**Форма № 24**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
**Національний університет «Запорізька політехніка»**

Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри )

# Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ДОСТАВКИ ЛІКІВ РОБОТОМ-ФАРМАЦЕВТОМ
SOFTWARE IMPLEMENTATION OF MEDICATION DELIVERY BY A PHARMACY ROBOT

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

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Косенков І.С.
(прізвище та ініціали)
Керівник Олійник А.О.
(прізвище та ініціали)
Рецензент Гофман Є.О.
(прізвище та ініціали)

2022

**Форма № 25**

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**
“____” _____2022року

# З А В Д А Н Н Я

## НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Косенкова Ігоря Сергійовича
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація методів доставки ліків роботом-фармацевтом. Software Implementation of Medication Delivery by a Pharmacy Robot
керівник проєкту (роботи) Олійник А. О., д.т.н., проф.,
( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “ 27 ” квітня 2022 року № 102
2. Строк подання студентом проєкту (роботи) 7 червня 2022 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 познак. Вступ. 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Засоби програмної реалізації програмного продукту. 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 | Олійник А.О., проф. | | |
| Нормоконтроль | Липовець М.В., асистент | | |

7. Дата видачі завдання “ 04 ” квітня 2022 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. | 1 тиждень | Розділ 1 |
| 3 | Вибір мови програмування та інших технологій | 2 тиждень | Розділ 2 |
| | розробки. | | |
| 4 | Розробка архітектури програми. | 2 тиждень | Розділ 3 |
| 5 | Розробка програми. | 3-4 тижні | Розділ 3, 4 |
| 6 | Тестування та експериментальне дослідження | 5 тиждень | Розділ 4 |
| | програмного забезпечення. | | |
| 7 | Оформлення пояснювальної записки та документів | 6 тиждень | Додатки |
| | до неї. | | |
| 8 | Нормоконтроль та рецензування. | 7 тиждень | |
| 9 | Захист роботи. | 8 тиждень | |

**Студент** _____ Косенков І. С.
( підпис ) (прізвище та ініціали)

**Керівник проєкту (роботи)** _____ Олійник А.О.
( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 137 с., 72 рис., 17 табл., 4 дод., 16 джерел.

СИСТЕМА ДОСТАВКИ ЛІКІВ, ЕВОЛЮЦІЙНИЙ АЛГОРИТМ, КОМБІНАТОРНИЙ АНАЛІЗ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ШЛЯХ ДОСТАВКИ, C#.

Об’єкт дослідження – процес прогнозування руху руки маніпулятора робота-фармацевта за ліками.

Предмет дослідження – методи прогнозування результатів процесу доставки ліків роботом-фармацевтом до користувача.

Мета роботи – програмна реалізація методів доставки ліків роботом-фармацевтом до користувача.

Актуальність та проблематика роботи полягає в тому, що через пандемію та воєнні дії на території України збільшився попит на ліки. Внаслідок чого збільшилися черги за ліками та завантаженість як самих фармацевтів так і роботів-фармацевтів. У зв’язку з цим збільшився попит на ліки тому потрібно оптимізувати роботу роботів-фармацевтів.

В роботі наведено аналіз та дослідження методів доставки ліків роботом-фармацевтом. Розроблено шість методів доставки ліків роботом фармацевтом. Розроблені методи вирішують питання оптимізації доставки одразу багатьох препаратів з урахуванням розташування одразу всіх вибраних препаратів до користувача.

Принципова відмінність створених методів від існуючих полягає у тому, що запропоновані методи використовують передові алгоритми побудування шляху доставки та мають максимальну адаптацію до специфіки розв'язуваної задачі.

Матеріали, методи та технічні засоби: структурне та об’єктно-орієнтоване програмування, мова програмування C#, персональний комп’ютер з процесором Intel Pentium G4620 під управлінням операційної системи Microsoft Windows 10.

Результати. Розроблено програмний продукт, який призначений для вирішення завдання доставки ліків роботом-фармацевтом, а саме розміщення ліків на складі, доставка декількох препаратів, та зручне відображення статистики використання роботу.

Висновки. Розроблено програмну систему для вирішення завдання доставки ліків роботом-фармацевтом, а саме розміщення ліків на складі, доставка декількох препаратів, та зручне відображення статистики використання роботу.

Галузь використання – видача ліків роботом-фармацевтом в аптеці.

# ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 137 pages, 72 figures, 17 tables, 4 appendixes, 60 sources.

DRUG DELIVERY SYSTEM, EVOLUTIONARY ALGORITHM, COMBINATION ANALYSIS, GENETIC ALGORITHM, DELIVERY ROUTE, C#.

The object of research is the process of predicting the movement of the hand of the manipulator robot pharmacist on drugs.

The subject of research - methods of predicting the results of the process of drug delivery by a robot pharmacist to the user.

The purpose of the work is the software implementation of methods of drug delivery by a robot pharmacist to the user.

The urgency and problems of the work is that due to the pandemic and hostilities in Ukraine, the demand for drugs has increased. As a result, the queues for drugs and the workload of both pharmacists and robot pharmacists have increased. Due to this, the demand for drugs has increased, so it is necessary to optimize the work of robotic pharmacists.

The paper presents an analysis and study of methods of drug delivery by a robot pharmacist. Six methods of drug delivery by a robot pharmacist have been developed. The developed methods solve the problem of optimizing the delivery of many drugs at once, taking into account the location of all selected drugs to the user.

The fundamental difference between the created methods and the existing ones is that the proposed methods use advanced algorithms for constructing the delivery path and have the maximum adaptation to the specifics of the problem to be solved.

Materials, methods and technical means: structural and object-oriented programming, C # programming language, personal computer with Intel Pentium G4620 processor running Microsoft Windows 10 operating system.

Results. A software product has been developed to solve the problem of drug delivery by a robot pharmacist, namely the placement of drugs in a warehouse, delivery of several drugs, and convenient display of statistics on the use of the work.

Conclusions. A software system has been developed to solve the problem of drug delivery by a robot pharmacist, namely the placement of drugs in a warehouse, delivery of several drugs, and convenient display of statistics on the use of work.

Field of use - dispensing of drugs by a robot pharmacist in a pharmacy.

# ЗМІСТ

С.
Перелік скорочень та умовних познак ..... 10
Вступ ..... 11
1 Аналіз предметної області ..... 12
1.1 Різновиди роботів-фармацевтів ..... 12
1.2 Огляд існуючих методів ..... 18
1.3 Постановка завдання до роботи ..... 23
1.4 Висновки за розділом ..... 25
2 Вибір та обґрунтування структури системи, яка проєктується ..... 26
2.1 Вибір інструментарію розробки програмного продукту ..... 26
2.2 Структурна схема програмного продукту ..... 34
2.3 Висновки за розділом ..... 38
3 Засоби програмної реалізації програмного продукту ..... 39
3.1 Функціональні вимоги до програмного продукту ..... 39
3.2 Опис архітектури та обраних технологій ..... 42
3.3 Порівняння шляху доставки різними алгоритмами ..... 45
3.4 Висновки за розділом ..... 68
4 Керівництво програміста ..... 69
4.1 Призначення та умови застосування програми ..... 69
4.2 Умови, необхідні для виконання програми ..... 69
4.3 Характеристика програми ..... 70
4.4. Звертання програми ..... 71
4.5 Вхідні та вихідні дані ..... 71
4.6 Повідомлення ..... 72
4.7 Висновки за розділом ..... 72
5 Керівництво оператора ..... 73
5.1 Призначення програми ..... 73
5.2 Умови виконання програми ..... 73
5.3 Запуск програми ..... 73

5.4 Виконання програми ..... 74
5.5 Повідомлення оператору ..... 78
5.6 Висновок ..... 78
Висновки ..... 80
Перелік використаних джерел ..... 81
Додаток А Технічне завдання ..... 83
Додаток Б Опис програмного продукту ..... 89
Додаток В Лістинги файлів роботи ..... 102
Додаток Д Слайди презентації ..... 130

# ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IDE – Integrated Development Environment;

БД – база даних;

ГА – генетичний алгоритм;

ПЗ – програмне забезпечення;

ПП – програмний продукт;

СКБД – система керування базами даних.

# ВСТУП

У зв'язку з високою інформатизацією суспільства в повсякденну роботу різних сфер промисловості та підприємництва все частіше вводиться автоматизована та роботизована техніка. Фармацевтичний бізнес не став винятком. Для підприємств фармації сьогодні більш ніж в іншій області, актуально впровадження інформаційних технологій і робототехніки.

Фармацевтичний бізнес - це завжди великий обіг медикаментів і їх облік. Грамотне ведення аптечних роботів допомагають підвищити рентабельність і знизити ризики можливих помилок, а також спростити сам процес роботи, але їх проблемою є неоптимальний вибір шляху пошуку ліків у своїй системі.

Розв'язання даної проблеми є впровадження нових алгоритмів пошуку та доставлення ліків до термінала.

Роботизований аптечний склад або робо склад, являє собою комплекс, в який можуть входити: шафи для зберігання лікарських засобів, роботизовані системи приймання ліків, механізми для переміщення лікарських засобів (маніпулятори, транспортери), електронні системи, які здійснюють управління складами.

Спосіб обслуговування за принципом роботизованої аптеки викликає у споживачів інтерес, підтримує увагу постійних клієнтів і залучає нових покупців в аптеку. Комплекс переваг роботів для аптек створює всі передумови для підвищення числа постійних покупців і збільшення товарообігу.

Мета роботи – програмна реалізація методів доставки ліків роботом-фармацевтом до користувача.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Актуальність та проблематика роботи полягає в тому, що через пандемію та воєнні дії на території України збільшився попит на ліки. Внаслідок чого збільшилися черги за ліками та завантаженість як самих фармацевтів так і роботів-фармацевтів. У зв’язку з цим збільшився попит на ліки тому потрібно оптимізувати роботу роботів-фармацевтів.

В першому підрозділі наведено види існуючих роботів-фармацевтів, опис існуючих моделей від різних компаній та наведено їх порівняльну характеристику.

## 1.1 Різновиди роботів-фармацевтів

а) Consis B2 (рис. 1.1):

1) термін видачі 1 упаковки дорівнює 7 секундам;
2) площа 3.3 м²;
3) довжина 2650 см;
4) ширина 1250 см;
5) висота 2250 см;
6) вміщається 7200 упаковок [1];

Рисунок 1.1 - Consis B2 [1]

б) Consis B4 (рис. 1.2):

1) термін видачі 1 упаковки дорівнює 7 секундам;

2) площа 4.8 м²;

3) довжина 3850 см;

4) ширина 1250 см;

5) висота 2250 см;

6) вміщається 10800 упаковок [1];

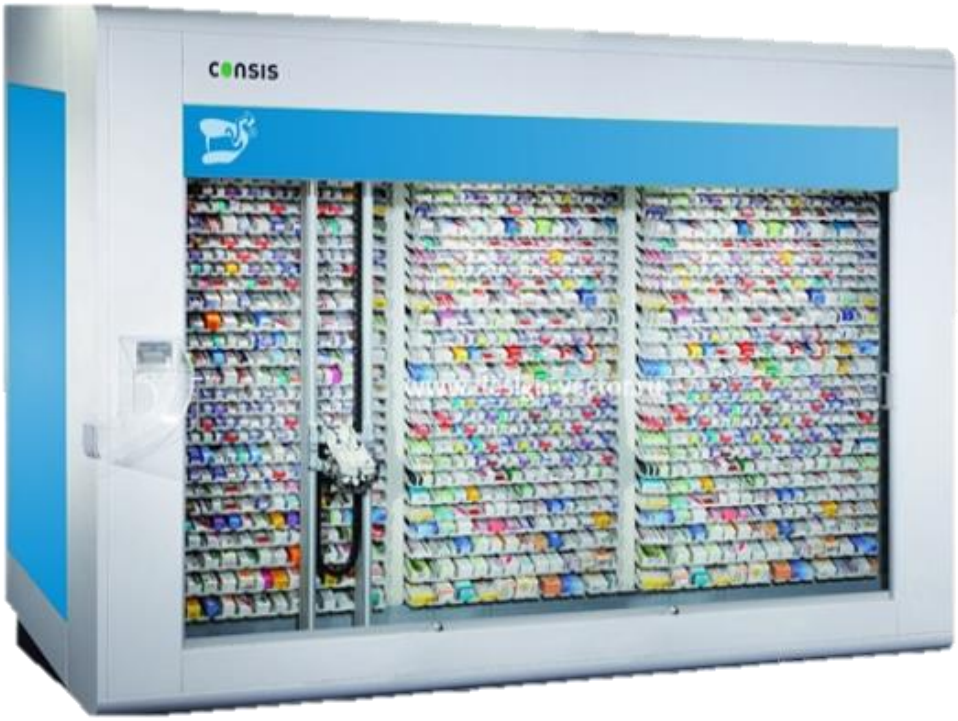



Рисунок 1.2 - Consis B4 [1]

в) PharmBot (рис. 1.3):

1) термін видачі 1 упаковки дорівнює 7 секундам;

2) площа 8.3 м²;

3) довжина 3070 см;

4) ширина 2700 см;

5) висота 1900 см;

6) вміщається 11000 упаковок [2];

Рисунок 1.3 – PharmBot [2]

г) ISO-FIXU (рис. 1.4):

1) термін видачі 1 упаковки дорівнює 9 секундам;
2) площа 16.65 м²;
3) довжина 5550 см;
4) ширина 3000 см;
5) висота 2000 см;
6) вміщається 32000 упаковок [3].

Рисунок 1.4 - ISO-FIXU [3]

Для аналізу найкращих показників була зроблена таблиця 1.1.

Таблиця 1.1 – Порівняння роботів

| | Consis B2 | Consis B4 | PharmBot | ISO-FIXU |
|---|---|---|---|---|
| Термін видачі 1 упаковки | 7 | 7 | 7 | 9 |
| Площа | 3.3 м² | 4.8 м² | 8.3 м² | 16.65 м² |
| Довжина | 2650 см | 3850 см | 3070 см | 5550 см |
| Ширина | 1250 см | 1250 см | 2700 см | 3000 см |
| Висота | 2250 см | 2250 см | 1900 см | 2000 см |
| Місткість | 7200 упаковок | 10800 упаковок | 11000 упаковок | 32000 упаковок |

Під час проєктування алгоритму за основу будуть взяті данні роботу PharmBot, очікуваний термін видачі 1 упаковки буде 6 секунд.

### 1.1.1 Автоматичний диспенсер

Загалом, роботи - диспенсери - це велика аптечна шафа (АС), де зберігаються ліки, оснащені інноваційними механічними приводами, що ведуть до процесу відпуску ліків за допомогою комп’ютера аптечного працівника (із спеціалізованим програмним забезпеченням, встановленим на робочому місці).. Тобто вся послідовність дій при використанні робота -дозатора така:

– фармацевт отримує замовлення від пацієнта;

– за допомогою провізора комп'ютерного програмного забезпечення працівник вводить назву препарату в інформаційну систему управління логістикою до аптеки;

– автоматичний аптечний робот видаляє ліки з відповідних місць (осередків, полиць тощо) складського комплексу;

– аптечний робот видає заявлені пацієнтом ліки в аптечній касі.

Вся операція займає 6-8 секунд робочого часу. Цей комплекс програмного та апаратного забезпечення дуже простий і швидкий у

використанні; може переробляти та зберігати до 12 тисяч ліків; Він має компактні розміри, що є дуже актуальним, коли йдеться про організацію робочого процесу аптеки та у разі високих орендних ставок районів, приклад аптечного роботу - диспенсеру можна побачити на рисунку 1.5 [4].

Рисунок 1.5 - Аптечний робот - диспенсер [5]

### 1.1.2 Роботизований склад

Роботизовані склади являють собою окремі бокси, розташовані всередині аптечного складу. Всередині боксу, з обох сторін, розміщені стелажі зі скляними полицями. Між стелажами переміщається електромеханічний маніпулятор (один або два, залежно від комплектації), який під керуванням програми здійснює забір упаковок і доставку їх до місця зберігання або видачі. Завантаження складу автоматизована, співробітник аптеки послідовно викладає товар на стрічку конвеєра, далі по стрічці товар подається на вхідний вузол, де шляхом сканування ідентифікується і з використанням оптичних систем відбувається вимір його геометричних розмірів. Програма прораховує оптимальне місце для зберігання і дає команду маніпулятору на паркан і доставку. До місця видачі в торговому залі товар доставляється стрічковим

конвеєром, на який його вивантажує маніпулятор, приклад аптечного роботу складу можна побачити на рисунку 1.6.

Рисунок 1.6 - Аптечний робот склад [6]

### 1.1.3 Роботи-продавці

Для реалізації з аптеки товарів, користуються підвищеним попитом, використовують роботи-продавці. Роботи- продавці можуть повністю розташовуватися в торговому залі аптеки або вбудовуватися в стіну будівлі: корпус знаходиться в приміщенні, лицьова панель виходить на вулицю. При такому варіанті реалізація товарів не вимагає присутності фармацевтичних працівників і може здійснюватися цілодобово. Для вибору товарів застосовується клавіатура або сенсорний екран. Покупцям зручніше користуватися сенсорним екраном. Крім того, в апараті з відео монітором можна гнучко змінювати асортимент і надавати користувачам інформацію про особливості застосування реалізованих товарів.

## 1.2 Огляд існуючих методів

### 1.2.1 Завдання комівояжера

Сформульована задача - завдання цілочисленна. Нехай $x_{ij} = 1$ , якщо мандрівник переїжджає з і-ого міста в j-ий і $x_{ij} = 0$, якщо це не так.

Формально введемо $(n + 1)$ місто, розташоване там же, де і перше місто, тобто відстані від $(n + 1)$ міста до будь-якого іншого, відмінного від першого, рівні відстаням від першого міста. При цьому, якщо з першого міста можна лише вийти, то в (n+1) місто можна лише прийти.

Введемо додаткові цілі змінні, рівні номеру відвідування цього міста на шляху.$u_1 = 0$ , $u_{n+1} = n$. Для того, щоб уникнути замкнутих шляхів, вийти з першого міста і повернутися в $(n + 1)$ введемо додаткові обмеження, що зв'язують змінні $x_{ij}$ і змінні $u_i$ ($u_i$ цілі невід'ємні числа) [7].

### 1.2.2 Алгоритм Дейкстри

Алгоритм Дейкстри - алгоритм на графах, винайдений нідерландським вченим Едсгер Дейкстрой в 1959 році. Знаходить найкоротші шляхи від однієї з вершин графа до всіх інших.

У простій реалізації для зберігання чисел $d[i]$ можна використовувати масив чисел, а для зберігання приладдя елемента безлічі U - масив булевих змінних.

На початку алгоритму відстань для початкової вершини рівні нулю, а всі інші відстані заповнюються великим позитивним числом (великим максимального можливого шляху в графі). Масив прапорів заповнюється нулями. Потім запускається основний цикл.

На кожному кроці циклу ми шукаємо вершину v з мінімальною відстанню і прапором рівним нулю. Потім ми встановлюємо в ній прапор в 1 і перевіряємо всі сусідні з нею вершини u. Якщо в них (в u) відстань більше, ніж

сума відстані до поточної вершини і довжини ребра, то зменшуємо його. Цикл завершується, коли прапори всіх вершин стають рівні 1, або коли у всіх вершин с прапором $d[i] = \infty$. Останній випадок можливий тоді і тільки тоді, коли граф G незв'язний [8].

Складність алгоритму = $O(V^2)$

### 1.2.3 Побудова дерева рішень

Метод побудови дерева рішень зазвичай використовується для знаходження найкоротшого шляху в орієнтованому графі, при цьому від вихідного графа переходять до матриці суміжності. Потім переглядають рядки матриці суміжності, і граф послідовно «розшаровується» в дерево рішень.

Якщо простір пошуку включає в себе всі можливі варіанти рішень, то дерево, за допомогою якого здійснюється пошук в такому просторі, називається деревом повного перебору. Використання алгоритму повного перебору реально можна здійснити лише для дуже обмеженого простору пошуку. Якщо ж число варіантів вирішення досить велике, то дерево перебору набуває комбінаторні розміри, і вирішити задачу повного перебору (наприклад, пошуку шляхів між заданими вершинами графа) стає неможливим навіть для сучасних комп'ютерів. Тому виникає необхідність в пошуку не всіляких варіантів вирішення, а лише таких, які задовольняють певним умовам [9].

### 1.2.4 Одношарова штучна нейронна мережа

Одношарова штучна нейронна мережа – 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 змоделювати 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 [10].

Одношарові штучні нейронні мережі 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 [10].

### 1.2.5 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 [11].

### 1.2.6 Генетичний алгоритм

Генетичний алгоритм являє собою метод, що відображає природну еволюцію методів вирішення проблем, і в першу чергу задач оптимізації. Генетичні алгоритми — це процедури пошуку, засновані на механізмах природного відбору і спадкоємства. У них використовується еволюційний принцип виживання найбільш пристосованих особин. Вони відрізняються від традиційних методів оптимізації декількома базовими елементами. Зокрема, генетичні алгоритми:

– обробляють не значення параметрів самого завдання, а їх закодовану форму;

– здійснюють пошук рішення виходячи не з єдиної точки, а з їх деякої популяції;

– використовують тільки цільову функцію, а не її похідні або іншу додаткову інформацію;

– застосовують імовірнісні, а не детерміновані правила вибору.

Перераховані чотири властивості, які можна сформулювати також як кодування параметрів, операції на популяціях, використання мінімуму інформації про завдання і рандомізація операцій приводять у результаті до стійкості генетичних алгоритмів і до їх переваги над іншими широко вживаними технологіями [12].

### 1.2.7 Порівняльний аналіз існуючих методів вирішення завдання

Можна побачити з табл. 1.2, що алгоритми Дейкстри та Комівояжера виявилися найкращими. Алгоритм Дерева рішень ускладняється за простотою вихідної та вхідної інформації як і алгоритми нейронної мережі та мурашиних

колоній. Алгоритм штучної нейронної мережі має складну та непрозору реалізацію як і алгоритм мурашиних колоній. Якщо порівняти алгоритми за складністю то найкращім виявиться алгоритм штучної нейронної мережі але цей результат залежить від якості навчанні нейронної мережі та може сильно відрізнятися на практиці, дещо складнішим виявився алгоритм Дейкстри який на відміну від нейронної мережі має просту та швидку реалізацію, найскладнішим виявився алгоритм мурашиних колоній. Найгіршим виявився алгоритм мурашиних колоній який має крім описаних мінусів ще й низьку адаптивність до особливостей вирішувальної задачі.

Таблиця 1.2 – Порівняння алгоритмів

| Алгоритм | Дейк-стри | Комівояжера | Дерева рішень | Одношарова штучна нейронна мережа | Мураши-них колоній | ГА(про-стий) |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Концептуальна простота та прозорість реалізації | Так | Так | Так | Ні | Ні | Ні |
| Висока часова залежність від розмірності задачі | Так | Так | Так | Так | Так | Так |
| Простота кодування вхідної і вихідної інформації | Так | Так | Ні | Ні | Ні | Так |
| Можливість розпаралелювання | Так | Так | Так | Ні | Ні | Так |

Продовження таблиці 1.2

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Здатність оцінити точність знайденого розв’язку | Так | Так | Так | Так | Так | Так |
| Висока залежність результатів роботи методу від початкових параметрів пошуку | Так | Так | Так | Так | Ні | Так |
| Висока адаптивність до особливостей вирішувальної задачі | Так | Так | Так | Так | Ні | Ні |
| Складність алгоритму | 2 | 2 | 2 | 1 | 3 | 3 |

### 1.3 Постановка завдання до роботи

В даній роботі розглядається проблема доставки препаратів роботом фармацевтом, складність задачі можна описати даною формулою (1.1):

$$F = x_1, x_2, x_3, x_4, x_5, x_6, x_7\,, \tag{1.1}$$

де $x_1$ – вага препаратів;

$x_2$ – габарити препарату;

$x_3$ – кількість препаратів;

$x_4$ – характеристики руки-маніпулятора;

$x_5$ – шлях доставки;

$x_6$ – кількість вільних місць під препарати;

$x_7$ – мінімальна кількість ходок з максимальним заповненням та з максимальною загрузкою.

Вага препаратів може бути представлена за формулою (1.2):

$$x_1 = [(\sum_{i=1}^{n} M_i > 800\text{гр}\,, \tag{1.2}$$

де $M_{1,2...n}$ – вага препарату.

Кількість препаратів може бути представлена за формулою (1.3):

$$x_2 = [(\sum_{i=1}^{n} M_i > 1{,}5\text{м}^3 => M_i = \begin{bmatrix} \begin{cases} P_1 < 100\text{см} \\ P_2 < 60\text{см} \\ P_3 < 250\text{см} \end{cases} \end{bmatrix}, \tag{1.3}$$

де $P_1$ – ширина препарату;

$P_2$ – висота препарату;

$P_3$ – довжина препарату.

Характеристики руки-маніпулятора можуть бути представлені за формулою (1.4):

$$x_4 = \begin{bmatrix} \begin{cases} T_1 < 5\text{км/год} \\ T_2 < 800\text{гр} \\ T_3 < 1{,}5\text{м}^3 \end{cases} \end{bmatrix}, \tag{1.4}$$

де $T_1$ – швидкість руху;

$T_2$ – вантажопідйомність;

$T_3$ – робочий об`єм руки маніпулятора.

Всі ці параметри напряму залежать від складності розв'язувальної задачі, та головною метою проєкту, є зменшення шляху доставки препарату до користувача.

## 1.4 Висновки за розділом

У цьому розділі розглядаються основні поняття та терміни, також розглянуті аналоги розробки та порівняно характеристики цих комплексів.

Було досліджено основні методи розв’язання задач та надано порівняльну характеристику цих методів.

Були сформульовані та визначені завдання для вирішення.

# 2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

## 2.1 Вибір інструментарію розробки програмного продукту

Важливим етапом життєвого циклу інформаційної системи, а особливо проєктування ПП, є вибір інструментів і підключення необхідної технології до проєкту.

Основна задача ПП – це програмна реалізація методів доставки ліків.

Розроблюваний ПП повинен обов’язково містити такі модулі:

– модуль алгоритмів;

– модуль обробки вхідної інформації;

– модуль роботи з базою даних;

– БД;

– графічний інтерфейс.

Для проєктування ПП необхідно обрати такі засоби, як мова програмування, середовище розробки (IDE) та система керування базами даних (СКБД).

### 2.1.1 Вибір мови програмування

Для вирішення проблеми, яка полягає у впровадженні методів доставки ліків роботом-фармацевтом, необхідно переглянути сучасні мови програмування та порівняти їх між собою, щоб вибрати ту, яка найкраще підходить для вирішення цієї проблеми. Нижче наведено опис найпопулярніших мов програмування для розробки програмного забезпечення з виділенням їх переваг і недоліків.

C++ (Сі-плюс-плюс) – мова програмування високого рівня з підтримкою кількох парадигм програмування: об’єктно-орієнтованої, узагальненої та процедурної [9].

Переваги мови C++:

– швидкодія. Швидкість роботи програм на C++ практично не поступається програмам на C, хоча програмісти отримали в свої руки нові можливості і нові засоби;

– масштабованість. На мові C++ розробляють програми для найрізноманітніших платформ і систем;

– можливість створення узагальнених алгоритмів для різних типів даних, їхня спеціалізація, і обчислення на етапі компіляції, з використанням шаблонів;

– підтримуються різні стилі та технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, мета програмування (шаблони, макроси) [13].

Недоліки C++:

– відсутність системи модулів. C++ успадкувала від C підключення заголовних файлів за допомогою препроцесора. Це змушує дублювати опис об'єктів, породжує неочевидні вимоги до коду і збільшує обсяг компільованого тексту, а значить і час компіляції;

– наявність більш ніж одного механізму для виконання одних і тих же завдань, що ускладнює мову і призводить до неоптимального і небезпечного кодування;

– відсутність вбудованих механізмів статичної валідації часу життя об'єктів, що призводить до раптового краху програми через звернення до видаленої змінної, або через неправильну багато потокову роботу з об'єктами [13].

Отже, C++ потужна мова програмування загального призначення, що володіє блискавичною продуктивністю. Питання невисокої популярності C++ в Data Science пояснюється вибором продуктивності обчислень проти продуктивності мови [13].

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду.

Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій [14].

Переваги мови Python:

– дуже популярний і має безліч розширень і підтримку співтовариства розробників;

– пакети, такі як pandas, scikit-learn і Tensorflow, роблять Python відмінним варіантом для сучасних додатків з машинним навчанням [14].

Недоліки Python:

– мова з динамічною типізацією. Так що слід бути уважним і очікувати час від часу помилок на зразок тих, де метод очікує отримати на вході integer, а отримує рядок.

Java – популярна високорівнева мова програмування загального призначення, яка працює на віртуальній машині Java (JVM). Це абстрактна обчислювальна система, яка забезпечує плавну переносимість між платформами [15].

Переваги:

– дуже популярна мова програмування. Багато сучасних систем і додатків були створені на Java;

– строго типізована мова. По відношенню до визначення типів Java вкрай серйозна. Для додатків, що працюють з великими об’ємами даних це безцінно;

– високопродуктивна компільована мова загального призначення. Одна й та сама мова може бути використана і для написання бізнес-логіки, і для аналітики великих об’ємів даних, на що не здатні інші мови програмування для Data Science [15].

Недоліки:

– для вузько направленого аналізу і специфічних статистичних додатків синтаксис Java занадто багатослівний. Динамічно типізовані R і Python тут принесуть куди більше користі;

– для Java створено не так багато бібліотек для роботи з аналізу даних [15].

C# — це об'єктно-орієнтована мова програмування. Розроблена у 1998-2001 роках командою інженерів Microsoft під керівництвом Андерса Халесберга та Скотта Вільтаумота як мова розробки програмного забезпечення для Microsoft .NET Framework. C# належить до сімейства мов із C-подібним синтаксисом, синтаксис якого знаходиться поруч із C++ і Java [16].

Переваги:

– розширюваність системи (в C# можна спокійно відвантажувати будь-які exe, імпортувати класи та об'єкти з інших програм);

– кросплатформенність (mono, концепція NET);

– складність розробки та супроводу (підбір кадрів, читання коду, документованість мови);

– ступінь відкритості вихідних текстів бібліотек, виконуваних програм, кількість літератур та допомогу (MSDN);

– можливість залучення сторонніх розробників при розробці системи для програмування вузько-спеціалізованих завдань (складання, модулі, також exe);

– захищеність і контроль версій алгоритмів, що підключаються (концепція NET);

– трудомісткість написання (той самий NET);

– швидкість роботи (Розподіл процесів, розподіл даних, швидкість роботи з даними);

– зручність розробки (середовище розробки в порівнянні з Delphi7).

Недоліки:

– пріоритетна орієнтованість на платформу Windows;

– мова безкоштовна тільки для невеликих фірм, індивідуальних програмістів, стартапів та учнів. Великій компанії купівля ліцензійної версії цієї мови обійдеться в круглу суму.

Тому кожна з наведених вище мов програмування має великий набір корисних функцій, які можуть бути корисними при вирішенні задачі програмної реалізації методів доставки ліків роботом-фармацевтом. Завдяки таким мовам програмування дані відображаються, обробляються та аналізуються. Ці мови програмування відповідають вимогам, яким повинні відповідати дані програмування. Слід зазначити, що вищенаведені мови програмування розвиваються і реалізують можливості, які раніше не були доступні для обробки та візуалізації інформації [13] - [16].

При аналізі завдань [13] - [16] була створена порівняльна характеристика мов програмування, яка наведена в таблиці. 2.1.

Таблиця 2.1 – Порівняльна характеристика мов програмування

| Критерії порівняння | C# | C++ | Python | Java |
|---|---|---|---|---|
| Ліцензія | Ліцензія Apache 2.0 .NET Core | – | Python Softwa-re Founda-tion License | GNU GPL |
| Кросплатформеність | Так | Так | Так | Так |
| Можливість обробки великої вибірки даних | Так | Так | Так | Так |
| Наявність зрозумілої та повної документації | Так | Так | Так | Так |
| Наявність якісної візуалізації даних | Так | Ні | Так | Ні |
| Простота вивчення та розробки | Так | Так | Так | Так |
| Наявність якісних предметно-орієнтованих пакетів з відкритим вихідним кодом | Так | Так | Так | Так |
| Наявність великої кількості наборів статистичних функцій та методів | Так | Ні | Так | Ні |
| Паралельне програмування | Так | Так | Ні | Так |
| Взаємодіє з іншими мовами | Так | Так | Так | Так |

Як видно з табл. 2.1 за результатами порівняльної характеристики мова C# виявилася найбільш оптимальною, оскільки має переваги, що забезпечують найкращу візуалізацію даних, зручне та легке написання коду. C# має велику кількість бібліотек і вже реалізовані методи з відкритим кодом. Ця мова широко поширена, має велике співтовариство розробників і документацію, яка детально описує реалізацію функцій і допомагає в реалізації будь-яких розробок. Вартість впровадження та підтримки також впливає на вибір, слід зазначити, що мова C# вільна для розповсюдження і проста в обслуговуванні. Серед переваг – ця мова актуальна на ринку, а це означає, що в майбутньому не виникне серйозних проблем з додаванням нових функцій або навіть адаптацією проєкту під нову розробку. Тому, підводячи підсумок для розробки програмного забезпечення, була обрана мова програмування C#.

### 2.1.2 Вибір середовища розробки

Для розробки ПП на вибраній мові програмування, треба вибрати найкраще середовище розробки.

Для розробки на мові програмування C# можна виділити 3 основні середовища розробки: Visual Studio, Rider, Eclipse.

Visual Studio IDE це платформа для завантаження для введення, налагодження, компіляції та публікації програм. Додаток до базового редактора та налагодження, які в більшості середовищ IDE, Visual Studio включає в себе транслятори, доробку коду, графічні дизайнери та багато іншого, щоб покращити процес розробки.

Переваги:

– офіційна. Оскільки і мова, і середовище розробки створені в Microsoft, логічно припустити, що нічого функціональнішого ви не знайдете у всьому Інтернеті. У деяких випадках без Visual Studio не обійтися, наприклад, при використанні технологій UWP та WPF;

– безкоштовна. Версії Community edition для рядового користувача буде достатньо. Тим більше, тепер можна підключати плагіни (на відміну старої версії Express);

– функціональна. У Visual Studio безліч якісних плагінів. З їх допомогою можна розширити функціональність програми та підключити інші мови;

– підтримує платформи. Visual Studio має широкі можливості розробки додатків під Windows, у тому числі в .NET-сегмент;

– хмарні сховища. Зареєструйтесь у спільноті Visual Studio - і отримайте доступ до хмарного сховища, де зможете розташовувати файли проєктів;

– корпоративність. Технологія бэклога дозволяє членам команди взаємодіяти за гнучкої методології розробки.

Недоліки:

– баги при переходах із тріал-версії. При переході на платну версію можуть губитися налаштування та порушувати роботу корпоративного сервера;

– складність. Самостійно освоїти Visual Studio новачку буде непросто - дуже багато доступних функцій, захованих у підрозділах меню.

Rider – це кросплатформне середовище розробки програмного забезпечення для платформи .NET, розроблена в компанії JetBrains.

Переваги:

– ReSharper. Це плагін, спочатку розроблений підвищення продуктивності Visual Studio. Тепер на його основі випущено IDE;

– підтримка повного циклу. Фірмова характеристика товарів JetBrains, втілена і в Project Rider. З ним ви зможете організувати весь цикл створення програмного забезпечення: від ідеї до підтримки;

– функціональність. Project Rider дозволяє підключити MSBuild та XBuild, працювати з CLI-проєктами та організувати налагодження програм .NET and Mono. Багато опцій для швидкого створення коду покращує продуктивність;

– multiple runtime. Підтримка кількох запущених програм;

– кросплатформність. Project Rider працює з Windows, Linux та MacOS;

– контроль версії. Вбудований інструмент дозволяє безпосередньо організувати роботу з Git, Mercurial та TFS.

Недоліки:

– молодість. Частина функціональності ще у розробці, не всі стартові баги виправлені;

– вартість. Найдешевша версія Project Rider коштуватиме 139 доларів за перший рік використання. Але є тріал-версія та спеціальні пропозиції для студентів та непрофільних організацій.

Eclipse – це безкоштовний інтегрований модуль розробки модулів для різних платформних додатків. Розроблений та підтримується Eclipse Foundation.

Переваги:

– безліч плагінів. У Eclipse чи не найбільша кількість надбудов - "на всі випадки життя";

– активне співтовариство. Допомагає якнайшвидше освоїти середовище розробки, випускає нові плагіни;

– відмінні компілятор та відладчик. Перший працює на порядок швидше, ніж у конкурентів, другий показує потоки, перетину, дозволяє гнучко керувати ходом налагодження;

– кастомізація. Завдяки плагінам та настройкам можна повністю персоналізувати Eclipse;

– безкоштовність. Це open-source проєкт абсолютно безкоштовний;

– висока функціональність. Завдяки розробникам-офіціалам та членам спільноти за допомогою Eclipse можна провести будь-який C#-продукт за повним циклом розробки.

Недоліки:

– складність. Як і будь-який функціональний продукт, Eclipse може здатися новачкові надто складним;

– немає гарантій надійності. Оскільки плагіни створюються спільнотою, за їхню якість відповідає лише розробник. Крім того, самі творці Eclipse з кожною новою версією плодять баги, не встигаючи часом виправляти старі.

Під час аналізу середовищ розробки було створено порівняльну характеристику середовищ розробки, яка наведена у табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика середовищ розробки

| Критерії порівняння | Visual Studio | Rider | Eclipse |
|---|---|---|---|
| Ліцензія | Freemium | Community edition: Apache License 2.0 | Eclipse Public License |
| Кросплатформеність | Так | Так | Так |
| Наявність автодоповнення | Так | Так | Так |
| Підтримка репозиторіїв | Так | Так | Так |
| Наявність «plugin manager» | Так | Так | Так |
| Простота встановлення | Так | Так | Ні |
| Підтримка аналізу коду | Так | Так | Так |
| Підтримка рефакторингу коду | Так | Так | Так |
| Наявність інтерактивної консолі | Так | Так | Ні |
| Зручний користувацький інтерфейс | Так Так | Так | Ні |
| Швидкість модернізації | Так Так | Так | Ні |

Як видно з табл. 2.2 найбільші переваги має середовище розробки Visual Studio, бо воно було розроблене компанією яка і створила мову програмування C#. Середовище дозволяє просто керувати різними проєктами та користуватися системами контролю версій. Отже, підбиваючи підсумки для розробки ПП було обрано середовище Visual Studio.

## 2.2 Структурна схема програмного продукту

Розроблюваний ПП повинен обов’язково містити такі модулі:

– модуль алгоритмів;

– модуль обробки вхідної інформації;

– модуль роботи з базою даних;

– БД;

– графічний інтерфейс.

Розроблюваний ПП використовує дані про розміщення препаратів та споживачів, а також можливі варіанти розміщення препаратів в роботі, тобто комбінації кількості і типорозмірів робота-фармацевта. Ці дані можуть бути представлені у таких сховищах:

– текстові файли (CSV, txt, xml,…);

– бінарні файли даних;

– не реляційні бази даних (NoSQL сховища);

– реляційні бази даних (MySql, MS SQL Server, Firebird, …) [13].

Обраний тип зберігання повинен забезпечувати достатню гнучкість, високу швидкість отримання даних, здатність зберігати великі об’єми даних, здатність до розширення функціоналу та простоту налаштування. Всім цим вимогам відповідають текстові xml файли. Переваги використання xml файлів:

– підприємство може мати багато XML в стандартному форматі, який існує;

– може знадобитися, щоб дані показали або сприйняли як XML, тому використання іншого формату, таких як реляційних сил подвійного моделювання даних;

– xml дуже добре підходить до розріджених даних, які глибоко; захованих даних та змішаного змісту (наприклад текст із вбудованими мітками)

– xml є легким для читання людьми, тоді як реляційні таблиці вимагають знань для доступу;

– метадані часто доступні як XML;

– семантичні веб-дані доступні як RDF / XML.

В вибраному середовищі розробки є реалізовані класи та методи для управління текстовими xml файлами, що значно спрощує процес створення ПП.

Всі атрибути файлу та їх опис зазначено в табл. 2.3.

Таблиця 2.3 –Атрибути

| Атрибут | Опис сутності |
|---|---|
| namedrug | Назва препарату |
| seriadrug | Серійний номер препарату |
| sizedrug | Розміри коробки препарату |
| massdrug | Маса препарату |
| nom | Порядковий номер препарату |

Якщо перейти до модулів програмного продукту то можна виділити такі модулі: розрахунковий модуль, графічний інтерфейс та базу даних.

Модуль обробки вхідної інформації – це основний та найважливіший модуль, його призначення це відправка даних для обробки в алгоритми доставки та отримання даних від бази даних.

Модуль роботи з базою даних – це модуль потрібний для записування та вилучення даних про потрібний препарат з бази даних.

Модуль алгоритмів – це модуль де реалізовано основні алгоритми доставки препаратів.

Модуль графічного інтерфейсу – графічний інтерфейс буде розроблено за допомогою технології WinForms яка є інтуїтивно зрозумілою та вбудована в середовищу розробки Microsoft Visual Studio Community 2019.

Для більш наглядного представлення структурної схеми програмного продукту був створений макет розтушування модулів програмного продукту (рис. 2.2).

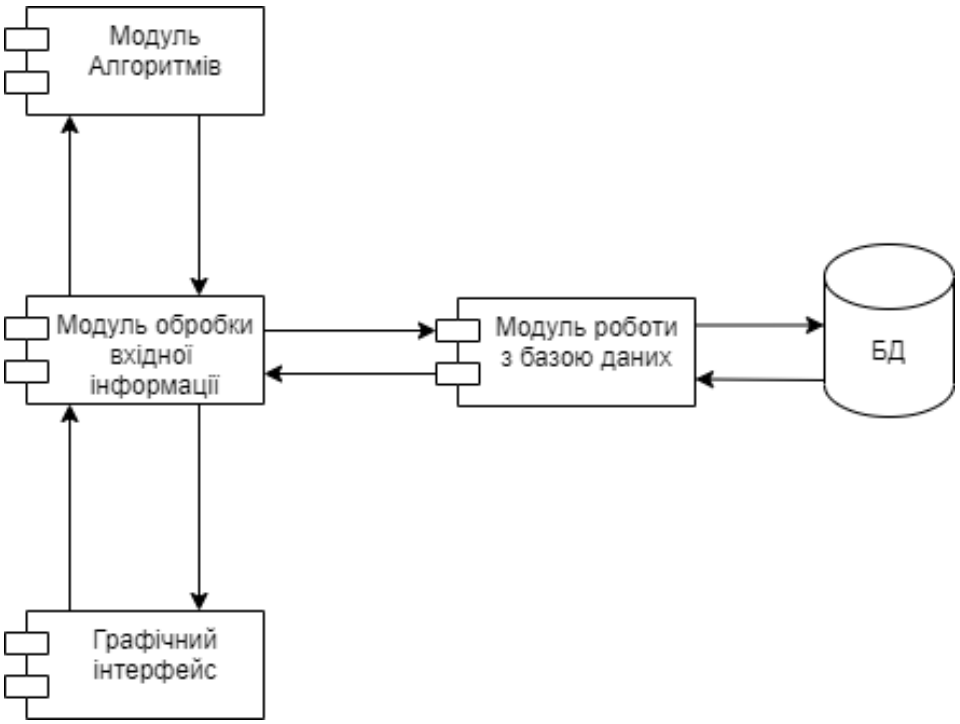



Рисунок 2.2 – Структурна схема програмного продукту

На рисунку 2.3 наведено діаграму класів де відображено назву та розташування методів та алгоритмів в класах.

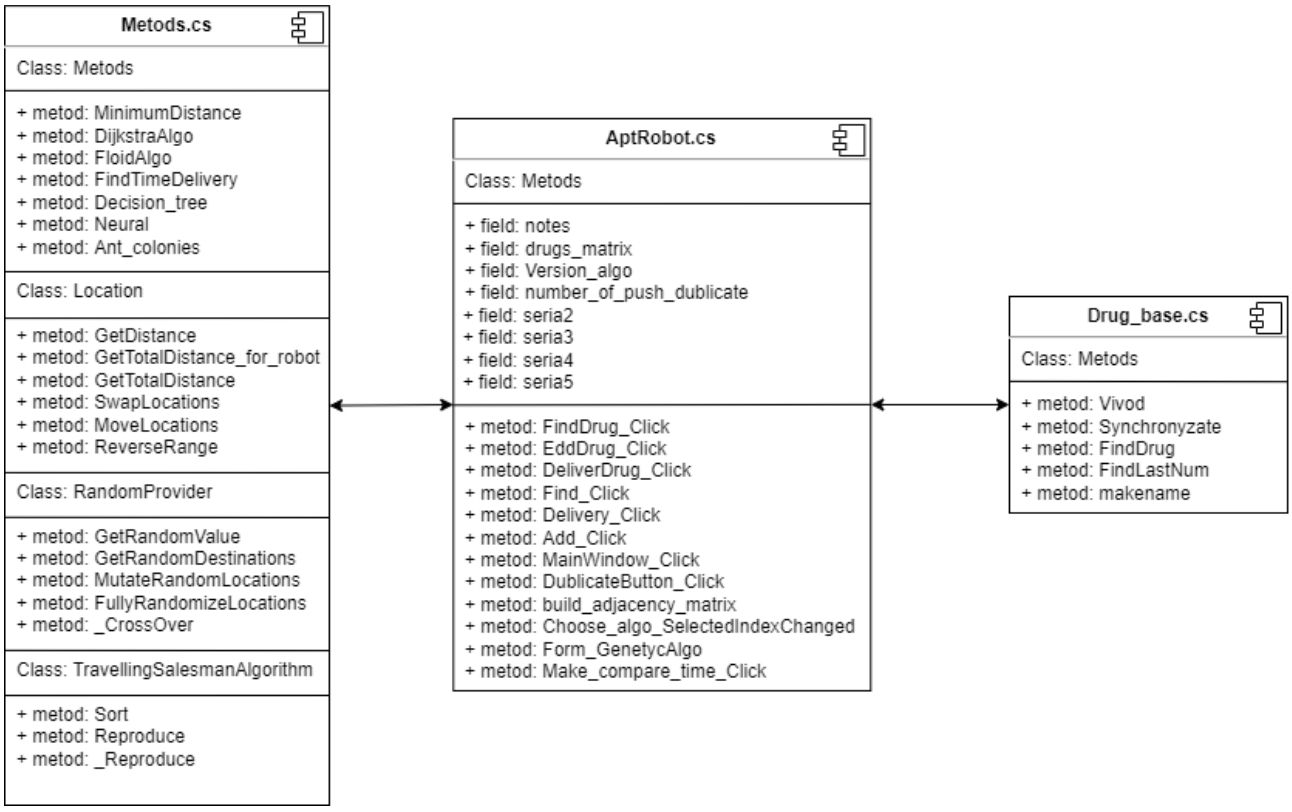



Рисунок 2.3 – Діаграма класів програмного продукту

## 2.3 Висновки за розділом

В даному розділі було проведено аналіз та вибір інструментарію розробки ПП та було спроєктовано архітектуру майбутнього ПП. Проведено порівняльну характеристику мов програмування та встановлено, що найбільші переваги для розробки розрахункового модуля має мова програмування C#. Також було проведено порівняльну характеристику середовищ розробки та встановлено, що найкращі результати отримала Microsoft Visual Studio Community 2019. Для розробки графічного інтерфейсу було взято технологію WinForms. В якості системи зберігання даних обрано текстовий xml файл. Також було відображено структурну схему програмного продукту та діаграму класів.

# 3 ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

У даній роботі розглядається проблема доставки препаратів роботом-фармацевтом, на першому етапі якої вирішуються питання реалізації новітніх алгоритмів доставки.

Для досягнення цієї мети було вирішено використати генетичний алгоритм, алгоритм Флойда — Воршелла, алгоритм Дейкстри, алгоритм дерева рішень, нейронний алгоритм та  алгоритм мурашиних колоній.

## 3.1 Функціональні вимоги до програмного продукту

Через збільшення ефективності роботи робота-фармацевта було вирішено що застосунок буде призначений для використання одним користувачем. У ході аналізу були виявлені такі необхідні користувачу варіанти використання:

– додавання препарату;

– пошук препарату;

– доставка препарату.

Вимоги зазначені у таблиці 3.1.

Таблиця 3.1 – Функціональні вимоги

| Варіант використання | Функціональна вимога | Пріоритет |
|---|---|---|
| 1 | 2 | 3 |
| Додавання препарату | Застосунок має надавати можливість додавати препарат до роботу-фармацевту | Високий |

Продовження таблиці 3.1

| 1 | 2 | 3 |
|---|---|---|
| | Застосунок має надавати можливість додавання назви до препарату | |
| | Застосунок має надавати можливість додавання ваги до препарату | |
| | Застосунок має надавати можливість додавання розміру до препарату | |
| | Застосунок має надавати можливість додавання серії до препарату | |
| Пошук препарату | Застосунок має надавати можливість пошуку препарату | Високий |
| | Застосунок має надавати можливість вводу серії препарату | |
| | Застосунок має надавати можливість виводити повну інформацію про препарат | |
| Доставка препарату | Застосунок має надавати можливість доставляти препарат | Високий |

Продовження таблиці 3.1

| 1 | 2 | 3 |
|---|---|---|
| | Застосунок має надавати можливість вводити серію препарату що доставляється | |
| | Застосунок має надавати можливість множинної доставки препаратів | |
| | Застосунок має надавати можливість вибору алгоритмів доставки | |
| | Застосунок має надавати можливість порівняння швидкості доставки алгоритмами | |

Варіанти використання ПП зображені на рисунку 3.1.

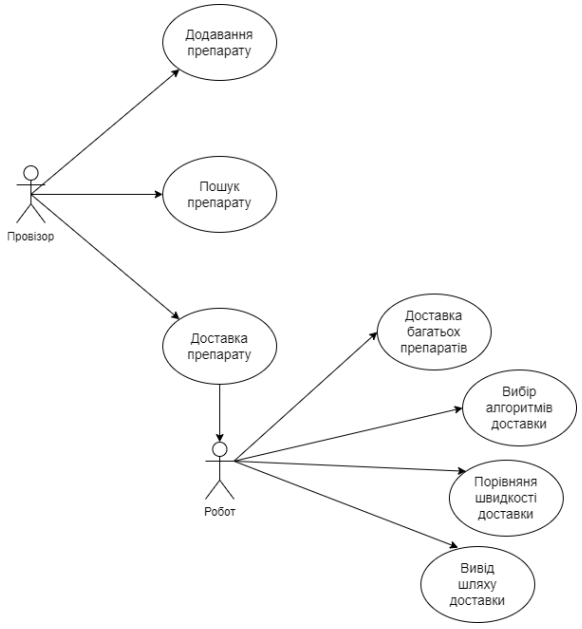



Рисунок 3.1 – Uml схема структурних варіантів використання

## 3.2 Опис архітектури та обраних технологій

Головним класом є клас “FormRobot”, з нього викликаються всі інші класи та виконується робота з інтерфейсом застосунку.

```
public partial class FormRobot : Form
    {
        public FormRobot()//конструктор
        private void FindDrug_Click(object sender, EventArgs e) //обробка натиску на кнопку відображення меню пошуку препарату
        private void EddDrug_Click(object sender, EventArgs e) //обробка натиску на кнопку відображення меню додавання препарату
        private void DeliverDrug_Click(object sender, EventArgs e) //обробка натиску на кнопку відображення меню доставки препарату
        private void Find_Click(object sender, EventArgs e) //обробка натиску на кнопку пошуку препарату
        private void Delivery_Click(object sender, EventArgs e) //обробка натиску на кнопку доставки препарату
       private double Form_GenetycAlgo(string[] drug, string[] drug2, string[] drug3, string[] drug4, string[] drug5) //обробка викликів генетичного алгоритму
        private void Add_Click(object sender, EventArgs e) //обробка натиску на кнопку додавання препарту
        private void MainWindow_Click(object sender, EventArgs e) //обробка натиску на кнопку повертання на головну сторінку
       private void DublicateButton_Click(object sender, EventArgs e) //обробка натиску на кнопку додавання препарату для доставки
        public void build_adjacency_matrix() //ствторення та обробка матриці суміжності
        private void Choose_algo_SelectedIndexChanged(object sender, EventArgs e) //обробка вибору алгоритму
        private void Make_compare_time_Click(object sender, EventArgs e) // обробка натиску на кнопку порівняння швидкості доставки різними методами
    }
```

Робота з базою даних здійснюються через клас “ Drug_base”, в ньому реалізовано: вивід, синхронізацію, пошук та робота з номером препарату.

```
class Drug_base
    {
        public string[,] Vivod(string[,] notes)//вивід бази даних
        public void Synchronyzate(string name, string seria, string mass, string size) //синхронізація бази даних
```

```
        public string FindDrug(string seriadrug) //пошук потрібного
        public int FindLastNum()//виводить номер останнього препарату
        public int Num_of_nodes()//виводить кількість препаратів в базі даних
        public string[] Makename()//виводить данні про препарат у вигляді масиву
    }
```

Основні алгоритми описані в классі “Metods”. В классі “Metods” реалізовано алгоритм Дейкстри через функції MinimumDistance() та DijkstraAlgo() та алгоритм Флойда — Воршелла через функцію FloidAlgo(), також в класі було реалізовано підрахування часу доставки препарату у секундах та вивід положення препарату в роботі для генетичного алгоритму.

```
    class Metods {
        private int MinimumDistance(int[] distance, bool[] shortestPathTreeSet, int verticesCount)//підрахування мінімальної дистанції
        public int DijkstraAlgo(int[,] graph, int source, int verticesCount,int num_ver) //алгоритм Дейкстри
        public int FloidAlgo(int[,] graph, int num_ver, int source)// алгоритм Флойда — Воршелла
        public int Decision_tree (int[,] graph, int num_ver, int source)// алгоритм дерева рішень
        public int Neural (int[,] graph, int num_ver, int source)// нейронний алгоритм
        public int Ant_colonies (int[,] graph, int num_ver, int source)// алгоритм мурашиних колоній
        public string FindTimeDelivery(int dist) //алгорит пошуку часу доставки
       public int[] Make_xy(int num_on_robot)//вивід положення препарату в роботі для генетичного алгоритму
      }
```

Клас Locations потрібен для зберігання та створення інформації про розташування препаратів в роботі

```
    class Location{
        public double GetDistance(Location other)//підрахування відстанні між точками
        public static double GetTotalDistance_for_robot(Location startLocation, Location[] locations, int num_of_findable_bestsolution) //підрахування відстані між двома визначеними точками
```

```
        public static double GetTotalDistance(Location startLocation, Location[] locations)// підрахування повної відстані
        public static void SwapLocations(Location[] locations, int index1, int index2) //змінити розташування двох препаратів в роботі
        public static void MoveLocations(Location[] locations, int fromIndex, int toIndex)//перемістити один препарат
        public static void ReverseRange(Location[] locations, int startIndex, int endIndex) )//від дзеркалити положення препаратів в роботі
    }
```

Клас RandomProvider потрібен для створення та обробки масиву з розташуванням препаратів в роботі

```
class RandomProvider {
        public static int GetRandomValue(int limit) //повертає випадкове число в певному діапазоні
        public static Location[] GetRandomDestinations(int count) //заповнює масив з розташуванням препаратів в роботі
        public static void MutateRandomLocations(Location[] locations) //дозволяє змінювати розташування препаратів за допомогою мутації
        public static void FullyRandomizeLocations(Location[] locations) //повністю випадкові положення препаратів в роботі
        internal static void CrossOver(Location[] locations1, Location[] locations2, bool mutateFailedCrossovers) //дозволити рекомбінацію
    }
```

Клас TravellingSalesmanAlgorithm це головний клас він потрібен для виклику всіх класів та виведення найкращого шляху

```
class TravellingSalesmanAlgorithm {
        public TravellingSalesmanAlgorithm(Location startLocation, Location[] destinations, int populationCount) //потрібен для пошуку найближчого не відвідуваного місця
            private static int Sort(KeyValuePair<Location[], double> value1, KeyValuePair<Location[], double> value2) //потрібен для сортування всіх положень препаратів
            public IEnumerable<Location> GetBestSolutionSoFar() //повертає найкраще рішення
        public void Reproduce() //сортує всі шляхи та видаляє невигідні
        private Location[] _Reproduce(Location[] parent) // сортує всі шляхи та видаляє невигідні також використовує мутацію для сортування
    }
```

## 3.3 Порівняння шляху доставки різними алгоритмами

У цьому підрозділі наведено порівняльну характеристику часу та шляху доставки за використання різних алгоритмів, а саме: простий генетичний алгоритм, алгоритм Флойда — Воршелла, алгоритм Дейкстри, алгоритм дерева рішень, нейронний алгоритм та  алгоритм мурашиних колоній.

### 3.3.1 Порівняння доставки 6 препаратів

В першому порівняні буде наведено розташування 6 препаратів та побудовано шлях доставки різними алгоритмами. Розташування препаратів наведено в таблиці 3.2.

Таблиця 3.2 – Розташування 6 препаратів

| Номер препарату | Координата x | Координата y |
|---|---|---|
| 1 | 50 | 501 |
| 2 | 784 | 721 |
| 3 | 736 | 168 |
| 4 | 32 | 287 |
| 5 | 464 | 77 |
| 6 | 100 | 200 |

За результатами першого порівняння в рисунках 3.2 - 3.4  наведено шлях доставки 6 препаратів генетичним алгоритмом, алгоритмом Дейкстри та алгоритмом Флойда — Воршелла.

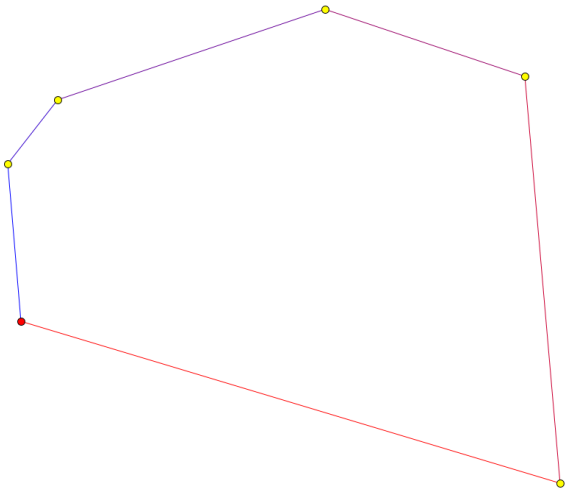


Рисунок 3.2 – Шлях генетичного алгоритму

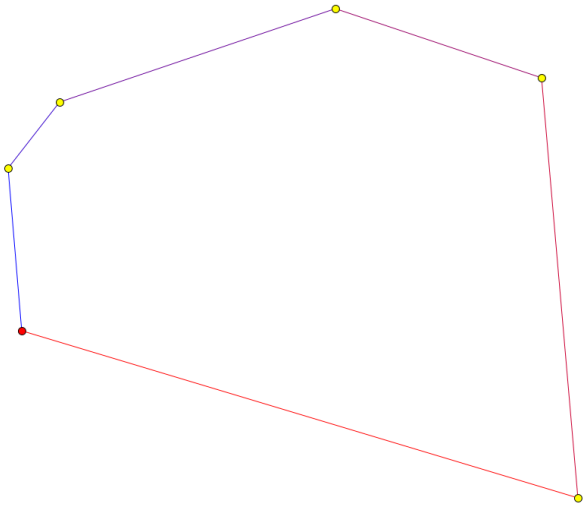


Рисунок 3.3 – Шлях алгоритму Дейкстри

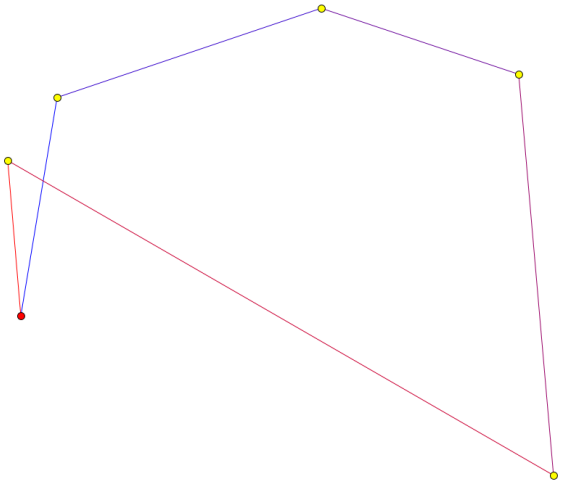


Рисунок 3.4 – Шлях алгоритму Флойда — Воршелла

За результатами першого порівняння в рисунках 3.5 - 3.7 наведено шлях доставки 6 препаратів мурашиним алгоритмом, алгоритмом нейронної мережі та алгоритмом дерева рішень.

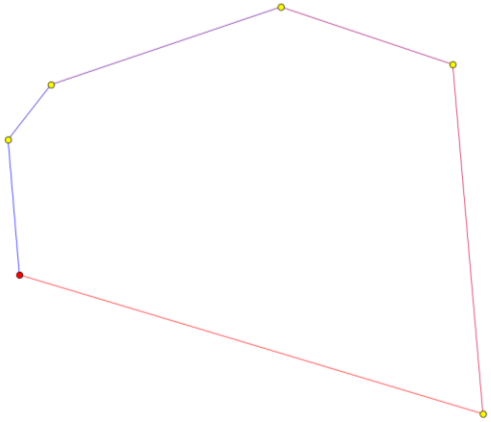


Рисунок 3.5 – Шлях мурашиного алгоритму

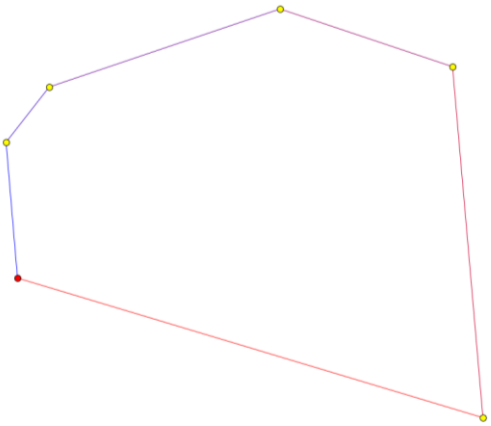


Рисунок 3.6 – Шлях нейронної мережі

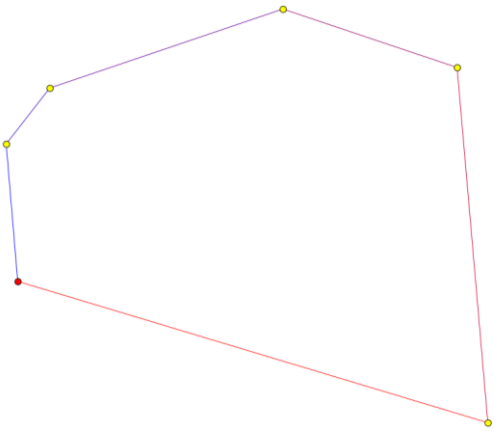


Рисунок 3.7 – Шлях алгоритму дерева рішень

В таблиці 3.3 наведено дистанцію та час доставки 6 препаратів алгоритмами.

Таблиця 3.3 – Дистанція та час доставки 6 препаратів алгоритмами

| Критерії порівняння | Простий генетичний алгоритм | Алгоритм Дейкстри | Алгоритм Флойда — Воршелла | Мурашиний алгоритм | Нейронний алгоритм | Алгоритм дерева рішень |
|---|---|---|---|---|---|---|
| Довжина шляху | 9,26м | 9,26м | 10,45м | 9,26м | 9,26м | 9,26м |
| Час доставки | 32,4сек | 32,4сек | 36,5сек | 32,4сек | 32,4сек | 32,4сек |

Якщо проаналізувати результати доставки 6 препаратів то можна визначити що майже всі алгоритми виявилися схожими за результатом, відрізнився лише алгоритм Флойда — Воршелла де шлях та час доставки виявся більшим за інші алгоритми.

### 3.3.2 Порівняння доставки 10 препаратів

В другому порівняні буде наведено розташування 10 препаратів та побудовано шлях доставки різними алгоритмами. Завдяки цьому порівнянню можна в результаті дізнатися який алгоритм краще працює з невеликою кількістю препаратів, а який більш кращій для доставки зразу багатьох препаратів. Розташування препаратів наведено в таблиці 3.4.

Таблиця 3.4 – Розташування препаратів

| Номер препарату | Координата x | Координата y |
|---|---|---|
| 1 | 50 | 501 |
| 2 | 304 | 728 |
| 3 | 560 | 203 |
| 4 | 376 | 315 |
| 5 | 280 | 317 |
| 6 | 112 | 273 |
| 7 | 480 | 196 |
| 8 | 712 | 623 |
| 9 | 720 | 406 |
| 10 | 56 | 329 |

За результатами другого порівняння в рисунках 3.8 - 3.11 наведено шлях доставки 10 препаратів генетичним алгоритмом, алгоритмом Дейкстри мережі, алгоритмом Флойда — Воршелла, та мурашиним алгоритмом.

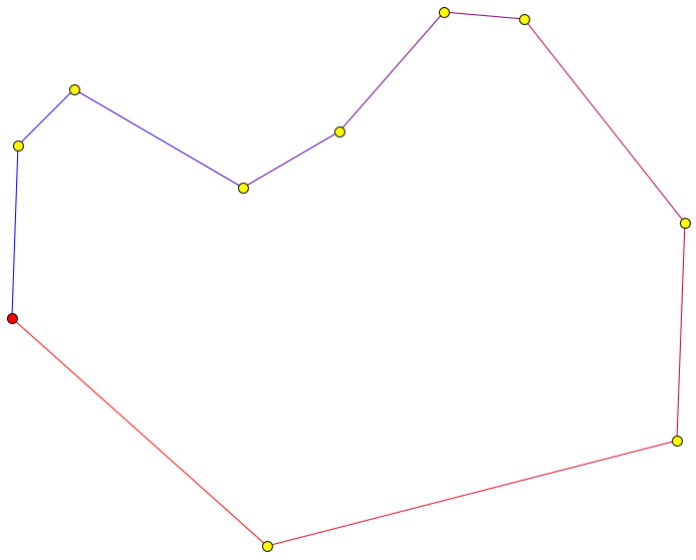


Рисунок 3.8 – Шлях генетичного алгоритму

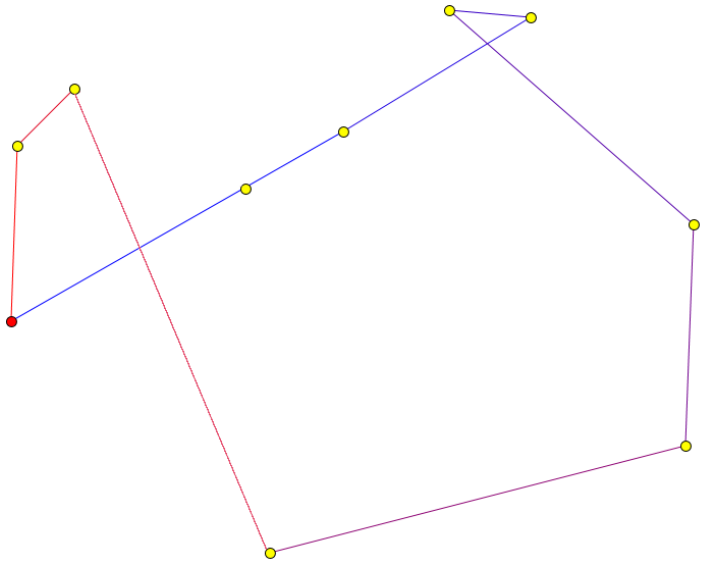


Рисунок 3.9 – Шлях алгоритму Дейкстри

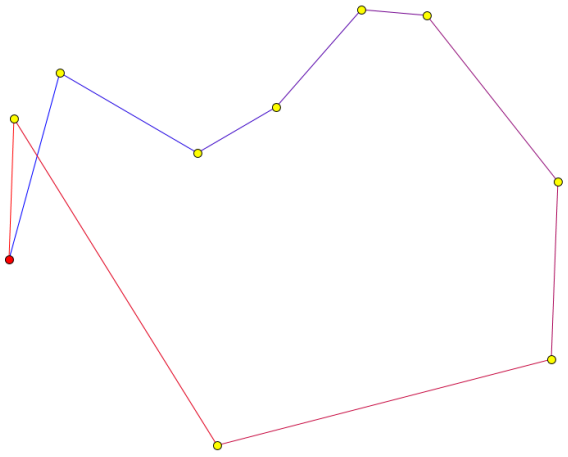


Рисунок 3.10 – Шлях алгоритму Флойда — Воршелла

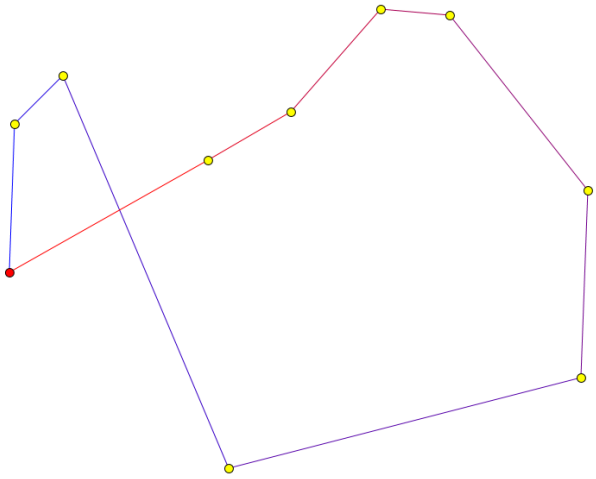


Рисунок 3.11 – Шлях мурашиного алгоритму

За результатами другого порівняння в рисунках 3.12, 3.13 наведено шлях доставки 10 препаратів алгоритмом нейронної меержі та алгоритмом дерева рішень.

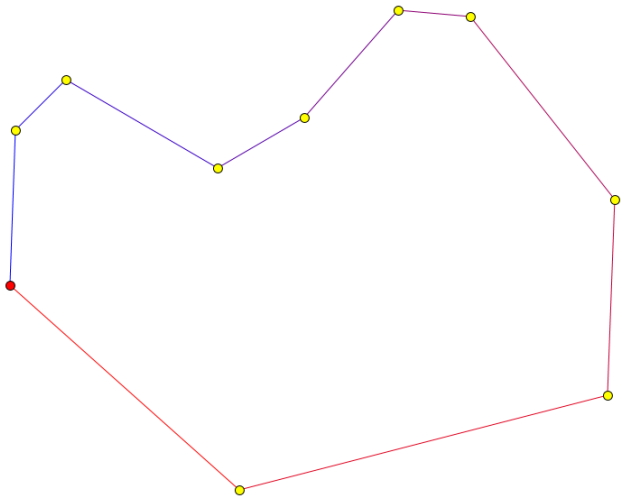


Рисунок 3.12 – Шлях нейронної мережі

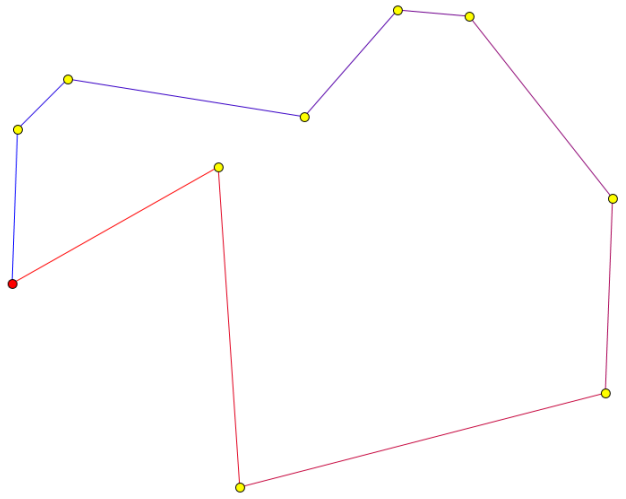


Рисунок 3.13 – Шлях алгоритму дерева рішень

В таблиці 3.5 наведено дистанцію та час доставки 10 препаратів алгоритмами. Якщо проаналізувати результати доставки 10 препаратів то можна визначити що найкращім був простий генетичний та нейронний алгоритм, потім мурашиний алгоритм, а найгіршим виявився алгоритм Дейкстри.

Таблиця 3.5 – Дистанція та час доставки 10 препаратів алгоритмами

| Критерії порівняння | Простий генетичний алгоритм | Алгоритм Дейкстри | Алгоритм Флойда — Воршелла | Мурашиний алгоритм | Нейронний алгоритм | Алгоритм дерева рішень |
|---|---|---|---|---|---|---|
| Довжина шляху | 8,12м | 9,72м | 9,27м | 9,02м | 8,12м | 9,1м |
| Час доставки | 28,4сек | 34сек | 32,4сек | 31,5сек | 28,42сек | 31,8сек |

### 3.3.3 Порівняння доставки 100 препаратів

В третьому порівняні буде наведено розташування 100 препаратів та побудовано шлях доставки різними алгоритмами. Завдяки цьому порівнянню можна в результаті дізнатися який алгоритм краще адаптується до збільшення препаратів доставки. Розташування препаратів наведено в таблиці 3.6.

Таблиця 3.6 – Розташування 100 препаратів

| Номер препарату | Координата x | Координата y |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 50 | 501 |
| 2 | 76 | 530 |
| 3 | 119 | 444 |
| 4 | 143 | 469 |
| 5 | 163 | 472 |
| 6 | 255 | 525 |
| 7 | 373 | 546 |

Продовження таблиці 3.6

| 1 | 2 | 3 |
|---|---|---|
| 8 | 411 | 539 |
| 9 | 309 | 489 |
| 10 | 297 | 498 |
| 11 | 279 | 501 |
| 12 | 279 | 480 |
| 13 | 267 | 424 |
| 14 | 272 | 341 |
| 15 | 271 | 356 |
| 16 | 183 | 258 |
| 17 | 214 | 225 |
| 18 | 290 | 288 |
| 19 | 297 | 291 |
| 20 | 285 | 327 |
| 21 | 366 | 280 |
| 22 | 372 | 296 |
| 23 | 410 | 304 |
| 24 | 406 | 346 |
| 25 | 443 | 324 |
| 26 | 460 | 355 |
| 27 | 481 | 381 |
| 28 | 424 | 395 |
| 29 | 414 | 74 |
| 30 | 402 | 384 |
| 31 | 356 | 406 |
| 32 | 348 | 455 |
| 33 | 369 | 460 |
| 34 | 474 | 421 |

Продовження таблиці 3.6

| 1 | 2 | 3 |
|---|---|---|
| 35 | 506 | 447 |
| 36 | 485 | 486 |
| 37 | 497 | 487 |
| 38 | 559 | 542 |
| 39 | 603 | 481 |
| 40 | 637 | 507 |
| 41 | 664 | 509 |
| 42 | 652 | 534 |
| 43 | 784 | 721 |
| 44 | 737 | 510 |
| 45 | 714 | 505 |
| 46 | 708 | 471 |
| 47 | 662 | 474 |
| 48 | 636 | 406 |
| 49 | 701 | 352 |
| 50 | 734 | 313 |
| 51 | 718 | 306 |
| 52 | 706 | 251 |
| 53 | 720 | 240 |
| 54 | 736 | 168 |
| 55 | 732 | 153 |
| 56 | 712 | 126 |
| 57 | 710 | 101 |
| 58 | 684 | 103 |
| 59 | 678 | 109 |
| 60 | 676 | 145 |
| 61 | 683 | 166 |

Продовження таблиці 3.6

| 1 | 2 | 3 |
|---|---|---|
| 62 | 641 | 144 |
| 63 | 634 | 157 |
| 64 | 617 | 262 |
| 65 | 581 | 340 |
| 66 | 533 | 389 |
| 67 | 520 | 363 |
| 68 | 553 | 270 |
| 69 | 471 | 217 |
| 70 | 543 | 144 |
| 71 | 580 | 89 |
| 72 | 580 | 60 |
| 73 | 566 | 50 |
| 74 | 510 | 51 |
| 75 | 481 | 81 |
| 76 | 464 | 77 |
| 77 | 415 | 102 |
| 78 | 405 | 115 |
| 79 | 391 | 87 |
| 80 | 371 | 55 |
| 81 | 344 | 130 |
| 82 | 314 | 132 |
| 83 | 320 | 182 |
| 84 | 204 | 121 |
| 85 | 177 | 149 |
| 86 | 116 | 51 |
| 87 | 124 | 144 |
| 88 | 113 | 171 |

Продовження таблиці 3.6

| 1 | 2 | 3 |
|---|---|---|
| 89 | 80 | 165 |
| 90 | 100 | 200 |
| 91 | 52 | 227 |
| 92 | 68 | 235 |
| 93 | 136 | 245 |
| 94 | 177 | 204 |
| 95 | 181 | 236 |
| 96 | 170 | 275 |
| 97 | 116 | 288 |
| 98 | 63 | 281 |
| 99 | 32 | 287 |
| 100 | 90 | 395 |

За результатами третього порівняння в рисунках 3.13 - 3.16 наведено шлях доставки 100 препаратів генетичним алгоритмом, алгоритмом Дейкстри мережі, алгоритмом Флойда — Воршелла, та мурашиним алгоритмом.

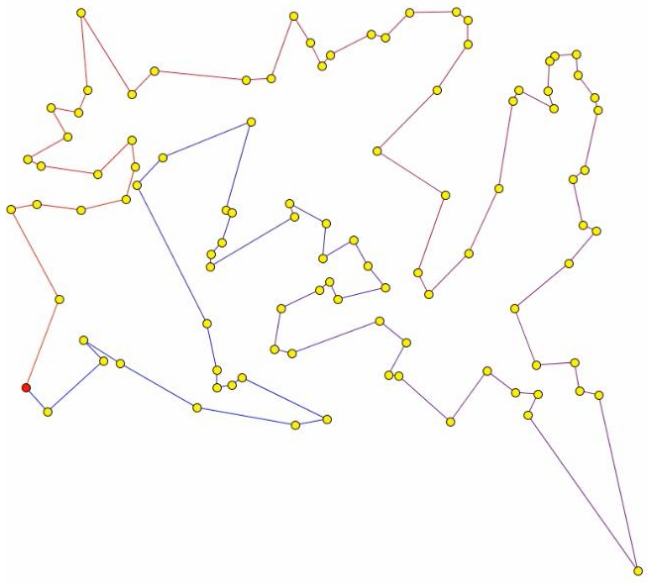


Рисунок 3.13 – Шлях генетичного алгоритму

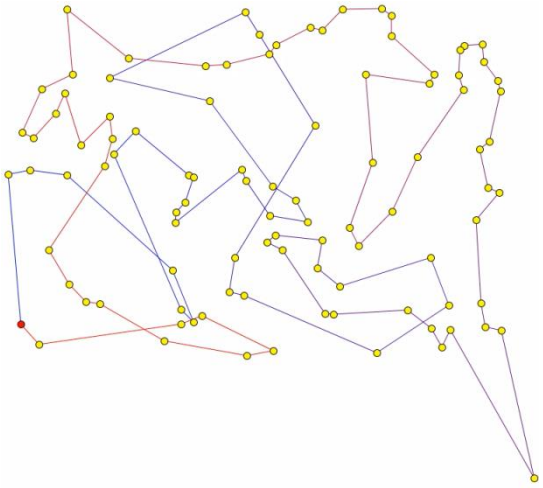


Рисунок 3.14 – Шлях алгоритму Дейкстри

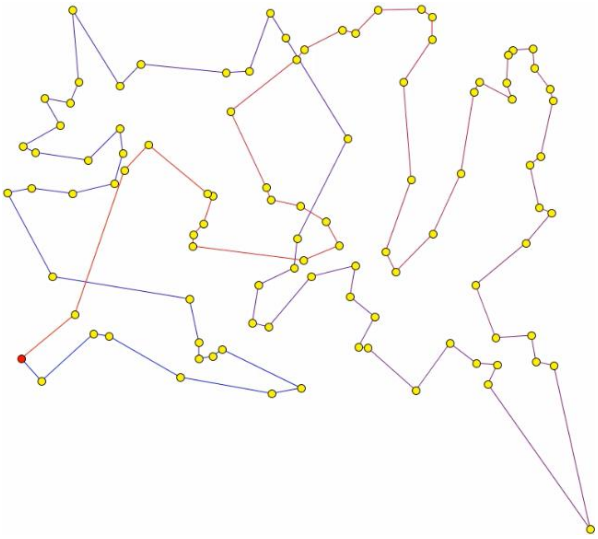


Рисунок 3.15 – Шлях алгоритму Флойда — Воршелла

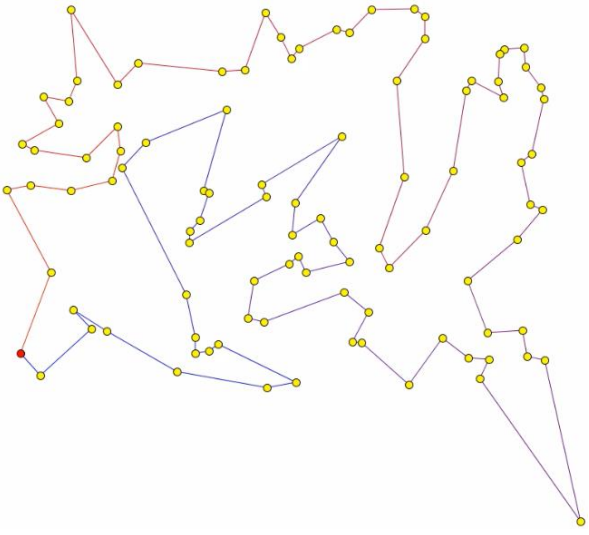


Рисунок 3.16 – Шлях мурашиного алгоритму

За результатами третього порівняння в рисунках 3.17, 3.18 наведено шлях доставки 100 препаратів алгоритмом нейронної меержі та алгоритмом дерева рішень.

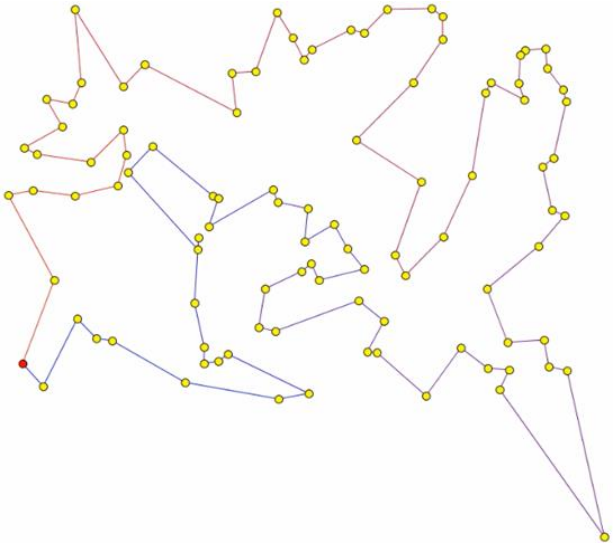


Рисунок 3.17 – Шлях нейронної мережі

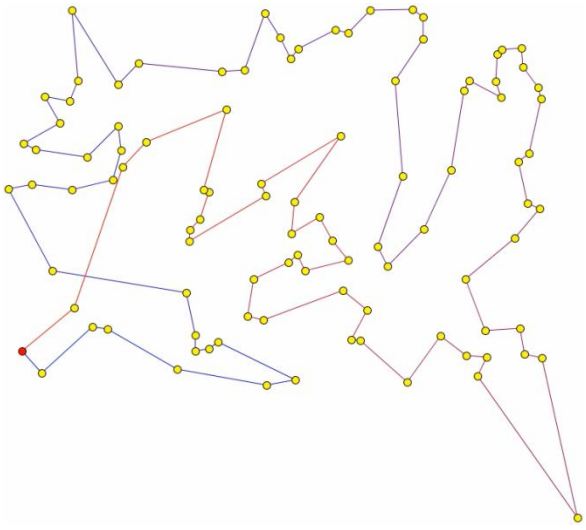


Рисунок 3.18 – Шлях алгоритму дерева рішень

В таблиці 3.7 наведено дистанцію та час доставки 100 препаратів алгоритмами. Якщо проаналізувати результати доставки 100 препаратів то можна визначити що найкращім був нейронний алгоритм, з невеликим відривом по результатам виявився простий генетичний алгоритм, потім мурашиний алгоритм, а найгіршим виявився алгоритм Дейкстри.

Таблиця 3.7 – Дистанція та час доставки 100 препаратів алгоритмами

| Критерії порівняння | Простий генетичний алгоритм | Алгоритм Дейкстри | Алгоритм Флойда — Воршелла | Мурашиний алгоритм | Нейронний алгоритм | Алгоритм дерева рішень |
|---|---|---|---|---|---|---|
| Довжина шляху | 22,27м | 28,55м | 23,89м | 22,79м | 21,94м | 23,19м |
| Час доставки | 77,9сек | 99,9сек | 89,6сек | 79,7сек | 76,7сек | 81,1сек |

### 3.3.4 Порівняння доставки 10 препаратів з переповненням каретки

В другому порівняні буде наведено розташування 10 препаратів та побудовано шлях доставки різними алгоритмами з урахуванням ваги та об’єму препаратів. Завдяки цьому порівнянню можна в результаті дізнатися який алгоритм краще працює в умовах приближених до реальних. Розташування препаратів наведено в таблиці 3.8.

Таблиця 3.8 – Розташування 10 препаратів

| Номер препарату | Координата x | Координата y |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 50 | 501 |
| 2 | 304 | 728 |
| 3 | 560 | 203 |
| 4 | 376 | 315 |
| 5 | 280 | 317 |
| 6 | 112 | 273 |

Продовження таблиці 3.8

| 1 | 2 | 3 |
|---|---|---|
| 7 | 480 | 196 |
| 8 | 712 | 623 |
| 9 | 720 | 406 |
| 10 | 56 | 329 |

За результатами четвертого порівняння в рисунках 3.19 - 3.23 наведено шлях доставки 10 препаратів генетичним алгоритмом, алгоритмом Дейкстри мережі, алгоритмом Флойда — Воршелла, мурашиним алгоритмом та алгоритмом нейронної мережі, з урахуванням переповнення каретки.

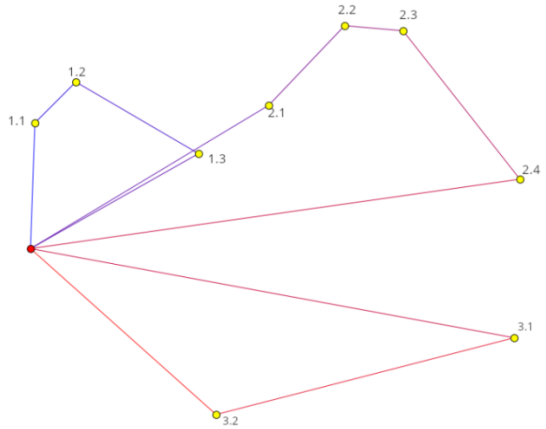



Рисунок 3.19 – Шлях генетичного алгоритму

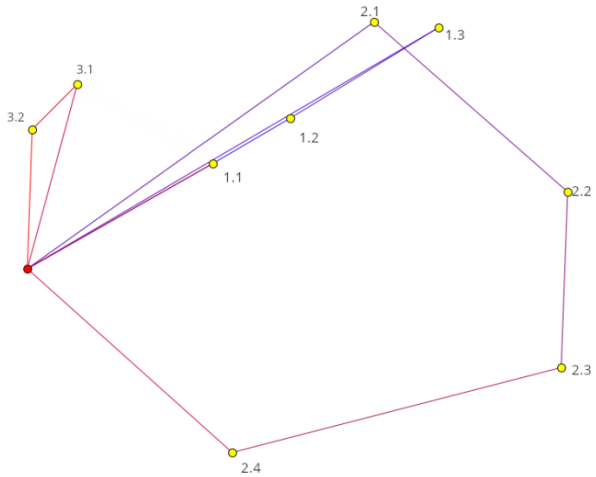



Рисунок 3.20 – Шлях алгоритму Дейкстри

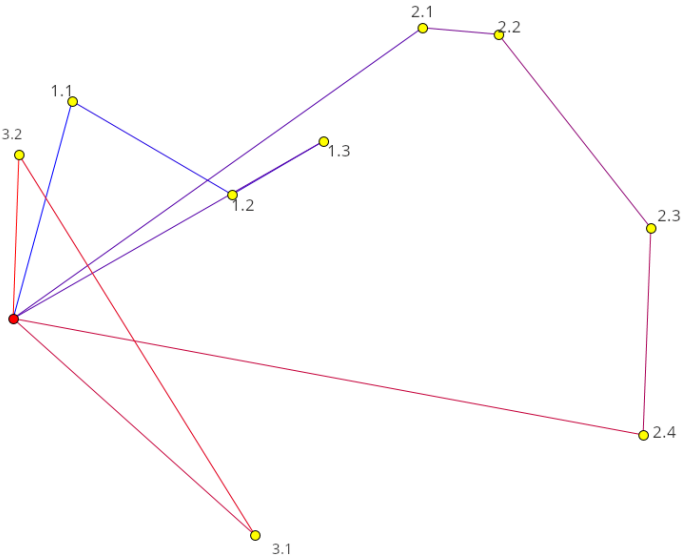



Рисунок 3.21 – Шлях алгоритму Флойда — Воршелла

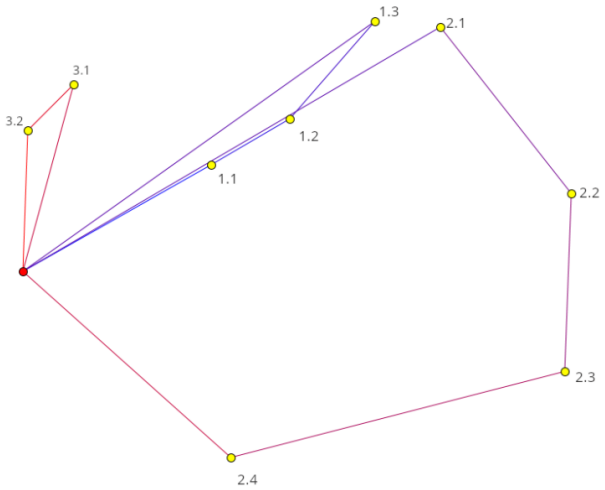



Рисунок 3.22 – Шлях мурашиного алгоритму

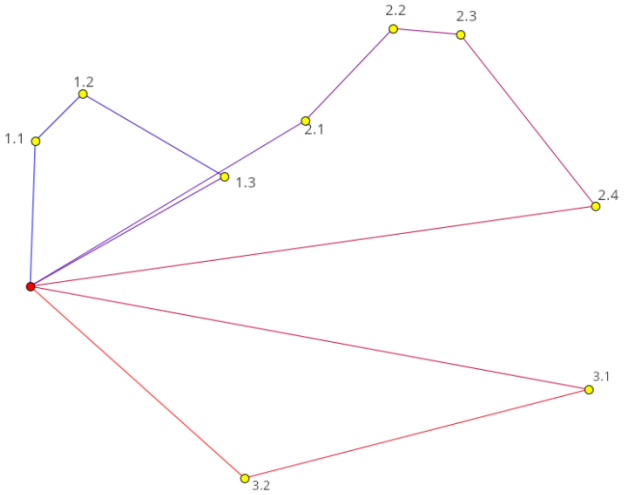



Рисунок 3.23 – Шлях нейронної мережі

За результатами четвертого порівняння в рисунку 3.24 наведено шлях доставки 10 препаратів алгоритмом дерева рішень, з урахуванням переповнення каретки.

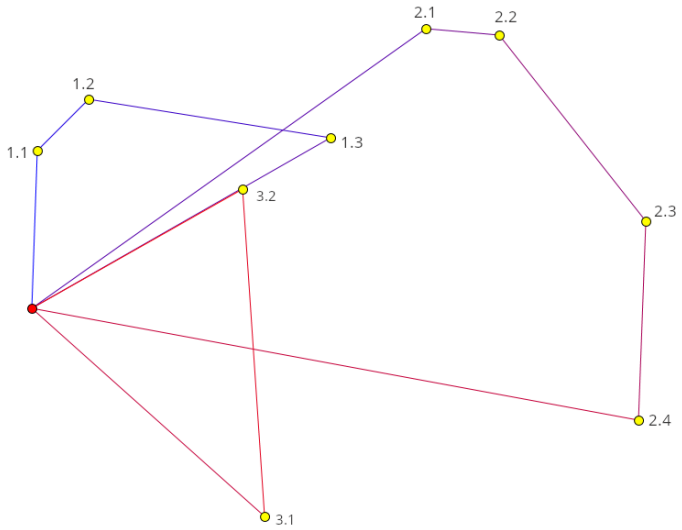



Рисунок 3.24 – Шлях алгоритму дерева рішень

В таблиці 3.9 наведено дистанцію та час доставки 10 препаратів алгоритмами з урахуванням переповнення каретки.

Таблиця 3.9 – Дистанція та час доставки 10 препаратів алгоритмами з урахуванням переповнення каретки

| Критерії порівняння | Простий генетичний алгоритм | Алгоритм Дейкстри | Алгоритм Флойда — Воршелла | Мурашиний алгоритм | Нейронний алгоритм | Алгоритм дерева рішень |
|---|---|---|---|---|---|---|
| Довжина шляху | 13,12м | 14,02м | 13,87м | 13,27м | 13,12м | 13,95м |
| Час доставки | 45,9сек | 49,1сек | 48,5сек | 46,4сек | 45,9сек | 48,8сек |

Якщо проаналізувати результати доставки 10 препаратів з урахуванням їх маси та об’єму то можна визначити що найкращім був простий генетичний(45,9сек) та нейронний алгоритм(45,9сек), потім мурашиний алгоритм(46,4сек), а найгіршим виявився алгоритм Дейкстри(49,1сек). Проаналізувавши результати з другим порівнянням то можна зауважити що алгоритм дерева рішень виявився гіршим за алгоритм Флойда — Воршелла.

### 3.3.5 Порівняння доставки 100 препаратів з переповненням каретки

В другому порівняні буде наведено розташування 100 препаратів та побудовано шлях доставки різними алгоритмами з урахуванням ваги та об’єму препаратів. Завдяки цьому порівнянню можна в результаті дізнатися який алгоритм краще працює зі значно більшою кількістю препаратів в умовах приближених до реальних. Розташування препаратів наведено в таблиці 3.10.

Таблиця 3.10 – Розташування 100 препаратів

| Номер препарату | Координата x | Координата y |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 50 | 501 |
| 2 | 76 | 530 |
| 3 | 119 | 444 |
| 4 | 143 | 469 |
| 5 | 163 | 472 |
| 6 | 255 | 525 |
| 7 | 373 | 546 |
| 8 | 411 | 539 |
| 9 | 309 | 489 |
| 10 | 297 | 498 |
| 11 | 279 | 501 |

Продовження таблиці 3.10

| 1 | 2 | 3 |
|---|---|---|
| 12 | 279 | 480 |
| 13 | 267 | 424 |
| 14 | 272 | 341 |
| 15 | 271 | 356 |
| 16 | 183 | 258 |
| 17 | 214 | 225 |
| 18 | 290 | 288 |
| 19 | 297 | 291 |
| 20 | 285 | 327 |
| 21 | 366 | 280 |
| 22 | 372 | 296 |
| 23 | 410 | 304 |
| 24 | 406 | 346 |
| 25 | 443 | 324 |
| 26 | 460 | 355 |
| 27 | 481 | 381 |
| 28 | 424 | 395 |
| 29 | 414 | 74 |
| 30 | 402 | 384 |
| 31 | 356 | 406 |
| 32 | 348 | 455 |
| 33 | 369 | 460 |
| 34 | 474 | 421 |
| 35 | 506 | 447 |
| 36 | 485 | 486 |
| 37 | 497 | 487 |
| 38 | 559 | 542 |

Продовження таблиці 3.10

| 1 | 2 | 3 |
|---|---|---|
| 39 | 603 | 481 |
| 40 | 637 | 507 |
| 41 | 664 | 509 |
| 42 | 652 | 534 |
| 43 | 784 | 721 |
| 44 | 737 | 510 |
| 45 | 714 | 505 |
| 46 | 708 | 471 |
| 47 | 662 | 474 |
| 48 | 636 | 406 |
| 49 | 701 | 352 |
| 50 | 734 | 313 |
| 51 | 718 | 306 |
| 52 | 706 | 251 |
| 53 | 720 | 240 |
| 54 | 736 | 168 |
| 55 | 732 | 153 |
| 56 | 712 | 126 |
| 57 | 710 | 101 |
| 58 | 684 | 103 |
| 59 | 678 | 109 |
| 60 | 676 | 145 |
| 61 | 683 | 166 |
| 62 | 641 | 144 |
| 63 | 634 | 157 |
| 64 | 617 | 262 |
| 65 | 581 | 340 |

Продовження таблиці 3.10

| 1 | 2 | 3 |
|---|---|---|
| 66 | 533 | 389 |
| 67 | 520 | 363 |
| 68 | 553 | 270 |
| 69 | 471 | 217 |
| 70 | 543 | 144 |
| 71 | 580 | 89 |
| 72 | 580 | 60 |
| 73 | 566 | 50 |
| 74 | 510 | 51 |
| 75 | 481 | 81 |
| 76 | 464 | 77 |
| 77 | 415 | 102 |
| 78 | 405 | 115 |
| 79 | 391 | 87 |
| 80 | 371 | 55 |
| 81 | 344 | 130 |
| 82 | 314 | 132 |
| 83 | 320 | 182 |
| 84 | 204 | 121 |
| 85 | 177 | 149 |
| 86 | 116 | 51 |
| 87 | 124 | 144 |
| 88 | 113 | 171 |
| 89 | 80 | 165 |
| 90 | 100 | 200 |
| 91 | 52 | 227 |
| 92 | 68 | 235 |

Продовження таблиці 3.10

| 1 | 2 | 3 |
|---|---|---|
| 93 | 136 | 245 |
| 94 | 177 | 204 |
| 95 | 181 | 236 |
| 96 | 170 | 275 |
| 97 | 116 | 288 |
| 98 | 63 | 281 |
| 99 | 32 | 287 |
| 100 | 90 | 395 |

Через значно більший за довжиною маршрут, відображення шляху доставки препаратів не буде зрозумілим та доцільним.

В таблиці 3.11 наведено дистанцію та час доставки 100 препаратів алгоритмами з урахуванням переповнення каретки.

Таблиця 3.11 – Дистанція та час доставки 100 препаратів алгоритмами з урахуванням переповнення каретки.

| Критерії порівняння | Простий генетичний алгоритм | Алгоритм Дейкстри | Алгоритм Флойда — Вершелла | Мурашиний алгоритм | Нейронний алгоритм | Алгоритм дерева рішень |
|---|---|---|---|---|---|---|
| Довжина шляху | 43,45м | 55,61м | 47,09м | 43,5м | 42,51м | 45,98м |
| Час доставки | 152сек | 194,6сек | 164,8сек | 152,3сек | 148,8сек | 161сек |

Якщо проаналізувати результати доставки 100 препаратів з урахуванням їх маси та об’єму то можна визначити що найкращім був нейронний алгорим(148,8сек), за невеликим відривом за результатами йде простий генетичний алгоритм(152сек), потім мурашиний алгоритм(152,3сек), а найгіршим виявився алгоритм Дейкстри(194,6сек).

За результатами цих порівнянь можна дійти до висновку що найкращім виявився нейронний алгоритм, за ним з невеликою різницею ефективності виявився простий генетичний алгоритм, алгоритм Дейкстри виявився більш ефективним за невеликої кількості препаратів, але став найгіршим в доставці великої кількості препаратів, алгоритм Флойда — Воршелла за результатами виявився повною протилежністю алгоритму Дейкстри, за великої кількості препаратів ефективність зростає, а за малої падає, мурашиний алгоритм та алгоритм дерева рішень показали гарні результати при доставці порівняно невеликої кількості препаратів та порівняно середній результат при обробці більшої кількості препаратів.

### 3.4 Висновки за розділом

В даному розділі було сформульовано основні функціональні вимоги до ПП, докладно описано та визначено пріоритети кожного з них. Наведено схему структурних варіантів використання користувачем. Було описано всю архітектуру ПП, детально описано всі класи та методи до них.

Також було проведено ряд порівнянь алгоритмів доставки за якими було визначено найефективніший алгоритм.

# 4 КЕРІВНИЦТВО ПРОГРАМІСТА

## 4.1 Призначення та умови застосування програми

### 4.1.1 Призначення програми

Під час відвідування аптеки виникає проблема в великих чергах за певними ліками, на допомогу приходять роботи-фармацевти вони пришвидшують роботу аптеки, але їх проблемою є неоптимальний вибір шляху пошуку ліків у своїй системі та неможливість доставки декількох препаратів одночасно.

Розв'язання даної проблеми є впровадження нових алгоритмів пошуку та доставки ліків до користувача.

### 4.1.2 Функції програми

Основними функціями програми є:

- надання можливості додавання препаратів до роботу;
- пошук препаратів у базі робота;
- доставка препаратів роботом та вивід часу доставки.

## 4.2 Умови, необхідні для виконання програми

### 4.2.1 Обсяг оперативної пам’яті

Для виконання всіх функції, програми «AptRobot» потрібно не менше 100 Мб оперативної пам’яті. Проте, через те, що функціонування операційної системи, яка потрібна для запуску програми, потребує як мінімум 2 Гб оперативної пам’яті, рекомендовано використовувати програму «AptRobot» на ПК на яких більш, ніж 2 Гб ОЗП.

### 4.2.2 Вимоги до складу периферійних пристроїв

Програма «AptRobot» не потребує особливих вимог до периферійних пристроїв.

### 4.2.3 Вимоги до програмного забезпечення

Системні програмні засоби, які використовуються програмою «AptRobot» – Microsoft Visual Studio – представлені у Windows 7 або пізнішій / (K)Ubuntu Linux 16.04 (64-bit) / macOS 10.13 або пізнішій.

### 4.2.4 Вимоги до програміста

Вимоги до програміста – знання роботи з мовою програмування C#, .xml файлами, з xml базою даних, та розуміння роботи алгоритмів.

У задачі, які виконуються програмістом, входять такі пункти:

- підтримка працездатності технічних засобів;
- підтримка працездатності системних засобів.

## 4.3 Характеристика програми

### 4.3.1 Опис основних характеристик програми

Програма «AptRobot» працює у ручному режимі, тобто користувач повинен запускати програму за необхідністю.

Працездатність програми «AptRobot» перевіряється лише під час роботи. Якщо протягом роботи з програмою були введені помилкові дані то програма виведе повідомлення про помилку.

### 4.3.2 Опис основних особливостей програми

Програма «AptRobot» має зрозумілий інтерфейс та дозволяє задіяти всі алгоритми програми. У користувача є можливість додавання препаратів до роботу , знайти препарат у базі віртуального робота та доставити один чи декілька препаратів роботом з можливістю вибору алгоритму та виведенням часу доставки.

## 4.4 Звертання програми

### 4.4.1 Завантаження й запуск програми

Завантаження й запуск програми відбувається шляхом виконання користувачем .exe-файлу.

### 4.4.2 Виконання програми

Програма завантажується ОС до оперативної пам’яті, та починає своє виконання з класу FormRobot.

## 4.5 Вхідні та вихідні дані

Вхідні дані – текст: назва, вага, розміри, серія препаратів.

Вихідні дані – підтвердження додавання препаратів, відображення часу доставки препаратів, повідомлення про помилку.

### 4.5.1 Організація початкової інформації, що використовується

Програма «AptRobot» протягом всієї роботи використовує введені в текстове поле чи базу даних дані.

### 4.5.2 Організація вихідної інформації, що використовується

Програма «AptRobot» в ході своєї роботи використовує вивід на екран інформації про результати введених користувачем запитів.

## 4.6 Повідомлення

Програма «AptRobot» може вивести помилки пов’язані з неправильним вводом серії препарату (рис. 4.1).

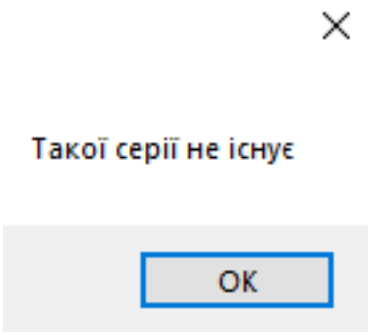



Рисунок 4.1 – Повідомлення про неправильно введену серію

## 4.7 Висновки за розділом

У цьому розділі наведено детальний опис призначення та функцій розробленого ПП. Визначено умови, необхідні для його встановлення та обслуговування на комп’ютері користувача. Описано основні характеристики та особливості створеного програмного забезпечення. Розроблене програмне забезпечення поставляється з відкритим вихідним кодом, тому його можуть редагувати інші розробники програмного забезпечення, а його модулі можна використовувати для створення інших систем. У цьому розділі також наведено опис вхідних і вихідних даних і список повідомлень, що генеруються під час роботи.

# 5 КЕРІВНИЦТВО ОПЕРАТОРА

## 5.1 Призначення програми

Під час відвідування аптеки виникає проблема в великих чергах за певними ліками, на поміч приходять роботи-фармацевти вони пришвидшують роботу аптеки, але їх проблемою є неоптимальний вибір шляху пошуку ліків у своїй системі, дана програма вирішує цю проблему.

## 5.2 Умови виконання програми

Програма споживає 100 Мб ОЗУ та 15 Мб місця на жорсткому диску.

Мінімальні системні вимоги, необхідні для роботи програми:

– операційна система: Windows 7 або пізніша;
– 64-розрядний (x64) процесор 1,8 ГГц;
– 2 ГБ (для 64-розрядної системи) оперативної пам’яті;
– 1 ГБ вільного місця на жорсткому диску;
– NET Framework 4.7.2 або вища.

## 5.3 Запуск програми

Чат-бот викликається користувачем через розширення .exe, яке знаходиться за адресою AptRobot\AptRobot\bin\Debug\ AptRobot.exe (рис. 5.1) або через середовище розробки Visual Studio 2019 та вище, за допомогою файлу в головній теці програми з розширенням .sln, яке має встановлений пакет інструментів для програмування на мові C#.

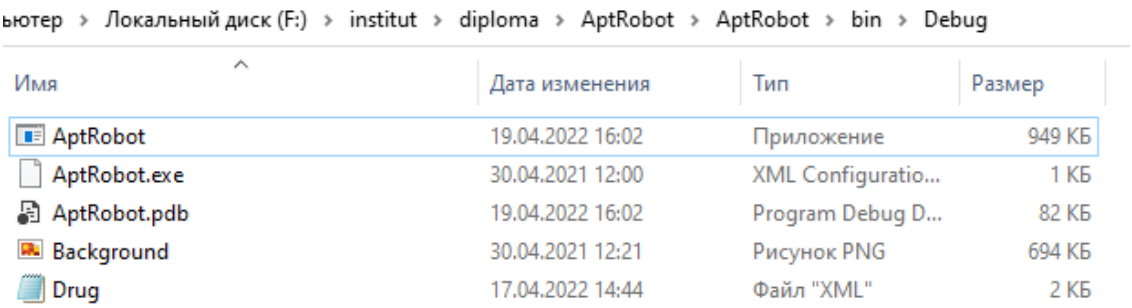



Рисунок 5.1 – Місце розташування файлу AptRobot.exe

### 5.4 Виконання програми

Для того, щоб запустити чат-бота його потрібно запустити файл з розширенням .exe.

Далі, натиснути на кнопку, в залежності від потрібності користувача. (рис. 5.2).

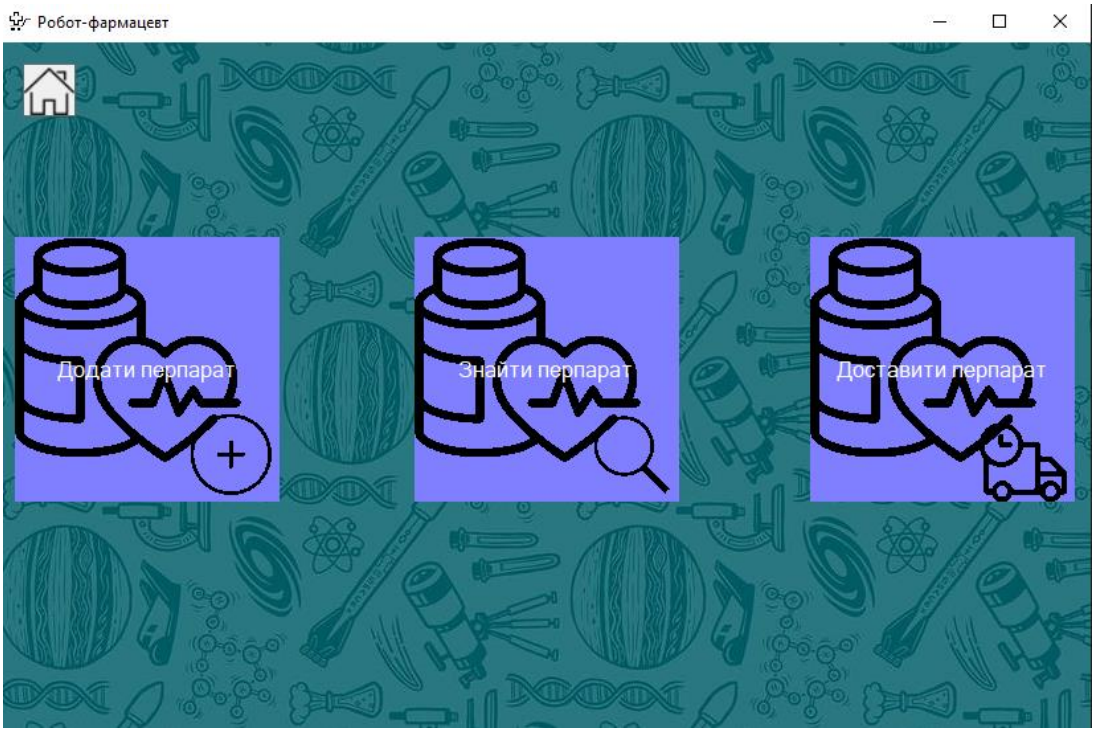



Рисунок 5.2 – Головна форма

Далі користувач натискає на одну з кнопок: “Додати препарат”, “Знайти препарат” , “Доставити препарат”.

Після натискання на кнопку “Додати препарат” користувач потрапляє до меню вводу даних препарату (рис. 5.3).

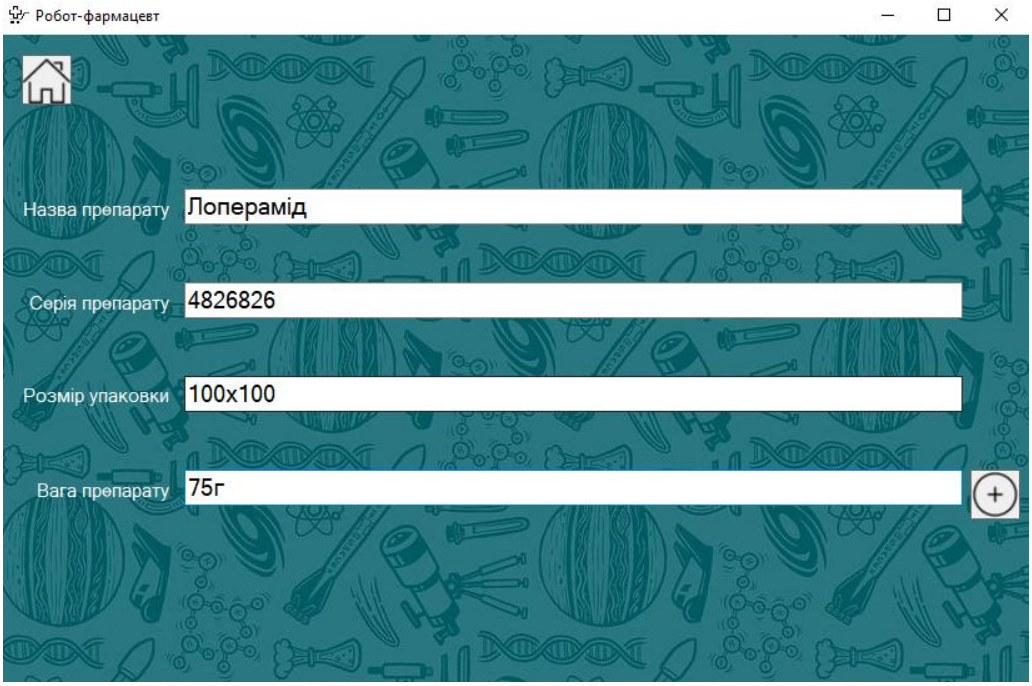



Рисунок 5.3 – Форма додавання нового препарату

Після вводу даних про препарат потрібно натиснути кнопку додавання препарату до бази даних. Якщо додавання пройшло успішно то буде виведене повідомлення про успішний ввід та повернемося на головну форму (рис. 5.4).

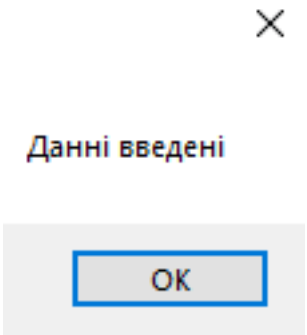



Рисунок 5.4 – Повідомлення про успішне введення даних

Якщо користувач натиснув на кнопку “Знайти препарат”, то відкриється меню вводу серії препарату (рис. 5.5).

Рисунок 5.5 – Форма пошуку

Коли користувач введе серійний номер, треба натиснути на кнопку пошуку, після чого з`явиться вся інформація про потрібний препарат (рис. 5.6).

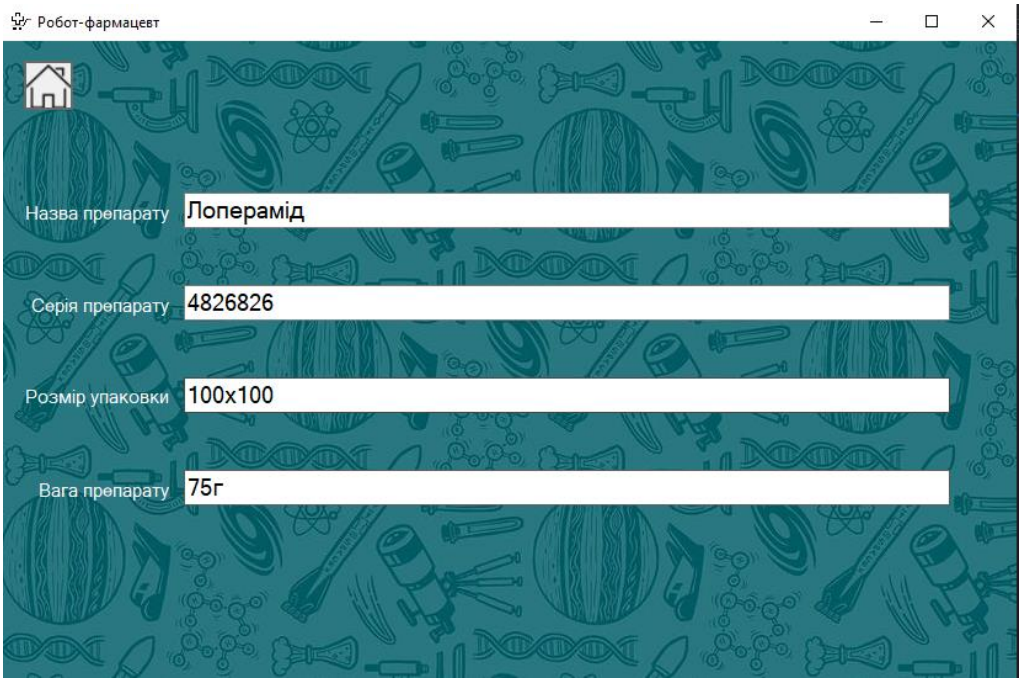



Рисунок 5.6 – Результат пошуку

Для виходу на головну сторінку користувач може використати кнопку додому.

Після натискання на кнопку “Доставити препарат” буде відкрито форму доставки препарату де треба ввести серійний номер потрібного препарату та за бажанням можна вибрати алгоритм доставки (за замовчуванням використовується алгоритм Дейкстри) чи збільшити кількість препаратів також можна вивести порівняння часу доставки всіх алгоритмів (рис. 5.7).

Рисунок 5.7 – Форма доставки

Після введення інформації треба натиснути на кнопку доставки препарату, як результат буде виведено повідомлення про час за який препарати були доставлені (рис. 5.8).

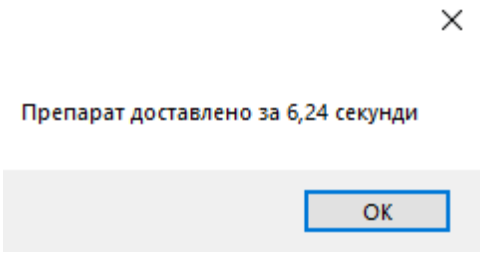



Рисунок 5.8 – Час доставки препарату

Якщо користувач після введення інформації натисне на кнопку порівняння часу доставки то буде виведено повідомлення про час доставки всіх алгоритмів (рис. 5.9).

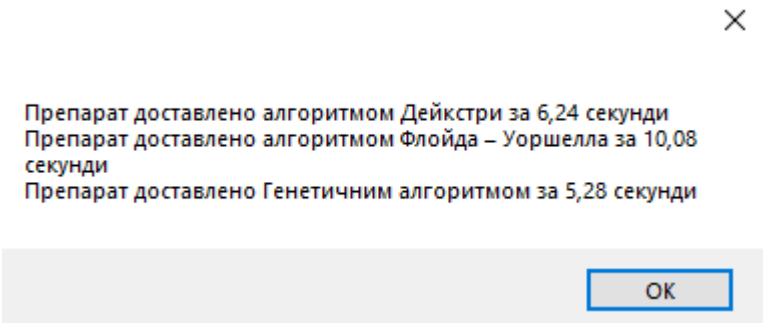



Рисунок 5.9 – Порівняння часу доставки

## 5.5 Повідомлення оператору

ПОВІДОМЛЕННЯ: «Данні введені.»

ДІЯ ОПЕРАТОРА: Натиснути кнопку “ок”.

ПОВІДОМЛЕННЯ: «Препарат доставлено за *** секунди.»

ДІЯ ОПЕРАТОРА: Натиснути кнопку “ок”.

ПОВІДОМЛЕННЯ: «Препарат доставлено алгоритмом Дейкстри за *** секунди.

Препарат доставлено алгоритмом Флойда-Уоршелла за *** секунди.

Препарат доставлено Генетичним алгоритмом за *** секунди.»

ДІЯ ОПЕРАТОРА: Натиснути кнопку “ок”.

## 5.6 Висновок

У цьому розділі описано призначення та функції розробленого програмного забезпечення та мінімально необхідні апаратні умови для його встановлення та використання. Дано детальну інструкцію щодо роботи оператора з ПП, а саме приклад додавання, пошуку та доставки препаратів до

користувача. Наведено перелік повідомлень, що формуються під час роботи ПП, а також приклад вхідних та вихідних даних. Користувальницький інтерфейс розробленого програмного забезпечення простий та інтуїтивно зрозумілий і не потребує додаткового навчання оператора.

# ВИСНОВКИ

У результаті виконання роботи було розглянуто проблематику поставленого завдання, наведено його актуальність та доцільність. Було досліджено існуючих роботів-фармацевтів, наведено алгоритми вирішення завдання, та поставлено завдання до роботи.

Далі було обґрунтовано інструментарій вирішення завдання такий як: мова програмування, середовище розробки, також було наведено структурну схему програмного продукту. В третьому розділі було наведено функціональні вимоги до програмного продукту, описано архітектуру та наведено багато порівнянь за результатами який вдалося дізнатися адаптивність алгоритмів до певних характеристик поставленої задачі.

В результаті було розроблено ПП який впроваджує новітні алгоритми пошуку найкоротшого шляху, завдяки чому вдалося зменшити доставку одного препарату роботом фармацевтом на 22%, також вдалося реалізувати одночасну доставку багатьох препаратів. У програмі реалізовані поставлені задачі, завдяки використанню мови програмування C#, середовища розробки Visual Studio Community 2019 та xml бази даних.

Було розроблено шість класів, які використовують основні методи об’єктно-орієнтованого програмування. Такі класи надають можливість записувати та вилучати дані з БД, взаємодіяти зі всіма компонентами програми, викликати методи вирішення завдання, зберігати дані про положення препаратів та надавати положення препаратів.

Програма має простий та інтуїтивно зрозумілий інтерфейс, який має декілька варіантів запитів користувача. Користувач може додати, знайти, доставити потрібний йому препарат, також може вибрати алгоритм доставки та вивести порівняння часу доставки завдяки взаємодії з розробленою програмою.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Робот для аптек Consis - автоматизация хранения медикаментов [Электрон. ресурс]. – Режим доступа: https://www.design-vector.ru/uslugi/robotizaciya/robotizaciya-apteki-consis/.

2. PharmBot - автоматизация аптек и аптечных сетей [Электрон. ресурс]. – Режим доступа: http://pharmbot.biz.

3. Pharmacy automation/ Automated dispensing system (ADS) for hospital pharmacies [Electronic resource]. – Access mode: https://newicon.fi/pharmacy-automation-in-pharmacies/automated-dispensing-system.

4. Ткаченко Ольга, Плісенко Владислав. Візуалізація та інтерактивні мультимедійні технології. 2020. — С. 36.

5. Аптечный робот - Wikiwand [Электрон. ресурс]. – Режим доступа: https://www.wikiwand.com/ru/Аптечный_робот.

6. Инновационный аптечный робот ISO-FIXU [Электрон. ресурс]. – Режим доступа: https://kiosksoft.ru/news/2014/10/30/3932.

7. Ананий В. Левитин (2006). Глава 3. Метод грубой силы: Задача коммивояжера. Алгоритмы: введение в разработку и анализ (Introduction to The Design and Analysis of Aigorithms). М.: «Вильямс». с. 159–160. ISBN 0-201-74395-7.

8. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е изд. — М.: «Вильямс», 2006. — С. 1296. — ISBN 0-07-013151-1.

9. Алгоритмы нахождения кратчайших путей на графах [Электрон. ресурс]. – Режим доступа: https://studref.com/370998/informatika/algoritmy_nahozhdeniya_kratchayshih_putey_grafah.

10. 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.

11. Олійник, А. О. Інтелектуальний аналіз даних / А. О. Олійник, С. О. Субботін, О. О. Олійник. – Запоріжжя : ЗНТУ, 2012. – 278 с.

12. Генетичні алгоритми. Ключові поняття і методи реалізації [Електрон. ресурс]. – Режим доступу: http://www.znannya.org/?view=ga_general

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

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

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

16. Либерти Д. Язык программирования C# // Программирование на C#. — Санкт-Петербург. — 2003: Символ-Плюс, 2003. — С. 26. — 688 с. — ISBN 5-93286-038-3.

# ДОДАТОК А

## Технічне завдання

## Вступ

Розроблюваний програмний продукт використовується для підвищення ефективності доставки ліків роботом-фармацевтом, завдяки впровадженню новітніх методів пошуку маршруту.

ПП забезпечує роботу з основними функціями які можуть знадобитися користувачу такі як: доставка, пошук та додавання препаратів до робота. Розроблюваний ПП підходить як для модернізації існуючих роботів так і для нових роботів що є неодмінним плюсом для власників аптек.

### А.1 Мета розробки і її призначення

Об’єкт дослідження – процес прогнозування руху руки маніпулятора робота-фармацевта за ліками.

Предмет дослідження – методи прогнозування результатів процесу доставки ліків роботом-фармацевтом до користувача.

Мета роботи – програмна реалізація методів доставки ліків роботом-фармацевтом до користувача.

Актуальність та проблематика роботи полягає в тому, що через пандемію та воєнні дії на території України збільшився попит на ліки. Внаслідок чого збільшилися черги за ліками та завантаженість як самих фармацевтів так і роботів-фармацевтів. У зв’язку з цим збільшився попит на ліки тому потрібно оптимізувати роботу роботів-фармацевтів. Це можна зробити за допомогою впровадження нових методів доставки ліків та впровадження можливості одночасно доставки великої кількості ліків.

Принципова відмінність створених методів від існуючих полягає у тому, що запропоновані методи використовують передові алгоритми побудування шляху доставки та мають максимальну адаптацію до специфіки розв'язуваної задачі.

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

Фармацевтичний бізнес - це завжди великий обіг медикаментів і їх облік. Грамотне ведення аптечних роботів допомагають підвищити рентабельність і знизити ризики можливих помилок, а також спростити сам процес роботи, але їх проблемою є неоптимальний вибір шляху пошуку ліків у своїй системі.

Розв'язання даної проблеми є впровадження нових алгоритмів пошуку та доставлення ліків до термінала.

Роботизований аптечний склад або робо склад, являє собою комплекс, в який можуть входити: шафи для зберігання лікарських засобів, роботизовані системи приймання ліків, механізми для переміщення лікарських засобів (маніпулятори, транспортери), електронні системи, які здійснюють управління складами.

Спосіб обслуговування за принципом роботизованої аптеки викликає у споживачів інтерес, підтримує увагу постійних клієнтів і залучає нових покупців в аптеку. Комплекс переваг роботів для аптек створює всі передумови для підвищення числа постійних покупців і збільшення товарообігу.

## А.3 Підстави для розробки

Підставою для виконання розробки є завдання на дипломний проєкт на тему “ Програмна реалізація методів доставки ліків роботом-фармацевтом ”, затверджене наказом по НУ «Запорізька політехніка» № 102 від 27 квітня 2022 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 можливість завантаження даних до БД;

– повинна бути реалізована можливість пошуку препарату в роботі;

– повинна бути реалізована можливість доставки одного препарату;

– повинна бути реалізована можливість багатьох препаратів одночасно;

– повинна бути реалізована можливість вибору алгоритму доставки;

– повинна бути реалізована можливість виводу часу доставки;

– повинна бути реалізована можливість виводу порівняльної характеристики методів доставки;

– повинна бути реалізована можливість обробки помилок під час виконання програми та виведення повідомлення про помилку.

### А.4.2 Вимоги до надійності

ПП повинна виконувати наступні вимоги до надійності:

– захищати ПП від помилкових дій користувача (наприклад користувач ввів невалідні дані);

– обробляти виняткові ситуації (наприклад, відсутність підключення до бази даних).

Програмне забезпечення має функціонувати належним чином під час безперервної роботи комп’ютера. При виході з ладу обладнання відновлення нормальної роботи програмного забезпечення має здійснюватися належним чином після вирішення проблем в системі.

### А.4.3 Умови експлуатації

Умови експлуатації:

– програмне забезпечення не потребує обслуговуючого персоналу;

– для користування програмним забезпеченням не потрібна додаткова кваліфікація.

### А.4.4 Вимоги до складу та параметрів технічних засобів

Програма споживає 100 Мб ОЗУ та 15 Мб місця на жорсткому диску.

Мінімальні системні вимоги, необхідні для роботи програми:

– операційна система: Windows 7 або пізніша;

– 64-розрядний (x64) процесор 1,8 ГГц;

– 2 ГБ (для 64-розрядної системи) оперативної пам’яті;

– 1 ГБ вільного місця на жорсткому диску;

– NET Framework 4.7.2 або вища.

### А.4.5 Вимоги до маркування та упакування

Програмне забезпечення не потребує додаткових дій при маркуванні та упакуванні.

### А.4.6 Вимоги до транспортування та збереження

Вимоги до транспортування й зберігання є аналогічними і до носія на якому записана програма.

### А.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 програмний продукт не потребує встановлення. Програмний продукт необхідно використовувати на комп’ютерах, що відповідають вимогам і параметрам технічних засобів. Програмний продукт може бути використаний користувачем у будь-який час без конфлікту з іншими програмами.

# ДОДАТОК Б

## Опис програмного продукту

## Б.1 Загальні відомості

Назва програмного продукту – «AptRobot».

Розроблений програмний продукт використовується для підвищення ефективності доставки ліків роботом-фармацевтом, завдяки впровадженню новітніх методів пошуку маршруту.

ПП забезпечує роботу з основними функціями які можуть знадобитися користувачу такі як: доставка, пошук та додавання препаратів до робота. Розроблений ПП підходить як для модернізації існуючих роботів так і для нових роботів що є неодмінним плюсом для власників аптек.

Програма споживає 100 Мб ОЗУ та 15 Мб місця на жорсткому диску.

Мінімальні системні вимоги, необхідні для роботи програми:

– операційна система: Windows 7 або пізніша;
– 64-розрядний (x64) процесор 1,8 ГГц;
– 2 ГБ (для 64-розрядної системи) оперативної пам’яті;
– 1 ГБ вільного місця на жорсткому диску;
– NET Framework 4.7.2 або вища.

## 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 можливість завантаження даних до БД;
– повинна бути реалізована можливість пошуку препарату в роботі;
– повинна бути реалізована можливість доставки одного препарату;
– повинна бути реалізована можливість багатьох препаратів одночасно;
– повинна бути реалізована можливість вибору алгоритму доставки;
– повинна бути реалізована можливість виводу часу доставки;

– повинна бути реалізована можливість виводу порівняльної характеристики методів доставки;

– повинна бути реалізована можливість обробки помилок під час виконання програми та виведення повідомлення про помилку.

### Б.3 Опис логічної структури

Розроблений ПП повинен обов’язково містити такі модулі:

– модуль алгоритмів;

– модуль обробки вхідної інформації;

– модуль роботи з базою даних;

– БД;

– графічний інтерфейс.

Розроблений ПП використовує дані про розміщення препаратів та споживачів, а також можливі варіанти розміщення препаратів в роботі, тобто комбінації кількості і типорозмірів робота-фармацевта. Ці дані можуть бути представлені у таких сховищах:

– текстові файли (CSV, txt, xml,…);

– бінарні файли даних;

– не реляційні бази даних (NoSQL сховища);

– реляційні бази даних (MySql, MS SQL Server, Firebird, …) [13].

Обраний тип зберігання повинен забезпечувати достатню гнучкість, високу швидкість отримання даних, здатність зберігати великі об’єми даних, здатність до розширення функціоналу та простоту налаштування. Всім цим вимогам відповідають текстові xml файли. Переваги використання xml файлів:

– підприємство може мати багато XML в стандартному форматі, який існує;

– може знадобитися, щоб дані показали або сприйняли як XML, тому використання іншого формату, таких як реляційних сил подвійного моделювання даних;

– xml дуже добре підходить до розріджених даних, які глибоко захованих даних та змішаного змісту;

– xml є легким для читання людьми, тоді як реляційні таблиці вимагають знань для доступу;

– метадані часто доступні як XML;

– семантичні веб-дані доступні як RDF / XML.

В вибраному середовищі розробки є реалізовані класи та методи для управління текстовими xml файлами, що значно спрощує процес створення ПП.

Всі атрибути файлу та їх опис зазначено в табл. Б.1.

Таблиця Б.1 – Атрибути

| Атрибут | Опис сутності |
|---|---|
| namedrug | Назва препарату |
| seriadrug | Серійний номер препарату |
| sizedrug | Розміри коробки препарату |
| massdrug | Маса препарату |
| nom | Порядковий номер препарату |

Якщо перейти до модулів програмного продукту то можна виділити такі модулі: розрахунковий модуль, графічний інтерфейс та базу даних.

Модуль обробки вхідної інформації – це основний та найважливіший модуль, його призначення це відправка даних для обробки в алгоритми доставки та отримання даних від бази даних.

Модуль роботи з базою даних – це модуль потрібний для записування та вилучення даних про потрібний препарат з бази даних.

Модуль алгоритмів – це модуль де реалізовано основні алгоритми доставки препаратів.

Модуль графічного інтерфейсу – графічний інтерфейс буде розроблено за допомогою технології WinForms яка є інтуїтивно зрозумілою та вбудована в середовищу розробки Microsoft Visual Studio Community 2019.

Для більш наглядного представлення структурної схеми програмного продукту був створений макет розтушування модулів програмного продукту (рис. Б.1).

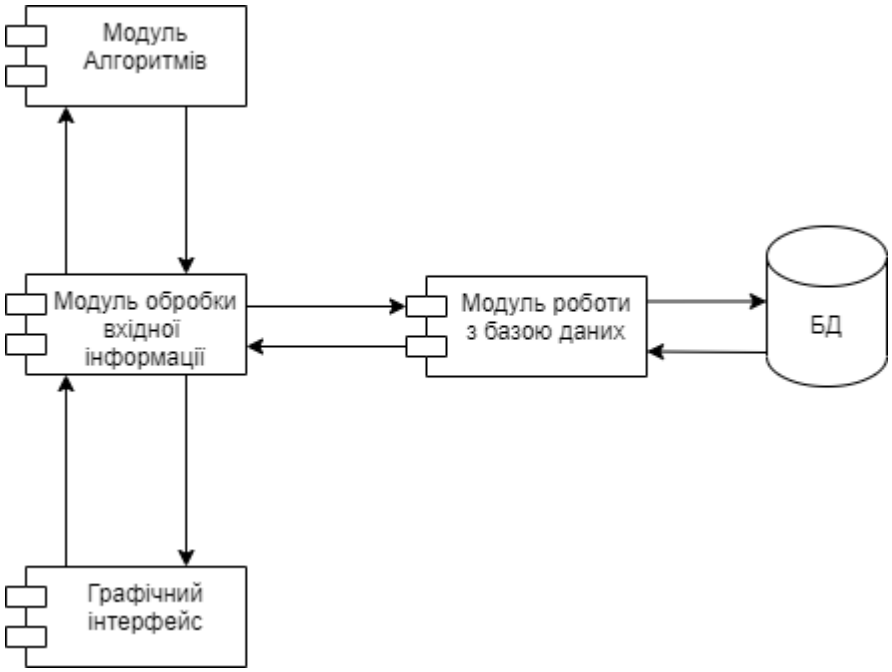



Рисунок Б.1 – Структурна схема програмного продукту

На рисунку Б.2 наведено діаграму класів де відображено назву та розташування методів та алгоритмів в класах.

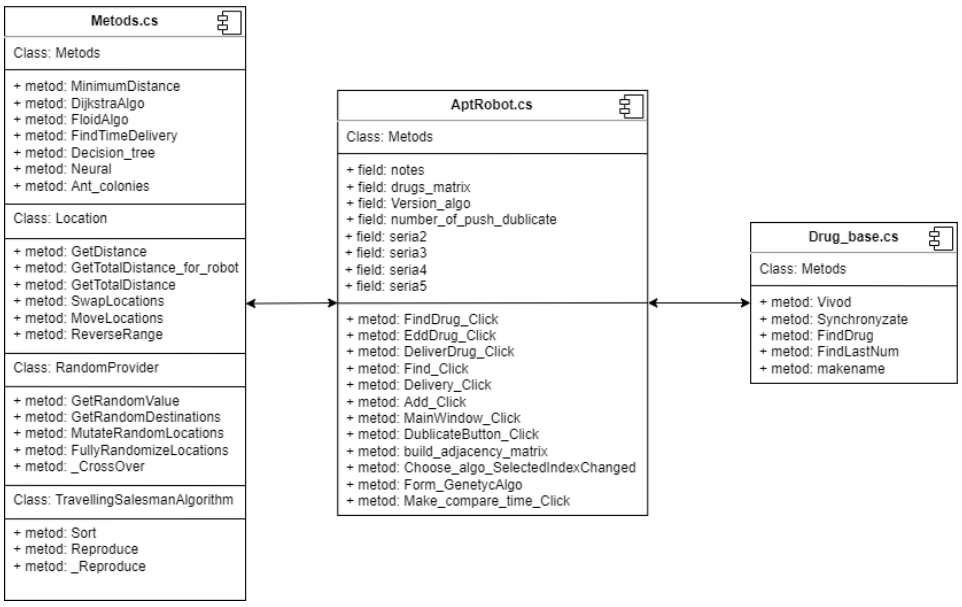



Рисунок Б.2 – Діаграма класів програмного продукту

## Б.4 Вибір інструментарію розробки програмного продукту

Для написання програми було вирішено вибрати мову C#, бо вона має велику кількість бібліотек і вже реалізовані методи з відкритим кодом. Ця мова широко поширена, має велике співтовариство розробників і документацію, яка детально описує реалізацію функцій і допомагає в реалізації будь-яких розробок. Вартість впровадження та підтримки також впливає на вибір, слід зазначити, що мова C# вільна для розповсюдження і проста в обслуговуванні. Серед переваг – ця мова актуальна на ринку, а це означає, що в майбутньому не виникне серйозних проблем з додаванням нових функцій або навіть адаптацією проєкту під нову розробку.

Для розробки ПП було обрано середовище Visual Studio Community 2019, бо воно підтримує функцію підсвітки коду, гарну співпрацю з системою контролю версій Git, мабуть найкращу технологію побудування інтерфейсів WinForms, має красивий та інтуїтивно зрозумілий інтерфейс, підтримує кастомізацію інтерфейсу та багато іншого. Та мабуть головною перевагою є те що Visual Studio була розроблена студією що розроблювала мову програмування C#, що означає що це середовище за замовчуванням підтримує максимальну кількість функцій що може бути закладена в саму мову програмування.

## Б.5 Виклик і завантаження

Чат-бот викликається користувачем через розширення .exe, яке знаходиться за адресою AptRobot\AptRobot\bin\Debug\ AptRobot.exe (рис. Б.3) або через середовище розробки Visual Studio 2019 та вище, за допомогою файлу в головній теці програми з розширенням .sln, яке має встановлений пакет інструментів для програмування на мові C#.

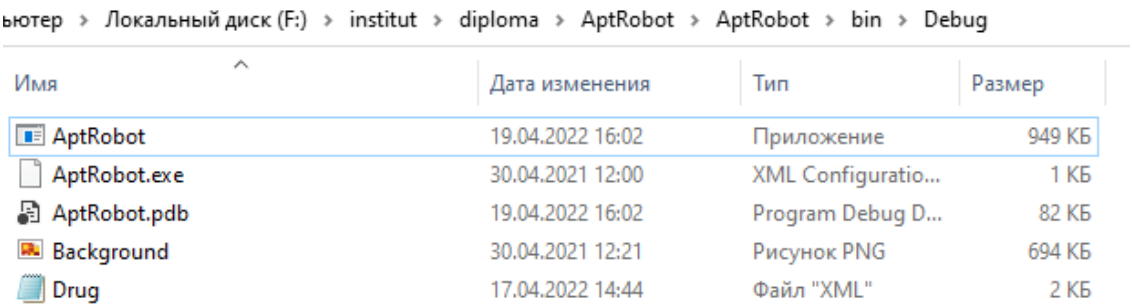



Рисунок Б.3 – Місце розташування файлу AptRobot.exe

## Б.6 Приклади реалізації

Для того, щоб запустити чат-бота його потрібно запустити файл з розширенням .exe.

Далі, натиснути на кнопку, в залежності від потрібності користувача. (рис. Б.4).

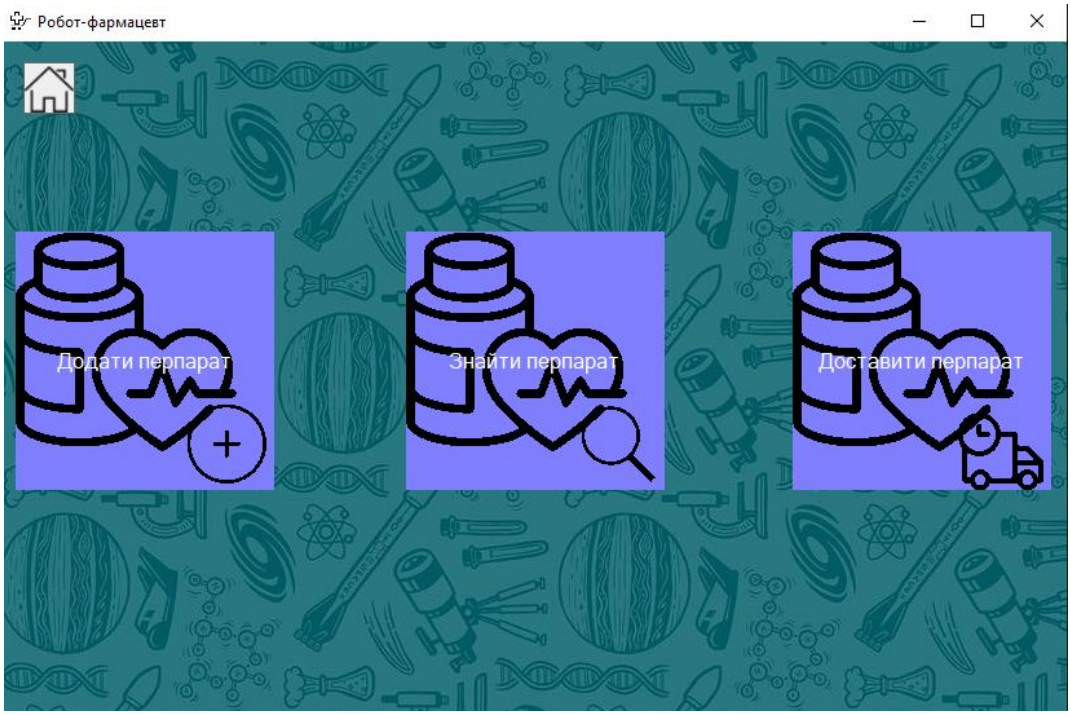



Рисунок Б.4 – Головна форма

Далі користувач натискає на одну з кнопок: “Додати препарат”, “Знайти препарат” , “Доставити препарат”.

Після натискання на кнопку “Додати препарат” користувач потрапляє до меню вводу даних препарату (рис. Б.5).

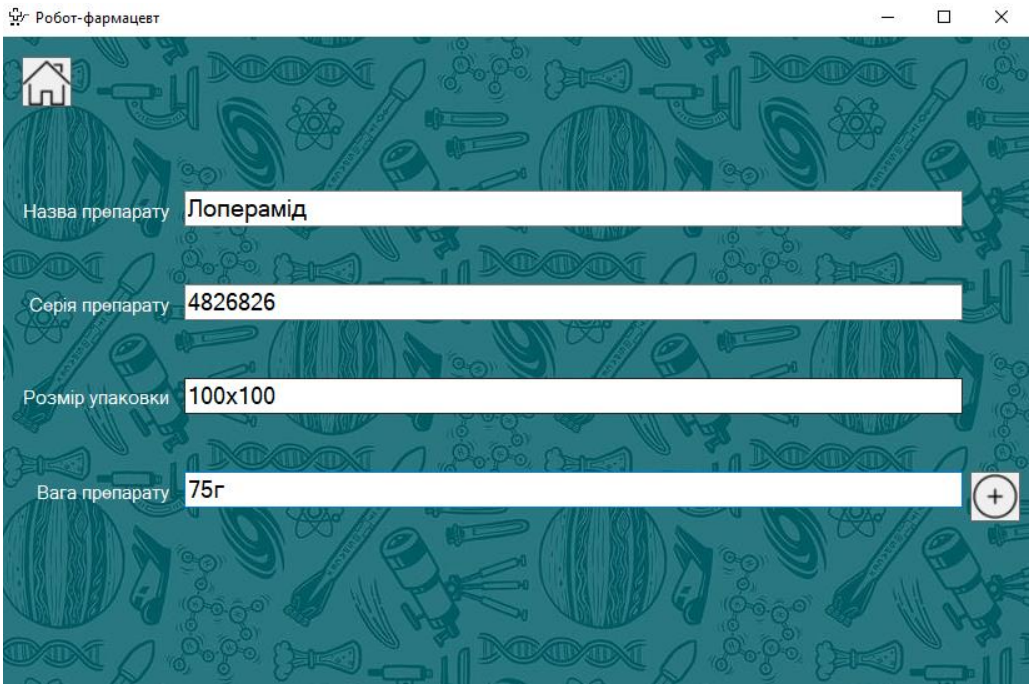



Рисунок Б.5 – Форма додавання нового препарату

Після вводу даних про препарат потрібно натиснути кнопку додавання препарату до бази даних. Якщо додавання пройшло успішно то буде виведене повідомлення про успішний ввід та повернемося на головну форму (рис. Б.6).

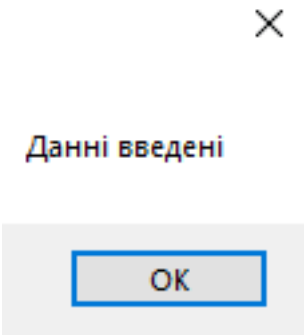



Рисунок Б.6 – Повідомлення про успішне введення даних

Якщо користувач натиснув на кнопку “Знайти препарат”, то відкриється меню вводу серії препарату (рис. Б.7).

Рисунок Б.7 – Форма пошуку

Коли користувач введе серійний номер, треба натиснути на кнопку пошуку, після чого з`явиться вся інформація про потрібний препарат (рис. Б.8).

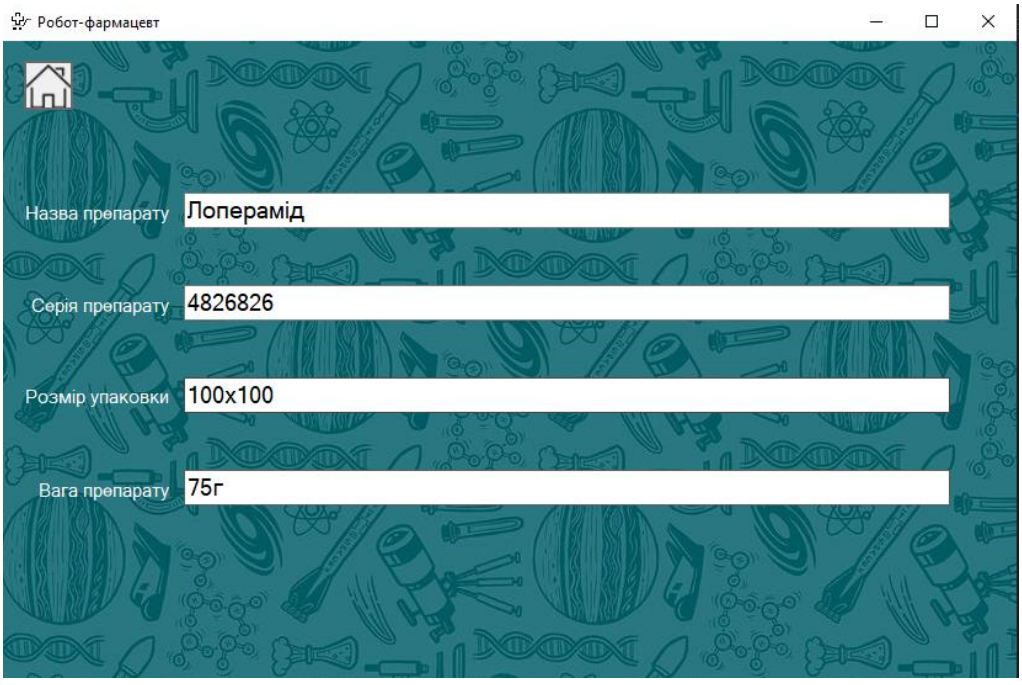



Рисунок Б.8 – Результат пошуку

Для виходу на головну сторінку користувач може використати кнопку додому.

Після натискання на кнопку “Доставити препарат” буде відкрито форму доставки препарату де треба ввести серійний номер потрібного препарату та за бажанням можна вибрати алгоритм доставки (за замовчуванням використовується алгоритм Дейкстри) чи збільшити кількість препаратів також можна вивести порівняння часу доставки всіх алгоритмів (рис. Б.9).

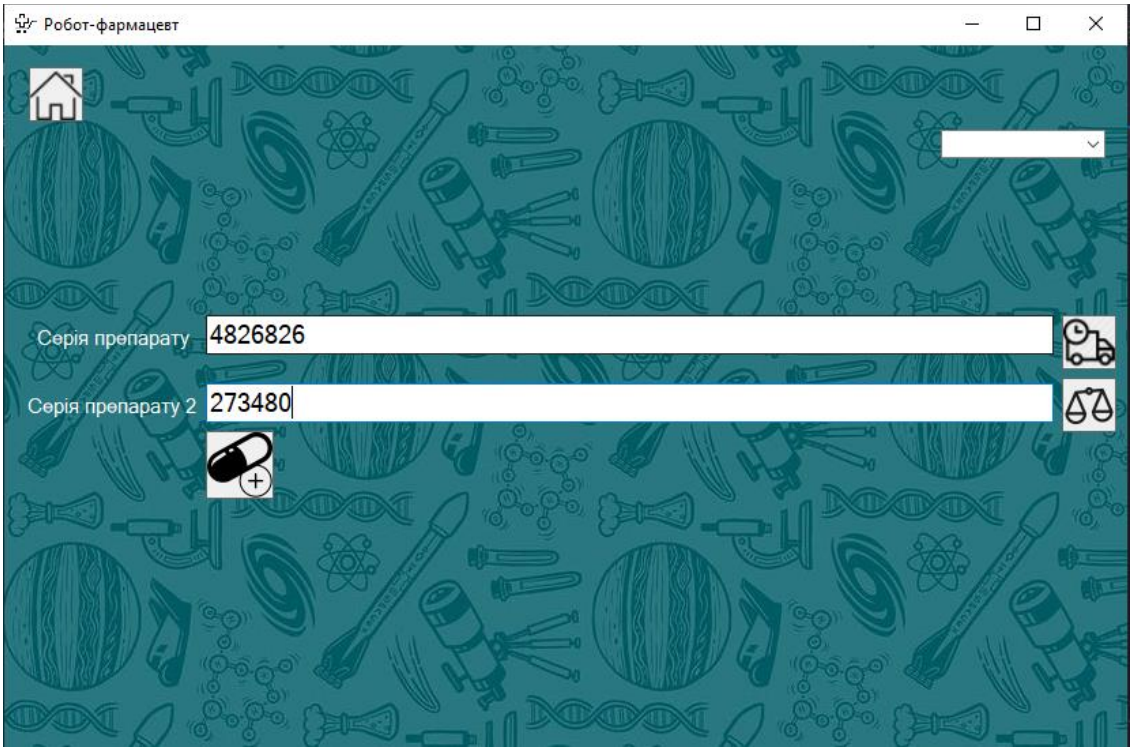



Рисунок Б.9 – Форма доставки

Після введення інформації треба натиснути на кнопку доставки препарату, як результат буде виведено повідомлення про час за який препарати були доставлені (рис. Б.10).

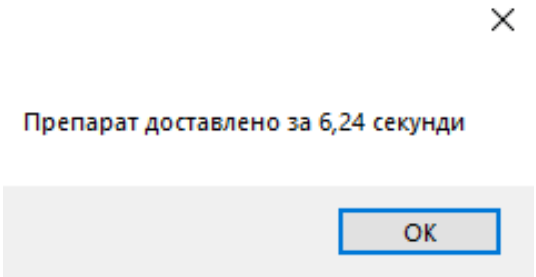



Рисунок Б.10 – Час доставки препарату

Якщо користувач після введення інформації натисне на кнопку порівняння часу доставки то буде виведено повідомлення про час доставки всіх алгоритмів (рис. Б.11).

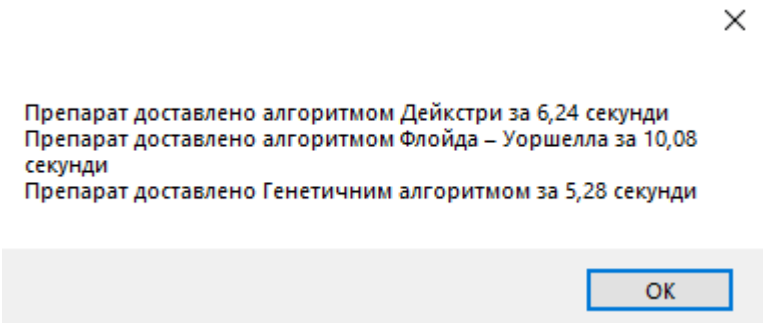



Рисунок Б.11 – Порівняння часу доставки

## Б.7 Вхідні та вихідні дані

Вхідними даними є введення користувачем інформації про назву, серію, розміри та масу препарату. Всі дані після заповнення оброблюються програмою та передаються до БД. На рис. Б.12 наведено приклад даних в БД.

Рисунок Б.12 – Інформація про існуючі препарати

Як видно з рис. Б.12 перший рядок namedrug у всіх препаратів це назва препаратів, другий seriadrug серія препаратів, третій sizedrug розміри препаратів, четвертий massdrug маса препаратів, та п’ятий nom це порядковий номер препарату в роботі цей номер не вводиться користувачем, а встановлюється програмою.

Вихідними даними ПП є повідомлення програми про введені дані в робот (рис. Б.13), повідомлення про помилку (рис. Б.14), повідомлення про час доставки препаратів(рис. Б.15) та порівняння часу доставки (рис. Б.16).

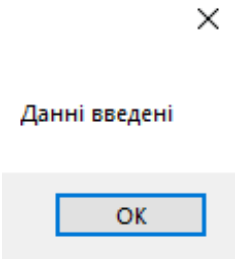



Рисунок Б.13 – Повідомлення про успішне введення даних

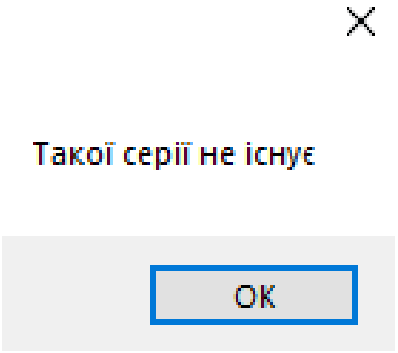



Рисунок Б.14 – Повідомлення про неправильно введену серію

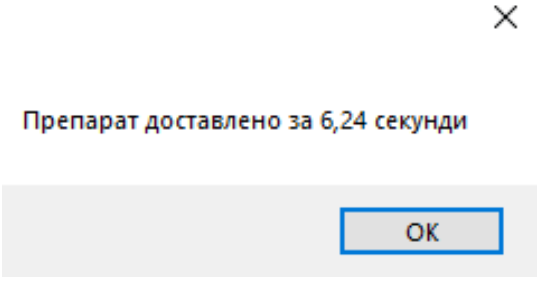



Рисунок Б.15 – Час доставки препарату

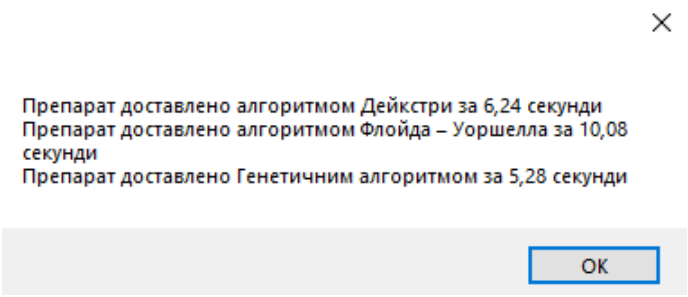



Рисунок Б.16 – Порівняння часу доставки

# ДОДАТОК В

## Лістинги файлів роботи

## В.1 Файл FormRobot.cs

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

namespace AptRobot
{
   public partial class FormRobot : Form
   {

      int number_of_push_dublicate = 1;
      string Version_algo = "Алгоритм Дейкстри";
      private const int _populationCount = 114;
      bool seria2 = true;
      bool seria3 = true;
      bool seria4 = true;
      bool seria5 = true;
      readonly int[,] drugs_matrix = new int[6000, 6000];

      readonly Drug_base Medizine = new Drug_base();
      readonly Metods metod = new Metods();
      readonly TextBox Dostavka_Vtorogo_Preparata = new TextBox();
      readonly TextBox Dostavka_Tretego_Preparata = new TextBox();
      readonly TextBox Dostavka_Chetvertogo_Preparata = new TextBox();
      readonly TextBox Dostavka_Patogo_Preparata = new TextBox();
      readonly Label Seria_2 = new Label();
      readonly Label Seria_3 = new Label();
      readonly Label Seria_4 = new Label();
      readonly Label Seria_5 = new Label();
      private Location _startLocation = new Location(0, 0);
      private TravellingSalesmanAlgorithm _algorithm;
      private Location[] _randomLocations;
      private Location[] _bestSolutionSoFar;

      public FormRobot()
      {
         InitializeComponent();
      }
```

```
private void FindDrug_Click(object sender, EventArgs e)
{
   NameDrug.Text = "";
   SeriaDrug.Text = "";
   SizeDrug.Text = "";
   MassDrug.Text = "";
   SeriaDrug.Visible = true;
   label2.Visible = true;
   FindDrug.Visible = false;
   EddDrug.Visible = false;
   DeliverDrug.Visible = false;
   Find.Visible = true;

}

private void EddDrug_Click(object sender, EventArgs e)
{
   NameDrug.Text = "";
   SeriaDrug.Text = "";
   SizeDrug.Text = "";
   MassDrug.Text = "";
   NameDrug.Visible = true;
   label1.Visible = true;
   SeriaDrug.Visible = true;
   label2.Visible = true;
   SizeDrug.Visible = true;
   label3.Visible = true;
   MassDrug.Visible = true;
   label4.Visible = true;

   FindDrug.Visible = false;
   EddDrug.Visible = false;
   DeliverDrug.Visible = false;
   Add.Visible = true;


}

private void DeliverDrug_Click(object sender, EventArgs e)
{
   NameDrug.Text = "";
   SeriaDrug.Text = "";
```

```
            SizeDrug.Text = "";
            MassDrug.Text = "";
            SeriaDrug.Visible = true;
            label2.Visible = true;
            FindDrug.Visible = false;
            EddDrug.Visible = false;
            DeliverDrug.Visible = false;
            Delivery.Visible = true;
            DublicateButton.Visible = true;
            Choose_algo.Visible = true;
            Make_compare_time.Visible = true;
        }

        private void Find_Click(object sender, EventArgs e)
        {
            SeriaDrug.Visible = false;
            label2.Visible = false;
            Find.Visible = false;
            _ = new string[4];
            string[] drug;
            Array.Copy(Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)), drug = new
string[Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)).Length],
Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)).Length);

            if (drug[0] == "")
            {
                SeriaDrug.Visible = true;
                SeriaDrug.Text = "";
                label2.Visible = true;
                Find.Visible = true;
            }
            else
            {

                NameDrug.Visible = true;
                SeriaDrug.Visible = true;
                SizeDrug.Visible = true;
                MassDrug.Visible = true;

                NameDrug.Text = drug[0];
                label1.Visible = true;
                SeriaDrug.Text = drug[1];
```

```
                label2.Visible = true;
                SizeDrug.Text = drug[2];
                label3.Visible = true;
                MassDrug.Text = drug[3];
                label4.Visible = true;
            }

        }

        private void Delivery_Click(object sender, EventArgs e)
        {
            _ = new string[4];
            string[] drug2 = new string[4];
            string[] drug3 = new string[4];
            string[] drug4 = new string[4];
            string[] drug5 = new string[4];
            string[] drug;
            Array.Copy(Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)), drug = new
string[Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)).Length],
Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)).Length);

            if (drug[0] == "")
            {
                MessageBox.Show("Серії 1 препарату не існує");
            }

            if (number_of_push_dublicate >= 2)
            {
                Array.Copy(Medizine.Makename(Medizine.FindDrug(Dostavka_Vtorogo_Preparata.Text)), drug2
= new string[Medizine.Makename(Medizine.FindDrug(Dostavka_Vtorogo_Preparata.Text)).Length],
Medizine.Makename(Medizine.FindDrug(Dostavka_Vtorogo_Preparata.Text)).Length);
                if (drug2[0] == "")
                {
                    MessageBox.Show("Серії 2 препарату не існує");
                }
            }
            if (number_of_push_dublicate >= 3)
            {
                Array.Copy(Medizine.Makename(Medizine.FindDrug(Dostavka_Tretego_Preparata.Text)), drug3
= new string[Medizine.Makename(Medizine.FindDrug(Dostavka_Tretego_Preparata.Text)).Length],
Medizine.Makename(Medizine.FindDrug(Dostavka_Tretego_Preparata.Text)).Length);
                if (drug3[0] == "")
```

```
                {
                    MessageBox.Show("Серії 3 препарату не існує");
                }
            }
            if (number_of_push_dublicate >= 4)
            {
                Array.Copy(Medizine.Makename(Medizine.FindDrug(Dostavka_Chetvertogo_Preparata.Text)),
drug4 = new string[Medizine.Makename(Medizine.FindDrug(Dostavka_Chetvertogo_Preparata.Text)).Length],
Medizine.Makename(Medizine.FindDrug(Dostavka_Chetvertogo_Preparata.Text)).Length);
                if (drug4[0] == "")
                {
                    MessageBox.Show("Серії 4 препарату не існує");
                }
            }
            if (number_of_push_dublicate >= 5)
            {
                Array.Copy(Medizine.Makename(Medizine.FindDrug(Dostavka_Patogo_Preparata.Text)), drug5 =
new                 string[Medizine.Makename(Medizine.FindDrug(Dostavka_Patogo_Preparata.Text)).Length],
Medizine.Makename(Medizine.FindDrug(Dostavka_Patogo_Preparata.Text)).Length);
                if (drug5[0] == "")
                {
                    MessageBox.Show("Серії 5 препарату не існує");
                }
            }

            if (drug[0] == "")
            {
                SeriaDrug.Visible = true;
                SeriaDrug.Text = "";
                Dostavka_Vtorogo_Preparata.Text = "";
                Dostavka_Tretego_Preparata.Text = "";
                Dostavka_Chetvertogo_Preparata.Text = ""; ;
                Dostavka_Patogo_Preparata.Text = "";
                label2.Visible = true;
                Delivery.Visible = true;
            }
            else
            {
                SeriaDrug.Visible = true;
                SeriaDrug.Text = "";
                Dostavka_Vtorogo_Preparata.Text = "";
                Dostavka_Tretego_Preparata.Text = "";
```

```
                Dostavka_Chetvertogo_Preparata.Text = ""; ;
                Dostavka_Patogo_Preparata.Text = "";
                label2.Visible = true;
                Delivery.Visible = true;

                Build_adjacency_matrix();
                double dist = 0;
                if (Version_algo == "Алгоритм Дейкстри")
                {
                   dist = metod.DijkstraAlgo(drugs_matrix, 0, 6000, Convert.ToInt32(drug[4]));

                   if (drug2[0] != null)
                   {
                      dist += metod.DijkstraAlgo(drugs_matrix, Convert.ToInt32(drug2[4]), 6000,
Convert.ToInt32(drug[4]));
                   }
                   if (drug3[0] != null)
                   {
                      dist += metod.DijkstraAlgo(drugs_matrix, Convert.ToInt32(drug3[4]), 6000,
Convert.ToInt32(drug2[4]));
                   }
                   if (drug4[0] != null)
                   {
                      dist += metod.DijkstraAlgo(drugs_matrix, Convert.ToInt32(drug4[4]), 6000,
Convert.ToInt32(drug3[4]));
                   }
                   if (drug5[0] != null)
                   {
                      dist += metod.DijkstraAlgo(drugs_matrix, Convert.ToInt32(drug5[4]), 6000,
Convert.ToInt32(drug4[4]));
                   }
                }

                if (Version_algo == "Алгоритм Флойда")
                {
                   dist = metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug[4]), 15);

                   if (drug2[0] != null)
                   {
                      dist += metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug2[4]),
Convert.ToInt32(drug[4]));
                   }
```

```
                if (drug3[0] != null)
                {
                    dist += metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug3[4]),
Convert.ToInt32(drug2[4]));
                }
                if (drug4[0] != null)
                {
                    dist += metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug4[4]),
Convert.ToInt32(drug3[4]));
                }
                if (drug5[0] != null)
                {
                    dist += metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug5[4]),
Convert.ToInt32(drug4[4]));
                }
            }

            if (Version_algo == "Генетичний алгоритм")
            {

                dist = Form_GenetycAlgo(drug,drug2,drug3,drug4,drug5);

            }

            MessageBox.Show("Препарат доставлено за " + metod.FindTimeDelivery(dist) + " секунди");
        }
    }

    private double Form_GenetycAlgo(string[] drug, string[] drug2, string[] drug3, string[] drug4, string[]
drug5) {


        double dist=0;
        int _destinationCount = Medizine.Num_of_nodes();
        int i = 0;
        while (i < 3000)
        {
            _randomLocations = RandomProvider.GetRandomDestinations(_destinationCount);
            _algorithm = new TravellingSalesmanAlgorithm(_startLocation, _randomLocations,
_populationCount)
            {
                MustMutateFailedCrossovers = false,
```

```
            MustDoCrossovers = true
        };
        _algorithm.Reproduce();
        _bestSolutionSoFar = _algorithm.GetBestSolutionSoFar().ToArray();
        i++;
    }

    Distance distace = new Distance();
    int num_on_robot = Convert.ToInt32(drug[4]);
    int num_of_findable_bestsolution = 0;

    for (int c = 0; c < _bestSolutionSoFar.Length; c++)
    {

        if (_bestSolutionSoFar[c].X == metod.Make_xy(num_on_robot)[0] && _bestSolutionSoFar[c].Y
== metod.Make_xy(num_on_robot)[1])//0-x 1-y
        {
            num_of_findable_bestsolution = c;
            break;
        }
    }

    if (drug2[0] != null && num_of_findable_bestsolution > 0)
    {
        num_on_robot = Convert.ToInt32(drug2[4]);
        int[] xy = new int[2];
        Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

        for (int c = num_of_findable_bestsolution; c >= 0; c--)
        {


            if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
            {

                seria2 = false;
                break;
            }

        }
    }
    if (drug3[0] != null && num_of_findable_bestsolution > 0)
```

```
{
   num_on_robot = Convert.ToInt32(drug3[4]);
   int[] xy = new int[2];
   Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

   for (int c = num_of_findable_bestsolution; c >= 0; c--)
   {


      if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
      {

         seria3 = false;
         break;
      }

   }
}
if (drug4[0] != null && num_of_findable_bestsolution > 0)
{
   num_on_robot = Convert.ToInt32(drug4[4]);
   int[] xy = new int[2];
   Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

   for (int c = num_of_findable_bestsolution; c >= 0; c--)
   {


      if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
      {

         seria4 = false;
         break;
      }

   }
}
if (drug5[0] != null && num_of_findable_bestsolution > 0)
{
   num_on_robot = Convert.ToInt32(drug5[4]);
   int[] xy = new int[2];
   Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);
```

```
                    for (int c = num_of_findable_bestsolution; c >= 0; c--)
                    {


                        if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
                        {

                            seria5 = false;
                            break;
                        }

                    }
                }

                dist += distace.Dist(_startLocation, _bestSolutionSoFar, num_of_findable_bestsolution);

                //-----------------------------------------------------------------------------------------------------------------------
----------
                if (drug2[0] != null && seria2)
                {

                    _destinationCount = Medizine.Num_of_nodes();
                    i = 0;
                    _startLocation = new Location(metod.Make_xy(num_on_robot)[0],
metod.Make_xy(num_on_robot)[1]);
                    while (i < 3000)
                    {
                        _randomLocations = RandomProvider.GetRandomDestinations(_destinationCount);
                        _algorithm = new TravellingSalesmanAlgorithm(_startLocation, _randomLocations,
_populationCount)
                        {
                            MustMutateFailedCrossovers = false,
                            MustDoCrossovers = true
                        };
                        _algorithm.Reproduce();
                        _bestSolutionSoFar = _algorithm.GetBestSolutionSoFar().ToArray();
                        i++;
                    }

                    num_on_robot = Convert.ToInt32(drug2[4]);
                    num_of_findable_bestsolution = 0;
```

```
for (int c = 0; c < _bestSolutionSoFar.Length; c++)
{

   if (_bestSolutionSoFar[c].X == metod.Make_xy(num_on_robot)[0] &&
_bestSolutionSoFar[c].Y == metod.Make_xy(num_on_robot)[1])//0-x 1-y
   {
      num_of_findable_bestsolution = c;
      break;
   }
}


if (drug3[0] != null && num_of_findable_bestsolution > 0)
{
   num_on_robot = Convert.ToInt32(drug3[4]);
   int[] xy = new int[2];
   Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

   for (int c = num_of_findable_bestsolution; c >= 0; c--)
   {


      if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
      {

         seria3 = false;
         break;
      }

   }
}


if (drug4[0] != null && num_of_findable_bestsolution > 0)
{
   num_on_robot = Convert.ToInt32(drug4[4]);
   int[] xy = new int[2];
   Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

   for (int c = num_of_findable_bestsolution; c >= 0; c--)
   {
```

```
                if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
                {

                    seria4 = false;
                    break;
                }

            }
        }


        if (drug5[0] != null && num_of_findable_bestsolution > 0)
        {
            num_on_robot = Convert.ToInt32(drug5[4]);
            int[] xy = new int[2];
            Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

            for (int c = num_of_findable_bestsolution; c >= 0; c--)
            {


                if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
                {

                    seria5 = false;
                    break;
                }

            }
        }

        dist += distace.Dist(_startLocation, _bestSolutionSoFar, num_of_findable_bestsolution);

    }
    //-------------------------------------------------------------------------------------------------------------------
---------
    if (drug3[0] != null && seria3)
    {


        _destinationCount = Medizine.Num_of_nodes();
```

```
                i = 0;
                _startLocation = new Location(metod.Make_xy(num_on_robot)[0],
metod.Make_xy(num_on_robot)[1]);
                while (i < 3000)
                {
                  _randomLocations = RandomProvider.GetRandomDestinations(_destinationCount);
                  _algorithm = new TravellingSalesmanAlgorithm(_startLocation, _randomLocations,
_populationCount)
                  {
                    MustMutateFailedCrossovers = false,
                    MustDoCrossovers = true
                  };
                  _algorithm.Reproduce();
                  _bestSolutionSoFar = _algorithm.GetBestSolutionSoFar().ToArray();
                  i++;
                }

                num_on_robot = Convert.ToInt32(drug3[4]);
                num_of_findable_bestsolution = 0;

                for (int c = 0; c < _bestSolutionSoFar.Length; c++)
                {

                  if (_bestSolutionSoFar[c].X == metod.Make_xy(num_on_robot)[0] &&
_bestSolutionSoFar[c].Y == metod.Make_xy(num_on_robot)[1])//0-x 1-y
                  {
                    num_of_findable_bestsolution = c;
                    break;
                  }
                }

                if (drug4[0] != null && num_of_findable_bestsolution > 0)
                {
                  num_on_robot = Convert.ToInt32(drug4[4]);
                  int[] xy = new int[2];
                  Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

                  for (int c = num_of_findable_bestsolution; c >= 0; c--)
                  {


                    if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
```

```
                    {

                        seria4 = false;
                        break;
                    }

                }
            }
            if (drug5[0] != null && num_of_findable_bestsolution > 0)
            {
                num_on_robot = Convert.ToInt32(drug5[4]);
                int[] xy = new int[2];
                Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

                for (int c = num_of_findable_bestsolution; c >= 0; c--)
                {


                    if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
                    {

                        seria5 = false;
                        break;
                    }

                }
            }

            dist += distace.Dist(_startLocation, _bestSolutionSoFar, num_of_findable_bestsolution);

        }
        //-------------------------------------------------------------------------------------------------------------
        if (drug4[0] != null && seria4)
        {

            _destinationCount = Medizine.Num_of_nodes();
            i = 0;
            _startLocation = new Location(metod.Make_xy(num_on_robot)[0], metod.Make_xy(num_on_robot)[1]);
            while (i < 3000)
            {
```

```
                    _randomLocations = RandomProvider.GetRandomDestinations(_destinationCount);
                    _algorithm = new TravellingSalesmanAlgorithm(_startLocation, _randomLocations,
_populationCount)
                    {
                        MustMutateFailedCrossovers = false,
                        MustDoCrossovers = true
                    };
                    _algorithm.Reproduce();
                    _bestSolutionSoFar = _algorithm.GetBestSolutionSoFar().ToArray();
                    i++;
                }

                num_on_robot = Convert.ToInt32(drug4[4]);
                num_of_findable_bestsolution = 0;

                for (int c = 0; c < _bestSolutionSoFar.Length; c++)
                {

                    if (_bestSolutionSoFar[c].X == metod.Make_xy(num_on_robot)[0] &&
_bestSolutionSoFar[c].Y == metod.Make_xy(num_on_robot)[1])//0-x 1-y
                    {
                        num_of_findable_bestsolution = c;
                        break;
                    }
                }



                if (drug5[0] != null && num_of_findable_bestsolution > 0)
                {
                    num_on_robot = Convert.ToInt32(drug5[4]);
                    int[] xy = new int[2];
                    Array.Copy(metod.Make_xy(num_on_robot), 0, xy, 0, 2);

                    for (int c = num_of_findable_bestsolution; c >= 0; c--)
                    {


                        if (_bestSolutionSoFar[c].X == xy[0] && _bestSolutionSoFar[c].Y == xy[1])
                        {

                            seria5 = false;
                            break;
```

```
                    }

                }
            }

            dist += distace.Dist(_startLocation, _bestSolutionSoFar, num_of_findable_bestsolution);

        }
        //-------------------------------------------------------------------------------------------------------------
        if (drug5[0] != null && seria5)
        {

            _destinationCount = Medizine.Num_of_nodes();
            i = 0;
            _startLocation = new Location(metod.Make_xy(num_on_robot)[0], metod.Make_xy(num_on_robot)[1]);
            while (i < 3000)
            {
                _randomLocations = RandomProvider.GetRandomDestinations(_destinationCount);
                _algorithm = new TravellingSalesmanAlgorithm(_startLocation, _randomLocations, _populationCount)
                {
                    MustMutateFailedCrossovers = false,
                    MustDoCrossovers = true
                };
                _algorithm.Reproduce();
                var newSolution = _algorithm.GetBestSolutionSoFar().ToArray();
                _bestSolutionSoFar = newSolution;
                i++;
            }

            num_on_robot = Convert.ToInt32(drug5[4]);
            num_of_findable_bestsolution = 0;

            for (int c = 0; c < _bestSolutionSoFar.Length; c++)
            {

                if (_bestSolutionSoFar[c].X == metod.Make_xy(num_on_robot)[0] && _bestSolutionSoFar[c].Y == metod.Make_xy(num_on_robot)[1])//0-x 1-y
                {
                    num_of_findable_bestsolution = c;
```

```
                break;
            }
        }

        dist += distace.Dist(_startLocation, _bestSolutionSoFar, num_of_findable_bestsolution);

    }

    return dist;
}

private void Add_Click(object sender, EventArgs e)
{
    NameDrug.Visible = false;
    label1.Visible = false;
    SeriaDrug.Visible = false;
    label2.Visible = false;
    SizeDrug.Visible = false;
    label3.Visible = false;
    MassDrug.Visible = false;
    label4.Visible = false;

    FindDrug.Visible = true;
    EddDrug.Visible = true;
    DeliverDrug.Visible = true;
    Add.Visible = false;


    Medizine.Synchronyzate(NameDrug.Text, SeriaDrug.Text, MassDrug.Text, SizeDrug.Text);


}

private void MainWindow_Click(object sender, EventArgs e)
{
    FindDrug.Visible = true;
    EddDrug.Visible = true;
    DeliverDrug.Visible = true;
    Find.Visible = false;
    Delivery.Visible = false;
    Add.Visible = false;
```

```
    NameDrug.Visible = false;
    NameDrug.Text = "";
    label1.Visible = false;

    SeriaDrug.Visible = false;
    SeriaDrug.Text = "";
    label2.Visible = false;

    SizeDrug.Visible = false;
    SizeDrug.Text = "";
    label3.Visible = false;

    MassDrug.Visible = false;
    MassDrug.Text = "";
    label4.Visible = false;

    DublicateButton.Visible = false;
    Dostavka_Vtorogo_Preparata.Visible = false;
    Dostavka_Vtorogo_Preparata.Text = "";
    Dostavka_Tretego_Preparata.Visible = false;
    Dostavka_Tretego_Preparata.Text = "";
    Dostavka_Chetvertogo_Preparata.Visible = false;
    Dostavka_Chetvertogo_Preparata.Text = "";
    Dostavka_Patogo_Preparata.Visible = false;
    Dostavka_Patogo_Preparata.Text = "";
    Seria_2.Visible = false;
    Seria_3.Visible = false;
    Seria_4.Visible = false;
    Seria_5.Visible = false;
    Choose_algo.Visible = false;
    DublicateButton.Location = new Point(148, 233);
    number_of_push_dublicate = 1;
    Make_compare_time.Visible = false;
}

private void DublicateButton_Click(object sender, EventArgs e)
{
    switch (number_of_push_dublicate)
    {
        case 1:
            Seria_2.Location = new Point(label2.Location.X-7, label2.Location.Y + 50);
            Seria_2.Text = "Серія препарату 2";
```

```
                Seria_2.Font = new Font(label2.Font.ToString(), label2.Font.Size);
                Seria_2.Size = label2.Size;
                Seria_2.Visible = true;
                Seria_2.BackColor = Color.Transparent;
                Seria_2.ForeColor = Color.White;
                Seria_2.Width += 14;
                Controls.Add(Seria_2);

                Dostavka_Vtorogo_Preparata.Location = new Point(SeriaDrug.Location.X,
SeriaDrug.Location.Y + 50);
                Dostavka_Vtorogo_Preparata.Name = "Dostavka_Vtorogo_Preparata";
                Dostavka_Vtorogo_Preparata.Size = SeriaDrug.Size;
                Dostavka_Vtorogo_Preparata.Font = new Font(SeriaDrug.Font.ToString(),
SeriaDrug.Font.Size);
                Dostavka_Vtorogo_Preparata.Visible = true;
                Dostavka_Vtorogo_Preparata.TabIndex = 3;
                Controls.Add(Dostavka_Vtorogo_Preparata);

                DublicateButton.Location = new Point(DublicateButton.Location.X,
DublicateButton.Location.Y + 50);
                number_of_push_dublicate++;
                break;

            case 2:
                Seria_3.Location = new Point(label2.Location.X-7, label2.Location.Y + 100);
                Seria_3.Text = "Серія препарату 3";
                Seria_3.Font = new Font(label2.Font.ToString(), label2.Font.Size);
                Seria_3.Size = label2.Size;
                Seria_3.Visible = true;
                Seria_3.BackColor = Color.Transparent;
                Seria_3.ForeColor = Color.White;
                Seria_3.Width += 14;
                Controls.Add(Seria_3);

                Dostavka_Tretego_Preparata.Location = new Point(SeriaDrug.Location.X,
SeriaDrug.Location.Y + 100);
                Dostavka_Tretego_Preparata.Name = "Dostavka_Tretego_Preparata";
                Dostavka_Tretego_Preparata.Size = SeriaDrug.Size;
                Dostavka_Tretego_Preparata.Font = new Font(SeriaDrug.Font.ToString(), SeriaDrug.Font.Size);
                Dostavka_Tretego_Preparata.Visible = true;
                Dostavka_Tretego_Preparata.TabIndex = 3;
                Controls.Add(Dostavka_Tretego_Preparata);
```

```
                DublicateButton.Location = new Point(DublicateButton.Location.X,
DublicateButton.Location.Y + 50);
                number_of_push_dublicate++;
                break;

            case 3:
                Seria_4.Location = new Point(label2.Location.X-7, label2.Location.Y + 150);
                Seria_4.Text = "Серія препарату 4";
                Seria_4.Font = new Font(label2.Font.ToString(), label2.Font.Size);
                Seria_4.Size = label2.Size;
                Seria_4.Visible = true;
                Seria_4.BackColor = Color.Transparent;
                Seria_4.ForeColor = Color.White;
                Seria_4.Width += 14;
                Controls.Add(Seria_4);

                Dostavka_Chetvertogo_Preparata.Location = new Point(SeriaDrug.Location.X,
SeriaDrug.Location.Y + 150);
                Dostavka_Chetvertogo_Preparata.Name = "Dostavka_Chetvertogo_Preparata";
                Dostavka_Chetvertogo_Preparata.Size = SeriaDrug.Size;
                Dostavka_Chetvertogo_Preparata.Font = new Font(SeriaDrug.Font.ToString(),
SeriaDrug.Font.Size);
                Dostavka_Chetvertogo_Preparata.Visible = true;
                Dostavka_Chetvertogo_Preparata.TabIndex = 3;
                Controls.Add(Dostavka_Chetvertogo_Preparata);

                DublicateButton.Location = new Point(DublicateButton.Location.X,
DublicateButton.Location.Y + 50);
                number_of_push_dublicate++;
                break;

            case 4:
                Seria_5.Location = new Point(label2.Location.X-7, label2.Location.Y + 200);
                Seria_5.Text = "Серія препарату 5";
                Seria_5.Font = new Font(label2.Font.ToString(), label2.Font.Size);
                Seria_5.Size = label2.Size;
                Seria_5.Visible = true;
                Seria_5.BackColor = Color.Transparent;
                Seria_5.ForeColor = Color.White;
                Seria_5.Width += 14;
                Controls.Add(Seria_5);
```

```
                    Dostavka_Patogo_Preparata.Location = new Point(SeriaDrug.Location.X, SeriaDrug.Location.Y
+ 200);
                    Dostavka_Patogo_Preparata.Name = "Dostavka_Patogo_Preparata";
                    Dostavka_Patogo_Preparata.Size = SeriaDrug.Size;
                    Dostavka_Patogo_Preparata.Font = new Font(SeriaDrug.Font.ToString(), SeriaDrug.Font.Size);
                    Dostavka_Patogo_Preparata.Visible = true;
                    Dostavka_Patogo_Preparata.TabIndex = 3;
                    Controls.Add(Dostavka_Patogo_Preparata);
                    DublicateButton.Location = new Point(DublicateButton.Location.X,
DublicateButton.Location.Y + 50);

                    number_of_push_dublicate++;
                    DublicateButton.Visible = false;
                    break;
                }

            }

            public void Build_adjacency_matrix()
            {

                Drug_base dr = new Drug_base();

                int last_num = dr.FindLastNum();

                for (int k = last_num; k >= 0; k--)
                {


                    if (k < 200 && k != 0 && k != 200)
                    {//niz

                        drugs_matrix[k, k + 1] = 8;
                        drugs_matrix[k + 1, k] = 8;

                        drugs_matrix[k, k - 1] = 8;
                        drugs_matrix[k - 1, k] = 8;



                        drugs_matrix[k, k + 200] = 7;
                        drugs_matrix[k + 200, k] = 7;
```

```
                }
                else if (k > 5801 && k != 5801 && k != 6000)
                {//verx

                    drugs_matrix[k, k + 1] = 8;
                    drugs_matrix[k + 1, k] = 8;

                    drugs_matrix[k, k - 1] = 8;
                    drugs_matrix[k - 1, k] = 8;

                    drugs_matrix[k, k - 200] = 7;
                    drugs_matrix[k - 200, k] = 7;
                }
                else if (k % 200 == 0 && k != 5801 && k != 0)
                {//pravo
                    drugs_matrix[k, k - 1] = 8;
                    drugs_matrix[k - 1, k] = 8;



                    drugs_matrix[k, k + 200] = 7;
                    drugs_matrix[k + 200, k] = 7;



                    drugs_matrix[k, k - 200] = 7;
                    drugs_matrix[k - 200, k] = 7;
                }
                else if (k == 201 || k == 401 || k == 401 || k == 801 || k == 1001 || k == 1201 || k == 1401 || k == 1601 || k == 1801 || k == 2001 || k == 2201 || k == 2401 || k == 2601 || k == 2801 || k == 3001 || k == 3201 || k == 3401 || k == 3601 || k == 3801 || k == 4001 || k == 4201 || k == 4401 || k == 4601 || k == 4801 || k == 5001 || k == 5201 || k == 5401 || k == 5801)
                {//levo
                    drugs_matrix[k, k - 1] = 8;
                    drugs_matrix[k - 1, k] = 8;



                    drugs_matrix[k, k + 200] = 7;
                    drugs_matrix[k + 200, k] = 7;



                    drugs_matrix[k, k - 200] = 7;
                    drugs_matrix[k - 200, k] = 7;
                }
```

```
else if (k == 0)
{

   drugs_matrix[k, k + 1] = 8;
   drugs_matrix[k + 1, k] = 8;


   drugs_matrix[k, k + 200] = 7;
   drugs_matrix[k + 200, k] = 7;
}
else if (k == 5801)
{

   drugs_matrix[k, k + 1] = 8;
   drugs_matrix[k + 1, k] = 8;


   drugs_matrix[k, k - 200] = 7;
   drugs_matrix[k - 200, k] = 7;
}
else if (k == 6000)
{

   drugs_matrix[k, k - 1] = 8;
   drugs_matrix[k - 1, k] = 8;


   drugs_matrix[k, k - 200] = 7;
   drugs_matrix[k - 200, k] = 7;
}
else if (k == 201)
{

   drugs_matrix[k, k - 1] = 8;
   drugs_matrix[k - 1, k] = 8;


   drugs_matrix[k, k + 200] = 7;
   drugs_matrix[k + 200, k] = 7;
}
else
{
```

```
                drugs_matrix[k, k + 1] = 8;
                drugs_matrix[k + 1, k] = 8;

                drugs_matrix[k, k - 1] = 8;
                drugs_matrix[k - 1, k] = 8;


                drugs_matrix[k, k + 200] = 7;
                drugs_matrix[k + 200, k] = 7;


                drugs_matrix[k, k - 200] = 7;
                drugs_matrix[k - 200, k] = 7;


            }
        }
    }

    private void Choose_algo_SelectedIndexChanged(object sender, EventArgs e)
    {
        Version_algo = Choose_algo.SelectedItem.ToString();
    }

    private void Make_compare_time_Click(object sender, EventArgs e)
    {
        _ = new string[4];
        string[] drug2 = new string[4];
        string[] drug3 = new string[4];
        string[] drug4 = new string[4];
        string[] drug5 = new string[4];
        string[] drug;
        Array.Copy(Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)), drug = new
string[Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)).Length],
Medizine.Makename(Medizine.FindDrug(SeriaDrug.Text)).Length);

        if (drug[0] == "")
        {
            MessageBox.Show("Серії 1 препарату не існує");
        }

        if (number_of_push_dublicate >= 2)
```

```
                {
                    Array.Copy(Medizine.Makename(Medizine.FindDrug(Dostavka_Vtorogo_Preparata.Text)), drug2 = new string[Medizine.Makename(Medizine.FindDrug(Dostavka_Vtorogo_Preparata.Text)).Length], Medizine.Makename(Medizine.FindDrug(Dostavka_Vtorogo_Preparata.Text)).Length);
                    if (drug2[0] == "")
                    {
                        MessageBox.Show("Серії 2 препарату не існує");
                    }
                }
                if (number_of_push_dublicate >= 3)
                {
                    Array.Copy(Medizine.Makename(Medizine.FindDrug(Dostavka_Tretego_Preparata.Text)), drug3 = new string[Medizine.Makename(Medizine.FindDrug(Dostavka_Tretego_Preparata.Text)).Length], Medizine.Makename(Medizine.FindDrug(Dostavka_Tretego_Preparata.Text)).Length);
                    if (drug3[0] == "")
                    {
                        MessageBox.Show("Серії 3 препарату не існує");
                    }
                }
                if (number_of_push_dublicate >= 4)
                {
                    Array.Copy(Medizine.Makename(Medizine.FindDrug(Dostavka_Chetvertogo_Preparata.Text)), drug4 = new string[Medizine.Makename(Medizine.FindDrug(Dostavka_Chetvertogo_Preparata.Text)).Length], Medizine.Makename(Medizine.FindDrug(Dostavka_Chetvertogo_Preparata.Text)).Length);
                    if (drug4[0] == "")
                    {
                        MessageBox.Show("Серії 4 препарату не існує");
                    }
                }
                if (number_of_push_dublicate >= 5)
                {
                    Array.Copy(Medizine.Makename(Medizine.FindDrug(Dostavka_Patogo_Preparata.Text)), drug5 = new string[Medizine.Makename(Medizine.FindDrug(Dostavka_Patogo_Preparata.Text)).Length], Medizine.Makename(Medizine.FindDrug(Dostavka_Patogo_Preparata.Text)).Length);
                    if (drug5[0] == "")
                    {
                        MessageBox.Show("Серії 5 препарату не існує");
                    }
                }

                if (drug[0] == "")
                {
```

```
            SeriaDrug.Visible = true;
            SeriaDrug.Text = "";
            Dostavka_Vtorogo_Preparata.Text = "";
            Dostavka_Tretego_Preparata.Text = "";
            Dostavka_Chetvertogo_Preparata.Text = ""; ;
            Dostavka_Patogo_Preparata.Text = "";
            label2.Visible = true;
            Delivery.Visible = true;
        }
        else
        {
            SeriaDrug.Visible = true;
            SeriaDrug.Text = "";
            Dostavka_Vtorogo_Preparata.Text = "";
            Dostavka_Tretego_Preparata.Text = "";
            Dostavka_Chetvertogo_Preparata.Text = ""; ;
            Dostavka_Patogo_Preparata.Text = "";
            label2.Visible = true;
            Delivery.Visible = true;

            Build_adjacency_matrix();

            double Deikstra_dist = metod.DijkstraAlgo(drugs_matrix, 0, 6000, Convert.ToInt32(drug[4]));
            if (drug2[0] != null)
            {
                Deikstra_dist += metod.DijkstraAlgo(drugs_matrix, Convert.ToInt32(drug2[4]), 6000,
Convert.ToInt32(drug[4]));
            }
            if (drug3[0] != null)
            {
                Deikstra_dist += metod.DijkstraAlgo(drugs_matrix, Convert.ToInt32(drug3[4]), 6000,
Convert.ToInt32(drug2[4]));

            }
            if (drug4[0] != null)
            {
                Deikstra_dist += metod.DijkstraAlgo(drugs_matrix, Convert.ToInt32(drug4[4]), 6000,
Convert.ToInt32(drug3[4]));
            }
            if (drug5[0] != null)
            {
```

```
                    Deikstra_dist += metod.DijkstraAlgo(drugs_matrix, Convert.ToInt32(drug5[4]), 6000, Convert.ToInt32(drug4[4]));
                }

                double Floid_dist = metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug[4]), 15);

                if (drug2[0] != null)
                {
                    Floid_dist += metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug2[4]), Convert.ToInt32(drug[4]));
                }
                if (drug3[0] != null)
                {
                    Floid_dist += metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug3[4]), Convert.ToInt32(drug2[4]));
                }
                if (drug4[0] != null)
                {
                    Floid_dist += metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug4[4]), Convert.ToInt32(drug3[4]));
                }
                if (drug5[0] != null)
                {
                    Floid_dist += metod.FloidAlgo(drugs_matrix, Convert.ToInt32(drug5[4]), Convert.ToInt32(drug4[4]));
                }
                double Genetic_dist = Form_GenetycAlgo(drug, drug2, drug3, drug4, drug5);

                MessageBox.Show("Препарат доставлено алгоритмом Дейкстри за " + metod.FindTimeDelivery(Deikstra_dist) + " секунди"+ "\n" + "Препарат доставлено алгоритмом Флойда – Уоршелла за " + metod.FindTimeDelivery(Floid_dist) + " секунди" + "\n"+ "Препарат доставлено Генетичним алгоритмом за " + metod.FindTimeDelivery(Genetic_dist) + " секунди");
                }
            }
        }

        //0-------------------------------------------------------------------------------------------------------------------------------------------------------------------

    }
```

# ДОДАТОК Д

## Слайди презентації

Рисунок Д.1 – Слайд № 1

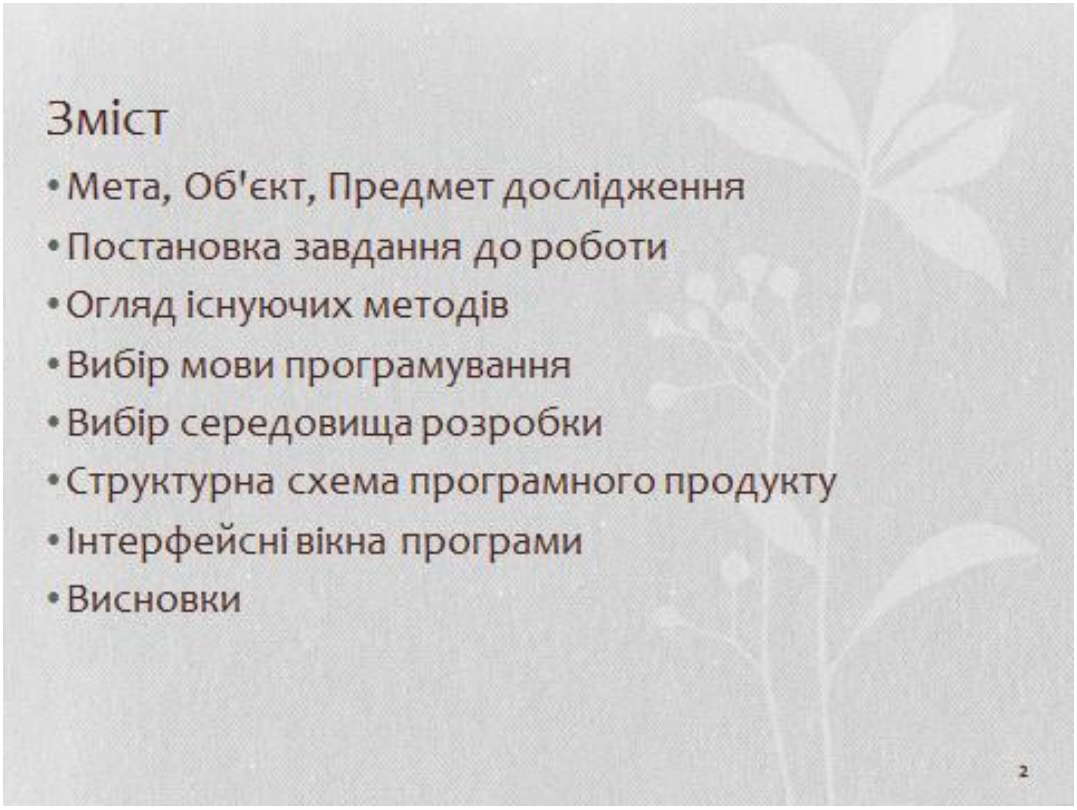



Рисунок Д.2 – Слайд № 2

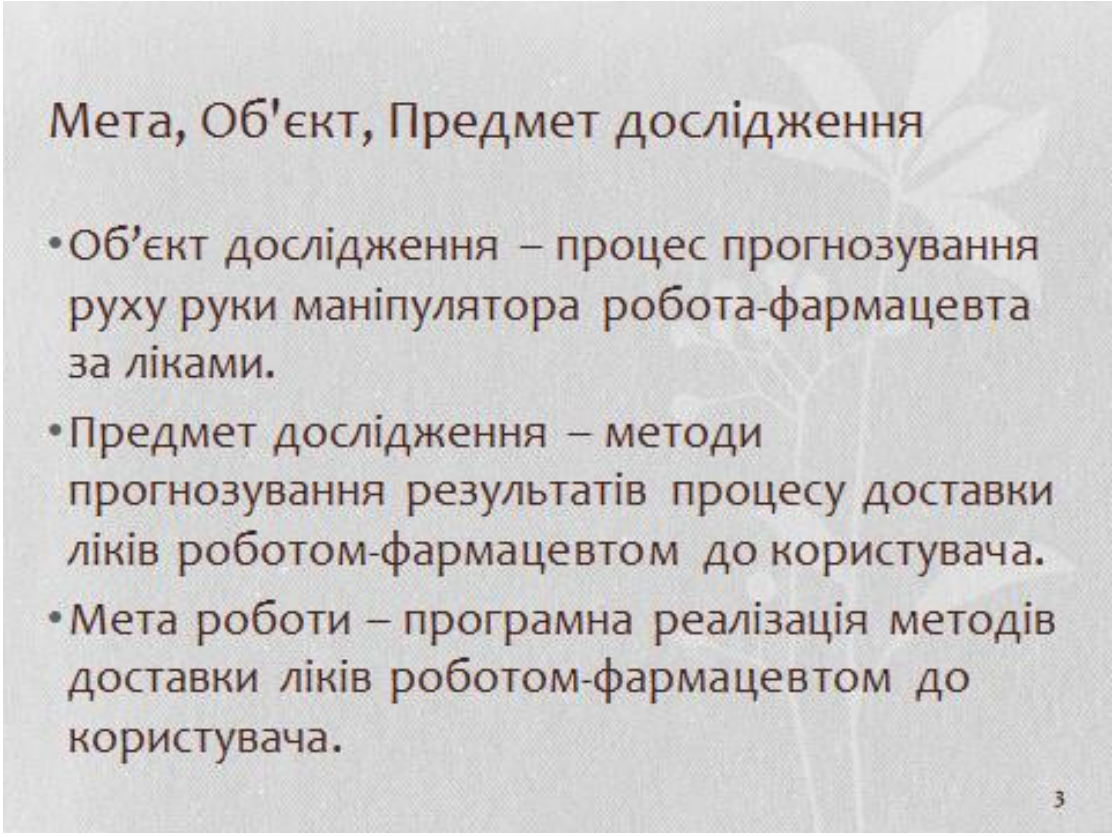



Рисунок Д.3 – Слайд № 3

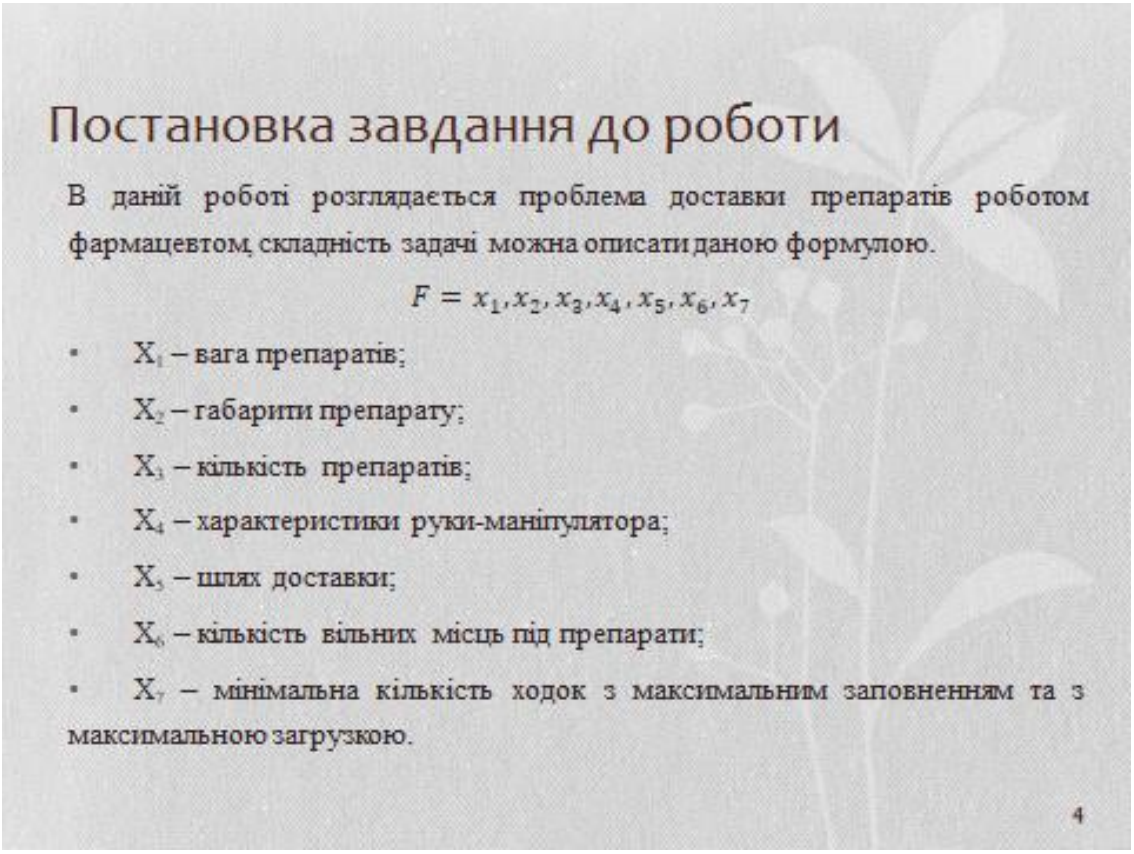



Рисунок Д.4 – Слайд № 4

## Огляд існуючих методів

| | Дейкстри | Комівояжера | Дерева рішень | Одношарова штучна нейронна мережа | Мурашиних колоній | Генетичний алгоритм(простий) |
|---|---|---|---|---|---|---|
| Простота та прозорість реалізації | Так | Так | Так | Ні | Ні | Ні |
| Висока часова залежність від розмірності задачі | Так | Так | Так | Так | Так | Так |
| Простота кодування вхідної і вихідної інформації | Так | Так | Ні | Ні | Ні | Так |
| Можливість розпаралелювання | Так | Так | Так | Ні | Ні | Так |
| Висока залежність результатів роботи методу від початкових параметрів пошуку | Так | Так | Так | Так | Ні | Так |
| Висока адаптивність до особливостей вирішувальної задачі | Так | Так | Так | Так | Ні | Ні |
| Складність алгоритму | 2 | 2 | 2 | 1 | 3 | 3 |

5

Рисунок Д.5 – Слайд №5

## Вибір мови програмування

| | C# | C++ | Python | Java |
|---|---|---|---|---|
| Кросплатформеність | Так | Так | Так | Так |
| Можливість обробки великої вибірки даних | Так | Так | Так | Так |
| Наявність зрозумілої та повної документації | Так | Так | Так | Так |
| Наявність якісної візуалізації даних | Так | Ні | Так | Ні |
| Простота вивчення та розробки | Так | Так | Так | Так |
| Наявність якісних предметно-орієнтованих пакетів з відкритим вихідним кодом | Так | Так | Так | Так |
| Наявність великої кількості наборів статистичних функцій та методів | Так | Так | Ні | Так |
| Паралельне програмування | Так | Так | Ні | Так |
| Взаємодія з іншими мовами | Так | Так | Так | Так |

6

Рисунок Д.6 – Слайд № 6

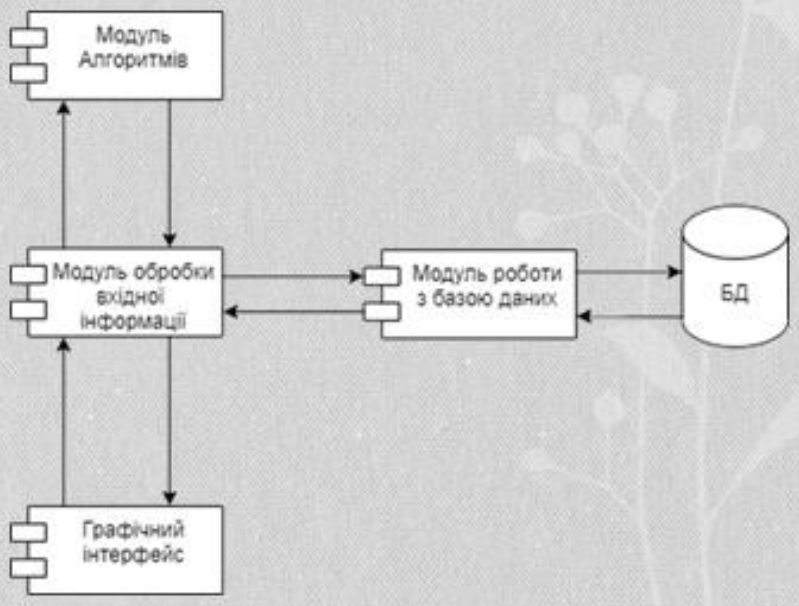
## Вибір середовища розробки

| | Visual Studio | Rider | Eclipse |
|---|---|---|---|
| Ліцензія | Freemium | Community edition: Apache License 2.0 | Eclipse Public License |
| Кросплатформеність | Так | Так | Так |
| Наявність автодоповнення | Так | Так | Так |
| Підтримка репозиторіїв | Так | Так | Так |
| Наявність «plugin manager» | Так | Так | Так |
| Простота встановлення | Так | Так | Ні |
| Підтримка аналізу коду | Так | Так | Так |
| Підтримка рефакторингу коду | Так | Так | Так |
| Наявність інтерактивної консолі | Так | Так | Ні |
| Зручний користувацький інтерфейс | Так Так | Так | Ні |
| Швидкість модернізації | Так Так | Так | Ні |

7

Рисунок Д.7 – Слайд № 7

## Структурна схема програмного продукту




8

Рисунок Д.8 – Слайд № 8

Рисунок Д.9 – Слайд № 9

Рисунок Д.10 – Слайд № 10

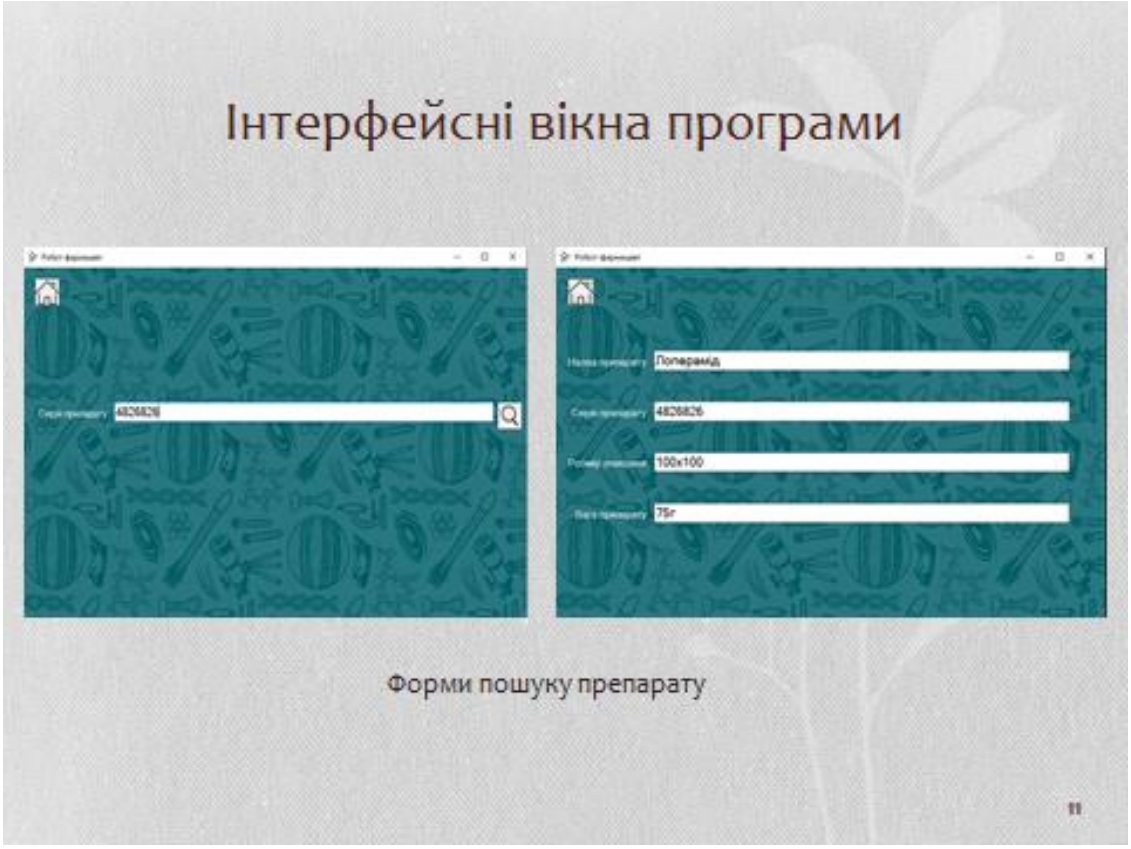



Рисунок Д.11 – Слайд № 11

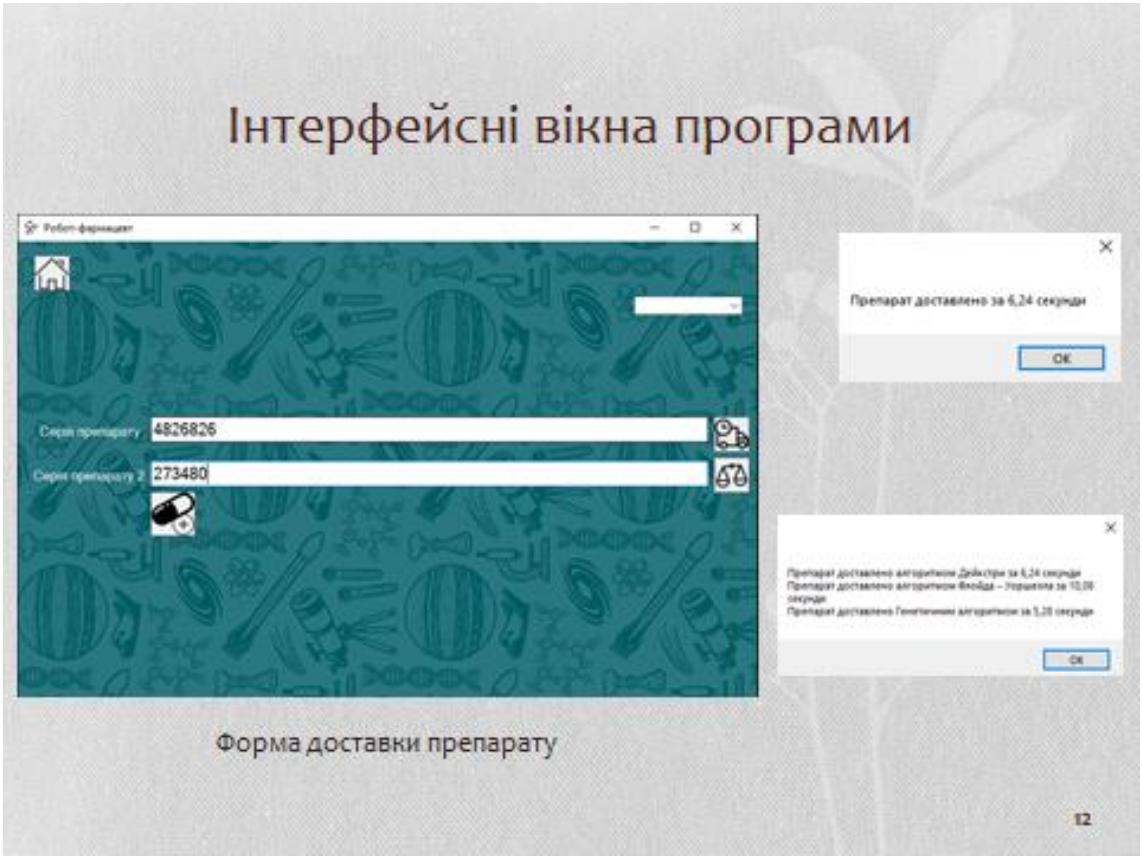



Рисунок Д.12 – Слайд № 12

Рисунок Д.13 – Слайд № 13