

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ВИРІШЕННЯ ЗАДАЧІ КОМІВОЯЖЕРА
ДЛЯ АВТОМАТУ ВИДАЧІ ЛІКІВ

SOFTWARE IMPLEMENTATION OF PROBLEM SOLVING WITH A
DRUG DISPENSING MACHINE FOR A TRAVELLING SALESMAN

Виконав: студент 4 курсу, групи КНТ-118сп
Спеціальності 121 Інженерія програмного
забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Головченко А.Л.
(прізвище та ініціали)

Керівник Степаненко О.О.
(прізвище та ініціали)

Рецензент Голуб Т.В.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Головченка Артема Леонідовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація методів вирішення задачі комівояжера для автомату видачі ліків. Software Implementation of Problem Solving with a Drug Dispensing Machine for a Travelling Salesman
керівник проєкту (роботи) Степаненко О. О., к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання),
затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103
2. Строк подання студентом проєкту (роботи) 14 травня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Реферат. Перелік скорочень та умовних позначок. Вступ. 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Висновки. Перелік джерел посилання. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Степаненко О.О., доцент		
Нормоконтроль	Камінська Ж. К., асистент		

7. Дата видачі завдання “ 20 ” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділи 4-5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування.	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент

(підпис) Головченко А. Л.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Степаненко О.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
110 с., 13 табл., 12 рис., 4 дод., 25 джерел.

ЗАДАЧА КОМІВОЯЖЕРА, ЖАДІБНИЙ АЛГОРИТМ, АПТЕКА,
АВТОМАТ ВИДАЧІ ЛІКІВ, C#.

Об'єкт дослідження – автоматизовані системи видачі ліків в аптеці.

Предмет дослідження – методи пошуку оптимального маршруту для маніпулятора автомата видачі ліків.

Мета роботи – створення програмної системи для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

Актуальність та проблематика роботи полягає в тому, що для скорочення часу доставки великої кількості ліків за допомогою маніпулятора автомата видачі ліків виникла необхідність розробки методів пошуку оптимального маршруту для маніпулятора автомата видачі ліків.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування C#, персональний комп'ютер з процесором Intel Core i5-6400 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмний продукт, який дозволяє оптимізувати роботу автомата видачі ліків, шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

Висновки. Розроблено програмну систему для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

Галузь використання – продаж та видача ліків фармацевтами у аптечних закладах.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 110 pages, 13 figures, 12 tables, 4 appendixes, 25 sources.

TRAVELLING SALESMAN PROBLEM, GREEDY ALGORITHM, PHARMACY, DRUG DISPENSING MACHINE, C#.

The object of study is automated drug delivery systems in pharmacies.

The subject of study is methods of finding the optimal route for the manipulator of the drug dispensing machine.

The purpose of the work is to create a software system to optimize the operation of the drug dispenser machine by finding the optimal route by solving the problem of the salesman.

The relevance and problems of the work is that to reduce the delivery time of a large number of drugs using the manipulator of the drug dispenser there is a need to develop methods to find the optimal route for the manipulator of the dispenser.

Materials, methods and technical means is structural and object-oriented programming, C # programming language, personal computer with Intel Core i5-6400 processor running Microsoft Windows 10 operating system.

Results. A software has been created that allows optimizing the operation of the dispensing machine by finding the optimal route by solving the problem of a salesman.

Conclusions. A software system has been developed to optimize the operation of the drug dispenser by finding the optimal route by solving the problem of a salesman.

Field of use is sale and dispensing of drugs by pharmacists in pharmacies.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	10
1.1 Основні поняття	10
1.2 Огляд існуючих автоматів видачі ліків	11
1.3 Огляд існуючих методів	12
1.4 Порівняльний аналіз існуючих методів вирішення завдання	21
1.7 Постановка завдання до роботи	22
1.8 Висновок	24
2 Вибір та обґрунтування структури системи, яка проєктується	25
2.1 Вибір інструментарію розробки програмного продукту	25
2.2 Структурна схема програмного продукту	32
2.3 Висновок	37
3 Основні рішення щодо реалізації компонентів системи	38
3.1 Метод пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків	38
3.2 Метод забезпечення авторизації користувачів	46
3.3 Проєктування інтерфейсу	47
3.4 Висновок	54
4 Керівництво програміста	55
4.1 Призначення й умови використання	55
4.2 Характеристики програмного продукту	55
4.3 Звертання до програмного продукту	56
4.4 Вхідні і вихідні дані	56
4.5 Повідомлення	57
4.6 Висновок	58
5 Керівництво оператора	59
5.1 Призначення й умови виконання програмного продукту	59

5.2 Виконання програмного продукту.....	59
5.3 Повідомлення оператору	63
5.4 Висновок	63
Висновки	64
Перелік джерел посилання.....	66
Додаток А Технічне завдання	69
Додаток Б Опис програмного продукту	75
Додаток В Текст програми.....	85
Додаток Д Слайди презентації.....	102

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- БД – база даних;
ПБ – прізвище, ім'я, по батькові;
ПП – програмний продукт;
СУБД – система управління базами даних;

ВСТУП

На сьогоднішній день автоматизація застосовується в багатьох сферах та галузях життя, в тому числі і в медицині. У багатьох країнах світу з'являються автоматизовані аптеки, в яких застосовуються автомати видачі ліків. Поступово ці автомати з'являються в тому числі і в Україні. Автомат видачі ліків виконує пошук потрібних ліків на складі і за допомогою механізму переміщення доставляє ліки покупцю. Використання подібного автомату дозволяє полегшити та оптимізувати роботу фармацевтів, тим самим обслуговування клієнтів проходить набагато швидше, а також це дозволяє зменшити вплив людського фактору, що знижує кількість помилок при пошуку ліків.

Але шлях, який долає маніпулятор автомату видачі ліків може бути не завжди оптимальним. Через це збільшується час обслуговування кожного клієнта, і таким чином автоматизація перестає бути достатньо ефективною. Для того, щоб усунути цю проблему було створено ПП, в якому було застосовано різні алгоритми розв'язання задачі комівояжера. Задача комівояжера це одна із задач комбінаторної оптимізації, вона полягає в тому, щоб знайти найвигідніший маршрут переміщення через задані точки з усіх можливих.

Мета роботи – створення програмної системи для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

В результаті створено ПП, за допомогою якого можна оптимізувати роботу автомата видачі ліків, шляхом вирішення задачі комівояжера.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Аптека – медично-санітарний заклад, що виготовляє і відпускає ліки за рецептами, продає готові лікарські засоби, які дозволено відпускати без рецепта, лікарські мінеральні води, перев'язувальні матеріали, предмети догляду за хворими та інші медичні вироби [1].

Ліки – речовини рослинного, тваринного, мікробного, синтетичного походження, які при введенні в організм людини або тварини нормалізують порушену функцію органа чи організму в цілому або запобігають розвитку хвороб.

Автомат видачі ліків – міні-складське обладнання, яке встановлюється в аптеці, аптечних складах і медичних установах для оптимізації зберігання, пошуку і видачі медикаментів до робочого місця фармацевта (першого столу) або для продажу безпосередньо покупцеві [2].

Спираючись на потребу в автоматизації логістичних процесів аптек з великим товарообігом і прохідністю в аптеці, німецькі інженери розробили робота, що дозволяє економити час на пошук ліків на складі на користь консультації покупця. На виставці Ecxorpharm в Мюнхені в 1996 році був представлений перший в світі робот-фармацевт для автоматизації видачі найбільш затребуваних медикаментів в аптеці. Пізніше подібні системи стали впроваджуватися в госпітальної та аптечній системі США [3].

Автомат видачі ліків виконує наступні функції:

- зберігання аптечних товарів у відповідних умовах;
- прийом товарів для подальшого їх зберігання.
- здійснення пошуку потрібних аптечних товарів за різними параметрами;
- видача аптечних товарів користувачу.

Автомат видачі ліків має полиці, на яких розташовані аптечні товари. Товари доставляються до користувача за допомогою маніпулятора, який має

електропривід для переміщення по автоматі та захват для того, щоб взяти ліки з полиці і передати їх покупцю. Також автомат видачі ліків може мати різні датчики для більш точного переміщення. Управління автоматом виконується за допомогою спеціального програмного забезпечення зі зручним для користувача інтерфейсом.

1.2 Огляд існуючих автоматів видачі ліків

На сьогоднішній день існує багато різних автоматів видачі ліків, особливості деяких з них наведені в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика існуючих автоматів видачі ліків

Назва автомату видачі ліків	Особливості автомату видачі ліків
EvoTec [4]	Зберігає від 5000 до 42000 різних типів ліків, дозволяє автоматично завантажувати ліки, має функцію автоматичного регулювання полиць з ліками, має зручний сенсорний інтерфейс.
Iso-Fixu [5]	Дозволяє управляти запасами ліків в реальному часі, підходить для зберігання ліків з різними типами упаковки, може бути адаптований для різного типу приміщення.
Consis [6]	Зберігає від 1000 до 1500 різних типів ліків при загальній кількості ліків від 8000 до 25000 упаковок, має високу швидкість маніпулятора, що дозволяє видавати ліки всього за 7 секунд, може інтегруватися з будь-яким аптечним програмним продуктом.
BD Rowa [7]	Зберігає від 500 до 74500 різних типів ліків, має два маніпулятори, що дозволяє швидше працювати з великим об'ємом товару, має модуль очистки, який допомагає підтримувати полиці з ліками у чистоті,
PharmBot [8]	Висока швидкість маніпулятора дозволяє видавати потрібні ліки за 8 секунд, висока надійність дозволяє уникати програмних збоїв, дозволяє вести облік проданих товарів протягом всього дня.

Як можна побачити з табл. 1.1, автомати видачі ліків можуть відрізнятися за формою, розміром, загальною кількістю упаковок з ліками та кількістю маніпуляторів.

Але всі вони мають недолік в тому, що маршрут проходження маніпулятора не оптимізований, і в деяких випадках автомат взагалі може за один раз доставляти лише одну упаковку ліків. Для того, щоб усунути цей недолік було вирішено застосувати методи, які дозволяють оптимізувати шлях проходження маніпулятора автомата видачі ліків, таким чином зменшуючи загальний час обслуговування клієнта.

1.3 Огляд існуючих методів

У випускній кваліфікаційній роботі бакалавра розглядаються різні методи вирішення задачі комівояжера. Задача комівояжера відноситься до задач комбінаторної оптимізації. Вона полягає в тому, щоб знайти найвигідніший маршрут, який проходить через декілька міст, при цьому треба відвідати кожне місто тільки один раз. Існує декілька методів вирішення задачі комівояжера. Всі вони поділяються на точні та евристичні. До точних відносяться такі методи як повний перебір, метод гілок та меж, метод імітації відпалу та інші. До евристичних відносять метод мурашиних колоній, штучні нейронні мережі, генетичний алгоритм, жадібний алгоритм та інші.

1.3.1 Метод повного перебору

Метод повного перебору полягає в тому, щоб вичерпати всі можливі варіанти на шляху до рішення задачі. Цей метод має перевагу в тому, що він є точним і простим у реалізації. Також цей метод є універсальним, тобто вважається, що цим методом можна вирішити будь-яку задачу з класу NP (non-deterministic polynomial). Але при великій кількості варіантів також зростає і час виконання задачі.

Існують також різні модифікації та удосконалення методу повного перебору. Наприклад метод гілок і меж, який використовує дерево рішень і виключає ті варіанти вирішення задачі, при яких неможливо отримати оптимальне рішення, тим самим зменшуючи загальний час розв'язання задачі.

Також існує пошук з поверненням, який полягає в тому, що виконується деяке часткове рішення, яке поступово розширюється. Якщо подальше розширення неможливе, то відбувається повернення до попереднього рішення і пошук продовжується. Цей метод також може зменшити загальний час розв'язання задачі через відкидання невідповідних варіантів.

Але навіть при більш досконалих варіантах методу повного перебору в нього залишається серйозний недолік в тому, що при збільшенні кількості варіантів вирішення задачі експоненційно зростає загальний час виконання. Наприклад, якщо застосовувати метод повного перебору для вирішення задачі комівояжера при загальній кількості міст 66 або більше, то це буде неможливо зробити навіть за допомогою найпотужнішого комп'ютера менше ніж за декілька мільярдів років.

Тому для вирішення задач з великою кількістю варіантів доцільно використовувати менш точні та більш складні евристичні методи.

1.3.2 Метод мурашиних колоній

Принцип методу мурашиних колоній полягає в моделюванні реальної поведінки мурах в природі. Початково мурахи блукають в просторі довільним чином, і по знаходженні їжі повертаються до своєї колонії, залишаючи по собі феромонний слід. Після цього якщо інші мурахи знаходять такий шлях, вони схильні до припинення своїх блукань, тобто вони починають слідувати позначеним шляхом, посилюючи його під час повернення у разі знайдення їжі. Проте, з часом, феромонові сліди випаровуються, одже привабливість шляхів зменшується. Тобто очевидно, чим більше часу потрібно мурасі, щоб подолати дорогу, тим більше часу мають феромонові сліди, щоб випаруватись.

Натомість, коротка дорога проходиться швидше (відповідно і частіше), отже щільність феромонів стає більшою на короткому шляху. Випаровування феромонів також надає перевагу уникнення локально найкращих шляхів. Якби випаровування не відбувалось взагалі, шляхи обрані першим мурахою ставали б вкрай привабливими для наступних, тобто розвідка можливих шляхів була б обмежена. Таким чином, коли мураха знаходить вдалий (тобто коротший з наявних) шлях з колонії до джерела їжі, інші мурахи більш ймовірно слідуватимуть йому, і такий позитивний зворотний зв'язок в підсумку призведе до обрання цього кращого шляху всіма мурахами. Отже, ідея мурашиного алгоритму полягає в наслідуванні поведінки колонії мурах, що прогулюються графом, який представляє проблему для розв'язання [9].

Отже, для розв'язання задачі комівояжера методом мурашиних колоній треба створити мурах і помістити їх в одну або декілька точок на графі, а також задати початковий рівень феромону. Після цього мураха повинна відвідати всі вузли графу, при цьому треба уникати зациклення. Після кожної подорожі потрібно підрахувати довжину шляху та оновити рівень феромону. Кожна наступна мураха буде включати таке ребро графа в свій маршрут, на якому кількість феромона більше, і з часом буде знайдений найбільш оптимальний маршрут. Але також треба змодельовати механізм випаровування, для того, щоб уникати не оптимальних шляхів.

Метод мурашиних колоній можна використовувати для задач великої розмірності, при цьому він знаходить рішення, які близькі до оптимальних за більш короткий час ніж точні методи.

1.3.3 Генетичний алгоритм

Генетичний алгоритм (ГА) являє собою евристичний алгоритм оптимізації, в основу якого покладені біологічні принципи природного відбору і мінливості. Процес роботи алгоритму представляє собою послідовну зміну поколінь, що складаються з фіксованого числа особин – точок простору рішень,

причому особини з великим значенням цільової функції (більш пристосовані) отримують більше нащадків в кожному наступному поколінні. Крім того, при формуванні наступного покоління частина нащадків повністю ідентична батькам, а частина змінюється деяким випадковим чином в результаті мутації і кросинговеру (схрещування) [10].

Генетичний алгоритм використовується для того, щоб вирішувати задачі оптимізації за допомогою різних комбінацій та варіацій певних параметрів, використовуючі механізми, які аналогічні до природнього відбору. Особливістю цього алгоритму є наявність оператора схрещування, за допомогою якого виконується операція рекомбінації рішень, аналогічно до схрещування в природі.

Тобто при роботі алгоритму випадковим чином створюються точки простору рішень, покоління яких послідовно змінюються. Це відбувається таким чином, що більш пристосовані особини в популяції отримують більшу кількість нащадків в наступному поколінні. При цьому важливу роль відіграє функція допасованості, за допомогою якої обираються більш пристосовані особини. Після того, як більш пристосовані особини були обрані, до них застосовуються мутації та схрещування, для того щоб отримати нові рішення. Потім найкращі рішення переходять в наступне покоління. Цей процес буде тривати до тих пір, доки не буде знайдене оптимальне рішення, не вичерпається кількість поколінь, або не вичерпається час, який був виділений на еволюцію.

Класичний варіант генетичного алгоритму складається з 12 кроків.

Крок 1. Встановити лічильник ітерацій (часу): $t = 0$. Виконати ініціалізацію (initialization) початкової популяції особин: $P_t = \{H_1, H_2, \dots, H_N\}$.

Крок 2. Оцінити особини поточної популяції (evaluating) шляхом обчислення їх фітнес-функції $f(H_j)$, $j = 1, 2, \dots, N$.

Крок 3. Перевірити умови закінчення пошуку (termination criteria). Як такі умови можуть бути використані: досягнення максимально допустимого часу функціонування методу, числа ітерацій, значення функції пристосованості

і т. ін. Якщо критерії закінчення пошуку задовільнено, тоді виконати перехід до кроку 12.

Крок 4. Збільшити лічильник ітерацій (часу): $t = t + 1$.

Крок 5. Вибрати частину популяції (батьківські особини) для схрещування (selection of parents) P' .

Крок 6. Сформувати батьківські пари (mating) з особин, що відібрані на попередньому кроці.

Крок 7. Схрестити (crossover) вибрані батьківські особини.

Крок 8. Застосувати оператор мутації (mutation) до особин P' .

Крок 9. Обчислити нову функцію пристосованості $f(H_j)$ особин, отриманих в результаті схрещування та мутації.

Крок 10. Сформувати нове покоління шляхом вибору особин, що вижили, виходячи з рівня їх пристосованості (replacing, selection of survivors).

Крок 11. Перейти до кроку 3.

Крок 12. Зупинення [11].

Генетичний алгоритм працює ефективніше за традиційні методи, добре підходить для задач з великим простором пошуку, при цьому зі зростанням часу покращує рішення задачі. Але за допомогою нього можна вирішувати не всі типи задач, і в деяких випадках цей алгоритм може потребувати велику кількість ресурсів та часу.

1.3.4 Штучна нейронна мережа

Штучна нейронна мережа (ШНМ) – математична модель, а також її програмне або апаратне втілення, побудована на принципі організації та функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, які відбуваються в мозку, та при намаганні змодельовати ці процеси. Першою такою спробою були нейронні мережі У. Маккалока та У. Піттса. Після розробки алгоритмів

навчання отримувані моделі стали використовуватися в практичних цілях: в задачах прогнозування, для розпізнавання образів, в задачах керування [12].

ШНМ являють собою систему з'єднаних між собою простих обробників (штучних нейронів), які взаємодіють. Кожен обробник подібної мережі має справу лише з сигналами, які він періодично отримує, і сигналами, які він періодично надсилає іншим обробникам. І тим не менш, будучи з'єднаними в достатньо велику мережу з керованою взаємодією, такі локально прості обробники разом здатні виконувати доволі складні завдання [12].

Штучний нейрон – це простий обробник, який має декілька входів (синапси) та один вихід (аксон). Основна функція ШН – змінювати стан вихідного сигналу в залежності від сигналу на вході. Математично, штучний нейрон зазвичай представляють, як деяку нелінійну функцію від єдиного аргументу – лінійної комбінації всіх вхідних сигналів. Цю функцію називають функцією активації або функцією спрацьовування, передавальною функцією [12].

Функціонування нейрона можна описати формулою (1.1):

$$f(x) = \begin{cases} 1, & \sum_{i=1}^N w_i u_i \geq v; \\ 0, & \sum_{i=1}^N w_i u_i < v, \end{cases} \quad (1.1)$$

де u – вихідний сигнал нейрона;

$w_1, w_2, \dots, w_N$ – синаптичні вагові коефіцієнти;

$u_1, u_2, \dots, u_N$ – вхідні сигнали ШН;

v – порогове значення [12].

Модель (1.1) може буде представлена у вигляді (1.2):

$$y = f\left(\sum_{i=0}^N w_i u_i\right), \quad (1.2)$$

де $w_0 = v$, $u_0 = 1$ [12].

Для (1.2) виконується умова (1.3)

$$f(x) = \begin{cases} 1, & x \geq 0; \\ 0, & x < 0. \end{cases} \quad (1.3)$$

У якості функції активації f може прийматися не тільки одинична функція (1.3), а й інші порогові функції вигляду (1.4) та (1.5):

$$f(x) = \begin{cases} 1, & x \geq 0; \\ -1, & x < 0, \end{cases} \quad (1.4)$$

$$f(x) = \begin{cases} 1, & x \geq 0; \\ -1, & x < 0; \\ x, & |x| \leq 1. \end{cases} \quad (1.5)$$

В наш час частіше використовують сигмоїдальну функцію (1.6):

$$f(x) = \frac{1}{1 + e^{-\beta x}} > 0, \quad (1.6)$$

де β – коефіцієнт куту нахилу функції [12].

Можна побачити, що при $\beta \rightarrow \infty$ характеристика (1.6) переходить у порогову уніполярну функцію (1.4). У якості альтернативи використовують функцію гіперболічного тангенсу (1.7):

$$f(x) = \tanh\left(\frac{\alpha x}{2}\right) = \frac{1 - e^{\alpha x}}{1 + e^{\alpha x}}, \quad (1.7)$$

де α – коефіцієнт куту нахилу функції [13].

В цьому разі характеристика (1.7) прямує до порогової біполярної функції (1.7) при $\alpha \rightarrow \infty$.

Штучний нейрон є базовим блоком для побудови більш крупної сутності ШНС – шару.

Шар ШНМ – це сукупність ШН, поєднана в одну логічну сутність.

Переважає більшість ШНМ має в своєму складі наступні шари:

- вхідний шар;
- вихідний шар;
- один або кілька прихованих шарів.

Виділяють два основні способи навчання ШНМ:

- навчання з учителем;
- навчання без учителя [13].

При навчанні з вчителем вихідний сигнал ШНМ в процесі тренування контролюється сторонньою системою, і відповідно до рішень цієї системи коригується матриця ваг зв'язків.

При навчанні без вчителя використовуються алгоритми корекції похибки (напр., градієнтний спуск або «відпалювання»).

Архітектура ШНМ визначає кількість шарів у мережі, їх топологію, призначення та методи навчання. Існують такі базові архітектури:

- повнозв'язні мережі прямого розповсюдження;
- згорткові нейронні мережі;
- рекурентні нейронні мережі.

Кожна з цих архітектур підходить для вирішення відповідного класу завдань, та має значну кількість похідних архітектур, які реалізують специфічні для їх класу завдання [13].

Також в широкому спектрі сучасних завдань використовуються т.з. ансамблі ШНМ. Ансамбль ШНМ – це особливий тип архітектури, коли задачі вирішує нейронна мережа, яка, в свою чергу, складається з нейронних мереж різних архітектур [13].

До головних переваг нейромереж можна віднести такі:

- можливість використання при моделюванні не тільки кількісної, чіткої експериментальної інформації про закономірності, як в економетричних моделях, а й нечіткої, якісної, яка подається експертами (спеціалістами) на природній мові;
- можливість моделювання за відносно малих (порівняно з економетричними підходами) експериментальних вибірках;
- можливість відтворювання складних нелінійних залежностей за великої кількості вхідних параметрів;
- здатність моделей до навчання.

Незважаючи на широкий спектр можливостей ШНМ, вирішення завдань з їх допомогою супроводжує ряд недоліків:

- більшість підходів для проєктування ШНМ є наближеними і часто не призводять до однозначних рішень;
- для побудови моделі об'єкта на основі ШНМ слід дотримуватися багатоциклової настройки внутрішніх елементів і зв'язків між ними;
- проблеми, що виникають при підготовці навчальної вибірки, пов'язані з труднощами знаходження достатньої кількості навчальних прикладів;
- навчання мережі в деяких випадків призводить до тупикових ситуацій;
- часові витрати на виконання процедури навчання часто не дозволяють застосовувати ШНМ в системах реального часу [14].

1.3.5 Жадібний алгоритм

Жадібний алгоритм є одним з евристичних алгоритмів і працює таким чином, що на кожному етапі виконання певної задачі знаходить найбільш

оптимальне рішення, але при цьому він не враховує можливі наслідки, припускаючи, що кінцевий розв'язок також буде оптимальним.

Перевага цього алгоритму полягає в тому, що його використання є дуже ефективним через те, що він дозволяє знаходити оптимальне рішення за короткий час навіть для задач великої розмірності. Але також він має недолік в тому, що його можна використовувати не для всіх типів задач. Він добре підходить тільки для тих задач, які мають оптимальну підструктуру, тобто виконання кожного наступного етапу повинно вести до знаходження загального оптимального рішення.

1.4 Порівняльний аналіз існуючих методів вирішення завдання

В ході дослідження та аналізу існуючих методів вирішення задачі комівояжера для автомата видачі ліків було зроблено порівняльний аналіз (табл. 1.2).

Таблиця 1.2 – Порівняльна характеристика існуючих методів для вирішення задачі комівояжера для автомата видачі ліків

Можливості	Алгоритм повного перебору	Метод мурашиних колоній	Генетичний алгоритм	Штучна нейронна мережа	Метод гілок та границь	Жадібний алгоритм
Простота реалізації	+	—	+	—	—	+
Залежність витрат часу на рішення від розмірності задачі	+	+	—	+	+	—
Простота кодування інформації	+	—	+	—	—	+

Продовження таблиці 1.2

Можливості	Алгоритм повного перебору	Метод мурашиних колоній	Генетичний алгоритм	Штучна нейронна мережа	Метод гілок та границь	Жадібний алгоритм
Точність рішення	+	+	—	+	+	—
Адаптивність до різних варіантів задач	+	—	+	+	+	—

Як видно з таблиці 1.2, точні методи мають високу залежність витрат часу від розмірності задачі, але при цьому вони видають максимально точний результат. Тобто можна зробити висновок, що при малій розмірності задачі доречно використовувати саме їх, але якщо розмірність буде занадто великою, то краще використовувати інші методи, які будуть видавати лише наближений результат, але за прийнятний проміжок часу. Через те, що для виконання поставленого завдання необхідно обробляти велику кількість інформації, використання точних алгоритмів є неприйнятним.

З табл. 1.2 видно, що жадібний алгоритм має такі переваги як простота реалізації та кодування, а також при його використанні витрати часу на рішення не залежать від розмірності задачі. Отже, для того, щоб виконати поставлене завдання було вирішено використати жадібний алгоритм.

1.5 Постановка завдання до роботи

В даній роботі досліджується проблема знаходження оптимального маршруту переміщення маніпулятора автомата видачі ліків.

Ліки, які обираються для замовлення розташовуються на точках з координатами $\{x_i, y_i\}$, $i = \overline{1, m}$ на площині в декартовій системі координат. Множина цих ліків становить множину $I = \{1, 2, \dots, m\}$. Загальна кількість ліків

також розташовується на площині в декартовій системі координат $\{x_j, y_j\}$, $j = \overline{1, n}$. Загальна множина ліків становить множину $J = \{1, 2, \dots, n\}$. При цьому число n більше за число m , тому що у маніпулятора автомата видачі ліків є певні обмеження, такі як максимально допустима вага упаковок ліків та їхній об'єм. Тому кількість ліків, яку може переносити маніпулятор автомату видачі ліків набагато менша за загальну кількість ліків в автоматі [15].

Також у маніпулятора автомату видачі ліків є швидкість, тобто знаючи загальну довжину шляху можна визначити час, який буде затрачено на проходження маніпулятором маршруту. Загальний час t розраховується за формулою (1.8):

$$t = \frac{S}{V}, \quad (1.8)$$

де S – загальна довжина шляху;

V – швидкість маніпулятора [16].

Метою роботи є створення програмної системи для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера

Для виконання поставленої мети потрібно виконати наступні завдання:

- створити метод для додавання інформації про замовлення в БД;
- створити метод для авторизації користувачів;
- створити метод для визначення загальної кількості ліків в замовленні;
- створити метод для побудови матриці відстаней для полиць з ліками, які будуть включені в маршрут проходження маніпулятора автомату видачі ліків;
- створити метод для знаходження оптимального маршруту, який проходить маніпулятор;
- створити метод для розрахунку стандартного маршруту проходження;

- створити метод для розрахунку загального часу проходження оптимального маршруту;
- виконати розрахунок загального часу проходження стандартного маршруту.

1.6 Висновок

В даному розділі було виконано дослідження предметної області. Було розглянуто та визначено основні поняття, які пов'язані з обраною предметною областю.

Також були проаналізовані та досліджені існуючі методи, кожен з яких можливо застосовувати для вирішення поставленої задачі, було визначено їх переваги та недоліки.

Був проведений порівняльний аналіз досліджених методів, які можуть бути використані для вирішення задачі комівояжера для автомата видачі ліків.

Було сформульовано і визначено завдання до роботи.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного продукту

Для розробки ПП необхідно обрати найбільш оптимальний інструментарій розробки, який буде відповідати всім вимогам поставленого завдання.

Основна задача ПП – це програмна реалізація пошуку оптимального маршруту для автомату видачі ліків за допомогою вирішення задачі комівояжера.

ПП повинен містити в собі такі модулі:

- зберігання даних;
- обчислювальний;
- вводу-виводу даних;
- авторизації.

Для проєктування програмного продукту треба обрати мову програмування, середовище розробки та СУБД.

2.1.1 Вибір мови програмування

Для того, щоб вирішити поставлену задачу, що полягає в реалізації методів для пошуку оптимального маршруту для автомату видачі ліків за допомогою вирішення задачі комівояжера треба дослідити декілька сучасних мов програмування та зробити порівняльну характеристику, на основі якої буде зроблений вибір на користь однієї з мов програмування, яка буде використана для розробки програмного продукту. Нижче буде наведений опис мов програмування та визначено їх відповідні переваги та недоліки.

R – це мова програмування, яка була розроблена спеціально для того, щоб виконувати статистичні обчислення та обробку даних. Також за

допомогою мови R можна працювати з графікою і відображати дані у вигляді різних діаграм та графіків.

R є інтерпретованою мовою програмування, тобто для роботи з нею використовується командний інтерпретатор. За синтаксисом мова програмування R схожа на функціональні та скриптові мови програмування, підтримує об'єктно-орієнтоване програмування, також є об'єктною мовою програмування. Мова програмування R поширюється безкоштовно та має відкритий вихідний код.

Переваги R:

- має додаткові бібліотеки, що розширяють базові можливості, та дозволяють розробляти статистичні програми для будь-якого типу задач;
- підтримує завантаження даних та таблиць у різних форматах;
- має багато інструментів для зручної візуалізації даних у вигляді різних таблиць, діаграм та графіків.

Недоліки R:

- низька швидкість роботи порівняно з іншими мовами програмування ;
- через специфічність мови R та несхожість її з іншими популярними мовами програмування виникає складність в її вивченні [17].

C++ (Сі-плюс-плюс) є мовою програмування високого рівня, підтримує процедурну, об'єктно-орієнтовану та узагальнену парадигму програмування.

Переваги C++:

- висока швидкість роботи;
- можливість запускати програми у різних операційних системах та на різних платформах;
- мультипарадигмальність;
- можливість керування пам'ятю, що дозволяє оптимізувати програмні продукти.

Недоліки C++:

- відсутність системи збору сміття;

- наявність покажчиків, які потребують велику кількість пам'яті та при неправильному використанні призводять до різних сбоїв в програмних продуктах;

- через доступність дружніх функцій, глобальних змінних та покажчиків можуть виникати проблеми безпеки та захищеності програмних продуктів [18].

C# (Сі-шарп) є об'єктно-орієнтованою мовою програмування. Синтаксис цієї мови дуже схожий на синтаксис мов програмування C++ та Java. Підтримує перевантаження операторів, ітератори, анонімні функції, має статичну типізацію, підтримує об'єктно-орієнтоване програмування та має систему збору сміття.

Переваги C#:

- підтримує автоматичний збір сміття;
- має високу надійність та захищеність;
- є повністю об'єктно-орієнтованою мовою програмування;
- має обширну бібліотеку класів, що спрощує реалізацію багатьох функцій.

Недоліки C#:

- не є гнучкою мовою програмування;
- необхідність компілювати код кожного разу при будь-яких змінах в ньому;
- C# базується на платформі Microsoft .NET, тому програмні продукти написані цією мовою будуть коректно працювати тільки в операційній системі Windows [19].

Python є високорівневою мовою програмування, має високу якість читабельності коду, підтримує різні додаткові пакети та модулі, має мінімалістичний синтаксис ядра.

Переваги Python:

- простий для розуміння та вивчення;
- має велику кількість бібліотек, що розширюють базовий функціонал;
- має високу продуктивність.

Недоліки Python:

- низька швидкість роботи через построкове виконання коду;
- погано взаємодіє з БД, порівняно з іншими мовами програмування;
- потребує великого об'єму пам'яті [20].

Java є об'єктно-орієнтованою мовою програмування, працює за допомогою віртуальної машини. Є платформо-незалежною мовою програмування, синтаксис схожий на C та C++.

Переваги Java:

- підтримка багатопоточності;
- незалежність від платформи;
- висока стабільність.

Недоліки Java:

- підтримує тільки одну парадигму програмування;
- низька швидкість роботи;
- не підходить для створення складного користувацького інтерфейсу [21].

Під час дослідження різних мов програмування було створено порівняльну характеристику, яка наведена у табл. 2.1.

Таблиця 2.1 – Порівняльна характеристика мов програмування

Критерії порівняння	R	C++	Python	Java	C#
Мова створювалася спеціально для аналізу даних	+	–	–	–	–
Ліцензія	GNU GPL	–	Python Software Foundation License	GNU GPL	.NET Core CLR: MIT / X11
Незалежність від платформи	+	+	+	+	–
Можливість обробляти велику кількість даних	+	+	+	+	+

Продовження таблиці 2.1

Критерії порівняння	R	C++	Python	Java	C#
Потужні інструменти для візуалізації даних	+	—	+	—	—
Простота вивчення та розробки	—	+	+	+	+
Висока швидкість роботи	—	+	—	—	+
Наявність додаткових бібліотек та пакетів	+	+	+	+	+
Можливість паралельного програмування	+	+	+	+	+
Наявність взаємодії з іншими мовами програмування	+	+	+	+	+

Як видно з табл. 2.1. мови програмування C++ та C# мають високу швидкість роботи та відповідають усім вимогам до поставленої задачі. Але мова C# більш сучасна ніж C++, має більш високу надійність, потребує менших об'ємів пам'яті та є повністю об'єктно-орієнтованою. Тому для розробки програмного продукту було обрано мову програмування C#.

2.1.2 Вибір середовища розробки

Для ефективної розробки програмного продукту треба обрати середовище розробки, яке буде підходити під обрану мову програмування.

Для мови програмування C# підходить декілька середовищ розробки. Для того, щоб обрати одне з них було вирішено дослідити їх та виділити основні переваги та недоліки.

Microsoft Visual Studio є середовищем розробки від фірми Microsoft. Дає змогу розробляти консольні та графічні програмні продукти, має вбудований відладчик коду, підтримує велику кількість додаткових бібліотек для розширення базового функціоналу, в тому числі для візуального проектування.

Переваги Microsoft Visual Studio:

- можливість безкоштовного використання;
- є офіційним середовищем розробки для мови C#;
- підтримка платформи .NET.

Недоліки Microsoft Visual Studio:

- складність освоєння через неочевидне розташування функцій в меню;
- при переході на іншу версію можуть виникати помилки та збої;
- необхідність зберігати зміни в коді вручну [17].

Project Rider є кросплатформенним середовищем розробки від компанії JetBrains. Підтримує повний цикл розробки програмного продукту, працює з операційними системами Windows, Linux та MacOS.

Переваги Project Rider:

- має багато опцій для швидкого написання коду ;
- підтримує запуск декількох програм одночасно;
- підтримує контроль версій.

Недоліки Project Rider:

- реліз середовища розробки відбувся лише в 2017 році, тому не всі функції розроблені до кінця, а також можуть виникати певні помилки та збої;
- середовище розробки не є безкоштовним [22].

Eclipse є інтегрованим середовищем розробки від компанії Eclipse Foundation. Воно орієнтоване для розробки програмних продуктів на мові програмування Java, але також підтримує мову програмування C#. Використовується для розробки модульних програмних продуктів.

Переваги Eclipse:

- компілятор та відладчик працюють ефективніше ніж в інших середовищах розробки;
- функціонал дозволяє кастомізувати середовище розробки для зручного використання;
- безкоштовність та відкритий вихідний код.

Недоліки Eclipse:

- додаткові плагіни не є офіційними, тому при їх використанні немає гарантій надійності;
- використовує велику кількість апаратних ресурсів та займає багато місця на диску [23].

Під час дослідження різних середовищ розробки було створено порівняльну характеристику, яка представлена в табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика середовищ розробки

Критерії порівняння	Microsoft Visual Studio	Project Rider	Eclipse
Ліцензія	freemium	proprietary	Eclipse Public License
Можливість використання в різних системах	+	+	+
Підтримка контролю версій	+	+	+
Наявність додаткових модулів	+	+	+
Простота в освоєнні	–	+	–
Можливість безкоштовного використання	+	–	+
Надійність	+	–	–
Зручний інтерфейс	–	+	+
Використання великої кількості апаратних ресурсів	–	–	+

В табл. 2.2 можна побачити, що середовище розробки Microsoft Visual Studio має більш високу надійність порівняно з іншими середовищами, не потребує великої кількості апаратних ресурсів та має безкоштовну версію з усім необхідним функціоналом. Також Microsoft Visual Studio є офіційним середовищем розробки для мови програмування C#. Саме тому для розробки програмного продукту було обране середовище розробки Microsoft Visual Studio.

2.2 Структурна схема програмного продукту

ПП повинен містити в собі такі модулі:

- зберігання даних;
- обчислювальний;
- вводу-виводу даних;
- авторизації.

В якості вхідних даних ПП повинен використовувати інформацію про ліки, для якої треба обрати відповідне сховище.

Виділяють декілька типів сховищ:

- текстовий файл;
- бінарний файл даних;
- нереляційна БД;
- реляційна БД.

Обране сховище для зберігання даних повинно бути надійним, надавати швидкий доступ до даних, мати можливість для зберігання великої кількості даних та їх зручної модифікації. Під ці параметри найкраще підходить реляційна БД.

Переваги реляційних БД:

- складається з таблиць, які зручні для використання та розуміння;
- відповідає правилам, які ґрунтуються на математичному апараті;
- дані, які зберігаються в реляційній БД не залежать від змін в коді програми.

Для зберігання даних програмного продукту було вирішено використати СУБД реляційного типу Microsoft Access. Ця СУБД відповідає всім вимогам до розробки, також вона є продуктом компанії Microsoft, тому її можна зручно підключити до середовища розробки Microsoft Visual Studio для подальшого використання при розробці програмного продукту.

При проектуванні БД треба виконати розробку моделі для відповідної предметної області. Ця модель включає в себе об'єкти БД та зв'язки між цими

об'єктами. Модель може бути представлена у вигляді ER-діаграми. ER-діаграма являє собою певну блок-схему, в яку входять сутності та їх властивості, а також взаємодія між ними.

Під час проектування БД було вирішено визначити сутності, які входять в БД та їх атрибути такі як назва атрибута, тип, обов'язковість заповнення та розмір. Результати представлені в табл. 2.3.

Таблиця 2.3 – Сутності та їх атрибути

Сутність	Атрибут	Тип	Розмір	Обов'язкове	Опис сутності
Ліки	Код_ліків	Лічильник	Довге ціле	Так	Містить інформацію про ліки та їх параметри
	Назва_ліків	Текстовий	255	Так	
	Вага_упаковки	Числовий	Довге ціле	Так	
	Розмір_упаковки	Числовий	Довге ціле	Так	
Фармацевти	Код_фармацевта	Числовий	Довге ціле	Так	Містить інформацію про фармацевтів
	ПІБ_фармацевта	Текстовий	255	Так	
	Пароль_фармацевта	Текстовий	255	Так	
Замовлення	Код_замовлення	Лічильник	Довге ціле	Так	Містить інформацію про замовлення
	Код_фармацевта	Числовий	Довге ціле	Так	
	Дата_замовлення	Числовий	Довге ціле	Так	

Як можна побачити в табл. 2.3 визначено декілька сутностей, а саме сутність Ліки, яка містить в собі назву ліків, а також вагу та розмір упаковки, сутність Фармацевти, яка містить ПІБ фармацевта та його особистий пароль для авторизації, сутність Замовлення, яка містить інформацію про ПІБ фармацевта, який робив замовлення та дату замовлення.

На рис. 2.1 представлена ER-діаграма БД.

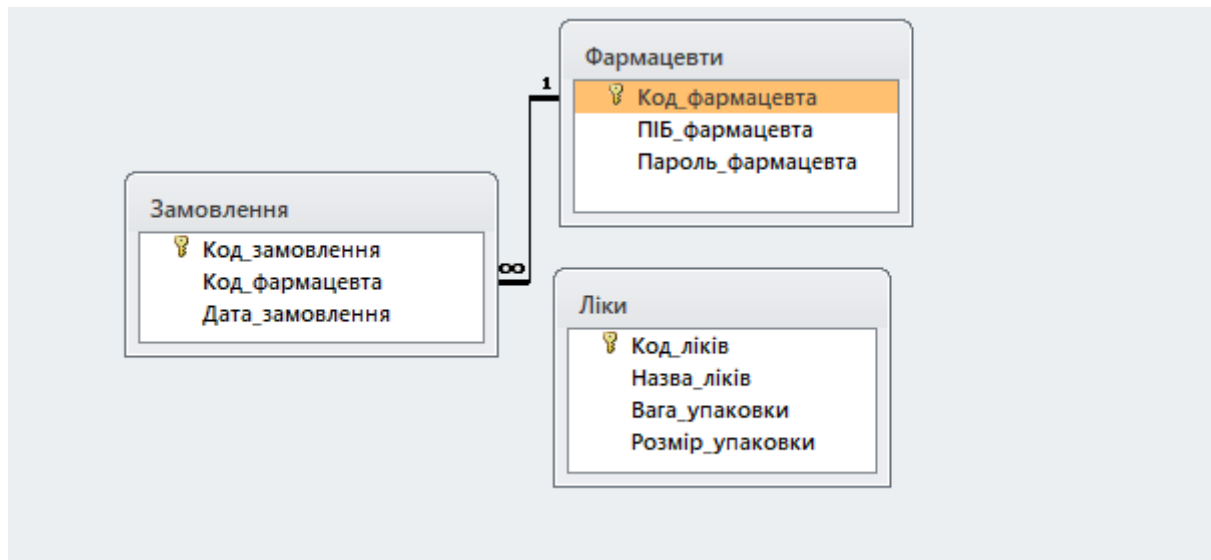


Рисунок 2.1 – ER-діаграма БД

Основним модулем програмного продукту є обчислювальний модуль. Його завдання полягає в тому, щоб визначити оптимальний маршрут проходження маніпулятора автомата видачі ліків. Цей модуль містить в собі такі функції:

- функція, яка визначає координати ліків, через які буде проходити маршрут;
- функція, розрахунку відстаней між ліками;
- функція розрахунку часу проходження стандартного маршруту;
- функція розрахунку часу проходження оптимального маршруту;
- функція обчислення довжини стандартного маршруту;
- функція обчислення довжини оптимального маршруту;
- функція розрахунку оптимального маршруту;
- функція пошуку оптимального маршруту.

Для реалізації обчислювального модуля була обрана мова програмування C# та середовище розробки Microsoft Visual Studio.

Модуль вводу-виводу даних дозволяє отримувати дані про ліки, замовлення та фармацевтів зі сховища та виводити їх на екран, а також

передавати дані обчислювальному модулю для подальшої їх обробки. Функції модуля вводу-виводу даних:

- функція додавання інформації про замовлення в БД;
- функція відображення інформації про замовлення;
- функція відображення інформації про ліки;
- функція, яка визначає назви ліків, між якими буде прокладено

маршрут;

- функція пошуку та відображення потрібного замовлення за датою;
- функція пошуку та відображення потрібного фармацевта за ПІБ;
- функція відображення матриці відстаней між ліками;
- функція відображення інформації про стандартний маршрут;
- функція відображення інформації про оптимальний маршрут.

Для реалізації модуля вводу-виводу даних була обрана мова програмування C# та середовище розробки Microsoft Visual Studio.

Важливим модулем також є модуль авторизації, який захищає ПП від несанкціонованого доступу за допомогою логіну та паролю. Також модуль авторизації дозволяє розмежувати доступ до програми, для того щоб вести облік даних різних користувачів.

Для реалізації модуля авторизації була обрана мова програмування C# та середовище розробки Microsoft Visual Studio.

Наступною важливою складовою програмного продукту є графічний інтерфейс, за допомогою якого користувач взаємодіє з програмою.

Основною функцією графічного інтерфейсу є можливість переглядати інформацію про ліки, фармацевтів та замовлення, додавати ліки в замовлення, а також переглядати дані про стандартний і оптимальний маршрути та час їхнього проходження.

Обране середовище розробки Microsoft Visual Studio дозволяє створити графічний інтерфейс для користувача за допомогою класів спеціальних класів Windows Forms, які є частиною .NET Framework.

Windows Forms має відкритий вихідний код та забезпечує доступ до елементів управління Windows, наприклад кнопки або текстові поля. Кожен елемент в Windows Forms є екземпляром головного класу Control, що дозволяє підтримувати функціональність конкретного елементу, наприклад розмір елемента, його колір, параметри тексту, а також різні події, пов'язані з елементом, наприклад при натисканні або переміщенні.

Всі елементи розміщуються на формі, яка виводиться на екран при запуску програми. Користувач взаємодіє з формою та елементами на ній, при цьому генеруються певні події, які потім обробляються кодом програмного продукту.

На рис 2.2 представлена структурна схема програмного продукту.

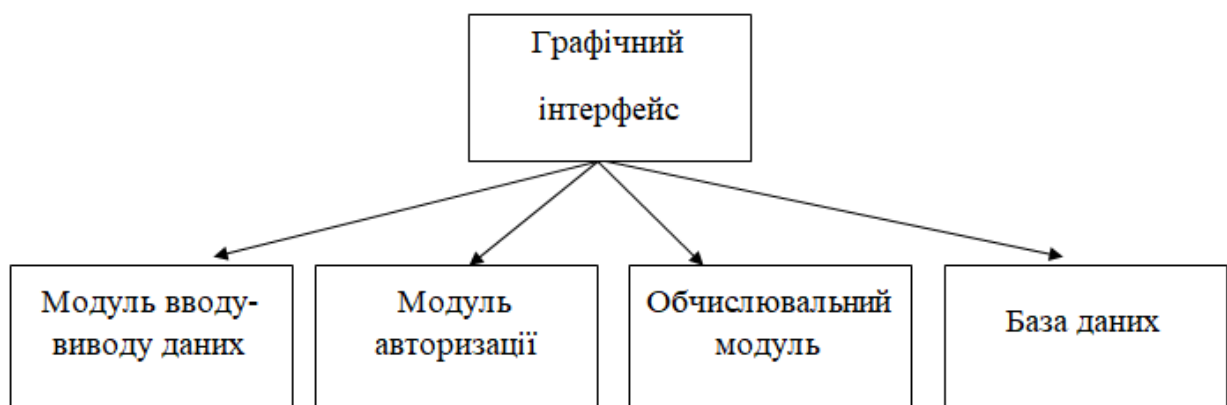


Рисунок 2.2 – Структурна схема програмного продукту

Як можна побачити на рис 2.2 ПП має графічний інтерфейс, який дозволяє візуалізувати вхідні та вихідні дані та надає користувачу зручний доступ до управління програмою. Графічний інтерфейс розробляється в середовищі розробки Microsoft Visual Studio на мові програмування C#.

Модуль вводу-виводу даних дозволяє надавати отримувати дані з БД та відображати їх за допомогою графічного інтерфейсу. Модуль авторизації дозволяє захистити ПП від несанкціонованого доступу, а також розмежувати доступ до програмного продукту.

Також ПП має обчислювальний модуль, який використовує вхідні дані для обчислення різних параметрів, таких як час проходження маршруту, довжина маршруту та довжина відстаней між точками маршруту. Модуль вводу-виводу даних, модуль авторизації та обчислювальний модуль також розробляється в середовищі розробки Microsoft Visual Studio та на мові програмування C#.

Вхідні дані такі як інформація про ліки, інформація про фармацевтів та інформація про замовлення зберігаються в реляційній БД, для цього використовується СУБД Microsoft Access.

2.3 Висновок

В даному розділі були розглянуті різні мови програмування та середовища розробки, які дозволяють розробляти програмні продукти. Було проведено їх аналіз, виділено основні переваги і недоліки та складено порівняльну характеристику. В ході цієї роботи було встановлено, що для програмного продукту найбільш ефективною мовою програмування є C#, тому що вона більш захищена і працює швидше за інші мови програмування. Також найбільш ефективним середовищем розробки було обрано середовище Microsoft Visual Studio, тому що воно є офіційним для мови програмування C#, не потребує великих обсягів апаратних ресурсів, а також дозволяє отримати безкоштовний доступ до всього необхідного функціоналу.

Також було розроблено структурну схему програмного продукту. Для зберігання вхідних даних було обрано найкращий тип сховища, а саме реляційну БД у СУБД Microsoft Access.

Інтерфейс програмного продукту було вирішено розробляти у середовищі розробки Microsoft Visual Studio за допомогою технології Windows Forms.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

В ході виконання роботи розглядається проблема пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків. Цю проблему можна вирішити, застосувавши один з існуючих алгоритмів розв'язання задачі комівояжера.

Завдання роботи полягає в створенні програмної системи для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

Для того, щоб виконати поставлене завдання треба створити метод пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків.

3.1 Метод пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків

Для пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків запропоновано створення метод із застосуванням алгоритму, який підходить для вирішення задачі комівояжера та відповідає заданій умові. Але перед тим як застосувати певний алгоритм для пошуку оптимального маршруту необхідно пройти декілька етапів.

На першому етапі необхідно створити метод визначення координат розташування полиць з ліками, які будуть складати маршрут проходження маніпулятора. Інформація про ліки зберігається в БД, яка підключена до програмного продукту.

Для того, щоб прив'язати ліки в БД до координат, треба створити двовимірний масив, в якому кількість стовпців та рядків буде дорівнювати кількості ліків по вертикалі та по горизонталі відповідно в автоматі видачі ліків. Після цього необхідно занести інформацію про назви ліків з БД у створений

двовимірний масив. Елементи в масиві будуть мати дві координати n та m . Координата n відповідає номеру стовпця в двовимірному масиві, а координата m відповідає номеру рядка в двовимірному масиві. При цьому координати елементів двовимірного масива будуть відповідати координатам розташування ліків у автоматі видачі ліків. А отже, після цього можна буде визначити координати розташування полиць з ліками, які будуть складати маршрут проходження маніпулятора автомату видачі ліків.

На другому етапі необхідно створити метод для побудови матриці відстаней, у якій будуть зберігатися та відображатися відстані між координатами розташування ліків. Матриця відстаней являє собою двовимірний масив, кількість стовпців та рядків у якому дорівнюють кількості ліків, які складають маршрут проходження, а координати елементів двовимірного масиву i та j відповідають координатам полиць з ліками, між якими буде розраховано відстань. Матриця відстаней є симетричною відносно до діагоналі, тобто елемент з координатами $i_n j_m$ буде дорівнювати елементу з координатами $i_m j_n$.

При цьому за умовою задачі комівояжера кожену полицю з ліками можна проходити лише один раз за маршрут. Для того, щоб виконати цю умову необхідно заповнити діагональ матриці відстаней нулями, і таким чином при проходженні маршруту буде уникнено зациклення.

Після побудови матриці відстаней розпочинається третій етап, в ході якого необхідно створити метод, який дозволить обирати ліки, які будуть складати маршрут проходження маніпулятор автомату видачі ліків. При цьому маніпулятор має певні обмеження за розміром упаковки ліків та вагою упаковки ліків. Максимальний загальний об'єм упаковок з ліками для маніпулятора складає 1500 см³. Максимальна загальна вага всіх упаковок з ліками складає 800 грамів. Інформація про вагу та об'єм упаковки з ліками зберігається в БД.

При додаванні ліків в замовлення треба передавати дані про вагу та об'єм упаковок з ліками до обчислювального модуля, який після додавання

кожної упаковки з ліками буде додавати ці параметри до тих пір, доки їх значення не досягнуть максимальних для маніпулятора. Після цього жодні упаковки з ліками вже не можна буде додати до маршруту проходження.

Під час додавання упаковок з ліками до маршруту проходження обчислювальний модуль повинен розрахувати відстані між їхніми відповідними координатами та записати цю інформацію в матрицю відстаней. Початкова точка маршруту завжди дорівнює нулю і відповідає за початкове місцезнаходження маніпулятора автомата видачі ліків. Після проходження всіх необхідних полиць з ліками маніпулятор повертається в початкову точку, для того, щоб фармацевт мав змогу забрати ці ліки і передати їх покупцеві. При цьому під час розрахунку кожної відстані треба враховувати розмір полиці з ліками, який становить 5 сантиметрів. Тобто після знаходження відстані між двома елементами в двовимірному масиві необхідно результат помножити на 5, для того, щоб отримати реальну відстань в автоматі між полицями з ліками. Розрахунок відстані між точками в масиві S обчислюється за формулою (3.1):

$$S = 5 \cdot (|i_2 - i_1| + |j_2 - j_1|), \quad (3.1)$$

де i_1 та j_1 – координати першої точки;

i_2 та j_2 – координати другої точки [24].

Під час розрахунку кожної відстані вона записується у відповідні координати в матрицю відстаней, для того, щоб обчислювальний модуль використав ці дані для пошуку оптимального маршрута за допомогою обраного метода вирішення задачі комівояжера.

Також в процесі розрахунку відстаней обчислювальний модуль розраховує стандартний маршрут проходження маніпулятора, який полягає в послідовному проходженні маніпулятором полиць з ліками. Цей маршрут не є оптимальним, але його потрібно розрахувати для того, щоб потім порівняти з оптимальним маршрутом, і зрозуміти наскільки ефективним є використання

саме оптимального маршруту. Стандартний маршрут проходження S_{st} розраховується за формулою (3.2):

$$S_{st} = \sum_{i=m}^n S_i, \quad (3.2)$$

де n – кількість ліків в замовленні;

i – індекс суми;

m – початкова точка проходження маршруту;

S_i – відстань між полицями з ліками [25].

Після заповнення матриці відстаней необхідно перейти до пошуку оптимального маршрута з використанням одного з алгоритмів. Точні алгоритми хоча і мають перевагу в тому, що знаходять найкраще рішення, але також в них є серйозний недолік в тому, що при збільшенні розмірності задачі експоненційно зростає час пошуку рішення. Тобто, якщо кількість упаковок з ліками в маршруті буде досягати шістнадцяти та більше, то використання точних алгоритмів буде не оптимальним через високий час пошуку рішення. Автомат видачі ліків розрахований на 10000 полиць з ліками, а маніпулятор автомата дозволяє переносити більше двадцяти різних ліків одночасно, за умови, що вага та об'єм їхньої упаковки потрапляє в діапазон максимальних параметрів маніпулятора.

Тому було вирішено обрати для пошуку оптимального маршрута евристичний алгоритм, а саме жадібний алгоритм. Цей алгоритм працює таким чином, що на кожному етапі знаходить найбільш оптимальне рішення, вважаючи, що кінцеве рішення також буде оптимальним.

Тобто, при пошуку оптимального маршрута алгоритм обирає відстань від початкової точки до найближчої полиці з ліками. Після цього пошук продовжується з цієї полиці з ліками до іншої найближчої, але при цьому ігноруються ті полиці, які вже було включено в маршрут. Також на кожному

етапі включення полиці в маршрут обчислювальний модуль розраховує відстань між полицями. Пошук буде продовжуватись до тих пір, доки всі задані полиці не будуть включені в маршрут. Після цього маршрут завершується поверненням з останньої включеної полиці в початкову точку, для того, щоб маніпулятор мав можливість повернутись для видачі ліків фармацевту.

Після побудови маршрута обчислювальний модуль розраховує загальну довжину оптимального маршруту проходження маніпулятора. Цей маршрут S_{min} розраховується за формулою (3.3):

$$S_{min} = \sum_{j=m}^p S_j, \quad (3.3)$$

де p – кількість мінімальних точок в маршруті;

j – індекс суми;

m – початкова точка проходження маршруту;

S_j – відстань між найближчими полицями з ліками в маршруті [25].

Після побудови стандартного та оптимального маршрутів також розраховується час їхнього проходження для подальшого порівняння. Час проходження стандартного маршруту t_{st} розраховується за формулою (3.4):

$$t_{st} = \frac{S_{st}}{V}, \quad (3.4)$$

де S_{st} – загальна довжина стандартного маршрута проходження маніпулятора;

V – швидкість маніпулятора [16].

Час проходження оптимального маршруту t_{min} розраховується за формулою (3.5):

$$t_{\min} = \frac{S_{\min}}{V}, \quad (3.5)$$

де $S_{\min}$ – загальна довжина оптимального маршруту проходження маніпулятора;
 V – швидкість маніпулятора [16].

Роботу метода пошуку оптимального маршруту можна продемонструвати на прикладі замовлення, в якому маніпулятору автомата видачі ліків треба якнайшвидше доставити 20 різних ліків до фармацевта.

Спочатку необхідно обрати потрібні ліки з БД, при цьому початкова точка буде нульовою, та визначити їх координати (табл. 3.1).

Таблиця 3.1 – Координати можливих ліків в замовленні

№	Назва ліків в таблиці	Координата i	Координата j
1	0	0	0
2	x286	14	6
3	x535	26	15
4	x836	41	16
5	x1174	58	14
6	x1296	64	16
7	x1667	83	7
8	x1816	90	16
9	x1920	96	0
10	x1922	96	2
11	x2048	102	8
12	x6964	348	4
13	x6968	348	8
14	x8452	422	12
15	x9624	481	4
16	x9914	495	14
17	x26	1	6
18	x978	48	18
19	x3736	186	16
20	x5763	288	3
21	x6628	331	8

Після того, як необхідні ліки було обрано, та визначено їхні відповідні координати, відстані між координатами обробляються обчислювальним модулем з використанням жадібного алгоритму. Також виконується розрахунок стандартного та пошук оптимального маршрутів проходження. Стандартний маршрут проходження, який полягає в проходженні маніпулятором через полиці з ліками в порядку додавання їх до замовлення представлений в табл. 3.2.

Таблиця 3.2 – Стандартний маршрут проходження

Назва ліків на полиці з ліками	Відстань між полицями з ліками
x286	100 см
x535	105 см
x836	80 см
x1174	95 см
x1296	40 см
x1667	140 см
x1816	80 см
x1920	110 см
x1922	10 см
x2048	60 см
x6964	1250 см
x6968	20 см
x8452	390 см
x9624	335 см
x9914	120 см
x26	2510 см
x978	295 см
x3736	700 см
x5763	575 см
x6628	240 см
0	1695 см

Оптимальний маршрут проходження, пошук якого здійснюється за допомогою методу пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків з використанням жадібного алгоритму представлений в табл. 3.3.

Таблиця 3.3 – Оптимальний маршрут проходження

Назва ліків на найближчій полиці з ліками	Відстань між найближчими полицями з ліками
x26	35 см
x286	65 см
x535	105 см
x836	80 см
x978	45 см
x1174	70 см
X1296	40 см
X1816	130 см
X1667	80 см
X1922	90 см
X1920	10 см
X2048	70 см
X3736	460 см
X5763	575 см
X6628	240 см
X6968	85 см
X6964	20 см
X8452	410 см
X9624	335 см
X9914	120 см
0	2545 см

Після розрахунку стандартного та пошуку оптимального маршрутів проходження розраховується загальна довжина та час їхнього проходження. Загальна довжина стандартного маршруту S_{st} розраховується за формулою 3.2 і дорівнює 8950 см. Час проходження стандартного маршруту t_{st} розраховується за формулою 3.4 і дорівнює 46,3 с.

Загальна довжина оптимального маршруту S_{min} розраховується за формулою 3.3 і дорівнює 5610 см. Час проходження оптимального маршруту t_{min} розраховується за формулою 3.5 і дорівнює 32,94 с.

При проходженні маніпулятором автомату для видачі ліків оптимальним маршрутом, який було знайдено з використанням жадібного алгоритму

загальний довжина проходження зменшується на 3340 см, а час зменшується на 13,42 с у порівнянні зі стандартним маршрутом.

Отже, було розроблено метод пошуку оптимального маршруту проходження для маніпулятора автомату видачі ліків. При тестуванні методу на прикладі було встановлено, що при проходженні маніпулятором саме оптимальним шляхом можна отримати значну економію часу завдяки зменшенню загальної довжини оптимального маршруту, у порівнянні зі стандартним.

Також важливо зазначити, що використання евристичного алгоритму для розробки метода, а саме жадібного алгоритму дозволило дуже швидко знайти оптимальний маршрут, тому що використання точних алгоритмів при такій умові вимагало б занадто великих обсягів часу та апаратних ресурсів, а тому застосування їх на практиці для такого завдання було б неможливим.

3.2 Метод забезпечення авторизації користувачів

Так як розроблюваний ПП призначений для використання в аптечних закладах, в яких може працювати декілька фармацевтів, а також інші співробітники, які не повинні отримати доступ до ПП, для нього дуже важливо створити метод забезпечення авторизації користувачів. Метод забезпечення авторизації користувачів дозволяє захистити ПП від несанкціонованого доступу, а також розмежувати доступ для різних фармацевтів.

Авторизація відбувається шляхом введення користувачем з клавіатури комбінації логіну та паролю. Логін та пароль користувача зберігається у відповідній таблиці БД. Для того, щоб захистити дані про логін та пароль користувача від несанкціонованого доступу, БД також захищена паролем, який знає лише адміністратор, що вносить ці дані в БД. Отже, для розробки методу необхідно під'єднати БД до середовища розробки, щоб отримати дані про логін та пароль користувача. Потім треба зробити перевірку на те, що комбінація логіну та паролю відповідає даним, які зберігаються в БД.

Якщо було введено коректні дані, то необхідно надати доступ користувачу до всіх функцій ПП. При цьому слід передати інформацію про логін користувача модулю вводу-виводу даних, для відображення її на головній формі ПП, для того, щоб було зрозуміло, який користувач зараз працює з програмою. Також ця інформація буде використана для створення замовлень від імені авторизованого користувача.

У випадку, якщо дані було введено некоректно, то необхідно вивести повідомлення про це користувачу, при цьому не надаючи йому доступ до інших функцій ПП.

Отже, було розроблено метод забезпечення авторизації користувачів, який дозволяє розмежувати доступ до програми для різних фармацевтів, а також захистити ПП від несанкціонованого доступу.

3.3 Проєктування інтерфейсу

Для розробки графічного інтерфейсу ПП було використано бібліотеку Windows Forms, яка є частиною .NET Framework. Кожен елемент в Windows Forms є екземпляром головного класу Control, що дозволяє підтримувати функціональність конкретного елемента, наприклад розмір елемента, його колір, параметри тексту, а також різні події, пов'язані з елементом, наприклад при натисканні або переміщенні.

Опис елементів головної форми ПП та форми авторизації буде наведено в наступних підрозділах.

3.3.1 Інтерфейс головної форми

Головна форма ПП складається з багатьох елементів керування, які забезпечують роботу модуля вводу-виводу даних та обчислювального модуля.

Взаємодія з елементами на головній формі ПП дозволяють користувачу переглядати дані про ліки, фармацевтів, замовлення, матрицю відстаней, а

також переглядати інформацію про маршрути проходження маніпулятора автомату видачі ліків, довжину та час їхнього проходження, додавати ліки в замовлення, виконувати розрахунок та пошук маршрутів проходження маніпулятора автомату видачі ліків, а також довжини та часу їхнього проходження.

Перелік елементів графічного інтерфейсу головної форми Form1 наведено у табл. 3.4.

Таблиця 3.4 – Елементи графічного інтерфейсу головної форми Form1

Назва елемента	Тип	Опис
Button1	Button	Кнопка, яка дозволяє додавати ліки в замовлення
Button2	Button	Кнопка створення нового замовлення
Button3	Button	Кнопка, яка дозволяє будувати матрицю відстаней для обраних ліків в замовленні, виконувати розрахунок та пошук маршрутів проходження маніпулятора автомату видачі ліків, а також довжини та часу їхнього проходження
Button4	Button	Кнопка, яка дозволяє скасувати замовлення
Button5	Button	Кнопка повернення до створення нового замовлення
Button6	Button	Кнопка виходу з програми
Button7	Button	Кнопка, яка дозволяє виконувати пошук замовлення в БД за датою замовлення
Button8	Button	Кнопка, яка дозволяє виконувати пошук фармацевта в БД за ПІБ
Button9	Button	Кнопка, яка дозволяє скасувати пошук замовлення в БД за датою замовлення
Button10	Button	Кнопка, яка дозволяє скасувати пошук фармацевта в БД за ПІБ
DataGridView1	DataGridView	Таблиця, яка відображає загальну інформацію про назву ліків, вагу упаковки та розмір упаковки з ліками і дозволяє обирати ліки в замовлення
DataGridView2	DataGridView	Таблиця, яка відображає матрицю відстаней для обраних ліків в замовленні

Продовження таблиці 3.4

Назва елементу	Тип	Опис
DataGridView3	DataGridView	Таблиця, яка відображає інформацію про оптимальний маршрут проходження маніпулятора автомату видачі ліків
DataGridView4	DataGridView	Таблиця, яка відображає інформацію про назву обраних ліків в замовленні
DataGridView5	DataGridView	Таблиця, яка відображає інформацію про дату замовлення та ПІБ фармацевтів
Label6	Label	Напис, який відображає інформацію про ПІБ авторизованого фармацевта
MaskedTextBox1	MaskedTextBox	Текстове поле для пошуку замовлення в БД за датою замовлення
TextBox1	TextBox	Текстове поле для пошуку фармацевта в БД за ПІБ
TabControl1	TabControl	Панель вкладок, яка містить вкладки «Розрахунок маршрута» та «Перелік замовлень», та дозволяє перемикатися між ними
Panel1	Panel	Панель, яка об'єднує в собі елементи, які відповідають за відображення інформації про матрицю відстаней та стандартний та оптимальний маршрути

Функції головної форми Form1, які реалізовані за допомогою елементів графічного інтерфейсу, наведено у табл. 3.5.

Таблиця 3.5 – Функції-оброблювачі подій головної форми Form1 та її елементів

Назва функції	Синтаксис	Опис
Form1_Load	private void Form1_Load(object sender, EventArgs e)	Містить події, які виконуються при завантаженні головної форми
Button1	private void button1_Click(object sender, EventArgs e)	При натисканні на кнопку виконується додавання обраних ліків в замовлення
Button2	private void button2_Click(object sender, EventArgs e)	При натисканні на кнопку виконується створення нового замовлення

Продовження таблиці 3.5

Назва функції	Синтаксис	Опис
Button4	private void button4_Click(object sender, EventArgs e)	При натисканні на кнопку створене замовлення скасовується
Button3	private void button3_Click(object sender, EventArgs e)	При натисканні на кнопку будується матриця відстаней для обраних ліків в замовленні, виконується розрахунок та пошук маршрутів проходження маніпулятора автомату видачі ліків, а також довжини та часу їхнього проходження
Button5	private void button5_Click(object sender, EventArgs e)	При натисканні на кнопку виконується повернення до створення нового замовлення
Button6	private void button6_Click(object sender, EventArgs e)	При натисканні на кнопку виконується вихід з програми
Button7	private void button7_Click(object sender, EventArgs e)	При натисканні на кнопку виконується пошук замовлення в БД за датою замовлення
Button8	private void button8_Click(object sender, EventArgs e)	При натисканні на кнопку виконується пошук фармацевта в БД за ПІБ
Button9	private void button9_Click(object sender, EventArgs e)	При натисканні на кнопку пошук замовлення в БД за датою замовлення скасовується
Button10	private void button10_Click(object sender, EventArgs e)	При натисканні на кнопку пошук фармацевта в БД за ПІБ скасовується

На рис. 3.1 представлений вигляд вкладки «Розрахунок маршрута» на головній формі Form1. На рис. 3.2 представлений вигляд вкладки «Перелік замовлень» на головній формі Form1. На рис. 3.3 представлений вигляд головної форми Form1 після натискання на кнопку «Розрахувати маршрут».

Фармацевт: Головченко А.Л. Вийти з програми

Розрахунок маршрута Перелік замовлень

Створити замовлення

Виберіть ліки для замовлення:

	Назва_ліків	Вага_упаковки	Розмір_упаковки
	x1	150	36
	x2	225	247
▶	x3	223	215
	x4	129	185
	x5	10	112
	x6	233	295
	x7	243	45
	x8	206	207

Ліки в замовленні

▶	0
	x5
	x1
	x4
	x2
	x3
*	

Габарити ліків
Загальна вага упаковки: 737 / 800 г
Загальний об'єм упаковки: 795 / 1500 куб. см

Додати ліки Скасувати замовлення

Розрахувати маршрут

Рисунок 3.1 – Вигляд вкладки «Розрахунок маршрута» на головній формі Form1

Фармацевт: Головченко А.Л. Вийти з програми

Розрахунок маршрута Перелік замовлень

ПІБ_фармацевта	Дата_замовлення
▶ Головченко А.Л.	24.04.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021
Головченко А.Л.	03.05.2021

Пошук замовлення за датою:

ОК Скасувати

Пошук замовлення за ПІБ фармацевта:

ОК Скасувати

Рисунок 3.2 – Вигляд вкладки «Перелік замовлень» на головній формі Form1

AutoPharm

Фармацевт: Головченко А.Л.

Вийти з програми

Матриця відстаней між ліками:

	0	x5	x1	x4	x2	x3
▶	0	25	5	20	10	15
	25	0	20	5	15	10
	5	20	0	15	5	10
	20	5	15	0	10	5
	10	15	5	10	0	5
*	15	10	10	5	5	0

Маршрут проходження:

▶	0 -> x1	відстань 5 см
	x1 -> x2	відстань 5 см
	x2 -> x3	відстань 5 см
	x3 -> x4	відстань 5 см
	x4 -> x5	відстань 5 см
*	x5 -> 0	відстань 25 см

Інформація про стандартний маршрут

Загальна довжина маршруту: 90 см

Час проходження маршруту: 3,36 с

Інформація про оптимальний маршрут

Загальна довжина маршруту: 50 см

Час проходження маршруту: 3,2 с

Повернутися до створення замовлення

Рисунок 3.3 – вигляд головної форми Form1 після натискання на кнопку «Розрахувати маршрут»

3.3.2 Інтерфейс форми авторизації

Форма авторизації складається з елементів, які забезпечують роботу модуля авторизації. Елементи керування дозволяють користувачу отримати доступ до головної форми ПП за умови, якщо він знає відповідний логін та пароль, який зберігається в БД.

Перелік елементів графічного інтерфейсу форми авторизації Form2 наведено у табл. 3.6.

Таблиця 3.6 – Елементи графічного інтерфейсу форми авторизації Form2

Назва елементу	Тип	Опис
Button1	Button	Кнопка, яка дозволяє отримати доступ до головної форми ПП
Button2	Button	Кнопка, яка дозволяє вийти з програми

Продовження таблиці 3.6

TextBox1	TextBox	Текстове поле для вводу логіну користувача
TextBox2	TextBox	Текстове поле для вводу пароля користувача

Перелік елементів графічного інтерфейсу форми авторизації Form2 наведено у табл. 3.7.

Таблиця 3.7 – Функції-оброблювачі подій форми авторизації Form2 та її елементів

Назва елементу	Тип	Опис
Form2_Load	private void Form2_Load(object sender, EventArgs e)	Містить події, які відбуваються при завантаженні форми авторизації
Button1	private void button1_Click(object sender, EventArgs e)	При натисканні на кнопку виконується відображення головної форми ПП
Button2	private void button2_Click(object sender, EventArgs e)	При натисканні на кнопку виконується вихід з програми

На рис. 3.4 представлений загальний вигляд форми авторизації Form2

Рисунок 3.5 – Загальний вигляд форми авторизації Form2

3.4 Висновок

В даному розділі було виконано опис створення методу пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків. В розробленому методі передбачено розрахунок стандартного маршруту проходження та пошук оптимального маршруту. Також він дозволяє виконати розрахунок загальної довжини стандартного та оптимального маршрутів та час їхнього проходження.

Також було виконано тестування створеного методу на конкретному прикладі, в якому було описане замовлення з 20 можливих ліків. При тестуванні методу було побудовано стандартний та оптимальний маршрути проходження маніпулятором 20 можливих ліків та встановлено, що при використанні саме оптимального маршруту можна отримати економію часу, яка становить 14,28 с завдяки скороченню загальної відстані оптимального маршруту на 3570 см у порівнянні зі стандартним маршрутом.

Метод був створений та протестований з використанням евристичного жадібного алгоритму, що дозволило дуже швидко знайти та розрахувати оптимальний маршрут проходження при великій кількості ліків.

Також в даному розділі було описано проєктування інтерфейсу ПП, який складається з головної форми та форми авторизації.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання

Розроблений ПП дозволяє оптимізувати роботу автомата видачі ліків, шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

ПП повинен відповідати наступним мінімальним системним вимогам:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою 1,8 ГГц.
- 1 ГБ оперативної пам'яті;
- 1 ГБ вільного місця на жорсткому диску.

4.2 Характеристики програмного продукту

ПП містить в собі такі модулі:

- зберігання даних. Для зберігання даних застосовується СУБД Microsoft Access;
- обчислювальний. Цей модуль є основним модулем програмного продукту, його завдання полягає в тому, щоб визначити оптимальний маршрут проходження маніпулятора автомата видачі ліків. Для того, щоб внести зміни в обчислювальний модуль необхідно використати середовище програмування Microsoft Visual Studio;
- вводу-виводу даних. Цей модуль дозволяє отримувати дані про ліки, замовлення та фармацевтів зі сховища та виводити їх на екран, а також передавати дані обчислювальному модулю для подальшої їх обробки. Для того, щоб внести зміни в модуль вводу-виводу даних необхідно використати середовище програмування Microsoft Visual Studio;
- авторизації. Цей модуль захищає ПП від несанкціонованого доступу за допомогою логіну та паролю, а також дозволяє розмежувати доступ до

програмного продукту, для того щоб вести облік даних різних користувачів. Для того, щоб внести зміни в модуль авторизації необхідно використати середовище програмування Microsoft Visual Studio.

4.3 Звертання до програмного продукту

Для того, щоб завантажити програму треба виконати копіювання файлу програми AutoPharm.exe, а також БД AutoPharmDB.mdb з флеш-накопичувача у потрібну директорію.

Для того, щоб почати працювати з програмою треба запустити на виконання файл AutoPharm.exe.

4.4 Вхідні і вихідні дані

В якості вхідних даних ПП використовуються дані про назву ліків, вагу упаковки та об'єм упаковки, які зберігаються в БД. Також вхідними даними є значення ПІБ фармацевта та дати замовлення, які вводяться користувачем з клавіатури. Приклад вхідних даних, які зберігаються в БД наведено на рис 4.1.

Назва_ліків	Вага_упаковки	Розмір_упаковки
x1	150	36
x2	225	247
x3	223	215
x4	129	185
x5	10	112
x6	233	295
x7	243	45

Рисунок 4.1 – Інформація про ліки

Як можна побачити з рис 4.1 у першому стовпчику вказано назву ліків, у другому – вагу упаковки, а в третьому – розмір упаковки.

Вихідними даними є матриця відстаней, таблиця проходження оптимального маршруту, інформація про довжину стандартного маршруту проходження, інформація про довжину оптимального маршруту проходження, інформація про час стандартного маршруту проходження, інформація про час оптимального маршруту проходження.

4.5 Повідомлення

Під час роботи розробленого ПП дії користувача можуть призводити до помилок. ПП обробляє всі помилки, а також генерує повідомлення про них, для того, щоб користувач мав можливість їх виправити. Повідомлення про помилки, що генеруються ПП описані в табл. 4.1.

Таблиця 4.1 – Повідомлення про помилки при роботі з «AutoPharm»

Повідомлення	Опис
Габарити ліків за межами діапазону!	Повідомлення про те, що необхідно обрати ліки, параметри яких не перевищують максимальний діапазон
Помилка при додаванні ліків!	Повідомлення про те, що при додаванні ліків в замовлення виникла помилка
Введіть дату замовлення!	Повідомлення означає, що треба ввести інформацію про дату замовлення у відповідне текстове поле
Введіть ПІБ фармацевта!	Повідомлення означає, що треба ввести інформацію про ПІБ фармацевта у відповідне тектове поле
Введено невірний логін та/або пароль!	Повідомлення означає, що необхідно ввести коректні дані про логін та пароль користувача у відповідні текстові поля

4.6 Висновок

В даному розділі був зроблений опис призначення та визначено умови використання розробленого ПП, визначено мінімальні системні вимоги, при яких робота ПП буде можливою. Вказано дії, які потрібно виконати для запуску ПП. Було описано характеристики ПП, в яких визначено модулі та їх функції. Виконано опис вхідних та вихідних даних, а також перелік повідомлень для користувача, які ПП генерує при обробці помилок, що користувач може робити під час роботи.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення й умови виконання програмного продукту

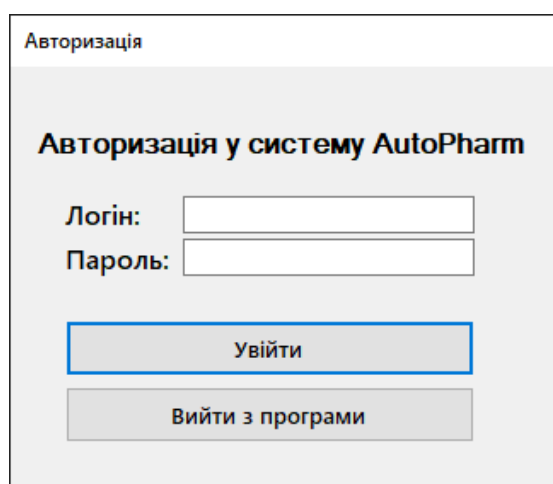
Розроблений ПП дозволяє оптимізувати роботу автомата видачі ліків, шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

ПП повинен відповідати наступним мінімальним системним вимогам:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою 1,8 ГГц.
- 1 ГБ оперативної пам'яті;
- 1 ГБ вільного місця на жорсткому диску.

5.2 Виконання програмного продукту

Після того, як файл AutoPharm.exe буде запущений на виконання відкриється вікно авторизації, яке представлено на рис 5.1.



Авторизація

Авторизація у систему AutoPharm

Логін:

Пароль:

Увійти

Вийти з програми

Рисунок 5.1 – Вікно авторизації ПП

Для того, щоб пройти авторизацію треба ввести логін користувача в поле «Логін» та пароль користувача в поле «Пароль». Після цього слід

натиснути кнопку «Увіти». Для виходу з програми на етапі авторизації треба натиснути кнопку «Вийти з програми».

Після проходження авторизації відкриється головне вікно програми на вкладці «Розрахунок маршрута», яке представлено на рис 5.2.

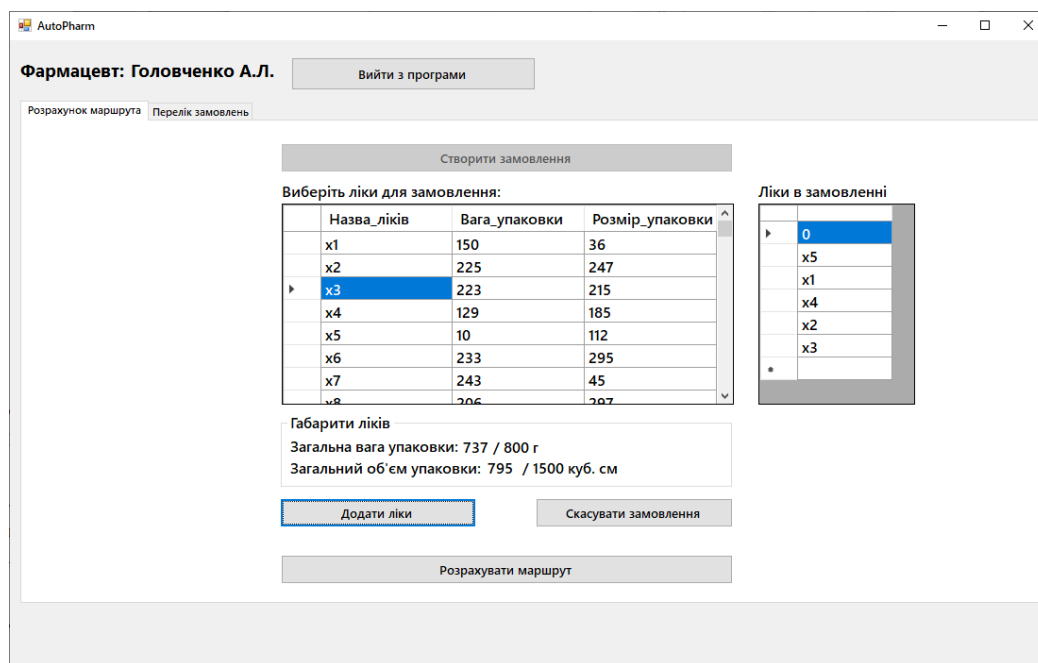


Рисунок 5.2 – Вигляд вкладки «Розрахунок маршрута» у головному вікні ПП

Для розрахунку маршрута треба спочатку створити замовлення, для цього необхідно натиснути кнопку «Створити замовлення» на вкладці «Розрахунок маршрута». Після цього треба додати потрібні ліки в замовлення, для цього необхідно натиснути на назву ліків в таблиці під написом «Виберіть ліки для замовлення», а потім натиснути на кнопку «Додати ліки». Назви ліків в замовленні при цьому відображаються в таблиці під написом «Ліки в замовленні». Після додавання всіх необхідних ліків до замовлення треба натиснути кнопку «Розрахувати маршрут» для розрахунку маршрута, або натиснути кнопку «Скасувати замовлення» для скасування замовлення.

Після натискання кнопки «Розрахунок маршрута» головне вікно ПП буде представлено в загальному вигляді, при цьому у головному вікні буде відображено інформацію про матрицю відстаней, а також про стандартний та

оптимальний маршрути проходження. Вигляд головного вікна ПП після натискання на кнопку «Розрахувати маршрут» представлений на рис 5.3.

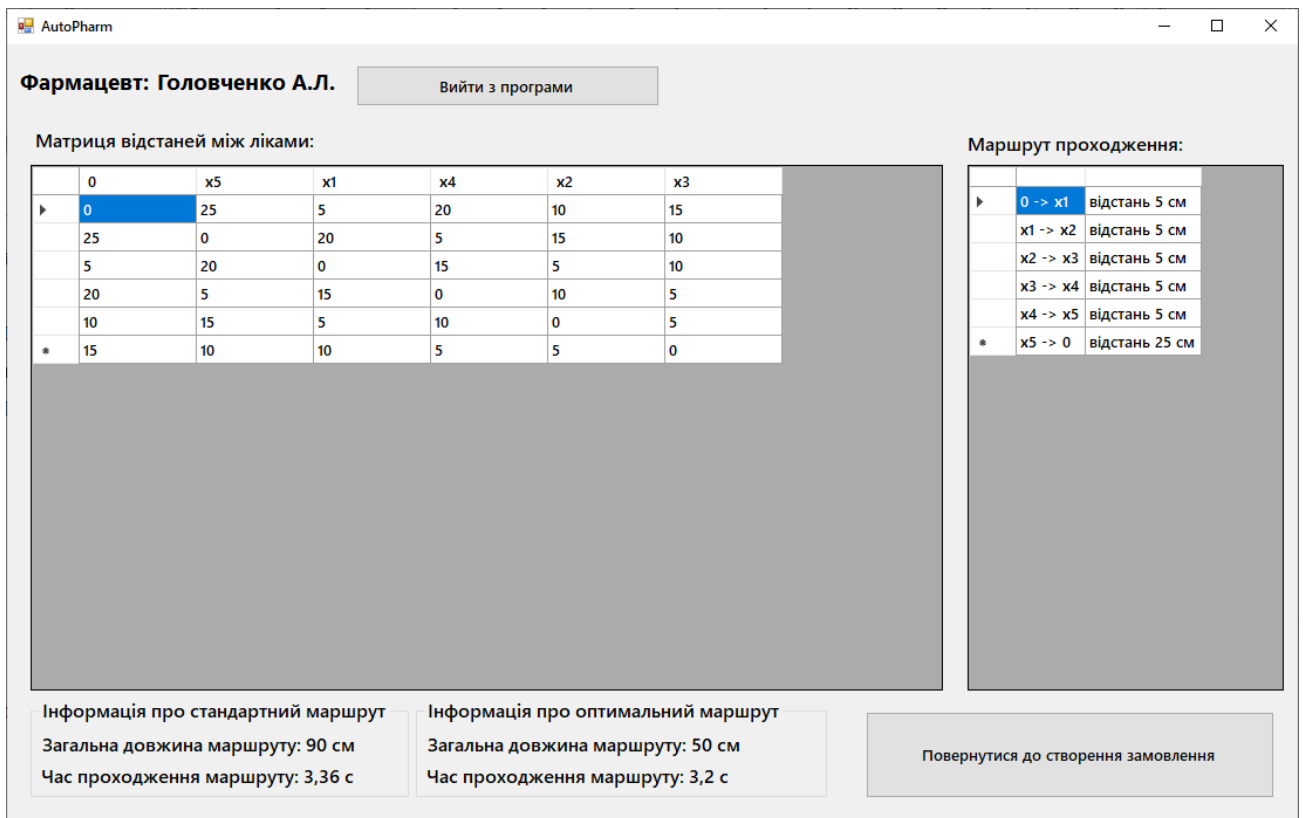


Рисунок 5.3 – Вигляд головного вікна ПП після натискання на кнопку «Розрахувати маршрут»

Для того, щоб повернутися на етап створення нового замовлення необхідно натиснути кнопку «Повернутися до створення замовлення».

Після повернення на етап створення нового замовлення у головному вікні знову з'явиться панель з вкладками. Для того, щоб переглянути створені замовлення, необхідно у головному вікні перейти на вкладку «Перелік замовлень». Головне вікно на вкладці «Перелік замовлень» представлено на рис 5.4.

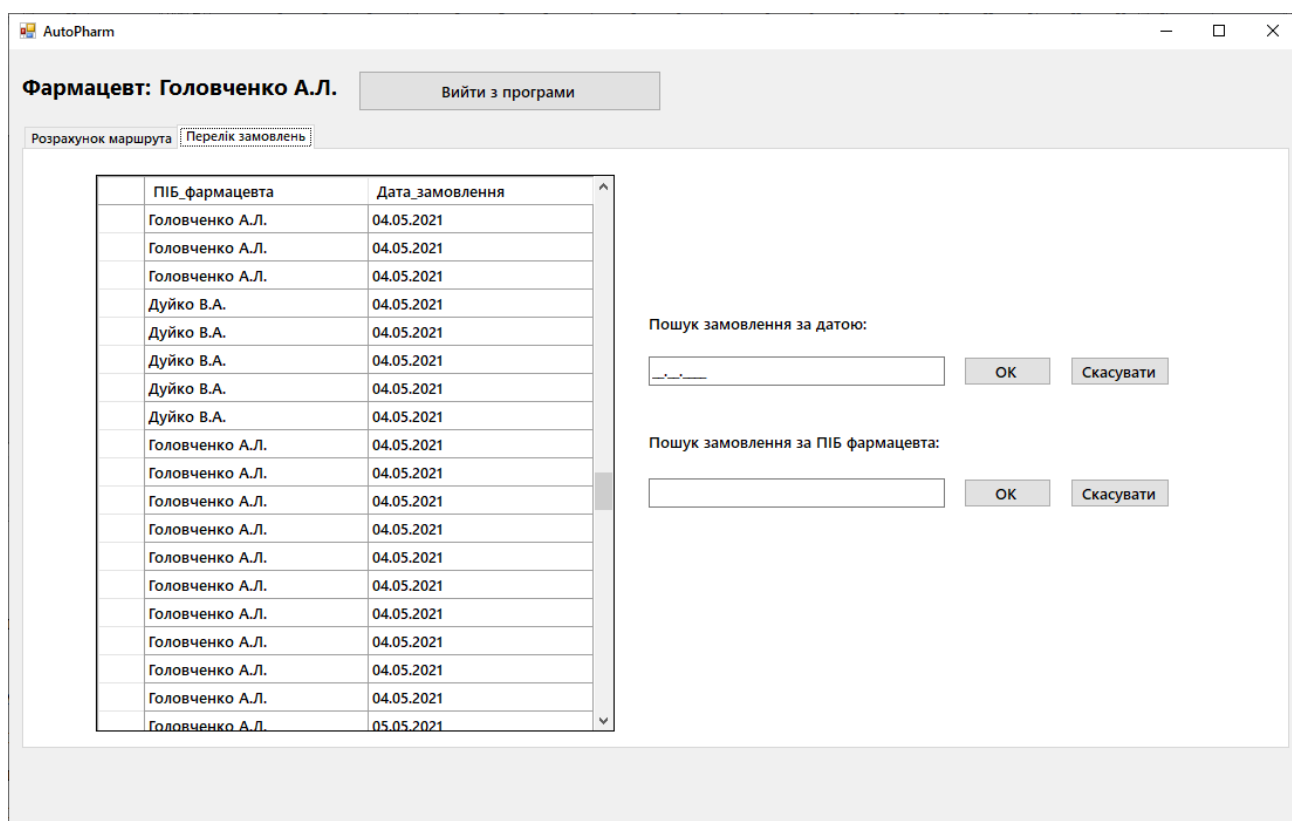


Рисунок 5.4 – Вигляд вкладки «Перелік замовлень» у головному вікні ПП

Для пошуку замовлень за датою необхідно ввести дату замовлення у текстове поле під написом «Пошук замовлення за датою» та натиснути кнопку «ОК» праворуч від цього текстового поля. Для того, щоб скасувати пошук замовлень за датою треба натиснути кнопку «Скасувати» праворуч від текстового поля.

Для пошуку замовлень за ПІБ фармацевта необхідно ввести дату замовлення у текстове поле під написом «Пошук замовлення за ПІБ фармацевта» та натиснути кнопку «ОК» праворуч від цього текстового поля. Для того, щоб скасувати пошук замовлень за ПІБ фармацевта треба натиснути кнопку «Скасувати» праворуч від текстового поля.

Для того, щоб завершити роботу з програмою треба натиснути кнопку «Вийти з програми».

5.3 Повідомлення оператору

Під час роботи розробленого ПП дії користувача можуть призводити до помилок. ПП обробляє всі помилки, а також генерує повідомлення про них, для того, щоб користувач мав можливість їх виправити. Повідомлення про помилки, що генеруються ПП описані в розділі «Керівництво програміста» в табл. 4.1.

5.4 Висновок

В даному розділі був зроблений опис призначення та визначено умови використання розробленого ПП, визначено мінімальні системні вимоги, при яких робота ПП буде можливою. Була описана покрокова інструкція для роботи оператора з ПП, в якій вказано які дії користувач потрібен робити для авторизації, при створенні замовлення та розрахунку маршрута проходження, а також при пошуку замовлень за датою та ПІБ фармацевта. Графічний інтерфейс користувача є інтуїтивним та простим для розуміння. При виникненні помилок при роботі з ПП генеруються повідомлення, які дозволяють оператору зрозуміти причину виникнення помилки, що дозволить швидко її виправити.

ВИСНОВКИ

Під час виконання роботи було виконано дослідження предметної області, розглянуто та визначено основні поняття, які пов'язані з обраною предметною областю. Також були проаналізовані та досліджені існуючі методи, кожен з яких можливо застосовувати для вирішення поставленої задачі, було визначено їх переваги та недоліки. Був проведений порівняльний аналіз досліджених методів, які можуть бути використані для вирішення задачі комівояжера для автомата видачі ліків. Було сформульовано і визначено завдання до роботи.

Метою роботи є створення програмної системи для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

Для виконання поставленої мети було виконано наступні завдання:

- створено метод для забезпечення авторизації користувачів;
- створено метод для додавання інформації про замовлення в БД;
- створено метод для визначення загальної кількості ліків в замовленні;
- створено метод для побудови матриці відстаней для полиць з ліками, які будуть включені в маршрут проходження маніпулятора автомату видачі ліків;
- створено метод для знаходження оптимального маршруту, який проходить маніпулятор;
- створено метод для виконання розрахунку стандартного маршруту проходження;
- створено метод для виконання розрахунку загального часу проходження оптимального маршруту;
- створено метод для виконання розрахунку загального часу проходження стандартного маршруту.

Також в роботі були розглянуті різні мови програмування та середовища розробки, які дозволяють розробляти програмні продукти. Було проведено їх

аналіз, виділено основні переваги і недоліки та складено порівняльну характеристику. В ході цієї роботи було встановлено, що для програмного продукту найбільш ефективною мовою програмування є C#, тому що вона більш захищена і працює швидше за інші мови програмування. Також найбільш ефективним середовищем розробки було обрано середовище Microsoft Visual Studio, тому що воно є офіційним для мови програмування C#, не потребує великих обсягів апаратних ресурсів, а також дозволяє отримати безкоштовний доступ до всього необхідного функціоналу.

Для роботи було розроблено структурну схему програмного продукту. Встановлено, що для зберігання вхідних даних найкращий тип сховища є реляційна БД у СУБД Microsoft Access.

Інтерфейс програмного продукту розроблено у середовищі розробки Microsoft Visual Studio за допомогою технології Windows Forms.

В результаті було створено програмну систему для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера. Створена програмна система дозволяє оптимізувати роботу автомата видачі ліків через скорочення часу на доставку ліків до фармацевта, а отже її можна використовувати в аптеках з такими автоматами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Бажан, М.П. Українська радянська енциклопедія / М.П. Бажан. – Київ, 1974–1985. – 542 с.
2. Иванов, А. А. Основы робототехники / А. А. Иванов – М.: ИНФРА-М, 2017. – 223 с.
3. Lawrence, F. Wolper. Health Care Administration: Planning, implementing, and managing organized delivery systems / F. Wolper Lawrence – US : Jones & Bartlett Learning, 2004. – P. 720.
4. EvoTec Pharmacy Automation Robot [Electronic resource]. – Access mode: <https://www.swisslog-healthcare.com>
5. Automated dispensing system (ADS) for pharmacies [Electronic resource] – Access mode: <https://newicon.fi/pharmacy-automation-in-pharmacies/automated-dispensing-system>
6. CONSIS Dispensing Systems [Electronic resource] – Access mode: <https://www.willach-pharmacy-solutions.com/EN/consis-robotic-systems>
7. BD Rowa pharmacy robots for the storage of your medication [Electronic resource] – Access mode: <https://rowa.de/en/solutions/store-pick/>
8. PharmBot – провизор, которому не нужен отдых [Электронный ресурс] – Режим доступа: <http://pharmbot.biz/>
9. Comparative Study of Ant Colony Algorithms for Multi-Objective Optimization / [J. Ning, C. Zhang, P. Sun et al.] // Information. – 2018. – Vol. 10, № 1. – P. 4–11.
10. Субботін С. О. Інтелектуальні системи : Навчальний посібник / С. О. Субботін, А. О. Олійник ; під заг. ред. проф. С. О. Субботіна. – Запоріжжя: ЗНТУ, 2014. – 216 с.
11. Олійник, А. О. Інтелектуальний аналіз даних / А. О. Олійник, С. О. Субботін, О. О. Олійник. – Запоріжжя : ЗНТУ, 2012. – 278 с.
12. Субботін С. О. Нейронні мережі : Навчальний посібник / С. О. Субботін, А. О. Олійник ; під заг. ред. проф. С. О. Субботіна. –

Запоріжжя: ЗНТУ, 2014. – 135 с.

13. Subbotin, S. Radial-Basis Function Neural Network Synthesis on the Basis of Decision Tree / S. Subbotin // Optical Memory and Neural Networks. – 2020. – Vol. 29. – P. 7–18.

14. Subbotin, S. A. The training set quality measures for neural network learning / S. A. Subbotin // Optical Memory and Neural Networks (Information Optics). – 2010. – Vol. 19, № 2. – P. 126–139.

15. Клепко, В. Ю. Вища математика в прикладах і задачах / В. Ю. Клепко, В.Л. Голець – К. : Центр учбової літератури, 2009. – 162 с.

16. Маркеев, А. П. Теоретическая механика / А.П. Маркеев – М.: Наука, 1990. – 416 с.

17. Wickham, H. R for Data Science: Visualize, Model, Transform, Tidy, and Import Data / H. Wickham, G. Grolemund. – Sebastopol : O'Reilly Media, 2016. – 520 p.

18. Прата, С. Язык программирования C++. Лекции и упражнения / С. Прата. – М. : Диалектика-Вильямс, 2016. – 1248 с.

19. Perkins, B. Hammer and J. Reid, Beginning c# 7 programming with visual studio / B. Perkins, J. Hammer, J. Reid. – Indianapolis : John Wiley & Sons, 2017. – 884 p

20. МакГрат, М. Программирование на Python для начинающих. Лекции и упражнения / М. МакГрат. – М. : Диалектика, 2015. – 192 с.

21. Сьерра, К. Изучаем Java. Лекции и упражнения / К. Сьерра, Б. Бэйтс. – М. : Эксмо, 2017. – 720 с.

22. JetBrains .NET IDE hits RTM [Electronic resource] – Access mode: <https://blog.jetbrains.com/dotnet/2017/08/03/rider-2017-1-jetbrains-net-ide-hits-rtm/>

23. Carlson, D, Eclipse Distilled / D. Carlson. – Addison-Wesley Professional, 2001. – 108 p

24. Извольский, Н. А. Геометрия на плоскости (планиметрия) / Н. А. Извольский – Л. : Типография имени Н. Бухарина, 1924. – 85 с.

25. Rosen, H. Kenneth, Handbook of Discrete and Combinatorial

Mathematics / Kenneth H. Rosen – CRC Press, 1999. – 93 p

ДОДАТОК А
Технічне завдання

Вступ

Створений ПП застосовується для того, щоб вирішити проблему, пов'язану з часом доставки великої кількості ліків в замовленні автоматом видачі ліків, шляхом пошуку оптимального маршруту проходження маніпулятора за допомогою вирішення задачі комівояжера.

ПП «AutoPharm» має функцію авторизації користувачів, дозволяє додавати інформацію про замовлення у БД, визначати загальну кількість ліків в замовленні, будувати матрицю відстаней для полиць з ліками, які будуть включені в маршрут проходження маніпулятора автомата видачі ліків, а також знаходити оптимальний маршрут, який проходить маніпулятор та розраховувати його загальну відстань та час проходження.

A.1 Мета розробки і її призначення

Об'єкт дослідження – автоматизовані системи видачі ліків в аптеці.

Предмет дослідження – методи пошуку оптимального маршруту для маніпулятора автомата видачі ліків.

Мета роботи – створення програмної системи для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

Актуальність та проблематика роботи полягає в тому, що на сьогоднішній день у багатьох аптеках використовуються сучасні автомати для видачі ліків, які містять в собі велику кількість ліків і мають можливість обробляти замовлення великого обсягу. Тому для скорочення часу доставки великої кількості ліків в замовленні до покупця виникла необхідність розробки методів пошуку оптимального маршруту для маніпулятора автомата видачі ліків.

А.2 Стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків

На сьогоднішній день автоматизація застосовується в багатьох сферах та галузях життя, в тому числі і в медицині. У багатьох країнах світу з'являються автоматизовані аптеки, в яких застосовуються автомати видачі ліків. Поступово ці автомати з'являються в тому числі і в Україні.

Автомат видачі ліків виконує пошук потрібних ліків на складі і за допомогою механізму переміщення доставляє ліки покупцю. Використання подібного автомату дозволяє полегшити та оптимізувати роботу фармацевтів, тим самим обслуговування клієнтів проходить набагато швидше, а також це дозволяє зменшити вплив людського фактору, що знижує кількість помилок при пошуку ліків. Але шлях, який долає маніпулятор автомату видачі ліків може бути не завжди оптимальним. Через це збільшується час обслуговування кожного клієнта, і таким чином автоматизація перестає бути достатньо ефективною.

А.3 Підстави для розробки

Підставою для виконання розробки є завдання на дипломний проєкт на тему «Програмна реалізація вирішення задачі комівояжера для автомату видачі ліків», затверджене наказом по НУ «Запорізька політехніка» № 103 від 30 березня 2021 року.

А.4 Основні вимоги до програми, що розробляється, початковим даним і результату

А.4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

– повинен бути реалізований графічний інтерфейс користувача;

- повинна бути реалізована можливість завантаження даних з БД;
- повинна бути реалізована функція для створення замовлення;
- повинна бути реалізована можливість додавати інформацію про замовлення в БД;
- повинна бути реалізована функція для додавання ліки до замовлення;
- повинен бути реалізований метод визначення координат полиць з ліками в замовленні;
- повинен бути реалізований метод побудови матриці відстаней для полицями з ліками в замовленні;
- повинен бути реалізований метод пошуку оптимального маршруту проходження для матриці відстаней;
- повинен бути реалізований метод розрахунку стандартного маршруту проходження для матриці відстаней;
- повинен бути реалізований метод розрахунку інформації про час та довжину оптимального маршруту проходження для матриці відстаней;
- повинен бути реалізований метод розрахунку інформації про час та довжину стандартного маршруту проходження для матриці відстаней;
- повинна бути реалізована можливість для обробки помилок користувача під час виконання програми та відображення повідомлення про помилки.

А.4.2 Вимоги до надійності

ППП повинен відповідати таким вимогам до надійності:

- захищати PPP від несанкціонованого доступу;
- захищати PPP від помилкових дій користувача;
- здійснювати обробку виняткових ситуацій.

ППП повинен нормально функціонувати без програмних збоїв та помилок при безперервній роботі комп'ютера. У разі виникнення апаратних збоїв, нормальна робота PPP може бути відновлена після перезавантаження системи.

A.4.3 Умови експлуатації

Умови експлуатації:

- ПП не потребує обслуговуючого персоналу;
- для того, щоб використовувати ПП не потрібна додаткова кваліфікація.

A.4.4 Вимоги до складу та параметрів технічних засобів

ПП повинен відповідати наступним мінімальним системним вимогам:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою 1,8 ГГц.
- 1 ГБ оперативної пам'яті;
- 1 ГБ вільного місця на жорсткому диску.

A.4.5 Вимоги до маркування та упакування

Програмне забезпечення не потребує додаткових дій при маркуванні та упакуванні.

A.4.6 Вимоги до транспортування та збереження

Вимоги до транспортування й збереження є аналогічні вимогам, які висуваються до носія, на якому записаний ПП.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- керівництво програміста;

- керівництво оператора;
- опис програми.

А.6 Основні обмеження на установку і використання програмного продукту

Розроблюваний ПП не потребує інсталяції. Програмний продукт повинен використовуватись тільки на тих комп'ютерах, які задовільняють мінімальним вимогам та параметрам технічних засобів. Програмний продукт не конфліктує з іншими програмними продуктами та його використання може здійснюватись користувачем у будь-який момент часу.

ДОДАТОК Б
Опис програмного продукту

Б.1 Загальні відомості

Назва програмного продукту – «AutoPharm».

Створений ПП застосовується для того, щоб вирішити проблему, пов'язану з часом доставки великої кількості ліків в замовленні автоматом видачі ліків, шляхом пошуку оптимального маршруту проходження маніпулятора за допомогою вирішення задачі комівояжера.

ПП «AutoPharm» має функцію авторизації користувачів, дозволяє додавати інформацію про замовлення у БД, визначати загальну кількість ліків в замовленні, будувати матрицю відстаней для полиць з ліками, які будуть включені в маршрут проходження маніпулятора автомату видачі ліків, а також знаходити оптимальний маршрут, який проходить маніпулятор та розраховувати його загальну відстань та час проходження.

ПП повинен відповідати наступним мінімальним системним вимогам:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою 1,8 ГГц.
- 1 ГБ оперативної пам'яті;
- 1 ГБ вільного місця на жорсткому диску.

Б.2 Функціональне призначення

Вимоги до функціональних характеристик:

- повинен бути реалізований графічний інтерфейс користувача;
- повинна бути реалізована можливість завантаження даних з БД;
- повинна бути реалізована функція для створення замовлення;
- повинна бути реалізована можливість додавати інформацію про замовлення в БД;
- повинна бути реалізована функція для додавання ліки до замовлення;

- повинен бути реалізований метод визначення координат полиць з ліками в замовленні;
- повинен бути реалізований метод побудови матриці відстаней для полицями з ліками в замовленні;
- повинен бути реалізований метод пошуку оптимального маршруту проходження для матриці відстаней;
- повинен бути реалізований метод розрахунку стандартного маршруту проходження для матриці відстаней;
- повинен бути реалізований метод розрахунку інформації про час та довжину оптимального маршруту проходження для матриці відстаней;
- повинен бути реалізований метод розрахунку інформації про час та довжину стандартного маршруту проходження для матриці відстаней;
- повинна бути реалізована можливість для обробки помилок користувача під час виконання програми та відображення повідомлення про помилки.

Б.3 Опис логічної структури

ПП повинен містити в собі такі модулі:

- зберігання даних;
- обчислювальний;
- вводу-виводу даних;
- авторизації.

В якості вхідних даних ПП повинен використовувати інформацію про ліки, яка зберігається в БД.

Опис сутностей БД та їх атрибути такі як назва атрибута, тип, обов'язковість заповнення та розмір представлені в табл. Б.1.

Таблиця Б.1 – Сутності та їх атрибути

Сутність	Атрибут	Тип	Розмір	Обов'язкове	Опис сутності
Ліки	Код_ліків	Лічильник	Довге ціле	Так	Містить інформацію про ліки та їх параметри
	Назва_ліків	Текстовий	255	Так	
	Вага_упаковки	Числовий	Довге ціле	Так	
	Розмір_упаковки	Числовий	Довге ціле	Так	
Фармацевти	Код_фармацевта	Числовий	Довге ціле	Так	Містить інформацію про фармацевтів
	ПІБ_фармацевта	Текстовий	255	Так	
	Пароль_фармацевта	Текстовий	255	Так	
Замовлення	Код_замовлення	Лічильник	Довге ціле	Так	Містить інформацію про замовлення
	Код_фармацевта	Числовий	Довге ціле	Так	
	Дата_замовлення	Числовий	Довге ціле	Так	

Як можна побачити в табл. Б.1 визначено декілька сутностей, а саме сутність Ліки, яка містить в собі назву ліків, а також вагу та розмір упаковки, сутність Фармацевти, яка містить ПІБ фармацевта та його особистий пароль для авторизації, сутність Замовлення, яка містить інформацію про ПІБ фармацевта, який робив замовлення та дату замовлення.

На рис. Б.1 представлена ER-діаграма БД.

Наступною важливою складовою програмного продукту є графічний інтерфейс, за допомогою якого користувач взаємодіє з програмою.

Основною функцією графічного інтерфейсу є можливість переглядати інформацію про ліки, фармацевтів та замовлення, додавати ліки в замовлення, а також переглядати дані про стандартний і оптимальний маршрути та час їхнього проходження.

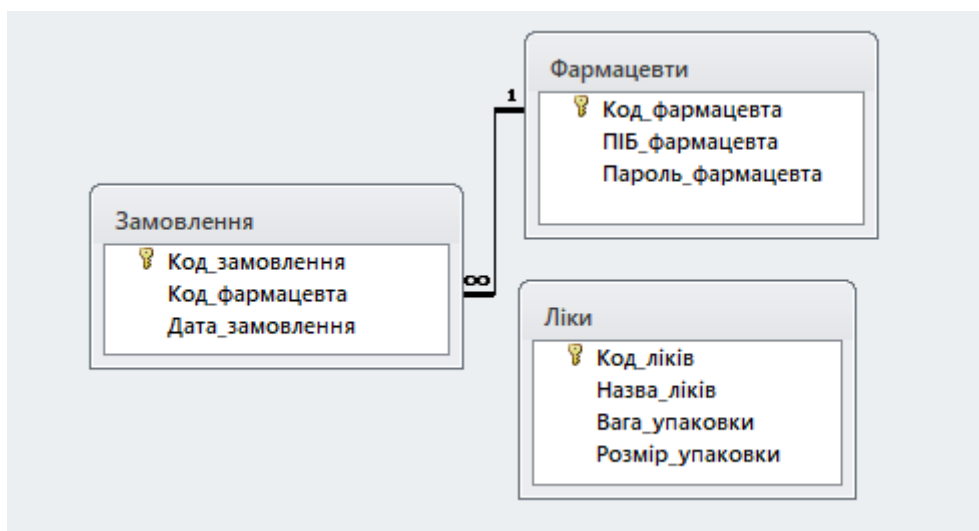


Рисунок Б.1 – ER–діаграма БД

Основним модулем програмного продукту є обчислювальний модуль. Його завдання полягає в тому, щоб визначити оптимальний маршрут проходження маніпулятора автомата видачі ліків. Цей модуль містить в собі такі функції:

- функція, яка визначає координати ліків, через які буде проходити маршрут;
- функція, розрахунку відстаней між ліками;
- функція розрахунку часу проходження стандартного маршруту;
- функція розрахунку часу проходження оптимального маршруту;
- функція обчислення довжини стандартного маршруту;
- функція обчислення довжини оптимального маршруту;
- функція розрахунку оптимального маршруту;
- функція пошуку оптимального маршруту.

Для реалізації обчислювального модуля була обрана мова програмування C# та середовище розробки Microsoft Visual Studio.

Модуль вводу-виводу даних дозволяє отримувати дані про ліки, замовлення та фармацевтів зі сховища та виводити їх на екран, а також передавати дані обчислювальному модулю для подальшої їх обробки.

Функції модуля вводу-виводу даних:

- функція додавання інформації про замовлення в БД;
- функція відображення інформації про замовлення;
- функція відображення інформації про ліки;
- функція, яка визначає назви ліків, між якими буде прокладено маршрут;
- функція пошуку та відображення потрібного замовлення за датою;
- функція пошуку та відображення потрібного фармацевта за ПІБ;
- функція відображення матриці відстаней між ліками;
- функція відображення інформації про стандартний маршрут;
- функція відображення інформації про оптимальний маршрут.

Важливим модулем також є модуль авторизації, який захищає ПП від несанкціонованого доступу за допомогою логіну та паролю. Також модуль авторизації дозволяє розмежувати доступ до програми, для того щоб вести облік даних різних користувачів.

На рис Б.2 представлена структурна схема програмного продукту.

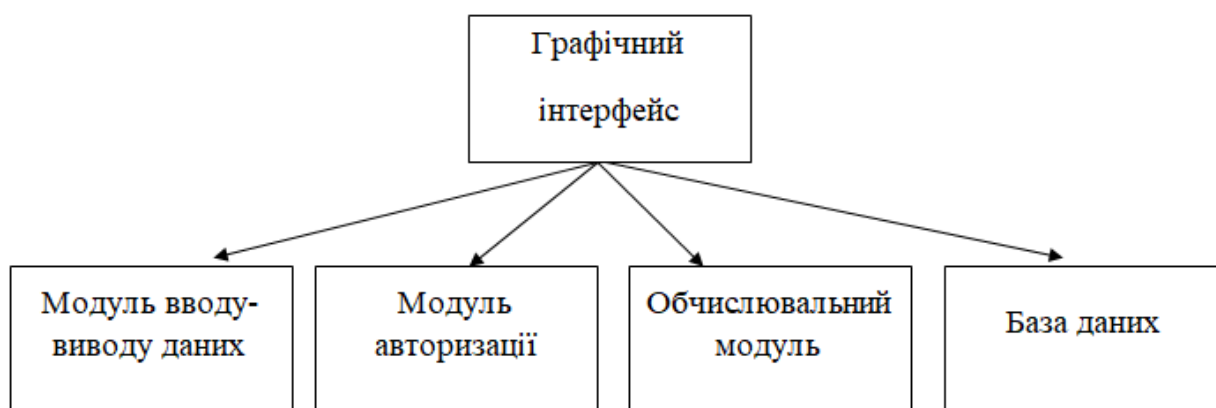


Рисунок Б.2 – Структурна схема програмного продукту

Як можна побачити на рис Б.2 ПП має графічний інтерфейс, який дозволяє візуалізувати вхідні та вихідні дані та надає користувачу зручний доступ до управління програмою. Графічний інтерфейс розробляється в середовищі розробки Microsoft Visual Studio на мові програмування C#.

Модуль вводу-виводу даних дозволяє надавати отримувати дані з БД та відображати їх за допомогою графічного інтерфейсу. Модуль авторизації дозволяє захистити ПП від несанкціонованого доступу, а також розмежувати доступ до програмного продукту.

Також ПП має обчислювальний модуль, який використовує вхідні дані для обчислення різних параметрів, таких як час проходження маршруту, довжина маршруту та довжина відстаней між точками маршруту. Модуль вводу-виводу даних, модуль авторизації та обчислювальний модуль також розробляється в середовищі розробки Microsoft Visual Studio та на мові програмування C#.

Вхідні дані такі як інформація про ліки, інформація про фармацевтів та інформація про замовлення зберігаються в реляційній БД, для цього використовується СУБД Microsoft Access.

Б.4 Вибір інструментарію розробки програмного продукту

Для розробки ПП було обрано мову програмування C#, тому що вона є об'єктно-орієнтованою мовою програмування, підтримує автоматичний збір сміття, має високу надійність та захищеність, є повністю об'єктно-орієнтованою мовою програмування, а також має обширну бібліотеку класів, що спрощує реалізацію багатьох функцій.

Для розробки ПП було обрано середовище програмування Microsoft Visual Studio, тому що воно надає можливість безкоштовного використання, є офіційним середовищем розробки для мови C# та підтримує платформу .NET. Також воно надає змогу розробляти консольні та графічні програмні продукти, має вбудований відладчик коду, підтримує велику кількість додаткових бібліотек для розширення базового функціоналу, в тому числі для візуального проектування. Для зберігання даних програмного продукту була обрана використати СУБД реляційного типу Microsoft Access.

Б.5 Виклик і завантаження

Для того, щоб завантажити програму треба виконати копіювання файлу програми AutoPharm.exe, а також БД AutoPharmDB.mdb з флеш-накопичувача у потрібну директорію.

Для того, щоб почати працювати з програмою треба запустити на виконання файл AutoPharm.exe.

Б.6 Приклади реалізації

Після проходження авторизації відкриється головне вікно програми на вкладці «Розрахунок маршрута», яке представлено на рис Б.3.

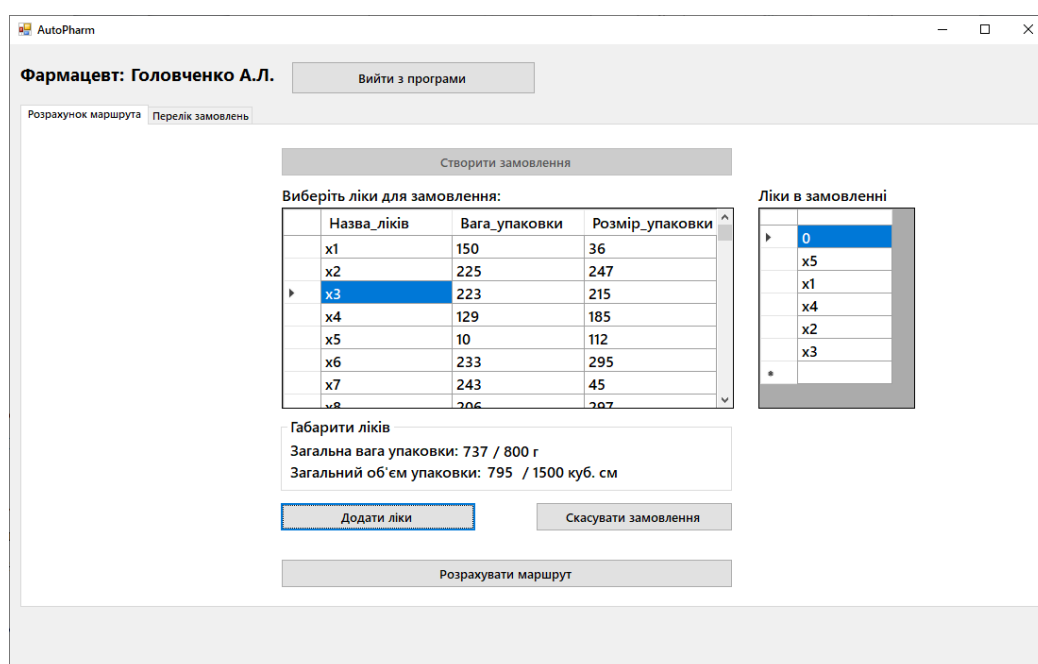


Рисунок Б.3 – Вигляд вкладки «Розрахунок маршрута» у головному вікні ПП

Для розрахунку маршрута треба спочатку створити замовлення, для цього необхідно натиснути кнопку «Створити замовлення» на вкладці «Розрахунок маршрута». Після цього треба додати потрібні ліки в замовлення, для цього необхідно натиснути на назву ліків в таблиці під написом «Виберіть

ліки для замовлення», а потім натиснути на кнопку «Додати ліки». Назви ліків в замовленні при цьому відображаються в таблиці під написом «Ліки в замовленні». Після додавання всіх необхідних ліків до замовлення треба натиснути кнопку «Розрахувати маршрут» для розрахунку маршрута, або натиснути кнопку «Скасувати замовлення» для скасування замовлення.

Після натискання кнопки «Розрахунок маршрута» головне вікно ПП буде представлено в загальному вигляді, при цьому у головному вікні буде відображено інформацію про матрицю відстаней, а також про стандартний та оптимальний маршрути проходження. Приклад відображення результатів роботи представлений на рис Б.4.

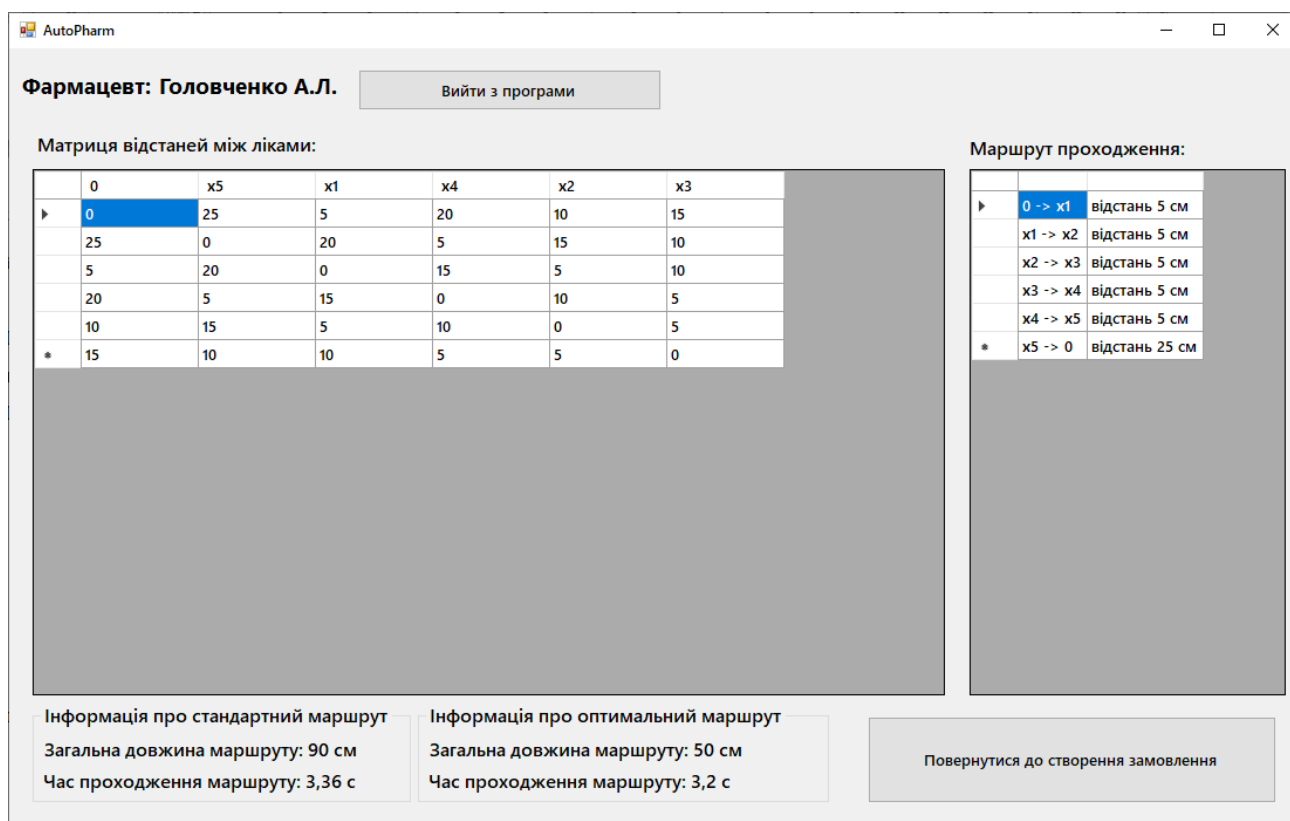


Рисунок Б.4 – Вигляд головного вікна ПП після натискання на кнопку «Розрахувати маршрут»

Б.7 Вхідні та вихідні дані

В якості вхідних даних ПП використовуються дані про назву ліків, вагу упаковки та об'єм упаковки, які зберігаються в БД. Також вхідними даними є значення ПІБ фармацевта та дати замовлення, які вводяться користувачем з клавіатури. Приклад вхідних даних, які зберігаються в БД наведено на рис Б.5.

Назва_ліків	Вага_упаковки	Розмір_упаковки
x1	150	36
x2	225	247
x3	223	215
x4	129	185
x5	10	112
x6	233	295
x7	243	45

Рисунок Б.5 – Інформація про ліки

Як можна побачити з рис Б.5 у першому стовпчику вказано назву ліків, у другому – вагу упаковки, а в третьому – розмір упаковки.

Вихідними даними є матриця відстаней, таблиця проходження оптимального маршруту, інформація про довжину стандартного маршруту проходження, інформація про довжину оптимального маршруту проходження, інформація про час стандартного маршруту проходження, інформація про час оптимального маршруту проходження.

ДОДАТОК В
Текст програми

В.1 Файл Form1.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;

namespace AutoPharm
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        int first_len = 0;
        int fFarmY = 1;
        int iPharm = 0;
        int jPharm = 1;
        int second_len = 0;
        int sFarmX = 3;
        int sFarmY = 1;
        int sFarmXinc = 3;
        int sFarmYinc = 1;
        int i1 = 0;
        int i2 = 0;
        int j1 = 0;
        int j2 = 0;
        int len = 0;
        int allPharmNum = 0;
        string tovname;
        string[,] allMas = new string[500, 20];
        string tov_name1 = "";
        string tov_name2 = "";
        int kolPharm = 1;
    }
}

```

```

int tov_weight = 0;
int tov_size = 0;
int tov_weightDB = 0;
int tov_sizeDB = 0;
int name_num = 1;
int PharmNow = 0;
double path = 0;
double time = 0;
int toStartPharm = 1;
double start_path = 0;
double start_time = 0;
private void Form1_Load(object sender, EventArgs e)
{
    Form2 dialog = new Form2();
    DialogResult result = dialog.ShowDialog();
    if (result == DialogResult.Yes)
    {
        label6.Text = dialog.getName();
    }
    else Close();
    string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
        "Data Source=AutoPharmDB.mdb; " +
        "Jet OLEDB:Database Password=passw";
    OleDbConnection connection = new OleDbConnection();
    connection.ConnectionString = ConnectionString;
    string sql = "SELECT Фармацевти.ПІБ_фармацевта, Замовлення.Дата_замовлення FROM
Фармацевти "+
        "INNER JOIN Замовлення ON Фармацевти.Код_фармацевта = Замовлення.Код_фармацевта
ORDER BY Замовлення.Код_замовлення;";
    OleDbCommand myCommand = new OleDbCommand(sql, connection);
    connection.Open();
    OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
    DataSet ds = new DataSet();
    da.Fill(ds, "Замовлення");
    dataGridView5.DataSource = ds.Tables["Замовлення"].DefaultView;
    connection.Close();
    connection = new OleDbConnection();
    connection.ConnectionString = ConnectionString;
    string sql1 = "SELECT Код_фармацевта FROM Фармацевти WHERE ПІБ_фармацевта = '" +
label6.Text + "'";
    OleDbCommand myCommand1 = new OleDbCommand(sql1, connection);
    connection.Open();

```

```

OleDbDataAdapter da1 = new OleDbDataAdapter(myCommand1);
DataSet ds1 = new DataSet();
da1.Fill(ds1, "Фармацевти");
dataGridView6.DataSource = ds1.Tables["Фармацевти"].DefaultView;
connection.Close();
}
private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        tovname = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
        tov_weightDB = Convert.ToInt32(dataGridView1.Rows[e.RowIndex].Cells[1].Value);
        tov_sizeDB = Convert.ToInt32(dataGridView1.Rows[e.RowIndex].Cells[2].Value);
    }
}
private void button1_Click(object sender, EventArgs e)
{
    tov_weight = tov_weight + tov_weightDB;
    tov_size = tov_size + tov_sizeDB;
    if ((tov_weight < 800) && (tov_size < 1500))
    {
        label7.Text = tov_weight.ToString();
        label12.Text = tov_size.ToString();
        button3.Enabled = true;
        if (tov_name2 != "")
        {
            tov_name1 = tov_name2;
        }
        if (tov_name1 == "")
        {
            tov_name1 = tovname;
        }
        else
        {
            tov_name2 = tovname;
        }
        if (tov_name1 != tov_name2)
        {

```

```

kolPharm++;
dataGridView4.RowCount = kolPharm;
dataGridView4.ColumnCount = 1;

dataGridView4.Rows[0].Cells[0].Value = 0;
if (kolPharm > 3)
{
    dataGridView4.Rows[name_num].Cells[0].Value = tovname;
    name_num++;
}
else if (kolPharm == 3)
{
    dataGridView4.Rows[name_num].Cells[0].Value = tovname;
    name_num++;
}
allPharmNum = 0;
for (int i = 0; i < 500; i++)
{
    for (int j = 0; j < 20; j++)
    {
        allMas[i, j] = "x" + allPharmNum.ToString();
        allPharmNum++;
    }
}
dataGridView2.RowCount = kolPharm;
dataGridView2.ColumnCount = kolPharm;

for (int i = 0; i < 500; i++)
{
    for (int j = 0; j < 20; j++)
    {
        if (tov_name2 != "")
        {
            if (allMas[i, j] == tov_name1)
            {
                i1 = i;
                j1 = j;
            }
            else if (allMas[i, j] == tov_name2)
            {
                i2 = i;

```

```

        j2 = j;
    }
}
else
{
    i1 = 0;
    j1 = 0;
    len = 0;
}
}
}
if (tov_name2 != "")
{
    len = 5 * (Math.Abs((i2 - i1)) + Math.Abs((j2 - j1)));
}
if ((len > 0) && (iPharm < kolPharm) && (jPharm < kolPharm))
{
    dataGridView2.Rows[iPharm].Cells[jPharm].Value = len;
    dataGridView2.Rows[jPharm].Cells[iPharm].Value = len;

    start_path = start_path + len;

    iPharm++;
    jPharm++;
}
if ((len > 0) && (fFarmY < kolPharm))
{
    first_len = 5 * (i2 + j2);

}
if ((len > 0) && (fFarmY < kolPharm))
{
    dataGridView2.Rows[0].Cells[fFarmY].Value = first_len;
    dataGridView2.Rows[fFarmY].Cells[0].Value = first_len;

    fFarmY++;
}
}
else
{
    MessageBox.Show("Помилка при додаванні ліків!");
    tov_weight = tov_weight - tov_weightDB;
}

```

```

        tov_size = tov_size - tov_sizeDB;
        tov_weightDB = 0;
        tov_sizeDB = 0;
        label7.Text = tov_weight.ToString();
        label12.Text = tov_size.ToString();
    }
}
else
{
    MessageBox.Show("Габарити ліків за межами діапазону!");
    tov_weight = tov_weight - tov_weightDB;
    tov_size = tov_size - tov_sizeDB;
}
}

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        tovname = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
        tovname = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
        tov_weightDB = Convert.ToInt32(dataGridView1.Rows[e.RowIndex].Cells[1].Value);
        tov_sizeDB = Convert.ToInt32(dataGridView1.Rows[e.RowIndex].Cells[2].Value);
    }
}

private void button3_Click(object sender, EventArgs e)
{
    if (kolPharm > 2)
    {
        string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
            "Data Source=AutoPharmDB.mdb; " +
            "Jet OLEDB:Database Password=passw";
        OleDbConnection connection = new OleDbConnection();
        connection.ConnectionString = ConnectionString;
        OleDbCommand myCommand;
        string sql;
        sql = " INSERT INTO Замовлення ( [Код_фармацевта], [Дата_замовлення]) VALUES (" +
dataGridView6.Rows[0].Cells[0].Value + ", '" + DateTime.Now.ToString("dd/MM/yyyy") + "')";
        myCommand = new OleDbCommand(sql, connection);
        connection.Open();
        myCommand.ExecuteNonQuery();
        connection.Close();
        connection = new OleDbConnection();
    }
}

```

```

connection.ConnectionString = ConnectionString;
sql = "SELECT  Фармацевти.ПІБ_фармацевта,  Замовлення.Дата_замовлення  FROM
Фармацевти " +
      "INNER JOIN Замовлення ON Фармацевти.Код_фармацевта = Замовлення.Код_фармацевта
ORDER BY Замовлення.Код_замовлення;";
myCommand = new OleDbCommand(sql, connection);
connection.Open();
OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
DataSet ds = new DataSet();
da.Fill(ds, "Замовлення");
dataGridView1.DataSource = ds.Tables["Замовлення"].DefaultView;
connection.Close();

tabControl1.Visible = false;
panel1.Visible = true;
bool checkPharm(int keyPharm, int[] masPharm, int kilk)
{
    for (int j = 0; j < kilk; j++)
        if (masPharm[j] == keyPharm)
            return false;
    return true;
}
string[] nameMas = new string[kolPharm - 1];
for (int i = 0; i < kolPharm - 1; i++)
{
    nameMas[i] = dataGridView4.Rows[i].Cells[0].Value.ToString();
    dataGridView2.Columns[i].HeaderText = nameMas[i];
}
for (int i = 0; i < kolPharm; i++)
{
    for (int j = 0; j < kolPharm; j++)
    {
        if ((sFarmX < kolPharm) && (sFarmY < kolPharm))
        {

            if          (Convert.ToInt32(dataGridView2.Rows[0].Cells[sFarmY].Value)          !=
Convert.ToInt32(dataGridView2.Rows[0].Cells[sFarmX].Value))
            {
                second_len = Math.Abs(Convert.ToInt32(dataGridView2.Rows[0].Cells[sFarmY].Value)
- Convert.ToInt32(dataGridView2.Rows[0].Cells[sFarmX].Value));
            }
            else second_len = 2 * Convert.ToInt32(dataGridView2.Rows[0].Cells[sFarmX].Value);

```

```

dataGridView2.Rows[sFarmX].Cells[sFarmY].Value = second_len;
dataGridView2.Rows[sFarmY].Cells[sFarmX].Value = second_len;
sFarmX++;
sFarmYinc++;

if (dataGridView2.Rows[dataGridView2.Rows.Count - 1].Cells[sFarmY].Value != null)
{
    sFarmY++;
    sFarmYinc = sFarmY;

    sFarmXinc++;
    sFarmX = sFarmXinc;
}
}
}
for (int i = 0; i < kolPharm; i++)
{
    for (int j = 0; j < kolPharm; j++)
    {
        if (i == j)
        { dataGridView2.Rows[i].Cells[j].Value = 0; }
    }
}
int[,] finmas = new int[kolPharm - 1, kolPharm - 1];

for (int i = 0; i < kolPharm - 1; i++)
{
    for (int j = 0; j < kolPharm - 1; j++)
    {
        finmas[i, j] = Convert.ToInt32(dataGridView2.Rows[i].Cells[j].Value);
    }
}
dataGridView2.RowCount = kolPharm - 1;
dataGridView2.ColumnCount = kolPharm - 1;
dataGridView3.RowCount = kolPharm - 1;
dataGridView3.ColumnCount = 2;
for (int i = 0; i < kolPharm - 1; i++)
{
    for (int j = 0; j < kolPharm - 1; j++)

```

```

        {
            dataGridView2.Rows[i].Cells[j].Value = finmas[i, j];
        }
    }
    int[] routePharm = new int[kolPharm - 1];
    for (int i = 0; i < kolPharm - 1; i++)
    {
        routePharm[i] = -1;
    }
    routePharm[0] = 0;
    for (int i = 1; i < kolPharm - 1; i++)
    {
        toStartPharm++;
        int min = int.MaxValue, min_Pharm = 0;
        for (int j = 0; j < kolPharm - 1; j++)
        {
            if (checkPharm(j, routePharm, kolPharm - 1) && finmas[PharmNow, j] < min &&
finmas[PharmNow, j] > 0)
            {
                min = finmas[PharmNow, j];
                min_Pharm = j;
            }
        }
        path += min;
        routePharm[i] = min_Pharm;
        dataGridView3.Rows[i - 1].Cells[0].Value = nameMas[PharmNow] + " -> " +
nameMas[routePharm[i]];
        dataGridView3.Rows[i - 1].Cells[1].Value = "відстань " + min + " см";
        PharmNow = routePharm[i];
    }
    if (toStartPharm == kolPharm - 1)
    {
        dataGridView3.Rows[toStartPharm - 1].Cells[0].Value = nameMas[PharmNow] + " -> " + "0";
        dataGridView3.Rows[toStartPharm - 1].Cells[1].Value = "відстань " + finmas[0, PharmNow] +
" см";
    }
    path += finmas[0, PharmNow];
    time = path / 250 + 0.5 * (kolPharm - 1);
    start_path = start_path + Convert.ToDouble(dataGridView2.Rows[kolPharm - 2].Cells[0].Value);
    start_time = start_path / 250 + 0.5 * (kolPharm - 1);
    label4.Text = "Загальна довжина маршруту: " + start_path + " см";
    label3.Text = "Час проходження маршруту: " + start_time + " с";
    label15.Text = "Загальна довжина маршруту: " + path + " см";
    label16.Text = "Час проходження маршруту: " + time + " с";

```

```

        button3.Enabled = false;
    }
    else MessageBox.Show("Додайте ліки в замовлення!");
}
private void button2_Click(object sender, EventArgs e)
{

    string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
        "Data Source=AutoPharmDB.mdb; " +
        "Jet OLEDB:Database Password=passw";
    OleDbConnection connection = new OleDbConnection();
    connection.ConnectionString = ConnectionString;
    string sql = "SELECT Назва_ліків, Вага_упаковки, Розмір_упаковки FROM Ліки ORDER BY
Код_ліків";

    OleDbCommand myCommand = new OleDbCommand(sql, connection);
    connection.Open();
    OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
    DataSet ds = new DataSet();
    da.Fill(ds, "Ліки");
    dataGridView1.DataSource = ds.Tables["Ліки"].DefaultView;
    connection.Close();
    button1.Enabled = true;
    button1.PerformClick();
    button4.Enabled = true;
    button2.Enabled = false;
}
private void button4_Click(object sender, EventArgs e)
{
    allPharmNum = 0;
    dataGridView1.DataSource = null;
    tovname = null;
    tov_name1 = "";
    tov_name2 = "";
    dataGridView2.Rows.Clear();
    dataGridView2.Columns.Clear();
    dataGridView3.Rows.Clear();
    dataGridView3.Columns.Clear();
    dataGridView4.Rows.Clear();
    dataGridView4.Columns.Clear();
    kolPharm = 1;
    fFarmY = 1;
    iPharm = 0;

```

```

jPharm = 1;
sFarmX = 3;
sFarmY = 1;
sFarmXinc = 3;
sFarmYinc = 1;
i1 = 0;
i2 = 0;
j1 = 0;
j2 = 0;
len = 0;
first_len = 0;
second_len = 0;
tov_weight = 0;
tov_size = 0;
tov_weightDB = 0;
tov_sizeDB = 0;
label7.Text = tov_weight.ToString();
label12.Text = tov_size.ToString();
PharmNow = 0;
path = 0;
time = 0;
toStartPharm = 1;
start_path = 0;
start_time = 0;
name_num = 1;
button4.Enabled = false;
button1.Enabled = false;
button3.Enabled = false;
button2.Enabled = true;
for (int i = 0; i < 500; i++)
{
    for (int j = 0; j < 20; j++)
    {
        allMas[i, j] = "";
    }
}
}

private void button5_Click(object sender, EventArgs e)
{
    panel1.Visible = false;
    tabControl1.Visible = true;
    button4.PerformClick();
}

```

```

string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
    "Data Source=AutoPharmDB.mdb; " +
    "Jet OLEDB:Database Password=passw";
OleDbConnection connection = new OleDbConnection();
connection.ConnectionString = ConnectionString;

string sql = "SELECT Фармацевти.ПІБ_фармацевта, Замовлення.Дата_замовлення FROM
Фармацевти " +
    "INNER JOIN Замовлення ON Фармацевти.Код_фармацевта = Замовлення.Код_фармацевта
ORDER BY Замовлення.Код_замовлення;";
OleDbCommand myCommand = new OleDbCommand(sql, connection);
connection.Open();

OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
DataSet ds = new DataSet();
da.Fill(ds, "Замовлення");
dataGridView5.DataSource = ds.Tables["Замовлення"].DefaultView;
connection.Close();
}

private void button6_Click(object sender, EventArgs e)
{
    this.Close();
}

private void button7_Click(object sender, EventArgs e)
{
    if (maskedTextBox1.MaskCompleted == true)
    {
        string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
            "Data Source=AutoPharmDB.mdb; " +
            "Jet OLEDB:Database Password=passw";
        OleDbConnection connection = new OleDbConnection();
        connection.ConnectionString = ConnectionString;
        string orderDate = maskedTextBox1.Text;
        string sql = "SELECT Фармацевти.ПІБ_фармацевта, Замовлення.Дата_замовлення FROM
Фармацевти " +
            "INNER JOIN Замовлення ON Фармацевти.Код_фармацевта = Замовлення.Код_фармацевта "
+
            "WHERE Дата_замовлення Like '" + orderDate + "' ORDER BY Замовлення.Код_замовлення";

        OleDbCommand myCommand = new OleDbCommand(sql, connection);
        connection.Open();
    }
}

```

```

        OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
        DataSet ds = new DataSet();
        da.Fill(ds, "Замовлення");
        dataGridView5.DataSource = ds.Tables["Замовлення"].DefaultView;
        connection.Close();
        maskedTextBox1.Text = "";
    }
    else MessageBox.Show("Введіть дату замовлення!");
}

private void button9_Click(object sender, EventArgs e)
{
    string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
        "Data Source=AutoPharmDB.mdb; " +
        "Jet OLEDB:Database Password=passw";
    OleDbConnection connection = new OleDbConnection();
    connection.ConnectionString = ConnectionString;

    string sql = "SELECT Фармацевти.ПІБ_фармацевта, Замовлення.Дата_замовлення FROM
Фармацевти " +
        "INNER JOIN Замовлення ON Фармацевти.Код_фармацевта = Замовлення.Код_фармацевта
ORDER BY Замовлення.Код_замовлення;";
    OleDbCommand myCommand = new OleDbCommand(sql, connection);
    connection.Open();

    OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
    DataSet ds = new DataSet();
    da.Fill(ds, "Замовлення");
    dataGridView5.DataSource = ds.Tables["Замовлення"].DefaultView;
    connection.Close();
    maskedTextBox1.Text = "";
}

private void button8_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "")
    {
        string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
            "Data Source=AutoPharmDB.mdb; " +
            "Jet OLEDB:Database Password=passw";
        OleDbConnection connection = new OleDbConnection();
        connection.ConnectionString = ConnectionString;
        string orderPIB = textBox1.Text;
    }
}

```

```

        string sql = "SELECT Фармацевти.ПІБ_фармацевта, Замовлення.Дата_замовлення FROM
Фармацевти " +
        "INNER JOIN Замовлення ON Фармацевти.Код_фармацевта = Замовлення.Код_фармацевта "
+
        "WHERE ПІБ_фармацевта Like '" + orderPIB + "' ORDER BY Замовлення.Код_замовлення";

        OleDbCommand myCommand = new OleDbCommand(sql, connection);
        connection.Open();
        OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
        DataSet ds = new DataSet();
        da.Fill(ds, "Замовлення");
        dataGridView5.DataSource = ds.Tables["Замовлення"].DefaultView;
        connection.Close();
        textBox1.Text = "";
    }
    else MessageBox.Show("Введіть ПІБ фармацевта!");
}

private void button10_Click(object sender, EventArgs e)
{
    string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
        "Data Source=AutoPharmDB.mdb; " +
        "Jet OLEDB:Database Password=passw";
    OleDbConnection connection = new OleDbConnection();
    connection.ConnectionString = ConnectionString;
    string sql = "SELECT Фармацевти.ПІБ_фармацевта, Замовлення.Дата_замовлення FROM
Фармацевти " +
        "INNER JOIN Замовлення ON Фармацевти.Код_фармацевта = Замовлення.Код_фармацевта
ORDER BY Замовлення.Код_замовлення";
    OleDbCommand myCommand = new OleDbCommand(sql, connection);
    connection.Open();
    OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
    DataSet ds = new DataSet();
    da.Fill(ds, "Замовлення");
    dataGridView5.DataSource = ds.Tables["Замовлення"].DefaultView;
    connection.Close();
    textBox1.Text = "";
}
}
}

```

В.2 Файл Form2.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
namespace AutoPharm
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }
        public string getName()
        { return dataGridView1[1, 0].Value.ToString(); }
        private void Form2_Load(object sender, EventArgs e)
        {
            textBox2.PasswordChar = '*';
        }
        private void button1_Click(object sender, EventArgs e)
        {
            string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " +
                "Data Source=AutoPharmDB.mdb; " +
                "Jet OLEDB:Database Password=passw";
            OleDbConnection connection = new OleDbConnection();
            connection.ConnectionString = ConnectionString;
            string emp_name = textBox1.Text;
            string emp_pass = textBox2.Text;
            string sql = "SELECT * FROM Фармацевти WHERE ПИБ_фармацевта = '" + emp_name + "' and
Пароль_фармацевта = '" + emp_pass + "'";

            OleDbCommand myCommand = new OleDbCommand(sql, connection);
            connection.Open();

```

```
OleDbDataAdapter da = new OleDbDataAdapter(myCommand);
DataSet ds = new DataSet();
da.Fill(ds, "Фармацевти");
dataGridView1.DataSource = ds.Tables["Фармацевти"].DefaultView;
connection.Close();
if (dataGridView1[0, 0].Value != null)
{
    button3.PerformClick();
}
else MessageBox.Show("Введено невірний логін та/або пароль!");
}

private void button2_Click(object sender, EventArgs e)
{
    Environment.Exit(0);
}
}
}
```

ДОДАТОК Д
Слайди презентації

Національний університет «Запорізька політехніка»

ПРОГРАМНА РЕАЛІЗАЦІЯ ВИРІШЕННЯ ЗАДАЧІ КОМІВОЯЖЕРА ДЛЯ АВТОМАТУ ВИДАЧІ ЛІКІВ

Виконав:

студент групи КНТ-118сп

А.Л. Головченко

Керівник:

Доцент, к.т.н,

О.О. Степаненко

Рисунок Д.1 – Слайд № 1

2

Мета, об'єкт, предмет дослідження

- **Об'єкт дослідження** – автоматизовані системи видачі ліків в аптеці.
- **Предмет дослідження** – методи пошуку оптимального маршруту для маніпулятора автомата видачі ліків.
- **Мета роботи** – створення програмної системи для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.
- **Актуальність та проблематика роботи** полягає в тому, що на сьогоднішній день у багатьох аптеках використовуються сучасні автомати для видачі ліків, які містять в собі велику кількість ліків і мають можливість обробляти замовлення великого обсягу. Тому для скорочення часу доставки великої кількості ліків в замовленні до покупця виникла необхідність розробки методів пошуку оптимального маршруту для маніпулятора автомата видачі ліків.

Рисунок Д.2 – Слайд № 2

Постановка завдання до роботи

Для виконання поставленої мети потрібно виконати наступні завдання:

- – створити метод для додавання інформації про замовлення в базу даних;
- – створити метод для авторизації користувачів;
- – створити метод для визначення загальної кількості ліків в замовленні;
- – створити метод для побудови матрицю відстаней для полиць з ліками, які будуть включені в маршрут проходження маніпулятора автомату видачі ліків;
- – створити метод для знаходження оптимального маршруту, який проходить маніпулятор;
- – створити метод для розрахунку стандартного маршруту проходження;
- – створити метод для розрахунку загального часу проходження оптимального маршруту;
- – виконати розрахунок загального часу проходження стандартного маршруту.

Рисунок Д.3 – Слайд № 3

Порівняльна характеристика існуючих методів

Можливості	Алгоритм повного перебору	Метод мурашиних колоній	Генетичний алгоритм	Штучна нейронна мережа	Метод гілок та границь	Жадібний алгоритм
Простота реалізації	+	–	+	–	–	+
Залежність витрат часу на рішення від розмірності задачі	+	+	–	+	+	–
Простота кодування інформації	+	–	+	–	–	+
Точність рішення	+	+	–	+	+	–
Адаптивність до різних варіантів задач	+	–	+	+	+	–

Рисунок Д.4 – Слайд № 4

Порівняльна характеристика існуючих мов програмування

Критерії порівняння	R	C++	Python	Java	C#
Мова створювалася спеціально для аналізу даних	+	–	–	–	–
Ліцензія	GNU GPL	–	Python Software Foundation License	GNU GPL	.NET Core CLR: MIT / X11
Незалежність від платформи	+	+	+	+	–
Можливість обробляти велику кількість даних	+	+	+	+	+
Потужні інструменти для візуалізації даних	+	–	+	–	–
Простота вивчення та розробки	–	+	+	+	+

Рисунок Д.5 – Слайд №5

Порівняльна характеристика існуючих середовищ розробки

Критерії порівняння	Microsoft Visual Studio	Project Rider	Eclipse
Ліцензія	freemium	proprietary	Eclipse Public License
Можливість використання в різних системах	+	+	+
Підтримка контролю версій	+	+	+
Наявність додаткових модулів	+	+	+
Простота в освоєнні	–	+	–
Можливість безкоштовного використання	+	–	+
Надійність	+	–	–
Зручний інтерфейс	–	+	+
Використання великої кількості апаратних ресурсів	–	–	+

Рисунок Д.6 – Слайд № 6

ER-діаграма БД

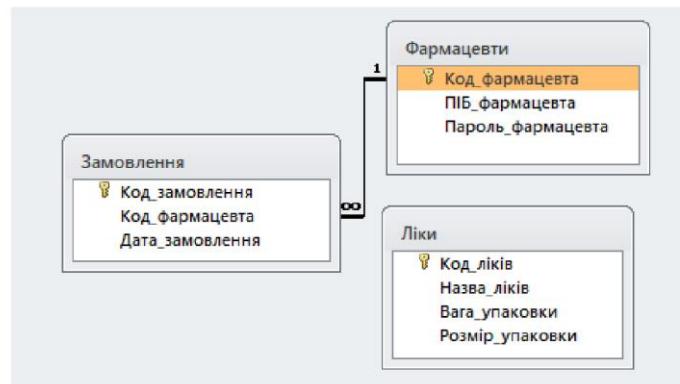


Рисунок Д.7 – Слайд № 7

Метод пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків

Для реалізації методу пошуку оптимального маршруту використовувались такі формули:

Формула розрахунку відстані між координатами:

$$S = 5 \cdot (|i_2 - i_1| + |j_2 - j_1|),$$

де i_1 та j_1 – координати першої точки;

i_2 та j_2 – координати другої точки.

Формула розрахунку довжини оптимального маршруту:

$$S_{\min} = \sum_{j=m}^p S_j,$$

де p – кількість мінімальних точок в маршруті;

j – індекс суми;

m – початкова точка проходження маршруту;

S_j – відстань між найближчими полицями з ліками в маршруті.

Рисунок Д.8 – Слайд № 8

Метод пошуку оптимального маршруту проходження маніпулятора автомату видачі ліків

Формула розрахунку часу проходження оптимального маршруту:

$$t_{\min} = \frac{S_{\min}}{V},$$

де $S_{\min}$ – загальна довжина оптимального маршруту проходження маніпулятора;

V – швидкість маніпулятора.

Рисунок Д.9 – Слайд № 9

Схема структури ПП



Рисунок Д.10 – Слайд № 10

Інтерфейсні вікна програми

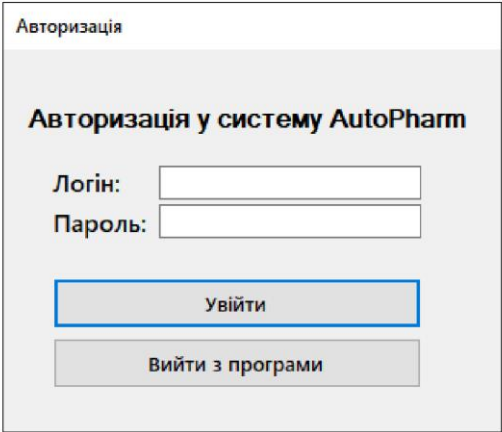


Рисунок Д.11 – Слайд № 11

Інтерфейсні вікна програми

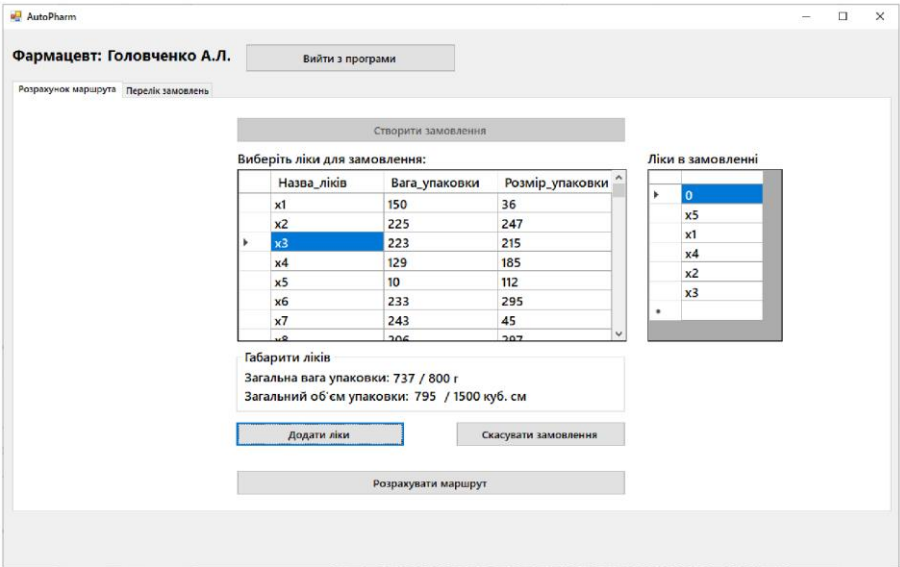


Рисунок Д.12 – Слайд № 12

Інтерфейсні вікна програми

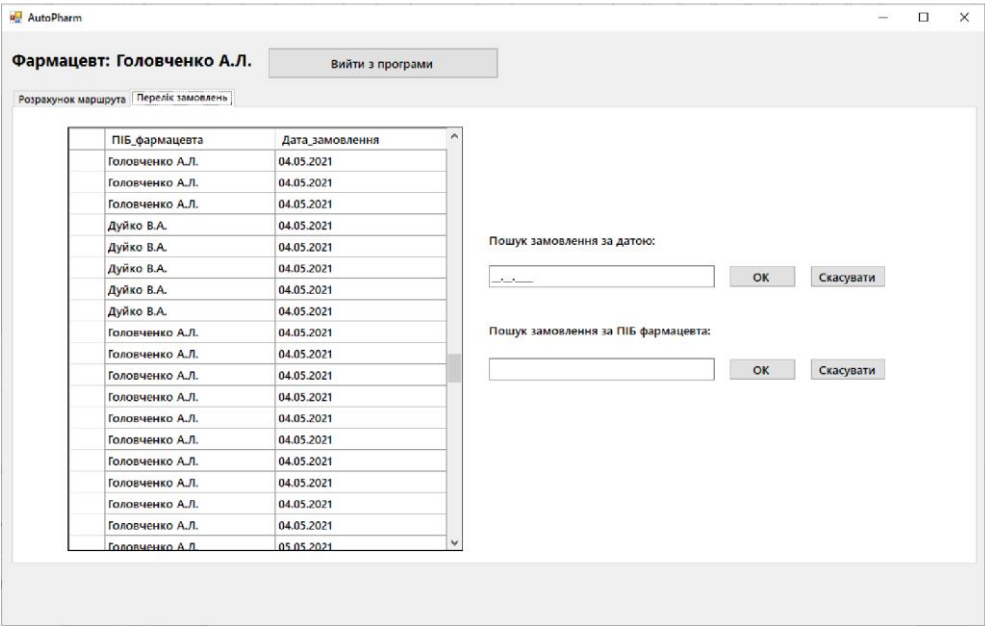


Рисунок Д.13 – Слайд № 13

Інтерфейсні вікна програми

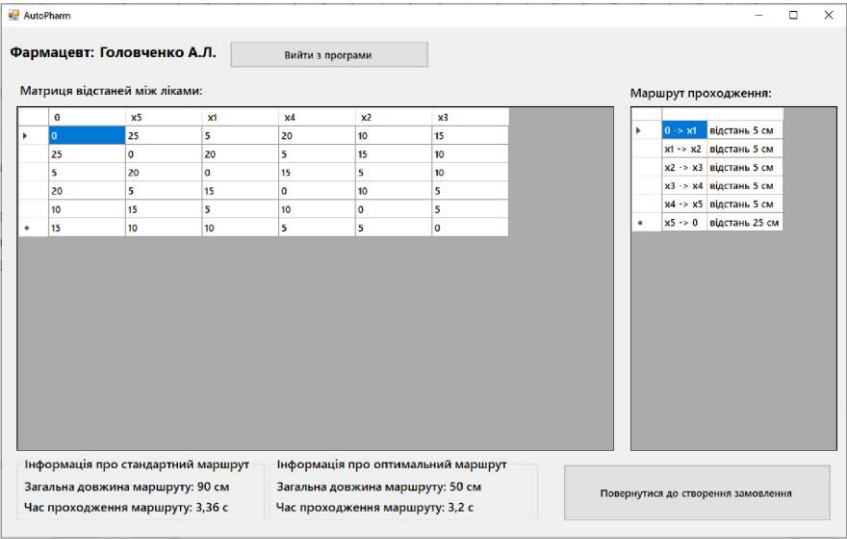


Рисунок Д.14 – Слайд № 14

Висновок

Під час виконання роботи було виконано дослідження предметної області, розглянуто та визначено основні поняття, які пов'язані з обраною предметною областю.

Розроблено метод пошуку оптимального маршруту за допомогою вирішення задачі комівояжера.

В результаті було створено програмну систему для оптимізації роботи автомата видачі ліків шляхом пошуку оптимального маршруту за допомогою вирішення задачі комівояжера. Створена програмна система дозволяє оптимізувати роботу автомата видачі ліків через скорочення часу на доставку ліків до фармацевта, а отже її можна використовувати в аптеках з такими автоматами.

Рисунок Д.15 – Слайд № 15