix} x_1 \\ \dots \\ x_n \end{pmatrix} \begin{matrix} \leq b_1 \\ \dots \\ \leq b_m \end{matrix}\right. \tag{2.6}$$

В прямій задачі наявні, такі задані параметри:

- Обсяги ресурсів (b_i);
- Норми витрат (x_i, y_i);
- Ціни від реалізації (i, j).

На відміну від прямої задачі в двоїстій задачі лінійного програмування необхідно визначити ціни ресурсів. Кожному ресурсу (b_i) ставимо відповідну оцінку (y_i) ціни за одиницю ресурсу.

Має вигляд:

$$f = b_1, b_2, \dots, b_m \rightarrow min \tag{2.7}$$

$$m\left\{\begin{pmatrix} a_{11}\ a_{12} \dots a_{m1} \\ a_{n1}\ a_{2n} \dots a_{mn} \end{pmatrix} * \begin{pmatrix} y_1 \\ \dots \\ y_m \end{pmatrix} \begin{matrix} \leq c_1 \\ \dots \\ \leq c_n \end{matrix}\right. \tag{2.8}$$

Розв'язування двоїстої задачі дозволяє визначити цінність кожного ресурсу, який використовується в прямій задачі.

Двоїста задача лінійного програмування утворюється з прямої задачі за такими правилами:

1. Кожному обмеженню прямої задачі відповідає змінна двоїстої задачі. Кількість невідомих двоїстої задачі дорівнює кількості обмежень прямої задачі.

2. Кожній змінній прямої задачі відповідає обмеження двоїстої задачі, причому кількість обмежень дорівнює кількості невідомих прямої задачі.

3. Якщо цільова функція прямої задачі задається на пошук найбільшого значення (max), то цільова функція двоїстої задачі – на визначення найменшого значення (min), і навпаки.

4. Коефіцієнтами при змінних в цільовій функції двоїстої задачі є вільні члени системи обмежень прямої задачі: $(b_1, \dots, b_m)$.

Правими частинами системи обмежень двоїстої задачі є коефіцієнти при змінних в цільовій функції прямої задачі: $(c_1, \dots, c_n)$ [3,15].

Матриця коефіцієнтів прямої задачі транспортується n⇔m, де n – рядок, m – стовпчик.

Тобто матриця прямої задачі має вигляд:

$$A = \begin{pmatrix} a_{11}\ a_{12} \dots a_{1n} \\ a_{21}\ a_{22} \dots a_{2n} \\ \dots\dots\dots\dots\dots \\ a_{m1}\ a_{m2} \dots a_{mn} \end{pmatrix} \tag{2.9}$$

Матриця двоїстої задачі лінійного програмування має вигляд:

$$A^T = \begin{pmatrix} a_{11}\ a_{21} \dots a_{m1} \\ a_{12}\ a_{22} \dots a_{m2} \\ \dots\dots\dots\dots\dots \\ a_{1n}\ a_{2n} \dots a_{mn} \end{pmatrix} \tag{3.0}$$

5. Якщо в якомусь з обмежень знак більше рівно та функція прямує до найбільшого значення (max), то змінюємо знаки на протилежний в заданій нерівності.

6. Якщо якийсь зі знаків рівності, то в двоїстій задачі на зміну в даному рівнянні накладається умова $\forall$, ($y_n \ \forall, \ y_n = (-\infty; +\infty)$).

Двоїсті пари задач лінійного програмування бувають симетричні та несиметричні.

У симетричних задачах обмеження прямої та двоїстої задач є нерівностями, а змінні обох задач можуть набувати лише невід'ємних значень.

Різні можливі форми прямих задач лінійного програмування та відповідні їм варіанти моделей двоїстих задач наведено далі [3].

Пряма задача	**Двоїста задача**
Симетричні	
$F = \sum c_j x_j \to max$	$F = \sum b_i y_i \to min$
$\sum a_{ij} x_j \le b_j$	$\sum a_{ij} y_i \ge c_j$
$x_j \ge 0$	$y_j \ge 0$
$F = \sum c_j x_j \to min$	$F = \sum b_i y_i \to max$
$\sum a_{ij} x_j \le b_i$	$\sum a_{ij} y_i \ge c_j$
$x_j \ge 0$	$y_j \ge 0$

Основні теореми двоїстості задач:

І теорема двоїстості

Якщо задача лінійного програмування має кінцевий оптимум, то двоїста до неї задача також має кінцевий оптимум, тобто вільний член, який співпадає з оптимумом початкової задачі (бо небазисні змінні дорівнюють нулю) [9].

$$\max f = \min f^*$$

ІІ теорема

В оптимальній функції мети початкової задачі f max значення коефіцієнтів при неосновних змінних дорівнюють значенням базисних змінних оптимального рішення двоїстої задачі [9].

1.3.4 Двоїстий симплекс метод розв’язання задач лінійного програмування

Двоїстий симплекс-метод (ДСМ) безпосередньо застосовується до розв'язування майже канонічної задачі лінійного програмування (МКЗЛП), яка формулюється таким чином [14]:

Знайти вектор x = (x_1,..., x_n), що мінімізує лінійну функцію

$$L(x) = c_1x_2 + \cdots + c_nx_n \quad (3.1)$$

і задовольняє систему лінійних обмежень

$$x_i + a_{i,m} + 1x_m + 1 + \cdots + a_{in}x_n = a_i0, i = 1, \dots, m, \quad (3.2)$$

$$x_j \geq 0, j = 1, \dots, n, \quad (3.3)$$

(компоненти a_i0 вектора обмежень A_0 можуть бути від'ємними) при додатковій умові: відносні оцінки (симплекс-різниці) Δj змінних x_j невід'ємні.

Вектор x = (x_1,..., x_n) називається майже допустимим базисним розв'язком (МДБР) МКЗЛП, якщо його компоненти задовольняють обмеження (3.2), і ненульовим компонентам x_jвідповідають лінійно незалежні вектори умов Aj [14].

Ознака оптимальності:

Якщо на деякому кроці ДСМ компоненти МДБР x* невід'ємні, то x* — оптимальний розв'язок МКЗЛП.

Ознака відсутності розв'язку:

Оптимального розв'язку МКЗЛП не існує, якщо на якому-небудь кроці ДСМ в рядку з a_i0< 0 всі компоненти a_{ij}≥ 0, j=1,...,n. В цьому випадку допустима множина розв'язків МКЗЛП порожня [14].

Алгоритм двоїстого симплекс-методу:

На кожному кроці ДСМ виконуються такі дії (розрахункові формули наводяться лише для першого кроку).

1. Розглядається МДБР x =(a_{10},..., a_{m0},0,...,0).

   Обчислюються відносні оцінки (симплекс-різниці) Δ_j небазисних змінних x_j, j=m+1, …, n, за формулою:

   Δj = c_j– (c_6, Aj),

де $c_б$= (c_1, …, c_m), Aj — вектор умов, що відповідає змінній x_j (відносні оцінки базисних змінних дорівнюють нулю).

Якщо для всіх i = 1, …, m виконується умова a_{i0}≥ 0, то МДБР x буде оптимальним розв'язком МКЗЛП. Кінець обчислень.

Якщо існує таке i, що a_{i0} < 0, а коефіцієнти a_{ij}≥ 0, j = 1, …, n, то МКЗЛП не має допустимих розв'язків. Кінець обчислень.

2. Якщо існують індекси i, для яких a_{i0}< 0, а серед відповідних компонентів a_{ij}, j = 1, …, n, є від'ємні, то знаходять l:

   $l = argmin\ a_{i0}$,

   i: $a_{i0} < 0$

   обчислюють відношення $\gamma_j = -\Delta_j / a_{lj}$ для всіх $a_{lj} < 0$ та визначають k:

   $k = argmin\ \gamma_i$,

   i: $a_{lj} < 0$

3. Переходять до нового МДБР, виключаючи з базису вектор A_l і вводячи до базису векторA_k. Згаданий перехід здійснюється за допомогою симплексперетворень (елементарних перетворень Жордана-Гаусса з ведучим елементом A_{lk}) над елементами розширеної матриці умов. Перехід до пункту 1 [14].

Отже, щоб вирішити двоїстий симплекс метод треба виконати алгоритм дій, який описаний в підпункті 1.3.3. Після успішного виконання, переходимо до підпункту 1.3.2 та вирішуємо задачу за допомогою алгоритму дій симплекс метода.

1.4 Опис середовища розробки Microsoft Visual Studio. Структура C#

Додаток що розробляється має відповідати наступним вимогам:

- Необхідно обчислювати алгоритм рішень задач лінійного програмування.

- Мати графічний інтерфейс.
- Підтримувати інтерактивний введення і редагування вихідних даних.
- Забезпечувати покроковий перегляд порядку рішення ЗЛП.

Для розробки програми, що задовольняє даним вимогам, була обрана мова C#.

Microsoft .NET - нова платформа, яка заснована на Windows.

Введення .NET утворює нову парадигму програмування, що поєднує різні стилі програмування. У фундаменті її реалізації лежить бібліотека класів .NET Framework. Вона містить високорівневу модель програмування, яка прискорює розробку програмного забезпечення. Для неї також характерна кращий захист додатків, широкі можливості розробки насичених і функціональних веб-додатків.

.NET Framework - ядро концепції .NET. Дана структура керує програмними додатками і запускає їх, містить бібліотеку класів .NET Framework (FCL), забезпечує захист і безліч інших можливостей.

.NET Framework - програмна технологія від компанії Microsoft, призначена для створення звичайних програм і web -додатків.

Архітектура .NET Framework на прикладі .NET 4.5 приведена на рис. 1.3.

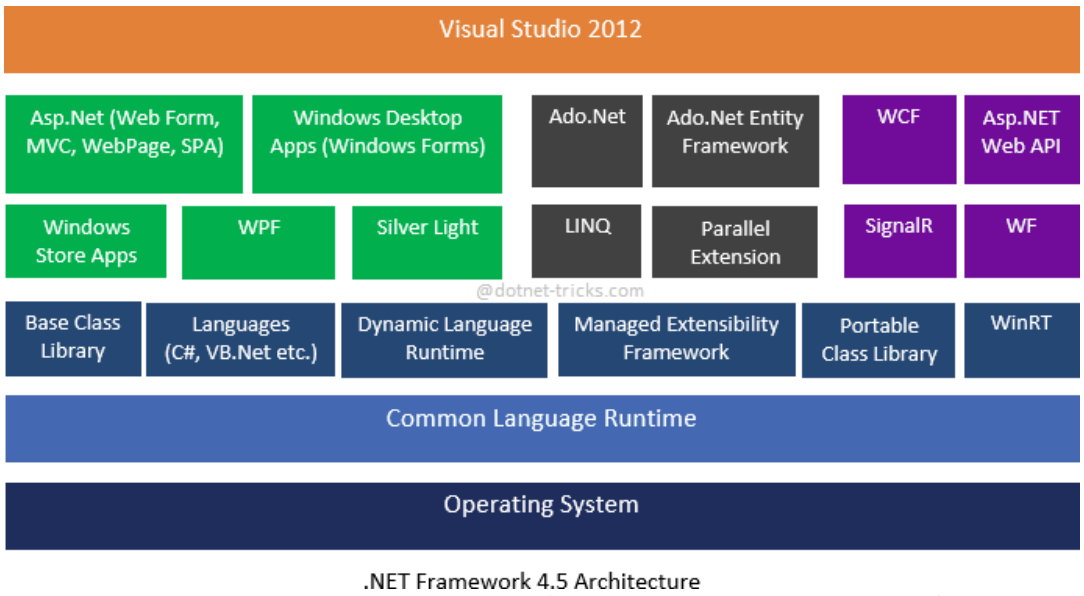


Рисунок 1.3 – Вид архітектури .NET Framework 4.5.

Однією з основних ідей Microsoft .NET є сумісність різних служб, написаних на різних мовах. Кожна бібліотека (збірка) в .NET має відомості про свої версії, що дозволяє усунути можливі конфлікти між різними версіями збірок. .NET є патентованої технологією корпорації Microsoft.

Подібно технології Java, середа розробки .NET створює байт-код, призначена для виконання віртуальною машиною. Мова цієї машини в .NET називається Microsoft Intermediate Language, або Common Intermediate Language або просто IL. Застосування байт-коду дозволяє отримати кросплатформенність на рівні скомпільованого проекту, а не тільки на рівні вихідного тексту. Перед запуском збірки в середовищі виконання CLR байт-код перетворюється вбудованим в середу JIT-компілятором в машинні коди цільового процесора.

Сучасна технологія динамічної компіляції дозволяє досягти рівня швидкодії, аналогічного традиційним «Статичним» компіляторам, і питання швидкодії часто залежить від якості того чи іншого компілятора.

Як об'єктно-орієнтована мова, C# підтримує поняття інкапсуляції, успадкування та поліморфізму.

Для розробки додатків з графічним інтерфейсом в Microsoft .NET існує бібліотека Windows Forms. Windows Forms - інтерфейс побудови додатків на базі класів .NET Framework.

Завдяки своїй архітектурі Windows Forms сприяє підвищенню однорідності і ступеня уніфікації програмної моделі в порівнянні з традиційними підходами. Це дозволяє усувати велику кількість помилок і протиріччя від використання таких інструментів, як Windows API і т.п. Наприклад, відомо, що деякі стилі вікна можуть застосовуватися тільки до створеного і існуючого вікна. При застосуванні Windows Forms можливо динамічна зміна параметрів вікон і їх властивостей.

Для вирішення конфліктних ситуацій і структурної організації багатьох видів Windows Forms має ієрархічну структуру (Рис. 1.4).

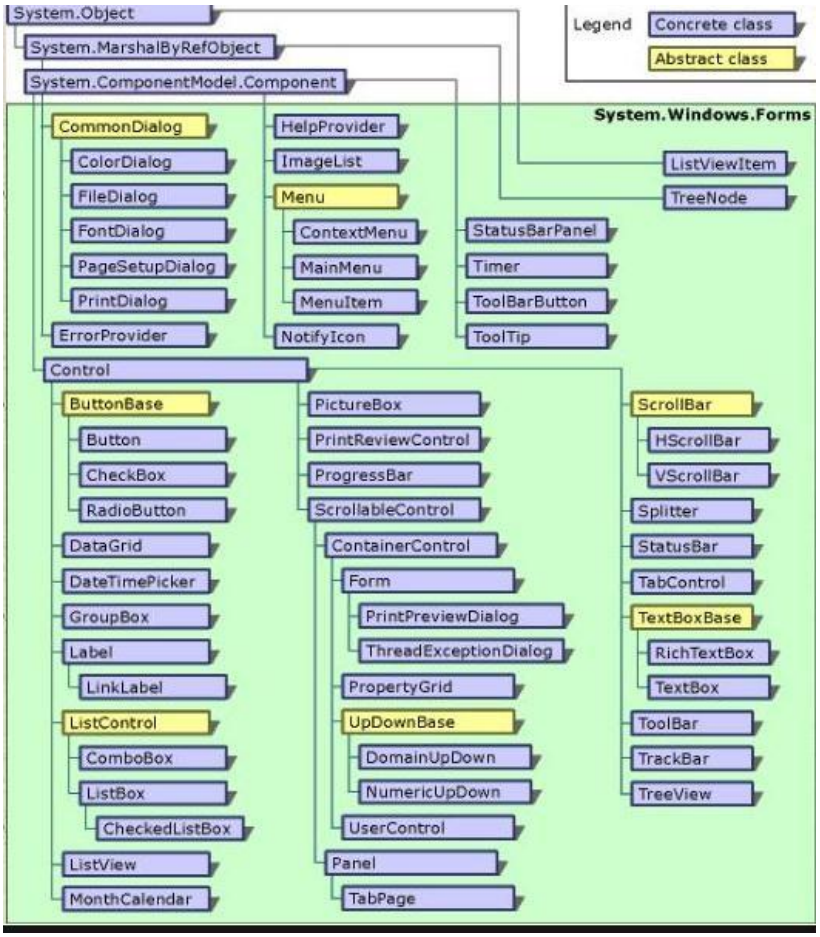


Рисунок 1.4 – Елементи архітектури Windows Forms

Кореневий розділ, System, визначає фундаментальні типи даних, використовувані всіма додатками .NET.

Важливою межею моделі програмування Windows Forms також є механізм, який використовується формами для відповіді на введення в меню, засобів управління і інших елементів програми.

У мові програмування C# так, як і в інших мовах, які підтримують .NET Common Language Runtime (CLR), події - члени типу першого класу нарівні з методами, полями і властивостями. Фактично всі керуючі класи (control classes) Windows Forms (а також і багато некеровані класи) створюють події. Такий підхід дозволяє істотно переглянути організацію управління подіями і забезпечує більш високий рівень звернення до них.

2 ВИМОГИ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Етапи життєвого циклу програмного забезпечення

Життєвий цикл програмного забезпечення (SDLC – Software Development Life Cycle) – це період часу, який починається з моменту прийняття рішення про необхідність створення програмного продукту і закінчується в момент його повного вилучення з експлуатації. Цей цикл – процес побудови і розвитку програмного забезпечення [22].

Фаза - це відрізок часу між двома важливими контрольними точками (milestones) процесу, в яких досягаються чітко визначені цілі, завершення створення робочих продуктів і приймається рішення про перехід до наступної фази.

Існує чотири фази життєвого циклу розробки:

- Початкова фаза.
- Розробка.
- Конструювання.
- Впровадження.

Початкова фаза (inception) - перша фаза процесу, протягом якої початкова ідея отримує достатнє обґрунтування для забезпечення переходу до фази розробки.

Розробка (elaboration) - друга фаза процесу, коли визнаються вимоги до продукту та архітектури. Системні вимоги можуть включати в себе широкий діапазон формулювань - від загальних пропозицій до детальних критеріїв оцінки, кожен з яких присвячений опису певного функціонального або нефункціонального поведінки системи і закладає основи для організації тестування.

Конструювання (construction) - третя фаза, коли виконуваний архітектурний прототип набуває форму, в котрій він може бути представлений

користувачам. На цьому етапі вимоги до системи, особливо критерії оцінки, постійно переглядаються відповідно до потреб. Також виділяються необхідні ресурси для зменшення ризиків.

Впровадження (transition) - четверта фаза процесу, в ході якої програмне забезпечення передається користувачам. Але процес розробки рідко завершується на цьому, тому що навіть на цій фазі система безперервно удосконалюється: усуваються помилки і додаються не увійшли в ранні версії нові функціональні можливості.

Один елемент, який відрізняє процес розробки в цілому і присутній у всіх чотирьох фазах, - це ітерація.

Ітерація - чітко визначена послідовність дій з чітко сформульованим планом і критеріями оцінки, яка веде до появи нової версії системи, яка може бути виконана, протестована і оцінена. Виконана система не повинна бути реалізована зовні. Оскільки ітерація породжує виконуваний продукт, досягнутий прогрес і рівень ризику можуть бути оцінені заново після кожної такої ітерації. Це означає, що життєвий цикл розробки програмного забезпечення можна характеризувати як безперервний потік виконуваних версій, що реалізують архітектуру системи з проміжною корекцією для зниження потенційного ризику. Це підкреслює значення архітектури як найважливішого елемента програмної системи і фокусує UML на моделюванні різних її уявлень.

2.2 Підходи до розроблення програмного забезпечення

Зміни в програмному забезпеченні призвели до нових змін у способах роботи програмних продуктів. Тим самим, збільшилися інтерактивні можливості програм.

Існує два підходи до розроблення ПЗ: структурний і об’єктно-орієнтований. Структурний підхід заснований на двох методах: діаграмах потоків даних (data flow diagrams) для моделювання процесів і діаграмах сутність-зв’язок (entity

relationship diagrams) для моделювання даних. Він є функціонально-орієнтованим і розглядає DFD-діаграми як рушійну силу розроблення ПЗ [15].

Наступним є об’єктно-орієнтований підхід. Асоціація виробників ПЗ Object Management Group (OMG) затвердила як стандартний засіб моделювання цього підходу мову UML. Порівняно зі структурним підходом об’єктно-орієнтований підхід більшою мірою орієнтований на дані – він розвивається довкола моделей класів. Зростаюче значення використання в мові UML претендентів сприяє незначному зсуву акцентів від даних до функцій. До найбільш важливих категорій додатків, для яких потрібна об’єктна технологія, відносяться обчислювальна обробка для робочих груп і системи мультимедіа. Об’єктний підхід до розроблення систем слідує ітеративному процесу з нарощуванням можливостей. Єдина модель конкретизується на етапах аналізу, проектування та реалізації.

Об’єктно-орієнтований підхід призводить до виникнення низки труднощів: – оскільки етап аналізу проводиться на ще вищому рівні абстракції і якщо серверна частина рішення з реалізації передбачає використання реляційної бази даних, семантичний розрив між концепцією і її реалізацією може бути значним; – керування проектом складно здійснювати; – висока складність рішення, що у свою чергу позначається на таких характеристиках ПЗ, як пристосованість до супроводу і масштабованість[15].

2.3 Моделювання програмного забезпечення мовою UML

Уніфікована мова моделювання (Unified Modeling Language - UML) - це мова для візуалізації, специфікації, конструювання та документування артефактів програмних систем [15].

Мова моделювання - це мова, словник і правила, які зосереджені на концептуальному і фізичному поданні системи. UML - це стандартний інструмент для розробки «креслень» програмного забезпечення.

UML підходить для моделювання будь-яких систем - від інформаційних систем масштабу підприємства до розподілених Web-додатків і навіть вбудованих систем реального часу. Це дуже виразна мова, призначений для представлення системи з усіх точок зору, що відносяться до її розробки та впровадження. Незважаючи на багатство виразних засобів, UML простий для розуміння і застосування. Ефективне використання UML починається з формування концептуальної моделі мови і трьома основними елементами: базовими будівельними блоками UML, правилами, що визначають, як ці блоки можуть поєднуватися між собою, а також деякими загальними механізмами мови.

Моделювання необхідно для розуміння системи. При цьому жодна модель не є абсолютно достатньою. Навпаки, щоб зрозуміти більшість систем, крім самих тривіальних, часто потрібно безліч взаємопов'язаних моделей. Відносно програмних систем це означає, що необхідна мова, засобами якого можна описати архітектуру системи з різних точок зору, причому протягом усього життєвого циклу її розробки.

Словник і правила такого мови, як UML, говорять про те, як створювати і читати добре узгоджені моделі, але не говорить про те, які саме моделі в яких випадках потрібно створювати. Це завдання всього процесу розробки програмного забезпечення. Добре організований процес повинен сам підказати, які будуть потрібні робочі продукти, які ресурси знадобляться для їх створення та управління ними, як їх використовувати для оцінки виконаної роботи і управління проектом в цілому.

2.3.1 Діаграма класів

Діаграми класів - найпоширеніші діаграми, що використовуються в UML. Діаграма класів складається з класів, інтерфейсів, асоціацій та співпраці. Діаграми класів в основному представляють об'єктно-орієнтоване уявлення про систему, яка має статичний характер.

Клас представляє статичний вигляд програми. Діаграма класів використовується не тільки для візуалізації, опису та документування різних аспектів системи, а й для побудови виконуваного коду програмного додатку.

Діаграма класу описує атрибути та операції класу, а також обмеження, накладені на систему. Діаграми класів широко використовуються при моделюванні об'єктно-орієнтованих систем, оскільки це єдині діаграми UML, які можна безпосередньо зіставити з об'єктно-орієнтованими мовами.

Як правило, діаграми UML безпосередньо не зіставляються з будь-якими об'єктно-орієнтованими мовами програмування, але діаграма класів є винятком.

Діаграма класів чітко демонструє відображення об’єктно-орієнтованих мов, таких як Java, C ++ тощо. З практичного досвіду, діаграма класів зазвичай використовується для побудови.

UML діаграму класів приведено на рисунку 2.1.

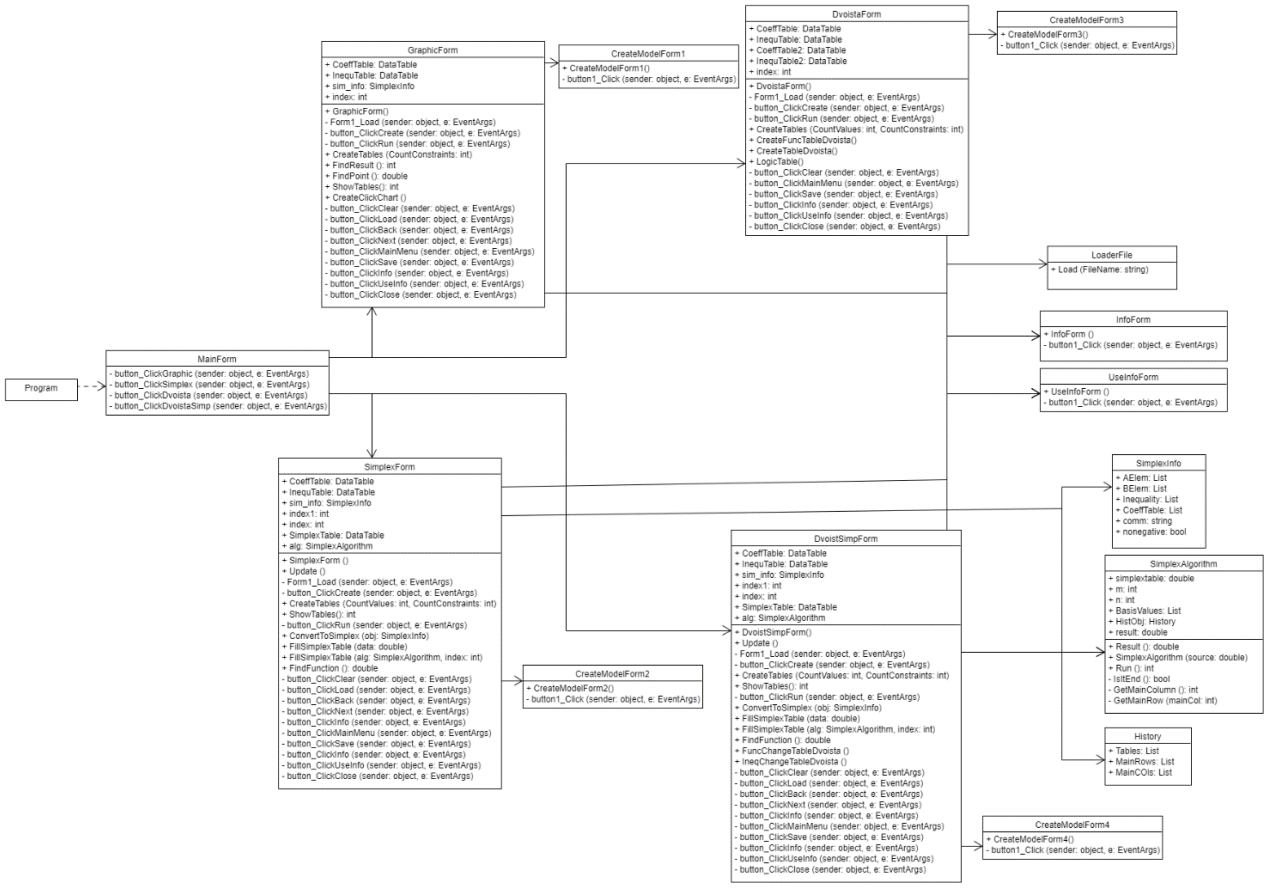


Рисунок 2.1 – UML діаграма класів

2.3.2 Діаграми станів

Діаграми станів - це один з п'яти видів діаграм UML, призначених для моделювання динамічних аспектів поведінки систем. Діаграма станів показує кінцевий автомат. Діаграми станів підходять для моделювання життєвого циклу об'єкта, яка демонструє потік управління від стану до стану всередині окремого об'єкта.

Діаграми станів можуть бути приєднані до класів, варіантів використання або до всієї системи з метою візуалізації, специфікування, конструювання та документування динаміки окремих об'єктів. Діаграми станів корисні не тільки для моделювання динамічних аспектів системи, але і для конструювання виконуваних систем за допомогою прямого і зворотного проектування.

Діаграма станів (state diagram) показує автомат, зосереджуючи увагу на потоці управління від одного стану до іншого. Зображується у вигляді графа з вершинами і дугами (ребрами).

Автомат (state machine) - це опис послідовності станів, через які проходить об'єкт протягом життєвого циклу, реагуючи на події, а також опис реакції на ці події.

Стан (state) - ситуація в життєвому циклі об'єкта, протягом якої він задовольняє деякому умовою, виконує деяку діяльність або очікує деякого події.

Подія (event) - специфікація суттєвого факту, який відбувається в часі і просторі. В контексті автомата подія - це вплив, який викликає перехід між станами.

Перехід (transition) - зв'язок між двома станами, що показує, що об'єкт, що знаходиться в першому стані, повинен виконати деякі дії і перейти до другого, як тільки відбудеться певна подія і будуть виконані певні умови.

Діяльність (activity) специфікує роботу, яка відбувається всередині автомата.

Дія (action) - примітивне виконується обчислення, що приводить до зміни стану моделі або поверненню значення.

UML діаграму станів приведено на рисунку 2.2.

Рисунок 2.2 – UML діаграма станів

2.3.3 Діаграми пакетів

Діаграмою пакетів є діаграма, що містить пакети класів і залежності між ними. Пакети і залежності між ними є елементами діаграми класів, тобто діаграма пакетів - це всього лише форма діаграми класів. Однак на практиці причини побудови таких діаграм різні.

Діаграма пакетів слугує, в першу чергу, для організації елементів в групи по будь-якою ознакою з метою спрощення структури і організації роботи з моделлю системи. Діаграми пакетів забезпечують відмінну можливість візуального представлення залежностей між частинами системи і часто використовуються для діагностики або визначення порядку компіляції. В пакетах можуть групуватися практично будь-які елементи UML (в тому числі і самі пакети).

Кожен пакет має ім'ям, яке уточнює область видимості кожного елемента пакета. Елементи, що входять в один пакет, можуть звертатися один до одного без уточнення імен.

Відповідно до створених методів та допоміжних частин, була створена така UML діаграма пакетів, яка приведена в додатку В.

Яка включає в себе:

- Пакет Graphic.

- Пакет Simplex.
- Пакет Dvoista.
- Пакет DvoistSimp.
- Пакет Info.
- Пакет Logic code.

2.4 Детальний опис основних програмних модулів

Опис основних файлів додатка приведено в таблиці 2.1.

Таблиця 2.1 – Опис основних файлів додатка

Ім'я файлу	Опис файлу
MainForm.cs	Головна форма додатка
GraphicForm.cs	Форма графічного методу ЗЛП
SimplexForm.cs	Форма симплекс методу ЗЛП
DvoistaForm.cs	Форма двоїстої задачі
DvoistSimpForm.cs	Форма двоїстого симплекс методу ЗЛП
CreateModelForm1.cs, CreateModelForm2.cs, CreateModelForm3.cs, CreateModelForm4.cs	Форми створення моделі ЗЛП
InfoForm.cs	Форма інформації про додаток
InfoUseForm.cs	Форма з інформацією, як використовувати додатка
LoaderFile.cs	Модуль роботи з файлами моделі ЗЛП
SimplexAlgorithm.cs	Модуль симплекс-методу

Основні класи додатку наведені в таблиці 2.2.

Таблиця 2.2 – Опис основних класів додатка

Ім'я класу	Опис класу
SimplexInfo	Клас інформаційного об'єкта ЗЛП
LoaderFile	Клас завантажувача файлів
History	Клас історії обчислень
SimplexAlgorithm	Клас алгоритму симплекс методу
MainForm	Клас головної форми
GraphicForm	Клас форми графічного методу ЗЛП
SimplexForm	Клас форми симплекс методу ЗЛП
DvoistaForm	Клас форми двоїстої задачі
DvoistSimpForm	Клас форми двоїстого симплекс методу ЗЛП

Список основних полів класу SimplexInfo наведено в таблиці 2.3.

Таблиця 2.3 – Список основних полів класу SimplexInfo

Ім'я поля / методу	Опис поля / методу
public List <List <double>> AElem	Лист в якому записано задана матриця обмежень

Продовження таблиці 2.3 – Список основних полів класу SimplexInfo

Ім'я поля / методу	Опис поля / методу
public List <List <double>> BElem	Лист в якому записано задана права частина обмежень
public List <string> Inequality	Лист в якому записано заданий список операцій обмежень, а саме більше рівно, менше рівно чи рівно
public List <double> CoeffTable	Лист в якому записано задані коефіцієнти цільової функції
public string comm	Вид оптимізації (максимум або мінімум)
public bool nonegative	Змінна невід'ємна або немає

Список основних полів класу History наведено в таблиці 2.4.

Таблиця 2.4 – Список основних полів класу History

Ім'я поля / методу	Опис поля / методу
List <double [,]> Tables	Лист в якому записано заданий список симплекс-таблиць
List <int> MainRows	Лист в якому записано задані номери провідних рядків
List <int> MainCols	Лист в якому записано задані номери провідних стовпців

Список основних полів і методів класу SimplexAlgorithm наведено в таблиці 2.5.

Таблиця 2.5 – Список основних полів і методів класу SimplexAlgorithm

Ім'я поля / методу	Опис поля / методу
double [,] simplextable ;	Задана симплекс таблиця
List <int> BasisValues	Лист в якому записано список базисних змінних
History HistObj	Об'єкт історії розрахунків
double [ ] result	Поле результату
public SimplexAlgorithm (double [,] source)	Конструктор симплекс таблиці
public void Run ( )	Метод запуску симплекс алгоритму
private bool IsItEnd ( )	Метод, критерій закінчення процесу перебору
private int GetMainColumn ( )	Метод, здійснюється пошук ведучого стовпця
private int GetMainRow (int mainCol)	Метод, здійснюється пошук провідною рядки по номеру стовпця

Список основних полів і методів GraphicForm наведено в таблиці 2.6.

Таблиця 2.6 – Список основних полів і методів класу GraphicForm

Ім'я поля / методу	Опис поля / методу
DataTable CoeffTable	Таблиця коефіцієнтів цільової функції

Продовження таблиці 2.6 – Список основних полів і методів класу GraphicForm

Ім'я поля / методу	Опис поля / методу
DataTable InequTable	Таблиця даних по обмеженням
int index	Позиція стовпчика з нерівностями
private void Form1_Load (object sender, EventArgs e)	Метод, завантаження даних в формі
private void button_ClickCreate (object sender, EventArgs e)	Метод, відкриття reateModelForm1
private void button_ClickRun (object sender, EventArgs e)	Метод, запуску рішення
public void CreateTables (int CountConstraints)	Метод, створення таблиць в моделі ЗЛП
public void FindResult ( )	Метод, обчислення значення цільової функції
public void FindPoint ()	Метод, обчислення точки графіка
void ShowTables ( )	Завантаження даних до таблиць
public void CreateClickChart ( )	Метод, побудови графіків
private void button_ClickCreate (object sender, EventArgs e)	Метод, створення таблиць ЗЛП

Продовження таблиці 2.6 – Список основних полів і методів класу GraphicForm

Ім'я поля / методу	Опис поля / методу
private void button_ClickLoad (object sender, EventArgs e)	Метод, завантаження даних ЗЛП до додатку
private void button_ClickNext (object sender, EventArgs e)	Метод, перегляду наступного кроку
private void button_ClickBack (object sender, EventArgs e)	Метод, перегляду попереднього кроку
private void button_ClickSave (object sender, EventArgs e)	Метод, збереження рішення
private void button_ClickInfo (object sender, EventArgs e)	Метод, відкриття інформації про додаток
private void button_ClickUseInfo (object sender, EventArgs e)	Метод, відкриття документації додатку
private void button_ClickClose (object sender, EventArgs e)	Метод, закриття додатку

Список основних полів і методів SimplexForm наведено в таблиці 2.7.

Таблиця 2.7 – Список основних полів і методів класу SimplexForm

Ім'я поля / методу	Опис поля / методу
DataTable CoeffTable	Таблиця коефіцієнтів цільової функції

Продовження таблиці 2.7 – Список основних полів і методів класу SimplexForm

Ім'я поля / методу	Опис поля / методу
DataTable InequTable	Таблиця даних по обиеженням
int index	Позиція стовпчика з нерівностями
DataTable SimplexTable	Симплекс-таблиця
SimplexAlgorithm alg	Об'єкт симплекс-методу
private void Form1_Load (object sender, EventArgs e)	Метод, завантаження даних в формі
private void button_ClickCreate (object sender, EventArgs e)	Метод, відкриття CreateModelForm2
public void CreateTables (int CountConstraints)	Створення таблиць в моделі ЗЛП
void ShowTables ( )	Завантаження даних до таблиць
void Update ( )	Оновлення даних
private void button_ClickRun (object sender, EventArgs e)	Метод, запуску рішення
double [,] ConvertToSimplex (SimplexInfo obj)	Перетворення моделі ЗЛП в матрицю
void FillSimplexTable (double [,] data)	Заповнення симплекс-таблиці даними

Продовження таблиці 2.7 – Список основних полів і методів класу SimplexForm

Ім'я поля / методу	Опис поля / методу
void FillSimplexTable (SimplexAlgorithm alg, int index)	Заповнення симплекс-таблиці даними по історії
private void button_ClickLoad (object sender, EventArgs e)	Метод, завантаження даних ЗЛП до додатку
private void button_ClickNext (object sender, EventArgs e)	Метод, перегляду наступного кроку
private void button_ClickBack (object sender, EventArgs e)	Метод, перегляду попереднього кроку
private void button_ClickMainMenu (object sender, EventArgs e)	Метод, повертання до головного меню
private void button_ClickSave (object sender, EventArgs e)	Метод, збереження рішення
private void button_ClickInfo (object sender, EventArgs e)	Метод, відкриття інформації про додаток
private void button_ClickUseInfo (object sender, EventArgs e)	Метод, відкриття документації додатку
private void button_ClickClose (object sender, EventArgs e)	Метод, закриття додатку

Список основних полів і методів DvoistaForm наведено в таблиці 2.8.

Таблиця 2.8 – Список основних полів і методів класу DvoistaForm

Ім'я поля / методу	Опис поля / методу
DataTable CoeffTable	Таблиця коефіцієнтів цільової функції
DataTable InequTable	Таблиця даних по обмеженням
DataTable CoeffTable2	Таблиця коефіцієнтів цільової функції двоїстої задачі
DataTable InequTable2	Таблиця даних по Обмеженням двоїстої задачі
int index	Позиція стовпчика з нерівностями
private void Form1_Load (object sender, EventArgs e)	Метод, завантаження даних в формі
private void button_ClickCreate (object sender, EventArgs e)	Метод, відкриття CreateModelForm3
private void button_ClickRun (object sender, EventArgs e)	Метод, запуску рішення
public void CreateTables (int CountConstraints)	Створення таблиць в моделі ЗЛП
public void CreateFuncTableDvoista ( )	Метод, створення цільової функції двоїстої задачі, зміна значень між цільової функції та правою частиною обмежень

Продовження таблиці 2.8 – Список основних полів і методів класу DvoistaForm

Ім'я поля / методу	Опис поля / методу
public void CreateTableDvoista ( )	Метод, створення матриці двоїстої задачі
public void LogicTable ( )	Метод, в залежності від вибраного знаку нерівностей створюється матриця
private void button_ClickCreate (object sender, EventArgs e)	Метод, створення таблиць ЗЛП
private void button_ClickNext (object sender, EventArgs e)	Метод, перегляду наступного кроку
private void button_ClickBack (object sender, EventArgs e)	Метод, перегляду попереднього кроку
private void button_ClickMainMenu (object sender, EventArgs e)	Метод, повертання до головного меню
private void button_ClickSave (object sender, EventArgs e)	Метод, збереження рішення
private void button_ClickInfo (object sender, EventArgs e)	Метод, відкриття інформації про додаток
private void button_ClickUseInfo (object sender, EventArgs e)	Метод, відкриття документації додатку
private void button_ClickClose (object sender, EventArgs e)	Метод, закриття додатку

Список основних полів і методів DvoistSimpForm наведено в таблиці 2.9.

Таблиця 2.9 – Список основних полів і методів класу DvoistSimpForm

Ім'я поля / методу	Опис поля / методу
DataTable CoeffTable	Таблиця коефіцієнтів цільової функції
DataTable InequTable	Таблиця даних по обмеженням
int index	Позиція стовпчика з нерівностями
DataTable SimplexTable	Симплекс-таблиця
SimplexAlgorithm alg	Об'єкт симплекс-методу
private void Form1_Load (object sender, EventArgs e)	Метод, завантаження даних в формі
Ім'я поля / методу	Опис поля / методу
private void button_ClickCreate (object sender, EventArgs e)	Метод, відкриття CreateModelForm4
public void CreateTables (int CountConstraints)	Створення таблиць в моделі ЗЛП
void ShowTables ( )	Завантаження даних до таблиць
void Update ( )	Оновлення даних

Продовження таблиці 2.9 – Список основних полів і методів класу DvoistSimpForm

Ім'я поля / методу	Опис поля / методу
private void button_ClickRun (object sender, EventArgs e)	Метод, запуску рішення
double [,] ConvertToSimplex (SimplexInfo obj)	Перетворення моделі ЗЛП в матрицю
void FillSimplexTable (double [,] data)	Заповнення симплекс-таблиці даними
void FillSimplexTable (SimplexAlgorithm alg, int index)	Заповнення симплекс-таблиці даними по історії
public void FindFunction ( )	Метод, обчислення значення цільової функції
public void FuncChangeTableDvoista ( )	Метод, створення цільової функції двоїстої задачі, зміна значень між цільової функції та правою частиною обмежень
public void IneqChangeTableDvoista ( )	Метод, в залежності від вибраного знаку нерівностей створюється матриця
private void button_ClickLoad (object sender, EventArgs e)	Метод, завантаження даних ЗЛП до додатку
private void button_ClickNext (object sender, EventArgs e)	Метод, перегляду наступного кроку
private void button_ClickBack (object sender, EventArgs e)	Метод, перегляду попереднього кроку

Продовження таблиці 2.9 – Список основних полів і методів класу DvoistSimpForm

Ім'я поля / методу	Опис поля / методу
private void button_ClickMainMenu (object sender, EventArgs e)	Метод, повертання до головного меню
private void button_ClickSave (object sender, EventArgs e)	Метод, збереження рішення
private void button_ClickInfo (object sender, EventArgs e)	Метод, відкриття інформації про додаток
private void button_ClickUseInfo (object sender, EventArgs e)	Метод, відкриття документації додатку
private void button_ClickClose (object sender, EventArgs e)	Метод, закриття додатку

Список основних полів і методів InfoForm наведено в таблиці 2.10.

Таблиця 2.10. Список основних полів і методів класу InfoForm

Ім'я поля / методу	Опис поля / методу
button1_Click (object sender, EventArgs e)	Метод, опис інформації про додаток та закриває форму

Список основних полів і методів UseInfoForm наведено в таблиці 2.11.

Таблиця 2.11. Список основних полів і методів класу UseInfoForm

Ім'я поля / методу	Опис поля / методу
button1_Click (object sender, EventArgs e)	Метод, документація використання додатка та закриває форму

Список основних полів і методів CreateModelForm1, CreateModelForm2, CreateModelForm3, CreateModelForm 4 наведено в таблиці 2.12.

Таблиця 2.12 – Список основних полів і методів класу CreateModelForm1, CreateModelForm2, CreateModelForm3, CreateModelForm4

Ім'я поля / методу	Опис поля / методу
button1_Click (object sender, EventArgs e)	Метод, задання кількісті обмежень та / або змінних та за допомогою цих даних створюється цільова та сиплекс таблиця

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ»

Для того, щоб повністю зрозуміти рішення певної задачі лінійного програмування, важливий не тільки результат, а й хід рішення, що призведе до отримання правильного результату.

Отже, був створений додаток за допомогою якого можливо побачити не тільки результат заданих даних, але й послідовні кроки рішення, реалізація яких веде до отримання відповіді.

3.1 Інструкція роботи з програмою графічний метод розв'язування задач лінійного програмування

Відкриваючи програму, з'являється головне меню програми. В головному меню наглядаємо вибір з чотирьох методів, а саме графічний метод, симплекс метод, двоїста задача та двоїстий симплекс метод (Рис. 3.1).

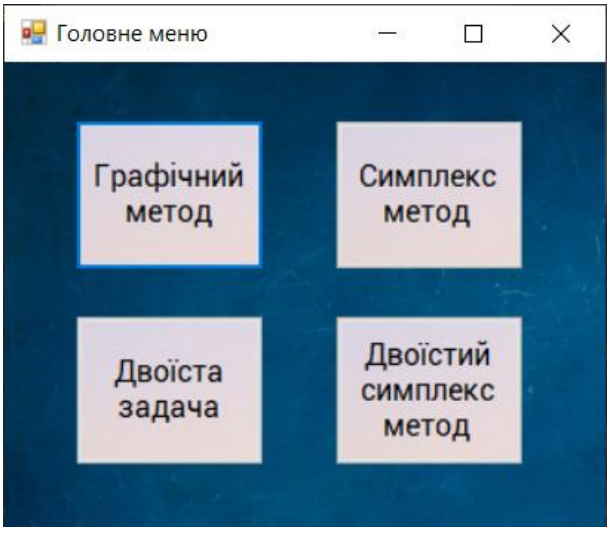


Рисунок 3.1 – Головного меню програми

Вибираємо графічний метод розв’язування задачі ЗЛП, відкривається вікно графічного методу, яке зображено на рисунку 3.2.

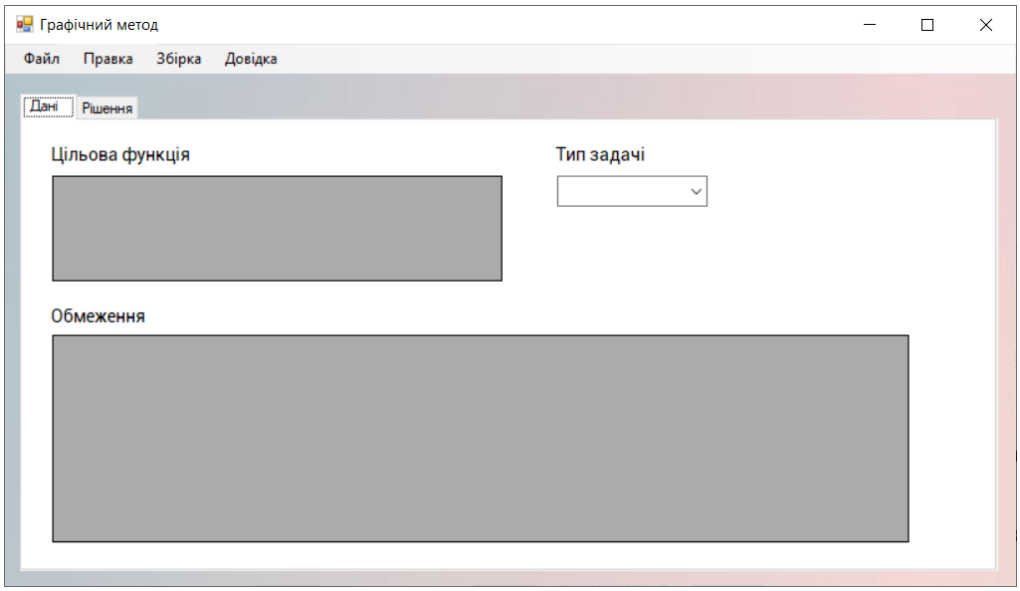


Рисунок 3.2 – Вікно графічного методу програми

Для того, щоб створити таблиці з даними цільової функції та обмеженнями, натискаємо «Файл» - «Створити метод».

Для зразка розглянемо задачу:

$$F = 3x_1 + 2x_2 \rightarrow max$$

$$\begin{cases} x_1 + x_2 \leq 6 \\ x_1 - 2x_2 \leq 3 \end{cases}$$

Відкривається вікно створити таблиці та вказуємо потрібку кількість обмежень. В даному прикладі вибираємо два (Рис. 3.3).

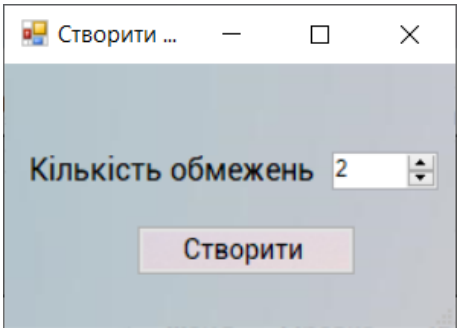


Рисунок 3.3 – Створити таблиці

Створюються таблиці. Вказуємо дані цільової функції, обмежень та вибираємо знак нерівності, а саме більше рівно, менше рівно чи рівно. Для задачі, яку розглядаємо, вожу відповідні значення (Рис. 3.3).

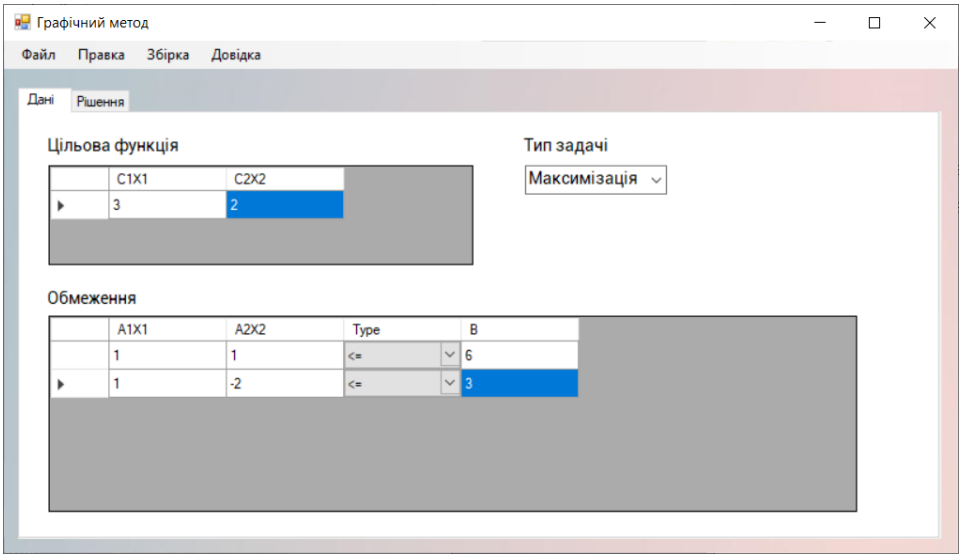


Рисунок 3.3 –Вікно графічного методу програми з введеними даними

Для того, щоб отримати рішення ЗЛП, переходимо до вкладки рішення та натискаємо «Збірка» - «Зібрати рішення» (Рис. 3.4).

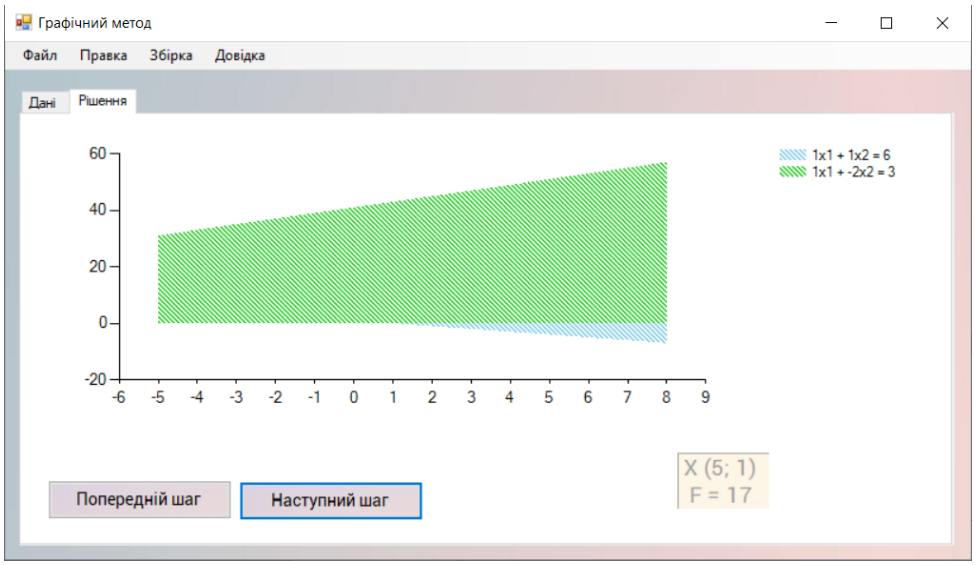


Рисунок 3.4 – Вікно рішення графічного методу з рішенням прикладу

В оранжевому квадраті отримали розв’язок графічного методу, а саме точку X та значення цільової функції при якому вона набуває максимума за умовою задачі.

За допомогою кнопок «Попередній шаг» та «Наступний шаг» можемо подивитися рішення покроково, графік прямих та їх областей, де утворилися.

Також можливо створювати задачу не вводячи дані, а загружаючи їх з файлу формату «.txt» і «.csv». Для цього потрібно створити текстовий файл, який складається з типу задачі (максимум чи мінімум), значень цільової функцій та обмеження з вказанням типу.

Для зразка візьмемо попередню задачу тільки вже загрузимо його з текстового файлу (Рис. 3.5) та робимо аналогічні дії з розв’язуванням (Рис. 3.6).

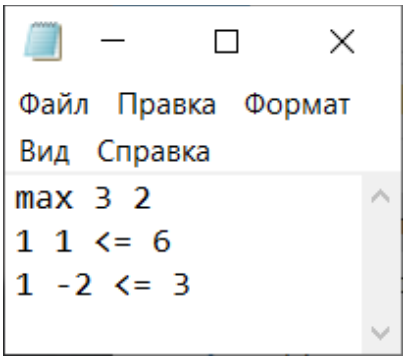


Рисунок 3.5 – Текстовий файл з прикладом

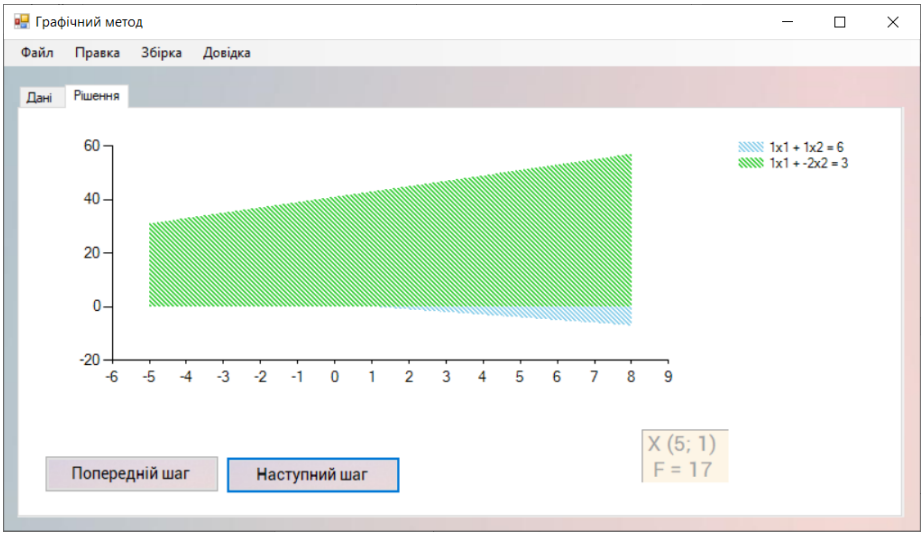


Рисунок 3.6 – Вікно рішення графічного методу з рішенням прикладу

Також є можливість зберегти результат рішення графічного методу лінійного програмування в текстовий файл формату «.txt». Для цього натискаємо «Файл» - «Зберегти рішення» та в модальному вікні, яке з'явилося вказуємо назву, місце зберігання даного файлу.

3.2 Інструкція роботи з програмою симплекс метод, двоїстий симплекс метод розв'язання задач лінійного програмування

Для початку обчислення симплекс метода виконуємо алгоритм дій аналогічний до підпункту 3.1.

Для зразка розглянемо рішення симплекс методу:

$$F = 4x_1 + 2x_2 \rightarrow min$$

$$5x_1 - 8x_2 \leq 16$$

$$x_1 + 3x_2 \geq -2$$

$$7x_1 + 2x_2 \leq 9$$

Створюємо таблиці, вказуючи кількість змінних та обмежень. Далі вибираємо тип задачі, вводимо дані зі зразка до цільової функцій та обмежень вказуючи тип обмежень (Рис. 3.7).

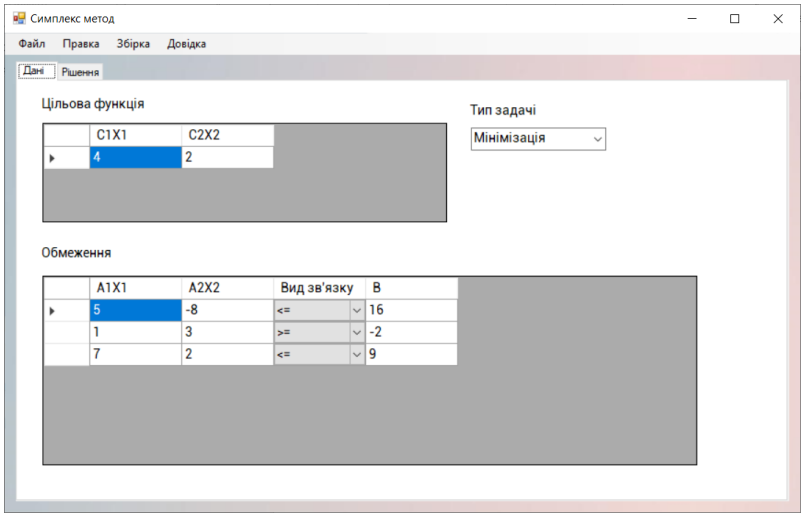


Рисунок 3.7 –Вікно симплекс методу програми з введеними даними

Для того, щоб отримати рішення ЗЛП, переходимо до вкладки рішення та натискаємо «Збірка» - «Зібрати рішення» (Рис. 3.8, 3.9).

В симплекс матриці отримали покрокове розв’язання таблиць з створеним базисом симплекс методу.

В векторі рішення отримали опорний (допустимий) план розв’язання симплекс методу.

На чистому полі спостерігаємо отримане рішення, а саме значення цільової функції при заданому плані - мінімум за умовою задачі.

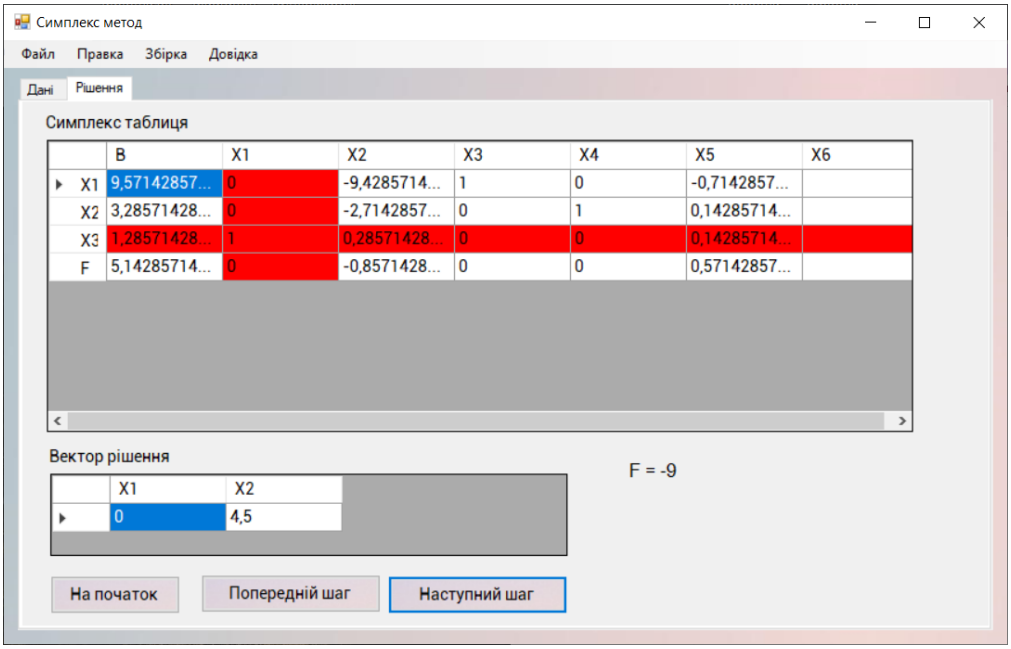


Рисунок 3.8 – Вікно рішення симплекс методу з рішенням прикладу

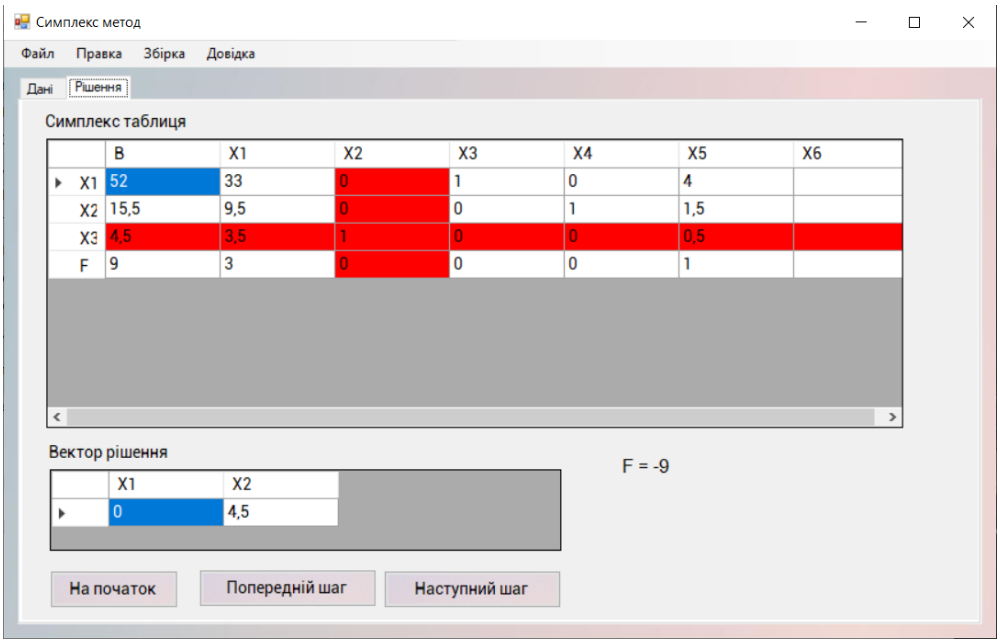


Рисунок 3.9 – Вікно рішення симплекс методу з рішенням прикладу

За допомогою кнопки «На початок» можемо подивитися першу симплекс таблицю, тобто вернутися на самий початок.

За допомогою кнопок «Попередній шаг» та «Наступний шаг» можемо подивитися рішення покроково, де червоним кольором вказано обраний розв’язувальний елемент.

Також можливо створювати задачу не вводячи дані, а загружаючи їх з файлу формату «.txt» і «.csv». Для цього потрібно створити текстовий файл, який складається з типу задачі (максимум чи мінімум), значень цільової функцій та обмеження з вказанням типу. Аналогічно до графічного методу.

Для зразка візьмемо попередню задачу тільки вже загрузимо його з текстового файлу (Рис. 3.10) та робимо аналогічні дії з розв’язуванням (Рис. 3.11).

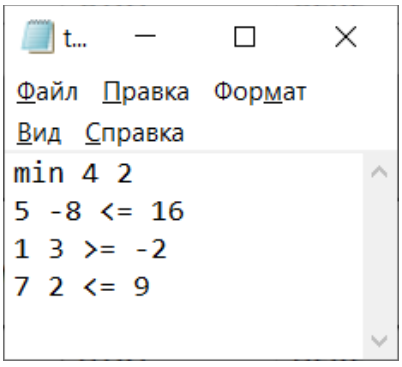


Рисунок 3.10 –Текстовий файл з прикладом

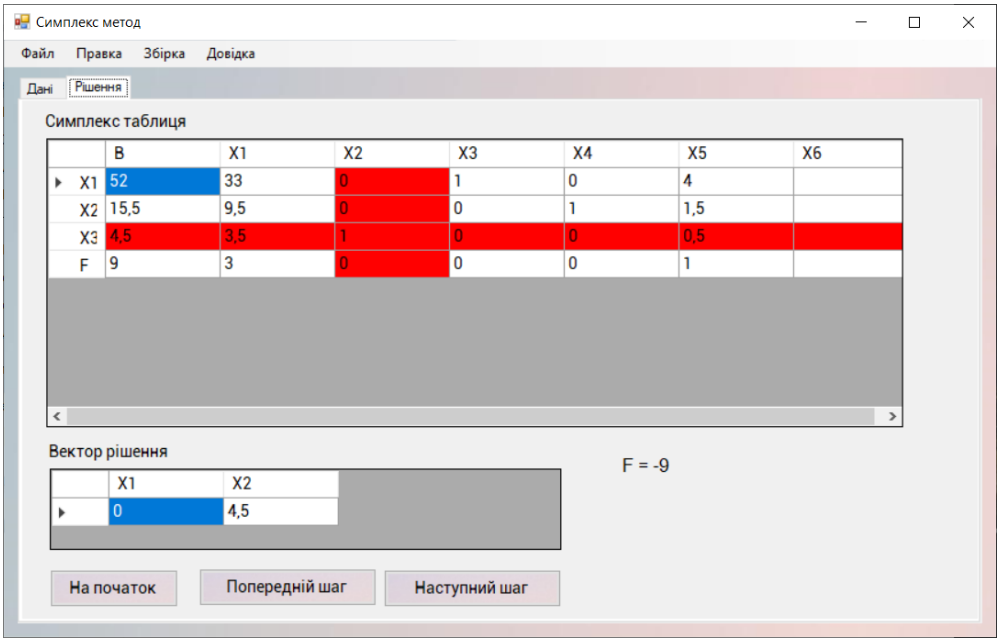


Рисунок 3.11 – Вікно рішення симплекс методу з рішенням прикладу

Є можливість зберегти результат рішення симплекс методу чи двоїстого симплекс методу лінійного програмування в текстовий файл формату «.txt». Для цього натискаємо «Файл» - «Зберегти рішення» та в модальному вікні, яке з’явилося вказуємо назву, місце зберігання даного файлу.

Якщо потрібно повернутися до головного меню натискаємо «Правка» - «Повернутися до головного меню», також через «Правка» є можливість переглядати шаги рішення, а саме «Повернутися до початку методу», «Наступний шаг рішення» та «Попередній шаг рішення методу».

Також можливо не тільки зібрати рішення, але й очистити таблиці з можливістю створення нової задачі. Для цього натискаємо «Збірка» - «Очистити рішення».

Якщо з’являються питання як працювати в додатку натискаємо «Довідка» - «Документація», відкривається вікно з наявним список питань щодо роботи даного додатку. Далі знаходимо потрібне питання та робимо алгоритм дій, який вказаний в вікні «Документація» (Рис. 3.12).

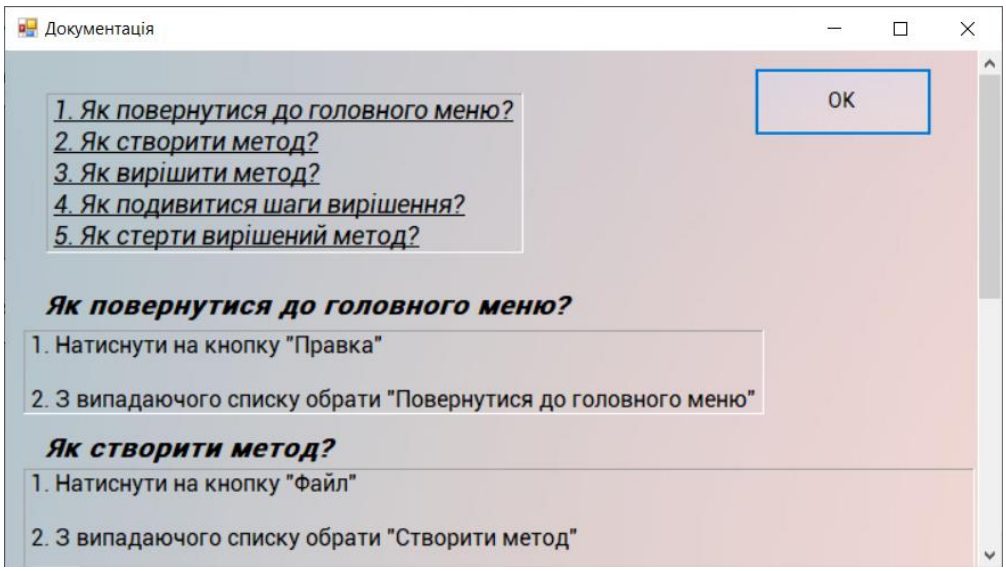


Рисунок 3.12 – Документація додатка

Якщо цікавить інформація щодо самого додатку натискаємо «Довідка» - «Про додаток» (Рис. 3.13).

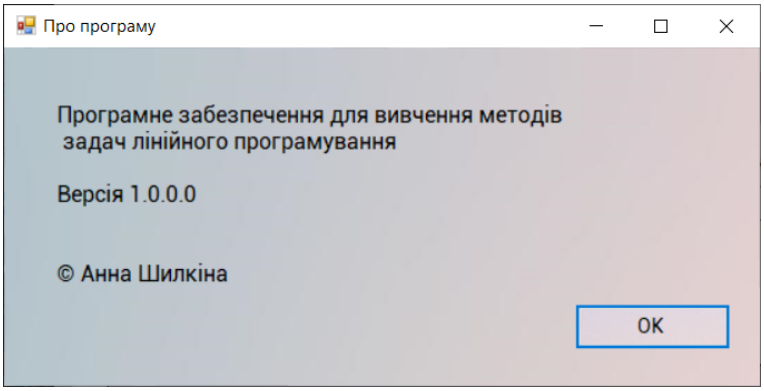


Рисунок 3.13 — Про програму

3.3 Інструкція роботи з програмою двоїста задача розв'язання задач лінійного програмування

Для початку обчислення двоїстої задачі виконуємо алгоритм дій аналогічний до підпункту 3.1.

Для зразка розглянемо рішення:

$$F = 4x_1 + 2x_2 \rightarrow min$$

$$5x_1 - 8x_2 \leq 16$$

$$7x_1 + 2x_2 \leq 9$$

Створюємо таблиці, вказуючи кількість змінних та обмежень. Далі вибираємо тип задачі, вводимо дані зі зразка до цільової функцій та обмежень вказуючи тип обмежень (Рис. 3.14).

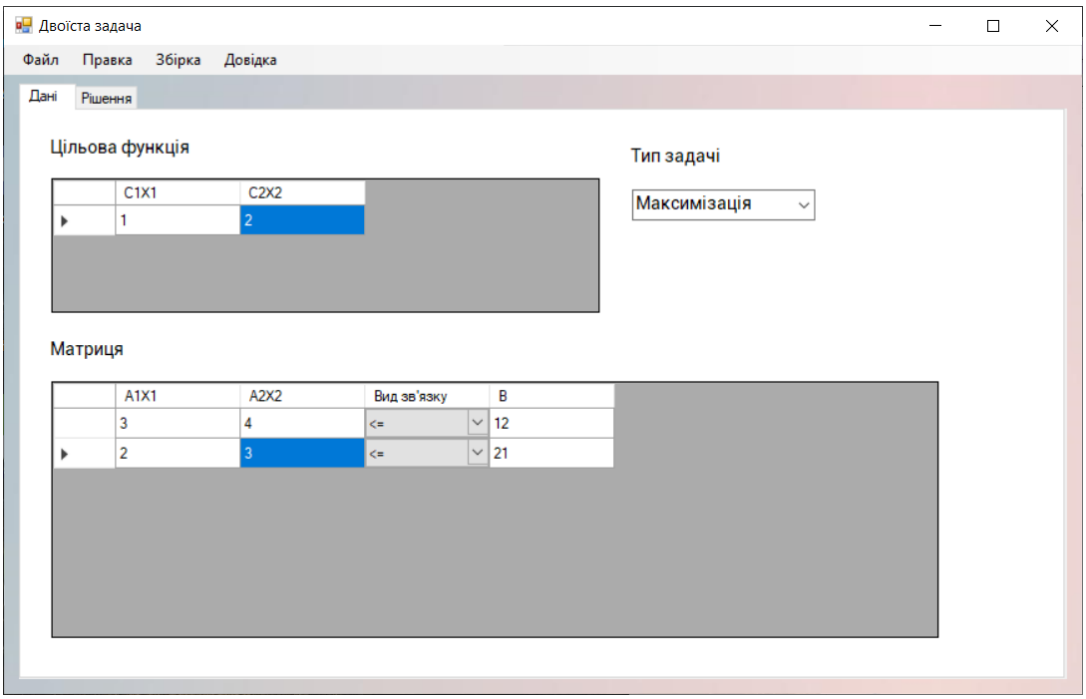


Рисунок 3.14 –Вікно двоїстої задачі програми з введеними даними

Для того, щоб отримати рішення ЗЛП, переходимо до вкладки рішення та натискаємо «Збірка» - «Зібрати рішення» (Рис. 3.15).

В цільовій функції отримали значення двоїстої задачі лінійного програмування.

В матриці отримали значення двоїстої задачі лінійного програмування.

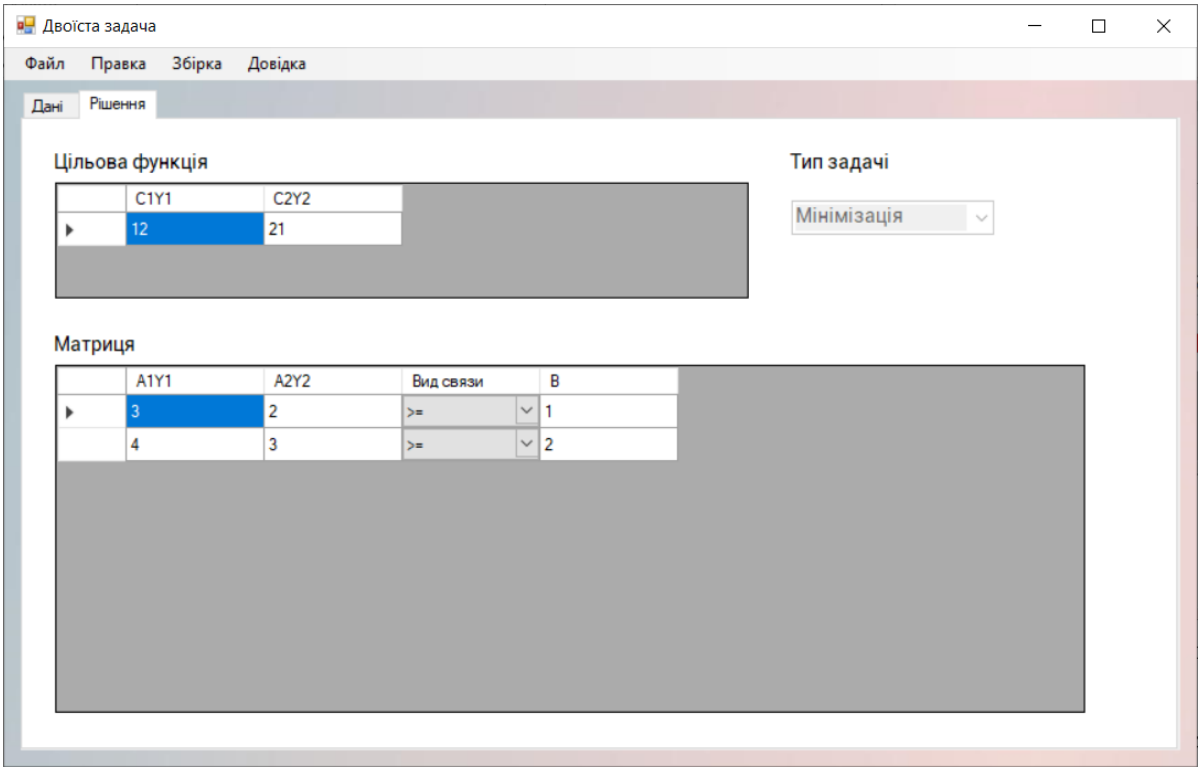


Рисунок 3.15 – Вікно рішення двоїстої задачі з рішенням прикладу

Є можливість зберегти результат рішення двоїстої задачі лінійного програмування в текстовий файл формату «.txt». Для цього натискаємо «Файл» - «Зберегти рішення» та в модальному вікні, яке з’явилося вказуємо назву, місце зберігання даного файлу.

Якщо потрібно повернутися до головного меню натискаємо «Правка» - «Повернутися до головного меню».

Також можливо не тільки зібрати рішення, але й очистити таблиці з можливістю створення нової задачі. Для цього натискаємо «Збірка» - «Очистити рішення».

Якщо з’являються питання як працювати в додатку натискаємо «Довідка» - «Документація», відкривається вікно з наявним список питань щодо роботи

даного додатку. Далі знаходимо потрібне питання та робимо алгоритм дій, який вказаний в вікні «Документація» (Рис. 3.16).

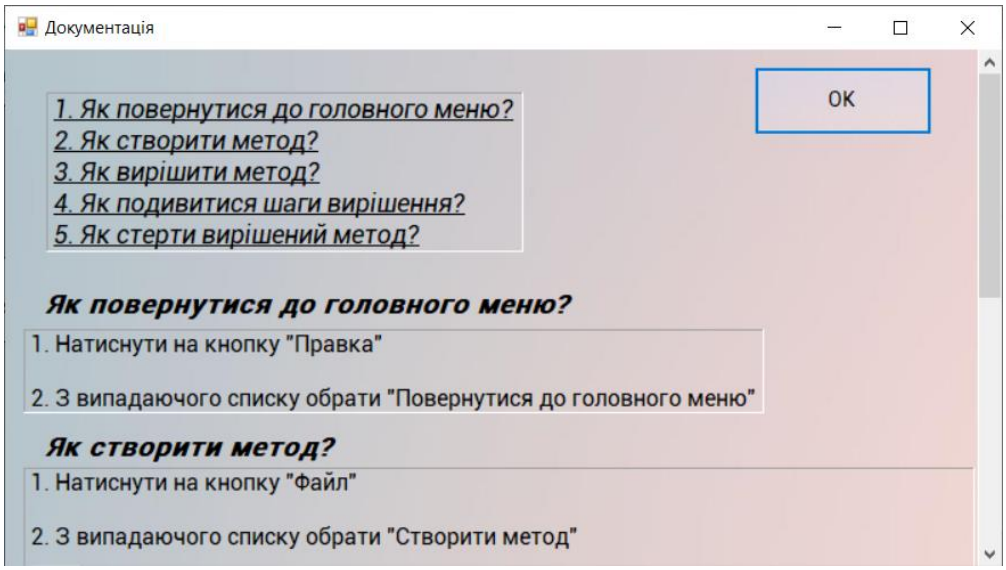


Рисунок 3.16 –Документація додатка

Якщо цікавить інформація щодо самого додатку натискаємо «Довідка» - «Про додаток» (Рис. 3.17).

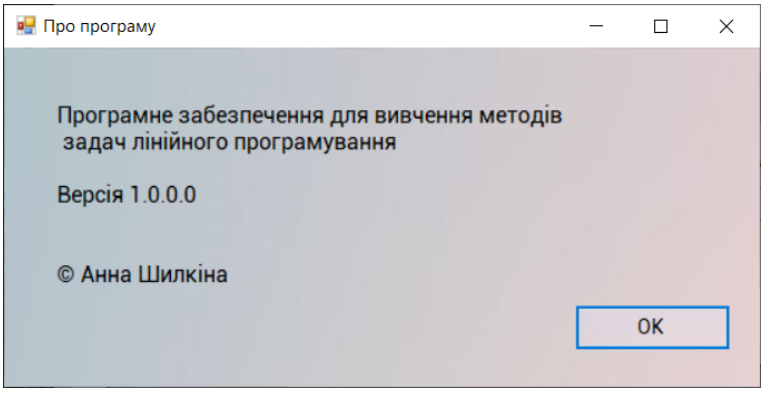


Рисунок 3.17 — Про програму

ВИСНОВКИ

Отже, виконуючи бакалаврську роботу мені вдалося проаналізувати предметну область дисципліни «Дослідження операцій», здійснити огляд методів лінійного програмування та проаналізувати інтегроване середовище розробки Microsoft Visual Studio, дослідити новітні підходи до вивчення дисципліни у вищій школі.

Особливе місце у вивчені дисципліни «Дослідження операцій» займають методи лінійного програмування, а саме симплекс метод, графічний, двоїста задача та двоїстий симплекс метод.

Під час виконання роботи мною було розроблено програмне забезпечення для покрокового вивчення дисципліни «Дослідження операцій» у вищій школі. Програмний продукт розроблений в середовищі Microsoft Visual Studio 2019, об'єктно-орієнтованою мовою програмування C#, за допомогою програмної платформи .NET Framework.

Додатковими можливостями ПЗ є:

- Завантаження файлів з комп'ютера користувача у текстовому форматі «.txt» та «.csv».
- Можливість завантаження готового результату.
- Наявність документації щодо використання розробленого ПЗ.

Програмний продукт має перспективи розвитку. Зокрема, він може бути перенесений на платформу ASP.NET, що дозволить застосовувати його на корпоративному рівні.

Вивчаючи дану дисципліну у вищій школі майбутні програмісти отримають знання, які в подальшому можуть використовувати у своїй роботі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Грязнов В.О., Єфіменко С.В. Основи програмування. Мова C#. Методичний посібник .– К.: КНУ, 2009.- 145 с.

2. Глоба Л.С. Математичні основи побудови інформаційно телекомунікаційних систем.-К.: Норіта-плюс, 2007.-360 с.

3. Єсіна В.О. Оптимізації методи і моделі. Конспект лекцій з дисципліни. Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2016. – 64 с.

4. Зайченко Ю.П. Дослідження операцій : Підручник. – К.: Вища школа, 1988. – 552с.

5. Зайченко О.Ю., Зайченко Ю. П. Дослідження операцій. Збірник задач. – К.: Видавничий Дім “Слово”, 2007.- 472 с.

6. Згуровський М.З., Панкратова Н.Д. Основи системного аналізу. - К.: Видавнича група BHV, 2007.-544с.

7. Ільченко М.Ю.. Кравчук С.О. Сучасні телекомунікаційні системи. - К.: НВП «Видавництво «Наукова думка» НАН України», 2008.- 328 с.

8. Калюжний О.Я. Моделювання систем передачі сигналів в обчислювальному середовищі MATLAB-Simulink: Навч. Посібник. –К.: ІВЦ «Видавництво «Політехніка»», 2004. - 136 с.

9. Кутовецький В.Я. Дослідження операцій. Навчальний посібник. – Миколаїв: Вид-во МДГУ ім. П. Могили, 2003. – 260 с.

10. Лавров Є. А., Перхун Л. П., Шендрик В. В. та ін. Математичні методи дослідження операцій : підручник / – Суми : Сумський державний університет, 2017. – 212 с.

11. Ларіонов Ю.І., Левикін В.М., Хажмурадов М.А. Дослідження операцій в інформаційних системах.-Харків.: Компанія СМІТ, 2005.-364 с.

12. Лисенко О.І, Алєксєєва І.В. Дослідження операцій. Конспект лекцій К: НТУУ «КПИ», 2016. – 196 с. 113

13. Ляшенко И.Н. и др. Линейное и нелинейное програмирование. – К.: Вища шк., 1975. – 372 с.

14. Онищенко В.В. Дослідження операцій. Частина 1. Навчально-методичний посібник.-К.: Вища школа., 2016.—44с.

15. Табунщик Г.В. Проектування та моделювання програмного забезпечення сучасних інформаційних систем. Навчальний посібник. Запоріжжя, 2016.- 245

16. Томашевський В.М. Моделювання систем. Підручник. -К.: Видавнича група BHV, 2007.- 352 с.

17. Шабанова Ю.О. Системний підхід у вищій школі : підруч. для студ. Магістратури. Донецьк : НГУ, 2014. 120 с

18. Chris Sells . Windows Forms Programming in C# .Publisher(s): Addison-Wesley Professional. August 2003

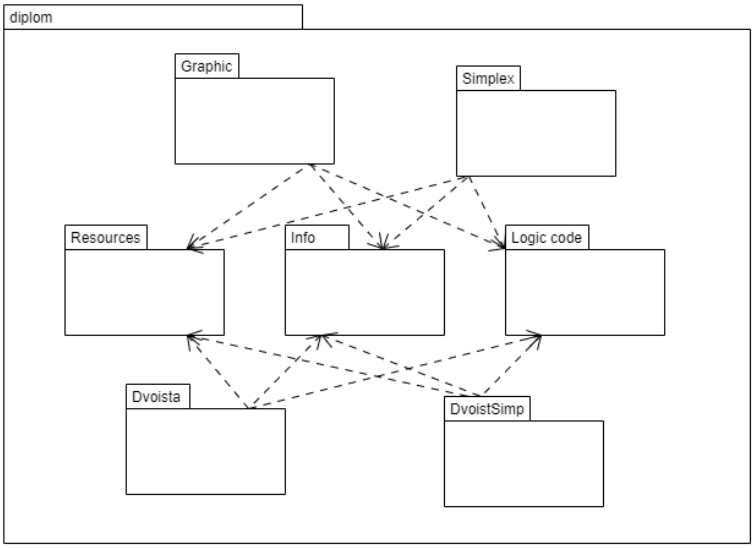
19. Erik Brown . Windows Forms in Action Second Edition of Windows Forms Programming with C#. April 2006.- 950 pages

20. Rob Miles. The C# Programming Yellow Book . July 2015. Department of Computer Science University of Hull

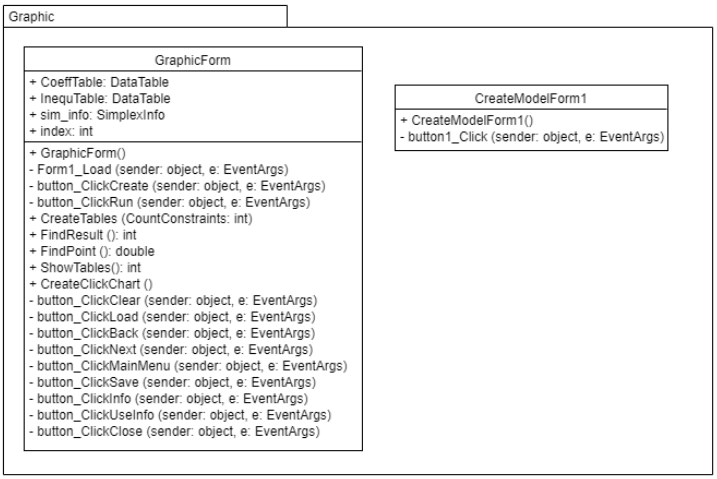
21. Max Beerbohm. Visual C#.NET: Windows Forms Programming with C#. 2019

22. IEEE Standard Glossary of Software Engineering Terminology,

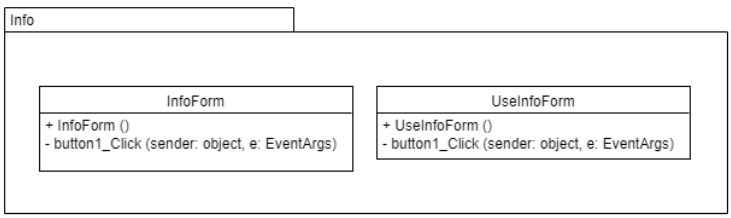
Додаток А



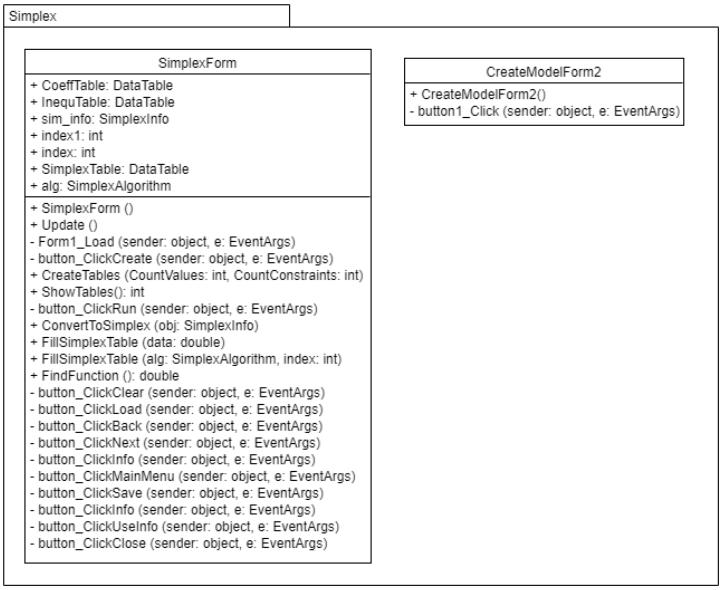
Пакет **Graphic:**



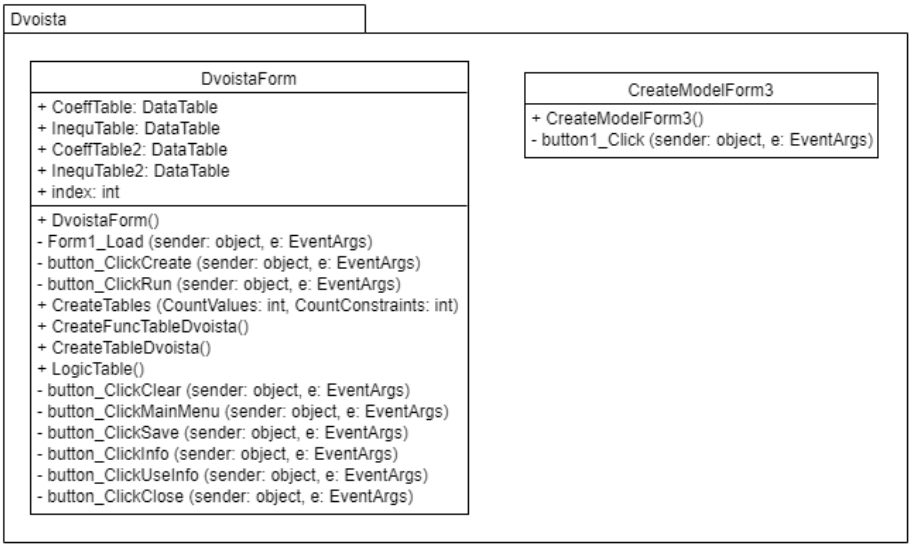
Пакет **Info:**



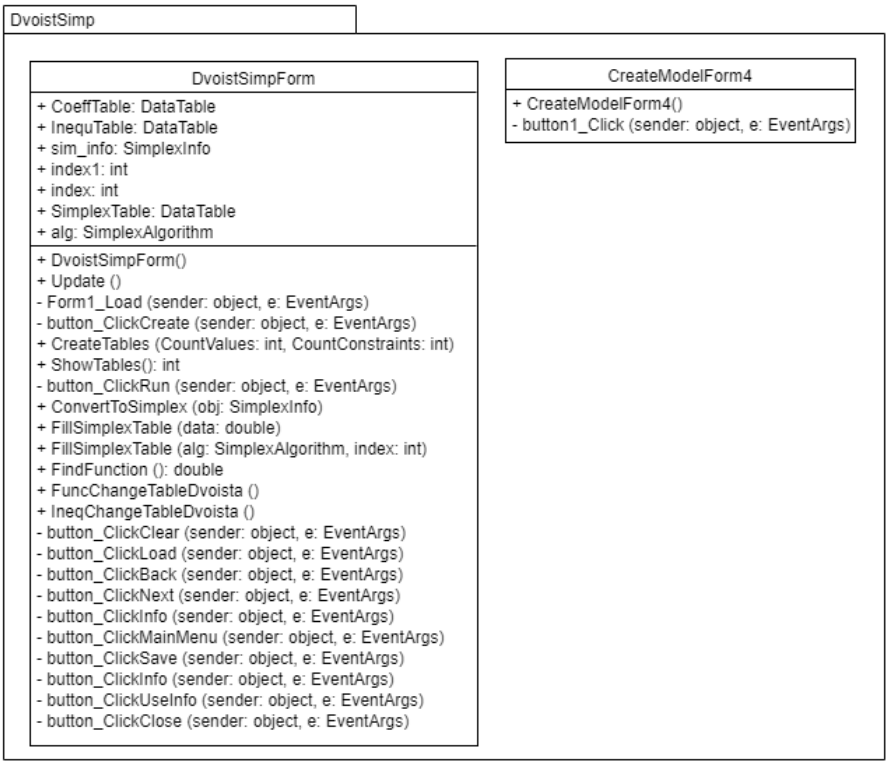
Пакет **Simplex:**



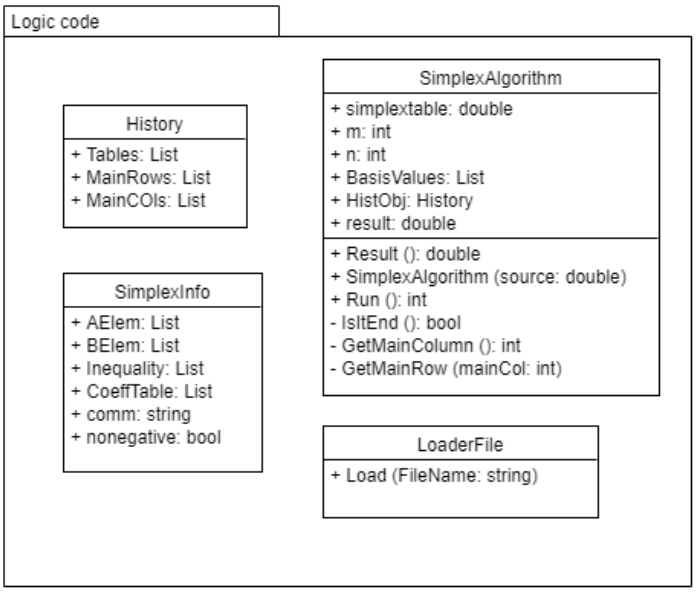
Пакет **Dvoista:**



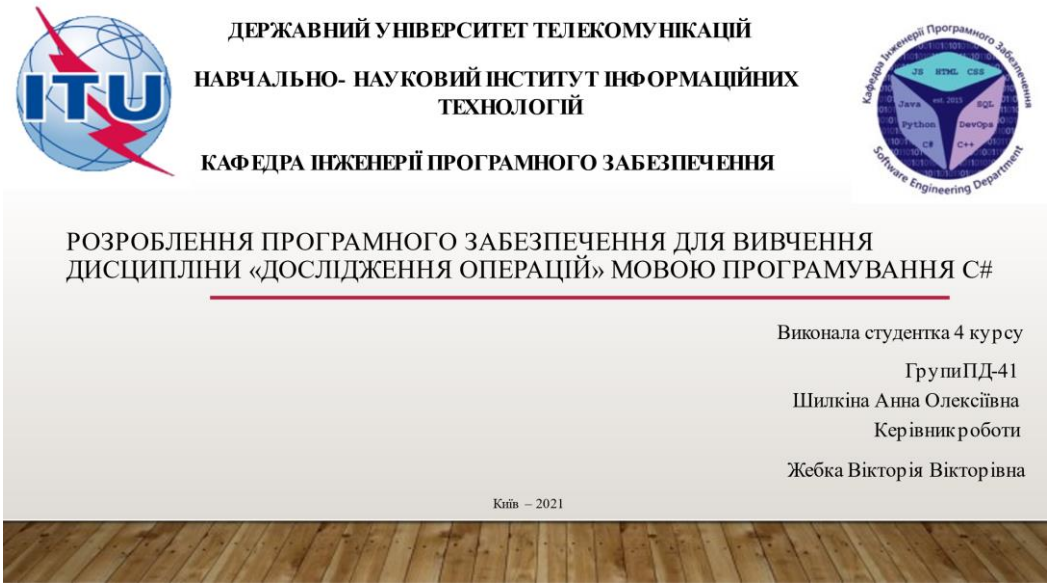
Пакет **DvoistSimp:**

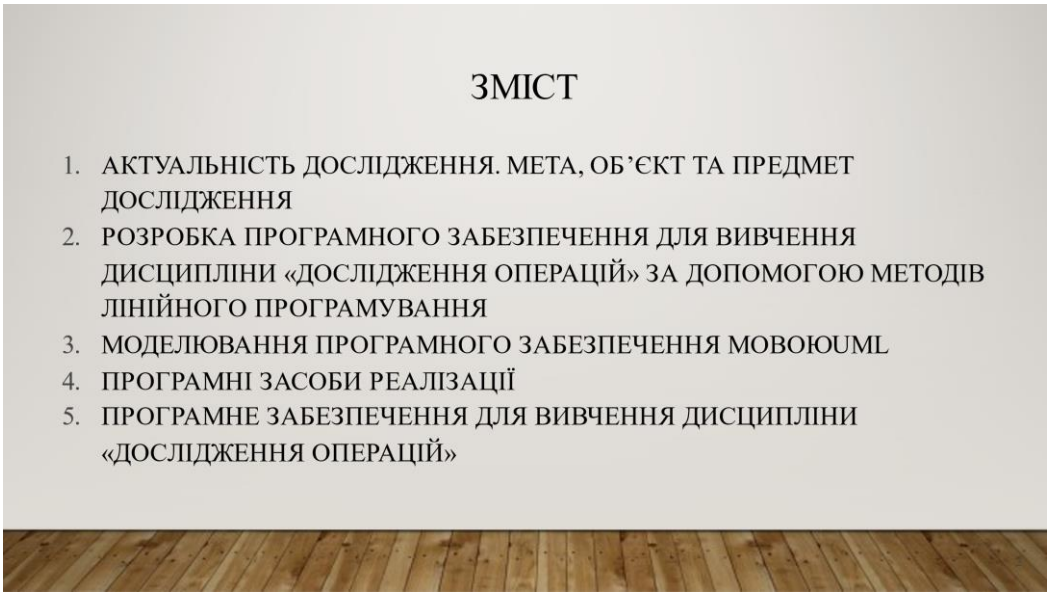


Пакет **Logic code:**



ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)





АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ. МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Фахівці із інженерії програмногозабезпечення та комп'ютерних наук, що мають знання та навички з проектування, моделювання та аналізу програмного забезпечення мають здійснити кардинальний перевороту розвиткуекономіки, науки, освіти та інших галузяхжиттєдіяльності суспільства.

- **Метою дослідження є** визначення новітніх підходів до методів викладання «Дослідження операцій» для майбутніх програмістів, формування ролі знань і вмінь з дослідження операцій у підготовці фахівців з комп'ютерних наук за допомогою розробленогопрограмногозабезпечення.
- **Предметом дослідження є** програмне забезпечення для вивчення дисципліни «Дослідження операцій».
- **Об'єктом досліджень є** викладання дисципліни «Дослідження операцій» у вищій школі за допомогою математичних моделей виробничих систем та застосуванню практичних методів для їх оптимального управління.

3

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» ЗА ДОПОМОГОЮ МЕТОДІВ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Для розв'язку двовимірних ЗЛП, тобто задач з двома змінними, а також деяких тривимірних задач застосовують**графічний метод,**що грунтується на геометричній інтерпретації та аналітичних властивостях ЗЛП.

Знайти екстремум функції:

$$F = c_1x_1 + c_2x_2 \rightarrow \max(\min)$$

за умов

$$\begin{cases} a_{11}x_1 + a_{12}x_2\{\leq, =, \geq\}b_1 \\ \dots\dots\dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2\{\leq, =, \geq\}b_m \end{cases}$$

$$x_1 \geq 0, x_2 \geq 0$$

4

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» ЗА ДОПОМОГОЮ МЕТОДІВ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Симплекеметод– це поетапна обчислювальна процедура, в основуякої покладено принцип послідовного поліпшення значень цільової функції переходом від одного опорного планузадачі лінійного програмуваннядо іншого

Цільовафункція з системою обмежень має вигляд:

$$F = c_1x_1 + c_2x_2 \rightarrow \max(\min)$$
$$a_{11}x_1 + a_{12} = b_1$$
$$\dots\dots\dots\dots\dots$$
$$a_{m1}x_1 + a_{m2}x_2 = b_m$$
$$x_1 \geq 0, x_2 \geq 0$$

Канонічна форма цільової функції має вигляд:

$$F = c_1x_1 + \dots + c_nx_n \rightarrow max$$
$$\begin{cases} a_{11}x_1 + \dots + a_{1n}x_n + u_1 = b_1 \\ \dots\dots\dots\dots\dots \\ a_{m1}x_1 + \dots + a_{mn}x_n + u_m = b_m \end{cases}$$
$$x_i \geq 0, b_i \geq 0, u_i \geq 0$$

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» ЗА ДОПОМОГОЮ МЕТОДІВ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Симплекс таблиця

№	Базис	C_b	P_0	c_1	c_2	…	c_k	…	c_m	c_{m+1}	c_{m+2}	…	c_n
				p_1	p_2	…	p_k	…	p_m	p_{m+1}	p_{m+2}	…	p_n
1	p_{m+1}	0	b_1	a_{11}	a_{12}	…	a_{1k}	…	a_{1m}	1	0	…	0
2	p_{m+2}	0	b_2	a_{21}	a_{22}	…	a_{2k}	…	a_{2m}	0	1	…	0
3	p_{m+3}	0	b_3	a_{31}	a_{32}	…	a_{3k}	…	a_{3m}	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…
n	p_n	0	b_n	a_{n1}	a_{n2}	…	a_{nk}	…	a_{nm}	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…
m	p_m	0	b_m	a_{m1}	a_{m2}	…	a_{mk}	…	a_{mm}	0	0	…	1
m+1			F_0	Δ_1	Δ_2	…	Δ_k	…	Δ_m	Δ_{m+1}	Δ_{m+2}	…	Δ_n

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» ЗА ДОПОМОГОЮ МЕТОДІВ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Пряма задача лінійного програмування визначає скільки необхідно виготовити продукту із заданих ресурсів з метою максимізації загальної виручки. Та має такий вигляд:

$$f = c_1, c_2, \dots, c_n \rightarrow max$$

$$m\left\{\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} * \begin{pmatrix} x_1 \\ \dots \\ x_n \end{pmatrix} \begin{matrix} \leq b_1 \\ \dots \\ \leq b_m \end{matrix}\right.$$

На відміну від прямої задачі в **двоїстій задачі** лінійного програмування необхідно визначити ціни ресурсів. Кожному ресурсу (b_i) ставимо відповідну оцінку (y_i) ціни за одиницю ресурсу.

Має вигляд:

$$f = b_1, b_2, \dots, b_m \rightarrow min$$

$$m\left\{\begin{pmatrix} a_{11} & a_{12} & \dots & a_{m1} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{2n} & \dots & a_{mn} \end{pmatrix} * \begin{pmatrix} y_1 \\ \dots \\ y_m \end{pmatrix} \begin{matrix} \leq c_1 \\ \dots \\ \leq c_n \end{matrix}\right.$$

7

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» ЗА ДОПОМОГОЮ МЕТОДІВ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Матриця коефіцієнтів прямої задачі транспортується n⇔m, де n – рядок, m – стовпчик.

Тобто матриця прямої задачі має вигляд:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

Матриця двоїстої задачі лінійного програмування має вигляд:

$$A^T = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix}$$

8

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ» ЗА ДОПОМОГОЮ МЕТОДІВ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Двоїстий симплекеметод безпосередньо застосовується до розв'язування майже канонічної задачі лінійного програмування (МКЗЛП), яка формулюється таким чином

Знайти вектор x = (x_1,...,x_n), що мінімізує лінійну функцію

$$L(x) = c_1 x_2 + \cdots + c_n x_n$$

і задовольняє систему лінійних обмежень

$$x_i + a_{i,m} + 1x_m + 1 + \cdots + a_{in} x_n = a_i 0, i = 1, \dots, m,$$

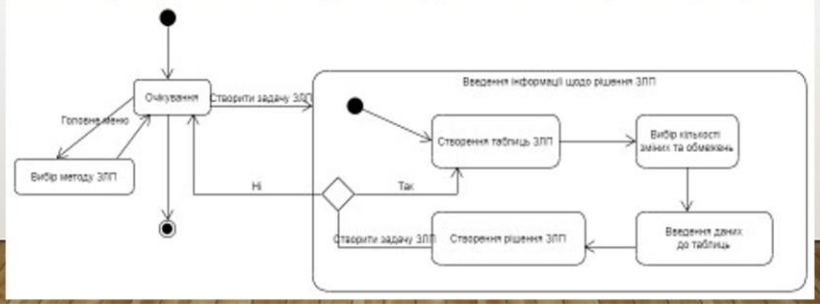
$$x_j \geq 0, j = 1, \dots, n,$$

9

МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОВОЮ UML

Уніфікована мова моделювання (UML)- це мова для візуалізації, специфікації, конструювання та документування артефактів програмних систем

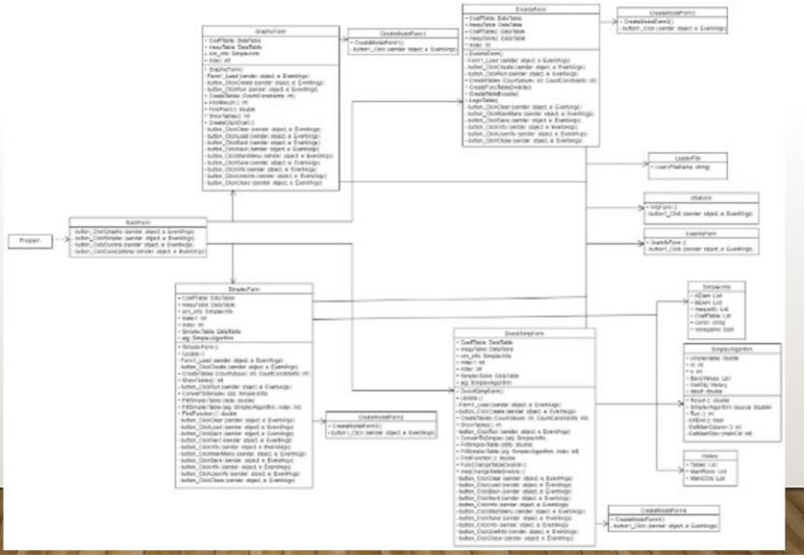
Діаграми станів - це один з п'яти видів діаграм UML, призначених для моделювання динамічних аспектів поведінки систем. Діаграма станів показує кінцевий автомат. Діаграми станів підходять для моделювання життєвого циклу об'єкта, яка демонструє потік управління від стану до стану всередині окремого об'єкта.



10

МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОВОЮ UML

Діаграми класів - найпоширеніші діаграми, що використовуються в UML Діаграма класів складається з класів, інтерфейсів, асоціацій та співпраці. Діаграми класів в основному представляють об'єктно-орієнтоване уявлення про систему, яка має статичний характер.



ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Visual Studio

Microsoft Visual Studio — інтегроване середовище розробки програмного забезпечення, за допомогою яких можно розроблятияк консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms.

WinForm Microsoft .NET

Windows Forms — інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework.

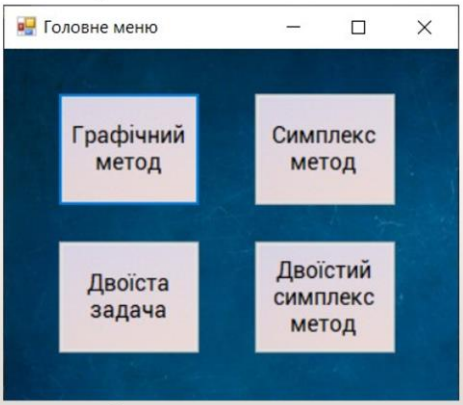
C#

C# — об'єктно-орієнтована мова програмуванняз безпечною системою типізації для платформи .NET.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ»

Мною створений додаток за допомогою якого можливо побачити не тільки результат заданих даних, але й послідовні кроки рішення, реалізація яких веде до отримання відповіді .

Програма складається з чотирьох методів, а саме графічний метод, симплекс метод, двоїста задача та двоїстий симплекс метод.



Головного меню програми

13

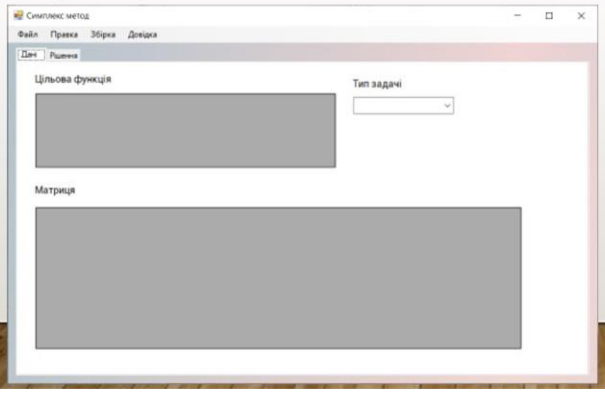
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ»

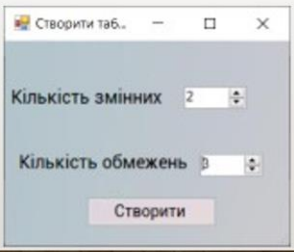
Для зразка розглянемо рішення симплекс методу.

Вибираємо симплекс метод розв'язування задачі ЗЛП, відкривається вікно.

Створюємо таблиці та вводимо дані

$$F = 4x_1 + 2x_2 \rightarrow min$$
$$5x_1 - 8x_2 \leq 16$$
$$x_1 + 3x_2 \geq -2$$
$$7x_1 + 2x_2 \leq 9$$

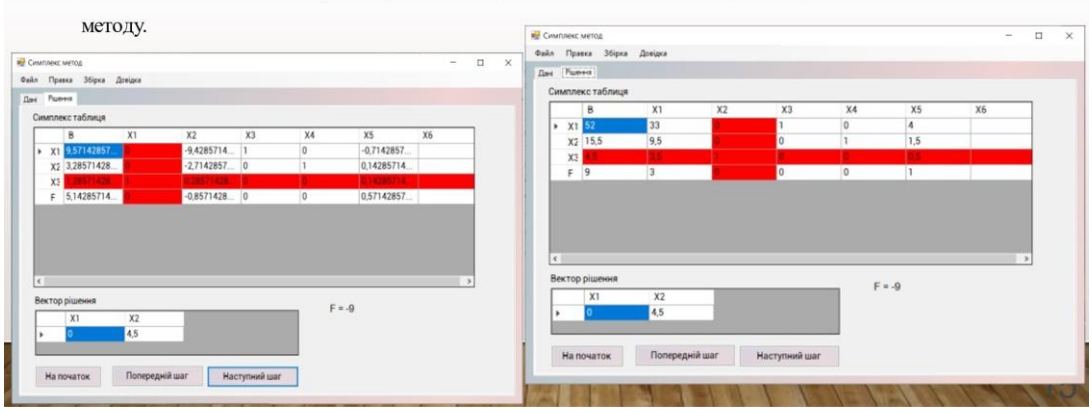




14

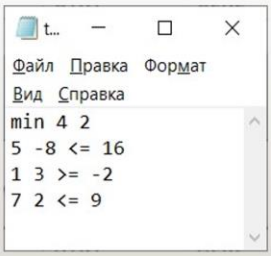
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ»

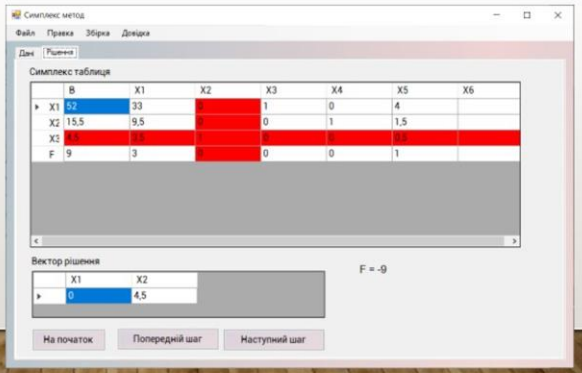
Щоб отримати рішення ЗЛП, переходимо до вкладки рішення та натискаємо «Збірка» - «Зібрати рішення». В симплекс матриці отримали покроковерозв'язаннятаблиць з створеним базисом симплекс методу.



ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ»

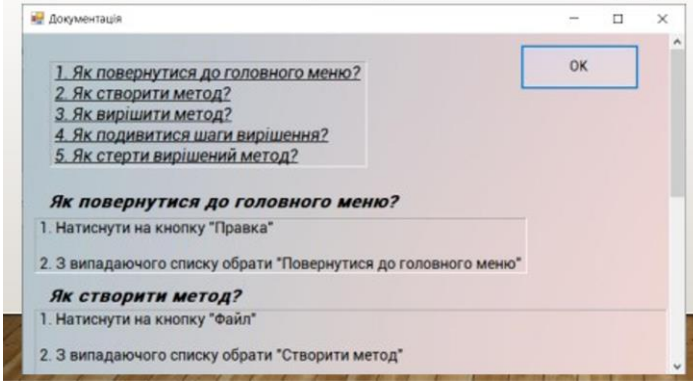
Також можливо створювати задачу не вводячи дані, а загружаючиїх з файлу формату «.txt» і «.csv». Для цього потрібно створити текстовий файл, який складається з типу задачі, значень цільової функційта обмеження з вказанням типу



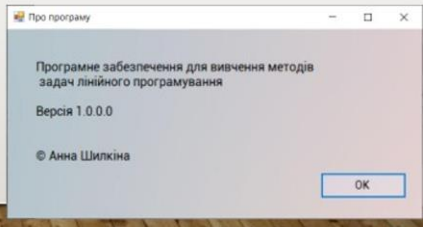


ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ ДИСЦИПЛІНИ «ДОСЛІДЖЕННЯ ОПЕРАЦІЙ»

Якщо з'являються питання як працювати в додатку натискаємо «Довідка» - «Документація», відкривається вікно з наявним список питань щодо роботи даного додатку. Далі знаходимо потрібне питання та робимо алгоритм дій, який вказаний в вікні «Документація»



Якщо цікавить інформація щодо самого додатку натискаємо «Довідка» - «Про додаток»



17

ВИСНОВКИ

1. Проаналізовано предметну область дисципліни «Дослідження операцій»,
2. Здійснено огляд методів лінійного програмування
3. Проаналізовано інтегроване середовище розробки Microsoft Visual Studio
4. Розроблено програмне забезпечення для покрокового вивчення дисципліни «Дослідження операцій»

18

ДЯКУЮ ЗА УВАГУ!