

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ
"АВТОМАТИЗОВАНИЙ СКЛАД ЛІКІВ"
DEVELOPMENT OF THE AUTOMATED DRUG
INVENTORY SOFTWARE APPLICATION

\

Виконав(ла): студент(ка) 4 курсу, групи КНТ-111 сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПЕТУНІН О.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент Коцур М.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
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.
Сergій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПЕТУНІНА Олександра
Володимировича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного застосунку «Автоматизований склад ліків»

керівник проєкту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олександрович
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування системи, що проєктується. 3. Основні рішення щодо реалізації системи, яка проєктується. 4. Керівництво програміста. 5. Керівництво оператора. 6. Тестування і результати. 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтроль	ЛИПОВЕЦЬ М. В., асистент		

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз предметної області.	1 тиждень	
2	Розробка та затвердження технічного завдання.	1 тиждень	Технічне завдання
3	Проектування структури системи.	2 тиждень	Структурна схема
4	Розробка схеми функціонування системи та модулів програми.	2 тиждень	Функціональна схема. Модулі
5	Створення програмного коду системи.	3-4- тижні	Текст програми
6	Тестування та відлагодження програми.	3-4 тижні	Працездатна система
7	Розробка розділів пояснювальної записки.	5 тиждень	ПЗ
8	Розробка слайдів презентації.	6-7 тижні	Слайди презентації
9	Захист роботи.	8 тиждень	

Студент(ка)

Підпис
(підпис)

Олександр ПЕТУНІН
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Андрій ОЛІЙНИК
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи містить: 77 с., 4 табл., 27 рис., 4 дод., 46 джерел.

МЕТАЕВРИСТИЧНІ АЛГОРИТМИ, АЛГОРИТМ ПОШУКУ A^* , ГРАФ, ВЕРШИНА, СКЛАДУВАННЯ, РУКА МАНІПУЛЯТОР, ЛІКИ, АВТОМАТИЗАЦІЯ, ЛОГІСТИКА, ЖАДІБНІ АЛГОРИТМИ, VRP.

Об'єкт дослідження – процес організації автоматизованого складу лікарських засобів.

Предмет дослідження – робота лікарських складів.

Методи дослідження – модифіковані методи дослідження операцій, зокрема жадібні алгоритми.

Важливість і проблема цієї роботи полягає в тому, що відбувся сплеск видів лікарських засобів, що змушує фармакологів витратити час на сортування та пошук. Великий обсяг даних і швидка зміна їх параметрів зумовлюють необхідність створення нових методів складування ліків.

У цій роботі розглядаються та вивчаються методи вирішення автоматизації лікарського складу. Було розроблено метод розрахунку маршрутів, які вирішують проблему комбінаторної оптимізації при визначенні оптимального маршруту для доставки ліків.

Основна відмінність нещодавно розробленого методу від існуючих полягає в тому, що він розглядає усі аспекти проблеми та використовують спеціалізовані методи кодування для потенційних рішень, що дозволяє оптимально адаптуватись до унікальних характеристик конкретної проблеми, яка розглядається.

Запропонований метод гарантує постійний темп у процесі пошуку для заданої кількості кроків, без довільного обмеження області пошуку або

вимагаючи додаткових експертних знань щодо життєздатності потенційних рішень.

Результатом роботи став програмний продукт призначений для визначення маршруту механічної руки на складі ліків.

Основні показники: Розраховані маршрути.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 77 pages, 4 tables, 27 figures, 4 appendixes, 46 sources.

METAHEURISTIC ALGORITHMS, A* SEARCH ALGORITHM, GRAPH, VERTEX, STACKING, HAND MANIPULATOR, DRUGS, AUTOMATION, LOGISTICS, GREEDY ALGORITHMS, VRP.

The object of research is the process of organizing the automated composition of medicines.

The subject of the research is the work of medicinal compounds.

Research methods – modified operations research methods, in particular greedy algorithms.

The importance and challenge of this work is that there has been a surge in drug types, forcing pharmacologists to spend time sorting and searching. A large volume of data and a rapid change in their parameters make it necessary to create new methods of drug storage.

This work examines and studies the methods of solving the automation of the pharmaceutical composition. A route calculation method was developed that solves the problem of combinatorial optimization when determining the optimal route for drug delivery.

The main difference between the newly developed method and the existing ones is that it considers all aspects of the problem and uses specialized coding methods for potential solutions, which allows optimal adaptation to the unique characteristics of the specific problem under consideration.

The proposed method guarantees a constant pace in the search process for a given number of steps, without arbitrarily limiting the search area or requiring additional expert knowledge about the viability of potential solutions.

The result of the work was a software product designed to determine the route of a mechanical arm in a drug warehouse.

Main indicators: Calculated routes.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	10
Вступ.....	11
1 Аналіз предметної області	14
1.1 Огляд існуючих методів вирішення завдання	14
1.2 Огляд існуючого програмного забезпечення, що вирішує аналогічні завдання	15
1.2.1 Огляд програми Softeon Warehouse Manager.....	15
1.2.2 Огляд програми Sortly Warehouse Manager.....	17
1.2.3 Огляд програми Netsuit Warehouse Manager.....	18
1.2.4 Порівняльна характеристики наявних систем.....	20
1.3 Постановка завдання роботи	21
1.4 Висновки за розділом 1	21
2 Вибір та обґрунтування системи, яка проєктується.....	23
2.1 Вибір інструментарію розробки програмного забезпечення	23
2.1.1 Огляд особливостей мови програмування.....	25
2.1.2 Огляд особливостей обраного середовища розробки.....	28
2.2 Структурна схема програми	28
2.3 Опис модулів розроблюваної системи	29
2.4 Висновки за розділом 2	31
3 Основні рішення щодо реалізації системи, яка проєктується.....	32
3.1 Алгоритм функціонування розробленої програми.....	32
3.1.1 Алгоритм пошуку A^*	32
3.1.2 Генетичний алгоритм мурашиної колонії	35
3.2 Висновки за розділом 3	37
4 Керівництво програміста	38
4.1 Призначення й умови застосування програми	38
4.2 Характеристики програми	38
4.3 Звертання до програми.....	39

	9
4.4 Початкові та вихідні дані	39
4.5 Повідомлення програмісту	39
4.6 Висновок	40
5 Керівництво оператора	41
5.1 Призначення програми.....	41
5.2 Умови виконання програми	41
5.3 Виконання програми	41
5.4 Повідомлення оператору.....	44
5.5 Висновок.....	45
6 Тестування і результати.....	46
6.1 Опис методики проведення тестування	46
6.2 Результати використання програмного забезпечення.....	47
Висновки.....	48
Перелік джерел посилань	49
Додаток А Технічне завдання.....	54
Додаток Б Опис програми	58
Додаток В Фрагменти тексту програми.....	65
Додаток Г Слайди презентації.....	70

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AGV – automated guided vehicles;
AIMS – automated inventory management system;
APS – automated picking system;
ARS– automatic replenishment system;
ASRC– automatic storage and retrieval crane;
AUS – automated inspection systems;
FIFO – first-in, first-out;
GTP – goods-to-person;
IDE – integrated development environment;
БСО – блок складського обліку;
МК – мікроконтролер;
ПЗ – програмне забезпечення;
ПП – програмний продукт;
ПРП – планування ресурсів підприємства;
PI – радіочастотна ідентифікація;
СУК – система управління складом;
ТРО – транспортно-розвантажувальне обладнання;
УОП – управління охорони праці;

ВСТУП

Складування є одним із найважливіших етапів будь-якого ланцюжка поставок. Воно включає в себе всі дії, пов'язані зі зберіганням і управлінням продуктами до їх доставки клієнтам. Основна мета складування — гарантувати, що продукти надходять до місця призначення вчасно та з найкращою можливою якістю [1]. Однак через свою складність складування часто вимагає багато часу та зусиль від працівників, яким потрібно вручну відстежувати рівень запасів, контролювати швидкість обертання акцій тощо, що може призвести до затримок та інших проблем, які впливають на якість послуг, що надаються постачальниками медичних послуг [2].

Індустрія охорони здоров'я зазнає швидких трансформацій, частково зумовлених прогресом у технологіях автоматизації. Однією з сфер, яка готова до значних змін, є медичний склад. Традиційно медичні склади були трудомісткими операціями, коли персонал вручну підбирав і пакував предмети для відправлення [3]. Але в міру розвитку технологій автоматизації медичні склади все частіше впроваджують автоматизовані системи, щоб оптимізувати свою роботу та підвищити ефективність. Автоматизація дає ряд переваг для медичних складів. Це може зменшити потребу в фізичній праці, звільнивши персонал, щоб зосередитися на більш важливих завданнях. Це також може підвищити точність і зменшити кількість помилок, гарантуючи, що пацієнти отримають потрібні ліки та медичні матеріали в потрібний час [4]. Але впровадження автоматизації на медичному складі не позбавлене труднощів. Це вимагає ретельного планування та врахування таких факторів, як розмір і планування складу, типи продуктів, що зберігаються та відвантажуються, а також потреби персоналу, який працюватиме з автоматизованими системами [5]. У цій статті ми дослідимо поточний стан автоматизації медичних складів, переваги та проблеми впровадження автоматизації, а також найкращі практики для успішного впровадження. Незалежно від того, чи є ви медичним працівником, менеджером складу чи просто зацікавлені у майбутньому автоматизації, ця стаття надасть цінну

інформацію про цю захоплюючу сферу, яка швидко розвивається. Медичні склади — це спеціалізовані приміщення, які керують переміщенням і зберіганням медичних засобів, обладнання та медикаментів [6]. Ці предмети потрібно відстежувати, зберігати та відправляти швидко й ефективно, щоб гарантувати, що вони прибудуть до місця призначення вчасно. Однак ручні процеси можуть займати багато часу та бути схильними до помилок, що призведе до затримок, втрати запасів і марної витрати ресурсів. Рішення для автоматизації роботи медичних складів спрямовані на оптимізацію цих процесів і підвищення ефективності. Ці рішення передбачають впровадження передових технологій, таких як роботизовані системи збирання та розміщення, штрих-коди, мітки RFID та автоматизоване програмне забезпечення для керування інвентаризацією та робочими процесами збирання та пакування [7]. Ці технології допомагають зменшити людські помилки, мінімізувати дефіцит і скоротити час виконання робіт, що зрештою призводить до покращення результатів лікування пацієнтів. Однак такі рішення автоматизації також створюють ряд проблем. Однією з ключових проблем є потреба в спеціалізованому обладнанні та програмному забезпеченні, придбання та обслуговування яких може бути дорогим. Крім того, впровадження таких систем потребує значних інвестицій у навчання та підтримку, а також може вимагати спеціалізованого персоналу для керування та обслуговування. Незважаючи на ці проблеми, переваги автоматизації операцій на складі медичних препаратів численні [8]. Спрощуючи процеси та підвищуючи точність, автоматизація може допомогти зменшити витрати, покращити догляд за пацієнтами та підвищити загальну ефективність і результативність операцій охорони здоров'я. Оскільки технології продовжують розвиватися, стає очевидним, що автоматизація відіграватиме дедалі важливішу роль у майбутніх операціях медичних складів [9].

Актуальність роботи полягає у зрості кількості видів препаратів та населення, що їх потребує. Звичайна людина не має можливості запам'ятовувати та швидко розрізняти велику кількість упаковок. Тому для підвищення швидкості роботи магазинів було розроблено автоматичні руки-контролери, які

контролюються нашою програмою [10]. Розробка призначена для вирахування маршрутів даних рук на складі лікарських засобів.

Програма буде виконувати наступні функції:

- визначення комірок, які необхідно відвідати;
- визначення кількості та параметрів механічних рук;
- розрахування оптимальних маршрутів та розподілів вмісту комірок;
- виведення розрахованих маршрутів графічно.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих методів вирішення завдання

Оскільки системи охорони здоров'я продовжують розвиватися, а попит на якісне обслуговування зростає, складність медичних складських операцій також зросла [11]. Щоб впоратися з цією складністю та забезпечити своєчасну доставку медичних товарів до закладів охорони здоров'я, медичні склади все частіше звертаються до автоматизації. Автоматизація медичних складських операцій передбачає використання робототехніки, роботизованих систем підбору та розміщення, штрих-кодів, тегів RFID, програмного забезпечення для автоматизації та інших передових технологій для оптимізації процесів комплектування, пакування та відвантаження [12]. Це не тільки підвищує ефективність і точність, але й допомагає зменшити кількість помилок, які можуть виникнути під час ручних операцій. Аналіз існуючих методів автоматизації медичних складів можна розділити на три категорії: роботизовані системи збирання та розміщення, електронні системи відстеження та управління запасами, а також автоматизовані системи пакування та доставки [13].

Роботизовані системи підбору та розміщення передбачають використання роботів для автоматизації процесів комплектування та пакування на медичних складах [14]. Ці роботи призначені для навігації складом і збирання товарів з полиць, розміщення їх у контейнерах або візках і переміщення до зони пакування. Цей процес зменшує потребу в ручній праці, знижує ризик помилок і покращує загальну ефективність складських операцій [15].

Електронні системи відстеження та управління запасами використовують штрих-коди, мітки RFID та інші технології відстеження для моніторингу руху медичних товарів та інвентарю по всьому складу [16]. Це допомагає зменшити помилки, запобігти дефіциту та забезпечити своєчасну доставку медичних товарів до закладів охорони здоров'я [17]. Інтегруючи ці системи з робототехнікою, можна контролювати процеси комплектування та пакування в

режимі реального часу, що може допомогти оптимізувати ефективність складських операцій [18].

Автоматизовані системи пакування та транспортування використовують автоматизовані машини для пакування медичних товарів і їх доставки до закладів охорони здоров'я [19]. Ці машини здатні наносити етикетки, запечатувати та транспортувати пакунки з високою швидкістю та ефективністю. Завдяки автоматизації процесів пакування та доставки медичні склади можуть зменшити потребу в ручній праці, підвищити точність процесу доставки та забезпечити своєчасну та ефективну доставку медичних товарів до закладів охорони здоров'я [20].

Підсумовуючи, автоматизація має потенціал для революції в роботі медичних складів і покращення загальної ефективності та результативності надання медичних послуг. Впроваджуючи новітні технології та об'єднуючи їх із передовим програмним забезпеченням, можна оптимізувати весь процес, зменшити кількість помилок і забезпечити своєчасну доставку медичних товарів до закладів охорони здоров'я ефективним способом. Хоча існують проблеми та недоліки автоматизації, такі як потреба в спеціалізованому персоналі та висока вартість впровадження, очевидно, що автоматизація відіграє все більш важливу роль у галузі охорони здоров'я і, ймовірно, продовжуватиме це робити в майбутньому.

1.2 Огляд існуючого програмного забезпечення, що вирішує аналогічні завдання

1.2.1 Огляд програми Softeon Warehouse Manager

Softeon (рис. 1.1) пропонує повністю інтегрований WES разом із нашою системою керування складом (WMS), яка допомагає керівникам ланцюгів постачання керувати широкою та складною інтеграцією разом із оркестровкою складу [21].



Рисунок 1.1 – Вікно програми Softeon Warehouse Manager [21]

Основні функції та характеристики програми включають:

- це облакова вебсистема;
- виконання складні функції управління запасами;
- масштабована архітектура;
- управління замовленнями;
- відстеження продуктивності;
- управління завданнями;
- повна інтеграція в автоматизовані системи обробки матеріалів.

Плюси:

- Softeon забезпечує дуже хорошу підтримку клієнтів;
- він може обробляти великі обсяги замовлень;
- ця система зменшила ймовірність того, що вона потребуватиме

постійної підтримки.

Мінуси:

- процеси впровадження не відповідають очікуванням;
- інструмент не надає функцій керування білінгом;
- система не сумісна з Mac OS і Linux OS.

1.2.2 Огляд програми Sortly Warehouse Manager

Sortly — це хмарне програмне забезпечення для керування запасами, розроблене для малого та середнього бізнесу. Він пропонує такі функції, як відстеження активів, сканування штрих-кодів, налаштовані етикетки та звіти про запаси. За допомогою Sortly користувачі можуть упорядковувати свій інвентар, створюючи візуальні каталоги та теги, що полегшує пошук і пошук товарів. Sortly також дозволяє користувачам встановлювати нагадування про терміни придатності, графіки технічного обслуговування та інші важливі дати, пов'язані з керуванням запасами (рис. 1.2).

Sortly — це зручне програмне забезпечення, для використання якого не потрібні технічні знання чи навчання. Він доступний з будь-якого пристрою з підключенням до Інтернету, що робить його зручним для компаній керувати своїми запасами з будь-якого місця. Загалом Sortly є надійним і економічно ефективним рішенням для підприємств, які прагнуть оптимізувати процеси управління складом і покращити відстеження запасів [22].

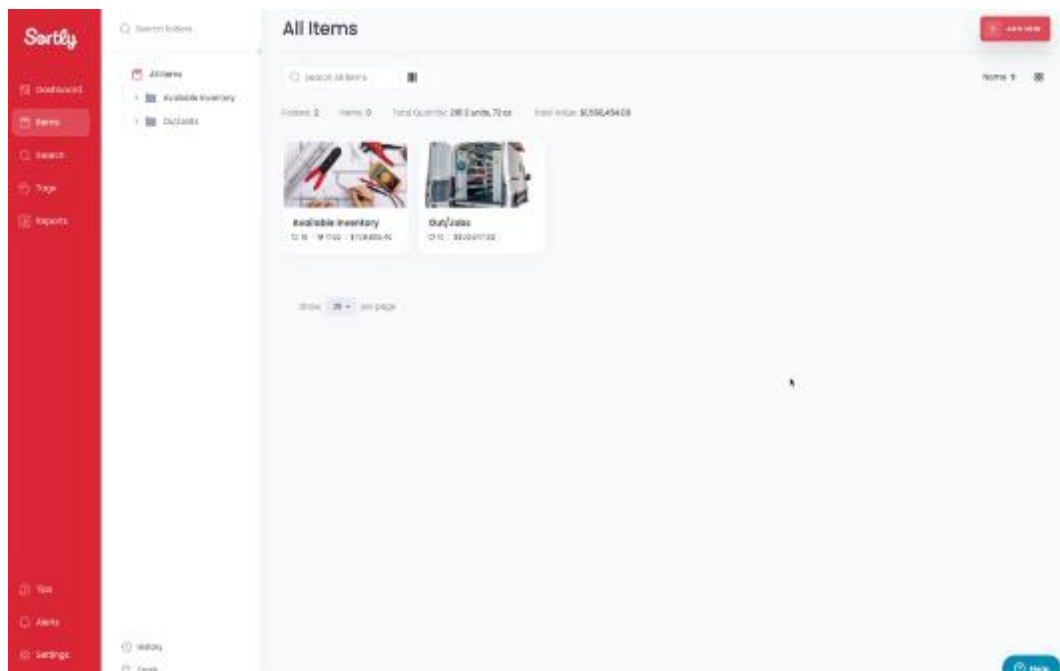


Рисунок 1.2 – Вікно програми Sortly Warehouse Manager [22]

Основні функції та характеристики програми включають:

- сканування й оновлення елементів за допомогою QR-міток і штрих-кодів;
- створення автоматичні сповіщення для відстеження рівня запасів;
- призначення ролей користувачів і керує правами доступу;
- відстежування запаси та активність користувачів;
- створення власних звітів CSV і PDF.

Плюси:

- дуже простий у використанні, не вимагає технічних знань;
- кросплатформне програмне забезпечення, яке працює майже на всіх типах пристроїв;
- чиста та повна інформаційна панель користувача;
- легко створювати персоналізовані QR-мітки та штрих-коди;
- доступна безкоштовна версія.

Мінуси:

- ніяких інтеграцій;
- надаватиметься лише довідковий центр і підтримка електронною поштою, якщо ви не виберете їхній індивідуальний план.

1.2.3 Огляд програми Netsuit Warehouse Manager

NetSuite WMS (Warehouse Management System) — це хмарне рішення, яке допомагає підприємствам ефективно керувати своїми складськими операціями. Він забезпечує видимість запасів, замовлень і виконання в реальному часі, дозволяючи підприємствам оптимізувати свої складські процеси, зменшити витрати та підвищити рівень задоволеності клієнтів (рис. 1.3).

Завдяки інтеграції NetSuite WMS з іншими модулями NetSuite, такими як керування замовленнями та фінанси, компанії можуть оптимізувати свої

складські та розподільні операції, підвищити точність і зменшити введення даних вручну [23].



Рисунок 1.3 – Вікно програми Netsuit Warehouse Manager [23]

Основні функції та характеристики програми включають:

- мобільне сканування штрих-кодів RF;
- визначення стратегії прибирання та збирання. Є три переваги цієї функції, а саме мобільне отримання, мобільний збір і підрахунок мобільного циклу;
- управління завданнями;
- повертає квитанцію авторизації;
- плани підрахунку циклу;
- існують також інші функції, як-от дані в реальному часі, керування замовленнями, відстеження штрих-кодів, виставлення рахунків, керування запасами, керування доставкою, керування робочою силою та завантаженням, карта складу, закупівлі тощо.

Плюси:

- це просто і так легко використовувати;
- компанія надає гарантію;
- контактну інформацію добре видно;
- він має сторонні плагіни;
- бездротове складування;

1.2.4 Порівняльна характеристики наявних систем

Програмами для керування складами, було створено їх порівняльну характеристику (табл. 1.2).

Таблиця 1.2 – Порівняльна характеристика наявних програмних засобів

Критерій порівняння	Системи автоматизованого керування складами		
	Softeon WMS [21]	Sortly WMS [22]	Netsuite WMS [23]
1	2	3	4
Основні функції	Управління запасами, виконання замовлень, підтримка ланцюга поставок	Управління запасами, відстеження активів	Управління запасами, виконання замовлень, інтеграція ERP
Операційна система	Windows	Хмара	Хмара або Windows
Галузева орієнтація	Різні галузі, включаючи роздрібну торгівлю, виробництво, логістику	Малі та середні підприємства, склади	Різні галузі, включаючи роздрібну торгівлю, виробництво, логістику
Тип програми	Десктопна	Вебдодаток	Десктопна та Вебдодаток
Інтерфейс користувача	Інтуїтивний, орієнтований на користувача	Простий та зрозумілий	Інтуїтивний, орієнтований на користувача
Популярні випадки використання	Управління складом, оптимізація ланцюга поставок	Управління інвентарем малого бізнесу, відстеження активів	Управління складом, інтеграція з ERP

Продовження таблиці 1.2

1	2	3	4
Формат збереження даних	CSV, Excel, PDF, JSON, XML, та CSV	CSV, Excel, PDF, JSON, XML, та CSV	CSV, Excel, PDF, JSON, XML, та CSV
Підтримка користувачів	24/7 підтримка, навчання та документація	Онлайн підтримка, навчання та документація	24/7 підтримка, навчання та документація
Формування звітів	Так	Так	Так

Зважаючи усі за характеристики програм найкращою була обрана Netsuite WMS, через можливість десктопного та веб застосування, що дозволяє не залежати від інтернет з'єднання, дистанційно відстежувати роботу та створювати бекапи і звіти онлайн.

1.3 Постановка завдання роботи

ПП повинен забезпечувати виконання нижче перерахованих функцій:

- введення даних для побудови маршруту;
- вирішення задачі маршрутизації;
- виведення результатів у графічному форматі;
- виведення попереджень при помилках.

1.4 Висновки за розділом 1

В першому розділі було проаналізовано предметну область, її історію та вже існуючі програмні аналоги проекту.

У ході аналізу було виявлено переваги та недоліки аналогів, функції, що повинен задовольняти проєкт.

Актуальність була доведена збільшеною кількістю видів медичних препаратів та необхідністю їх швидко розшукувати та переносити.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

Початок розробки будь-якого проєкту залежить від вибору середовища розробки, яке суттєво впливає на простоту кодування, доступність фреймворків, можливості рефакторингу та швидкість компіляції. Деякі з розглянутих програм були Visual Studio, Visual Studio Code та Eclipse.

Eclipse IDE - це інтегроване середовище розробки, яке використовується в комп'ютерному програмуванні [24]. Він містить фундаментальний робочий простір і можливість завантаження плагінів для адаптації до вимог користувача.. Eclipse переважно написано мовою Java і в основному використовується для роботи з нею, але його також можна використовувати для розробки додатків іншими мовами програмування за допомогою плагінів, зокрема C, C++, C#, Clojure, Haskell, JavaScript, Perl, PHP, Python, R, Ruby (включаючи структуру Ruby on Rails), Rust, Scala та Scheme. Також є можливість розробки документів з LaTeX (через плагін TeXlipse) і програмних пакетів Mathematica [25]. Eclipse IDE є безкоштовним і доступним для використання на різних операційних системах, включаючи Windows, macOS та Linux. Завдяки своїй гнучкості, розширюваності та підтримці численних мов програмування та технологій, він є популярним вибором серед розробників для створення різноманітних програмних рішень, від простих додатків до складних корпоративних систем.

Visual Studio Code, також відомий як VS Code, — це редактор вихідного коду, розроблений Microsoft з використанням Electron Framework для Windows, Linux і macOS [26]. Функції включають налагодження, кольорове підсвічування синтаксису, автоматичне завершення коду, рефакторинг та доступ до Git. Користувачі можуть налаштувати тему, комбінації клавіш, налаштування та завантажити розширення, для покращення функціональності. Visual Studio Code — це редактор вихідного коду, призначений для роботи з такими мовами як C, C#, C++, Java, JavaScript, Python. Visual Studio Code використовує компонент

редактора «Монасо», що використовується в Azure DevOps. Visual Studio Code забезпечує базові функції для програмування, такі як підсвічування синтаксису, зіставлення дужок, згортання коду та спеціальні фрагменти. Він також підтримує систему IntelliSense і підтримує роботу з Node.js. Додаткова підтримка мови виконується за допомогою безкоштовних плагінів на VS Code Marketplace. Visual Studio Code дає можливість працювати з одним або кількома каталогами водночас, які можна встановити як робочі області для майбутнього використання. Ця конфігурація робить його універсальним редактором для будь-якої мови. Функціональність Visual Studio розширюється за допомогою центрального сховища плагінів, серед яких є доповнення редактора та підтримка нових мов.

Visual Studio — це набір продуктів Microsoft, який містить інтегроване середовище розробки програмного забезпечення та різні інструменти. Ці продукти полегшують створення консольних додатків, ігор і програм із графічним інтерфейсом, UWP, сайти, додатки, служби на таких платформах, як Windows, Windows Mobile, .NET Framework, .NET Core, Xbox, iOS, Android. Visual Studio може похвалитися редактором вихідного коду, оснащеним технологією IntelliSense і простими можливостями рефакторингу коду. Має підтримку роботи налагоджувача як на рівні вихідного коду, так і на рівні машини. Серед інших інтегрованих інструментів наявні редактор форм для графічного інтерфейсу програми, редактор, дизайнер класів і дизайнер схем бази даних. Visual Studio дозволяє розробляти та інтегрувати додатки (плагіни) сторонніх виробників для такого розширення функціональності, як додавання підтримки систем контролю версій вихідного коду, нових наборів інструментів, або інструменти орієнтовані на інші.

На таблиці 2.1 порівнюються розглянуті середовища розробки. Для розробки цього проєкту було обрано Visual Studio Code, через простоту роботи, підтримку розширень, швидкість компіляції.

Таблиця 2.1 – Порівняння середовищ розробки

Критерій порівняння	Середовища розробки		
	Visual studio code	Visual Studio	Eclipse
Основна підтримувана мова	.NET (C#, VB.NET, F#), C++, Python, JavaScript	Різні (розширюване, підтримує багато мов через розширення)	Java, C/C++, PHP, Python, JavaScript тощо
Інтерфейс користувача	Комплексний і багатофункціональний	Легкий і швидкий	Комплексний і багатофункціональний
Вбудовані інструменти	Розширене налагодження, тестування, профілювання	Базове налагодження, тестування (розширюване)	Розширене налагодження, тестування, профілювання
Інтеграція з системами контролю версій	Вбудовані Git, GitHub, Azure DevOps	Вбудований Git, розширюваний іншими системами	Вбудовані Git, SVN, CVS та інші системи
Спільнота та підтримка	Сильна, офіційна та спільнотна підтримка	Сильна, офіційна та спільнотна підтримка	Сильна, офіційна та спільнотна підтримка
Популярні випадки використання	Корпоративні додатки, комплексні системи, великі проекти	Веб-розробка, скрипти, легка розробка	Java-додатки, великі корпоративні проекти

2.1.1 Огляд особливостей мови програмування

Python — це високорівнева мова програмування загального призначення, створена Гвідо ван Россумом і вперше випущена в 1991 році. Python відома своєю простотою, читабельністю та компактністю, що робить її чудовим

вибором як для новачків, так і для досвідчених програмістів[29]. Python є популярною мовою завдяки своїй простоті та читабельності, що робить її чудовим вибором для початківців, які тільки починають працювати з програмуванням. Python підтримує кілька парадигм програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування. Стандартна бібліотека модулів покриває усі потреби програміста та доступна для використання без додаткової інсталяції. Python часто використовується для розробки, аналізу даних, штучного інтелекту, машинного навчання, наукових обчислень і автоматизації. Python має величезну спільноту розробників, і є численні сторонні модулі та бібліотеки, доступні для використання. Спільнота Python дуже активна, і мова постійно розвивається новими функціями та вдосконаленнями [30]. Python є інтерпретованою мовою, що означає, що код не компілюється в машинний код перед його виконанням. Замість цього інтерпретатор Python читає вихідний код і перекладає його в машинний код під час виконання. Це полегшує вивчення та використання Python, оскільки немає потреби мати справу зі складними процесами збирання. Python також є кросплатформним, тобто він може працювати на кількох операційних системах, включаючи Windows, macOS і Linux. Код Python можна написати один раз і запустити на будь-якій платформі з інстальованим інтерпретатором Python. Python має велику та зростаючу екосистему інструментів і бібліотек, що робить його універсальною мовою для широкого спектру програм. Простота й читабельність Python роблять його чудовим вибором для початківців, а його потужність і гнучкість роблять його популярним вибором і для досвідчених програмістів [31]. Деякі популярні бібліотеки та фреймворки в Python включають NumPy, SciPy, Pandas, Matplotlib, TensorFlow, Keras, PyTorch, Flask, Django та Scikit-learn, серед багатьох інших. Ці бібліотеки та фреймворки забезпечують функціональність для аналізу даних, машинного навчання, розробки, наукових обчислень тощо. Python також є чудовою мовою для створення сценаріїв і автоматизації [32]. Сценарії Python можна використовувати для автоматизації повторюваних завдань, таких як обробка даних, керування файлами та

адміністрування системи. Python також широко використовується для розробки настільних програм, програм і мобільних програм. Python — це універсальна мова, яка широко використовується в промисловості, наукових колах і дослідженнях. Його простота, читабельність і потужність роблять його чудовим вибором для програмістів будь-якого рівня кваліфікації [33].

Pygame: Pygame — це набір модулів Python, призначених для написання відеоігор. Він надає функціональні можливості для обробки графіки, звуку та введення, що робить його популярним вибором для розробки 2D-ігор. Pygame простий у вивченні та використанні і постійно розвивається завдяки свої спільноті, яка робить внесок у його розвиток і надає необхідну підтримку [34]. Однак Pygame може бути не найкращим вибором для розробки 3D-ігор або ігор, які потребують розширених графічних можливостей.

PuLP: PuLP — це бібліотека Python для лінійного програмування. Він надає простий у використанні інтерфейс для створення та вирішення задач лінійного програмування. PuLP підтримує різноманітні розв’язувачі, зокрема розв’язувачі з відкритим кодом, такі як GLPK і COIN-OR, а також комерційні розв’язувачі, такі як CPLEX і Gurobi. PuLP є чудовим вибором для вирішення проблем оптимізації, таких як оптимізація ланцюжка поставок, розподіл ресурсів і планування. Однак він може бути непридатним для більш складних задач оптимізації, які вимагають передових методів математичного моделювання. Він надає простий і легкий у використанні API для створення вікон, кнопок, міток, текстових полів та інших компонентів графічного інтерфейсу [35].

Tkinter — це кросплатформна бібліотека, яка підтримує Windows, macOS і Linux. Це хороший вибір для простих програм із графічним інтерфейсом користувача, таких як калькулятори, блокноти та файлові провідники. Однак Tkinter може бути непридатним для більш складних програм із графічним інтерфейсом користувача, які вимагають розширених функцій, таких як анімація, відтворення відео або 3D-графіка. Загалом Pygame, PuLP і Tkinter є чудовими бібліотеками та фреймворками для своїх цілей. Pygame — чудовий вибір для розробки двовимірних ігор, PuLP — потужний інструмент для лінійного програмування, а

Tkinter — проста та зручна бібліотека для створення GUI. Однак важливо зазначити, що ці бібліотеки та фреймворки мають свої обмеження, і розробники повинні вибрати правильний інструмент для відповідної роботи на основі своїх конкретних потреб і вимог.

2.1.2 Огляд особливостей обраного середовища розробки

Таблиця 2.2 – Порівняння середовищ розробки

Критерій порівняння	Середовища розробки		
	Visual Studio	Visual Studio Code	Eclipse
Багатоплатформність	+	+	+
Швидкість модернізації	++	++	+
Наявність візуального конструктора	++	++	++
Підтримка репозиторіїв	+	+	+
Швидкість компіляції	++	++	+
Додатки	++	++	+
Зручність	++	++	+

2.2 Структурна схема програми

Розроблюваний ПП повинен обов’язково містити такі модулі:

- файл формату CSV;
- флгоритмів;
- графічний;
- модуль зв’язку всіх елементів ПП.

Зв’язок між модулями можна побачити на рис. 2.1.

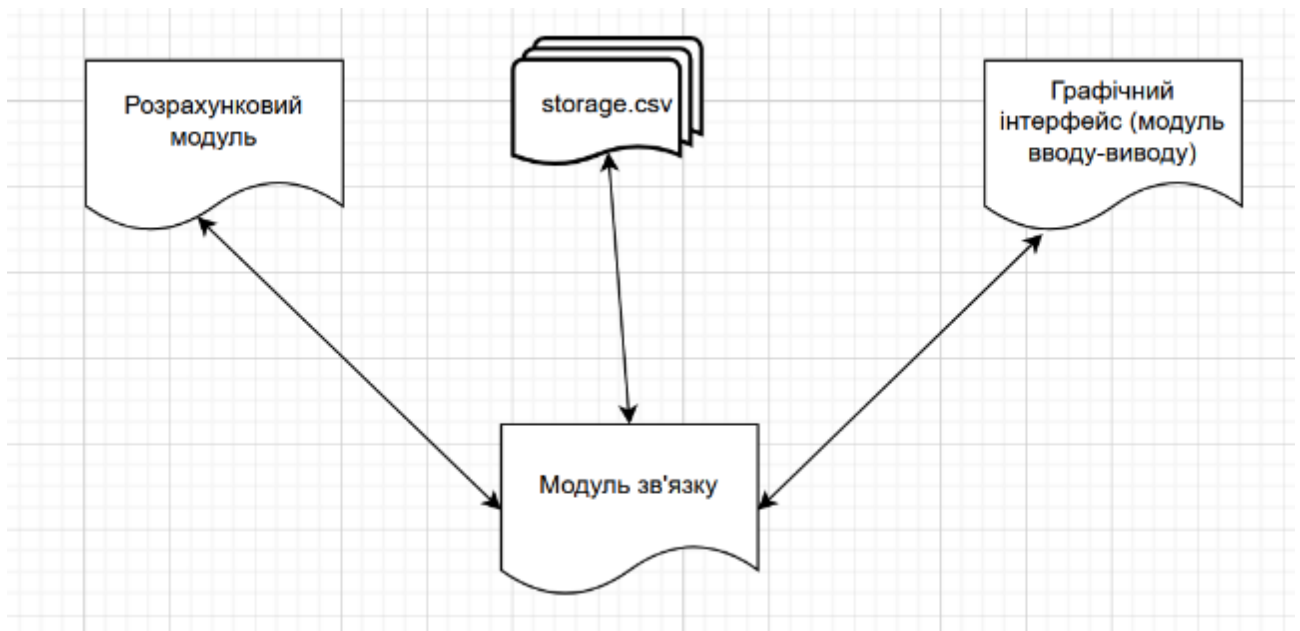


Рисунок 2.1 – Структурна схема програми

2.3 Опис модулів розроблюваної системи

Розроблюваний програмний продукт використовує дані про простір для побудови маршруту, що включає в себе інформацію про кількість та вміст комірок. Ці дані можуть бути збережені в різних типах сховищ:

- текстові файли (CSV, txt, xml,...);
- бінарні файли даних;
- не реляційні бази даних (NoSQL сховища);
- реляційні бази даних (MySQL, MS SQL Server, Firebird, ...).

Обраний метод зберігання повинен забезпечувати достатню гнучкість форматів, високу швидкість доступу до даних, здатність для зберігання великих об'ємів інформації, можливість розширення функціоналу і при цьому не призводити до надмірного використання пам'яті. До наведених функцій найбільше підходить CSV файл із описом середовища.

Переваги використання CSV файлу:

- простота і доступність при створенні та заповненні даних;
- повна незалежність даних. Зміни в прикладній програмі при зміні реляційної БД мінімальні;

- мінімальний обсяг файлів для збереження даних.

Виходячи з вищесказаного для зберігання даних було обрано CSV файл.

Розрахунковий модуль – це основний та найважливіший модуль, його призначенням є реалізація методів пошуку маршруту для кожної руки. Даний модуль включає в себе наступні субмодулі (скрипти):

- модуль для визначення положення рук у просторі;
- модуль, що реалізує розподіл замовлень по руках;
- модуль, що реалізує перегляд доступних шляхів;
- модуль, що реалізує метод визначення оптимального шляху.

Модуль вводу-виводу необхіден для зручного корегування зв'язку користувача з програмою і включає в себе наступні субмодулі (скрипти):

- модуль для введення параметрів середовища;
- модуль для введення комірок-замовлень;
- модуль для виведення позиції рук;
- модуль для виведення шляху руки.

Модуль зв'язку всіх елементів ПП, як і видно з назви потрібен для передачі даних між модулями

Самі модулі програми та їх зв'язки зображено на uml-діаграмі, яку можна побачити на рис. 2.2 для наочності.

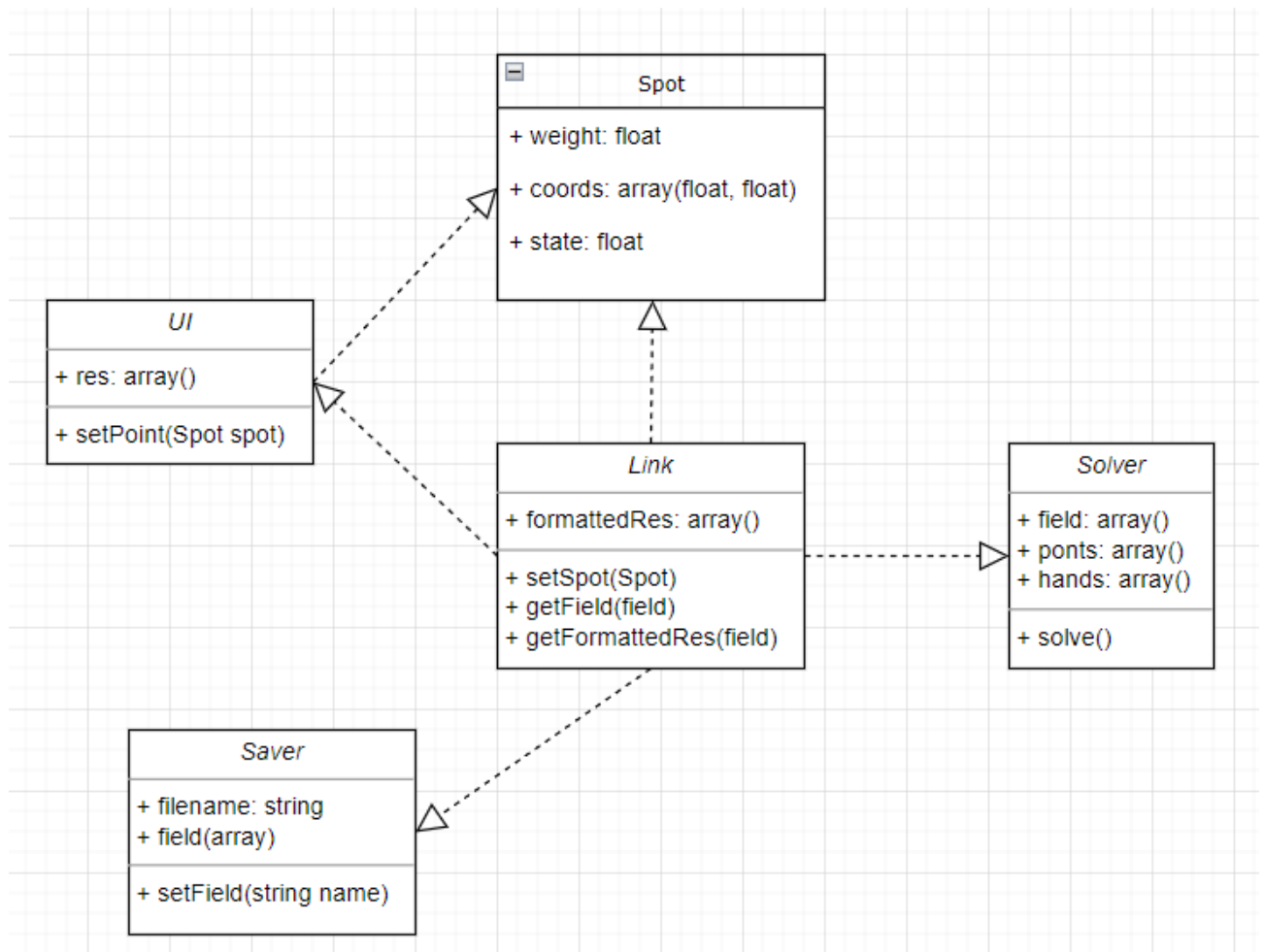


Рисунок 2.2 – Uml діаграма класів

2.4 Висновки за розділом 2

У другому розділі було розглянуто різноманітні середовища розробки та мови програмування, обрано один для зберігання даних ПП, проведено аналіз сильних і слабких сторін цих середовищ, побудовано структурну схему проєкту та надано опис модулів проєкту і зв'язків між ними.

У ході аналізу було виявлено сильні та слабкі сторони розглянутих середовищ, візуалізовано структурну схему та описано модулі проєкту були і зв'язки між ними.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм функціонування розробленої програми

Завданням роботи є програмна реалізація методів пошуку шляху для декількох однакових рук до певних комірок.

Для досягнення поставленої мети необхідно створити метод пошуку шляху для кожної руки та алгоритм відображення комірок. Для пошуку шляху, було обрано алгоритм пошуку A^* та генетичний алгоритм мурашиної колонії. Перший алгоритм потрібен для збільшення точності результату, а другий для пришвидшення початку виводу, що буде детальніше розглянуто далі.

3.1.1 Алгоритм пошуку A^*

Алгоритм A^* робить покроковий перебір всіх можливих шляхів, що ведуть від початкової вершини у кінцеву, поки не знайде мінімальний шлях (рис. 3.1) [36].

Алгоритм роботи методу A^* .

Крок 1. Переглянути вузли, що суміжні з початковим.

Крок 2. Обрати з них той, що має мінімальне значення за критерієм.

Крок 3. Визначити пріоритет шляху за визначенням критерію.

Крок 4. Перевірити, якщо значення критерію цільової вершини не менше, ніж будь-яке зі значень в черзі, перейти до кроку 5. В іншому випадку перейти до кроку 2.

Крок 5. Завершення роботи алгоритму.

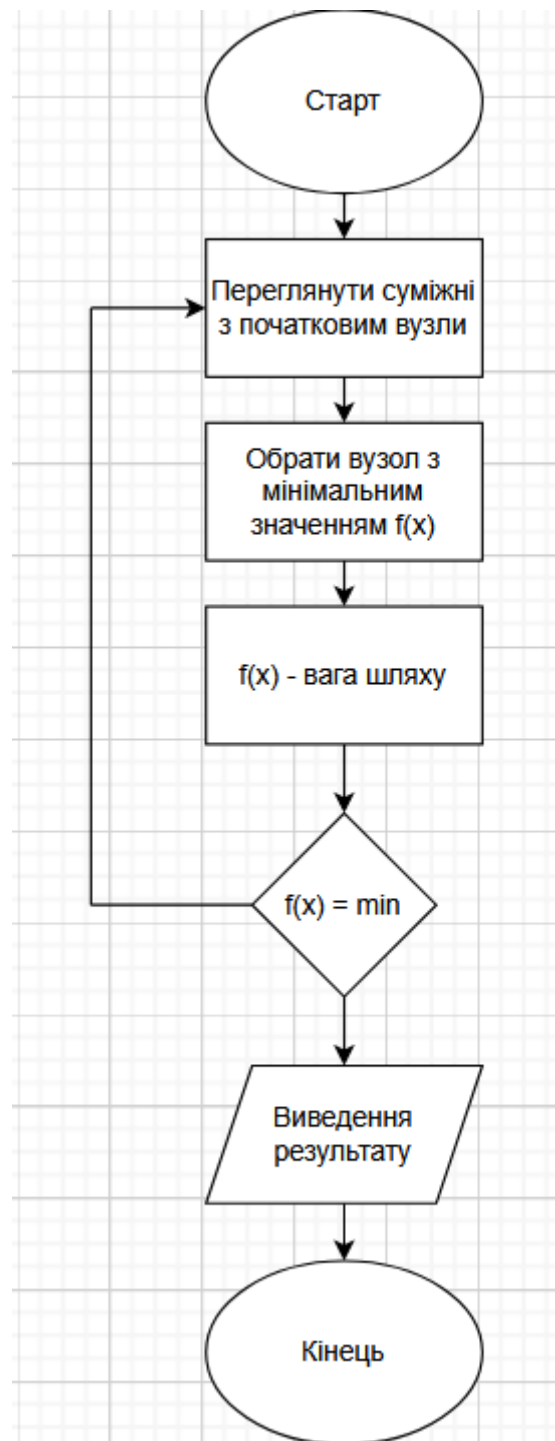


Рисунок 3.1 – Блок-схема алгоритму A*

Алгоритм пошуку A* поєднує в собі підходи алгоритму Дейкстри та жадібного пошуку. Спочатку визначається мапа, яка зберігає інформацію про попередні вершини, їх евристичні відстані до кінцевої вершини та вартості досягнення від початкової вершини [37]. Початкова вершина додається до мапи зі значеннями 0.

Далі створюються набори "відкритих" та "закритих" вершин. Початкова вершина додається до відкритого набору. Цикл продовжується, доки відкритий набір містить хоча б одну вершину. На кожній ітерації обирається вершина з найменшою сумою евристичної відстані та відстані досягнення з відкритого набору і видаляється з нього [38]. Для кожної сусідньої вершини, яка ще не знаходиться у закритому наборі, обчислюється її вартість та додається до мапи. Вершина-сусід також додається до відкритого набору, якщо вона вже там є і сума її вартостей у мапі менша або дорівнює сумі вартостей, обчислених на поточній ітерації.

Якщо вершина-сусід є кінцевою, пошук завершується. Після кожної ітерації поточна вершина додається до закритого набору, і починається наступна ітерація [39].

Після завершення циклу шлях відтворюється від кінцевої вершини до початкової за допомогою мапи, що містить інформацію про попередні вершини.

Порядок обходу вершин визначається евристичною функцією «відстань+ вартість» (зазвичай позначається як $f(x)$). Ця функція – сума двох інших: функції вартості досягнення даної вершини (x) з початковою (зазвичай позначається як $g(x)$ і може бути як евристичною, так і немає), і функції евристичної оцінки відстані від розглянутої вершини до кінцевої (позначається як $h(x)$), як у формулою (3.1):

$$f(x) = g(x) + h(x) \quad (3.1)$$

Функція $h(x)$, повинна бути допустимою евристичною оцінкою, тобто не повинна переоцінювати відстані до цільової вершини. Наприклад, для завдання маршрутизації $h(x)$, може являти собою відстань до цілі по прямій лінії, так як це фізично найменша можлива відстань між двома точками.

3.1.2 Генетичний алгоритм мурашиної колонії

Як вже було зазначено, алгоритм пошуку A^* робить покроковий перебір всіх можливих шляхів, що призводить до геометричного збільшення часу обробки задачі із збільшенням обсягу вхідних даних. Подібний недолік потребує оптимізації пошуку початкових розрахунків, що дозволить суттєво пришвидшити роботу програми, покращуючи досвід користувача при роботі з програмою [40].

Для початкової оптимізації, було вирішено обрати швидкий алгоритм, що не потребує повного перебору шляхів і усі обрані критерії зміг задовольнити алгоритм мурашиної колонії, що відноситься до виду генетичних алгоритмів. Суть підходу полягає в аналізі та використанні моделі поведінки мурах, що шукають шляхи від колонії до джерела живлення [40].

Агенти моделювання, відомі як штучні «мурахи», оптимізують рішення, просуваючись через простір параметрів, який охоплює всі можливі варіанти. Подібно до справжніх мурах, вони використовують феромони для направлення один одного до ресурсів та досліджують навколишнє середовище [41]. Феромоновий слід сприяє пошуку нових маршрутів для мурах у випадку перешкод на звичайному шляху. Якщо найкоротший маршрут стає недоступним через перешкоду, мурахам потрібно знайти новий найкоротший шлях. Спочатку вони масово обминатимуть перешкоду з обох сторін - як у дорозі до їжі, так і у зворотньому напрямку. Ті мурахи, що оберуть коротший шлях, будуть швидше проходити його в обидва напрямки, що призведе до більшого насичення феромоном цього маршруту. Оскільки вища концентрація феромону на маршруті сприяє більш ймовірному руху мурах по ньому, наступні мурахи вибиратимуть цей маршрут з більшою впевненістю, утворюючи більшу кількість феромону, поки він залишається доступним [42].

Глобальне правило оновлення феромонів в алгоритмах мурахиної колонії може включати збільшення або зменшення рівня феромонів в

залежності від того, чи були певні шляхи частиною "хороших" або "поганих" рішень.

Наприклад, одним із глобальних правил оновлення є формула (3.2):

$$t_{xy} \leftarrow (1 - \rho)t_{xy} + \sum_k^m \Delta t_{xy}^k, \quad (3.2)$$

де t_{xy} — кількість феромону, що осідає для переходу стану $xу$, ρ — коефіцієнт випаровування феромону, m — кількість мурах і Δt_{xy}^k — це кількість феромону, що виділяється k -тою мурахою.

Стало зрозуміліше, як накопичується феромон на шляху, але як же обирається для кожної мурахи наступний крок і наскільки феромон впливає на цей вибір? Розберемося на формулі (3.3):

$$p_{ij}(t) = \frac{t_{ij}(t)^a (\frac{1}{d_{ij}})^\beta}{\sum_{j \in \text{allowed notes}} T_{ij}(t)^a (\frac{1}{d_{ij}})^\beta}, \quad (3.3)$$

де $T_{ij}(t)^a$ — рівень феромону, d_{ij} — евристична відстань, а, β - константи. Якщо $a=0$, то метод вироджується до жадібного (повного перебору) — вибір найближчого елементу для кроку стає найімовірніший.

Якщо $\beta=0$, то вибір відбуватиметься виключно базуючись на рівні феромону, що призведе до локальних оптимальних розв'язків.

У мурах, шлях використовується доки не закінчиться їжа, а у нашому випадку, кількість проходжень по графу залежить виключно від часу і тому має бути обмежене кількістю ітерацій чи ступенем дослідження, щоб забезпечити ефективну роботу алгоритму. Оскільки ступень дослідження, або точність дослідження, забезпечується першим алгоритмом, для мурашиної колонії було встановлено кількість ітерацій у 1000 проходжень, що дозволить почати відображення шляху із мінімальною затримкою.

3.2 Висновки за розділом 3

У третьому розділі було розглянуто алгоритми пошуку шляхів і описані їх модифікації.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Програма призначена для вирішення задачі маршрутизації. Це забезпечить прискорення виконання задач різних підприємств.

Програма виконує наступні функції:

- введення початкових даних;
- розрахунок оптимальних маршрутів для введених даних;
- виведення розрахованих маршрутів на графічному інтерфейсі.

Програма має наступні апаратні вимоги для нормального функціонування програми. Щоб забезпечити оптимальну продуктивність, важливо відповідати цим специфікаціям або перевищувати їх.

Ось короткий перелік вимог до обладнання:

- процесор: Intel Core i5 або інший процесор з архітектурою x86, який має частоту не менше 2.5 ГГц і підтримує SSE2 та більш сучасні розширення;
- пам'ять: мінімум 8 Гб оперативної пам'яті;
- відеокарта: підтримка DirectX 11 і WDDM версії 2.0 або новішої;
- дисковий простір: SSD з ємністю не менше 256 ГБ;
- роздільна здатність екрана: Full HD (1920 x 1080 пікселів) або вище;
- пристрій введення: комфортна клавіатура і миша, які підтримують сучасні стандарти зв'язку.

Для нормального функціонування програми потрібна наявність операційної системи Windows версії 8 або вище.

4.2 Характеристики програми

Додаток виконує наступні функції:

- графічне відображення сапи
- розрахунок оптимальних маршрутів;

- збереження розрахованих маршрутів;
- візуалізація розрахованих маршрутів на карті;

4.3 Звертання до програми

Для доступу до програми, потрібно запустити файл `main.py` як звичайний скрипт пайтон, впевнившись, що усі інші модулі пристуні у папці з програмою.

4.4 Початкові та вихідні дані

Початковими даними програми є дані про середовище функціонування рук, вантажомісткість рук та про вміст комірок.

Вихідними даними є оптимальні маршрути для механічних рук.

4.5 Повідомлення програмісту

Під час роботи розроблений ПП виводить у консоль дані про шлях поточної механізованої руки, що можна побачити на рис.4.1.

```
Assign_5_1 = 0.0
Assign_5_2 = 1.0
Assign_5_3 = 0.0
Assign_5_4 = 0.0
Assign_5_5 = 0.0
Assign_5_6 = 0.0
Assign_6_1 = 1.0
Assign_6_2 = 0.0
Assign_6_3 = 0.0
Assign_6_4 = 0.0
Assign_6_5 = 0.0
Assign_6_6 = 0.0
Value of Objective Function = 143.0
```

Рисунок 4.1 – приклад повідомлення програмісту

4.6 Висновок

У даному розділі було докладно описано призначення та функції розробленого програмного продукту. Визначено умови, необхідні для його встановлення та підтримки на комп'ютері користувача та сервері. Окреслено основні характеристики та особливості створеного програмного продукту, який має відкритий код і може бути відредагований іншими розробниками програмного забезпечення, а також його модулі можуть бути використані для створення інших систем. Додатково надано опис вхідних та вихідних даних, а також перелік повідомлень, що генеруються під час роботи програми.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Додаток призначений для часткової автоматизації роботи складу медичних препаратів шляхом створення маршрутів для руки-маніпулятора. Одним з основних завдань є створення муршруту руху рук-маніпуляторів до визначених товарів.

Додаток створено в текстовому редакторі Visual Studio code.

5.2 Умови виконання програми

Для роботи програми, потрібно запускати пайтон скрипт (.py), тож необхідне встановлене середовище python.

Прорамний продукт призначений для запуску в середовищі Windows з наступники параметрами:

- процесор: Pentium-сумісна архітектура x86 з частотою 2 ГГц або вище, підтримкою PAE і NX SSE2 або система на чіпі;
- пам'ять: 4 Гб оперативної пам'яті;
- відеокарта: DirectX 9 і WDDM версії 1.3 або новішої;
- дисковий простір: 20 Гб або більше вільного дискового простору;
- роздільна здатність екрана: 800 x 600 або вище;
- пристрій введення: клавіатура та миша або аналогічний пристрій введення;
- операційна система Windows версії 8 або вище.

5.3 Виконання програми

Після вдалого запуску ПП відкривається вікно програми, що представлено на рис. 5.1. На цьому вікні можна побачити карту складу лікарських засобів. Разними кольорами позначені різні типи комірок: чорним позначені порожні чи недоступні, білим комірки із доступними ліками, а

різнокольорові комірки зліва показують поточне місцеположення рук, для яких і будуть побудовані шляхи до потрібних комірок.

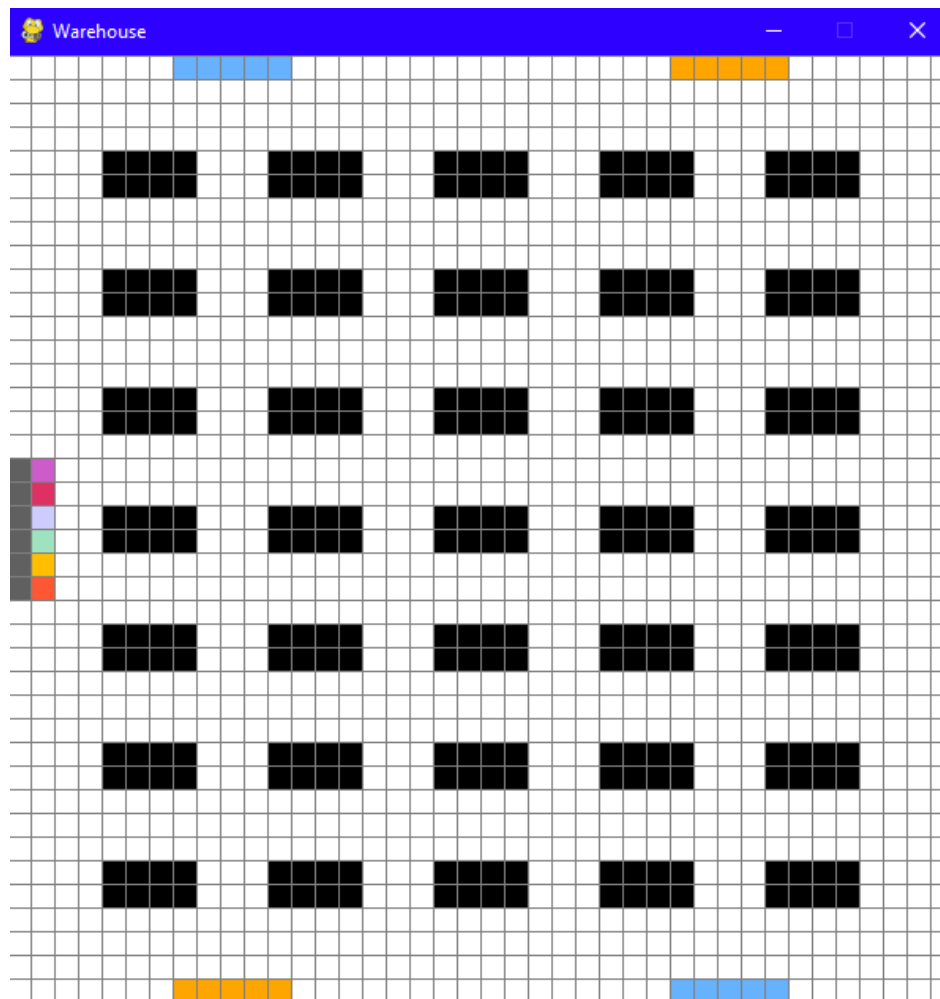


Рисунок 5.1 – Програма при запуску

Карта повинна бути завчасно побудована і внесена до програми у вигляді файлу .csv, після чого вона буде одразу відображена при запуску.

Далі користувач вводить комірки, з яких потрібно забрати вміст. Ввод може бути зроблений натисканням мишей на відповідні комірки і усі комірки, які потрібно відвідати, будуть позначені зеленим, що можна побачити на рис.5.2.

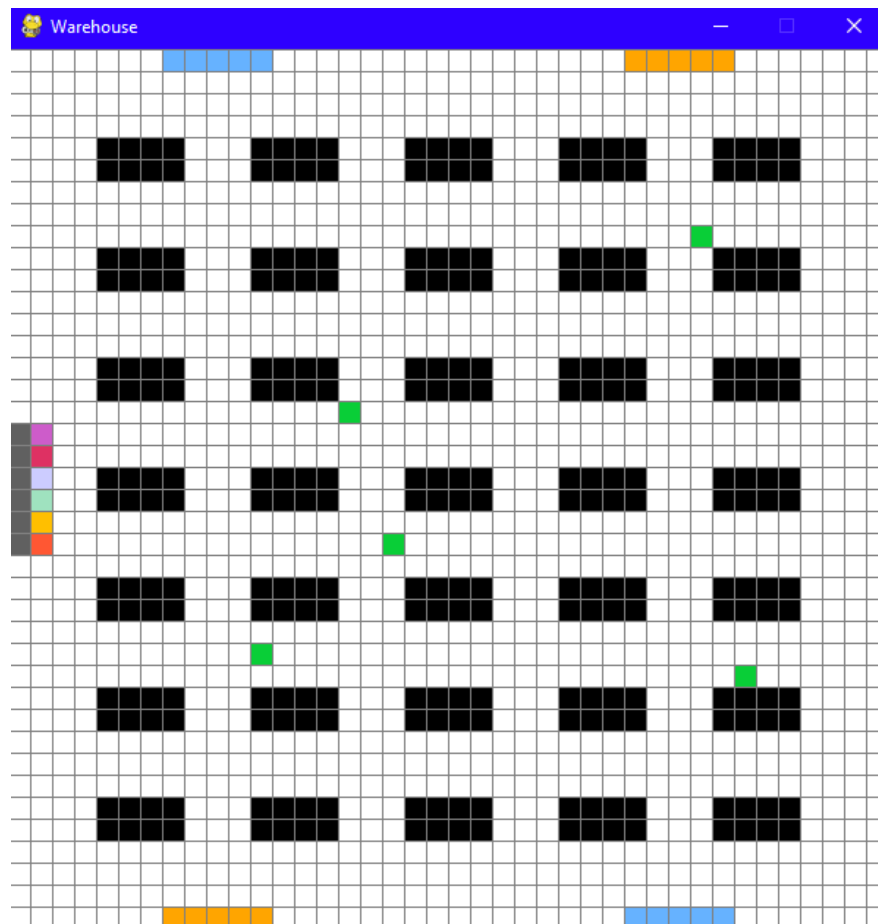


Рисунок 5.2 – Ввод комірок

Далі програма очікує натискання клавіші «Пробіл» для запуску алгоритму пошуку шляху.

Після закінчення розрахунків, програма покроково зобразить переміщення рук до комірок і вони залишаться там, де було взято останнє замовлення в очікуванні додаткових введених даних, що можна побачити на рис.5.3.

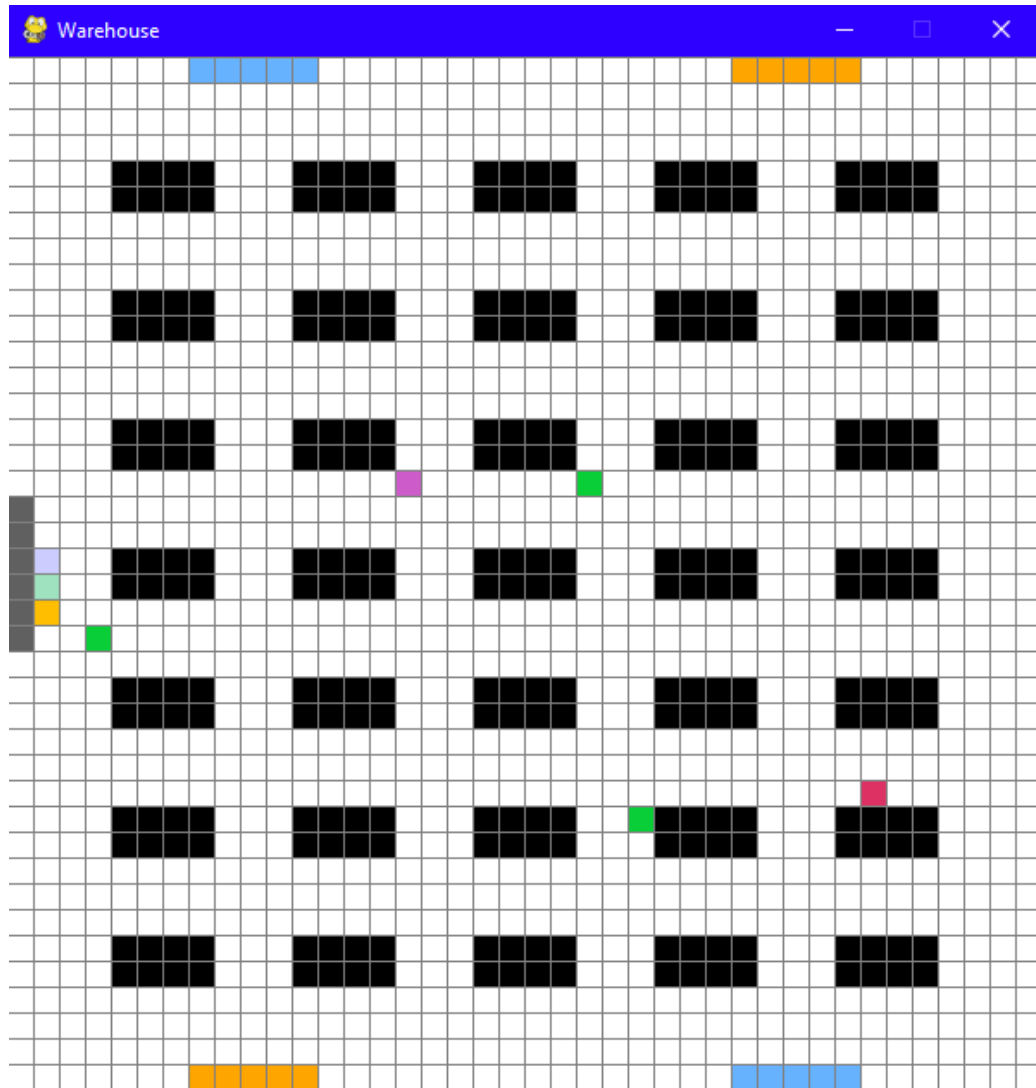


Рисунок 5.3 – Додаткові замовлення

Таким чином, програма у реальному часі будує шляхи для виконання всіх замовлень.

5.4 Повідомлення оператору

Під час виконання розробленого програмного продукту автоматично створюються повідомлення, які призначені для інформування розробника про різні події, такі як помилки, початок та завершення обчислень тощо. Ці повідомлення не вимагають втручання користувача. Перелік можливих повідомлень, їх зміст і дії, що виконуються, описані у розділі "Керівництво

програміста". Користувач бачить графічне відображення всіх кроків виконання програми, без використання текстових повідомлень.

5.5 Висновок

У цьому розділі було детально описано призначення та функції розробленого програмного продукту, а також мінімально необхідні апаратні вимоги для його встановлення та використання. Надано докладну інструкцію щодо взаємодії оператора з програмним продуктом, зокрема з приводу введення інформації та розрахунку ефективних маршрутів. Також було представлено перелік повідомлень, які генеруються під час роботи програми, а також наведено приклади вхідних та вихідних даних. Інтерфейс користувача розробленого програмного продукту є простим та інтуїтивно зрозумілим і не вимагає додаткового навчання для його операційного персоналу.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

6.1 Опис методики проведення тестування

Для проведення тестування, була введена тестова карта, що зображена на рис.6.1.

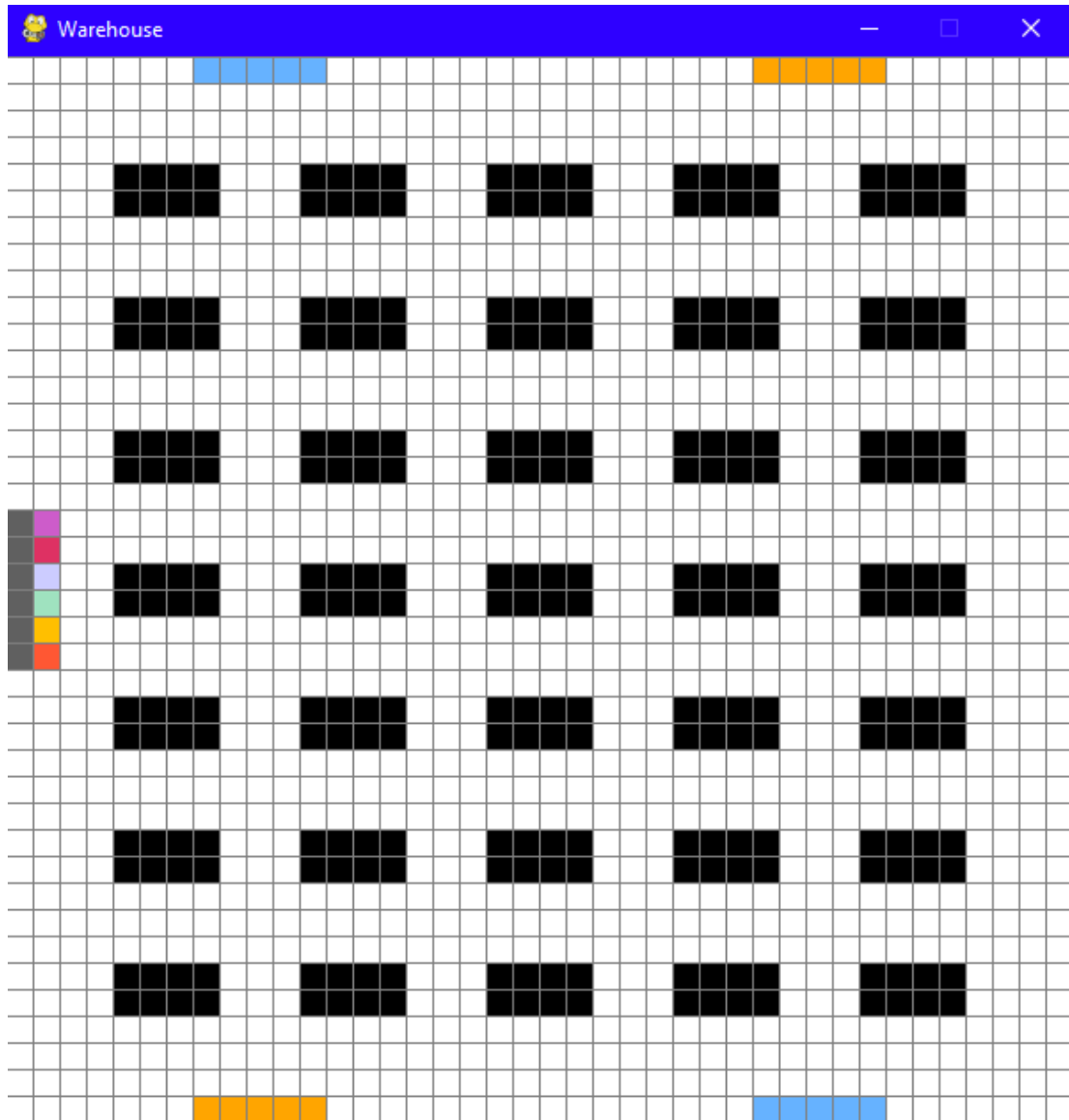


Рисунок 6.1 – Тестова карта

На цій карті було проведено інтеграційне тестування, результати якого можна побачити в наступному підрозділі.

6.2 Результати використання розробленого програмного забезпечення

Для тестування, було введено задачі із вже розрахованими оптимальними шляхами і результати представлено у вигляді табл.6.1.

Таблиця 6.1 – Результати тестування ПП

Довжина оптимального шляху	Довжина за результатом роботи ПП	Час розрахунку(с)
50	50	0,1
100	101	0,1
150	150	0,2
200	200	0,2
250	250	0,2
300	301	0,3
350	350	0,3

Як видно з таблиці, програма демонструє достатню швидкість і точність роботи.

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи бакалавра, проведено детальне ознайомлення із предметною областю – автоматизацією складу медичних препаратів. Також було розглянуто доступний інструментарій вирішення задачі і реалізовано одним із методів.

У першому розділі було проведено аналіз предметної області, включаючи її історію та існуючі аналоги проєкту, і підтверджено актуальність виконання цього проєкту.

Другий розділ охоплював вибір мови програмування Python та вивчення відповідних бібліотек, таких як `pulp`, `pygame` та `tkinter`, а також опис модулів програми.

У третьому розділі розглядались застосовані алгоритми проєкту. У результаті розроблено програмне забезпечення, яке успішно втілює усі визначені завдання та функціональні вимоги, зокрема:

1. Введення початкових даних користувача.
2. Розрахунок оптимальних маршрутів і розподілу завантажень на основі введених даних.
3. Візуалізація розрахованих маршрутів на графічному інтерфейсі.

Результати тестування програмного забезпечення підтвердили ефективність обраного алгоритму, який дозволяє досягати очікуваних результатів при роботі з тестовими даними. У майбутньому можливо розширити можливості програми шляхом додавання функціоналу автоматичного введення даних з файлів різних форматів, що дозволить значно зменшити час, необхідний для підготовки вхідних даних для обробки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Bartholdi J., Hackman S. Warehouse & Distribution Science. 4th edition. Supply Chain and Logistics Institute, Georgia Institute of Technology, 2014. C. 117–156 URL: [https://doi.org/ 10.11601/ijates.v2i3.61](https://doi.org/10.11601/ijates.v2i3.61) (date of access: 10.06.2024).
2. René Victor Valqui Vidal, European Journal of Operational Research, Volume 182, Issue 3, 2007, Pages 1492-1493, URL: <https://doi.org/10.1016/j.ejor.2006.11.001>. (date of access: 10.06.2024).
3. Chun Park, B. Performance of automated storage/retrieval systems with non-square-in-time racks and two-class storage. International Journal of Production Research, 2016, C. 1107–1123, URL: <https://doi.org/10.1080/00207540500357070> (date of access: 10.06.2024).
4. Leyla Ozsen & Collette R. Coullard & Mark S. Daskin. "Capacitated warehouse location model with risk pooling," Naval Research Logistics (NRL), John Wiley & Sons, vol. 55(4), 2008, C. 295-312. URL: <https://doi.org/10.1002/nav.20282> (date of access: 10.06.2024).
5. De Koster R., Le-Duc T., Roodbergen K. J. Design and Control of Warehouse Order Picking: A Literature Review. European Journal of Operational Research, 2017. Vol. 182. C. 481–501. URL: <https://doi.org/10.1016/j.ejor.2006.07.009> (date of access: 10.06.2024).
6. Rouwenhorst B., Reuter B., Stockrahm V., Van Houtum G. J., Mantel R. J., Zijm W. H. M. Warehouse Design and Control: Framework and Literature Review. European Journal of Operational Research, 2015. Vol. 122. C. 515–533. URL: [https://doi.org/10.1016/S0377-2217\(99\)00053-X](https://doi.org/10.1016/S0377-2217(99)00053-X) (date of access: 10.06.2024).
7. Johnson E. L., Krishnamoorthy D. M. R. Optimization and Warehousing Automation. Computers & Operations Research, 2016. Vol. 65. C. 433–447. URL: <https://doi.org/10.1016/j.cor.2015.02.004> (date of access: 10.06.2024).
8. Min S., Mentzer J. T. Developing and Measuring Supply Chain Management Concepts. Journal of Business Logistics, 2015. Vol. 36. C. 63–99. URL:

<https://doi.org/10.1002/j.2158-1592.2004.tb00170.x> (date of access: 10.06.2024).

9. Ten Hompel H., Schmidt T. Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems. Springer, 2015. C. 75–102. URL:

https://www.researchgate.net/publication/43184892_Warehouse_Management_Automation_and_Organisation_of_Warehouse_and_Order_Picking_Systems_Intralogistik (date of access: 10.06.2024).

10. Li Y., Tang J. Warehouse Automation: A Systematic Review and Analysis. Industrial Management & Data Systems, 2020. Vol. 120. C. 1389–1416. URL: <https://doi.org/10.18178/ijmlc.2021.11.3.1042> (date of access: 10.06.2024).

11. Jaghbeer, Y., Hanson, R., & Johansson, M. I. Automated order picking systems and the links between design and performance: a systematic literature review. International Journal of Production Research, 2020, 58(15), C. 4489–4505. URL: <https://doi.org/10.1080/00207543.2020.1788734> (date of access: 10.06.2024).

12. Cakici, Ozden & Groenevelt, Harry & Seidmann, Abraham. Using RFID for the management of pharmaceutical inventory-system optimization and shrinkage control. Decision Support Systems. 2011. 51. 842-852. URL: <https://doi.org/10.1016/j.dss.2011.02.003> (date of access: 10.06.2024).

13. Gu J., Goetschalckx M., McGinnis L. Research on Warehouse Operation: A Comprehensive Review. European Journal of Operational Research, 2010. Vol. 177. C. 1–21. URL: <https://doi.org/10.1016/j.ejor.2006.11.001> (date of access: 10.06.2024).

14. Kusiak E. Healthcare Operations Management: A System Engineering Approach. International Journal of Production Economics, 2015. Vol. 147. C. 242–252. URL: <https://doi.org/10.1016/j.ijpe.2013.09.012> (date of access: 10.06.2024).

15. Chung D. H., Lee S. H. Implementation of RFID in Healthcare Supply Chain. Journal of Healthcare Engineering, 2013. Vol. 4. C. 83–96. URL: https://www.researchgate.net/publication/241686806_International_Journal_of_Operations_Production_Management_Learning_on_lean_a_review_of_thinking_and_research (date of access: 10.06.2024).

16. Automation of Supply Chain Management for Healthcare. In: Troiano, L., Vaccaro, A., Kesswani, N., Díaz Rodriguez, I., Brigui, I. Progresses in Artificial Intelligence & Robotics: Algorithms & Applications. ICDLAIR. Lecture Notes in Networks and Systems, 2021, vol 441 C. 553–561. URL: https://doi.org/10.1007/978-3-030-98531-8_4 (date of access: 10.06.2024).
17. Womack J. P., Jones D. T. Lean Thinking in Healthcare Supply Chain. Journal of Operations Management, 2014. Vol. 32. C. 13–30. URL: <https://doi.org/10.1136/qshc.2009.037986> (date of access: 10.06.2024).
18. Saha, E., Ray, P.K. Modelling and analysis of healthcare inventory management systems. OPSEARCH 56, 2019, 1179–1198. Vol. 32. C. 144–162. URL: <https://doi.org/10.1007/s12597-019-00415-x> (date of access: 10.06.2024).
19. Robinson C. W., Johnson K. F. The Role of Automated Systems in Healthcare Logistics. International Journal of Logistics Management, 2020. Vol. 31. C. 42–56. URL: <https://doi.org/10.1108/IJLM-03-2018-0097> (date of access: 10.06.2024).
20. Warehouse Management System (WMS). Softeon. URL: <https://www.softeon.com/solutions/warehouse-management-system-wms/> (дата звернення: 20.05.2020).
21. Warehouse Fulfillment. NetSuite. URL: <https://www.netsuite.com/portal/products/erp/warehouse-fulfillment/wms.shtml> (дата звернення: 20.05.2020).
22. Warehouse Management System. Sortly. URL: <https://www.sortly.com/glossary/warehouse-management-system/> (дата звернення: 20.05.2020).
23. Eclipse Foundation. Eclipse IDE: The Leading Open Platform for Professional Developers. URL: <https://help.eclipse.org/2024-03/index.jsp> (дата звернення: 20.05.2020).
24. Wikipedia. Eclipse (software). 2023. C. 1–15. URL: [https://en.wikipedia.org/wiki/Eclipse_\(software\)](https://en.wikipedia.org/wiki/Eclipse_(software)) (date of access: 10.06.2024).
25. Microsoft. Visual Studio Code Documentation. URL:

<https://code.visualstudio.com/docs> (дата звернення: 20.05.2020).

26. Microsoft. Visual Studio IDE. URL: <https://learn.microsoft.com/uk-ua/visualstudio/ide/?view=vs-2022> (дата звернення: 20.05.2020).
27. Wikipedia. Microsoft Visual Studio. 2023. С. 1–18. URL: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio (date of access: 10.06.2024).
28. Python Software Foundation. Python Language Reference. 2023. С. 1–25. URL: <https://docs.python.org/3/reference/index.html> (date of access: 10.06.2024).
29. Lutz M. Learning Python, 5th Edition. O'Reilly Media, O'Reilly Media, Inc, 2013. С. 1–1600.
30. Matthes E. Python Crash Course, 2nd Edition. No Starch Press, 2019. С. 1–544.
31. Wikipedia. Python (programming language). 2023. С. 1–20. URL: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)) (date of access: 10.06.2024).
32. Downey A. Think Python: How to Think Like a Computer Scientist, 2nd Edition. O'Reilly Media, 2015. С. 1–300.
33. Pygame Community. Pygame Documentation. 2023. С. 1–50. URL: <https://www.pygame.org/docs/> (date of access: 10.06.2024).
34. Mitchell S. PuLP: A Linear Programming Toolkit for Python. 2023. С. 1–40. URL: <https://coin-or.github.io/pulp/> (date of access: 10.06.2024).
35. Russell S., Norvig P. Artificial Intelligence: A Modern Approach, 3rd Edition. Prentice Hall, 2010. С. 92–100. URL: https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf (date of access: 10.06.2024)
36. Dorigo M., Stützle T. Ant Colony Optimization. MIT Press, 2004. С. 41–46.
37. Thrun S., Burgard W., Fox D. Probabilistic Robotics. MIT Press, 2005. С. 163–174. Felner R., Stern G. A* with Lookahead. Journal of Artificial Intelligence Research, 2014. С. 533–570. URL: <https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf> (date of access:

10.06.2024).

38. Li X., Zeng W., Zheng S., Zhang Q. An Improved Ant Colony Algorithm for Multi-objective Optimization Problems. *IEEE Access*, 2018. C. 18720–18734. URL: <https://doi.org/10.1109/ACCESS.2019.2897580> (date of access: 10.06.2024).

39. Wu, H., Gao, Y., Wang, W. et al. A hybrid ant colony algorithm based on multiple strategies for the vehicle routing problem with time windows. *Complex Intell. Syst.* 9, 2023, 2491–2508 URL: <https://doi.org/10.1007/s40747-021-00401-1> (date of access: 10.06.2024).

40. Ibrahim, M. F., Mustofa, M. I., Meilanitasari, P., & Wijaya, S. U. An Improved Ant Colony Optimization Algorithm for Vehicle Routing Problem with Time Windows. *Jurnal Teknik Industri*, 23(2), 2022, C. 105–120. URL: <https://doi.org/10.22219/JTIUMM.Vol23.No2.105-120> (date of access: 10.06.2024).

41. Liang, YC., Lee, ZH. & Chen, YS. A novel ant colony optimization approach for on-line scheduling and due date determination. *J Heuristics* 18, 2012, C. 571–591. URL: <https://doi.org/10.1007/s10732-012-9199-1> (date of access: 10.06.2024).

42. L. Zuo, L. Shu, S. Dong, C. Zhu and T. Hara, A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing, *IEE Access*, vol. 3, 2015, C. 2687-2699. URL: <https://doi.org/10.1109/ACCESS.2015.2508940> (date of access: 10.06.2024).

ДОДАТОК А
Технічне завдання

A.1 Вступ

A.1.1 Назва програми

Назва розробленої програми – Warehouse.

A.1.2 Призначення та область застосування

Warehouse призначено для розрахунку оптимальних маршрутів та автоматизації збереження лікарських засобів.

Актуальність роботи полягає у зрості кількості видів препаратів та населення, що їх потребує. Звичайна людина не має можливості запам'ятовувати та швидко розрізняти велику кількість упаковок. Тому для підвищення швидкості роботи магазинів було розроблено автоматичні руки-контролери, які контролюються нашою програмою. Розробка призначена для вирахування маршрутів даних рук на складі лікарських засобів.

У роботі проведено аналіз і дослідження методів розв'язання задач маршрутизації. Був розроблений новий метод для розрахунку маршрутів, спрямований на комбінаторну оптимізацію вибору шляхів руху по складу ліків.

Основна відмінність запропонованого методу полягає в тому, що він враховує весь контекст завдання і використовує спеціалізовані способи кодування можливих рішень, що забезпечує максимальну адаптацію до умов конкретної задачі.

Дослідження показало, що запропонований підхід забезпечує стійку збіжність процесу пошуку за прийнятну кількість кроків без штучного обмеження простору пошуку і без необхідності використання додаткової експертної інформації про доцільність можливих рішень.

A.2 Підстави для розробки

Warehouse розроблено на підставі наказу «Про створення продукту для вирішення задач оптимізації», затверджене наказом директора по розробці операційного забезпечення № 145 від 20 січня 2023 року.

A.3 Призначення розробки

Додаток призначено для автоматизації складу ліків.

A.4 Вимоги до програми

A.4.1 Вимоги до функціональних характеристик

Додаток виконує наступні функції:

- виведення карти;
- можливість змінювати інформацію про маршрути;
- додавання нових маршрутів до списку;
- розрахунок оптимальних маршрутів;
- виведення розрахованих маршрутів;
- візуалізація розрахованих маршрутів на карті;

A.4.2 Вимоги до надійності

До основних вимог до надійності програми відносяться наступні:

- перевірка даних введених у комірки;
- перевірка колізій із розташування рук;
- зрозумілий формат виведення результатів.

A.4.3 Вимоги до технічних та програмних засобів

- процесор: Intel Core i5 або інший процесор з архітектурою x86, який має частоту не менше 2.5 ГГц і підтримує SSE2 та більш сучасні розширення;
- пам'ять: мінімум 8 Гб оперативної пам'яті;
- відеокарта: підтримка DirectX 11 і WDDM версії 2.0 або новішої;
- дисковий простір: SSD з ємністю не менше 256 ГБ;
- роздільна здатність екрана: Full HD (1920 x 1080 пікселів) або вище;
- пристрій введення: Комфортна клавіатура і миша, які підтримують сучасні стандарти зв'язку.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- вступ;
- зміст;
- перелік умовних скорочень;
- аналіз предметної області;
- вибір та обґрунтування структури системи, яка проєктується;
- основні рішення щодо реалізації компонентів системи;
- керівництво програміста;
- керівництво оператора;
- тестування результатів.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програма створена для автоматизації складу ліків. Для автоматизації було обрано не область зберігання, а область доставки ліків на склад та з нього. Програма виконана на мові програмування python.

Б.2 Функціональне призначення

Додаток виконує наступні функції:

- графічне відображення сапи
- розрахунок оптимальних маршрутів;
- збереження розрахованих маршрутів;
- візуалізація розрахованих маршрутів на карті;

Б.3 Опис логічної структури програми

Розроблюваний ПП повинен обов'язково містити такі модулі:

- зберігання даних;
- розрахунковий;
- вводу-виводу даних;
- модуль зв'язку всіх елементів ПП.

Зв'язок між модулями можна побачити на рис. Б.1.

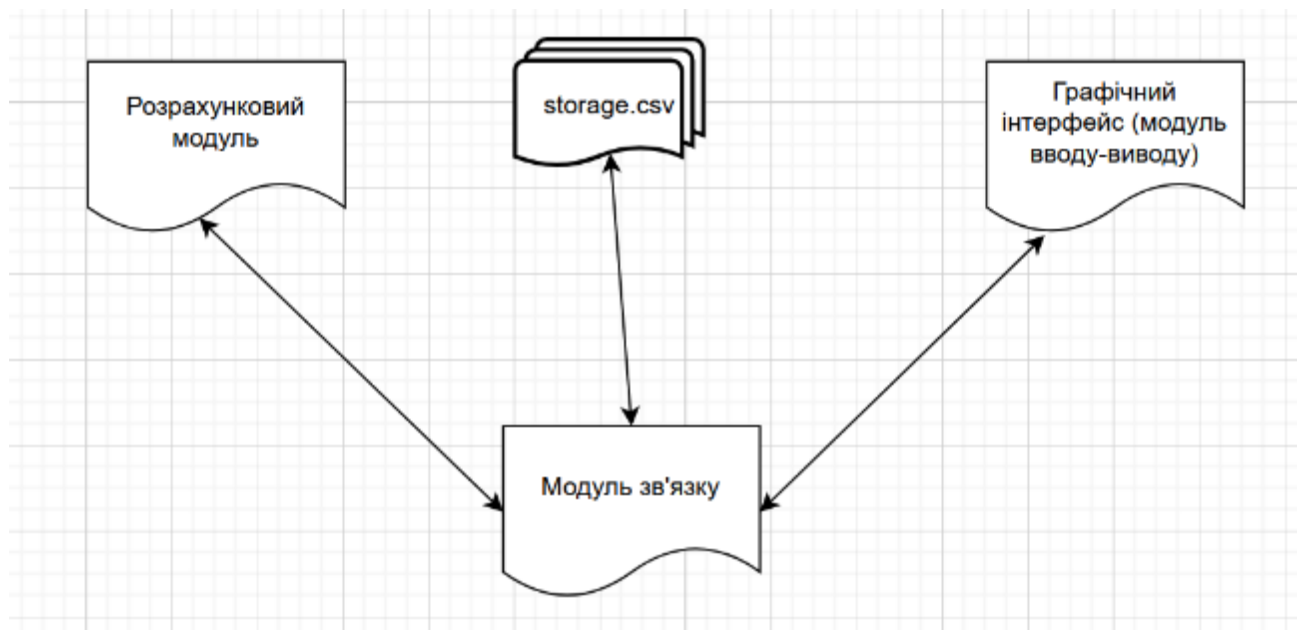


Рисунок Б.1 – Структурна схема програми

Розроблюваний програмний продукт використовує дані про простір для побудови маршруту, що включає в себе інформацію про кількість та вміст комірок. Ці дані можуть бути збережені в різних типах сховищ:

- текстові файли (CSV, txt, xml,...);
- бінарні файли даних;
- не реляційні бази даних (NoSQL сховища);
- реляційні бази даних (MySQL, MS SQL Server, Firebird, ...).

Обраний метод зберігання повинен забезпечувати достатню гнучкість форматів, високу швидкість доступу до даних, здатність для зберігання великих об'ємів інформації, можливість розширення функціоналу і при цьому не призводити до надмірного використання пам'яті. До наведених функцій найбільше підходить CSV файл із описом середовища.

Переваги використання CSV файлу:

- простота і доступність при створенні та заповненні даних;
- повна незалежність даних. Зміни в прикладній програмі при зміні реляційної БД мінімальні;
- мінімальний обсяг файлів для збереження даних.

Отже, підсумовуючи вищесказане у якості системи зберігання даних було обрано CSV файл.

Розрахунковий модуль – це основний та найважливіший модуль, його призначенням є реалізація методів пошуку маршруту для кожної руки. Даний модуль включає в себе наступні субмодулі (скрипти):

- модуль для визначення положення рук у просторі;
- модуль, що реалізує розподіл замовлень по руках;
- модуль, що реалізує перегляд доступних шляхів;
- модуль, що реалізує метод визначення оптимального шляху.

Модуль вводу-виводу необхіден для зручного корегування зв'язку користувача з програмою і включає в себе наступні субмодулі (скрипти):

- модуль для введення параметрів середовища;
- модуль для введення комірок-замовлень;
- модуль для виведення позиції рук;
- модуль для виведення шляху руки.

Модуль зв'язку всіх елементів ПП, як і видно з назви потрібен для передачі даних між модулями

Б.3.1 Алгоритм пошуку A*

Алгоритм A* робить покроковий перебір всіх можливих шляхів, що ведуть від початкової вершини у кінцеву, поки не знайде мінімальний шлях (рис. Б.2).

Алгоритм роботи методу A*.

Крок 1. Переглянути вузли, що суміжні з початковим.

Крок 2. Обрати з них той, що має мінімальне значення за критерієм.

Крок 3. Визначити пріоритет шляху за визначенням критерію.

Крок 4. Перевірити, якщо значення критерію цільової вершини не менше, ніж будь-яке зі значень в черзі, перейти до кроку 5. В іншому випадку перейти до кроку 2.

Крок 5. Завершення роботи алгоритму.

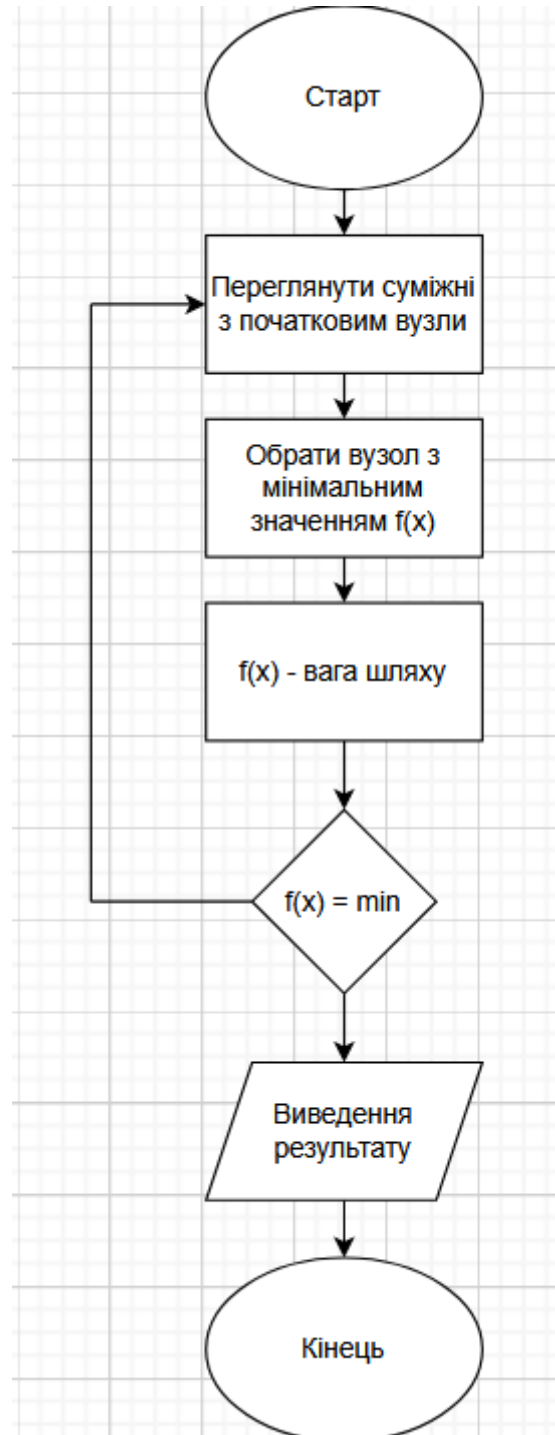


Рисунок Б.2 – Блок-схема алгоритму A*

Алгоритм пошуку A^* поєднує в собі підходи алгоритму Дейкстри та жадібного пошуку. Спочатку визначається мапа, яка зберігає інформацію про попередні вершини, їх евристичні відстані до кінцевої вершини та вартості досягнення від початкової вершини [37]. Початкова вершина додається до мапи зі значеннями 0.

Далі створюються набори "відкритих" та "закритих" вершин. Початкова вершина додається до відкритого набору. Цикл продовжується, доки відкритий набір містить хоча б одну вершину. На кожній ітерації обирається вершина з найменшою сумою евристичної відстані та відстані досягнення з відкритого набору і видаляється з нього [38]. Для кожної сусідньої вершини, яка ще не знаходиться у закритому наборі, обчислюється її вартість та додається до мапи. Вершина-сусід також додається до відкритого набору, якщо вона вже там є і сума її вартостей у мапі менша або дорівнює сумі вартостей, обчислених на поточній ітерації.

Якщо вершина-сусід є кінцевою, пошук завершується. Після кожної ітерації поточна вершина додається до закритого набору, і починається наступна ітерація [39].

Порядок обходу вершин визначається евристичною функцією «відстань+вартість» (зазвичай позначається як $f(x)$). Ця функція – сума двох інших: функції вартості досягнення даної вершини (x) з початковою (зазвичай позначається як $g(x)$) і може бути як евристичною, так і немає), і функції евристичної оцінки відстані від розглянутої вершини до кінцевої (позначається як $h(x)$), що можна побачити на формулі (Б.1):

$$f(x) = g(x) + h(x). \quad (\text{Б.1})$$

Функція $h(x)$ повинна бути допустимою евристичною оцінкою, тобто не повинна переоцінювати відстані до цільової вершини. Наприклад, для завдання маршрутизації $h(x)$ може являти собою відстань до цілі по прямій лінії, так як це фізично найменша можлива відстань між двома точками.

Б.4 Використані технічні засоби

При розробці були використані: пайтон 3.11.1. Для розгортання системи, достатньо запустити файл `main.py`.

Б.5 Виклик і завантаження

Для запуску програми, потрібно в консолі перейти до розташування кореневої директорії програми і запустити на виконання файл `main.py` командою `python main.py`. Для запуску, в вас повинна бути встановлена версія пайтон 3.

Б.6 Початкові дані

Початковими вхідними даними програми є CSV файл із наладуванням комірок та введені користувачем параметри поля і механізованих рук.

Вихідними даними ж є графічне відображення шляху руки для виконання замовлення.

ДОДАТОК В
Фрагменти тексту програми

Main.py

```
import pygame
```

```
import threading
```

```
from graphical_module import make_grid, draw, get_clicked_pos
```

```
from algorithmical_module import run_algorithms
```

```
WIDTH = 600
```

```
ROWS = 40
```

```
WIN = pygame.display.set_mode((WIDTH, WIDTH))
```

```
pygame.display.set_caption("Warehouse")
```

```
def main(win, width):
```

```
    grid = make_grid(ROWS, width)
```

```
    starts = [None] * 6
```

```
    ends = [None] * 6
```

```
    for i in range(6):
```

```
        starts[i] = grid[0][17 + i]
```

```
        starts[i].draw_robot()
```

```
    run = True
```

```
    while run:
```

```
        draw(win, grid, ROWS, width)
```

```
        for row in range(40):
```

```
            for column in range(40):
```

```
                if (row in [4, 5, 9, 10, 14, 15, 19, 20, 24, 25, 29, 30, 34, 35] and
```

```
                    column in [4, 5, 6, 7, 11, 12, 13, 14, 18, 19, 20, 21, 25, 26, 27, 28,
```

```
32, 33, 34, 35]):
```

```
                    cell = grid[row][column]
```

```
                    cell.make_barrier()
```

```

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        run = False

    if pygame.mouse.get_pressed()[0]: # left-mouse-click
        pos = pygame.mouse.get_pos()
        row, col = get_clicked_pos(pos, ROWS, width)

        for i in range(6):
            cell = grid[row][col]
            if not any(cell == start for start in starts) and not any(cell == end
for end in ends):
                if not starts[i]:
                    starts[i] = cell
                    starts[i].draw_robot()
                elif not ends[i]:
                    ends[i] = cell
                    ends[i].draw_target()

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE and all(starts) and all(ends):
                for row in grid:
                    for cell in row:
                        cell.check_neighbors(grid)

                algo_thread = threading.Thread(target=run_algorithms,
args=(win, width, grid, starts, ends))
                algo_thread.start()

        # Reset the starts and ends for next run

```

```

Algorithms.py
from queue import PriorityQueue
from collections import deque
import random
import csv

all_paths = []

def get_path_length(p1, p2):
    return sum(abs(a - b) for a, b in zip(p1, p2))

def find_path_around_obstacle(start, end, grid):
    queue = deque([(start, [start])])
    visited = set()

    while queue:
        curr, path = queue.popleft()
        visited.add(curr)

        if curr == end:
            return path

        for neighbor in curr.neighbors:
            if neighbor not in visited and not neighbor.is_obstacle():
                new_path = path + [neighbor]
                queue.append((neighbor, new_path))

    return [] # No path found

```

Graphics.py

```
import pygame
```

```
GREEN = (9, 206, 55)
```

```
WHITE = (255, 255, 255)
```

```
BLACK = (0, 0, 0)
```

```
GREY = (128, 128, 128)
```

```
ROBOT = (255, 87, 51)
```

```
class Cell:
```

```
    def __init__(self, row, col, width, total_rows):
```

```
        self.row = row
```

```
        self.col = col
```

```
        self.x = row * width
```

```
        self.y = col * width
```

```
        self.color = WHITE
```

```
        self.neighbors = []
```

```
        self.width = width
```

```
        self.total_rows = total_rows
```

```
    def get_pos(self):
```

```
        return self.row, self.col
```

```
    def is_obstacle(self):
```

```
        return self.color == BLACK or self.color == ROBOT
```

```
    def draw_robot(self):
```

```
        self.color = ROBOT
```

```
    def make_barrier(self):
```

ДОДАТОК Г
Слайди презентації

Розробка програмного застосунку «Автоматизований склад ліків»

ВИКОНАВ: ПЕТУНІН ОЛЕКСАНДР

СТ. ГРУПИ КНТ 111-СП

КЕРІВНИК: ОЛІЙНИК АНДРІЙ

Д.Т.Н., ПРОФЕСОР НУ "ЗАПОРІЗЬКА ПОЛІТЕХНІКА "

Рисунок Г.1 – Слайд 1

Автоматизація складів

2



Види систем автоматизації складів:

- Роботизовані системи підбору та розміщення
- Електронні системи відстеження та управління запасами
- Автоматизовані системи пакування та транспортування
- Системи керування складами

Рисунок Г.2 – Слайд 2

Об'єкт, предмет та мета роботи 3

- ▶ Об'єкт дослідження – процес організації автоматизованого складу лікарських засобів.
- ▶ Предмет дослідження – робота лікарських складів.
- ▶ Методи дослідження – модифіковані методи дослідження операцій, зокрема жадібні алгоритми.

Рисунок Г.3 – Слайд 3

Мета і задачі проєкту 4

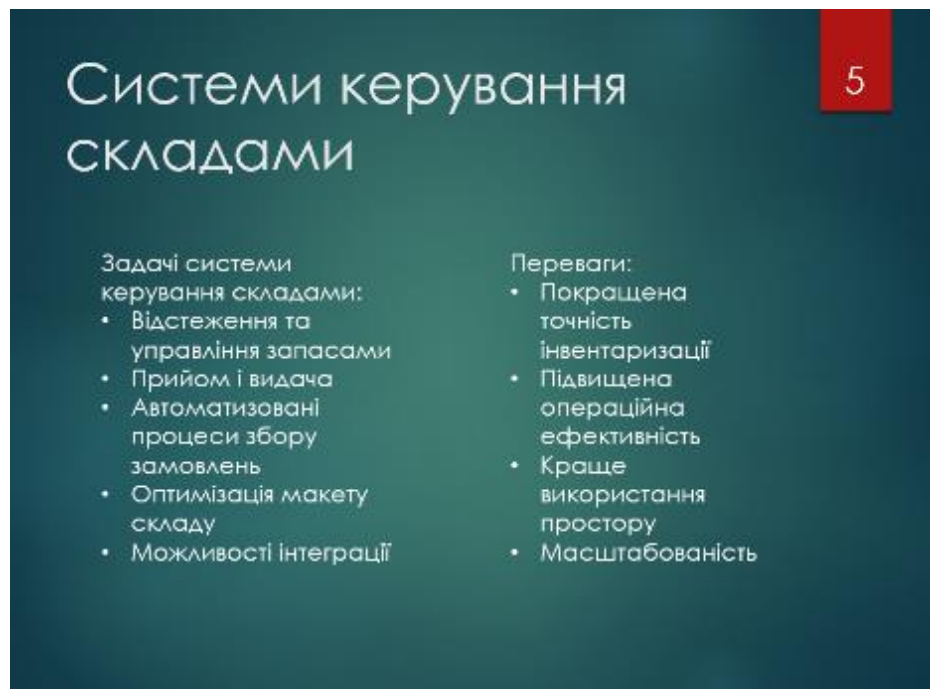
Мета роботи:
Розробити застосунок для керування складом

Задачі проєкту:

- Забезпечити можливість введення даних для побудови маршруту
- Розробити алгоритм вирішення задачі маршрутизації;
- Вивести результатів у графічному форматі;
- Виведення попереджень при помилках



Рисунок Г.4 – Слайд 4



Системи керування складами

5

Задачі системи керування складами:

- Відстеження та управління запасами
- Прийом і видача
- Автоматизовані процеси збору замовлень
- Оптимізація макету складу
- Можливості інтеграції

Переваги:

- Покращена точність інвентаризації
- Підвищена операційна ефективність
- Краще використання простору
- Масштабованість

Рисунок Г.5 – Слайд 5



Огляд особливостей середовища розробки

6

- ▶ Visual Studio Code — це редактор вихідного коду, який можна використовувати з різними мовами програмування, включаючи C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust.
- ▶ Він базується на структурі Electron, яка використовується для розробки вебдодатків Node.js, які працюють на механізмі компонування Blink. Visual Studio Code використовує той самий компонент редактора, який використовується в Azure DevOps (раніше називався Visual Studio Online і Visual Studio Team Services).

Активация Win

Рисунок Г.6 – Слайд 6

Огляд особливостей мови програмування

7

Python — це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою.

Простий, легкий для вивчення синтаксис Python підкреслює читабельність і знижує вартість обслуговування програми. Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду.

Рисунок Г.7 – Слайд 7

Функціональна схема програми

8

Дані завантажуються з і зберігаються у csv файлі обраному користувачем

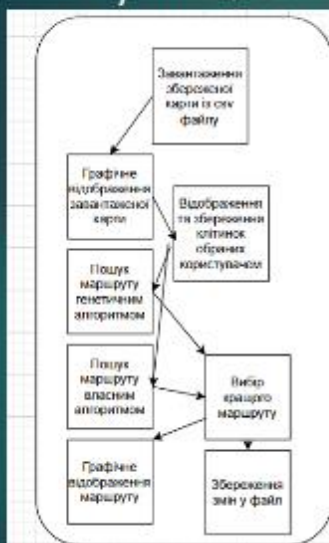


Рисунок Г.8 – Слайд 8

Генетичний алгоритм

9

Генетичний алгоритм (GA) – це метод оптимізації, натхненний процесами природного відбору і генетики. Він починається зі створення початкової популяції рішень, які оцінюються за фітнес-функцією. Найкращі рішення відбираються, схрещуються і мутують для створення нових рішень. Ці кроки повторюються, поки не буде досягнуто оптимального або наближеного до оптимального рішення.

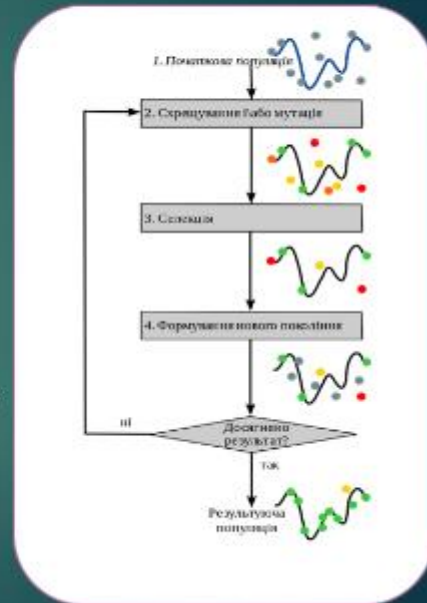
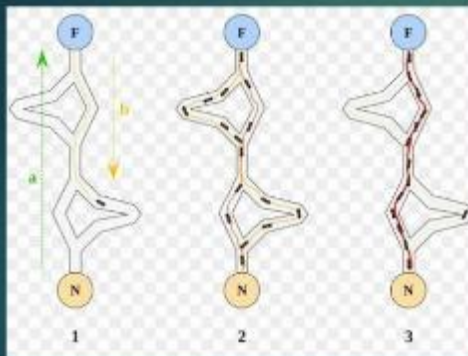


Рисунок Г.9 – Слайд 9

Алгоритм мурашиної колонії

10



Мурашиний алгоритм (ACO) – це метод оптимізації, який імітує поведінку мурах при пошуку їжі. Мурахи будують шляхи, керуючись інтенсивністю феромонів і евристичною інформацією. Шляхи з кращими характеристиками отримують більше феромонів, що підвищує ймовірність їх вибору в наступних ітераціях. Процес повторюється, поки не буде знайдено оптимальне або наближене до оптимального рішення.

Рисунок Г.10 – Слайд 10

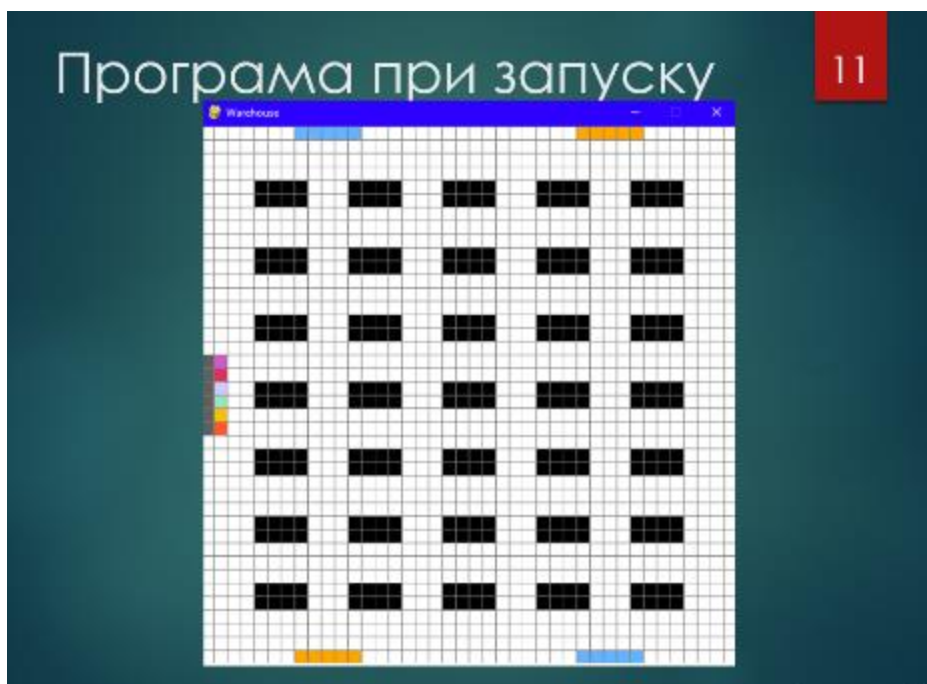


Рисунок Г.11 – Слайд 11



Рисунок Г.12 – Слайд 12

Рух по маршрутах

13

При натисканні пробілу програма вираховує найоптимальніший маршрут і запускає рух рук, зображених кольоровими квадратами. Вони рухаються обходячи перешкоди і одне одного до призначених вантажів. Після цього у обраних точках видається інформація про товари і руки повертаються на місце, а оновлена мапа зберігається у файл.

Рисунок Г.13 – Слайд 13

14

Дякую за увагу

Рисунок Г.14 – Слайд 14