

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 ПРОГРАМНА РЕАЛІЗАЦІЯ ОБОЛОНКИ ЕКСПЕРТНОЇ СИСТЕМИ
НА ОСНОВІ ПРОДУКЦІЙНОЇ МОДЕЛІ
SOFTWARE IMPLEMENTATION OF EXPERT SYSTEM SHELL BASED ON
PRODUCTION MODEL

Виконав(ла): студент(ка) 4 курсу, групи КНТ-211

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

КАТЕЛЕВСЬКИЙ Д.Є

(ПРИЗВИЩЕ та ініціали)

Керівник СУББОТІН С.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

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

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КАТЕЛЕВСЬКОГО Дмитра Євгеновича
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація оболонки експертної системи на основі продукційної моделі. Software Implementation of Expert System Shell Based on Production Model

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

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз задачі розробки експертної системи. 2. Технічні засоби та методи розробки. 3. Програмна реалізація оболонки експертної системи. 4. Практичне застосування та перспективи розвитку.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., зав. каф.		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 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 розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Дмитро КАТЕЛЕВСЬКИЙ
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Сергій СУББОТІН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 62 с., 3 табл., 36 рис., 2 дод., 14 джерел.

ЕКСПЕРТНА СИСТЕМА, КОМП'ЮТЕРНА ПРОГРАМА, ПРОДУКЦІЙНА МОДЕЛЬ, ГРАФІЧНИЙ ІНТЕРФЕЙС, АВТОМАТИЗАЦІЯ ПРИЙНЯТТЯ РІШЕНЬ.

Об'єкт дослідження – програмні засоби для створення експертних систем.

Предмет дослідження – оболонки експертних систем на основі продукційної моделі знань.

Мета роботи – розробка програмної оболонки експертної системи з графічним інтерфейсом для автоматизації процесу прийняття рішень на основі продукційних правил.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, продукційна модель знань, алгоритм прямого та зворотнього виведення, мова програмування Python 3.9.7, бібліотека Tkinter для графічного інтерфейсу, засоби збереження даних у форматі JSON, логування для аналізу роботи системи, ноутбук із процесором Intel Core i5, операційна система Microsoft Windows 11.

Результати. Створенню програмну оболонку експертної системи на основі продукційної моделі з графічним інтерфейсом, що підтримує прямий і зворотний вивід. Система дозволяє користувачам створювати та обробляти правила для автоматизації прийняття рішень.

Висновки. Розроблено універсальну оболонку експертної системи на основі продукційної моделі, що автоматизує процес виконання рішень та є гнучкою для використання в різних предметних областях.

Галузь використання – це автоматизація рішень у технічних, медичних, економічних сферах, та використання в навчальних програмах.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 62 pages, 3 tables, 36 figures, 2 appendixes, 14 sources.

EXPERT SYSTEM, COMPUTER PROGRAM, PRODUCT MODEL, GRAPHICAL INTERFACE, DECISION-MAKING AUTOMATION.

Object of research - software tools for creating expert systems.

Subject of research - expert system shells based on the productive model of knowledge.

Purpose - to develop an expert system software shell with a graphical interface to automate the decision-making process based on productive rules.

Materials, methods and techniques: object-oriented programming, productive knowledge model, forward and backward inference algorithm, Python 3.9.7 programming language, Tkinter library for the graphical interface, means of saving data in JSON format, logging for system analysis, laptop with Intel Core i5 processor, Microsoft Windows 11 operating system.

Results. The software shell of the expert system based on the product model with a graphical interface that supports forward and reverse output was created. The system allows users to create and process rules for automating decision-making.

Conclusions. A universal shell of an expert system based on the product model has been developed that automates the process of decision making and is flexible for use in various subject areas.

The field of application is the automation of decisions in technical, medical, economic fields, and use in educational programmes.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	7
Вступ	8
1 Аналіз задачі розробки експертної системи	9
1.1 Сутність та значення експертних систем	9
1.2 Огляд продукційних моделей знань	11
1.3 Структура та застосування продукційних правил	13
1.4 Постановка цілей і завдань дослідження	15
1.5 Висновок до розділу 1	16
2 Технічні засоби та методи розробки	17
2.1 Вибір і обґрунтування мови програмування	17
2.2 Вибір і обґрунтування бібліотеки Tkinter для створення графічного інтерфейсу (GUI) користувача	17
2.3 Вибір і обґрунтування інтеграції бази знань із JSON-файлом	18
2.4 Алгоритми прямого та зворотного виведення	19
2.5 Висновок до розділу 2	24
3 Програмна реалізація оболонки експертної системи	26
3.1 Архітектура системи та її компоненти	26
3.2 Опис функціональності програми	26
3.3 Структура графічного інтерфейсу та взаємодія з користувачем	34
3.4 Висновок до розділу 3	38
4 Практичне застосування та перспективи розвитку	40
4.1 Застосування системи в різних предметних областях	40
4.2 Обмеження розробленої оболонки та шляхи вдосконалення системи ..	40
Висновки	42
Перелік джерел посилання	43
Додаток А Текст програми	45
Додаток Б Слайди презентації	57

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- CSV – Comma-Separated Values;
- GUI – Graphical User Interface;
- JSON – JavaScript Object Notation;
- SQL – Structured Query Language;
- ООП – об'єктно-орієнтоване програмування;
- ШІ – штучний інтелект.

ВСТУП

Експертні системи – це програмний засіб, що використовує експертні знання у певній предметній області з метою ефективного вирішення задач у предметній області, яка цікавить користувача, на рівні середнього професіонала (експерта). Вони допомагають автоматизувати процес прийняття рішень у дуже різних галузях, завдяки імітації мислення експерта. Більшість подібних систем, для вирішення задач, використовує продукційні моделі знань, що використовують таке правило, як – «ЯКЩО – ТО» [1].

В сучасному світі, коли обсяги даних невпинно зростають, потреба в подібних автоматизованих експертних системах стає все дедалі більшою. Мета цієї роботи – це створення програмної оболонки експертної системи на основі продукційної моделі з графічним інтерфейсом. Для виконання завдань було використано мову програмування Python та бібліотеку Tkinter, яка дозволяє створювати зручний до використання графічний інтерфейс. Створена продукційна модель підтримує логічні оператори та два типи виведення – прямий і зворотний.

Розробка подібної системи дозволить не лише автоматизувати певні складні процеси, а й зробити їх більш доступними для користувачів, що не володіють глибокими знаннями у програмуванні.

1 АНАЛІЗ ЗАДАЧІ РОЗРОБКИ ЕКСПЕРТНОЇ СИСТЕМИ

1.1 Сутність та значення експертних систем

Сучасні експертні системи представляють собою спеціалізовані програмні комплекси, що входять до складу штучного інтелекту. Вони призначені для імітації мисленнєвої діяльності фахівців-експертів у вузькоспеціалізованих галузях. Робота таких систем базується на використанні структурованих знань, представлених у формі правил і фактів, що дозволяє їм вирішувати складні професійні завдання. Наприклад, медична експертна система може аналізувати симптоми пацієнта та пропонувати можливі діагнози або методи лікування. Аналогічно, фінансова експертна система може оцінювати ринкові тенденції та рекомендувати інвестиційні стратегії [2]-[3].

Ключові переваги експертних систем [2]-[3]:

- збереження експертного досвіду - цифровізація та систематизація знань кваліфікованих фахівців [2]-[3];
- зб'єктивність рішень - мінімізація суб'єктивного фактору при аналізі ситуацій [2]-[3];
- ефективність використання ресурсів - зменшення витрат часу та коштів на вирішення типових завдань [2]-[3];
- масштабованість - можливість одночасної роботи з багатьма користувачами [2]-[3].

Як і будь-яка технологія, експертні системи мають свої переваги та недоліки.

Переваги експертних систем [2]-[3]:

- узгодженість – експертні системи надають узгоджені та надійні рекомендації, зменшуючи мінливість, яка може виникати під час прийняття рішень людиною [2]-[3];
- доступність – вони доступні цілодобово та можуть обробляти кілька запитів одночасно, надаючи своєчасну допомогу та підтримку [2]-[3];

- економічна ефективність – автоматизуючи прийняття рішень на експертному рівні, організації можуть заощадити на витратах, пов'язаних з наймом та навчанням експертів-людей [2]-[3];

- збереження знань – експертні системи зберігають цінні знання та досвід, роблячи їх доступними, навіть якщо початкові експерти більше не доступні [2]-[3].

Обмеження експертних систем [2]-[3]:

- обмеження знань – ефективність експертної системи залежить від повноти та точності бази знань. Якщо знання застарілі або неповні, продуктивність системи може бути поставлена під загрозу [2]-[3];

- відсутність гнучкості – експертні системи обмежені правилами та знаннями, на основі яких вони запрограмовані. Вони можуть мати труднощі з новими або неоднозначними ситуаціями, які виходять за рамки їхніх заздалегідь визначених правил [2]-[3];

- технічне обслуговування – для підтримки актуальності та актуальності бази знань необхідні регулярні оновлення та технічне обслуговування, що може бути ресурсомістким [2]-[3].

Яскравим прикладом успішної реалізації є медична система MYCIN, розроблена у Стенфордському університеті. MYCIN була написана на мові програмування Лісп як докторська дисертація Едварда Шортлайфа (англ. Edward Shortliffe) під керівництвом Брюса Б'юкенана (англ. Bruce Buchanan), Стенлі Н. Коена (англ. Stanley N. Cohen) та інших [4].

Призначення цієї програми — бути асистентом лікаря, який не є вузьким спеціалістом в області застосування антибіотиків. В процесі роботи програма формує гіпотези діагнозу і надає їм певні ваги, але самостійно, як правило, не робить остаточного вибору [4].

MYCIN використовувала базу знань із приблизно 600-та правил і просту машину виведення. Під час роботи програма задавала користувачу (лікарю) серію запитань типу «так/ні» або текстових відповідей. У результаті система формувала перелік підозрюваних бактерій, відсортованих за

ймовірністю, із зазначенням довірчого інтервалу для діагнозів, їх обґрунтуванням, а також пропонувала курс лікування. Вона також пояснювала, які запитання та правила привели до такого ранжування [4].

Проте система ніколи не застосовувалася на практиці. Основною причиною стала не її ефективність, адже в тестах вона перевершувала навіть професорів Стенфорду, а технологічні обмеження того часу. MYCIN працювала як автономна система, що вимагала від користувача введення всіх даних вручну. Вона запускалася на сервері з поділом часу через ранній Інтернет, коли персональні комп'ютери ще не були поширені. У 1970-х роках сеанс роботи з MYCIN міг тривати 30 хвилин і більше, що було неприйнятним для зайнятого лікаря [4].

Крім того, виникали етичні та правові питання: хто несе відповідальність у разі неправильного прогнозу чи призначення лікування? У сучасних умовах подібна система могла б інтегруватися з медичними базами даних, автоматично отримуючи інформацію про пацієнтів, що значно зменшило б залежність від ручного введення даних. Хоча MYCIN не використовувалася в реальних умовах, вона стала важливим прикладом того, як експертні системи можуть допомагати лікарям у діагностиці [2]-[4].

1.2 Огляд продукційних моделей знань

Продукційні моделі – це найпопулярніший підхід для представлення знань в експертних системах. Вони базуються на правилах виду «ЯКЩО – ТО», де умова (антецедент) є речення-зразок, а дія (консеквент) – є дією, що виконується при успішному результаті [5].

Такі моделі включають кілька ключових компонентів, а саме [6]:

- база знань – централізоване сховище правил і фактів [6];
- механізм логічного виводу – алгоритм обробки правил [6];
- робоча пам'ять – іноді її називають списком фактів. Вона містить динамічну інформацію, яка змінюється під час роботи системи [6];

– механізм керування – визначає порядок, у якому правила застосовуються механізмом логічного висновку, та керує потоком процесу [6].

Продукційні моделі відрізняються високою гнучкістю, що дозволяє легко адаптувати їх до різних предметних областей. Вони підтримують два основних методи міркувань [6]:

- прямий ланцюжок;
- зворотний ланцюжок.

Експертні системи, що використовують продукційну модель, широко застосовуються в різних галузях, наприклад: медицина, технічна діагностика та економічний аналіз, тощо. Продукційні правила можуть включати логічні оператори, а саме: AND, OR, NOT. Завдяки цьому можна описати більш складні залежності між фактами, та тим самим, розширити можливості моделі щодо представлення знань.

Продукційна модель найчастіше використовується в промислових експертних системах. Наприклад, у медичній експертній системі правила «ЯКЩО, ТО» можуть використовуватися для встановлення взаємозв'язків між симптомами і діагнозами. Під час виведення реальний симптом зіставляється з тим, які є в лівих частинах правил і в разі збігу права частина відповідного правила вважається можливим діагнозом. Якщо є інші правила, що містять у лівих частинах отриманий можливий діагноз, то він розглядається як проміжний симптом. У цьому випадку здійснюється подальше виведення, яке триває доти, доки не буде отримано результат, з якого вже нічого не можна вивести. Якщо більше немає правил, на основі яких можна зробити виведення з отриманого можливого діагнозу, то він розглядається як «остаточний». На будь-якому кроці такого виведення може виявитися кілька застосовних правил і тоді породжується дерево виведення, що визначає множину діагнозів [5]-[6].

Основною перевагою продукційної моделі є її гнучкість до змін. Це дозволяє легко додавати, видаляти або змінювати правила без будь-якої

необхідності перебудови всієї системи. Завдяки цьому, продукційні моделі є ідеальними для створення універсальних експертних систем, які, в свою чергу, можуть адаптуватися до різних предметних областей. Також, подібні моделі можуть підтримувати два різних методи виведення: прямий, що генерує нові факти на основі існуючих, та зворотний – перевіряє досяжність заданої мети [7].

Незважаючи на переваги, продукційні моделі мають ряд обмежень: складність управління великими базами знань і відсутність вбудованої підтримки нечітких знань. Проте навіть так, ці моделі показують себе на високому рівні ефективності.

1.3 Структура та застосування продукційних правил

Продукційні правила є формальним способом представлення знань у вигляді рекомендацій, вказівок чи стратегій. Вони особливо корисні, коли знання предметної області формуються на основі емпіричного досвіду, накопиченого протягом багатьох років роботи в певній галузі [7].

Продукцією (продукційним правилом) називають вираз виду: $(i); Q; P; A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_k; N$. Відповідно [1]:

- i – ідентифікатор правила (назва чи номер) [1];
- Q – сфера застосування правила [1];
- $A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_k$ – ядро продукції, де « $\rightarrow$ » – знак секвенції, A_i – i передумови (умови), B_j – j -ий висновок (дії) [1];
- P – умова застосовності ядра правила [1];
- N – постумови продукції [1].

Правило інтерпретується так: якщо передумови A_1 та A_2 та ... та A_n є вірними, то виконати дії B_1 та B_2 та ... та B_k [1].

Розглянемо приклад правила з бази знань системи MYCIN, призначеної для діагностики інфекційних захворювань [8]-[9].

Якщо:

- місце виділення культури – кров, і [8]-[9];
- реакція мікроорганізму – грамвід’ємна, і [8]-[9];
- форма мікроорганізму – паличка, і [8]-[9];
- пацієнт відноситься до групи ризику, то [8]-[9];
- із упевненістю (0,6) назва мікроорганізму – *pseudomonias aeruginosa* [8]-[9].

Умови правила складаються з чотирьох фактів, об’єднаних оператором «і». Якщо всі умови виконуються, система робить відповідний висновок. Коефіцієнт упевненості (у цьому випадку 0,6) вказує на ймовірність правильності висновку, що дозволяє системі працювати з неточними даними [8]-[9].

У базі знань системи MYCIN факти представляються за допомогою триплету: об’єкт-атрибут-значення. Ось як це виглядає для наведеного правила (табл. 1.1) [8]-[9].

Таблиця 1.1 – Представлення фактів [8]-[9]

№ факту	Об’єкт	Атрибут	Значення
1	Культура	Місце	Кров
2	Мікроорганізм	Реакція	-
3	Мікроорганізм	Форма	Паличка
4	Пацієнт	Належить до групи ризику	Істина
5	Мікроорганізм	Назва	<i>pseudomonias aeruginosa</i>

Таке представлення дозволяє системі чітко структурувати знання та ефективно їх обробляти. Продукційні правила, подібні до наведеного, є основою для логічного виведення в експертних системах, забезпечуючи можливість формалізації знань і прийняття рішень на їх основі. Вони

дозволяють не лише визначати ймовірні діагнози, а й обґрунтовувати їх, що робить такі системи цінними для автоматизації складних процесів

1.4 Постановка цілей і завдань дослідження

Метою даної дипломної роботи є розробка програмної оболонки експертної системи на основі продукційної моделі знань із графічним інтерфейсом користувача. Вона забезпечить автоматизацію прийняття рішень у різних предметних областях. Система має бути універсальною, зручною для користувачів, що не мають глибоких знань у програмуванні та підтримувати збереження даних і логування для аналізу роботи.

Для досягнення цієї мети визначено наступні завдання: аналіз вимог до оболонок експертних систем; вибір мови програмування та інструментів (Python 3.9, Tkinter); розробка алгоритмів прямого та зворотного виведення з підтримкою логічних операторів; створення графічного інтерфейсу для введення правил і фактів; забезпечення збереження бази знань у JSON-форматі та логування операцій; тестування системи на прикладах із різних предметних областей; оцінка ефективності та надання інструкцій для користувачів. Ці завдання спрямовані на створення гнучкої та адаптивної системи, здатної вирішувати широкий спектр задач.

1.5 Висновок до розділу 1

У розділі 1 було проведено аналіз задачі розробки експертної системи, розглянуто її основні аспекти та ключові компоненти. Експертні системи є потужним інструментом штучного інтелекту, здатним імітувати експертні знання та приймати обґрунтовані рішення в різних галузях, таких як медицина, фінанси чи технічна діагностика. Аналіз системи MYCIN показав, як такі системи можуть перевершувати людських експертів у певних задачах,

хоча їхнє практичне застосування часто обмежується технологічними та етичними факторами.

Особливу увагу було приділено продукційним моделям знань, які є основою для багатьох експертних систем завдяки своїй гнучкості та простоті реалізації. Продукційні правила типу «ЯКЩО – ТО» дозволяють формалізувати знання й ефективно їх застосовувати для вирішення задач, що було продемонстровано на прикладі правил із системи MYCIN. Водночас експертні системи мають обмеження, такі як залежність від якості бази знань і складність адаптації до нових умов, що необхідно враховувати під час їх розробки. Проведений аналіз дає основу для подальшого дослідження та створення ефективної експертної системи, яка відповідатиме сучасним вимогам.

2 ТЕХНІЧНІ ЗАСОБИ ТА МЕТОДИ РОЗРОБКИ

2.1 Вибір і обґрунтування мови програмування

Для технічної реалізації оболонки експертної системи було обрано мову програмування Python версії 3.9. Завдяки її простоті, широкій підтримці бібліотек і кросплатформності, вона забезпечує швидку розробку та легке розгортання програми.

Python є особливо ефективним для розробки експертних систем завдяки своїй гнучкості та потужним інструментам для роботи з даними. Мова має простий та інтуїтивно зрозумілий синтаксис, що значно прискорює процес кодування та полегшує супровід програмного забезпечення. Крім того, велика спільнота розробників забезпечує постійне вдосконалення мови та створення нових бібліотек для різних завдань.

Важливою перевагою Python є його кросплатформність, що дозволяє запускати програму на різних операційних системах без необхідності внесення змін у вихідний код. Це особливо актуально для експертних систем, які можуть використовуватися в різних середовищах. Також Python підтримує роботу з різними форматами даних, що є ключовим для систем, що базуються на знаннях.

Ще однією важливою характеристикою є наявність потужних інструментів для наукових обчислень та обробки даних, таких як NumPy, Pandas та SciPy. Ці бібліотеки можуть бути корисними для реалізації складних алгоритмів обробки знань та прийняття рішень. Крім того, Python має відмінні інструменти для роботи з машинним навчанням, що може знадобитися для подальшого розширення функціоналу системи.

2.2 Вибір і обґрунтування бібліотеки Tkinter для створення графічного інтерфейсу (GUI) користувача

Бібліотека Tkinter була обрана для створення графічного інтерфейсу користувача через її стандартну інтеграцію з Python. Це усуває потребу в

додаткових залежностях, а також з її допомогою можна створити інтуїтивно зрозумілий інтерфейс.

Вона надає всі необхідні інструменти для створення сучасного інтерфейсу, включаючи кнопки, поля введення, списки та інші елементи управління. Це дозволяє швидко реалізовувати функціональність без необхідності вивчення складних фреймворків.

Важливою перевагою Tkinter є її легкість у використанні та низький поріг входження для початківців. Бібліотека має чітку документацію та численні приклади використання, що значно прискорює процес розробки. Крім того, Tkinter дозволяє створювати крос-платформні додатки, які однаково добре працюють на Windows, macOS та Linux.

Незважаючи на свою простоту, Tkinter дозволяє створювати досить складні інтерфейси завдяки можливості кастомізації та розширення. Для більш складних завдань можна використовувати додаткові бібліотеки, такі як `ttk`, яка надає сучасніші візуальні стилі. Також Tkinter підтримує роботу з `Canvas` для створення графічних елементів, що може бути корисним для візуалізації складних структур даних.

2.3 Вибір і обґрунтування інтеграції бази знань із JSON-файлом

Додатково було використано стандартні модулі Python, такі як: `JSON` для збереження бази знань і `Logging` для запису операцій у файл. Ці інструменти були обрані через їхню доступність, наявність великої кількості гарної документації та відповідність вимогам щодо універсальності оболонки. Їх використання дозволяє користувачам без глибоких знань в програмуванні ефективно працювати з системою.

Формат `JSON` був обраний для зберігання бази знань через свою простоту, читабельність та універсальність. Він дозволяє легко представляти складні структури даних у вигляді, зручному для читання як людиною, так і комп'ютером. `JSON` підтримує всі основні типи даних, що необхідні для

представлення знань у експертній системі, включаючи списки, словники та вкладені структури.

Важливою перевагою JSON є його широка підтримка в різних мовах програмування та платформах. Це дозволяє легко інтегрувати базу знань з іншими системами або мігрувати дані між різними середовищами. Крім того, JSON файли можуть бути легко відредаговані вручну за допомогою будь-якого текстового редактора, що спрощує процес оновлення бази знань.

Модуль Logging був обраний для реєстрації подій у системі через свою гнучкість та потужність. Він дозволяє налаштовувати рівні логування, форматовувати вихідні повідомлення та направляти логи у різні джерела (файли, консоль, мережеві ресурси). Це особливо важливо для експертних систем, де потрібно відстежувати процес прийняття рішень та діагностувати можливі проблеми. Logging також забезпечує стабільність роботи системи, запобігаючи втраті важливої інформації при непередбачених помилках.

2.4 Алгоритми прямого та зворотного виведення

Прямий вивід – це коли система використовує правила для генерації нових фактів. Процес прямого мислення починається з набору вже відомих фактів та перевіркою умови правил у базі знань. Далі, якщо умова істинна, то його дія додасть новий факт до робочої пам'яті, і цикл буде повторюватися доти, доки не закінчатся усі правила. Наприклад, якщо факти «А» і «В» відомі, то в такому разі система повинна вивести «С». І якщо стає відоме «С», то далі вона виведе «D» і т.д [5]-[7].

Прямий ланцюжок виведення передбачає послідовне застосування продукційних правил для генерації нових фактів на основі вхідних даних. Алгоритм роботи полягає в тому, що система спочатку стартує з набору початкових фактів, завантажених у робочу пам'ять, далі механізм виведення послідовно перевіряє умови всіх правил з бази знань. При виконанні умов правила його висновок додається до робочої пам'яті як новий факт, ну і

процес повторюється ітеративно доти, доки не будуть вичерпані всі можливі правила [8]-[9].

Ключові ж особливості прямого виведення:

- використовує стратегію «від даних до висновків» [8]-[9];
- застосовується при наявності чітко визначених вхідних даних [8]-[9];
- особливо ефективний для задач класифікації та діагностики [8]-[9].

Розглянемо приклад для кращого розуміння. На рисунку 2.1 зображено множину правил-продукцій, які використовуються системою. Під час роботи машини виведення застосовується стратегія розв’язання конфліктів, за якою активується останнє правило, що успішно співставлене. Правила співставляються відповідно до їхнього порядку нумерації [8]-[9].

Маємо наступні початкові данні: $START = \{L, M, N\}$ [8]-[9].

1) $G \wedge H \rightarrow C$
6) $C \rightarrow A$
10) $A \rightarrow goal$
2) $I \wedge K \rightarrow D$
7) $D \rightarrow A$
11) $B \rightarrow goal$
3) $L \wedge M \rightarrow E$
8) $E \rightarrow B$
4) $N \rightarrow F$
9) $F \rightarrow B$
5) $O \rightarrow F$

Рисунок 2.1 – Множина правил-продукції [8]-[9]

Процес виведення відбувається у кілька етапів (табл. 2.1) [8]-[9].

Таблиця 2.1 – Послідовність прямого виведення [8]-[9]

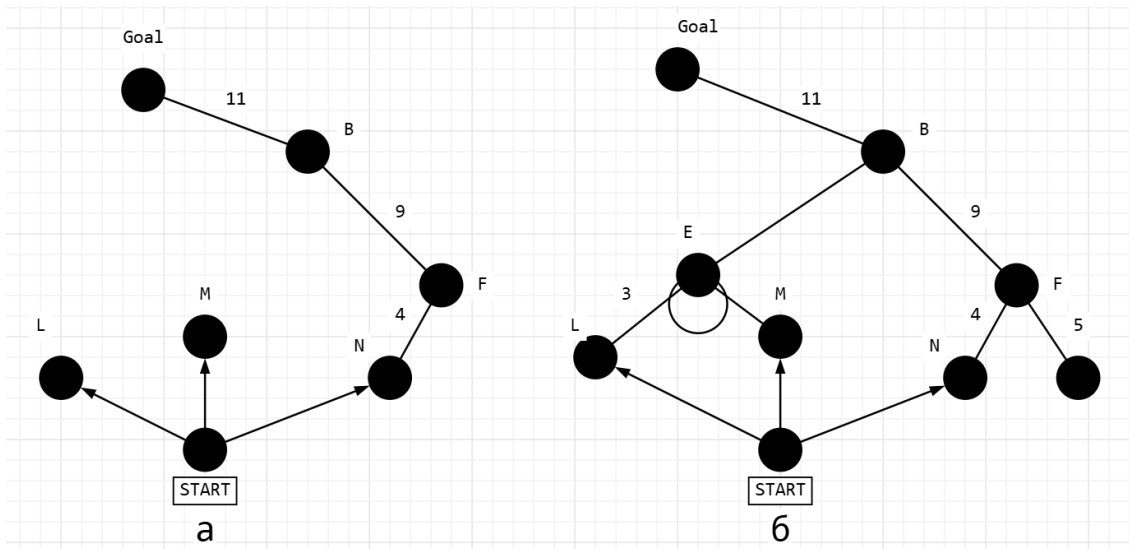
№ кроку	Робоча пам'ять	Конфліктна множина	Активізоване правило
0	L, M, N	—	—
1	L, M, N	3.4	4
2	L, M, N, F	3.9	9
3	L, M, N, F, B	3.11	11
4	L, M, N, F, B, Goal	3	Зупинка

Система може використовувати різні стратегії вибору правил з конфліктної множини, наприклад такі як пошук в глибину та пошук в ширину.

Як приклад, можна зобразити пошук в глибину у вигляді графу. Вершини відповідають висловам, а ребра — правилам виведення. Процес починається з розміщення початкових фактів L, M і N у робочій пам'яті й завершується, коли до неї додається цільова вершина Goal. Напрямок виведення, вказаний на графі, відображає рух від початкових даних до кінцевої мети (рис. 2.2, а) [8]-[9].

Отже, використана стратегія активізації останнього правила з конфліктної множини відповідає методу пошуку в глибину. Цей підхід є типовим для продукційних експертних систем, оскільки він імітує логіку мислення експерта під час розв'язання задачі. Для користувача поведінка системи виглядає природною: він спостерігає, як система послідовно розбиває головну мету на підцілі, рухаючись до кінцевого результату [8]-[9].

Якщо ж на кожному етапі роботи машини виведення активувати перше правило з конфліктної множини, граф рішення будуватиметься за методом пошуку в ширину (рис. 2.2, б). У цьому випадку всі альтернативні ланцюжки міркування, що розташовані на графі рішень, продовжуються паралельно, і жодна не обганяє іншу [8]-[9].



а – пошук в глибину; б – пошук в ширину.

Рисунок 2.2 – Пряме виведення [8]-[9]

Зазвичай подібного роду метод прямого виведення використовується у системах діагностики [5]-[7].

Зворотній вивід – це процес мислення, що розпочинається з гіпотези та поступової перевірки того, чи можна вивести результат з вже наявних фактів та правил. Наприклад, щоб встановити чи потрібно носити людині сьогодні пальто, система буде шукати правило де це є висновком: «ЯКЩО йде дощ ТА низька температура ТО потрібно вдягнути пальто». Потім система перевіряє та намагається встановити чи взагалі йде дощ та чи низька наразі температура. І якщо всі проміжні умови підтверджуються, то в такому випадку гіпотеза вважається істинною [5]-[7].

Зворотний вивід починається з цільового твердження, що фіксується в робочій пам'яті. Після цього система знаходить правило-продукцію, висновок якого збігається з поставленою метою. Умови цього правила додаються до робочої пам'яті як нові підцілі, і процес повторюється, доки не будуть знайдені факти, що підтверджують або спростовують цільове твердження. Для ілюстрації цього процесу скористаємося множиною продукцій із попереднього прикладу [8]-[9].

Процес виводу починається з того, що в робочу пам'ять розміщують цільове твердження Goal, істинність якого необхідно підтвердити чи спростувати, а також множину вихідних фактів, $\{L, M, N\}$, що вважаються істинними твердженнями. Змінні в правилах розглядаються як об'єкти з трьома можливими станами: «не визначено», «неправда» або «істина». Початково змінна Goal має стан «не визначено», тоді як факти L, M і N мають значення «істина» [8]-[9].

Зворотний ланцюжок міркувань можна переглянути у таблиці 2.2, де факти, що мають значення “істина”, виділені курсивом. Виведення починається з цільового твердження Goal, яке виступає як поточна підціль. Усі правила, висновки яких відповідають цій підцільі, додаються до конфліктної множини. На кожному етапі активується перше правило з цієї множини, а його умови додаються до робочої пам'яті, стаючи новими підцільями для наступного кроку. Такий підхід до вирішення конфліктів відповідає стратегії пошуку в ширину (рис. 2.3) [8]-[9].

Таблиця 2.2 – Послідовність зворотного виводу [8]-[9]

№ кроку	Робоча пам'ять	Конфліктна множина	Активізоване правило
1	2	3	4
0	Goal, <i>L</i> , <i>M</i> , N	10, 11	10
1	Goal, A, <i>L</i> , <i>M</i> , N	11, 6, 7	11
2	Goal, A, B, <i>L</i> , <i>M</i> , N	6, 7, 8, 9	6
3	Goal, A, B, C, <i>L</i> , <i>M</i> , N	7, 8, 9, 1	7
4	Goal, A, B, C, D, <i>L</i> , <i>M</i> , N	8, 9, 1, 2	8
5	Goal, A, B, C, D, E, <i>L</i> , <i>M</i> , N	9, 1, 2, 3	9

Продовження таблиці 2.2

1	2	3	4
6	Goal, A, B, C, D, E, F, L, M , N	1, 2, 3, 4, 5	1
7	Goal, A, B, C, D, E, F, G, H, L, M , N	2, 3, 4, 5	2
8	Goal, A, B, C, D, E, F, G, H, I, K, L, M , N	3, 4, 5	3
9	Goal, A, B , C, D, E , F, G, H, I, K, L, M , N	4, 5	Зупинка

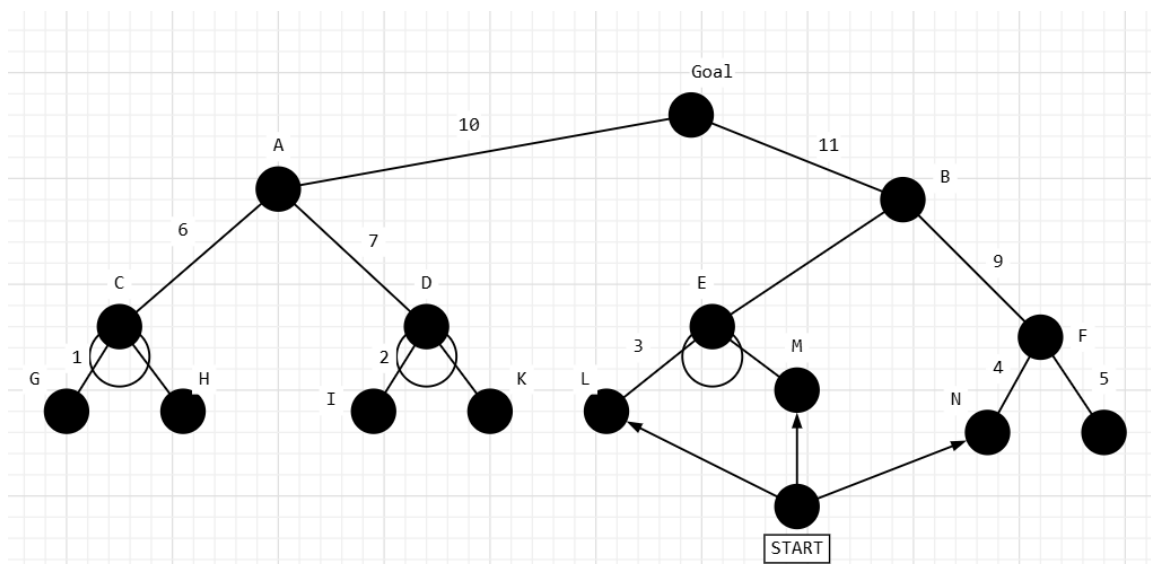


Рисунок 2.3 – Граф зворотного виведення [8]-[9]

Цей процес триває, доки умови певного правила не співпадуть із фактами, наявними в робочій пам'яті. У цьому випадку вивід, правила теж одержує значення “істина”. Дане істинне значення поширюється вгору по ланцюжку активізованих правил, кщо цільове твердження Goal набуває значення «істина», процес завершується. Якщо ж цього не відбувається, система знову обирає перше правило з конфліктної множини і продовжує виведення і т.д. [8]-[9].

Зазвичай подібного роду метод виведення використовується для перевірки досяжності цілей – системах прогнозування [5]-[7].

2.5 Висновок до розділу 2

Розділ 2 був присвячений аналізу технічних засобів і методів, що були застосовані для розробки експертної системи. Вибір мови програмування Python версії 3.9 обґрунтовано її простотою, широким спектром бібліотек і кросплатформною сумісністю, що дозволяє створювати гнучкі та масштабовані рішення.

Для створення зручного графічного інтерфейсу було використано бібліотеку Tkinter, яка інтегрується зі стандартною бібліотекою Python і забезпечує інтуїтивно зрозумілий дизайн без додаткових залежностей. Її легкість у використанні, підтримка кастомізації та можливість роботи на різних операційних системах роблять ідеальним вибором для розробки.

Інтеграція бази знань із JSON-файлами та використання модуля Logging забезпечують зручне зберігання даних і детальну реєстрацію операцій. Формат JSON гарантує простоту редагування та сумісність із різними платформами, тоді як Logging дозволяє відстежувати процеси та діагностувати можливі збої, що підвищує стабільність і надійність системи. Такий підхід спрощує управління знаннями та забезпечує їх доступність для користувачів із різним рівнем підготовки.

Розглянуті алгоритми прямого та зворотного виведення демонструють різні стратегії логічних міркувань, які є основою роботи експертних систем. Прямий вивід – базується на генерації нових фактів із відомих даних, а зворотний вивід – перевіряє досяжність цілей через аналіз гіпотез.

Загалом, обрані технічні засоби та методи створюють міцну основу для розробки ефективної й універсальної експертної системи, готової до подальшого вдосконалення та практичного застосування.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОБОЛОНКИ ЕКСПЕРТНОЇ СИСТЕМИ

3.1 Архітектура системи та її компоненти

Архітектура оболонки експертної системи побудована на основі модульного підходу з використанням об'єктно-орієнтованого програмування (ООП). Система складається з двох основних класів:

- «ExpertSystem» – відповідає за логіку обробки знань, а саме: база знань, робоча пам'ять, алгоритми виведення;
- «ExpertSystemGUI» – реалізує графічний інтерфейс для взаємодії з користувачем.

Ці компоненти взаємодіють між собою через чітко визначені інтерфейси, що забезпечує гнучкість і можливість розширення системи.

База знань була представлена як список продукційних правил, а робоча пам'ять — як множина фактів, що зберігаються в оперативній пам'яті та синхронізуються з JSON-файлом [10]-[12].

Машина виведення обробляє правила, використовуючи алгоритми прямого та зворотного виведення.

3.2 Опис функціональності програми

Основний функціонал програми був реалізований у двох класах: «ExpertSystem» та «ExpertSystemGUI».

Клас «ExpertSystem» містить наступні методи:

- add_rule – додає правило до бази знань;
- add_fact – додає факт до робочої пам'яті;
- forward_chaining – виконує прямий вивід повертаючи нові факти;
- backward_chaining – виконує зворотний вивід та перевіряє досяжність тих чи інших цілей;
- evaluate_condition – виконує оцінку умови враховуючи логічні оператори.

«add_rule» - відповідає за додавання нового продукційного правила до бази знань. Користувач на головному екрані може натиснути на кнопку «Додати правило» (рис. 3.1).

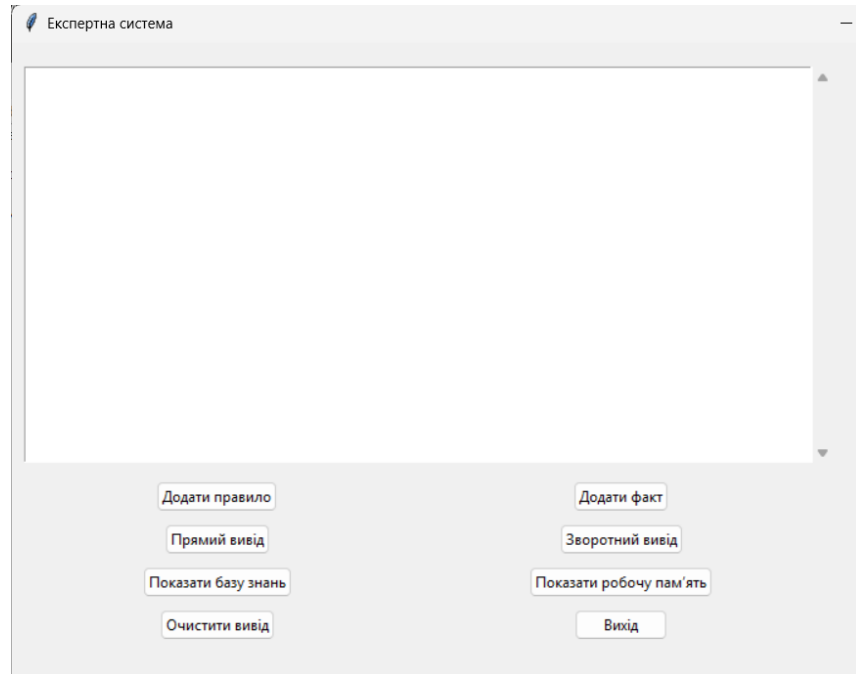


Рисунок 3.1 – Головний екран

Після її натикання користувач викликає спливаюче вікно (рис. 3.2).

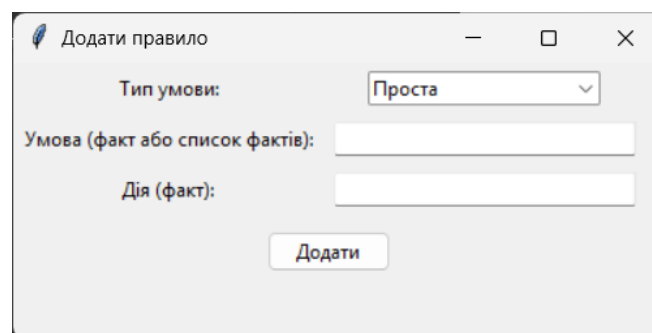


Рисунок 3.2 – Вікно «Додати правило»

В цьому вікні користувач може: обрати тип умови логічного оператора дії, і дію. Наприклад, для діагностики обладнання користувач обирає «AND» у випадальному списку. Далі, вписує умову «Overheat» та «LowOil», а в поле дії вводить «Warning» (рис. 3.3).

Рисунок 3.3 – заповнення вікно «Додати правило»

Таким чином формується наступне правило: «IF Overheat AND LowOil THEN Warning». Далі, метод перевіряє коректність введеного правила та додає його до бази знань, а також записує операцію у локальний файл під назвою «expert_system.log».

У текстовому полі, що знаходиться на головному екрані програми, після додавання правила, виводиться повідомлення про успішне його додавання (рис. 3.4).

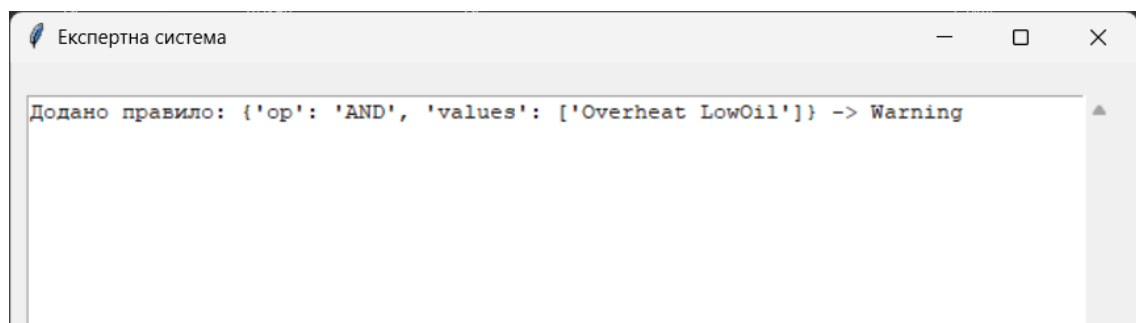


Рисунок 3.4 – Повідомлення про успішне додавання правила

«add_fact» - метод, що додає новий факт до робочої пам'яті. На головному екрані (рис. 3.1) користувач може натиснути на кнопку «Додати факт», що розташована у верхній частині поряд із кнопкою «Додати правило». Після її натискання відкривається вікно з одним єдиним текстовим полем під назвою «Факт» (рис. 3.5).

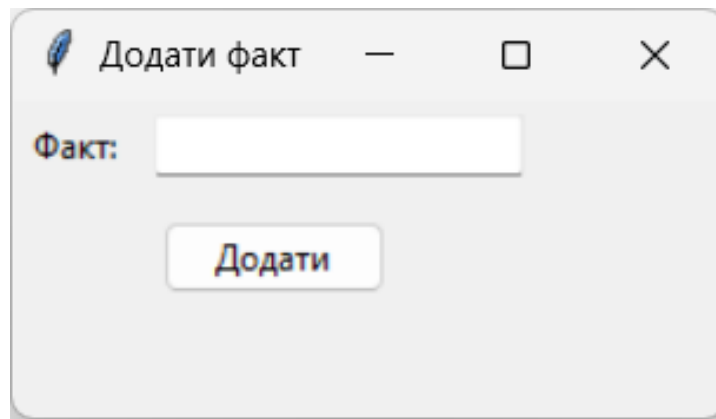


Рисунок 3.5 – Вікно «Додати факт»

У цьому текстовому полі користувач вводить потрібний йому факт, наприклад, для вказівки стану обладнання користувач може ввести «Overheat» та натиснути кнопку «Додати» (рис. 3.6).

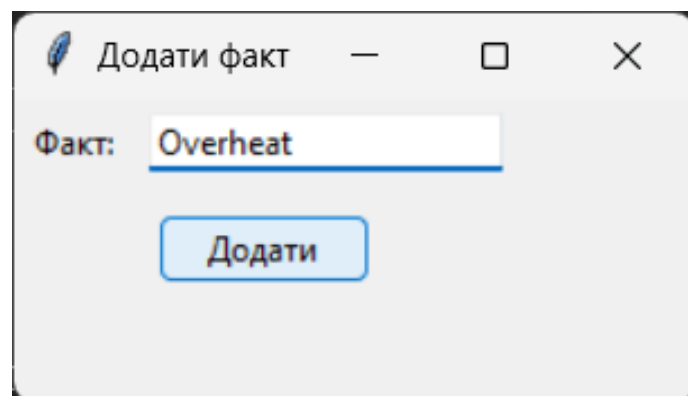


Рисунок 3.6 – Вікно «Додати факт» з введеним фактом

Після чого метод спочатку перевіряє чи не є цей факт порожнім, після перевірки додає його до робочої пам'яті. Згодом, якщо все пройшло успішно, то інформація про додавання факту до робочої пам'яті буде виведенна на текстове поле, що на головному екрані програми (рис. 3.7).

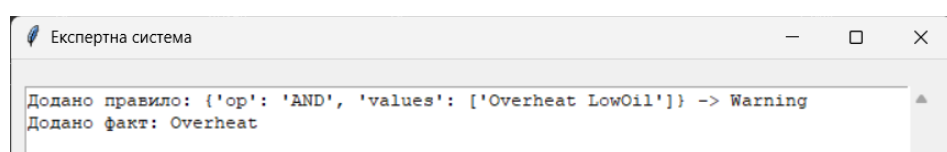


Рисунок 3.7 – Повідомлення про успішне додавання факту

«evaluate_condition» – це метод, що оцінює правила із логічними операторами, які використовуються у прямому і зворотному виводі. Наприклад, для правила: «IF Overheat AND LowOil THEN Warning» цей метод перевіряє чи наявні ці факти у робочій пам'яті, і якщо так, то метод повертає значення «True». Цей метод викликається автоматично, і його результати суттєво впливають на те, що буде відображатися у текстовому полі на головному екрані програми.

Метод «forward_chaining» – виконує прямий вивід, генеруючи нові факти і правила на основі вже існуючих. На головному екрані програми користувач натискає на кнопку «Прямий вивід» (рис. 3.8).

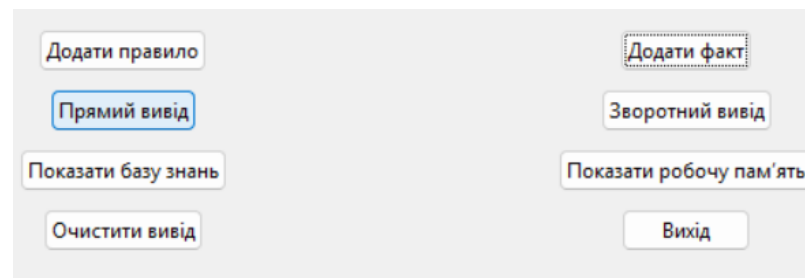


Рисунок 3.8 – Кнопка «Прямий вивід» на головному екрані

Після її натискання програма одразу виконує запит без відкриття додаткових вікон. У текстовому полі ми бачимо наступне (рис. 3.9).

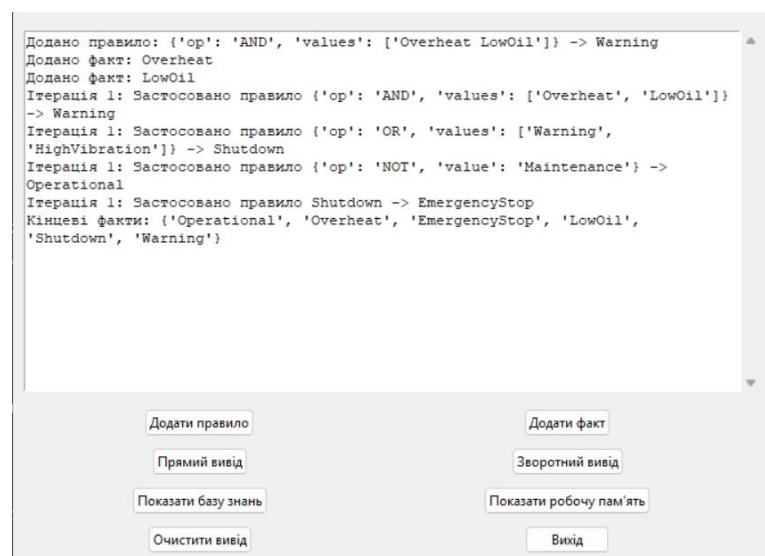


Рисунок 3.9 – Прямий вивід

Оскільки, в нас наявні такі факти як: «Overheat» і «LowOil», то виконується умова яку користувач додав раніше: «Overheat AND LowOil», і таким чином додається новий факт «Warning». Далі, в базі знань було пристунє правило з умою: «Warning OR HighVibration». Оскільки, факт «HighVibration» не є присутнім, але факт «Warning» є, то умова стає істиною і додається новий факт – «Shutdown». Потім, перевіряється чи є обладнання на технічному обслуговуванні – «Maintenance». В даному випадку ні, а тому додається новий факт – «Operational», тобто обладнання в робочому стані. І згодом метод «evaluate_condition» перевіряє умову «Shutdown», та оскільки вона вже присутня, тобто істина, то метод «forward_chaining» додає новий факт «EmergencyStop». А далі в текстовому полі виводяться кінцеві факти, що тепер містяться в робочій пам'яті.

Метод «backward_chaining» – реалізовує зворотний вивід для того аби перевірити досяжність тих чи інших цілей. Нариклад, користувач натискає на кнопку «Зворотний вивід» (рис. 3.10).

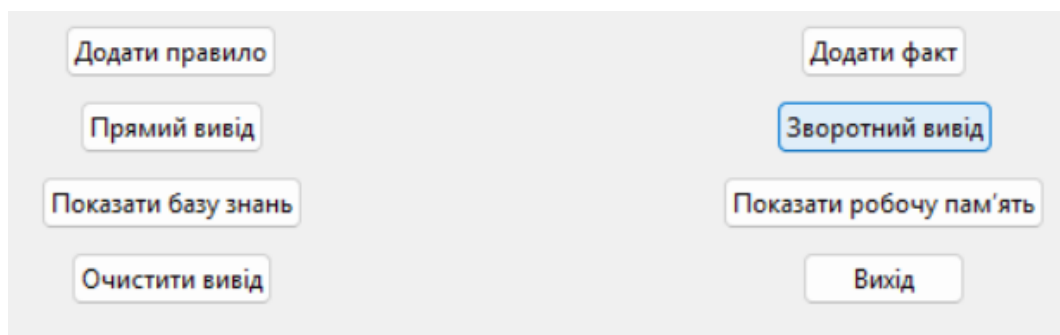


Рисунок 3.10 – Кнопка «Зворотний вивід»

Після чого, відкривається вікно де користувач може написати ціль яку потрібно перевірити. Отже, в нашому випадку нехай ціллю буде «Shutdown» (рис. 3.11).

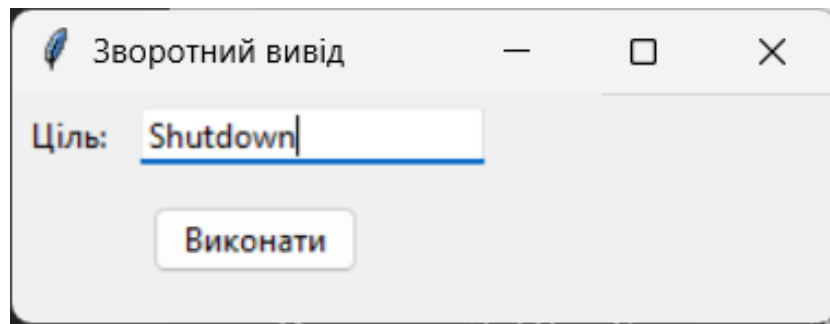


Рисунок 3.11 – Вікно з ціллю для перевірки

Далі, метод рекурсивно перевіряє чи можна вивести «Shutdown» за правилом: «IF Warning OR HighVibration THEN Shutdown», і якщо ціль досяжна, то в такому випадку у текстове поле на головному екрані виводиться повідомлення: «Мета Shutdown досягнута». Проте, якщо ні, то – «Мета Shutdown не може бути досягнута» (рис. 3.12).

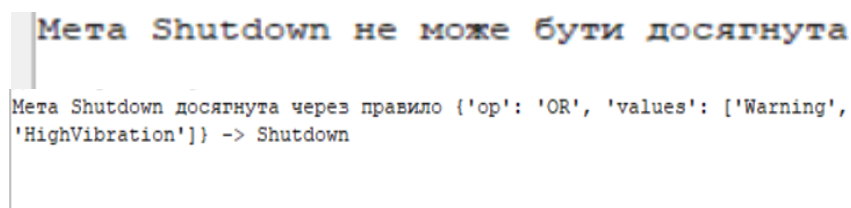


Рисунок 3.12 – Результат зворотного виводу

Далі поговоримо про базу знань. В програмі вона представлена як список продукційних правил, а саме структурований словник Python. В ньому кожне правило має умови з логічними операторами та діями, наприклад: «{"conditions": {"op": "AND", "values": ["Overheat", "LowOil"]}, "action": "Warning"}» (рис. 3.13).

```
База знань:
[
  {
    "conditions": {
      "op": "AND",
      "values": [
        "Overheat",
        "LowOil"
      ]
    },
    "action": "Warning"
  },

```

Рисунок 3.13 – База знань

Як зазначалося вище, користувач може змінювати та взаємодіяти з базою знань через кнопку «Додати правило». Фізично база знань зберігається у локальному JSON-файлі та оновлюється автоматично після кожної операції [11]-[12].

Робоча пам'ять – реалізована, як множина фактів Python «set()». В ній зберігаються поточні відомі факти, наприклад: «{'Overheat', 'HighVibration', 'LowOil'}» (рис. 3.14).

```
Робоча пам'ять: {'Overheat', 'HighVibration', 'LowOil'}
```

Рисунок 3.14 – Робоча пам'ять

Як було сказано раніше, користувач має змогу змінювати робочу пам'ять за допомогою кнопки «Додати факт». Факти оновлюються під час прямого введення, а робоча пам'ять синхронізується з базою знань. Завдяки чому це дозволяє зберігати стан між сеансами.

3.3 Структура графічного інтерфейсу та взаємодія з користувачем

Графічний інтерфейс користувача (GUI), було реалізовано за допомогою бібліотеки Python «Tkinter». Ця бібліотека забезпечує зручну

взаємодію з експертною системою через головне вікно з набором кнопок і текстовим полем для відображення результатів (рис. 3.15) [13]-[14].

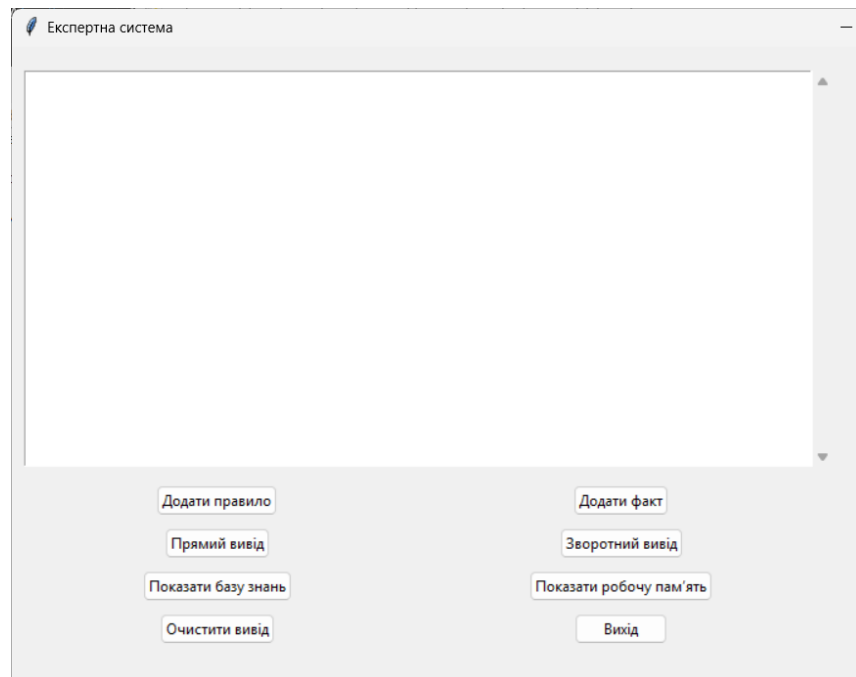


Рисунок 3.15 – Головний екран

Перше, що кидається в око це велике текстове поле на головному екрані програми. В ньому будуть відображатися усі взаємодії користувача з системою, а також факти виконання тих чи інших операцій (додавання правила, факту, тощо) (рис. 3.16).

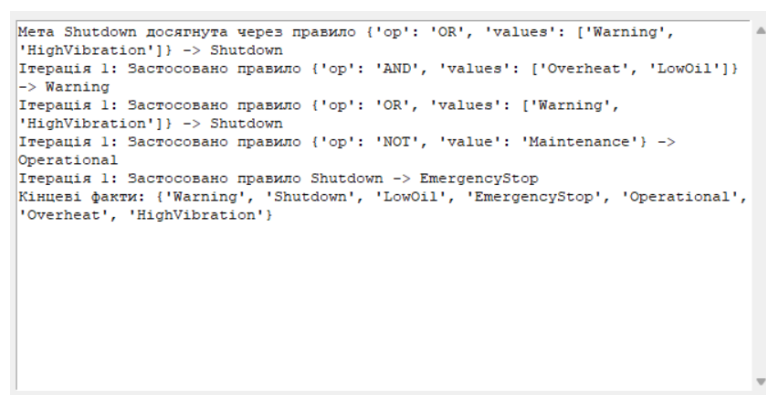


Рисунок 3.16 – Відображення взаємодій з системою

Нижче користувач бачить два стовбці з кнопками (рис. 3.17).

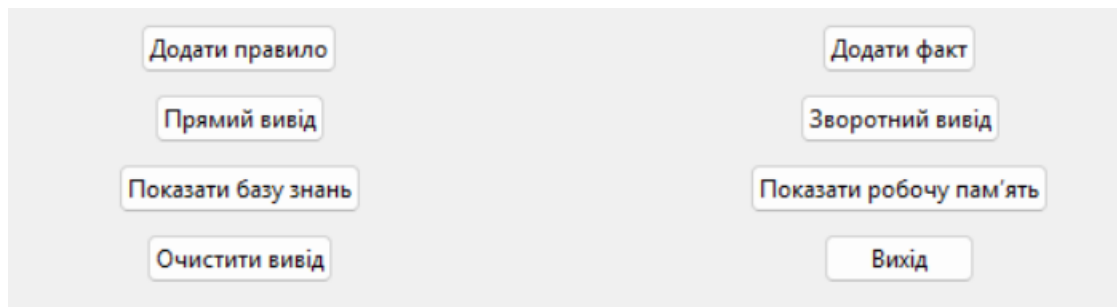


Рисунок 3.17 – Стовбці з кнопками

Почнемо з лівої частини. Перша кнопка - це «Додати правило». Як вже зазначалося у пункті 3.2 ця кнопка відповідає за додавання правил у базу знань експертної системи. При її натисканні відкривається відповідне вікно де користувач вводить ті правила які йому потрібні (рис. 3.18).

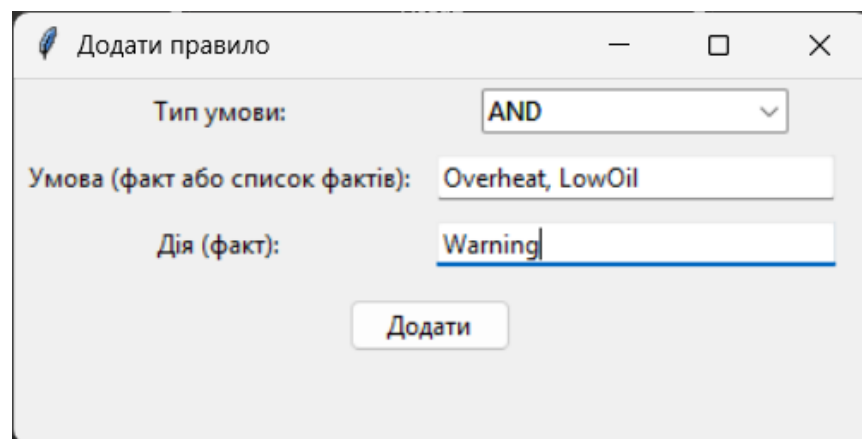


Рисунок 3.18 – Вікно додавання правила

Нижче, під кнопкою «Додати правило», розташовується кнопка «Прямий вивід». Вона відповідає за пряме мислення продукційної системи, тобто за генерацію нових фактів і правил на основі вже існуючих. Результат такого виводу можна побачити у текстовому полі, жодне додаткове вікно при цьому не відкривається (рис. 3.19) [5]-[7].

```

Ітерація 1: Застосовано правило {'op': 'AND', 'values': ['Overheat', 'LowOil']}
-> Warning
Ітерація 1: Застосовано правило {'op': 'OR', 'values': ['Warning',
'HighVibration']} -> Shutdown
Ітерація 1: Застосовано правило {'op': 'NOT', 'value': 'Maintenance'} ->
Operational
Ітерація 1: Застосовано правило Shutdown -> EmergencyStop
Кінцеві факти: {'Warning', 'Shutdown', 'LowOil', 'EmergencyStop', 'Operational',
'Overheat', 'HighVibration'}

```

Рисунок 3.19 – Прямий вивід

Далі йде кнопка «Показати базу знань». Ця кнопка потрібна для того, аби користувач міг подивитися базу знань одразу в програмі, а не намагався шукати її десь у себе в зовнішній системі. При натисканні база знань виводиться у текстове поле програми та оскільки вона є великою, то користувач має змогу пролистувати її донизу (рис. 3.20).

```

База знань:
[
  {
    "conditions": {
      "op": "AND",
      "values": [
        "Overheat",
        "LowOil"
      ]
    },
    "action": "Warning"
  },
  {
    "conditions": {
      "op": "OR",
      "values": [
        "Warning",
        "HighVibration"
      ]
    },
  },
]

```

Рисунок 3.20 – База знань у текстовому полі програми

Потім йде кнопка «Очистити вивід». Тобто, вона буквально очищає текстове поле від тої інформації, що в ній була.

Далі ми переходимо до правого стовбчика. Перша кнопка в нас це – «Додати факт». Як зазначалося у пункті 3.2 при її натисканні користувач відкриває вікно «Додати факт» з одним текстовим полем «Факт» та кнопкою

«Додати» (рис. 3.21). В ньому користувач вписує той факт який йому потрібен та натискає кнопку «Додати», щоб цей факт додався до робочої пам'яті.

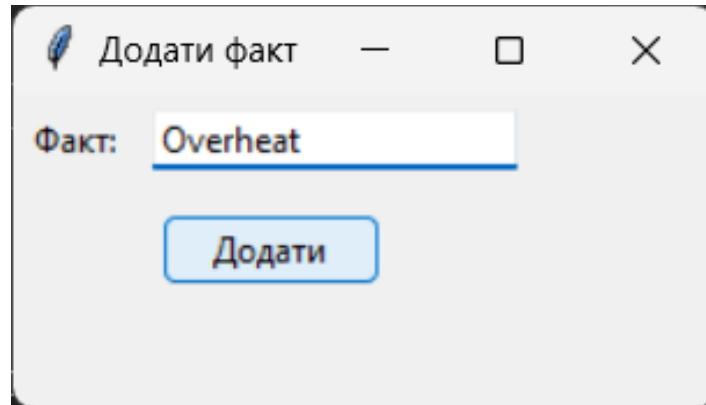


Рисунок 3.21 – Вікно «Додати факт»

Далі в нас йде кнопка «Зворотний вивід». У пункті 3.2 згадувалося, що ця кнопка відповідає за зворотнє мислення продукційної системи аби перевірити досяжність тих чи інших цілей. При її натисканні відкривається додаткове вікно з одним текстовим полем та кнопкою «Виконати». Користувач у текстовому полі вводить ту ціль яку потрібно перевірити та натискає кнопку «Виконати» (рис. 3.22) [5]-[7].

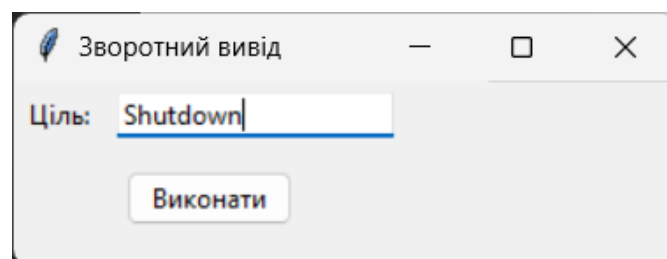


Рисунок 3.22 – Вікно «Зворотний вивід»

Нижче від кнопки «Зворотний вивід» розташовується кнопка «Показати робочу пам'ять». Натиснувши на неї користувачу у текстове поле виводяться ті факти, що зараз знаходяться у робочій пам'яті (рис. 3.23).

```
Робоча пам'ять: {'Warning', 'Shutdown', 'LowOil', 'EmergencyStop',  
'Operational', 'Overheat', 'HighVibration'}
```

Рисунок 3.23 – Робоча пам'ять

І остання кнопка це – «Вихід». При її натисканні користувач завершує роботу програми.

3.4 Висновок до розділу 3

У розділі 3 було детально описано програмну реалізацію експертної системи. Вона базується на модульній архітектурі з використанням об'єктно-орієнтованого програмування. Також, система включає ключові компоненти, такі як клас «ExpertSystem» для обробки логіки знань, включаючи базу знань, робочу пам'ять і алгоритми виведення, та клас «ExpertSystemGUI» для забезпечення зручного графічного інтерфейсу. Такий підхід забезпечує гнучкість, розширюваність і чітку взаємодію між модулями, що є основою для ефективної роботи системи в різних сценаріях.

Функціональність програми реалізовано через набір методів, які дозволяють користувачам додавати правила та факти, виконувати прямий і зворотний вивід, а також оцінювати умови з урахуванням логічних операторів.

Інтеграція бази знань із JSON-файлами та використання модуля Logging забезпечують зручне зберігання даних і детальну реєстрацію операцій. Приклади роботи з інтерфейсом – додавання правил типу: «IF Overheat AND LowOil THEN Warning», демонструють практичну застосовність розробленого рішення.

Графічний інтерфейс, створений за допомогою бібліотеки Tkinter, забезпечує інтуїтивно зрозумілу взаємодію з користувачем через головне вікно з кнопками та текстовим полем для відображення результатів. Структура інтерфейсу включає: функції додавання фактів, виконання

виведень і перегляду бази знань чи робочої пам'яті. Це дозволяє системі бути доступною для широкого кола застосувань, від діагностики обладнання до прогнозування в різних галузях.

Загалом, реалізація оболонки експертної системи демонструє вдале поєднання технічних інструментів і логічних алгоритмів, таких як прямий і зворотний вивід, що забезпечують високу ефективність і надійність.

Розроблена система готова до практичного використання, а її модульна структура відкриває можливості для подальшого вдосконалення та адаптації до нових задач, що підтверджує її потенціал у сучасних умовах.

4 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

4.1 Застосування системи в різних предметних областях

Розроблена оболонка експертної системи є універсальним інструментом, що можна використати, наприклад, у таких предметних галузях, як: технічна діагностика, економічний аналіз, освіта, тощо. В даному випадку було використано за приклад технічну діагностику. Користувач додавав нове правило: «IF Overheat AND LowOil THEN Warning», для того аби виявити несправності в обладнанні. Далі, експертна система обробила надані факти та правила та прийшла до висновку: «Warning» і «Shutdown». Подібного роду результати, є підтвердженням адаптивності системи щодо різних реальних сценаріїв діагностики.

В освітній сфері така експертна система може бути використана для того аби студенти мали змогу створювати бази знань, вводити факти, перевіряти досягнись тих чи інших цілей, тощо.

В економічному сенсі – система може оцінювати ризики, аналогічні до того, як це було в прикладах щодо технічної діагностики.

4.2 Обмеження розробленої оболонки та шляхи вдосконалення системи

Дана експертна система хоч і має деяку, так звану, універсальність, але все ж в ній присутні певні обмеження, наприклад:

- відсутність підтримки ймовірнісних правил – це дає обмеження на перевірку нечітких правил;
- візуалізація бази знань – оскільки вона відображається лише у текстовому полі, без будь-якого графічного представлення, то такий підхід може бути важким для розуміння цієї бази знань користувачем;
- відсутність імпорту правил із зовнішніх джерел.

Ці обмеження ускладнюють використання системи у великих проєктах. Проте їх все ж можна усунути додавши підтримку ймовірнісних правил, розробивши графічний редактор для бази знань. Також, можна інтегрувати підтримку SQL для великих баз знань, та автоматичний імпорт і експорт правил у CSV, разом із валідацією даних.

Якщо виправити обження реалізувавши усі запропоновані удосконалення, то таким чином це покращить взаємодію користувача із системою, полегшить обробку великих баз знань, забезпечить ширше застосування системи не тільки в простих, але і в складних проєктах. Також, впровадження подібних речей може підвищити конкурентноспроможність даної експортної системи.

ВИСНОВКИ

Розроблена оболонка експертної системи на основі продукційної моделі успішно реалізує поставлені завдання. Вона забезпечує автоматизацію прийняття рішень в різного роду предметних областях. Також, підтримує додавання правил і фактів, виконує прямий та зворотий вивід та має змогу зберігати дані у JSON-файлі і логувати операції. Графічний інтерфейс, що був реалізований за допомогою бібліотеки «Tkinter», забезпечує зручність використання для користувачів.

Незважаючи на досягнуті результати, система все ж має ряд обмежень, а саме відсутність: ймовірнісних правил, графічної візуалізації бази знань, підтримки SQL для великих баз знань та імпорту правил із зовнішніх джерел. Подібні та інші напрями вдосконалення відкривають перспективи для подальшого розвитку системи.

Таким чином, розроблена експертна система є ефективним рішенням для автоматизації експертних задач і може бути основою для створення більш складних інтелектуальних систем [1].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: Навчальний посібник. — Запоріжжя: ЗНТУ, 2008. — 341 с.
2. Експертна система // Вікіпедія [Електрон. ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Експертна_система.
3. Expert Systems // GeeksforGeeks [Electronic resource]. — Access mode: <https://www.geeksforgeeks.org/expert-systems/>.
4. MYCIN // Вікіпедія [Електрон. ресурс]. — Режим доступу: <https://uk.wikipedia.org/wiki/Mycin>.
5. Продукційна модель // Вікіпедія [Електрон. ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Продукційна_модель.
6. What is a Production System in AI? // GeeksforGeeks [Electronic resource]. — Access mode: <https://www.geeksforgeeks.org/what-is-a-production-system-in-ai/>.
7. Продукційні правила // Вікіпедія [Електрон. ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Продукційні_правила.
8. Комп'ютерні науки (лекції) для кафедри АСУ НУ "ЛП". Л4 – Продукційні моделі // [Електрон. ресурс]. — Режим доступу: <https://knhelp.wordpress.com/2012/04/19/%d0%bb4-%d0%bf%d1%80%d0%be%d0%b4%d1%83%d0%ba%d1%86%d1%96%d0%b9%d0%bd%d1%96-%d0%bc%d0%be%d0%b4%d0%b5%d0%bb%d1%96/>.
9. Лекції 3-4 // [Електрон. ресурс]. — Режим доступу: https://baklaniv.at.ua/PRSES/lekcija_3-4_2016.pdf.
10. Субботін С. О., Олійник А. О. Інтелектуальні системи: Навчальний посібник. — Запоріжжя: ЗНТУ, 2014. — 218 с.
11. База знань // Вікіпедія [Електрон. ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/База_знань.
12. Role of Knowledge Bases in Intelligent Systems // GeeksforGeeks

[Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/role-of-knowledge-bases-in-intelligent-systems/>.

13. Python [Электрон. ресурс]. — Режим доступа: <https://www.python.org/>.

14. Tkinter [Электрон. ресурс]. — Режим доступа: <https://docs.python.org/3/library/tkinter.html>.

ДОДАТОК А
Текст програми

A.1 Файл expertSYS.py

```

import json
import logging
import tkinter as tk
from tkinter import messagebox, ttk, scrolledtext
from datetime import datetime

# Налаштування логування
logging.basicConfig(filename='expert_system.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

class ExpertSystem:
    def __init__(self, knowledge_file='knowledge_base.json'):
        self.knowledge_base = [] # Список правил: [{'conditions': ..., 'action': ...}]
        self.working_memory = set() # Набір фактів
        self.knowledge_file = knowledge_file
        self.load_knowledge_base()
        logging.info("Експертна система ініціалізована")

    def load_knowledge_base(self):
        """Завантажує базу знань із JSON-файлу."""
        try:
            with open(self.knowledge_file, 'r') as f:
                data = json.load(f)
                self.knowledge_base = data.get('rules', [])
                self.working_memory = set(data.get('facts', []))

```

```

        logging.info("База знань успішно завантажена")
except FileNotFoundError:
    logging.warning("Файл бази знань не знайдено, створюється новий")
    self.save_knowledge_base()

```

```

def save_knowledge_base(self):
    """Зберігає базу знань у JSON-файл."""
    data = {'rules': self.knowledge_base, 'facts': list(self.working_memory)}
    with open(self.knowledge_file, 'w') as f:
        json.dump(data, f, indent=4)
    logging.info("База знань збережена")

```

```

def add_rule(self, conditions, action):
    """Додає правило до бази знань."""
    self.knowledge_base.append({'conditions': conditions, 'action': action})
    self.save_knowledge_base()
    logging.info(f"Додано правило: {conditions} -> {action}")

```

```

def add_fact(self, fact):
    """Додає факт до робочої пам'яті."""
    self.working_memory.add(fact)
    self.save_knowledge_base()
    logging.info(f"Додано факт: {fact}")

```

```

def evaluate_condition(self, condition):
    """Оцінює умову, враховуючи логічні оператори."""
    if isinstance(condition, dict):
        op = condition.get('op')
        if op == 'AND':
            return all(self.evaluate_condition(c) for c in condition['values'])
        elif op == 'OR':
            return any(self.evaluate_condition(c) for c in condition['values'])
        elif op == 'NOT':
            return not self.evaluate_condition(condition['value'])
    return condition in self.working_memory

def forward_chaining(self):
    """Виконує прямий вивід."""
    output = []
    changes = True
    iteration = 0
    while changes:
        changes = False
        iteration += 1
        for rule in self.knowledge_base:
            if self.evaluate_condition(rule['conditions']) and rule['action'] not in
self.working_memory:
                self.working_memory.add(rule['action'])
                changes = True
                log = f"Ітерація {iteration}: Застосовано правило
{rule['conditions']} -> {rule['action']}\n"
                output.append(log)
                logging.info(log.strip())
    logging.info("Прямий вивід завершено")

```

```
output.append(f"Кінцеві факти: {self.working_memory}\n")
return output
```

```
def backward_chaining(self, goal):
    """Виконує зворотний вивід для перевірки мети."""
    output = []
    if goal in self.working_memory:
        log = f"Мета {goal} вже є у робочій пам'яті\n"
        output.append(log)
        logging.info(log.strip())
        return True, output

    for rule in self.knowledge_base:
        if rule['action'] == goal:
            if self.evaluate_condition(rule['conditions']):
                log = f"Мета {goal} досягнута через правило {rule['conditions']} -> {rule['action']}\n"
                output.append(log)
                logging.info(log.strip())
                return True, output
            else:
                if isinstance(rule['conditions'], dict):
                    conditions = rule['conditions']['values'] if rule['conditions']['op'] in ['AND', 'OR'] else [rule['conditions']['value']]
                else:
                    conditions = [rule['conditions']]
```

```

    success = True
    for cond in conditions:
        if isinstance(cond, str):
            sub_success, sub_output = self.backward_chaining(cond)
            output.extend(sub_output)
            success = success and sub_success
    if success:
        self.working_memory.add(goal)
        log = f"Мета {goal} досягнута через зворотний вивід\n"
        output.append(log)
        logging.info(log.strip())
        return True, output
    log = f"Мета {goal} не може бути досягнута\n"
    output.append(log)
    logging.warning(log.strip())
    return False, output

```

```

class ExpertSystemGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Експертна система")
        self.root.geometry("800x600")
        self.es = ExpertSystem()

    # Основний фрейм
    self.main_frame = ttk.Frame(self.root, padding="10")
    self.main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

```

```

# Текстове поле для виведення результатів
self.output_text = scrolledtext.ScrolledText(self.main_frame, height=20,
width=80, wrap=tk.WORD)
self.output_text.grid(row=0, column=0, columnspan=2, pady=10)


# Кнопки
tk.Button(self.main_frame, text="Додати правило",
command=self.add_rule_window).grid(row=1, column=0, pady=5)
tk.Button(self.main_frame, text="Додати факт",
command=self.add_fact_window).grid(row=1, column=1, pady=5)
tk.Button(self.main_frame, text="Прямий вивід",
command=self.run_forward_chaining).grid(row=2, column=0, pady=5)
tk.Button(self.main_frame, text="Зворотний вивід",
command=self.run_backward_chaining).grid(row=2, column=1, pady=5)
tk.Button(self.main_frame, text="Показати базу знань",
command=self.show_knowledge_base).grid(row=3, column=0, pady=5)
tk.Button(self.main_frame, text="Показати робочу пам'ять",
command=self.show_working_memory).grid(row=3, column=1, pady=5)
tk.Button(self.main_frame, text="Очистити вивід",
command=self.clear_output).grid(row=4, column=0, pady=5)
tk.Button(self.main_frame, text="Вихід",
command=self.exit_program).grid(row=4, column=1, pady=5)

```

```

def add_rule_window(self):
    """Відкриває вікно для додавання правила."""
    window = tk.Toplevel(self.root)
    window.title("Додати правило")
    window.geometry("400x300")

    ttk.Label(window, text="Тип умови:").grid(row=0, column=0, padx=5,
pady=5)
    condition_type = ttk.Combobox(window, values=["Проста", "AND", "OR",
"NOT"])
    condition_type.grid(row=0, column=1, padx=5, pady=5)
    condition_type.set("Проста")

    ttk.Label(window, text="Умова (факт або список фактів:").grid(row=1,
column=0, padx=5, pady=5)
    condition_entry = ttk.Entry(window, width=30)
    condition_entry.grid(row=1, column=1, padx=5, pady=5)

    ttk.Label(window, text="Дія (факт:").grid(row=2, column=0, padx=5,
pady=5)
    action_entry = ttk.Entry(window, width=30)
    action_entry.grid(row=2, column=1, padx=5, pady=5)

    def submit():
        cond_type = condition_type.get()
        cond_text = condition_entry.get().strip()
        action = action_entry.get().strip()

```

```

if not cond_text or not action:
    messagebox.showerror("Помилка", "Заповніть усі поля!")
    return

try:
    if cond_type == "Проста":
        condition = cond_text
    else:
        values = [v.strip() for v in cond_text.split(",") if v.strip()]
        if not values:
            raise ValueError("Список фактів порожній")
        if cond_type == "NOT" and len(values) > 1:
            raise ValueError("NOT підтримує лише один факт")
        condition = {'op': cond_type, 'values' if cond_type in ["AND", "OR"]
else 'value': values if cond_type in ["AND", "OR"] else values[0]}

        self.es.add_rule(condition, action)
        self.output_text.insert(tk.END, f"Додано правило: {condition} ->
{action}\n")
        window.destroy()
except Exception as e:
    messagebox.showerror("Помилка", f"Неправильний формат: {str(e)}")

    ttk.Button(window, text="Додати", command=submit).grid(row=3,
column=0, columnspan=2, pady=10)

```

```

def add_fact_window(self):
    """Відкриває вікно для додавання факту."""
    window = tk.Toplevel(self.root)
    window.title("Додати факт")
    window.geometry("300x150")

    ttk.Label(window, text="Факт:").grid(row=0, column=0, padx=5, pady=5)
    fact_entry = ttk.Entry(window, width=20)
    fact_entry.grid(row=0, column=1, padx=5, pady=5)

    def submit():
        fact = fact_entry.get().strip()
        if not fact:
            messagebox.showerror("Помилка", "Введіть факт!")
            return
        self.es.add_fact(fact)
        self.output_text.insert(tk.END, f"Додано факт: {fact}\n")
        window.destroy()

    ttk.Button(window, text="Додати", command=submit).grid(row=1,
column=0, columnspan=2, pady=10)

def run_forward_chaining(self):
    """Виконує прямий вивід."""
    results = self.es.forward_chaining()
    for line in results:
        self.output_text.insert(tk.END, line)

```

```

def run_backward_chaining(self):
    """Відкриває вікно для зворотного виведення."""
    window = tk.Toplevel(self.root)
    window.title("Зворотний вивід")
    window.geometry("300x150")

    ttk.Label(window, text="Ціль:").grid(row=0, column=0, padx=5, pady=5)
    goal_entry = ttk.Entry(window, width=20)
    goal_entry.grid(row=0, column=1, padx=5, pady=5)

    def submit():
        goal = goal_entry.get().strip()
        if not goal:
            messagebox.showerror("Помилка", "Введіть ціль!")
            return
        success, output = self.es.backward_chaining(goal)
        for line in output:
            self.output_text.insert(tk.END, line)
        window.destroy()

    ttk.Button(window, text="Виконати", command=submit).grid(row=1,
column=0, columnspan=2, pady=10)

def show_knowledge_base(self):
    """Показує базу знань."""
    self.output_text.insert(tk.END,
знань:\n{json.dumps(self.es.knowledge_base, indent=2)}\n")
f"База

```

```

def show_working_memory(self):
    """Показує робочу пам'ять."""
    self.output_text.insert(tk.END, f'Робоча пам'ять: {self.es.working_memory}\n')

def clear_output(self):
    """Очищає текстове поле."""
    self.output_text.delete(1.0, tk.END)

def exit_program(self):
    """Завершує програму."""
    self.root.destroy()
    logging.info("Програма завершена")

if __name__ == "__main__":
    # Ініціалізація початкових даних
    es = ExpertSystem()
    es.add_rule({'op': 'AND', 'values': ['Overheat', 'LowOil']}, 'Warning')
    es.add_rule({'op': 'OR', 'values': ['Warning', 'HighVibration']}, 'Shutdown')
    es.add_rule({'op': 'NOT', 'value': 'Maintenance'}, 'Operational')
    es.add_rule('Shutdown', 'EmergencyStop')
    es.add_fact('Overheat')
    es.add_fact('LowOil')
    # Запуск GUI
    root = tk.Tk()
    app = ExpertSystemGUI(root)
    root.mainloop()

```

ДОДАТОК Б
Слайди презентації

Національний університет “Запорізька політехніка”
Кафедра програмних засобів

Кваліфікаційна робота на тему:
Програмна реалізація оболонки
експертної системи на основі
продукційної моделі

Виконав
Ст. гр. КНТ-211
Керівник
д.т.н., професор

Дмитро КАТЕЛЕВСЬКИЙ
Сергій СУББОТІН

2025

1

Рисунок Б.1 – Слайд 1

Мета та завдання

Мета роботи полягала в розробці програмної оболонки експертної системи з графічним інтерфейсом для автоматизації прийняття рішень на основі продукційних правил.

Для досягнення мети роботи потрібно було виконати наступні завдання:

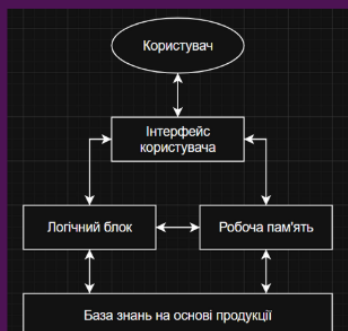
- Аналіз вимог до експертних систем;
- Вибір технологій розробки (Python 3.9, Tkinter);
- Розробка алгоритмів прямого та зворотного виведення;
- Створення зручного GUI та інтеграція з JSON.

2

Рисунок Б.2 – Слайд 2

Експертна система та продукційна модель

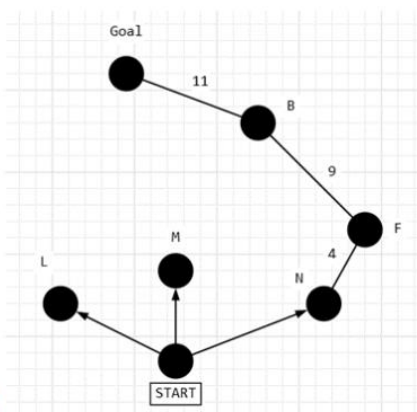
Експертні системи - це системи, що імітують мислення експерта, та створені для автоматизації прийняття рішень. Вони використовують продукційні моделі, що керуються правилом «ЯКЩО – ТО», мають базу знань та робочу пам'ять. Переваги: гнучкість, простота змін.



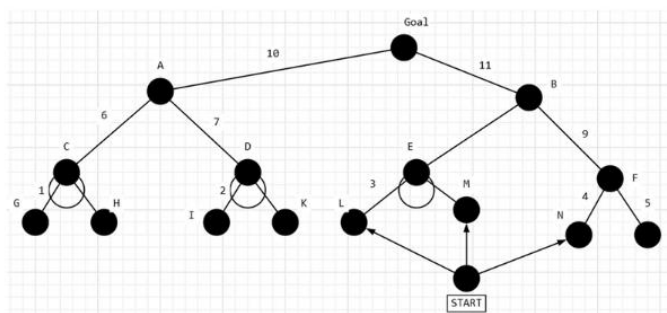
3

Рисунок Б.3 – Слайд 3

Алгоритми прямого та зворотного виведення



Пряме виведення



Зворотне виведення

4

Рисунок Б.4 – Слайд 4

Функціональні компоненти системи

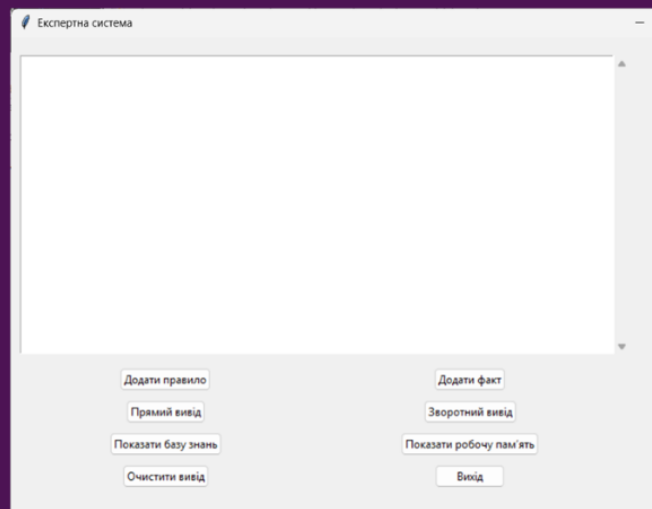
- Додавання правила до бази знань (метод: `add_rule`).
- Додавання факту до робочої пам'яті (метод: `add_fact`).
- Оцінка умов із логічними операторами AND/OR/NOT (метод: `evaluate_condition`).
- Виконання прямого виведення для генерації нових фактів (метод: `forward_chaining`).
- Виконання зворотного виведення для перевірки досяжності цілей (метод: `backward_chaining`).
- Збереження бази знань і фактів у JSON-файл (метод: `save_knowledge_base`).
- Завантаження бази знань із JSON-файлу (метод: `load_knowledge_base`).
- Логування операцій у файл `expert_system.log` (використання модуля `logging`).

5

Рисунок Б.5 – Слайд 5

Графічний інтерфейс

Графічний інтерфейс користувача було реалізовано за допомогою бібліотеки Tkinter через її стандартну інтеграцію з Python



6

Рисунок Б.6 – Слайд 6

Приклад роботи системи

Сценарій - діагностика обладнання. **Додано правило:** «IF Overheat AND LowOil THEN Warning». **Факти:** «Overheat», «LowOil». **Результат прямого виведення:** «Warning», «Shutdown», «EmergencyStop». **Зворотний вивід:** перевірка цілі «Shutdown» → досягнута.

```

Експертна система
Додано правило: {'op': 'AND', 'values': ['Overheat LowOil']} -> Warning
Додано факт: Overheat
Додано факт: LowOil
Ітерація 1: Састосовано правило {'op': 'AND', 'values': ['Overheat', 'LowOil']}
-> Warning
Ітерація 1: Састосовано правило {'op': 'OR', 'values': ['Warning', 'HighVibration']} -> Shutdown
Ітерація 1: Састосовано правило {'op': 'NOT', 'value': 'Maintenance'} -> Operational
Ітерація 1: Састосовано правило Shutdown -> EmergencyStop
Кінцеві факти: {'Operational', 'Overheat', 'EmergencyStop', 'LowOil', 'Shutdown', 'Warning'}
  
```

Мета Shutdown досягнута через правило {'op': 'OR', 'values': ['Warning', 'HighVibration']} -> Shutdown

Прямий вивід

Зворотний вивід

7

Рисунок Б.7 – Слайд 7

Обмеження та перспективи розвитку

Обмеження:

- Відсутність ймовірнісних правил;
- Не графічна візуалізація бази знань;
- Немає імпорту правил із зовнішніх джерел.

Вдосконалення:

- Додати ймовірнісні правила;
- Розробити графічний редактор бази знань;
- Інтеграція з SQL, імпорт/експорт CSV.

8

Рисунок Б.8 – Слайд 8

Висновок

Розроблено універсальну оболонку експертної системи на основі продукційної моделі знань, яка підтримує прямий і зворотний вивід для автоматизації прийняття рішень.

Графічний інтерфейс, створений за допомогою Tkinter, забезпечує зручність використання для широкого кола користувачів.

Інтеграція з JSON та логування операцій підвищують надійність системи та спрощують управління даними.

Система може успішно застосовуватися в таких галузях, як діагностика, освіта та економічний аналіз. Водночас вона має потенціал для вдосконалення, зокрема шляхом додавання ймовірнісних правил та графічної візуалізації бази знань.

9

Рисунок Б.9 – Слайд 9

Дякую за увагу

10

Рисунок Б.10 – Слайд 10