Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Шорінова Микити Олексійовича*
(ПІБ)

академічної групи *122-18-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: Розробка застосунку для вирішення та оптимізації транспортної задачі

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*Проф. Якунін А.О.*			
розділів:				
спеціальний	*Проф. Якунін А.О.*			
економічний	*доц.Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-2* (група) *Шорінов М.О.* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка застосунку для вирішення та оптимізації транспортної задачі*

затверджена наказом ректора НТУ «ДП» від *18.05.2022 р.* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *Проф. Якунін А.О.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Шорінов М.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 72 с., 9 рис., 3 дод., 20 джерел.

Об'єкт розробки: Застосунок для вирішення та оптимізації транспортної задачі на мові програмування C#.

Мета кваліфікаційної роботи: розробка додатку для транспортного підприємства з метою підвищення ефективності перевізних процесів, що характеризуються різнорідними параметрами.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточняється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціювання, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні програмного додатка, що дозволяє ефективно організувати перевізний процес з урахуванням різнорідних критеріїв.

Актуальність розробленого проекту полягає у тому, що цей додаток дозволяе вирішувати наступні актуальні питання: централізації функції планування перевезень і оперативного управління транспортним процесом, мінімізації втрат робочого часу, підвищити коефіцієнт використання вантажопідйомності транспортних засобів.

Список ключових слів: АВТОМАТИЗОВАНА ІНФОРМАЦІЙНА СИСТЕМА, ТРАНСПОРТНА СИСТЕМА, АЛГОРИТМ, ІНТЕРФЕЙС, ДОДАТОК, VISUAL STUDIO.

ABSTRACT

Explanatory note: 72 pp., 9 pics, 3 add., 20 sources.

Object of development: Application for solving and optimizing the transport problem in the C # programming language.

The purpose of the qualification work: development of an application for a transport company in order to increase the efficiency of transportation processes, which are characterized by heterogeneous parameters.

The introduction examines the current problem statement, specifies the purpose of the qualification work and the area of its application, justifies the relevance of the topic and specifies the problem statement.

In the first section carries out the analysis of the subject area, determines the relevance of the task and the dedication of the development, creates task statement, the software and hardware requirements of the product, specifies technologies and tools for development.

In the second section analyzes the existing solutions, chose the platform for development, creates design and finish development of the product, describes the algorithm, structure and architecture solutions in the system, defines the input and output data, describes characteristics of the technical resources used, describes how to run a program, features of user interaction, differences between serve and client parts of product.

The economic section defines the complexity of the information system, calculates the cost of work on creating the application and defines the time for its creation.

The practical significance lies in the creation of a software application that allows you to effectively organize the transportation process taking into account heterogeneous criteria.

The relevance of the developed project is that this application allows to address the following pressing issues: centralization of the function of transportation planning and operational management of the transport process, minimization of loss of working time, increase the capacity utilization of vehicles.

List of keywords: AUTOMATED INFORMATION SYSTEM, TRANSPORT SYSTEM, ALGORITHM, INTERFACE, APPENDIX, VISUAL STUDIO.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ПК – персоальний комп’ютер;

ПЗ – програмне забезпечення;

VS – Visual Studio;

WinForms – Windows Forms;

WPF – Windows Presentation Foundation;

ОС – операційна система;

ЦП – центральний процесор;

API – Application Programming Interface;

GPU – Graphics processing unit;

GUI – Graphical user interface;

ЗМІСТ

ВСТУП

Завдання даної кваліфікаційної роботи та об’єкт його діяльності безпосередньо пов’язані з напрямом підготовки та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври спеціальності 122 «Комп’ютерні науки».

Транспорт є однією із галузей економіки, у якій також розвивається підприємницька діяльність: транспорт пропонує на ринку товарів і послуг свою продукцію – транспортні послуги, за які отримує доходи і має прибуток. При організації транспортних перевезень приходиться враховувати множину суперечливих параметрів: економічні показники, час, вагу, габарити, важливість та ін. Практичні задачі транспортної оптимізації за своєю суттю багатокритеріальні. Природа їх така, що з поліпшенням одних критеріїв якості інші погіршуються. Звести багатокритеріальні задачі до однокритеріальних в загальному випадку не вдається. Тому останнім часом значна увага приділяється дискретним задачам в багатокритеріальних постановках.

Слід зазначити, що рішення багатокритеріальних дискретних задач в порівнянні з однокритеріальними пов'язане зі значними складностями. Це зумовлено необхідністю розробки спеціальних принципів оптимальності і відповідних схем рішення, а також набагато більшою обчислювальною складністю зазначених схем.

Ключовою характеристикою багатокритеріальних рішень є рівність для всіх. Вона відображає баланс критеріального виграшу і програшу при виборі деякої альтернативи. Задоволення цього принципу, зазвичай, призводить до втрати якості рішень. Слід зазначити, що при вирішенні складних багатокритеріальних задач, що характеризуються великою розмірністю та числом критеріїв, завдання вибору альтернатив особою.

Проблеми багатокритеріальної оптимізації транспортних перевезень виникають через неповноту даних про предметну область. При цьому ми

вважаємо, що недостатня для прийняття рішення інформація завжди може бути отримана від експерта. В якості експерта або особи, що приймає рішення в нашому випадку виступає керівництво або диспетчер транспортного підприємства. Для підвищення ефективності процесу прийняття рішень пропонується створити програмний модуль інформаційної системи транспортного підприємства. Ця інформаційна система, повинна реалізувати процес прийняття рішень на основі обчислювальних процедур.

Мета кваліфікаційної роботи: розробка додатку для транспортного підприємства з метою підвищення ефективності перевізних процесів, що характеризуються різнорідними параметрами.

Основними вимогами до додатку є:

1) функціональна працездатність елементів ресурсу;

2) взаємодія сервера з клієнтською частиною;

3) забезпечення коректної обробки даних;

4) забезпечення належного рівня безпеки даних;

5) зручність роботи з додатком

РОЗДІЛ І
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Автоматизована інформаційна система (АІС) представляє собою сукупність інформації, економіко-математичних методів і моделей, технічних, програмних, технологічних засобів і фахівців, призначена для обробки інформації та прийняття управлінських рішень.

Створення АІС сприяє підвищенню ефективності підприємства і забезпечує якість управління. Найбільша ефективність АІС досягається при оптимізації планів роботи підприємств, швидкої виробленні оперативних рішень, чіткому маневруванні матеріальними і фінансовими ресурсами. Система управління перевезеннями являє собою сукупність завдань планування, організації, контролю, регулювання, обліку перевізного процесу, для розв’язання яких використовуються економіко-математичні методи і сучасні електронно-обчислювальні засоби.

Основна мета розробки АІС автотранспортного підприємства (АТП) - підвищення ефективності роботи рухомого складу (ПС) шляхом централізації функції планування перевезень і оперативного управління транспортним процесом. Підвищення ефективності використання ПС і зниження витрат на перевезення передбачаються за рахунок мінімізації втрат робочого часу з організаційних причин, зменшення питомої ваги порожніх пробігів і величини понаднормативних простоїв автомобілів під вантажно-розвантажувальними операціями, підвищення коефіцієнта використання вантажопідйомності транспортних засобів [1-6].

Напрями функціонування інформаційної системи:

Тепер розглянемо характеристики транспортних інформаційних систем [1, 3, 6].

Організаційний напрям:

– диспетчеризація транспортних перевезень по ряду основних операцій (комерційний огляд вантажів, перевірка транспортних засобів, здійснення переадресування вантажів, проведення перевалки вантажів);

– розробка нормативної бази (розрахунок норм завантаження транспортних засобів, диференціація нормативів для насипних, навалювальних, наливних, лісових вантажів);

– проведення підготовчих процедур (підготовка необхідної перевізної документації, маркування вантажів, здійснення вантажно-розвантажувальних операцій, визначення термінів доставки вантажів, формування вантажів в відповідний транспортний пакет, складання вантажного плану для різних видів транспорту).

Економічний напрям:

– аналіз і оцінка змінних витрат, пов'язаних з перевезенням вантажів (витрати на паливо, мастильні матеріали, електроенергію, технічне обслуговування і поточний ремонт);

– аналіз і оцінка постійних витрат, пов'язаних з перевезенням вантажів;

– аналіз і оцінка логістичних складових витрат;

– оцінка співвідношення собівартості перевезень вантажів і оборотних коштів, вкладених в вантажі (вартість вантажної маси), що знаходяться в процесі транспортування;

– визначення величини вивільнених оборотних коштів при прискоренні доставки вантажів;

– оптимізація витрат на транспортні операції (початково-кінцеві операції, переміщення вантажів, додаткові операції);

– мінімізація коштів технічної оснащеності фронту навантаження і вивантаження за вартісним критерієм.

– Перевізний напрямок:

– вибір оптимальних варіантів транспортування вантажів (вибір виду транспортування або системи доставки вантажів, вибір виду або кількох видів транспорту, вибір або оптимальне поєднання способу транспортування, оптимальне поєднання учасників перевізного процесу);

– складання раціональних маршрутів руху транспортних засобів (розрахунок графіків руху транспортних засобів за різними маршрутами, використання особистих видів транспорту, розробка гнучких схем маршрутизації перевезень);

– раціональне використання транспортних засобів за вантажопідйомністю (здійснення комбінованих перевезень, можливість використання додаткового обладнання в процесі перевезення, розробка оптимальних схем укладання вантажів в транспортний засіб);

– транспортування вантажів (доставка вантажів вантажоодержувачам у встановлені терміни, здійснення розвезення мелкопартіонних вантажу в місце призначення, розв’язання спорів, що виникають в процесі транспортування).

– Сервісний напрям:

– здійснення повернення багатооборотної тари (піддони, контейнери) з організацією її доставки;

– вибір посередників на основі критеріїв замовника (витрати, надійність і час доставки, збереження вантажів);

– зберігання, складування, сортування, комплектація вантажів; інформаційні послуги, страхування і охорона вантажів.

Основні задачі транспортних підприємств:

Відмінною рисою роботи транспортних підприємств в нових умовах конкуренції на ринку транспортних послуг стає розробка політик комплексного розв’язання транспортних завдань [4-6].

Транспортні послуги:

– виконання і оформлення розрахунків за перевезення вантажів;

– визначення ціни за перевезення;

– виконання складських операцій;

– вибір оптимального маршруту доставки товарів;

– контроль вантажів, що знаходяться на шляху прямування;

– організацію електронного обміну даними між усіма учасниками логістичного процесу і зберігання інформації;

– контроль за товарно-матеріальними запасами, виконанням замовлень;

– експлуатацію парку транспортних засобів.

– В останні роки на транспорті ряду промислово розвинених країн дослідженням потреб стали займатися спеціальні логістичні центри. Вони проводять аналіз вантажопотоків і розподілу їх по мережі. На основі даних аналізу готуються пропозиції:

– по організації оптимальних вантажопотоків;

– за розподілом перевезень між різними видами транспорту;

– по комплектуванню груп товарів;

– по порядку укладення договорів на перевезення і ін.

– Задачі в області комунікацій наступні:

– інформувати клієнтів про пропоновані пакети послуг і постійно надавати необхідний вплив на клієнтуру, щоб вона могла використовувати послуги в можливо більшому обсязі;

– сприяти розширенню і вдосконаленню взаємодії транспортних фірм і вантажовідправників на основі використання обчислювальної техніки, головним чином за допомогою електронного обміну даними.

Вибір транспорту для конкретного перевезення. Характеристики вантажу: тип вантажу, місце відправки і призначення, відстань, вартість вантажу, терміновість, надійність доставки та ін.

Характеристики транспорту: особливості різних видів транспорту, їх переваги та недоліки, безпека, графіки і частота доставки. Виділяють наступні характеристики:

– час доставки;

– частота відправлень вантажу;

– надійність дотримання графіка доставки;

– здатність перевозити різні вантажі;

– здатність доставити вантаж у будь-яку точку території;

– вартість перевезення.

Характеристики перевізника: репутація перевізника, безпеку, показники збитків і пошкоджень.

Експертна оцінка значимості різних чинників показує, що при виборі транспорту, в першу чергу беруть до уваги: надійність дотримання графіка доставки; час доставки; вартість перевезення. Вибір способу перевезення залежить від типу вантажу, який треба перевезти, місць відправлення та призначення, відстані, вартості вантажу, терміновості перевезення та множина інших чинників. Тому більшість практичних завдань управління на транспорті є багатокритеріальної. У ці завдання необхідно враховувати безліч факторів або параметрів, які суперечать один одному.

Найбільш поширений в світі автомобільний транспорт. Сфера його застосування - міські, приміські та міжміські вантажні і пасажирські перевезення, а також перевезення на середні та дальні відстані малотоннажних цінних і швидкопсувних вантажів.

Розглянемо основні фактори, що визначають ефективність перевезення.

Чинники що визначають вартість автомобільного перевезення:

– відстань перевезення;

– маса вантажу;

– об'ємна вага вантажу, що характеризує можливість використання вантажопідйомності автомобіля;

– вантажопідйомність автомобіля;
– загальний пробіг;
– час використання автомобіля;
– тип автомобіля;
– район, в якому здійснюється перевезення та ін.

Інформаційні системи і технології у транспортних системах:

Необхідною умовою злагодженої роботи всіх ланок транспортного ланцюга є наявність автоматизованих інформаційних систем. АІС - гнучка структура, що складається з персоналу, виробничих об'єктів, засобів обчислювальної техніки, необхідних довідників, комп'ютерних програм, різних інтерфейсів і процедур (технологій), об'єднаних пов'язаної інформацією, використовуваної в управлінні організацією для планування, контролю, аналізу і регулювання логістичної системи. Часто використовується тотожний термін «логістична інформаційна система» (АІС), які, як правило, представляють собою автоматизовані системи управління логістичними процесами.

Архітектура інформаційної системи характеризує її загальну логічну структуру, апаратне забезпечення, програмне забезпечення, описує методи кодування інформації, тобто процесу представлення даних послідовністю символів, визначає інтерфейс користувача з системою [2].

Апаратне забезпечення (Hardwaге) - це комплекс електронних, електричних і механічних пристроїв, що входять до складу інформаційної системи або мережі.

Програмне забезпечення(ПО) (software) - це комплекс комп'ютерних програм, що забезпечує обробку або передачу даних, а також розробку нових програм.

інтерфейс користувача- це система взаємодії людини з інформаційною системою. Адаптація функціонування комплексів прикладних процесів до образу мислення людини вимагає створення дружніх інтерфейсів.

З технічної точки зору ІС, як кожна відкрита система, призначена для виконання двох основних функцій: обробки даних і передачі даних. З практичної точки зору набір функцій і завдань АІС дуже різноманітний.

Функції інформаційних систем:

– прогнозування попиту і планування потреб в матеріалах;

– координація подій, операцій і процесів по всьому ланцюгу просування матеріальних цінностей і послуг;

– моніторинг і контроль протікання операцій. Поточний моніторинг покликаний створювати основи для регулювання процесів з метою підвищення їх безперебійності;

– оперативне управління логістичними процесами, Особливо поставками, транспортуванням, зберіганням, фізичної дистрибуцією і т.д.

– Джерела ефекту від впровадження інформаційних систем і технологій:

– скорочення часу проходження процесу, оскільки можна заздалегідь оптимізувати хід майбутніх транспортних, складських, вантажно-розвантажувальних, виробничих процесів і скоротити час їх проходження;

– раціональне використання ресурсів. Своєчасна інформація про хід реалізації логістичних процесів і про стан логістичних інфраструктур дозволяє здійснити більш розумне використання транспортних шляхів і засобів, вантажно-розвантажувальне устаткування, персонал та ін.;

– підвищення якості логістичного процесу. Інформаційна прозорість ходу реалізації логістичного процесу дозволяє в режимі реального часу приймати рішення, краще реагувати на збої, неефективно організовані ділянки роботи, зайві витрати ресурсів і т.д;

– скорочення помилок;

– скорочення обсягу паперової документації.

1.2. Призначення розробки та галузь застосування

Виконана кваліфікаційна робота має назву «Розробка застосунку для вирішення та оптимізації транспортної задачі».

Призначення розробки: процес управління транспортними перевезеннями виконується в складному інформаційному середовищі, коли результати прийнятих рішень оцінюються не по одному, а по сукупності багатьох показників (час, вартість, відстань, кваліфікація, терміновість і т.п.), і ці параметри потрібно враховувати одночасно. Тому задачі складання оптимальних планів перевезень з урахуванням не лише вартості, а й часу і пріоритетності є актуальними. Цю задачу покликаний вирішувати розроблений додаток з управління перевезеннями, створення якого є метою даної роботи.

1.3. Підстави для розробки

Підставами для розробки є:

- освітня програма спеціальності 122 “Комп’ютерні науки”;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету “Дніпровська політехніка” № 268-с від 18.05.2022р;
- завдання на кваліфікаційну роботу на тему “Розробка застосунку для вирішення та оптимізації транспортної задачі”.

1.4. Постановка завдання

Транспортно-логістична компанія здійснює перевезення однорідних вантажів від вантажовідправника (m пунктів постачання) до вантажоотримувача (n пунктів споживання).

Відома наявна кількість вантажу у кожного з постачальників та потреби у вантажі кожного споживача. Відома вартість перевезення одиниці вантажу від кожного постачальника до кожного споживача, час виконання замовлень та пріоритетність постачальника, яку визначає клієнт.

Метою кваліфікаційної роботи є розробка додатку який буде складати план перевезень, щоб задовільнити найбільшу кількість замовлень з мінімальною вартістю перевезень, мінімальним часом виконання замовлень та з урахуванням пріорітеності замовлення.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Система також повинна мати високу швидкодію, щоб не змушувати користувача чекати і щоб дійсно скоротити час виконання рутинних дій.

Вимоги до інтерфейсу. Розробляючи інтерфейс програмного продукту, необхідно враховувати, що він створюється для непрофесійного користувача. Це накладає на інтерфейс ряд вимог:

– інтерфейс програми повинен бути інтуїтивно зрозумілим. Мається на увазі, що інтерфейс не вимагає багато часу на його вивчення, за умови, що користувач розбирається в даній предметній області і має навички роботи на комп'ютері;

– в разі якщо користувач допустив якусь помилку при роботі з даною програмою, повинні бути передбачені повідомлення для користувача, що містять опис помилки і рекомендації щодо їх виправлення;

Система повинна відповідати таким вимогам:

– надійності;

– безпечності;

– вимогам до захисту інформації від несанкціонованого доступу. Має бути передбачений захист від несанкціонованого доступу до даних, введення даних, їх видалення;

– дані повинні зберігатися відповідно до наявних нормативних вимог;

– інформація, яка зберігається в системі, повинна бути захищена від аварійних ситуацій.

1.5.2. Вимоги до інформаційної безпеки

До основних вимог інформаційної безпеки є: тримати данні у зашифрованому вигляді, використовувати надійне та ліцензійне противирусне програмне забезпечення, регулярно робити бекап на випадок поломки сховища даних

1.5.3. Вимоги до складу та параметрів технічних засобів

Додаток потребує більшість вбудованих можливостей .NET Framework 4.0 і тому вимагає установки його runtime версії для свого запуску. Також необхідно мати наступні мінімальні характеристики системи, на якій буде відбуватися запуск:

- наявність ОС Windows версій 7, 8 , 8.1 або 10;
- ЦП розрядністю x86 або x64;
- частоту ЦП мінімум у 4 GHz;
- відеоадаптер (GPU);
- оперативна пам’ять мінімум 1 гігабайт;

1.5.4. Вимоги до інформаційної та програмної сумісності

Додаток був розроблений на мові програмування C#. Середовищем розробки було обрано Microsoft Visual Studio – інтегроване середовище розробки, яке дозволяє розробляти як консольні програми, так і програми з графічним інтерфейсом. Додаток потребує наявності .NET Framework не молодше версії 4.0

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Результатом даної кваліфікаційної роботи має стати додаток для транспортного підприємства з метою підвищення ефективності перевізних процесів, що характеризуються різнорідними параметрами.

Основне призначення додатку - складати план перевезень, щоб задовільнити найбільшу кількість замовлень з мінімальною вартістю перевезень, мінімальним часом виконання замовлень та з урахуванням пріорітеності замовлення.

Розроблений додаток дозволяє диспетчерам підприємства визначати оптимальні плани транзитних перевезень, вартість і час доставки вантажів.

З точки зору інтерфейсного рішення слід зазначити, що система реалізована у вигляді багатовіконного додатку, виконаного в стилі операційної системи Windows. Форма додатку містить меню і вікна з закладками, на яких розташовується основний графічний інтерфейс (GUI).

2.2. Опис застосованих математичних методів

В задачах математичного програмування з одним критерієм потрібно визначити значення цільової функції, відповідне, наприклад, мінімальним витратам або максимальному прибутку. Однак, практично в будь-якій реальній ситуації виявляється кілька цілей, які суперечать одна одній.

У зв'язку з цим виникає необхідність використання дещо іншого підходу до розв’язання задач даного класу –за допомогою методів багатокритеріальної оптимізації.

Постановка задачі багатокритеріальної оптимізації [7, 8] має вигляд:

$$\max \{f1\ (x) = F1\},$$

max {$f2$ (x) = $F2$}, (2.1)

...

max {fk (x) = Fk}, При $x \in X$,

де X - множина допустимих значень змінних x;

k - число цільових функцій (критеріїв);

F_i- значення i-го критерію (цільової функції), "Max" - означає, що даний критерій потрібно максимізувати.

Зауважимо, що, по суті, багатокритеріальна задача відрізняється від звичайної задачі оптимізації тільки наявністю декількох цільових функцій замість однієї. Критерії можуть мати різну важливість для особи приймаючої рішення (ОПР).

Слід сказати, що рішення такого завдання не дасть найкращих значень для кожного критерію, так як часто поліпшення одного критерію викликає погіршення іншого. Таким чином, при вирішенні багатокритеріальної задачі отримуємо деякий компромісне рішення [7-8].

Існуючі на сьогоднішній день методи багатокритеріальної оптимізації умовно можна розділити на 2 групи [9-10]. Методи першої групи зводять багатокритеріальну задачу до однокритеріальної шляхом згортання векторного критерію в суперкритерій, який оптимізується одним з методів однокритеріальної оптимізації. Існують різні види згорток. Найбільш поширеним способом згортання векторного критерію є лінійна згортка виду:

$$\Phi(x) = \sum_{i=1}^{m} \alpha_i F_i(x), \ \alpha_i \geq 0. \qquad (2.2)$$

Існують різні способи вибору коефіцієнтів α_i. Одним з них є призначення α_i в залежності від відносної важливості критеріїв [11]. Такий підбір зазначених коефіцієнтів можна виконувати згідно з таблицею 2.1

Таблиця 2.1 Важливість критеріїв

Важливість	Визначення
1	Рівна важливість порівнюваних вимог
3	Помірне (слабке) перевага одного над іншим
5	Сильне (істотне) перевага
7	Очевидна перевага
9	Абсолютна (переважна) перевага
2, 4, 6, 8	Проміжні рішення між двома сусідніми оцінками

До другої групи можна віднести інші методи багатокритеріальної оптимізації, які не виконують згортання локальних критеріїв в скалярний суперкритерій.

Транспортно-логістична компанія здійснює перевезення однорідних вантажів від вантажовідправника (*m* пунктів постачання) до вантажоотримувача (*n* пунктів споживання).

Відома наявна кількість вантажу у кожного з постачальників та потреби у вантажі кожного споживача. Відома вартість перевезення одиниці вантажу від кожного постачальника до кожного споживача, час виконання замовлень та пріоритетність постачальника, яку визначає клієнт.

Необхідно скласти такий план перевезень, щоб задовільнити найбільшу кількість замовлень з мінімальною вартістю перевезень, мінімальним часом виконання замовлень та з урахуванням пріорітеності замовлення.

Математична модель задачі оптимізації перевезень об'єднує моделі машин, робіт, відношень, а також цілі та додаткові обмеження на перевезення в рамках єдиного процесу ефективної доставки вантажів. Ключовий елемент моделі задачі – це цільова функція. У класичному розумінні мета транспортного підприємства

– доставити найбільше число вантажів за найменшу сумарну ціну. Тому задачі пошуку оптимального плану транспортних перевезень представляють математичною моделлю ТЗ [12].

Нехай є M постачальників (машин), у кожного з яких є задана кількість штучних вантажів (товару) і N споживачів (робіт).

a_i – потужність (продуктивність) машини i, що означає число одиниць вантажу, яке є у постачальника i.

b_j – потужність роботи j, що означає число одиниць вантажу, яке необхідно споживачеві або сумарна потужність машин необхідних для виконання роботи j .

c_{ij}– вартість доставки (роботи).

x_{ij}– число одиниць вантажу, що передаються від постачальника i споживачеві j, $x_{ij} \geq 0$.

Тоді, математична модель задачі має вигляд

$$\begin{cases} w\left(E^*_{a_i,b_j}\right) = \sum_i^M \sum_j^N \left(x_{ij}c_{ij}\right) \to min \\ \sum_{i=1}^M x_{ij} = b_j, j = \overline{1,N} \\ \sum_{j=1}^N x_{ij} = a_i, i = \overline{1,M} \end{cases} \qquad (2.3)$$

Слід зазначити, що класична ТЗ являє задачу лінійного програмування з лінійною цільовою функцією (2.3) і лінійної системою обмежень.

У наведеній вище постановці ТЗ цільова функція мінімізується, однак якщо в транспортній моделі тариф означає, наприклад, прибуток, обсяг продажів, оцінку якості то цільову функцію потрібно максимізувати. Для цього потрібно всі транспортні тарифи помножити на мінус 1. У даній моделі даних мінімальний від'ємний тариф означає максимальний прибуток.

В прикладних транспортних задачах можуть бути наступні варіанти поєднання сумарних потужностей машин і робіт.

Рівність $\sum_{i=1}^{M} a_i = \sum_{j=1}^{N} b_j$. У цьому випадку модель задачі називається збалансованою або закритою, а в іншому – незбалансованою або відкритою.

Надлишок машин або нестача робіт $\sum_{i=1}^{M} a_i > \sum_{j=1}^{N} b_j$.

Нестача машин або надлишок робіт $\sum_{i=1}^{M} a_i < \sum_{j=1}^{N} b_j$.

Задачі другого типу часто називають ТЗ в умовах перевиробництва або ТЗ з надлишком запасів, тому що потужність машини можна інтерпретувати, як надлишкову ємність або запас деякого абстрактного постачальника. А задачі третього типу називають ТЗ в умовах дефіциту або ТЗ з надлишком заявок, тобто потужність робіт інтерпретують, як обсяг заявок споживача, що перевершує можливості постачальника.

Більшість методів розв'язання ТЗ вимагає, щоб модель задачі була закритою. Така модель будується досить просто. У разі надлишку потужності машин необхідно додати фіктивну роботу потужності $b_{N+1} = \sum_{i=1}^{M} a_i - \sum_{j=1}^{N} b_j$, а при нестачі машин додати фіктивну машину потужності $a_{M+1} = \sum_{j=1}^{N} b_j - \sum_{i=1}^{M} a_i$. Ціни виконання фіктивних робіт зручно призначити рівними нулю (мінімально допустимими), але, за необхідності, їх можна призначити рівними деякій константі $C \geq 0, C \neq \infty$. Відзначимо, що перехід до закритої моделі здійснюється без використання забороняючих тарифів.

У багатокритеріальній постановці елемент транспортної таблиці β_{ij} представляє складний тип даних, званий вектором критеріїв.

$$\beta_{ij} = \{c_{ij}^k, k = \overline{1,2}\}, \qquad (2.4)$$

де c_{ij}^k – значення k-го критерію елемента рішення β_{ij}.

Функціонал трикритеріальної транспортної задачі має вигляд

$$\begin{cases} w\left(E^1_{a_i,b_j}\right) = \sum_i^M \sum_j^N \left(x_{ij} c_{ij}\right) \to min \\ w\left(E^2_{a_i,b_j}\right) = \sum_i^M \sum_j^N \left(p_{ij}\right) \to min \\ w\left(E^3_{a_i,b_j}\right) = \sum_i^M \sum_j^N \left(t_{ij}\right) \to min \\ \sum_{i=1}^M x_{ij} = b_j, j = \overline{1,N} \\ \sum_{j=1}^N x_{ij} = a_i, i = \overline{1,M} \end{cases}, \quad (2.5)$$

де x_{ij} – обсяг вантажу, що перевозиться від постачальника *i* до споживача *j*;

c_{ij} – вартість доставки одиниці вантажу, відповідно від постачальника *i* до споживача *j*;

p_{ij} – пріоритет постачальника, який оцінюється у балах від 1 до 3;

t_{ij} – середній час виконання замовлення;

a_i – запаси вантажу у постачальників;

b_j – потреби замовників.

На відміну від однокритеріальних задач, наприклад, класичної ТЗ, рішення багатокритеріальних задач суперечливі, оскільки, неможливо з абсолютною впевненістю стверджувати, що те чи інше рішення строго оптимальне. Тому, в основі алгоритмів розв'язання багатокритеріальних задач завжди лежить певна схема пошуку компромісу.

2.3. Опис використаних технологій та мов програмування

Додаток у кваліфікаційній роботі був розроблений за допомогою наступних технологій та засобів:

- інтегроване середовище розробки MS Visual Studio;
- технологія Windows Forms;
- мова програмування C#.

Термін IDE розшифровується як Integrated Development Environment або «інтегроване середовище розробки». Це набір інструментів, які дозволяють

створювати програми. Іншими словами, якщо говорити дуже просто, то IDE – це програма, де створюються інші програми.

Спочатку, на зорі розвитку комп'ютерної техніки, програмісти записували код на папері, після чого його вводили в ЕОМ за допомогою перфострічок або перфокарт (до речі, те й інше також виготовлялося з паперу). Одна найменша помилка могла призвести до непрацездатності програми. Згодом, коли обчислювальна техніка досягла певного рівня розвитку, з'явилася можливість редагування коду прямо на екрані терміналу. А трохи пізніше з'явилися і засоби, що дозволяють набирати текст програми на екрані, уникаючи «паперової тяганини».

Насправді, це набагато складніший інструмент. Саме собою середовище розробки зазвичай включає і спеціалізований текстовий редактор, «заточений» до роботи з кодом. Але для повноцінного програмування цього, звичайно, недостатньо.

Потрібна також наявність хоча б компілятора та відладчика. Перший необхідний для того, щоб перевести текст програми, створений з використанням команд, написаних англійською (зазвичай) мовою, у машинні коди, зрозумілі комп'ютеру. Відладчик же використовується для знаходження та усунення помилок, що неминуче виникають при написанні коду.

По факту ж сучасні IDE включають безліч найрізноманітніших інструментів, покликаних вирішувати ті чи інші завдання. Наприклад, там можуть бути інструменти для візуальної розробки, що дозволяють буквально «намалювати» програму, використовуючи для цього спеціальний графічний редактор.

Перші IDE мали простий (навіть скажімо більше, примітивний) текстовий інтерфейс. Потім з'явилися рішення з графічним інтерфейсом. Деякі сучасні середовища розробки відрізняються високою складністю і часом просто щоб розібратися в них, навіть досвідченому розробнику необхідно спочатку прочитати відповідну документацію.

В даній кваліфікаційної роботи для розробки було обране IDE - Visual Studio. Visual Studio - це інтегроване середовище розробки (IDE), розроблене Microsoft для розробки графічного інтерфейсу користувача (GUI), консолі, веб-додатків, веб-додатків, мобільних додатків, хмарних і веб-сервісів і т. д. За допомогою цієї IDE ви Ви можете створювати керований код, а також власний код. Він використовує різні платформи програмного забезпечення Microsoft для розробки програмного забезпечення, такі як магазин Windows, Microsoft Silverlight, Windows API і т. д. Це не мовне середовище IDE, оскільки ви можете використовувати його для написання коду C#, C++, VB (Visual Basic), Python, JavaScript та багато інших мов. Він забезпечує підтримку 36 різних мов програмування. Він доступний як для Windows, так і MacOS.

Інтегроване середовище розробки (IntegratedDevelopmentEnvironment - IDE) Visual Studio пропонує ряд високорівневих функціональних можливостей, що виходять за рамки базового керування кодом.

Нижче наведено основні переваги IDE-середовища Visual Studio.

Вбудований Web-сервер. Для обслуговування Web-додатка ASP.NET необхідний Web-сервер, який чекатиме на Web-запити і обробляти відповідні сторінки. Наявність у Visual Studio інтегрованого Web-сервера дозволяє запускати Web-сайт прямо з середовища проектування, а також підвищує безпеку, за винятком ймовірності отримання доступу до тестового Web-сайту з якогось зовнішнього комп'ютера, оскільки тестовий сервер може приймати з'єднання тільки з локального комп'ютера.

Підтримка багатьох мов під час розробки. Visual Studio дозволяє писати код своєю мовою або будь-яких інших мовах, використовуючи весь час один і той же інтерфейс (IDE). Більш того, Visual Studio також ще дозволяє створювати Web-сторінки різними мовами, але поміщати їх все в один і той же Web-додаток. Єдиним обмеженням є те, що в кожній Web-сторінці можна використовувати лише якусь одну мову (очевидно, що інакше проблем при компіляції було б просто не уникнути).

Менше за код для написання. Для створення більшості програм потрібна пристойна кількість стандартного стереотипного коду та Web-сторінки ASP. NET тому не виняток. Наприклад, додавання Web-елемента керування, приєднання обробників подій та коригування форматування потребує встановлення у розмітці сторінки ряду деталей. У Visual Studio такі деталі встановлюються автоматично.

Інтуїтивний стиль кодування. За промовчанням Visual Studio форматує код у міру його введення, автоматично вставляючи необхідні відступи та застосовуючи колірне кодування для виділення елементів типу коментарів. Такі незначні відмінності роблять код зручнішим для читання і менш схильним до помилок. Застосовувані Visual Studio автоматично параметри форматування можна навіть налаштовувати, що дуже зручно у випадках, коли розробник віддає перевагу іншому стилю розміщення дужок (наприклад, стиль K&R, при якому дужка розташовується на тому ж рядку, що і оголошення, якому вона передує).

Вища швидкість розробки. Багато функціональних можливостей Visual Studio спрямовані на те, щоб допомагати розробнику робити свою роботу якнайшвидше. Зручні функції, на зразок функції IntelliSense (яка вміє перехоплювати помилки та пропонувати правильні варіанти), функції пошуку та заміни (яка дозволяє відшукувати ключові слова як в одному файлі, так і у всьому проекті) та функції автоматичного додавання та видалення коментарів (яка може тимчасово приховувати блоки коду), дозволяють розробнику працювати швидко та ефективно.

Можливості налагодження. Пропоновані Visual Studio інструменти налагодження є найкращим засобом для відстеження загадкових помилок і діагностування дивної поведінки. Розробник може виконувати свій код за рядком за раз, встановлювати інтелектуальні точки переривання, за бажанням зберігаючи їх для використання в майбутньому, і будь-коли переглядати поточну інформацію з пам'яті.

Visual Studio також має безліч інших функцій: можливість управління проектом; вбудована функція керування вихідним кодом; можливість

рефакторизації коду; потужна модель розширюваності. Більше того, у разі використання Visual Studio 2008 Team System розробник отримує розширені можливості для модульного тестування, спільної роботи та управління версіями коду (що значно більше того, що пропонується у більш простих інструментах типу Visual SourceSafe).

Як недолік можна відзначити неможливість відладчика (Microsoft Visual Studio Debugger) відстежувати у коді режиму ядра. Налагодження в Windows у режимі ядра в загальному випадку виконується під час використання WinDbg, KD або SoftICE.

C#, C-sharp, сі-шарп - мова програмування, що поєднує об'єктно-орієнтовані та аспектно-орієнтовані концепції. Розроблений в 1998-2001 роках групою інженерів під керівництвом Андерс Хейлсберг в компанії Microsoft як основна мова розробки додатків для платформи Microsoft .NET. Компілятор з C# входить до стандартної установки самої .NET, тому програми на ньому можна створювати і компілювати навіть без інструментальних засобів на кшталт Visual Studio.

C# відноситься до сім'ї мов з C-подібним синтаксисом, їх синтаксис найбільш близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, навантаження операторів, покажчики на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників - мов C++, Java, Delphi, Модула і Smalltalk - C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так C# не підтримує множинне успадкування класів (на відміну від C++).

.NET – це велика платформа для розробників. Вона складається з різних інструментів, мов програмування та бібліотек. І допомагає програмістам розробляти веб-сайти, програми, ігри та послуги. Існують різні версії. Кожна

дозволяє коду .NET виконуватись у різних системах - Linux, macOS, Windows, iOS, Android та ін.

.NET Framework – це оригінальна реалізація .NET. Вона підтримує роботу з веб-сайтами, службами та настільними програмами у Windows.

Основні риси .NET:

- Підтримка багатьох мов.
- Кросплатформність.
- Потужна бібліотека класів.
- Різноманітність технологій.
- Продуктивність.

Також слід відзначити таку особливість мови C# і фреймворку .NET, як автоматичне складання сміття. А це означає, що нам здебільшого не доведеться, на відміну від C++, дбати про звільнення пам'яті. Вищезазначене загальномовне середовище CLR сама викличе збирач сміття та очистить пам'ять.

2.4. Опис структури системи та алгоритмів її функціювання

Діаграми варіантів використання мають велике значення для візуалізації, специфікування і документування поведінки елемента. Вони полегшують розуміння систем, підсистем або класів, представляючи погляд ззовні на те, як дані елементи можуть бути використані у відповідному контексті. Крім того, такі діаграми важливі для тестування виконуваних систем в процесі прямого проектування і для розуміння їх внутрішнього устрою при зворотному проектуванні.

Варіант використання являє собою послідовність дій, виконуваних системою у відповідь на подію, що ініціюється деяким зовнішнім об'єктом (дійовою особою). Варіант використання описує типову взаємодію між користувачем і системою. Варіант використання визначається в процесі обговорення з користувачем тих функцій, які він хотів би реалізувати. Дійова особа (actor) – це роль, яку користувач грає по відношенню до системи.

На діаграмах варіантів використання підтримується кілька типів зв'язків між елементами діаграми.

Зв'язок комунікації – це зв'язок між варіантом використання і дійовою особою. Зв'язки комунікації показують за допомогою односпрямованої асоціації (суцільної лінії).

Зв'язок включення застосовується в тих ситуаціях, коли є який-небудь фрагмент поведінки системи, який повторюється більше ніж в одному варіанті використання. За допомогою таких зв'язків зазвичай моделюють багаторазово використовувану функціональність.

Рисунок 1.1 – Діаграма варіантів використання

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів складається з множини елементів, які в сукупності відображають декларативні знання про предметну область. Ці знання інтерпретуються в базових поняттях мови UML, таких як класи, інтерфейси і відносини між ними і елементами з яких вони складаються. При цьому окремі

елементи цієї діаграми можуть організовуватися в пакети для представлення більш загальної моделі системи.

Клас (class) – абстрактне опис безлічі однорідних об'єктів, що мають однакові атрибути, операції і відносини з об'єктами інших класів.

Атрибут (attribute) – змістовна характеристика класу, що описує безліч значень, які можуть приймати окремі об'єкти цього класу.

Атрибут класу служить для представлення окремої властивості або ознаки, який є загальним для всіх об'єктів даного класу.

Видимість в мові UML специфікується за допомогою квантора видимості (visibility), який може приймати одне з 4-х можливих значень і відображатися за допомогою спеціальних символів.

Загальнодоступний (public) атрибут доступний або видимий з будь-якого іншого класу пакета, в якому визначена діаграма.

Захищений (protected) атрибут недоступний або не видимий для всіх класів, за винятком підкласів даного класу.

Закритий (private) атрибут недоступний або не видтмий для всіх класів без виключення.

Операція (operation) – це сервіс, що надається кожним екземпляром або об'єктом класу на вимогу своїх клієнтів, в якості яких можуть виступати інші об'єкти, в тому числі і екземпляри даного класу.

Крім внутрішнього устрою або структури класів важливу роль при розробці проектованої системи мають різні відносини між класами, які також можуть бути зображені на діаграмі класів.

- Базовими відносинами на діаграмах класів є:
- відношення асоціації (association relationship);
- відношення узагальнення (generalization relationship);
- відношення агрегації (aggregation relationship);
- відношення композиції (composition relationship);
- відношення залежності (dependency relationship).

Розроблювана програма включає до свого складу три основні частини: клас нормалізаційного методу, інтерфейсний клас і класи сутностей.

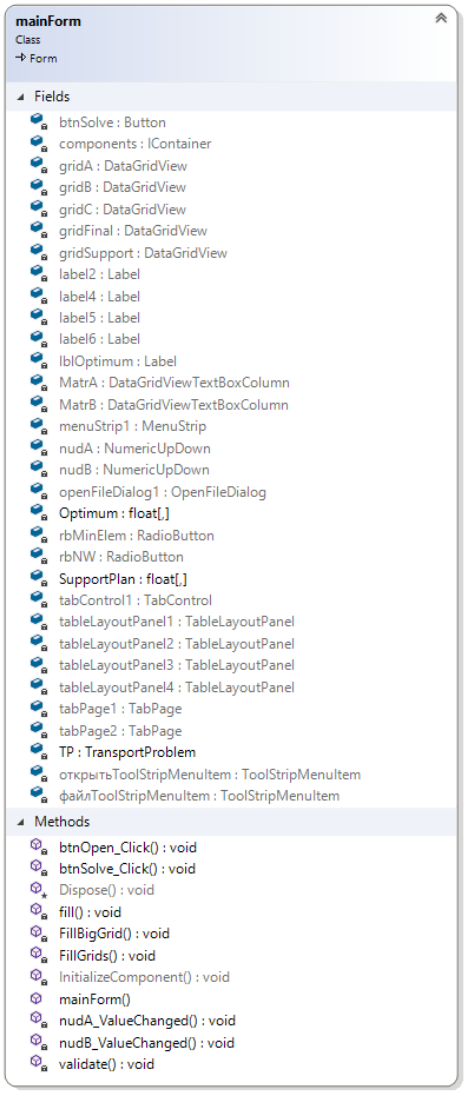

Рисунок 1.2 – Інтерфейсний клас головного вікна

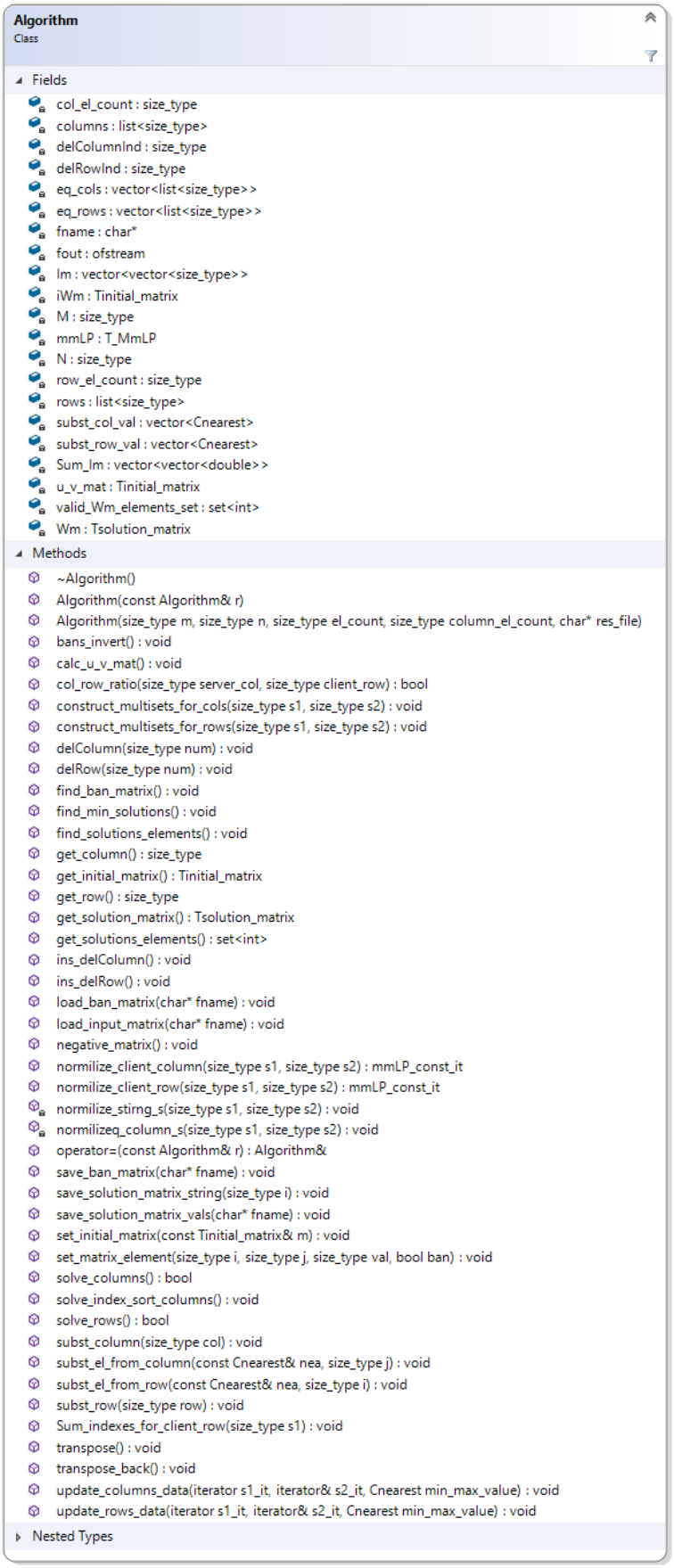

Рисунок 1.3 – Клас нормалізаційного методу

Класи сутностей мають наступний вигляд.

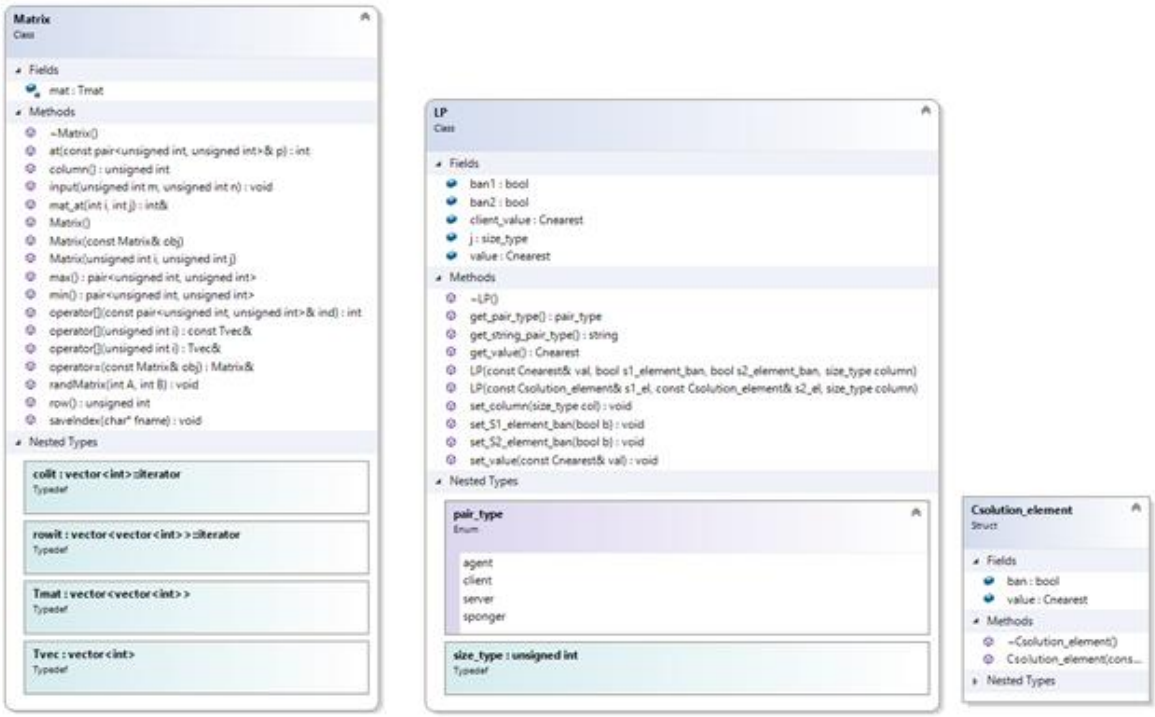

Рисунок 1.4 – Класи матриць ТЗ, Лінійної пари та Елемента розв’язку

Модель даних нормалізаційного алгоритму

Типи

size_type – тип лічильника;

edge_type – тип ваги ребра графа;

LP_value_type – тип значення (ваги) ЛП;

LP_type – тип (клас) ЛП;

MLP_type – тип МЛП;

MLP_iterator – тип покажчика на елемент МЛП.

Дані

M, N – кількість рядків і стовпців;

ini_B (M, N) – вихідна матриця суміжності дводольного графа (призначень);

B (M, N) – робоча матриця класів еквівалентності рядків і стовпців;

list <size_type> rows (M) – множина лінійно нерівних рядків розміру M;

list <size_type> cols (N) – множина лінійно нерівних стовпців;

vector <list <size_type>> eq_rows (M) – масив множин лінійно рівних рядків відповідного рядка;

vector <list <size_type>> eq_cols (N) – масив множин лінійно рівних стовпців відповідного стовпчика;

vector <LP_value_type> subst_col_val (M) – масив значень лінійних коефіцієнтів, які додаються до всіх елементів відповідних стовпців;

vector <LP_value_type> subst_row_val (N) – масив значень лінійних коефіцієнтів, які додаються до всіх елементів відповідних рядків;

bool line_equal – ознака закінчення алгоритму, дорівнює true, якщо всі рядки або стовпці лінійно рівні.

Службові функції.

min_element, max_element – повертають мінімальний елемент послідовності;

container.begin (), container.end () – метод отримання покажчика або ітератора початкового і граничного елемента послідовності container. Граничний елемент – це фіктивний елемент послідовності, який розташовується безпосередньо за останнім дійсним елементом.

container.size () – метод отримання розміру послідовності.

Основні методи класу нормалізаційного алгоритму

Головна функція Solve.

Функція складається з двох розділів: ініціалізації і рішення задачі пошуку 2-фактора. У другій розділ входять дві еквівалентні функції normalize_rows, і normalize_cols, які будують МЛП рядків і стовпців. У роботі наводиться опис однієї функції normalize_rows, тому що методи аналогічні.

Solve (in ini_B, in M, in N, out B)

1) Ініціалізація

h = N / M

for (i = 0; i <M; ++ i)

rows [i] = i

eq_rows [i] = i

subst_row_val [i] = 0

for (j = 0; j <N; ++ j)

cols [j] = j

eq_cols [j] = j

subst_col_val [j] = 0

for (i = 0; i <M; ++ i)

for (j = 0; j <N; ++ j)

B [i, j] = ini_B [i, j]

all_line_equal = false

2) Рішення

while (not all_line_equal)

- нормалізація рядків $O\left(n^3\right)$

normalize_rows (out B, out rows, in cols

, Out eq_rows, out subst_row_val

, Out all_line_equal)

- нормалізація стовпців $O\left(n^3\right)$

normalize_cols (out B, in rows, out cols

, Out eq_cols, out subst_ col_val

, Out all_line_equal)

Перший розділ ініціалізації має складність $O\left(n^2\right)$. Складність другого розділу рішення визначає оцінку всього алгоритму і становить $O\left(n^4\right)$. Вона визначається складністю одного з еквівалентних методів normalize_rows або normalize_cols, які будуть розглянуті далі.

Основна функція Normalize_rows.

Метод будує МЛП для кожної пари рядків робочої матриці *B*. Потім здійснює: пошук нормалізатора в кожному побудованому МЛП і нормалізацію заборонених елементів клієнтського рядку *B*. Далі алгоритм виконує перевірку лінійної рівності рядків. Якщо два рядки рівні, то вони об’єднуються у

відповідному класі еквівалентності функцією equivalence_rows. Зазначені дії виконуються, поки всі рядки не стануть лінійно рівними. Алгоритм повертає матрицю *B*, яка містить нормалізовані ваги ребер.

Функция normalize_rows.

normalize_rows (out B, out rows, in cols, out eq_rows, out subst_row_val, out all_line_equal)

1) Для всіх різноманітних пар (s1,s2) лінійно нерівних рядків

for(s1_it = rows.begin(); s1_it != rows.end(); ++s1_it)

s1 = *s1_it

for(s2_it = s1_it,++s2_it ; s2_it != rows.end(); ++s2_it)

s2 = *s2_it

2) Побудувати МЛП для лінійно нерівних рядків

MLP = construct_MLP(in B, in s1, in s2, in eq_cols)

3) Знайти нормалізатор

norm_it = find_norm(in MLP, in h, in s1, in s2, in eq_rows, in eq_cols)

4) Нормалізувати клієнтський рядок *s*1 по т. нормалізації

normilize_client_row(out B, in s1, in *norm_it)

MLP = construct_MLP(out B, in s2, in s1, in eq_cols)

norm_it = find_norm(in MLP, in h, in s2, in s1, in eq_rows, in eq_cols)

normilize_client_row(out B, in s2, in *norm_it)

5) Якщо рядки *s*1 и *s*2 лінійно рівні

if (min_element (MLP) == max_element (MLP))

об’єднати лінійно рівні рядки *s*1,*s*2 в один клас еквівалентності

обновити rows, eq_rows, subst_row_val

equivalence_rows(in s1, in s2, out rows, in cols, out eq_rows, out subst_row_val)

6) Якщо всі рядки лінійно рівні, то повернути *B*, all_line_equal – кінець

if(rows.size() == 2)

відновити матрицю еквівалентності *B* рядків і стовпців

restore(out B, in rows, in cols, in eq_rows, in eq_cols, in subst_row_val, in subst_col_val)

all_line_equal = true

return B, all_line_equal

Функція normilize_client_row.

Метод зменшує клієнтські значення максимальних ЛП мультимножини по теоремі нормалізації. Значення цих ЛП суворо більше нормалізатора.

normilize_client_row (B, s1, s2, * norm_it)

for (j_it = rows.begin (); j_it! = rows.end (); ++ j_it)

  if (B [s1, * j_it]> * norm_it) B [s1, * j_it] = B [s2, * j_it] + * norm_it

Метод має лінійну складність.

Функція equivalence_rows.

Метод додає *s*2 і всі рівні йому рядки в список лінійно рівних рядків рядка *s*1 (eq_rows [s1]), оновлює значення лінійних коефіцієнтів subst_row_val, видаляє рядок *s*2 з rows. Алгоритм має лінійну складність $O(n)$.

equivalence_rows (in s1, in s2, out rows, out eq_rows, out subst_row_val)

min_max_value = B [s1] [cols.begin ()] - B [s2] [cols.begin ()]

додати лінійно рівний рядок *s*2 в список eq_rows [s1]

eq_rows [s1] .push_back (s2)

оновити значення коефіцієнтів і список рівних *s*1 рядків

subst_row_val [s2] = subst_row_val [s2] - min_max_value

for (l_it = eq_rows [s2] .begin (); l_it! = eq_rows [s2] .end (); ++ l_it)

  eq_rows [s1] .push_back (* l_it)

  subst_row_val [* l_it] = subst_row_val [* l_it] - min_max_value

Функція restore.

restore (out B, in rows, in cols, in eq_rows, in eq_cols, in subst_row_val, in subst_col_val)

Алгоритм функції має складність $O(n^2)$. Він тривіальний, але досить громіздкий, тому не наводиться.

Функція містить два вкладених цикли, в яких відновлюються всі елементи «віддалених» лінійно рівних рядків і стовпців матриці *B*. Обчислення елементів реалізується на підставі лінійних коефіцієнтів subst_row_val і subst_col_val відповідних лінійно рівних рядків (rows і eq_rows) і стовпців (cols і eq_cols).

Функція construct_MLP (B, s1, s2, cols): MLP_type.

Простий алгоритм, який реалізує віднімання елементів рядків *s*1 і *s*2 індекси стовпців, яких визначаються безліччю cols. Алгоритм має лінійну складність.

Функція find_norm. Визначення складності функції normalize_rows.

Функція find_norm визначає покажчик нормалізатора. Для цього вона спочатку обчислює максимальне число елементів рядка ban_count, які можна заборонити за допомогою вхідного МЛП. А потім знаходить покажчик на n + 1-й максимальний елемент, він є нормалізатором. Якщо для заданого МЛП нормалізація неможлива, то функція повертає покажчик максимального елемента МЛП.

```
find_norm (in MLP, in h, in s1, in s2, in eq_rows, in eq_cols): MLP_iterator
// число лінійно рівних рядків для серверного рядка s2
sr_eq_row_count = eq_rows [s2] .size ()
// допустиму кількість стовпців для sr_eq_row_count
ban_count = sr_eq_row_count * h
it = max_element (MLP)
// пошук нормалізатора
k = eq_cols [it -> j] .size ()
while (k <= ban_count)
  MLP.erase (max_it)
it = max_element (MLP)
  k = k + eq_cols [it -> j] .size ()
```

return it

Складність алгоритму find_norm в гіршому випадку становить $O(n)$.

Складність функції normalize_rows в гіршому випадку становить $O(n^3)$.

Визначення складності нормалізаційного алгоритму. Функція solve.

Розглянемо фрагмент коду функції.

while (not all_line_equal)

  normalize_rows (...)

normalize_cols (...)

Нормалізація одного елемента становить $O(n^2)$, оскільки всього n^2 елементів, складність функції solve і нормалізаційного алгоритму становить $O(n^4)$.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

В даній кваліфікаційній роботі оброблюються дані користувача, які він вводить у форму додатка. Вони є вхідними даними. Вхідні дані перевіряються та валідуються, у разі не вірних даних користувачеві буде вивидено повідомленя з помилкою.

Оскільки додаток призначений тільки для розрахунку даних, то ніяких маніпуляцій з файлами не відбувається. Вихідними даними є результат транспортної задачі при розрахунку якої використовувались вхідни дані.

2.6. Опис розробленого додатку

2.6.1. Використані технічні засоби

При розробці та тестуванні системи була використана клієнтська персональна ЕОМ з наступними мінімальними характеристиками:

- процесор Intel Core i5 6500 3.6 GHz;

- відеокарта Radeon Rx 570;
- монітор 60Hz;
- не менше 2000Мб ОЗУ;
- 50Мб вільного місяця для розробленого додатку;
- клавіатура;
- комп’ютерна миша;

2.6.2. Використані програмні засоби

Для розробки додатку було використано безкоштовний IDE від Microsoft – Visual Studio. Версія IDE – Community 2019. Дизайн програми було розроблено за допомогою програм Adobe Photoshop 2019 та онлайн сервісу Figma.

2.6.3. Виклик та завантаження програми

Для початку роботи з програмою, необхідно запустити .exe файл у кореневій папці проекту. Назва файлу – «TransportTask.exe».

2.6.4. Опис інтерфейсу користувача

Результатом роботи є спроектований та розроблений застасунок, для якого також було розроблено унікальний користувацький інтерфейс.

При запуску програми, користувачу буде відображено наступне вікно (рис 2.13.):

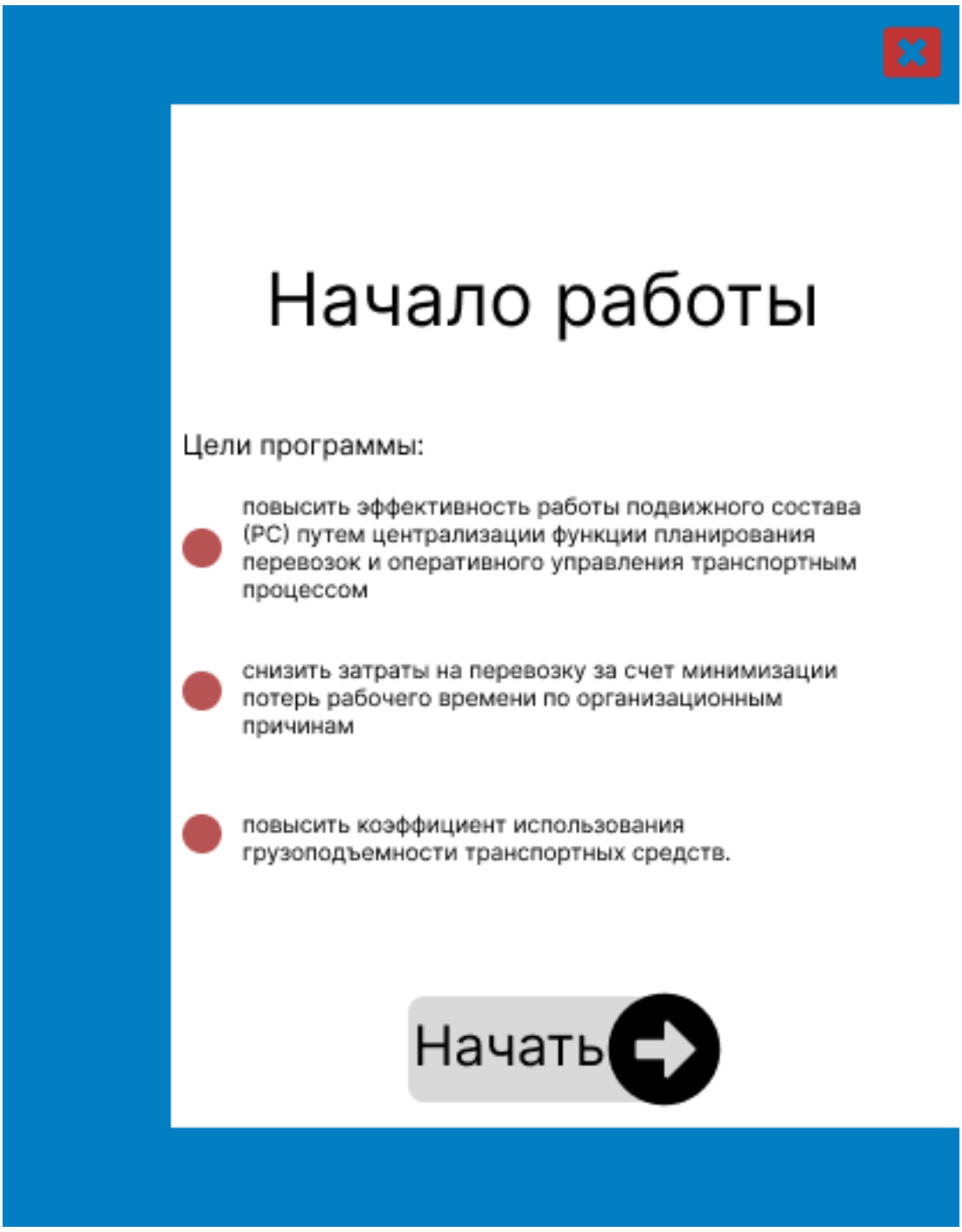

Рис. 2.1. Вигляд інтерфейсу застосунка при запуску

В цьому вікні користувачеві доступна інформація що до основних цілей програми. Також у цьому вікні є кнопка «Начать», після натискання на яку, користувачеві відображається головне меню.(рис. 2.1)

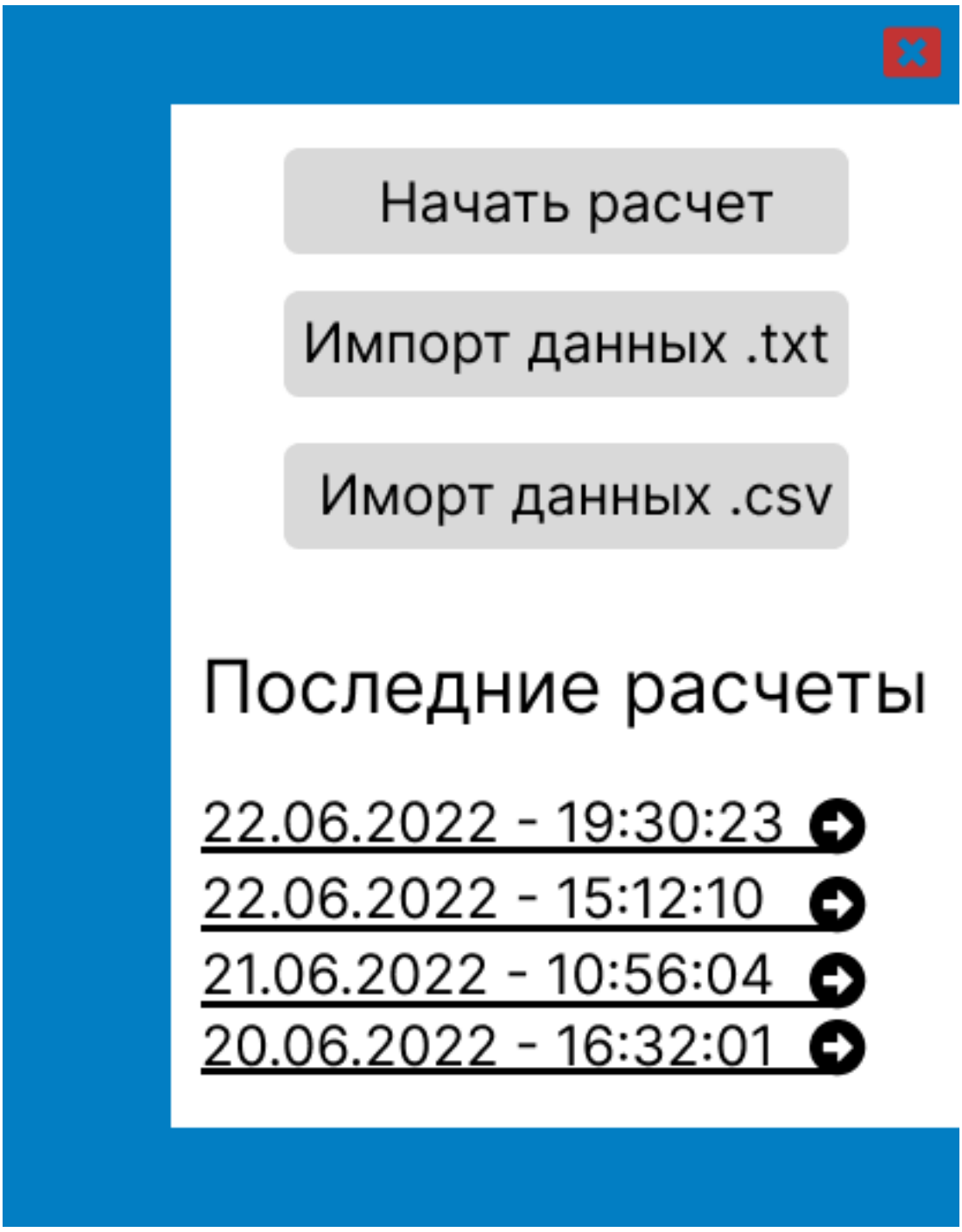

Рис. 2.2. Інтерфейс головного меню

У головному меню користувачеві досутпні такі функції:

Кнопка «Начать расчет» яка дозволяє перейти до сторінки вводу даних (рис. 2.3) для подальшого разрохунку задачи.

Кнопка «Импорт данных .txt» - при виборі цього варіанту користувачеві буде дозволено обрати файл формату .txt, при завантажені якого користувача буде направлено до сторінкі, на якій будуть імпортовані дані з файлу.

Кнопка «Импорт данных .csv» - при виборі цього варіанту користувачеві буде дозволено обрати файл формату .csv, при завантажені якого користувача буде направлено до сторінкі, на якій будуть імпортовані дані з файлу (рис. 2.3).

Список останніх розрахунків – список що містить останні чотири розрахунка. Вони зберігаються автоматично при виході з програми. Можливо обрати проект на якому ви залишились щоб продовжити розрахунки на тому місці на якому ви зупинилися.

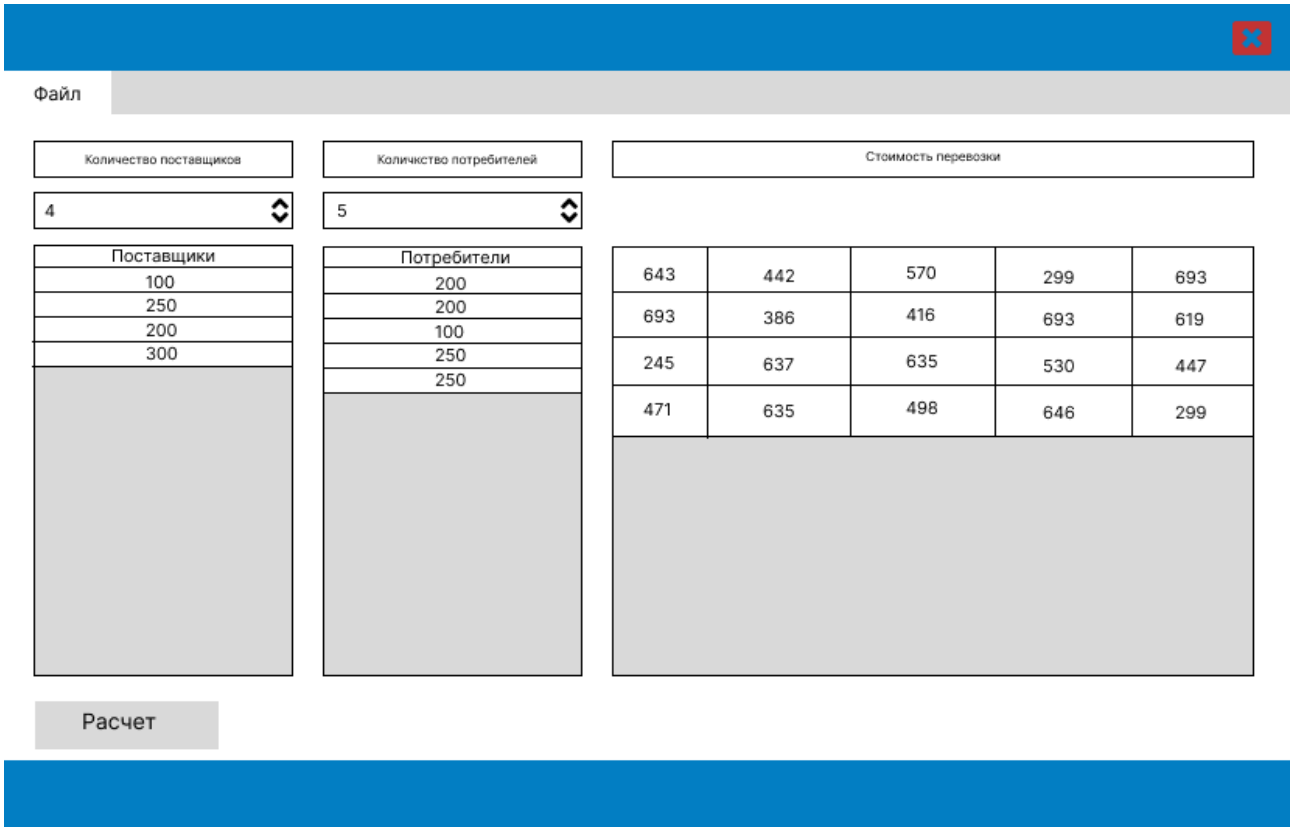

Рис. 2.3. Сторінка розрахунків

Це основна сторінка для розрахунка даних, вона використовується для виришеня транспортної задачі. Вхідними даними являються дані які користувач внес до програми використовуючи графічний інтерфейс або дані які було імпортовано з файлу формату .txt або .csv. Для отримання результату треба натиснути кнопку «Расчет» і відкриється вікно результата (рис. 2.3). Також є

кнопка «Файл» при натисканні на неї користувач бачe контекстне меню (рис. 2.4).

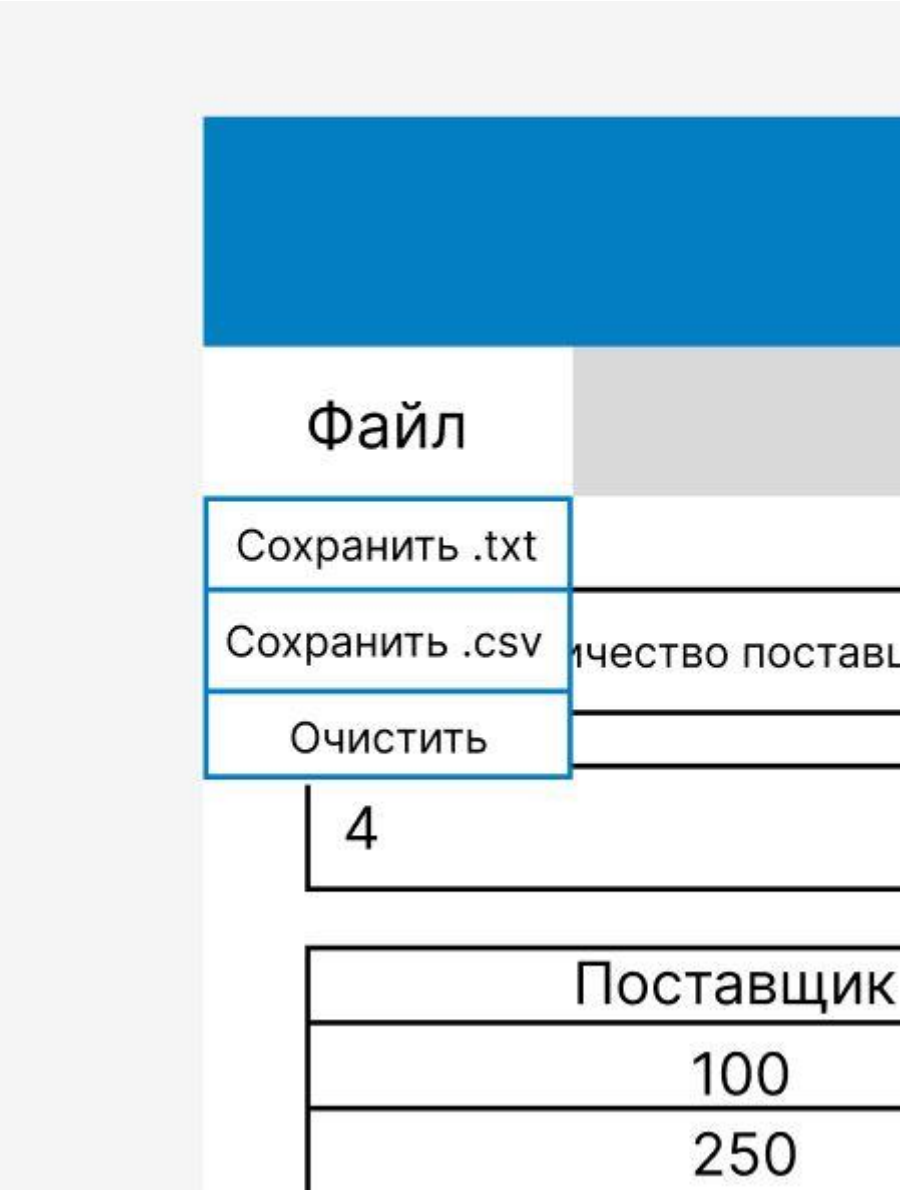

Рис. 2.4. Контекстне меню

В цьому меню є три кнопки:

– Кнопка «Сохранить .txt» - зберігає вхідні дані та результат у файл формата .txt.

– Кнопка «Сохранить .csv» - зберігає вхідні дані та результат у файл формата .csv.

– Кнопка «Очистить» - видаляє всі вхідні дані з полів та пам’яті

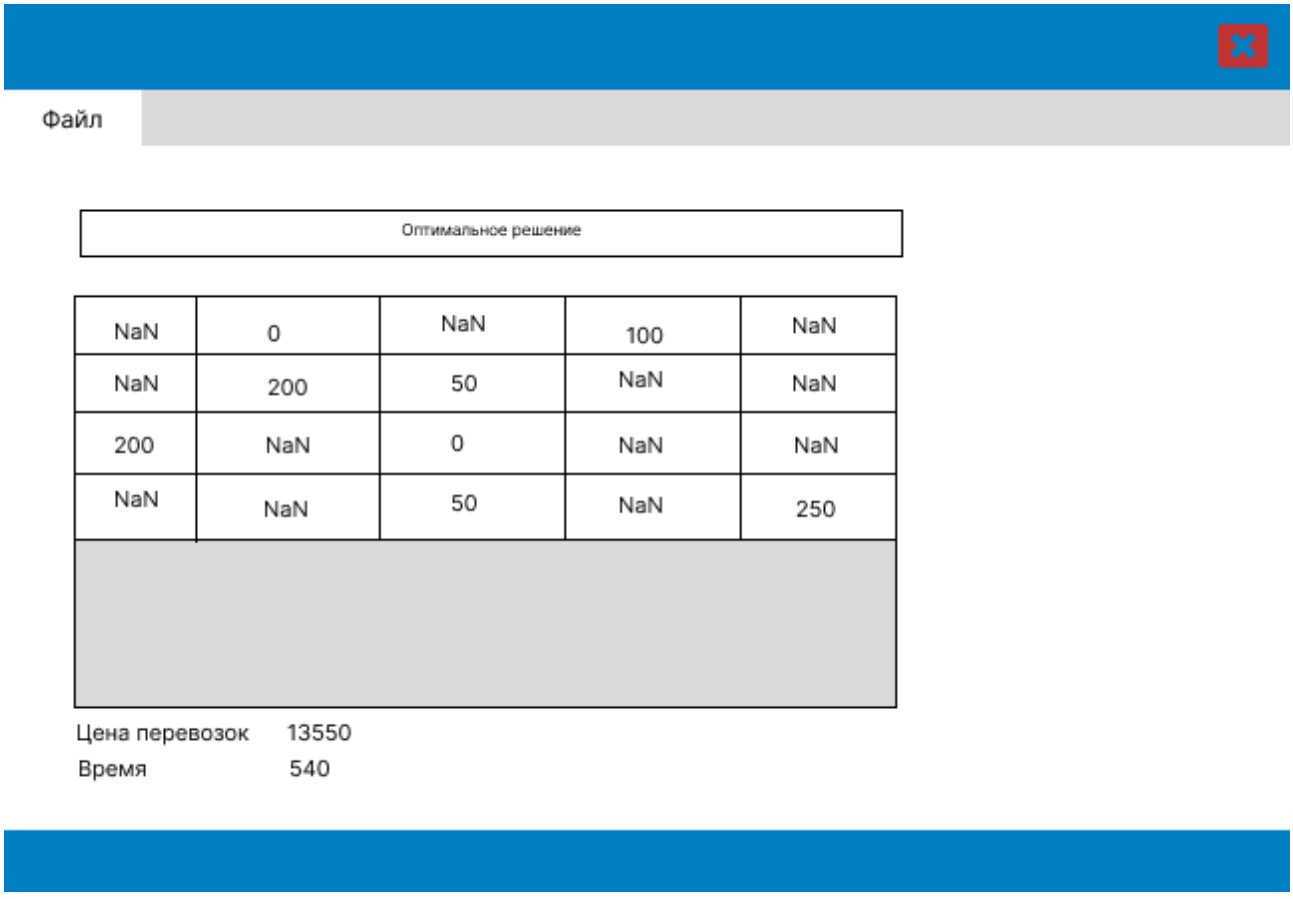

Рис. 2.5. Вікно результату

У цьому вікні відображено результат розрахунку транспортної задачі з вхідними даними які були взяті з попереднього вікна. Користувач дізнався скльки буде коштувати ціна перевізок та за який час пересилки будуть доставлені.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми - 524;
2. коефіцієнт складності програми – 1,1;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста – 140 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,1;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;
7. вартість машино-години ЕОМ – 14 грн/год

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u– витрати праці на дослідження алгоритму рішення задачі;

t_a– витрати праці на розробку блок-схеми алгоритму;

t_n– витрати праці на програмування по готовій блок-схемі;

$t_{отл}$– витрати праці на налагодження програми на ЕОМ;

$t_д$– витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q=q\cdot C\cdot(1+p), \text{ де} \qquad (3.2)$$

q– передбачуване число операторів;

C– коефіцієнт складності програми;

p– коефіцієнт кореляції програми в ході її розробки.

$$Q= 524\cdot 1,1\cdot(1+0,05) = 605,22;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \bullet B}{(75 \ldots 85) \bullet K}, \text{ людино-годин,} \qquad (3.3)$$

де *B* – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_u = \frac{605.22 \cdot 1,1}{80 \cdot 1,1} = 7,56, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25)\cdot k}, \text{ людино-годин.} \qquad (3.4)$$

де *Q* – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), людино-годин:

$$t_a = \frac{605,22}{20 \cdot 1,1} = 30,26, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot k}\text{, людино-годин.} \quad (3.5)$$

$$t_a = \frac{605{,}22}{25 \cdot 1{,}1} = 22\text{, людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\text{, людино-годин.} \quad (3.6)$$

$$t_n = \frac{605{,}22}{5 \cdot 1{,}1} = 110{,}04\text{, людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл}; \quad (3.7)$$

$$t^{к}_{отл} = 1{,}2 \cdot 110{,}04 = 132{,}04\text{, людино-годин}$$

Витрати праці на підготовку документації:

$$t_д = t_{дp} + t_{до}; \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_д = \frac{Q}{(15...20)\cdot K}; \quad (3.9)$$

$$t_д = \frac{605{,}22}{15 \cdot 1{,}1} = 36{,}68\text{, людино-годин.}$$

$t_{до}$– трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}; \quad (3.10)$$

$$t_{до} = 0{,}75 \cdot 36{,}68 = 27{,}51\text{, людино-годин.}$$

$$t_д = 27{,}51 + 36{,}68 = 55{,}02\text{, людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

t =50+7,56+30,26+110,04+132,04+36,68=366,58, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 366,58 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$– середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 366{,}58 \cdot 140 = 51321, \text{ грн.}$$

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$– вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 366{,}58 \cdot 14 = 5132, \text{ грн.}$$

$$К_{ПО} = 51321 + 5132 = 56453, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \quad (3.14)$$

де B_k- число виконавців;

$F_р$– місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{366{,}58}{1 \cdot 176} = 2{,}08 \text{ міс.}$$

Висновки. На розробку даного програмного забезпечення піде 366,58 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 2,08 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 56453 грн.

ВИСНОВКИ

В ході аналізу предметної області було досліджено основні проблеми оптимізації і можливі шляхи їх розв’язання.

Розроблено алгоритм розв’язання транспортної задачі з використанням блокової нормалізації та методу згортки функціоналу на основі виділення домінуючого критерію.

Розроблено застасунку для розв’язання транспортної задачі, який дозволяє складати ефективні плани перевезень:

Розроблена модель варіантів використання модулю.

Побудовано діаграму класів додатку, описано методи класів та дані.

Обгранкувано вибір мови програмування та інструментального засобу для реалізації системи.

Розроблено інтуїтивно зрозумілий інтерфейс програмного забеспечення.

Проведено тестування програмного додатку.

Проведено обчислювальний експеримент, та аналіз отриманих результатів.

На завершення, у «Економічному розділі» кваліфікаційної роботи було визначено трудомісткість розробленого програмного продукту, яка складає 366,58 люд-год, проведений підрахунок вартості роботи по створенню програми, яка є 56453 грн. та розраховано час на його створення, який становить приблизно 2,08 місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mark J. Price. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#. United Kingdom : Packt Publishing, 2019. 818 p.
2. Reed M. C#: 2 books in 1 - The Ultimate Beginner & Intermediate Guides to Mastering C# Programming Quickly (Computer Programming), 2022. 327 p.
3. Yoon H. The Art of C# - Basics: Introduction to Programming in Modern C# - Beginner to Intermediate (Learn Real Programming Book 3), 2021. 787 p.
4. Vystavel R. C# Programming for Absolute Beginners: Learn to Think Like a Programmer and Start Writing Code, 2021. 400 p.
5. Quickly C. Learn C# Quickly: A Complete Beginner's Guide to Learning C#, Even If You're New to Programming (Crash Course With Hands-On Project Book 2), 2020. 180p.
6. Jamro M. C# Data Structures and Algorithms: Explore the possibilities of C# for developing a variety of efficient applications, 2018. 292p.
7. Burns S. Hands-On Network Programming with C# and .NET Core: Build robust network applications with C# and .NET Core, 2019. 825 p.
8. Felicia P. C# Programming from Zero to Proficiency (Introduction): Learning C# Made Easy for Beginners, 2021. 75 p.
9. Bouras A. C# and Algorithmic Thinking for the Complete Beginner (2nd Edition): Learn to Think Like a Programmer, 2019. 746 p.
10. Yao R. C#: C# Programming, In 8 Hours, For Beginners, Quick Start Guide: C# Language, Crash Course Textbook & Exercises (In 8 Hours Series 1), 2022 202p.
11. Tudor S. C# - #2020 Updated: The Practical Beginners Guide to Learn C Programming and Coding in One Day Step by Step: With Effective Computer Languages Skills, 2020. 458p.
12. Perkins B. Beginning C# and .NET, 2021. 864 p.

13. Akella R. Enterprise Application Development with C# 10 and .NET 6: Become a professional .NET developer by learning expert techniques for building scalable applications, 2nd Edition, 2022. 586 p.

14. Sinyagin A. Refresher on .NET and Software Design Fundamentals for C# Developers, 2014. 348 p.

15. Vostokov D. Accelerated .NET Memory Dump Analysis: Training Course Transcript and WinDbg Practice Exercises, Second Edition, 2017. 634p.

16. Adewole A. C# and .NET Core Test Driven Development: Dive into TDD to create flexible, maintainable, and production-ready .NET Core applications, 2018. 300p.

17. Muzec A. How to Write Better C# Code: An Enjoyable and intuitive Approach to getting started with C# coding and Framework core using Visual Studio, 2020. 254p.

18. Sutherland A. C#: 2 books in 1: The New Beginner's & Intermediate Step by Step Guide to Learn C# in One Week. Including Projects and Exercise to Mastering C#, 2020. 299p.

19. Tudor S. C#: This book Includes: The Ultimate Beginner's And Intermediate's Guide To Learn C# Programming In One Day Step-By-Step, 2019. 255 p.

20. Anggoro W. Functional C#: Uncover the secrets of functional programming using C# and change the way you approach your applications forever, 2016. 370p.

ДОДАТОК А

КОД ПРОГРАМИ

mainForm.cs

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;

namespace lab1
{
 public partial class mainForm : Form
 {
  TransportProblem TP = null;
  float[,] SupportPlan = null;
  float[,] Optimum;

  public mainForm( )
  {
   InitializeComponent();
  }

  private void btnOpen_Click( object sender, EventArgs e )
  {
   Stream myStream = null;
   OpenFileDialog openFileDialog1 = new OpenFileDialog();

   openFileDialog1.InitialDirectory = "D:\\";
   openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
   openFileDialog1.FilterIndex = 1;
   openFileDialog1.RestoreDirectory = true;

   if ( openFileDialog1.ShowDialog() == DialogResult.OK )
   {
    try
    {
     if ( ( myStream = openFileDialog1.OpenFile() ) != null )
     {
      StreamReader SR = new StreamReader( myStream );
      String[] Sizes = SR.ReadLine().Split( ' ' );
      int Asize = 0, Bsize = 0;
      int.TryParse( Sizes[0], out Asize );
      int.TryParse( Sizes[1], out Bsize );
      String A = SR.ReadLine();
      String B = SR.ReadLine();
      String[] C = new String[Asize];
      for ( int i = 0; i < Asize; i++ ) C[i] = SR.ReadLine();
      try
      {
       TP = new TransportProblem( Asize, Bsize, A, B, C );
      }
```

```
        catch ( Exception exc )
        { MessageBox.Show( exc.Message ); }
      }
      myStream.Close();
    }
    catch ( Exception ex )
    {
      MessageBox.Show( "Error: Could not read file from disk. Original error: " + ex.Message );
    }
  }

  nudA.Value = TP.dilersCount;
  nudB.Value = TP.customersCount;

  gridA.RowCount = TP.dilersCount;
  gridB.RowCount = TP.customersCount;
  gridC.ColumnCount = TP.customersCount;
  gridC.RowCount = TP.dilersCount;

  FillGrids();
}

private void FillGrids( )
{
  for ( int i = 0; i < TP.dilersCount; i++ )
  {
    gridA.Rows[i].Cells[0].Value = TP.dilers[i].ToString();
    gridA.Rows[i].HeaderCell.Value = "A" + ( i + 1 ).ToString();
  }

  for ( int i = 0; i < TP.customersCount; i++ )
  {
    gridB.Rows[i].Cells[0].Value = TP.customers[i].ToString();
    gridB.Rows[i].HeaderCell.Value = "B" + ( i + 1 ).ToString();
  }

  FillBigGrid( gridC, TP.transportationPrices );
}

private void FillBigGrid( DataGridView grid, float[,] arr )
{
  for ( int i = 0; i < TP.dilersCount; i++ )
  {
    grid.Rows[i].HeaderCell.Value = "A" + ( i + 1 ).ToString();
    for ( int j = 0; j < TP.customersCount; j++ )
    {
      grid.Rows[i].Cells[j].Value = arr[i, j].ToString();
      grid.Columns[j].HeaderText = "B" + ( j + 1 ).ToString();
    }
  }
}
```

```
private void validate()
{
  float totalProducts = 0;
  for ( int i = 0; i < TP.dilersCount; ++i )
  {
    totalProducts += TP.dilers[i];
  }

  float totalNeeds = 0;
  for ( int i = 0; i < TP.customersCount; ++i )
  {
    totalNeeds += TP.customers[i];
  }

  if ( totalProducts != totalNeeds )
  {
    MessageBox.Show( "Количество товаров не соответствует потреблению" );
    /*float[] tmp = new float[TP.customersCount + 1];
    for ( int i = 0; i < TP.customersCount; ++i )
    {
      tmp[i] = TP.customers[i];
    }
    tmp[TP.customersCount] = totalProducts - totalNeeds;
    TP.customersCount++;
    TP.customers = tmp;*/
  }
  /*else if ( totalProducts < totalNeeds )
  {
    float[] tmp = new float[TP.dilersCount + 1];
    for ( int i = 0; i < TP.dilersCount; ++i )
    {
      tmp[i] = TP.dilers[i];
    }
    tmp[TP.dilersCount] = totalNeeds - totalProducts;
    TP.dilersCount++;
    TP.customers = tmp;
  }*/
}

private void nudA_ValueChanged( object sender, EventArgs e )
{
  int rowCount = Convert.ToInt32( nudA.Value );
  gridA.RowCount = rowCount;
  gridC.RowCount = rowCount;

  gridA.Rows[rowCount - 1].HeaderCell.Value = "A" + rowCount.ToString();
  gridC.Rows[rowCount - 1].HeaderCell.Value = "A" + rowCount.ToString();
}

private void nudB_ValueChanged( object sender, EventArgs e )
{
  int colCount = Convert.ToInt32( nudB.Value );
```

```
  gridB.RowCount = colCount;
  gridC.ColumnCount = colCount;

  gridB.Rows[colCount - 1].HeaderCell.Value = "B" + colCount.ToString();
  gridC.Columns[colCount - 1].HeaderText = "B" + colCount.ToString();
}

private void btnSolve_Click( object sender, EventArgs e )
{
  fill();
  validate();

  gridSupport.RowCount = TP.dilersCount;
  gridFinal.RowCount = TP.dilersCount;

  gridSupport.ColumnCount = TP.customersCount;
  gridFinal.ColumnCount = TP.customersCount;

  if ( rbNW.Checked )
  {
    SupportPlan = TP.NordWest();
  }
  else if ( rbMinElem.Checked )
  {
    SupportPlan = TP.MinEl();
  }

  FillBigGrid( gridSupport, SupportPlan );

  Optimum = TP.PotenMeth( SupportPlan );
  FillBigGrid( gridFinal, Optimum );

  float Sum = 0;
  for ( int i = 0; i < Optimum.Length; i++ )
  {
    int j = ( i - i % TP.customersCount ) / TP.customersCount;
    int k = i % TP.customersCount;
    if ( Optimum[j, k] == Optimum[j, k] )
      Sum += Optimum[j, k] * TP.transportationPrices[j, k];
  }
  lblOptimum.Text = "Цена перевозок: " + Sum.ToString();
}

private void fill( )
{
  TP = new TransportProblem();
  TP.dilersCount = gridA.RowCount;
  TP.customersCount = gridB.RowCount;

  TP.dilers = new float[TP.dilersCount];
  TP.customers = new float[TP.customersCount];
  TP.transportationPrices = new float[TP.dilersCount, TP.customersCount];
```

```
    try
    {
      for ( int i = 0; i < TP.dilersCount; ++i )
      {
        TP.dilers[i] = Convert.ToInt32( gridA.Rows[i].Cells[0].Value );
      }

      for ( int i = 0; i < TP.customersCount; ++i )
      {
        TP.customers[i] = Convert.ToInt32( gridB.Rows[i].Cells[0].Value );
      }

      for ( int i = 0; i < TP.dilersCount; ++i )
      {
        for ( int j = 0; j < TP.customersCount; ++j )
        {
          TP.transportationPrices[i, j] = Convert.ToInt32( gridC.Rows[i].Cells[j].Value );
        }
      }
    }
    catch (System.Exception ex)
    {
      MessageBox.Show( "ex" );
    }
  }

      private void label6_Click(object sender, EventArgs e)
      {

      }

      private void gridB_CellContentClick(object sender, DataGridViewCellEventArgs e)
      {

      }
  }

  public class TransportProblem
{
  class InvalidInpFormat : ApplicationException
  {
    public InvalidInpFormat( ) : base() { }
    public InvalidInpFormat( string str ) : base( str ) { }
    public override string ToString( )
    {
      return Message;
    }
  }
  // склады
  public float[] dilers;
  // потребители
```

```
    public float[] customers;
    // Издержки
    public float[,] transportationPrices;
    public int dilersCount;
    public int customersCount;

    // Конструкторы
    public TransportProblem( float[] nA, float[] nB, float[,] nC )
    {
      if ( ( nA.Length != nC.GetLength( 0 ) ) || ( nB.Length != nC.GetLength( 1 ) ) )
        throw new InvalidInpFormat( "Размеры массива затрат не соответствуют размерам
массивов поставщиков и складов" );

      this.dilers = nA;
      this.customers = nB;
      this.transportationPrices = nC;

      this.dilersCount = nA.Length;
      this.customersCount = nB.Length;
    }

    public TransportProblem( int _Asize, int _Bsize, string sA, string sB, string[] sC )
    {
      dilersCount = _Asize; customersCount = _Bsize;
      float x = 0;
      string[] StrArr = sA.Split( ' ' );
      if ( StrArr.Length != dilersCount )
        throw new InvalidInpFormat( "Размеры массива A не соответствуют заявленным" );
      dilers = new float[dilersCount];
      for ( int i = 0; i < dilers.Length; i++ ) if ( float.TryParse( StrArr[i], out x ) ) dilers[i] = x;

      StrArr = sB.Split( ' ' );
      if ( StrArr.Length != customersCount )
        throw new InvalidInpFormat( "Размеры массива B не соответствуют заявленным" );
      customers = new float[customersCount];
      for ( int i = 0; i < customers.Length; i++ ) if ( float.TryParse( StrArr[i], out x ) ) customers[i] =
x;

      float sumA = 0;
      Array.ForEach( dilers, delegate( float f ) { sumA += f; } );
      float sumB = 0;
      Array.ForEach( customers, delegate( float f ) { sumB += f; } );
      float dif = sumA - sumB;
      if ( dif > 0 )
      {
        float[] bufArr = customers;
        customers = new float[bufArr.Length + 1];
        bufArr.CopyTo( customers, 0 );
        customers[customers.Length - 1] = Math.Abs( dif );
        customersCount++;
      }
      else if ( dif < 0 )
```

```
      {
        float[] bufArr = dilers;
        dilers = new float[bufArr.Length + 1];
        bufArr.CopyTo( dilers, 0 );
        dilers[dilers.Length - 1] = Math.Abs( dif );
        dilersCount++;
      }

      transportationPrices = new float[dilersCount, customersCount];
      for ( int j = 0; j < sC.Length; j++ )
      {
        StrArr = sC[j].Split( ' ' );
        if ( StrArr.Length != _Bsize )
          throw new InvalidInpFormat( "Длина одной из строк входного файла не соответствует
длине массива B" );
        for ( int i = 0; i < _Bsize; i++ ) if ( float.TryParse( StrArr[i], out x ) ) transportationPrices[j, i]
= x;
      }
    }

    public TransportProblem( )
    {
    }

    // Строим опорные планы тут
    bool isEmpty( float[] arr )
    {
      return Array.TrueForAll( arr, delegate( float x ) { return x == 0; } );
    }

    private void NanToEmpty( float[,] outArr )
    {
      int i = 0, j = 0;
      for ( i = 0; i < dilersCount; i++ )
        for ( j = 0; j < customersCount; j++ )
          if ( outArr[i, j] == 0 ) outArr[i, j] = float.NaN;
    }

    float findMin( float[,] Arr, bool[,] pr, out int indi, out int indj )
    {
      indi = -1; indj = -1;
      float min = float.MaxValue;
      for ( int i = 0; i < dilersCount; i++ )
        for ( int j = 0; j < customersCount; j++ )
          if ( ( pr[i, j] ) && ( Arr[i, j] < min ) )
          {
            min = Arr[i, j];
            indi = i; indj = j;
          }
      return min;
    }
    // Метод северо-западного угла
```

```
public float[,] NordWest( )
{
  float[] Ahelp = dilers;
  float[] Bhelp = customers;
  int i = 0, j = 0;
  float[,] outArr = new float[dilersCount, customersCount];
  NanToEmpty( outArr );
  while ( !( isEmpty( Ahelp ) && isEmpty( Bhelp ) ) )
  {
    float Dif = Math.Min( Ahelp[i], Bhelp[j] );
    outArr[i, j] = Dif;
    Ahelp[i] -= Dif; Bhelp[j] -= Dif;
    if ( ( Ahelp[i] == 0 ) && ( Bhelp[j] == 0 ) && ( j + 1 < customersCount ) )
    {
      outArr[i, j + 1] = 0;
    }
    if ( Ahelp[i] == 0 )
    {
      i++;
    }
    if ( Bhelp[j] == 0 )
    {
      j++;
    }

    if ( i >= dilersCount || j >= customersCount )
    {
      break;
    }
  }
  return outArr;
}

class FindWay
{
  FindWay Father;
  Point Root;
  FindWay[] Childrens;
  Point[] mAllowed;
  Point Begining;
  bool flag;
  public FindWay( int x, int y, bool _flag, Point[] _mAllowed, Point _Beg, FindWay _Father )
  {
    Begining = _Beg;
    flag = _flag;
    Root = new Point( x, y );
    mAllowed = _mAllowed;
    Father = _Father;
  }
  public Boolean BuildTree( )
  {
    Point[] ps = new Point[mAllowed.Length];
```

```
    int Count = 0;
    for ( int i = 0; i < mAllowed.Length; i++ )
      if ( flag )
      {
        if ( Root.Y == mAllowed[i].Y )
        {
          Count++;
          ps[Count - 1] = mAllowed[i];
        }

      }
      else
        if ( Root.X == mAllowed[i].X )
        {
          Count++;
          ps[Count - 1] = mAllowed[i];
        }

    FindWay fwu = this;
    Childrens = new FindWay[Count];
    int k = 0;
    for ( int i = 0; i < Count; i++ )
    {
      if ( ps[i] == Root ) continue;
      if ( ps[i] == Begining )
      {
        while ( fwu != null )
        {
          mAllowed[k] = fwu.Root;
          fwu = fwu.Father;
          k++;
        };
        for ( ; k < mAllowed.Length; k++ ) mAllowed[k] = new Point( -1, -1 );
        return true;
      }

      if ( !Array.TrueForAll<Point>( ps, p => ( ( p.X == 0 ) && ( p.Y == 0 ) ) ) )
      {
        Childrens[i] = new FindWay( ps[i].X, ps[i].Y, !flag, mAllowed, Begining, this );
        Boolean result = Childrens[i].BuildTree();
        if ( result ) return true;
      }
    }
    return false;
  }

}

// Метод минимального элемента
public float[,] MinEl( )
{
  float[] Ahelp = this.dilers;
```

```
float[] Bhelp = this.customers;
int i = 0;
int j = 0;
float min = float.MaxValue;
float[,] outArr = new float[this.dilersCount, this.customersCount];
bool[,] pArr = new bool[this.dilersCount, this.customersCount];
for ( i = 0; i < this.dilersCount; i++ )
{
  for ( j = 0; j < this.customersCount; j++ )
  {
    pArr[i, j] = true;
  }
}
i = 0;
j = 0;
int k;
int count = 0;
while ( !this.isEmpty( Ahelp ) || !this.isEmpty( Bhelp ) )
{
  min = this.findMin( this.transportationPrices, pArr, out i, out j );
  float Dif = Math.Min( Ahelp[i], Bhelp[j] );
  outArr[i, j] += Dif; count++;
  Ahelp[i] -= Dif;
  Bhelp[j] -= Dif;
  if ( Ahelp[i] == 0f )
  {
    k = 0;
    while ( k < this.customersCount )
    {
      pArr[i, k] = false;
      k++;
    }
  }
  if ( Bhelp[j] == 0f )
  {
    for ( k = 0; k < this.dilersCount; k++ )
    {
      pArr[k, j] = false;
    }
  }
}
this.NanToEmpty( outArr );

// Нуль-загрузка
int difference = ( dilersCount + customersCount - 1 ) - count;
for ( int l = 0; l < difference; l++ )
{
  //выбираем непустые
  Allowed = new Point[count + 1];
  k = 0;
  for ( i = 0; i < dilersCount; i++ )
    for ( j = 0; j < customersCount; j++ )
```

```
        if ( outArr[i, j] == outArr[i, j] )
        {
          Allowed[k] = new Point( i, j );
          k++;
        }
    // ищем куда загрузить
    Boolean p = true;
    Point Nl = new Point( 0, 0 );
    for ( i = 0; ( i < dilersCount ) && p; i++ )
      for ( j = 0; ( j < customersCount ) && p; j++ )
      {
        Nl = Allowed[9] = new Point( i, j );
        FindWay fw = new FindWay( i, j, true, Allowed, new Point( i, j ), null );
        p = fw.BuildTree();
      }
    if ( !p ) outArr[Nl.X, Nl.Y] = 0;
  }

  return outArr;
}

// Оптимизация методом потенциалов
// вспомогательные функции
// функция заполняет вспомогательные массивы U и V
// пока работает...
private void FindUV( float[] U, float[] V, float[,] HelpMatr )
{
  //для проверки вычислена ли Ui Vi будем использовать массив boolean'ов
  //даже 2 массива. в одном признак того вычислена ли соответствующий потенциал
  //во втором прошлись ли мы по строке/строчке этого потенциала
  //алгоритм позволит за конечное число итераций вычислить все потенциалы. ура.
  bool[] U1 = new bool[dilersCount];
  bool[] U2 = new bool[dilersCount];
  bool[] V1 = new bool[customersCount];
  bool[] V2 = new bool[customersCount];
  //V[BSize - 1] = 0;
  //V1[BSize - 1] = true;
  // пока все элементы массивов V1 и U1 не будут равны true
  while ( !( AllTrue( V1 ) && AllTrue( U1 ) ) )
  {
    int i = -1;
    int j = -1;
    for ( int i1 = customersCount - 1; i1 >= 0; i1-- )
      if ( V1[i1] && !V2[i1] ) i = i1;
    for ( int j1 = dilersCount - 1; j1 >= 0; j1-- )
      if ( U1[j1] && !U2[j1] ) j = j1;

    if ( ( j == -1 ) && ( i == -1 ) )
      for ( int i1 = customersCount - 1; i1 >= 0; i1-- )
        if ( !V1[i1] && !V2[i1] )
        {
          i = i1;
```

```
          V[i] = 0;
          V1[i] = true;
          break;
        }
    if ( ( j == -1 ) && ( i == -1 ) )
      for ( int j1 = dilersCount - 1; j1 >= 0; j1-- )
        if ( !U1[j1] && !U2[j1] )
        {
          j = j1;
          U[j] = 0;
          U1[j] = true;
          break;
        }

    if ( i != -1 )
    {
      for ( int j1 = 0; j1 < dilersCount; j1++ )
      {
        if ( !U1[j1] ) U[j1] = HelpMatr[j1, i] - V[i];
        if ( U[j1] == U[j1] ) U1[j1] = true;
      }
      V2[i] = true;
    }

    if ( j != -1 )
    {
      for ( int i1 = 0; i1 < customersCount; i1++ )
      {
        if ( !V1[i1] ) V[i1] = HelpMatr[j, i1] - U[j];
        if ( V[i1] == V[i1] ) V1[i1] = true;
      }
      U2[j] = true;
    }

  }
}

private Boolean AllPositive( float[,] m )
{
  Boolean p = true;
  for ( int i = 0; ( i < dilersCount ) && p; i++ )
    for ( int j = 0; ( j < customersCount ) && p; j++ )
      if ( m[i, j] < 0 ) p = false;
  return p;
}

private bool AllTrue( bool[] arr )
{
  return Array.TrueForAll( arr, delegate( bool x ) { return x; } );
}

// дозаполняет матрицу S оценками
```

```
  private float[,] MakeSMatr( float[,] M, float[] U, float[] V )
  {

    float[,] HM = new float[dilersCount, customersCount];
    for ( int i = 0; i < dilersCount; i++ )
      for ( int j = 0; j < customersCount; j++ )
      {
        HM[i, j] = M[i, j];
        if ( HM[i, j] != HM[i, j] )
          HM[i, j] = transportationPrices[i, j] - ( U[i] + V[j] );
      }
    return HM;
  }

  private Point[] Allowed;// хранит координаты клеток, в которых есть груз

  public int[] arra = new int[5];

  private Point[] GetCycle( int x, int y )
  {
    Point Beg = new Point( x, y );
    FindWay fw = new FindWay( x, y, true, Allowed, Beg, null );
    fw.BuildTree();
    Point[] Way = Array.FindAll<Point>( Allowed, delegate( Point p ) { return ( p.X != -1 ) && (
p.Y != -1 ); } );
    return Way;
  }

  // находит плохой цикл и крутит его
  private void Roll( float[,] m, float[,] sm )
  {
    Point minInd = new Point();
    float min = float.MaxValue;
    int k = 0;
    Allowed = new Point[dilersCount + customersCount];
    for ( int i = 0; i < dilersCount; i++ )
      for ( int j = 0; j < customersCount; j++ )
      {
        if ( m[i, j] == m[i, j] )
        {
          Allowed[k].X = i;
          Allowed[k].Y = j;
          k++;
        }
        // заодно ищем макс по модулю отр элемент
        if ( sm[i, j] < min )
        {
          min = sm[i, j];
          minInd.X = i;
          minInd.Y = j;
        }
      }
```

```
  // Ищем цикл
  Allowed[Allowed.Length - 1] = minInd;
  Point[] Cycle = GetCycle( minInd.X, minInd.Y );
  float[] Cycles = new float[Cycle.Length];
  Boolean[] bCycles = new Boolean[Cycle.Length];
  for ( int i = 0; i < bCycles.Length; i++ )
    bCycles[i] = i == bCycles.Length - 1 ? false : true;
  min = float.MaxValue;
  /* проблема в следующем:
   * цикл мы находим правильно
   * а вот посчитать правильно не можем
   * ниже поиск минимального элемента
   */
  // поиск минимального
  for ( int i = 0; i < Cycle.Length; i++ )
  {
    Cycles[i] = m[Cycle[i].X, Cycle[i].Y];
    if ( ( i % 2 == 0 ) && ( Cycles[i] == Cycles[i] ) && ( Cycles[i] < min ) )
    {
      min = Cycles[i];
      minInd = Cycle[i];
    }
    if ( Cycles[i] != Cycles[i] ) Cycles[i] = 0;
  }
  // вычитание-прибавление
  for ( int i = 0; i < Cycle.Length; i++ )
  {
    if ( i % 2 == 0 )
    {
      Cycles[i] -= min;
      m[Cycle[i].X, Cycle[i].Y] -= min;
    }
    else
    {
      Cycles[i] += min;
      if ( m[Cycle[i].X, Cycle[i].Y] != m[Cycle[i].X, Cycle[i].Y] ) m[Cycle[i].X, Cycle[i].Y] = 0;
      m[Cycle[i].X, Cycle[i].Y] += min;
    }
  }
  m[minInd.X, minInd.Y] = float.NaN;
}

// сама оптимизация
public float[,] PotenMeth( float[,] SupArr )
{
  // расчитываем Ui и Vi
  //подготовка
  int i = 0, j = 0;
  float[,] HelpMatr = new float[dilersCount, customersCount];
  for ( i = 0; i < dilersCount; i++ )
    for ( j = 0; j < customersCount; j++ )
      if ( SupArr[i, j] == SupArr[i, j] ) HelpMatr[i, j] = transportationPrices[i, j];
```

```
        else HelpMatr[i, j] = float.NaN;

      //расчёт
      float[] U = new float[dilersCount];
      float[] V = new float[customersCount];
      FindUV( U, V, HelpMatr );
      float[,] SMatr = MakeSMatr( HelpMatr, U, V );
      //пока все потенциалы не станут положительнымми, будем снова и снова считать
      while ( !AllPositive( SMatr ) )
      {
        Roll( SupArr, SMatr );
        for ( i = 0; i < dilersCount; i++ )
          for ( j = 0; j < customersCount; j++ )
          {
            if ( SupArr[i, j] == float.PositiveInfinity )
            {
              HelpMatr[i, j] = transportationPrices[i, j];
              SupArr[i, j] = 0;
              continue;
            }
            if ( SupArr[i, j] == SupArr[i, j] ) HelpMatr[i, j] = transportationPrices[i, j];
            else HelpMatr[i, j] = float.NaN;
          }
        FindUV( U, V, HelpMatr );
        SMatr = MakeSMatr( HelpMatr, U, V );
      }

      return SupArr;
    }
  }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

Перелік файлів на диску

Перелік документів на магнітному носії

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Шорінов.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Шорінов.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Шорінов.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Шорінов.pptx	Презентація кваліфікаційної роботи