**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Західноукраїнський національний університет**

**Факультет комп’ютерних інформаційних технологій**

**Кафедра комп’ютерної інженерії**

**Рутецький Юрій Орестович**

**Алгоритми оптимізації туристичних маршрутів України / Algorithms for optimizing tourist routes in Ukraine**

спеціальність: 123 - Комп’ютерна інженерія

освітньо-професійна програма - Комп’ютерна інженерія

Кваліфікаційна робота

Виконав студент групи КІм-21

Рутецький Юрій Орестович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Науковий керівник:

д.т.н., професор О.М. Березький

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кваліфікаційну роботу допущено

до захисту:

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р.

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Тернопіль – 2022**

ЗМІСТ

[ВСТУП 7](#_Toc119618701)

[1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ОГЛЯД ВІДОМИХ МОДЕЛЕЙ ТА МОВ ПРЕДСТАВЛЕННЯ ЗНАНЬ КОНСУЛЬТУВАННЯ ТУРИСТІВ 10](#_Toc119618702)

[1.1 Аналіз відомих онлайн пошукових систем для вибору туристичних маршрутів 10](#_Toc119618703)

[1.2 Аналіз моделей представлення знань 19](#_Toc119618704)

[1.3 Огляд та обґрунтування вибору мови програмування для експертної системи 26](#_Toc119618705)

[1.4 Постановка задач дослідження 30](#_Toc119618706)

[1.5 Висновки до розділу 1 32](#_Toc119618707)

[2 РОЗРОБКА АРХІТЕКТУРИ ЕКСПЕРТНОЇ СИСТЕМИ 33](#_Toc119618708)

[2.1 Етапи розробки експертної системи 33](#_Toc119618709)

[2.2 Структура експертної системи 34](#_Toc119618710)

[2.3 Алгоритм роботи експертної системи 44](#_Toc119618711)

[2.4 Висновки до розділу 2 44](#_Toc119618712)

[3 ПРОГРАМНА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ЕКСПЕРТНОЇ СИСТЕМИ 54](#_Toc119618713)

[3.1 Технічні вимоги до функціонування експертної системи 54](#_Toc119618714)

[3.2 Програмна реалізація бази знань та даних 55](#_Toc119618715)

[3.2.1 Програмна реалізація бази знань та даних для подорожі поїздом 55](#_Toc119618716)

[3.2.2 Блок інтерфейсу користувача та блок прийняття рішення при подорожі автобусом 59](#_Toc119618717)

[3.2.3 Програмна реалізація бази знань та даних для подорожі літаком 62](#_Toc119618718)

[3.3 Розробка інтерфейсу системи. Тестування системи 68](#_Toc119618719)

[3.3.1 Розробка інтерфейсу системи подорожі поїздом. Тестування системи 69](#_Toc119618720)

[3.3.2 Розробка інтерфейсу системи подорожі автобусом. Тестування системи 72](#_Toc119618721)

[3.3.2 Розробка інтерфейсу системи подорожі літаком. Тестування системи 76](#_Toc119618722)

[3.4 Висновки до розділу 3 79](#_Toc119618723)

[ВИСНОВКИ 80](#_Toc119618724)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 81](#_Toc119618725)

[Додаток А 86](#_Toc119618726)

[Лістинг коду програми 86](#_Toc119618727)

[Додаток Б 111](#_Toc119618728)

[Світлокопії виданих публікацій 111](#_Toc119618729)

[Додаток В 113](#_Toc119618730)

[Довідка про використання 113](#_Toc119618731)

# ВСТУП

Написання кваліфікаційної роботи я розпочав ще до російсько-української війни. На той час в Україні достатньо сильно був розвинений автобусний, залізничний транспорт і ринок авіаційних перевезень. Залізничний і автобусний транспорт ходив регулярно по всій Україні, міжнародний аеропорт «Львів» ім. Данила Галицького здійснював по 20 вильотів на годину. Подорожувати Україною було безпечно. Через війну подорожі стали практично неможливими.

Тому база даних в кваліфікаційній роботі приведена на 23 лютого 2022 року. Віримо в ЗСУ і в те, що туристичні маршрути відновляться по всій Україні.

Штучний інтелект, як наукова дисципліна, складається з декількох великих течій. Однією з таких течій є експертні системи. Експертні системи виникли як напрям досліджень в галузі штучного інтелекту. Вони створюють обчислювальні системи, які вміють знаходити рішення, які співпадають з рішеннями експертів у певній предметній області.

В основному, експертні системи створюються для розв’язання практичних задач у деяких областях вузької спеціалізації [1-10]. В таких областях велику роль грають знання досвідчених фахівців. Завдяки експертним системам появилася можливість залучити велику цікавість до результатів досліджень в галузі штучного інтелекту [11-13].

Експертне знання – це синтез теорії і практики. Фундаментом експертної системи є база знань. Її будують на основі експертних знань фахівців. Експертна система буде працювати повноцінно тоді, коли буде правильно підібраний експерт і він вдало формалізував знання. Тому практичність експертної системи на 90% визначається якістю створеної бази знань.

Експертна система не є простою програмою, яку пише лише один або декілька програмістів. Її створюють експерти з даної предметної області, інженери по знаннях і програмісти.

Мета і завдання дослідження. Метою кваліфікаційної роботи є розробка алгоритмів оптимізації туристичних маршрутів України і експертної системи консультування пасажирів.

У відповідності із поставленою метою кваліфікаційна робота включає розв’язки таких задач:

* Провести аналіз відомих онлайн пошукових систем для консультування пасажирів залізничного, автобусного і авіа транспорту.
* Проаналізувати моделі представлення знань.
* Провести аналіз експертних систем.
* Спроектувати структуру експертної системи.
* Розробити модель представлення знань;
* Сформувати базу знань;
* Програмно реалізувати експертну систему;
* Зробити тестування розробленого продукту.

Об’єкт дослідження – туристичні маршрути України.

Предмет дослідження – алгоритми вибору оптимальних маршрутів.

Методи досліджень. У кваліфікаційній роботі використано методи аналізу та синтезу, методи обчислювального інтелекту, моделювання, методи пошуку рішень та представлення знань.

Наукова новизна одержаних результатів полягає в розробці алгоритмів оптимальних маршрутів подорожей різними видами транспорту.

Практичне значення одержаних результатів полягає в розробці експертної системи планування подорожей.

Публікації результатів досліджень. За результатами досліджень опубліковані двоє тез доповідей VІ науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі» (10 листопада 2022 р., м. Тернопіль, Західноукраїнський національний університет) [14, 15]:

1. Рутецький Ю. О. Експертна система для пошуку оптимальних варіантів подорожей. *Інтелектуальні комп’ютерні системи та мережі* : тези доп. VІ Наук.-практ. конф. молодих вчених і студентів (10 лист. 2022 р.). Тернопіль : ЗУНУ, 2022.
2. Березький М. О., Рутецький Ю. О. Мови штучного інтелекту і машинного навчання. *Інтелектуальні комп’ютерні системи та мережі* : тези доп. VІ Наук.-практ. конф. молодих вчених і студентів (10 лист. 2022 р.). Тернопіль : ЗУНУ, 2022.

Кваліфікаційна робота складається із трьох розділів, висновків, списку використаної літератури та додатків [16 - 18].

У першому розділі здійснено аналіз відомих онлайн пошукових систем для вибору туристичних маршрутів, проаналізовані моделі представлення знань, зроблено огляд та обґрунтування вибору мови програмування для експертної системи, здійснено аналіз завдання на кваліфікаційну роботу та постановку задач.

У другому розділі спроектовано архітектуру експертної системи, приведено етапи розробки та структуру експертної системи, алгоритми роботи експертної системи.

У третьому розділі реалізована і протестована експертна система. Приведено технічні вимоги до функціонування експертної системи, програмна реалізація бази знань та даних, розроблено інтерфейс системи, тестування системи

У додатку приведено лістинг коду програми, світлокопії виданих публікацій, довідка про використання.

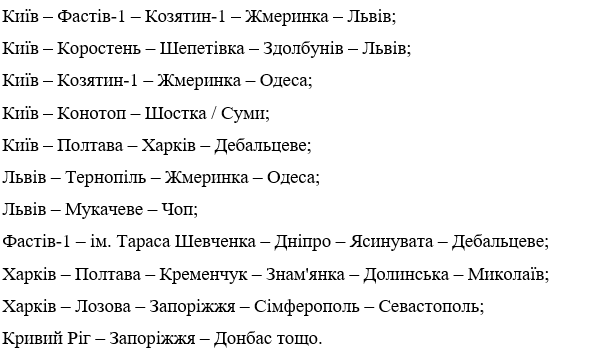
# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ОГЛЯД ВІДОМИХ МОДЕЛЕЙ ТА МОВ ПРЕДСТАВЛЕННЯ ЗНАНЬ КОНСУЛЬТУВАННЯ ТУРИСТІВ

## 1.1 Аналіз відомих онлайн пошукових систем для вибору туристичних маршрутів

Україна є дуже великою європейською державою з розвиненою мережею автомобільних доріг для пасажирських перевезень. У кожному місті України є автовокзал чи автовокзал, а з огляду на неповне покриття території країни залізничним транспортом, автобуси найчастіше є єдиним видом транспорту [19]. Автобуси в Україні відіграють важливу соціальну та інфраструктурну роль. Автобусні маршрути об'єднують усі населені пункти – міста та села України.

Залізничний транспорт – це галузь промисловості, частина транспортної мережі логістичних ланцюгів. Він сприяє міжнародній торгівлі і економічному зросту. У загальному залізничний транспорт України складає світову залізничну мережу [20].

Приведемо основні залізничні магістралі України:



Найбільшими залізничними вузлами, які мають розвинене станційне господарство є: Київ, Львів, Харків, Дніпро, Фастів-1, Козятин-1, Ясинувата, Жмеринка, ім. Тараса Шевченка та ін.

В Україні існують шість залізниць:

Південно-Західна (центром є Київ);

Львівська;

Південна (центр – Харків);

Донецька (центр з 2014 року – Лиман);

Придніпровська (центр – Дніпро);

Одеська.

Залізничний транспорт України загального користування організаційно підпорядковується Державній адміністрації залізничного транспорту. Україна інтегрується у європейську єдину залізничну систему. Найважливішою справою при цьому виявилося звуження ширини української залізничної колії (1520 +8; -4 мм) до європейського зразка (1435 мм). Така колія є у сусідніх країнах: Польща, Словаччина, Румунія та Угорщина. На даний час на кордоні з цими державами є складні системи переведення поїздів з колії на колію. Такі переводи значно знижують ефективність роботи залізничного транспорту, зокрема існує затримка в часі.

Український ринок авіаперевезень успішно розвивається. Минулий рік став одним із найуспішніших для повітряного транспорту в історії незалежної України [21].

На ринку пасажирських перевезень у 2021 році працювало 18 українських авіакомпаній. Тут абсолютними лідерами є «Міжнародні авіалінії України», «Роза вітрів», «Azur Air Ukraine», «Atlasjet Україна» і «Bravo Airways».

В українських авіакомпаній у парку не більше 10-15 літаків, у МАУ найбільше – понад 40. У світових лідерів набагато більше. Наприклад, American Airlines Group (США) – понад 900 літаків, China Southern Airlines (КНР) – понад 540, Ryanair (Ірландія) – 420, Lufthansa (Німеччина) та British Airways (Великобританія) – понад 270 кожна, Emirates (ОАЕ) понад 260, Air France (Франція) – понад 220.

Оскільки українські авіакомпанії переважно використовують літаки іноземного виробництва, середній вік їхнього парку становить від 11 до 35 років. Минулого року регулярні пасажирські перевезення здійснювалися у 43 країнах світу. На цих лініях перевезено 5 837 500 осіб (+18,1%), що також стало історичним рекордом. При цьому було відкрито 10 нових напрямків. Чартерними рейсами та авіакомпаніями з інших країн перевезено 3787200 осіб (+49%) [21].

Розглянемо онлайн пошукові системи для квитків залізничного транспорту. Їх є багато. Серед них онлайн пошукова система для квитків – Tickets.ua. Tickets.ua є українським онлайн-оператором, що підшукує електронні авіа-, залізничні та автобусні квитки для подорожей Україною, країнами Європи, трансфери із/в аеропорт, бронювання готелів, квитки на події, концерти, кінотеатри, оренду авто, оформлення страхових полісів для авто та для туристів [22].

На рисунку 1.1 показано головне вікно Tickets.ua і портфоліо послуг.

За версією журналу «Forbes Україна» компанія Tickets.ua в 2013 році посіла ІІ-е місце у рейтингу найбільших компаній Уанету.

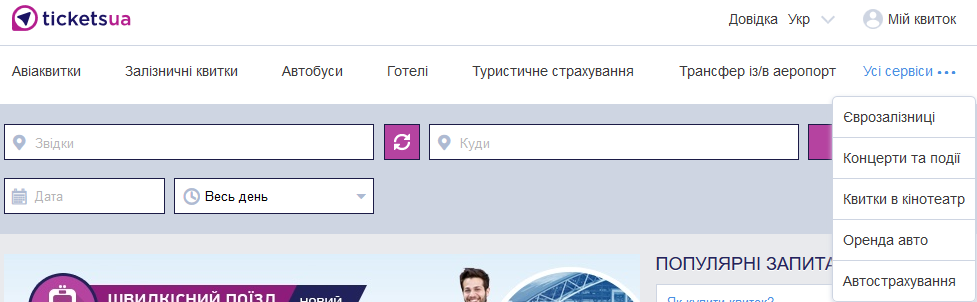


Рисунок 1.1 – Головне вікно Tickets.ua

Партнерами Tickets.ua являються світові лідери бізнесу подорожей: Booking.com, Amadeus, TripAdvisor.

Філії (тревел-сервіси) компанії Tickets.ua працюють в США, Німеччині, Польщі, Україні, Росії, Молдові, Білорусі, Казахстані та Азербайджані.

Засновником і генеральним директором Tickets.ua є Сергій Кравець. Заснована ця компанія була як традиційна агенція з продажу авіаквитків ще в 2009 році, а через 2 роки було створено систему онлайн-бронювання. Стратегічними партнерами з 2012 року Tickets.ua є провідна глобальна дистрибутивна система Amadeus і лідер онлайн-продажу туристичних послуг та перевезень КИЙ АВIА.

У системі Tickets.ua можливі кілька способів оплати квитків та послуг:

* банківською карткою;
* через Приват24;
* webmoney;
* на рахунок будь-якого банку України;
* безготівковим розрахунком (для юридичних осіб);
* через термінали iBox.

Для клієнтів Tickets.ua передача платіжних даних захищена сертифікатом SSL 128 біт, який підписаний компанією Thawte. Сайт відповідає стандартам безпеки платіжних систем Visa і MasterCard (PCI compliancy).

Офіси компанії Tickets.ua знаходяться у Києві та у Львові. В компанії працює чуть більше 200 людей.

На сайті компанії при купівлі залізничних квитків можна скористатися сервісами для мандрівників, що включають:

* розклад поїздів (розклад руху, маршрути, станції);
* пошук квитків за пасажира;
* повернення квитків онлайн.

Друга послуга «Шукаємо для вас» дуже корисна. Якщо на даний момент система не знайшла місця на вказаному маршруті або не знайшла потрібних місць (наприклад, ми хочемо нижні полиці, а є тільки верхні полиці), то Вам необхідно вказати номер поїзда, тип автомобіля, кількість та параметри місць (не більше 4- х), оптимальний час у дорозі та ціна, а також час, до якого ви повинні стежити за місцями. З появою місць замовник моментально отримує повідомлення на телефон та адресу електронної пошти, які він вказав при виборі цієї послуги. Після отримання повідомлення необхідно одразу перейти за посиланням у листі, забронювати квитки та сплатити замовлення протягом 30 хвилин з моменту його створення.

Розглянемо наступну онлайн-систему продажу квитків – I-Тravels (рис. 1.2). Онлайн-пошуковик цієї компанії дозволяє здійснювати пасажирські перевезення онлайн. I-Travels об'єднує співтовариство транспортних компаній України, що постійно росте, створює інформаційний зв'язок між перевізником і замовником. Сервіс компанії має необхідний рівень відповідно до зростаючих вимог ринку [23].

На сайті I-Travels є універсальний модуль пошуку транспорту для всіх видів транспорту та всіх видів перельотів: прямих та комбінованих. За відсутності транспорту між населеними пунктами або відсутності місць на рейсі (на певну дату), система автоматично пропонує альтернативний варіант проїзду.



Рисунок 1.2 – Головне вікно І-Тревелс

Якщо у вас виникли питання щодо бронювання або купівлі квитків на сайті, ви можете звернутися до служби онлайн підтримки системи, яка працює цілодобово, для їх вирішення.

Електронний квиток – це електронна форма замовлення. Він використовується замість звичного паперового документа. Бронювання місця в автобусі, літаку або поїзді здійснюється звичайним способом, але інформація про подорож зберігається в інформаційній базі онлайн-систем у вигляді цифрового запису.

Головна відмінність електронного квитка від паперового – зручність. Квиток не забуто, не втрачено, не знищено фізично. Продаж квитків здійснюється онлайн. Більше не потрібно йти до каси та витрачати час у черзі за електронним квитком.

Оплачується будь-яким способом. Процес замовлення квитка досить простий і дозволяє самостійно підібрати потрібний рейс, за часом і датою. Замовлення квитків обробляються онлайн-системою на сайті. Копія листа про бронювання надсилається на адресу електронної пошти одночасно з повним оформленням замовлення. Після надходження оплати квиток моментально змінює статус та закріплюється за пасажиром.

Наступні онлайн системи орієнтовані на один з видів транспорту.

Система купівлі квитків – Booking.uz [24] орієнтована лише на купівлю залізничних квитків. Головне вікно Booking.uz показано на рисунку 1.3.

На сайті при виборі квитків дуже зручно можна скористатися популярними напрямками, популярними містами (не вводити з клавіатури, а лише вибрати їх), зручним вибором дати: сьогодні, завтра, післязавтра, а всі інші за календарем, прямими поїздками, або поїдками з пересадками (при потребі, або при відсутності прямих рейсів).

При натисненні кнопки «Пошук поїздів» на екран буде виведено перелік поїздів з наявними вільними місцями відповідно до введених параметрів пошуку. Також буде відображено час відправлення та прибуття, тривалість поїздки тощо.

На сайті можна здійснювати бронювання квитків, повернення квитків.

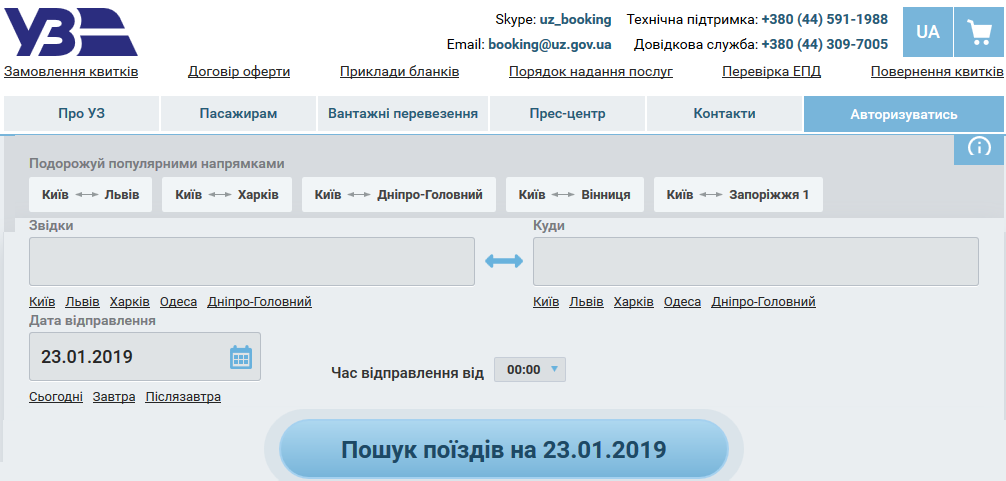


Рисунок 1.3 – Головне вікно Booking.uz

В системі є можливість переглянути поїзди без вільних місць, натиснувши на посилання «Показати без вільних місць» над списком поїздів. Щоб приховати поїзди без вільних місць, треба натиснути на посилання «Приховати без вільних місць».

На сайті існує можливість оформити проїзні документи для пільгових категорій користувачів. Для цього перейшовши до сторінки введення даних про пасажира та послуги, треба заповнити поля «Прізвище» та «Ім’я», після чого натиснути кнопку «Пільговий».

У спливаючому вікні обрати «Пільговий» тип квитка, вказати категорію пільги, а також зазначити серію та номер пільгового посвідчення, натиснути кнопку «ОК» у спливаючому вікні.

Повернення проїзних документів на сайті booking.uz.gov.ua здійснюється через особистий кабінет користувача (за умови, що проїзний документ попередньо не був роздрукований у квитковій касі АТ «Укрзалізниця»).

Busfor (Україна) – це зручний сервіс підбору міжнародного і міжміського автобусного маршруту, бронювання та придбання квитків [25]. Перевізники, з якими співпрацює портал, надають повні дані і про майбутню поїздку, і, по можливості, про стан рухомого складу. Замовивши квиток на сайті Busfor.ua, пасажир буде ознайомлений не тільки з часом і місцем відправлення, але, в більшості випадків, зможе дізнатися на якому автобусі поїде, де розміщене його місце, які населені пункти буде проїжджати і т.д.

На рисунку 1.4 показано зразок оформлення квитка на сайті Busfor.ua.

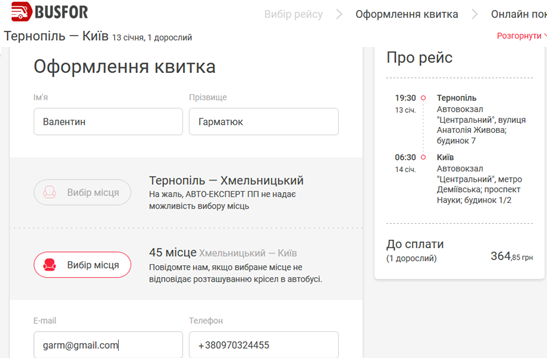


Рисунок 1.4 – Зразок оформлення квитка на сайті Busfor.ua

Маршрути плануються з більшості міст України. Якщо немає прямого маршруту, система Busfor.ua запропонує пересадку (пересадки). При цьому будуть заброньовані квитки на всі етапи руху, а рейси підібрані з урахуванням найменшого очікування. відповідними запиту. На сторінці результатів пошуку є можливість вибрати тільки прямі рейси. Для цього на сайті є позначка «Тільки прямі рейси». Знявши її, будуть показані маршрути з пересадками. У кожної пропозиції є функція «Детальніше» (посилання зліва від кнопки «Вибрати»). Натиснувши на неї, відкриється вся інформація по конкретному маршруту, місце і час пересадки, скільки доведеться чекати і багато іншого.

На сайті також вказані популярні напрямки з міста, в якому знаходиться пасажир, або з будь-якого іншого міста.

При виборі квитка рекомендуються найвигідніші пропозиції за ціною квитка.

Наступною розглянемо онлайн-систему для купівлі квитків на авіа перельоти TripMyDream (рисунок 1.5).

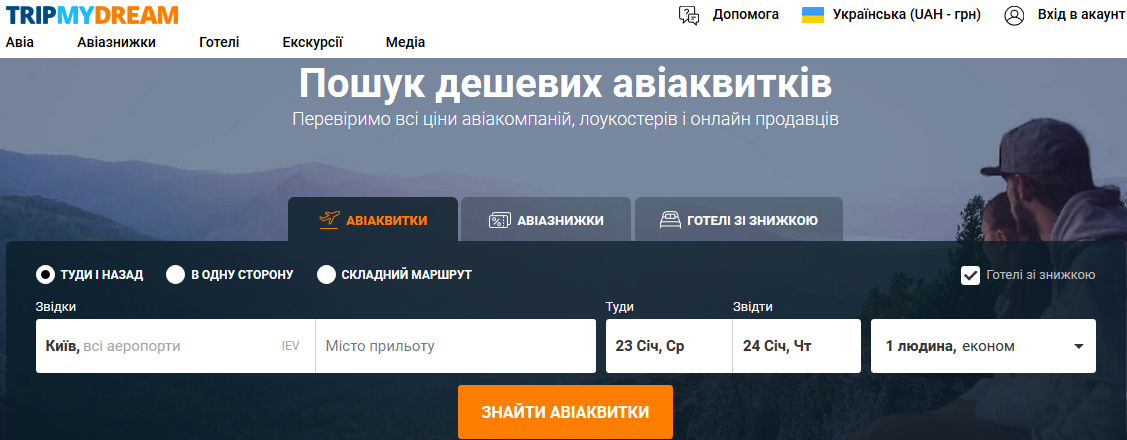


Рисунок 1.5 – Головне вікно TripMyDream

Сайт-агрегатор авіаквитків TripMyDream пропонує найдешевші авіаквитки на літак в 220 країн світу. Він здійснює пошук та порівняння ціни на авіаквитки серед кращих агентств та більш ніж 500 авіакомпаній [26].

На сайті можна знайти популярні бюджетні пропозиції на перельоти по всьому світу.

Також на сайті є багато рекомендацій, наприклад:

* + за найдешевшими авіалініями.

Дешево забронювати квитки на літак пропонують компанії-лоукостери, тому в першу чергу рекомендується ретельно переглянути їх авіарейси. Найпопулярніші серед них: Wizz Air, Fly Dubai, Aegean Air, German Wings, Pegasus Airlines, Air Baltic, Vueling Airlines, Air Arabia, Estonian Air і багато інших.

* + за пошуком дешевих авіаквитків.

Особливу увагу варто приділити акціям, які постійно пропонують перевізники. Знайти найдешевші квитки на літак можна переважно у міжсезоння, коли мало туристів. Весна та осінь — найкращий час для початку продажу квитків. І, до речі, найкращий час для екскурсій! Насамперед слід шукати авіаквитки на прямі рейси. Насамперед, це економить дорогу для мандрівника валюту – час. По-друге, у деяких випадках ціни на прямі рейси можуть дорівнювати маршрутам з пересадками або навіть дешевшими. Якщо всі квитки на прямий рейс розпродані або їх просто немає, зверніть увагу на рейси з пересадкою. У більшості випадків ціна квитка на такий рейс завжди буде нижчою, ніж на прямі рейси.

* + за підготовкою перед бронюванням квитка.

Перед бронюванням квитків на рейс з пересадкою рекомендується уточнити кілька питань про місто, в якому проходитиме проміжна посадка. Чи потрібна мені транзитна віза? чи можна встигнути з'їздити до центру та подивитися основні пам'ятки, якщо пересадка довга; Чи є в аеропорту магазини безмитної торгівлі Duty Free, де можна скоротити час. Якщо планується справжній Євротур, краще купити квитки на рейс із тривалою пересадкою, щоб повною мірою насолодитися транзитним містом, побачити основні пам'ятки та скуштувати місцеві страви. Врахуйте, що деякі авіакомпанії пропонують послугу stopover – безкоштовне розміщення в хорошому готелі та екскурсію.

В даному підрозділі розглянуто найбільш відомі онлайн системи купівлі, бронювання, замовлення квитків всіх видів транспорту.

## 1.2 Аналіз моделей представлення знань

Щоб перейти то етапу формалізації, де усі ключові поняття і відносини виражаються за допомогою деякої формальної мови, слід вибрати модель представлення знань і засоби, за допомогою яких буде створюватися експертна система [27, 28].

Для маніпулювання знаннями із реального світу за допомогою обчислювальної техніки використовують їх моделювання. При використанні моделі представлення знань враховуються такі фактори, як однорідність представлення, простота розуміння експертами та користувачами. Однорідне представлення приводить до спрощеного механізму управління знаннями та логічними висновками. Типовими моделями представлення знань є: продукційна, фреймова, модель семантичної мережі, логічна модель.

У продукційній моделі знання представлені сукупністю правил вигляду «ЯКЩО-ТО». Правило продукцій (продукція) складається з двох частин – «умова – дія» («антецедент – консеквент»), яка характеризує одну дозу знань, необхідних для вирішення задачі. Умовна частка правила – це приклад (шаблон), який означує, коли це правило може бути використане для вирішення деякого етапу завдання. Тобто ЯКЩО вказує «умови», «попередній», «посилка» або «підстава». Частина дія означує належний крок в рішенні задачі, таким чином частина ТО представляє відповідно «висновок», «наслідок» або «гіпотезу». Перераховані елементи другої частини генеруються при істинності умов першої частини.

У системах, заснованих на знаннях, існує чітка грань між наслідком, або гіпотезою, з одного боку, і висновком – з іншою. Наслідок, або гіпотеза, носить можливий характер, оскільки він отриманий на підставі фактів, які можуть бути як істинними, так і помилковими. Висновком є факт, отриманий шляхом виводу на підставі фактів, які вважаються за істинні.

Продукційну систему утворює база знань – це сукупність продукцій разом з механізмом їх керування.

Продукційна система забезпечує управління процесом рішення задачі за зразком і складається з трьох компонентів: набору продукційних правил (production rule), робочої пам'яті (working memory) і циклу управління «розпізнавання-дія» (рисунок 1.6).

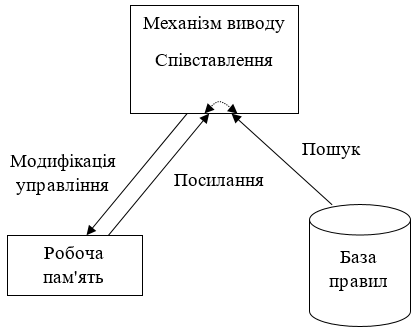


Рисунок 1.6 – Конфігурація системи продукцій

Цикл «розпізнавання-дія» (механізм логічного висновку). Структура продукційної системи управління проста: робоча пам'ять ініціалізована початковим описом завдання. Поточний стан розв'язання задачі представлений набором відліків у робочій пам'яті. Ці моделі порівнюються з умовами продукційних правил; який генерує підмножину правил виведення, так зване конфліктне безліч. Умови цих правил узгоджуються з моделями робочої пам'яті. Продукції, які у конкуруючій множині, називаються допустимими. Вибирається та активується один із продуктів набору конфліктів (дозвіл конфліктів). Активація правила означає виконання його дії. Це змінює вміст робочої пам'яті. Після того, як обране правило спрацювало, цикл управління повторюється для зміненої робочої пам'яті. Процес завершується, якщо вміст оперативної пам'яті не відповідає будь-якій умові.

Розглянемо визначення та класифікацію семантичних мереж. На думку фахівців зі штучного інтелекту, ця модель є найбільш загальним способом представлення знань. Модель семантичної мережі відображає сукупність об'єктів предметної області і відносин між ними. При цьому об'єктам відповідають вершини (або вузли) мережі, а відносинам з'єднуючі їх дуги. Як об'єкти можуть виступати події, процеси, дії узагальнені поняття або властивості суб'єктів, що використовуються для опису класів. Вершини мережі сполучаються дугою, якщо відповідні об'єкти предметної області знаходяться в якому-небудь відношенні, наприклад; <бути> (бути елементом класу), <мати>, <володіти властивістю> (мати значення властивості) і інші.

Семантичні мережі є найбільш широким класом мережних моделей, у яких поєднані різні типи зв’язків. Часткові випадки семантичних мереж відповідають спеціальним типам зв’язків. Виділяють, зокрема, класифікуючі мережі, функціональні мережі та сценарії.

У семантичних мережах використовуються чотири основні типи об'єктів: поняття, події, властивості і значення. Поняття представляють собою відомості про абстрактні або фізичні об'єкти предметної області. Вони можуть бути задані множиною доменів (параметрів або констант).

Події – це дії, які можуть перевести предметну область з поточного стану в деякий новий. Можна перевести предметну область в певний бажаний (цільовий) стан, поставити завдання, відшукати на семантичній мережі шлях подій, що приводять до цільового стану.

Властивості використовуються для характеристики (уточнення) понять, подій, процесів і т. д. Властивостями понять можуть бути колір, розміри, якість, а властивостями дій – час, тривалість, місце і т. д.

Значення – це значення з якої-небудь множини, які можуть приймати властивості.

Розглянемо фреймові моделі. Поняття фрейму було введене відомим американським вченим Марвіном Мінським в 1975 році. Ця теорія відноситься до психологічних понять і стосується способів розуміння сприйняття (бачимо, чуємо) явищ, процесів, об'єктів і т. п.

М. Мінський розглядав фрейм як структуру даних для представлення множини стереотипних ситуацій, подій і об'єктів, а також їх характеристик, ознак і властивостей. У основі теорії фреймів лежить сприйняття фактів за допомогою отриманої ззовні інформації про деяке явище з вже наявними даними, які накопичені дослідним шляхом або отримані в результаті обчислень. Коли людина потрапляє в нову ситуацію, вона викликає зі своєї пам'яті основну структуру, що називається фреймом. Фрейм (рамка) – це одиниця представлення знань, яка запам'яталася у минулому та деталі якої можуть бути змінені згідно поточної ситуації. Клас деяких об'єктів (процесів) може визначатися одним типовим (базовим) об'єктом, який включає найбільш істотні характеристики об'єктів даного класу. Так деяку характеристику об'єкту можна представити трійкою (Об'єкт, атрибут\_, значення\_). Тут кожна пара (атрибут, значення) є слотом. Неважко замітити, що фрейм містить різні (найбільш важливі) слоти, що характеризують даний об'єкт «постачання».

Зібравши всі трійки, що стосуються даного об'єкту, отримаємо об'єктне представлення області міркувань, щодо даного об'єкту. Спільна форма цього представлення така:

Об'єкт(атрибут\_ значення\_), = 1, ..., т.

Таким чином, замість побудови різних незалежних формул будуємо більш велику структуру повної інформації про об'єкт, яку називають фреймом. Якщо потрібна інформація про деякий об'єкт, то звертаються до відповідного фрейму, усередині якого знаходяться властивості і факти щодо даного об'єкту. Відмітимо, що об'єктне представлення можна отримати як з логічного, так і з інших представлень знань.

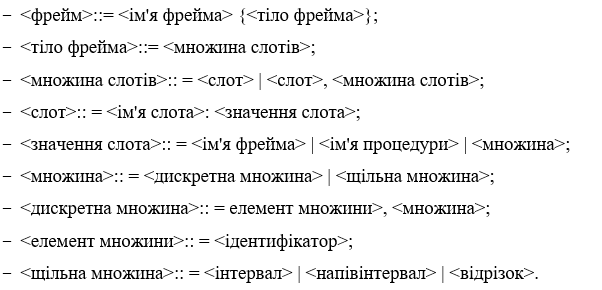
Перетворення інформації всередині фрейму і зв'язки його з іншими фреймами забезпечують як інформаційні, так і процедурні елементи. Елементами фрейму є так звані слоти, а значення їх – заповнювачі слотів.

Фрейм представлений певною структурою даних, складається з довільного числа слотів, серед яких є системні слоти і слоти, що визначаються користувачем. Кожен слот характеризується певною структурою і унікальним ім'ям усередині даного фрейму. Як системні можуть, наприклад, бути визначені наступні слоти: покажчик фрейму-батька, покажчик прямого дочірнього фрейму, користувач фрейму, дата визначення фрейму і його останньої модифікації, а також деякі інші. Системні слоти використовуються при редагуванні БД і управлінні виводом.

Фреймова система – це ієрархічна структура, вузли якої – подібні фрейми. Зупинимося на значеннях кожного елемента фреймворку:

* + ім'я фрейму – це ідентифікатор, який приписується фрейму. Фрейм повинен мати особливе ім'я, тобто єдине в даній системі фреймів. Кожен фрейм має довільне число слотів. Деякі з них завжди встановлюються системою для виконання певних функцій, а інші визначаються користувачем. До них відносяться: слот IS-A, що є фреймом-батьком цього фрейму, слот покажчиків дочірніх фреймів (це список покажчиків на ці фрейми), слот для введення імені користувача, дати визначення, дати внесення змін, тексту примітки і ін. слоти. Кожен слот, своєю чергою, також представляється певною структурою даних.
  + ім'я слота – це ідентифікатор, який присвоюється слоту. Слот, як і фрейм, повинен мати оригінальне ім'я у фреймі, якому він належить. Найчастіше ім'я слота не має змісту і є просто ідентифікатором даного слота, але в деяких випадках ім'я має специфічний зміст. До відомих імен належать: IS-A (відношення IS-A), DDESEN-DANTS (покажчик прямого дочірнього фрейму), FINEDBY (користувач, що визначає фрейм), DEFINEDON (дата започаткування фрейму), М001Р1Е0СЖ / дата модифікації фрейму), СОММЕNТ (коментар) і т. п. Також до специфічних відносяться імена, що використовуються для представлення структур: HASPART, RELATIONS та інші. Це є системні слоти, які використовуються при зміні бази знань і управлінні висновком.

Приведемо тепер точніше визначення фрейма в нотації Бекуса-Наура:



Переваги фреймових моделей:

* ефективність структурованого опису складних понять і рішення задач з різними способами висновку внаслідок різної виродженості родо-видових зв'язків;
* можливість економії робочої пам'яті комп'ютера, оскільки значення слотів зберігаються в єдиному примірнику і. включаються тільки в одну фрейм, що містить слот з даним ім'ям;
* допустимість обчислення значення будь-якого слота за допомогою відповідних процедур або евристично.

Недоліки:

* Відносно висока складність (трудомісткість внесення змін в родовидову ієрархію).
* Затруднені процеси обробки виключень.
* Складність управління завершеністю і постійністю цілісного образу знань.

Розглянемо логічну модель. В основі представлення знань цією моделлю лежить мова математичної логіки, що поєднує в собі мову висловлювань і обчислення предикатів і дозволяє формально описувати поняття предметної області і зв’язки між ними. Висловлювання – вираз, де стверджується, або заперечується наявність властивостей певного об’єкту. Предикат – це функція, визначена на будь-якій множині, але яка приймає лише два значення (істина, чи хибність), і призначення для вираження властивостей об’єктів та зв’язків між ними.

Найпростіша мова логіки – обчислення висловлювань (логіка нульового порядку, в якій відсутні змінні). Для іменування об’єктів предметної області використовують константи. Логічні висловлювання, або пропозиції утворюють атомарні формули. Висловлювання логічно виходить із заданих посилок і воно істинне тоді, коли істинні посилки.

Правило висновку називається не суперечним, якщо їх висновки логічно виходять з посилок. Обчислення висловлювань не є виразним засобом для обробки знань, бо в них не можуть бути представлені пропозиції, що включають в себе з мінні із кванторами. Розширення обчислень висловлювання є обчислення предикатів із кванторами (логіка першого порядку), в якому для вираження відношень між об’єктами використовуються не тільки константи, але й змінні.

Мова логіки першого порядку задається синтаксисом, що допускає 4 типи виразів: константи, змінні, функціональні імена і предикатні імена. Константи служать іменами індивідуумів (на відміну від імен сукупностей об’єктів, людей чи подій). Змінні позначають імена сукупностей. Предикатні імена задають правила з’єднань констант та змінних (процедури, операції, чи граматики).

До позитивних рис логічної моделі відносять:

* єдність теоретичного обґрунтування
* безпосереднє програмування логічних висновків і маніпулювання значеннями.

Логічна інтерпретація формальних граматик привела до виникнення мов логічного програмування, таких як Пролог.

Саме ця модель представлення даних була вибрана нами при написанні даної експертної системи.

## 1.3 Огляд та обґрунтування вибору мови програмування для експертної системи

В якості інструменту створення експертної системи було обрано – SWI-Prolog.

Зупинимося на історії мови Пролог. Перші повідомлення про неї з’явились на початку 70-их років. Пролог належить до класу логічних мов програмування. Основні ідеї до розробки були запропоновані Р. Ковальським та П. Хейсом. В 1973 році був розроблений перший інтерпретатор Прологу. Розробниками були французи з міста Марсель, які працювали під керівництвом А. Колмерое. Наступну версію (Единбурзька реалізація Прологу) виконав Д. Уореном на машині DEC-10. Ця версія перевела мову і разом з нею логічне програмування з теоретичної площини у площину практичного програмування. Після цієї розробки Пролог став добрим інструментом для рішення різноманітних задач штучного інтелекту. Про те, що мова Пролог має великі можливості свідчить і та подія, що японські вчені обрали його як опорну мову для виготовлення обчислювальних систем V-го покоління [29-32].

Пролог не є традиційною мовою. Для програмування необхідно мислити в термінах цілей. Традиційні мови детально описують варіанти розв’язків, набори визначень ситуацій і формулювань задач. Пролог описує структуру задачі, а не процедуру її рішення. В основі Прологу лежить набір програмних механізмів, який включає в себе: співставлення зразків, задавання структур даних типу дерева й автоматичне повернення. Пролог можна зарахувати до мов програмування, які ґрунтуються на описовому чи декларативному підході до програмування.

Виділяють два ступені характеристики програми Пролог: декларативний і процедурний. Декларативний визначає відношення і за їх допомогою, яким повинен бути результат роботи програми. Процедурний визначає яким чином цей результат було отримано і які відношення дійсно оброблялись. Велику частку процедурних деталей пролог-системи виконують самостійно, без допомоги програміста. Якщо програмісту не важливою є ефективність обчислень, то він може приділяти більше уваги декларативному аспекту програми, і не звертати увагу на організацію процесу обчислень, [33-36].

Спочатку Пролог був теоретичною мовою програмування і найбільш часто використовувався як інструмент для наукових досліджень. Довгий час американські вчені не бачили чи не хотіли бачити його переваг для розв’язання задач ШІ. І на той час мови штучного інтелекту ЛІСП і Мікропленнер були невдалими і реалізованими дуже неефективно. Та після створення швидких інтерпретаторів і компіляторів Пролог став найбільш вживаною мовою розв’язання задач штучного інтелекту, а також набув популярності у спеціалістів по реляційних базах даних, у програмній інженерії, при моделюванні знань, в експертних системах та ін.

В основу Прологу було взято з предикатних логік поняття відношення – предикат.  «предикат» використовується в розділі математичної логіки, в якому розглядаються операції над логічними висловленнями. У логіці предикатів під предикатом розуміють деяку властивість, логічну функцію від довільного числа аргументів, що приймає тільки два значення – «істина» (true) або ж «хибність» (false).

Логіка предикатів – спеціальний математичний апаратом формалізації людського мислення. За цієї причини мови програмування логічного типу являються найзручнішими при роботі з базами знань.

Основними елементами числення предикатів є:

* константні терми , , …;
* змінні терми , , …;
* функціональні терми , , …;
* предикатні букви , , …;
* логічні символи →, &, ∪, ~, ∀, ∃;
* спеціальне висловлювання .

Елементарне (просте) висловлення – це предикат та пов'язані з ним терміни. Складні оператори будуються із елементарних операторів з використанням логічних з'єднувачів. До них відносимо: «і» (and, &), «або» (or, ∪) , «ні» (not, ~) та імплікація (→). Остання читається «якщо… то…» і використовується для побудови конкретних правил.

У численні предикатів, для використання змінних термів, застосовується особлива структура – квантор.

Вони використовуються щоб вказати, наскільки значення змінних повинні бути істинними, щоб усе висловлення була істинним. Розрізняють квантори загальності (∀) та існування (∃).

У логіці предикатів елементарний об'єкт, що оцінюється як «істина», є атомарною формулою. До нього входять символи предиката і термів, які відіграють роль аргументів цього присудка. Загалом нотація предикату — це ім'я відношення, що існує між аргументами.

Атомарна формула записується в предикатний запис і має вигляд , де – це предикатний запис та члени .

Кількість термів в кожному предикаті фіксована і вона називається його арністю. Терми визначаються так:

* константний терм – це є терм;
* змінний терм – це є терм;
* якщо арність функціональної букви є *п*, а  – терми, тоді  теж є термом.

Правильно побудовані формули (ППФ) отримуються комбінацією атомарних формул при використанні логічних зв’язок.

Символом  позначають хибну формулу і визначають поняття «протиріччя». Наприклад, формула *А* → значитиме хибність *А* і її можна записати формулою *~А.*

Виділяють серед формул спеціальний клас, які називають тотожно істинними формулами (аксіомами). Приклад аксіоми:

~ ~ *А* ← → *А*.

Виділимо наступні переваги Прологу, врахувавши вище викладене:

* Пролог має чіткий математичний ґрунт, близький до мислення людини;
* використання на Пролозі числення предикатів, як єдиної мови специфікацій для опису вимог до програм і опису самої програми поєднує процес написання програми та її верифікацію;
* застосування відношення як опорного поняття Прологу дозволяє ефективно працювати з реляційними базами даних;
* організація обчислень на паралельному принципі дозволяє вільно й невимушено реалізовувати програми Прологу;
* Пролог підтримує обчислення, які ґрунтуються на пошуку зворотнім ланцюжком міркувань. Цей факт особливо використовується при побудові експертних систем;
* за допомогою «логічних змінних» Пролог зберігає перехідні результати обчислень, що дає змогу їх наступного використання. Ця властивість просто вирішує питання організації логічного виведення.

Зазначимо такі недоліки Прологу.

* через те, що порядок побудови виведення результату на Пролозі є «невидимим», то виникають труднощі в розумінні процесу виконання програми;
* неідеальними є засоби для виявлення екстралогічних особливостей, а саме для операторів динамічного приєднання та вилучення тверджень;
* відсутніми є досконалі засоби для розробки і настроювання величезних програм;
* засоби підтримки модульного принципу програмування є недостатніми.

## 1.4 Постановка задач дослідження

Метою кваліфікаційної роботи є програмна реалізація алгоритмів оптимізації туристичних маршрутів України.

Для досягнення поставленої мети необхідно:

* проаналізувати відомі онлайн пошукові системи для консультування туристів;
* провести аналіз моделей представлення знань;
* проаналізувати експертні системи;
* спроектувати структуру експертної системи;
* розробити модель представлення знань;
* сформувати базу знань;
* програмно реалізувати експертну систему;
* провести апробацію розробленого продукту.

Для побудови експертної системи оптимізації туристичних маршрутів України розроблено дерево рішень (рисунок 1.7).



Рисунок 1.7 – Дерево рішень побудови експертної системи

## 1.5 Висновки до розділу 1

Результатами першого розділу є:

* проведений аналіз відомих онлайн пошукових систем для вибору туристичних маршрутів;
* проведений аналіз моделей представлення знань;
* проведений огляд та зроблено обґрунтування вибору мови програмування для реалізації продукту.

Таким чином, на основі проведеного аналізу показано актуальність проведення наступних досліджень та здійснено постановку задач на наступні розділи.

# 2 РОЗРОБКА АРХІТЕКТУРИ ЕКСПЕРТНОЇ СИСТЕМИ

## 2.1 Етапи розробки експертної системи

В даний час визначилась технологія розробки експертних систем, що включає наступні етапи:

* ідентифікація (вибір предметної області);
* концептуалізація (аналіз предметної області, типів даних, зв’язки між ними);
* формалізація (для цього слід вибрати модель представлення знань і засоби створення експертної системи – мову програмування);
* виконання (написання програми, створення бази знань);
* тестування;
* експлуатація (рисунок 2.1).

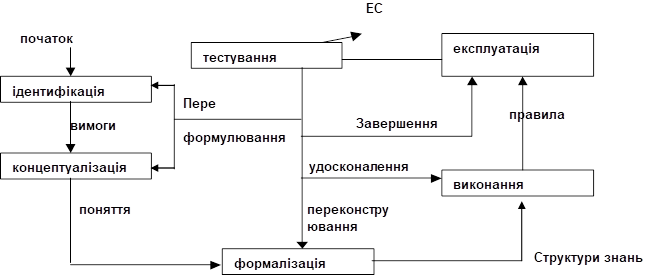


Рисунок 2.1 – Методика (етапи) розробки ЕС

Найважливішими етапами розробки будь-якої ЕС є наповнення бази знань фактами, побудова алгоритму будування висновків на основі фактів, а також діалог з експертом або користувачем даної системи [27, 37-46].

## 2.2 Структура експертної системи

Структура розробленої експертної системи (рисунок 2.2) складається із наступних блоків [47-50]:

* блок бази даних, що містить інформацію напрямки руху поїздів по ст. Тернопіль та інформацію про маршрути Тернопільського автовокзалу;
* блок прийняття рішень (логічного виводу), що містить блок знаходження результату і блок обробки відповідей;
* блок виводу запитань для вибору маршруту;
* блок виводу результату пошуку з усіма характеристиками.

Останні два становлять разом блок взаємодії з користувачем.

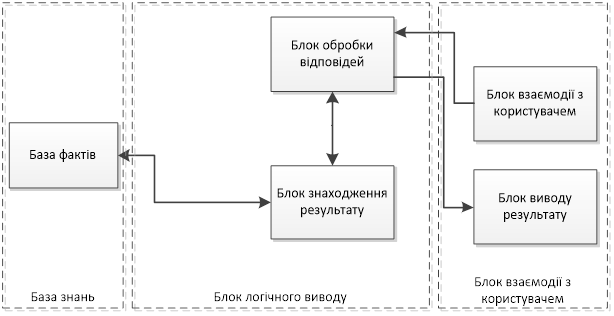


Рисунок 2.2 – Структура експертної системи

Для розробки експертної системи було обрано мову Prolog.

В базі даних (фактів) маршрути залізницею описані за допомогою наступних характеристик: початковий пункт відправлення; кінцевий пункт прибуття; номер поїзда; список днів, в які є маршрут; весь час в дорозі поїзда; години прибуття і відправлення з Тернополя; список пунктів, через які пролягає маршрут з часами відправлення та часами прибуття.

Другий тип вміщує в себе значення характеристик автобусного маршруту, а список – список пунктів з їх характеристиками, через які пролягає маршрут.

В базі даних (фактів) авіаперевезень рейси описані за допомогою наступних характеристик: початковий пункт вильоту; кінцевий пункт прильоту; час вильоту; час прильоту; номер рейсу; компанія, що здійснює переліт; тип повітряного судна; список днів, в які є маршрут або атом alldays, якщо рейс здійснюється кожного дня тижня.

Дані записи – це складені терми.

2.2.1 База даних

Кожен факт бази даних представляє собою складений терм (таблиця 2.1). Арність предиката *train* та *autobus* дорівнює 9. Список *List* – це список пунктів з їх характеристиками (час приїзду, час відправлення), через які проходить маршрут, інший список *List* – це список транзитних пунктів з їх характеристиками (час прибуття і вибуття з пункту приїзду, відстань з Тернополя до пункту, вартість проїзду). Арність предиката *flight* дорівнює 3. Список *Listflight* – це список характеристик рейсу (конкретизовано в таблиці 2.1).

Таблиця 2.1 – Типи записів даних в БД

|  |  |
| --- | --- |
| Тип запису | Вигляд запису |
| 1 | *train*(*P*1*, P*2*, G, А, B,'Тернопіль',Time1,Time2, List*)*.* |
| 2 | *List*=[*H*11, *C*11:*С*21, *E*11:*Е*21, …, *H*1*n*, *C*1*n*:*С*2*n*, *E*1*n*:*Е*2*n*] |
| 3 | *autobus*(*bus\_tour*(*time*1(*C*1:*С*2), *P*1, *P*2, *time*2(*E*1:*Е*2), *B*1, *A*, *G*, *B*2, *List1*)) |
| 4 | *List*=[*station*(*H*1, *time*1(*H*2), *time*2(*H*3), *H*4, *H*5), …, *station*(*H*6, *time*1(*H*7), *time*2(*H*8), *H*9, *H*10)] |
| 5 | *flight*(*Point*1, *Point*2, *Listflight*)*.* |
| 6 | *Listflight* = [*Time*1, *Time*2, *Number*, *Company*, *Typeofaircraft*, *Fare*, *Listofdays*] |

Змінні запису 1-го типу мають наступні значення:

*Р*1 і *Р*2 – початковий і кінцевий пункти відправлення маршруту відповідно;

*G* – номер маршруту;

*А* – дні проходження маршруту. Змінна *А* може набувати значення: '*по непарних*', '*по парних*', '*щоденно*';

*В* – повний час маршруту в дорозі (від початкового до кінцевого пунктів);

*'Тернопіль'* – цей атом присутній у всіх фактах. Він вказує на те, що маршрут пролягає через м. Тернопіль;

*Time1* – час приїзду в Тернопіль;

*Time2* – час відправлення з Тернополя.

В даних запису 2-го типу (список *List*) змінні мають наступні значення:

*H*1*і*– назва *і*-го транзитного пункту;

*C*1*n*:*С*2*n* – час прибуття в *і*-й пункт у форматі ГОД:ХВ;

*E*1*n*:*Е*2*n* – час відправлення з *і*-го пункту у форматі ГОД:ХВ.

Для даних запису 3-го типу змінні мають наступні значення:

*time*1(*C*1:*С*2) – предикат, що задає час відправлення рейсу з Тернополя,   
*С*1 – година, *С*2 – хвилини;

*Р*1 і *Р*2 – початковий і кінцевий пункти відправлення рейсу відповідно;

*time*2(*E*1:*Е*2) – предикат, що задає час прибуття автобуса в пункт *Р*2, *Е*1 – година, *Е*2 – хвилини;

*В*1 – вартість проїзду;

*А* – марка автобуса;

*G* – номер рейсу;

*B*2 – довжина шляху;

*List* – список пунктів з їх характеристиками, через які пролягає маршрут.

Для даних запису 4-го типу змінні мають наступні значення:

*station*(*H*1, *time*1(*H*2), time2(*H*3), *H*4, *H*5) – це предикат, що описує транзитний пункт:

*H*1 – назва пункту, *time*1(*H*2) – предикат часу прибуття в пункт, *time*2(*H*3) – предикат часу вибуття з пункту, *H*4 – відстань в кілометрах з Тернополя до пункту *H*1, *H*5 – вартість проїзду з Тернополя до пункту *H*1. Час *H*2 і *H*3 теж задається у форматі ГОД:ХВ.

Аналогічно предикат station(*H*6, *time*1(*H*7), *time*2(*H*8), *H*9, *H*10) описує наступний пункт.

Змінні запису 5-го типу мають наступні значення:

*Point*1 – початковий пункт вильоту;

*Point*2 – кінцевий пункт прильоту.

Конкретизуємо список *Listflight*.

*Time*1 – час вильоту;

*Time*2 – час прильоту;

*Number* – номер рейсу;

*Company* – компанія, що здійснює переліт;

*Typeofaircraft* – тип повітряного судна;

*Listofdays* – список днів (запис 6 типу), в які є маршрут або атом *alldays*, якщо рейс здійснюється кожного дня тижня.

Час вильоту і прильоту задається у форматі ГОД:ХВ.

Один факт 5-го типу згідно приведеної структури буде наступним:

flight('ЛЬВІВ', 'СТАМБУЛ', [time1(22:10), time2(00:10), 'THY0444', 'Turkish\_Airlines', 'A320', [1,2,4,5,7]]).

2.2.2 База знань

База знань повинна містити відомості, які відносяться до предметної області, в тому числі такі інформаційні компоненти, як прості факти про цю предметну область, правила або обмеження, які описують відношення, або можливо також методи, евристики та ідеї для вирішення задач в даній області [28].

Документації, інструкції, статті технічного забезпечення зберігаються в простих базах знань. Основна мета утворення таких баз – допомога менш досвідченим людям знайти існуючий опис того, як вирішити проблему предметної області.

Однією з основних проблем, характерних для систем збереження знань, є проблема подання знань. Це пояснюється тим, що форма подання знань впливає на характеристики й властивості системи.

Бази знань у Пролозі описуються у вигляді фактичних фактів та правил логічного висновку про базу даних та процедури обробки інформації, які представляють інформацію та знання про предметну область. У відповідь на прості запити до баз знань система логічного програмування Пролог повертає «істина» та «хибність» значення залежно від наявності відповідних фактів [31].

В підрозділі 1.2 кваліфікаційної роботи розписані типи знань. Рисунок 2.3 підсумовує відомості про подання знань.

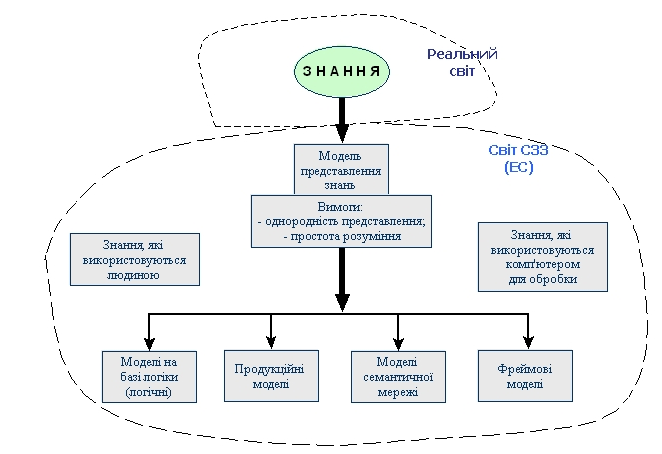


Рисунок 2.3 – Подання знань

В кваліфікаційній роботі мною використана продукційна модель представлення знань, тобто сукупність правил вигляду "ЯКЩО-ТО".

На основі зібраної інформації за маршрутами залізничного вокзалу Тернополя в кваліфікаційній роботі реалізовуємо здійснення пошуку за всією базою поїздів, що проходять через станцію Тернопіль; всіма зупинками, в які можна доїхати без пересадки; за номером поїзда; за маршрутами мінімального і максимального часу; маршрутами за певною годиною та ін. (таблиця 2.2).

Таблиця 2.2 – Перелік процедур пошуку маршруту поїздом

|  |  |  |
| --- | --- | --- |
| Класифікація | Назва правила | Результати |
| за всією базою поїздів | passenger\_train | усі поїзди |
| за кінцевими зупинками поїздів | passenger\_train2 | усі міста, які є кінцевими зупинками |
| за номером поїзда | passenger\_train3 | всі характеристики поїзда |
| за поїздами з Тернополя в деяке місто | passenger\_train4 | перелік поїздів з тривалістю руху |
| за поїздом з Тернополя в деяке місто з мінімаль-ною тривалістю маршруту | passenger\_train5 | всі характеристики поїзда |
| за поїздом з відправкою в визначений час | passenger\_train6 | перелік поїздів з тривалістю руху |
| за поїздом з Тернополя в деяке місто з максималь-ною тривалістю маршруту | passenger\_train7 | всі характеристики поїзда |

Також на основі зібраної інформації по автобусних рейсах, рекомендацій експертів в даній роботі реалізовуємо здійснення пошуку за всією базою рейсів з Тернополя; всіма містами, які є кінцевими зупинками, прямими і транзитними містами; рейсами за; рейсами за мінімальною ціною; рейсами мінімальної довжини; рейсами за певною годиною і ін. Перелік процедур наведено в таблиці 2.3.

Таблиця 2.2 – Перелік процедурпошуку маршруту автобусом

|  |  |  |
| --- | --- | --- |
| Класифікація | Назва правила | Результати |
| за всією базою рейсів | ternopil | усі рейси |
| за містами, які є кінцевими зупинками | priamo | усі міста, які є кінцевими зупинками |
| за рейсами з Тернополя в деяке місто | ternopil3 | всі характеристики рейсів |
| за номерами рейсів | tern2 | всі характеристики рейсу |
| за рейсами мінімальної довжини | ternopil4 | всі характеристики рейсів |
| за рейсами мінімальної ціни | ternopil2 | всі характеристики рейсів |
| рейсами за певною годиною | ternopil5 | всі характеристики рейсів |

На основі зібраної інформації за рейсами даних міжнародного аеропорту «Львів» імені Данила Галицького і Віденського міжнародного аеропорту «Швехат» реалізовано здійснення пошуку за всіма рейсами міжнародного аеропорту «Львів» імені Данила Галицького, за частиною рейсів Віденського міжнародного аеропорту «Швехат», пошук рейсів прямих і з пересадкою через відповідні аеропорти в деяке місто; всіма містами, в які можна долетіти без пересадки; за номером рейсу; за рейсами у певний день тощо. Діаграма блоків (процедур) по ієрархії їх викликів зображена на рисунку 2.4. Перелік процедур наведено в таблиці 2.4.

*add\_way*

*dialog*

*lemberg*

*viden*

*lemberg4*

*viden4*

*reys*

*viden2*

*lemberg1*

*viden1*

*lemberg2*

*lemberg3*

*deny*

*deny*

*deny*

*deny*

*lemb2*

*lemb*

Рисунок 2.2 – Діаграма блоків (процедур) за ієрархією їх викликів

Таблиця 2.4 – Перелік процедурпошуку маршруту літаком

|  |  |  |
| --- | --- | --- |
| Класифікація | Назва правила | Результати |
| за всією базою авіарейсів зі Львова | lemberg | усі авіарейси зі Львова |
| за всією базою авіарейсів з Відня | viden | усі авіарейси з Відня |
| за всією базою міст зі Львова | lemberg4 | усі міста, в які можна долетіти зі Львова |
| за всією базою міст з Відня | viden4 | усі міста, в які можна долетіти з Відня |
| за номером рейсу | reys | всі дані про рейс |
| за всіма рейсами з Відня до Львова з пересадкою | viden2 | усі авіарейси з Відня до Львова з пересадкою |
| за прямими рейсами зі Львова в деяке місто | lemberg1 | всі дані про рейси |
| за рейсами з Відня в деяке місто | viden1 | всі дані про рейси |
| за рейсами зі Львова в деяке місто з пересадкою | lemberg2 | всі дані про рейси |
| за прямими і з пересадкою рейсами зі Львова в деяке місто | lemberg3 | всі дані про рейси |

Всі характеристики маршруту поїздом включають в себе:

* номер поїзда;
* назву маршруту;
* час відправлення;
* дні, у які він курсує;
* час прибуття;
* усі станції, через які він проходить, з годинами прибуття і відправлення.

Всі характеристики автобусного рейсу включають в себе:

– номер рейсу;

– час відправлення;

– час прибуття;

– вартість проїзду;

– довжину маршруту.

Всі дані про авіарейс включають в себе:

* номер рейсу;
* компанія перевізник;
* час відправлення;
* дні, у які є літак;
* час прибуття;
* тип повітряного судна;
* початковий і кінцевий пункт.

Процедура *clearDatabase* є блоком очистки бази даних.

2.2.3 Блок логічного виводу і взаємодії з користувачем

Машина логічного виводу повинна володіти здатністю активно використовувати знання, представлені у базі знань.

Задаючи характеристики потрібного маршруту, експертна система може не знайти потрібний. Для цього створений блок, який показує важливість кожного з параметрів у вигляді коефіцієнтів. Спираючись на цей блок, експертна система обирає найбільш найкращий маршрут, який обирається за важливістю коефіцієнтів.

Процедура *vybirka*(*D*1) є блоком обробки відповідей і передає отримані відповіді у блок знаходження результату *min\_mark*.

За допомогою блоку виведення результату, яка реалізована у вигляді процедур *passenger\_train*, виводиться найбільш оптимальна відповідь.

Блок взаємодії з користувачем працює наступним чином. Він по черзі, питається користувача кожну характеристику. Користувач, в свою чергу, відповідає на відповіді, вводячи номер відповіді. Потім блок взаємодії передає відповіді до блоку логічного виводу, який проганяє відповіді по базі даних.

## 2.3 Алгоритм роботи експертної системи

Взаємодія користувача працює наступним чином. Спочатку надходить питання до користувача по кожній з характеристик, а користувач, в свою чергу, відповідає на відповіді.

Експертна система допомагає вибрати найбільш оптимальний маршрут. Але якщо задані характеристики не співпадають з заданими характеристиками, то експертна система виводить на монітор результат «Поїзда немає», «В дане місто не існує маршруту», «Такий рейс не існує», «В дане місто не існує рейсу» або «false». Таким чином експертна система дає відповіді як при знаходженні потрібного варіанту так і при його відсутності.

Блок-схеми алгоритму роботи експертної системи приведені на   
рисунку 2.5 (подорож залізницею), рисунку 2.6 (подорож автобусом), та рисунку 2.7 (подорож літаком).

## 2.4 Висновки до розділу 2

В даному розділі описано технологію розробки експертних систем, розроблено структуру експертної системи, описано базу даних, базу знань, діаграму блоків (процедур) за ієрархією викликів, приведено алгоритм роботи експертної системи.

Рисунок 2.5 – Блок схема алгоритму маршруту поїздом

Початок

Вибір меню

Вивід усіх міст

(кінцевих зупинок)

*і*=3

так

Вивід усіх поїздів (ч/з Тернопіль)

*і*=1

ні

Номер є в базі?

Вивід характеристик маршруту

так

ні

А

так

*і*=2

так

ні

ні

Ввід номера поїзда

Продовження рисунку 2.5

так

так

*і*=4

ні

Місто є в базі?

Вивід характеристик маршруту

ні

Б

А

так

так

*і*=5

ні

Місто є в базі?

Формування БД маршрутів в місто

ні

Вивід з БД мінімального за часом маршруту

Ввід міста

Ввід міста

Продовження рисунку 2.5

ні

так

так

так

*і*=6

ні

Місто є в базі?

Вивід характеристик маршруту

ні

Б

Кінець

Час задо-вольняє ?

так

*і*=7

так

Місто є в базі?

ні

Формування БД маршрутів в місто

Вивід з БД максимального за часом маршруту

ні

Ввід міста

Ввід міста і часу

Рисунок 2.5 – Блок схема алгоритму подорожі автобусом

Рисунок 2.5 – Блок схема алгоритму маршруту автобусом

Початок

Вибір меню

Вивід усіх міст

(кінц. зуп.)

*і*=3

*і*=4

так

Вивід усієї бази рейсів

*і*=1

ні

Ввід міста

Місто є в базі?

Вивід характеристик рейсів

так

ні

А

так

*і*=2

так

так

ні

ні

ні

Вивід міст, для яких Тернопіль - транзитне

Рисунок 2.6 – Блок схема алгоритму маршруту автобусом

Продовження рисунку 2.6

так

так

*і*=5

ні

Місто є в базі?

Вивід транзитних рейсів

ні

Б

А

так

так

*і*=6

ні

Місто є в базі?

Вивід прямих і транзитних рейсів

ні

Ввід міста

Ввід міста

Продовження рисунку 2.6

так

так

*і*=7

ні

Рейс є в базі?

Вивід характеристик рейсу

ні

Б

Вивід рейсу мінімальної довжини

*і*=10

так

Вивід рейсу мінімальної ціни

*і*=8

ні

так

*і*=9

так

ні

ні

Вивід рейсу для визначеної години

Кінець

Ввід номеру рейсу

Початок

Вибір меню

Вивід усіх рейсів з Відня

*і*=3

так

Вивід усіх рейсів зі Львова

*і*=1

ні

А

так

*і*=2

так

ні

ні

Вивід усіх міст, в які рейси є зі Львова

*і*=4

так

Вивід усіх міст, в які рейси є з Відня

ні

Рисунок 2.7 – Блок схема алгоритму маршруту літаком

Продовження рисунку 2.7

так

так

*і*=5

ні

Рейс є в базі?

Вивід характеристик рейсу

ні

Б

А

так

*і*=6

ні

Ввід номера рейсу

Вивід усіх рейсів з Відня до Львова з пересадкою

Формування пар рейсів з Відня до Львова

так

так

*і*=7

ні

Місто є в базі?

ні

Ввід міста

Вивід характери-стик прямих рейсів зі Львова

Продовження рисунку 2.7

ні

так

так

*і*=8

ні

Вивід характери-стик рейсів з пересадками зі Львова

Б

Кінець

так

*і*=9

ні

Ввід міста

Ввід міста

так

Місто є в базі?

Вивід характери-стик прямих рейсів з Відня

ні

Місто є в базі?

так

*і*=10

Ввід міста

ні

так

Вивід характери-стик прямих і з пересадками рейсів зі Львова

ні

Місто є в базі?

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ЕКСПЕРТНОЇ СИСТЕМИ

## 3.1 Технічні вимоги до функціонування експертної системи

Розроблена експертна система коректно працює на всіх операційних системах сімейства Windows починаючи від версії Windows XP і вище.

Експертна система розроблена на мові Prolog в середовищі SWI Prolog.

Середній час безвідмовної роботи має складати не менше 5000 годин.

В експертній системі реалізований механізм збирання та виведення інформації.

Діалог з користувачем відбувається у режимі реального часу.

ЕС була розроблена на операційній системі Windows 10 64-x. Об’єм оперативної пам’яті комп’ютера складає 8 Гб. Процесор Intel Core –i3 фірми Intel.

Мінімальні вимоги до апаратного забезпечення, яке необхідне для коректної роботи системи наведено в таблиці 3.1.

Таблиця 3.1 – Мінімальні вимоги до апаратного забезпечення

|  |  |
| --- | --- |
| Характеристика | Параметри |
| Тактова частота процесора | 1200 Гц |
| Обсяг оперативної пам'яті | 2048 Мб |
| Обсяг вільного дискового простору | 150 Мб |
| Роздільна здатність монітора | 1024х768 |
| Операційна система | Windows |
| Периферійні пристрої | Принтер (при необхідності) |
| Додаткові пристрої | CD, DVD-ROM |

Повний лістинг програми приведений в додатку А.

## 3.2 Програмна реалізація бази знань та даних

Блок інтерфейсу користувача та блок прийняття рішення в коді програми тісно взаємопов’язані і для початку роботи необхідно викликати функцію *consulter*, яка виконує логічний зв'язок цих блоків. Дальше турист має вибрати вид транспорту, яким він хоче подорожувати.

## 3.2.1 Програмна реалізація бази знань та даних для подорожі поїздом

База даних спроектованої системи містить дані про поїзди, що проходять через станцію залізничну Тернопіль. Вона включає інформацію про 40 поїздів далекого прямування: початковий пункт, кінцевий пункт, номер поїзда, дні відправки, загальний час у дорозі, атом 'Тернопіль', час прибуття в Тернопіль, час відправлення з Тернополя, список станцій через які проходить поїзд з годинами прибуття і відправлення. Для кінцевого пункту цей час співпадає. БД задано предикатом *train*/9. Наприклад, один факт є наступним:

train('Львів','Новоолексіївка',86,'по парних', 24:05, 'Тернопіль', 13:24, 13:29,['Підволочиськ',14:20,14:22, 'Хмельницький', 15:23,15:28, 'Жмеринка',16:53,17:13, 'Вінниця',17:57,18:03, 'Козятин',18:55,18:57,'Біла Церква', 20:46, 20:48, 'Миронівка', 21:56, 21:58, 'Корсунь', 22:27,22:29, 'Цвіткове',23:04,23:06, 'Ім.Tараса Шевченка', 23:36,00:26, 'Фундукліївка',00:58,01:00, 'Знам''янка', 01:45,01:48, 'Олександрія',02:25,02:27, 'П''ятихатки', 03:42,03:44, 'Верхівцеве',04:15,04:17, 'Кам''янське', 04:42, 04:44, 'Дніпро', 05:18,05:43, 'Синельникове', 07:02, 07:04, 'Запоріжжя',07:57,08:02, 'Мелітополь', 09:44, 10:04, 'Якимівка',10:28,10:30, 'Партизани', 11:02,11:04, 'Новоолексіївка',11:15,11:15]).

База знань є внутрішньою, тобто вбудованою в нашу програму.

Програма містить безаргументні предикати: *passenger\_train*, *passenger\_train*2, *passenger\_train*3, *passenger\_train*4, *passenger\_train*5, *passenger\_train*6, *passenger\_train*7.

Предикат *passenger\_train* виводить всі поїзди, що проходять через ст. Тернопіль:

passenger\_train:-writeln('Через ст. Тернопіль проходять наступні поїзди: '), (train(A, B,C,E,\_,'Тернопіль',T1,T2,\_)),write('Поїзд №'),write(C), write(' '), write(A), write('-'), write(B), write(' Прибуття о '), write(T1),write(' год. '),write(' Відправлення о '),write(T2),write(' год. '),write(' у дні: '),write(E),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl,fail.

Предикат *passenger\_train*2 виводить усі міста, що є кінцевими зупинками:

passenger\_train2:-write('Усі міста, що є кінцевими зупинками '), nl,findall(B,(train(\_,B,\_,\_,\_,\_,\_,\_,\_)),List),write(List),fail.

Предикат *passenger\_train*3 видає інформацію про поїзд за номером, в тому числі і прохідні станції з годинами прибуття і відправлення:

passenger\_train3:-writeln('Введіть номер поїзда яким плануєте подорож з Тернополя : '), read(C), train(A,B,C,D,\_,\_,\_,G1:G2,List), write('Поїзд №'),write(C), write(' '), write(A), write('-'), write(B), write(' Відправлення о '), write(G1),write(' год. '),write(G2), write(' хв., '),write(D), nl,write(' Проходить через наступні станції з годинами прибуття і відправлення: '),nl,print\_list(List),nl, write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl,fail.

Предикат *passenger\_train*4 формує список поїздів, якими можна потрапити з Тернополя в деяке місто. Якщо прямого поїзда через Тернопіль не існує, тоді система видає інформацію: «Поїзда немає».

passenger\_train4:-writeln('Введіть місто до якого плануєте подорож з Тернополя в одинарних лапках: '), read(D1), train(A,B,C,D,\_,\_,\_,G1:G2,List),member(D1,List),append(\_,[D1,E1:E2|\_],List),write('Поїзд №'),write(C), write(' '), write(A), write('-'), write(B), write(' Відправлення о '),write(G1),write(' год. '),write(G2), write(' хв., '),write(D),nl,write('Прибуття у '),write(D1), write(' о '),write(E1), write(' год.'), write(E2), write(' хв. '),write(' Час перебування у дорозі: '),((G1=<E1,R is (E1-G1)\*60+E2-G2,S is R//60,T is R mod 60);(G1>E1,R is (E1-G1)\*60+E2-G2,S is R//60+24,T is R mod 60)), write(S),write(' год. '),write(T), write(' хв.'),nl, write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_') ; write('Поїзда немає '), nl,fail.

Предикати *passenger\_train*5 і *passenger\_train*7 (див. додаток А) здійснюють пошук поїзда, яким можна потрапити в деяке місто за мінімальний і максимальний час. В даних процедурах є перепосилання на процедури пошуку мінімального і максимального часу. Крім цього формуються динамічні бази *sity*(*A*,*B*,*C*,*D*,*X*) і *sity*2(*A*,*B*,*C*,*D*,*X*), в яких зберігається тимчасова інформація за поїздами в деяке місто.

passenger\_train5:-retractall(sity(\_,\_,\_,\_,\_)),writeln('Введіть місто для якого шукаєте з Тернополя маршрут за мінімальний час: '), read(D1),vybirka(D1), min\_mark1(X),sity(A,B,C,D,X), S is X//60,T is X mod 60,writeln('Мінімальний час у дорозі : '),write(S),write(' год. '),write(T), write(' хв.'),write('Поїзд №'),write(C), write(' '), write(A), write('-'), write(B), write(' '),write(D),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'); write('Поїзда немає '),nl, fail.

Предикат *passenger\_train*6 формує формує список поїздів, якими можна потрапити з Тернополя в деяке місто після певної години протягом однієї доби.

passenger\_train6:-writeln('Введіть місто до якого плануєте подорож з Тернополя в одинарних лапках: '), read(D1), writeln('Введіть годину після якої плануєте маршрут: '), read(G), G@=<G1, train(A,B,C,D,\_,\_,\_,G1:G2,List),member(D1,List),append(\_,[D1,E1:E2|\_],List),write('Поїзд №'),write(C), write(' '), write(A), write('-'), write(B), write(' Відправлення о '),write(G1),write(' год. '),write(G2), write(' хв., '),write(D),nl,write('Прибуття у '),write(D1), write(' о '),write(E1), write(' год.'), write(E2), write(' хв. '), write(' Час перебування у дорозі: '),((G1=<E1,R is (E1-G1)\*60+E2-G2,S is R//60,T is R mod 60);(G1>E1,R is (E1-G1)\*60+E2-G2,S is R//60+24,T is R mod 60)), write(S),write(' год. '),write(T), write(' хв.'),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'); write('Поїзда немає '),nl,fail.

В системі передбачені процедури очищення всіх фактів (*clear\_databaze*) і додавання нових фактів до динамічної процедури *train*/9:

add\_train:-

write('Введіть початковий пункт поїзда: '), read(P1),

write('Введіть кінцевий пункт поїзда: '), read(P2),

write('Введіть номер поїзда: '), read(C),

write('Введіть дні руху по ст. Тернопіль (наприклад: по парних, по непарних, щоденно'), read(E),

write('Введіть тривалість усього маршруту поїзда: '), read(G),

write('Введіть час прибуття в Тернопіль: '), read(A),

write('Введіть час відбуття з Тернополя: '), read(B1),

write('Введіть пункти проїзду у вигляді списку вигляду СТАНЦІЯ, Час прибуття, Час відправлення: '), read(H1),

assert((train(P1, P2,C,E,G,'Тернопіль',A,B1,H1))) ,

writeln('Записано в БД').

add\_train.

В цьому пункті йде пряма взаємодія з файлом БД, при вводі даних аналізується кожен з пунктів для введення, покроково, і коли введена інформація вже містяться в БД, то інформація про них відкидається і БД залишається без змін. В протилежному випадку проходить весь процес-діалогу (відповідь на питання поставлені системою), результатом цього процесу є додавання нового, оригінального запису в БД, з предикатом *train*. В процесі-діалозі потрібно дати відповіді на всі питання які є важливі, і надалі будуть використовуватись при запиті до системи.

## 3.2.2 Блок інтерфейсу користувача та блок прийняття рішення при подорожі автобусом

Блок інтерфейсу користувача та блок прийняття рішення в коді програми тісно взаємопов’язані і для початку роботи необхідно викликати функцію *consulter*, яка виконує логічний зв'язок цих блоків. Під час запуску даної функції вона розпочне послідовне виконання наступних функцій: *ternopil*, *priamo*, *tranzit\_city*, *tern*, *tern*1, *te*rnopil3, *tern*2, *ternopil*2, *ternopil*4, *ternopil*5. Умовно дані функції можна віднести у такі блоки: перша, друга, третя, восьма та дев’ята функція представляють інтерфейс користувача, а четверта, п’ята, шоста, сьома та десята – блок прийняття рішення.

consulter:-write('Виберіть дію для роботи з системою:'),nl, write('Подивитися всю базу рейсів з Тернополя - виберіть 1'),nl, write('Подивитися всі міста, які є кінцевою зупинкою, в які є прямі рейси з Тернополя - виберіть 2'),nl, write('Подивитися всі міста, для яких Тернопіль є транзитним містом - виберіть 3'),nl, write('Якщо ви шукаєте прямий рейс з Тернополя в деяке місто - виберіть 4'),nl,write('Якщо ви шукаєте транзитний рейс з Тернополя в деяке місто - виберіть 5'),nl,write('Якщо ви шукаєте будь-який рейс (прямий або транзитний)з Тернополя в деяке місто - виберіть 6'),nl, write('Якщо ви шукаєте рейс за номером - виберіть 7'),nl,write('Якщо ви шукаєте рейс за мінімальною ціною - виберіть 8'),nl,write('Якщо ви шукаєте рейс мінімальної довжини - виберіть 9'),nl,write('Якщо ви шукаєте рейс за годиною - виберіть 10'),nl,read(N),((N=1,ternopil);(N=2,priamo);(N=3,tranzit\_city);(N=4,tern);(N=5,tern1);(N=6,ternopil3);(N=7,tern2);(N=8,ternopil2);(N=9,ternopil4);(N=10,ternopil5)).

Розглянемо детальніше дані функції.

Процедура *ternopil* проводить пошук усіх рейсів з Тернополя, що є в базі даних.

ternopil:-writeln('Усі рейси з Тернополя: '), autobus(bus\_tour(time1(C1:C2), P1, P2, time2(E1:E2), \_, A, G, \_, \_)), write('Рейс №'),write(G), write(' '), write(P1), write('-'), write(P2), write(' Відправлення '),write(C1),write(' год. '),write(C2), write(' хв. '),write('Прибуття '),write(E1), write(' год.'), write(E2), write(' хв.'),write(' Марка автобуса: '),write(A),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl,fail.

Процедура *priamo* виводить усі міста, які є кінцевою зупинкою, в які є прямі рейси з Тернополя.

За допомогою процедури *tranzit\_city* можна побачити всі міста з БД, для яких Тернопіль є транзитним містом.

Процедура *tern* проводить пошук рейсів з Тернополя за певним містом. Користувачу необхідно ввести місто до якого він планує подорож прописними літерами і в одинарних лапках.

*tern*1, як і попередня процедура, проводить пошук рейсів з Тернополя за певним містом, але тільки транзитні рейси.

Наступна процедура – *ternopil*3 поєднує дві попередні: здійснює пошук прямих і транзитних рейсів з Тернополя. *ternopil*3 в свою чергу запускає допоміжні функції *ter* і *ter*1, які на свій вхід отримують номер запитання, яке необхідно задати користувачу та отримують відповідь.

ternopil3:-writeln('Введіть місто до якого плануєте подорож з Тернополя прописними літерами і в одинарних лапках: '), read(I),ter(I),ter1(I), nl,fail.

ternopil3.

ter(I):-autobus(bus\_tour(time1(C1:C2), P1, I, time2(E1:E2), B1, \_, G, B2,\_)),write('Рейс №'),write(G), write(' '), write(P1), write('-'), write(I), write(' Відправлення '),write(C1),write(' год. '),write(C2), write(' хв. '),write('Прибуття '),write(E1), write(' год.'), write(E2), write(' хв.'),write(' Вартість проїзду: '),write(B1),write(' Довжина маршруту: '),write(B2),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl.

ter.

ter1(I):-autob(bus\_tou(time1(C1:C2), P1, P2, \_,\_, \_, G, \_,List)),member(I,List),append(\_,[I,E1,B1,B2|\_],List),write('Рейс №'),write(G), write(' '), write(P1), write('-'), write(P2), write(' Відправлення '),write(C1),write(' год. '),write(C2), write(' хв. '),write('Прибуття '),write(E1), write(' год.'),write(' Вартість проїзду: '),write(B1),write(' Довжина маршруту: '),write(B2),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl.

ter1.

Процедура *tern*2 здійснює пошук за номером рейсу. Номер рейсу вводиться в одинарних лапках.

Наступні процедури *ternopil*2 та *ternopil*4, що представляють інтерфейс користувача, є інформаційні – здійснюють пошук рейсів з Тернополя за мінімальною ціною і за мінімальною довжиною відповідно. Ці процедури запускають допоміжні функції: *ismin*, *min\_mark* та *ismi*, *min\_mar*.

За допомогою процедури *ternopil*5 проводиться пошук рейсів в певну годину до вказаного міста:

ternopil5:-

writeln('Введіть місто до якого плануєте подорож з Тернополя прописними літерами і в одинарних лапках: '), read(P2), writeln('Введіть годину після якої ви би хотіли здійснити поїздку: '), read(X),

autobus(bus\_tour(time1(C1:C2), P1, P2, time2(E1:E2), M, \_, G, B2,\_)), X=<C1,write('Рейс №'),write(G), write(' '), write(P1), write('-'), write(P2), write(' Відправлення '),write(C1),write(' год. '),write(C2), write(' хв. '),write('Прибуття '),write(E1), write(' год.'), write(E2), write(' хв.'),write(' Вартість проїзду: '),write(M),write(' Довжина маршруту: '),write(B2),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'); write(' В дане місто не існує маршруту '),nl, fail.

ternopil5.

*clearDatabase* – виконує очищення попередніх відповідей, які збереглися в базі даних. Тобто стираються всі відповіді, які збережені у в базі *autobus*.

retractall(autobus(bus\_tour(\_,\_,\_,\_,\_,\_,\_,\_,\_))).

Процедура *add\_way* додає нові факти в базу даних:

add\_way:-

write('Введіть пункт виїзду: '), read(P1),

write('Введіть пункт приїзду: '), read(P2),

write('Введіть час виїзду: '), read(C),

write('Введіть час приїзду: '), read(E),

write('Введіть номер рейсу: '), read(G),

write('Введіть марку автобуса: '), read(A),

write('Введіть вартість проїзду: '), read(B1),

write('Введіть довжину шляху: '), read(B2),

write('Введіть пункти проїзду у вигляді списку вигляду СТАНЦІЯ, Час прибуття, Час відправлення, Відстань від початкового пункту, Вартість проїзду від початкового пункту: '), read(H1),read(H2),read(H3),read(H4),read(H5),nl, read(H6),read(H7),read(H8),read(H9),read(H10), nl, read(H11),read(H12),read(H13),read(H14),read(H15),

assert(autobus(bus\_tour(time1(C), P1, P2, time2(E), B1, A, G, B2, [station(H1, time1(H2), time2(H3), H4, H5), station(H6, time1(H7), time2(H8), H9, H10),station(H11, time1(H12), time2(H13), H14, H15)]))),

writeln('Записано в БД').

add\_way.

## 3.2.3 Програмна реалізація бази знань та даних для подорожі літаком

База даних (БД) спроектованої нами системи, це програмно-незалежний файл (*faircraft.bd*), який при загрузці основного модуля програми (expert.pro) підвантажується за допомогою, вбудованого правила прологу: *consult*("*faircraft.bd*")і надалі використовується програмою, для роботи з правилами та предикатами, які орієнтовані на базу даних. В базі даних містяться дані про Р1 і Р2 – пункти відправлення і прибуття відповідно, time1(C1:T1) і time2(C2:T2) – час відправлення і прибуття відповідно; Number – номер рейсу; Company – компанія, що здійснює переліт; Typeofaircraft – тип повітряного судна; Listofdays – список днів, в які є маршрут або атом alldays, якщо рейс здійснюється кожного дня тижня.

Предикат – це логічна функція від одного або декількох аргументів, тобто функція, яка діє в множині з двох значень: істина і хибність. Предикат Прологу записується у вигляді складеного терма: ім’я\_предиката (аргументи). В роботі ім’я предиката – flight, його арність – 8:

flight(Р1, Р2, time1(С1:Т1), time2(С2:Т2), Number, Company, Typeofaircraft, Listofdays).

БД представлено природнім способом у вигляді множини фактів [14]. Наведемо програмну реалізацію фактів:

flight('ЛЬВІВ','ВІДЕНЬ',[time1(05:20),time2(07:00),'AUA384','Austrian\_Airlines','DH8D',[2,4,5,6]]).

flight('СТАМБУЛ', 'ЛЬВІВ',[time1(08:05),time2(10:00), 'THY0441', 'Turkish\_Airlines', 'A320',[1,3,4,6,7]]).

flight('ЛЬВІВ','СТАМБУЛ',[time1(10:55),time2(13:00), 'THY0442', 'Turkish\_Airlines', 'A320',[1,3,4,5,7]]).

flight('МЮНХЕН','ЛЬВІВ',[time1(08:05),time2(10:00), 'DLH2550', 'Lufthansa', 'CRJ9', alldays]).

flight('ЛЬВІВ','МЮНХЕН', [time1(12:00),time2(13:25), 'DLH2551', 'Lufthansa', 'CRJ9', alldays]).

flight('ВАРШАВА', 'ЛЬВІВ',[time1(13:55),time2(14:55), 'LOT765', 'Polish\_Airlines', 'B738',alldays]).

flight('ЛЬВІВ','ВАРШАВА',[time1(15:45),time2(16:45), 'LOT766', 'Polish\_Airlines', 'B738',alldays]).

flight('ВАРШАВА', 'ЛЬВІВ',[time1(16:30),time2(17:30), 'LOT763', 'Polish\_Airlines', 'B738',alldays]).

Про базу знань, тобто про правила та предикати нашої програми можна сказати, що основна їх частина є внутрішньою, тобто вбудовані в нашу програму. Та передбачений варіант, коли правила формуються самостійно користувачем, вони зберігаються в базі даних, та мають специфічний синтаксис.

Запити в системі. Програма починається з запиту *lemberg*, який виведе всі рейси зі Львова:

lemberg:-writeln('Зі Львова можна вилетіти в міста: '), flight('ЛЬВІВ',B,[time1(C),time2(E),G,W,A,H]),write(B), write(' об '), write(C), write(' год. '),write('Прибуття '),write(E), write(' год.'), write(' Тип судна: '),write(A),nl,write('Компанія: '),write(W),write(' Рейс №'),write(G),write(' у дні тижня: '),write(H),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl,fail.

lemberg.

Правило *lemberg* не містить аргументів. Тут підключається база даних, в якій вже містяться дані по нашій предметній області.

Наступним етапом є виклик запиту *viden*, який виведе всі рейси з Відня:

viden:-writeln('З Відня можна вилетіти в міста: '), flight('ВІДЕНЬ',B,[time1(C:D),time2(E:F),G,W,A,H]),write(B), write(' об '), write(C), write(' год. '),write(D), write(' хв.'),write('Прибуття '),write(E), write(' год.'),write(F), write(' хв.'),write(' Тип судна: '),write(A),nl,write('Компанія: '),write(W),write(' Рейс №'),write(G),write(' у дні тижня: '),write(H),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl,fail.

viden.

Правило *viden* теж не містить аргументів. Тут також, як в попередньому випадку, під’єднується база даних, в якій вже містяться дані по нашій предметній області.

Запит *lemberg1* виводить прямі рейси зі Львова за певним містом. Змінна *В* позначає місто до якого планується подорож зі Львова. Назву міста треба ввести прописними літерами і в одинарних лапках:

lemberg1:-writeln('Введіть місто до якого плануєте подорож зі Львова прописними літерами і в одинарних лапках: '), read(B), flight('ЛЬВІВ',B,[time1(C:D),time2(E:F),G,W,A,H]),write('Рейс №: '),write(G),write(' До '),write(B), write(' виліт об '), write(C), write(' год. '),write(D), write(' хв.'),write(' прибуття об '),write(E), write(' год. '),write(F), write(' хв.'),nl,write(' Повітряне судно '),write(A),write(' компанія '),write(W),write(' у дні тижня: '),write(H),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl,fail.

lemberg1.

Аналогічно до попереднього, запит *viden*1 виводить прямі рейси з Відня за певним містом:

viden1:-writeln('Введіть місто до якого плануєте подорож з Відня прописними літерами і в одинарних лапках: '), read(B), flight('ВІДЕНЬ',B,[time1(C:D),time2(E:F),G,W,A,H]),write('Рейс №: '),write(G),write(' До '),write(B), write(' виліт об '), write(C), write(' год. '),write(D), write(' хв.'),write(' прибуття об '),write(E), write(' год. '),write(F), write(' хв.'),nl,write(' Повітряне судно '),write(A),write(' компанія '),write(W),write(' у дні тижня: '),write(H),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl,fail.

viden1.

Запит *reys* здійснює пошук усіх параметрів рейсу за його номером:

reys:-writeln('Введіть номер рейсу прописними літерами в одинарних лапках: '), read(G), flight(P,B,[time1(C:D),time2(E:F),G,W,A,H]),write(' Рейс '),write(P), write(' - '), write(B),nl,write(' Відправка об '),write(C), write(' год. '),write(D), write(' хв.'),nl,write(' Прибуття '),write(E), write(' год. '),write(F), write(' хв.'),nl,write(' повітряним судном '),write(A),write(' компанії '),write(W),write(' у дні тижня: '),write(H) ); write(' рейсу такого не існує'),nl,fail.

reys.

Якщо такого номера рейсу не існує, то система видасть відповідь: «Рейсу такого не існує».

В базі даних останній аргумент – це дні тижня, в які є рейси. Оскільки деякі рейси не існують кожного дня, то треба робити перевірку чи в певний день є той чи інший рейс.

Запит *deny* перевіряє чи існує переліт до деякого міста *В* у певний день тижня:

deny:-writeln('Введіть місто до якого плануєте подорож зі Львова прописними літерами і в одинарних лапках: '), read(B),writeln('Введіть порядковий день тижня подорожі: '), read(P),flight('ЛЬВІВ',B,[time1(C:D),time2(E:F),G,W,A,H]), member(P,H),write(' До '),write(B), write(' можна вилетіти об '), write(C), write(' год. '),write(D), write(' хв.'),write(' і прибути об '),write(E), write(' год. '),write(F), write(' хв.'),write(' повітряним судном '),write(A),write(' компанії '),write(W); write(' Рейсу у цей день не існує'),nl,fail.

Якщо такого перельоту не існує, то система видасть відповідь: «Рейсу у цей день не існує».

Запити *lemberg*2 і *viden*2 (додаток А) здійснюють пошук рейсів зі Львова та Відня відповідно за певним містом з пересадкою. В правилі проходить підрахунок часу, від прильоту одного рейсу і вильоту іншого, що дозволить пасажиру оцінити, чи часу для очікування йому буде достатньо, щоб зробити пересадку:

lemberg2:-writeln('Введіть місто до якого плануєте подорож зі Львова прописними літерами і в одинарних лапках: '), read(I), flight('ЛЬВІВ','ВІДЕНЬ',[time1(C:D),time2(E:F),G,W,A,\_]), flight('ВІДЕНЬ',I,[time1(U:J),time2(K:L),M,N,O,\_]),write('Рейс №: '),write(G), write( 'Виліт об '), write(C), write(' год. '),write(D), write(' хв.'),write(' прибуття у Відень об '),write(E), write(' год. '),write(F), write(' хв.'),nl,write(' Повітряне судно '),write(A),write(' компанія '),write(W),nl,write(' Виліт з Відня об '),write(U), write(' год. '),write(J), write(' хв.'),write('Рейс №: '),write(M),write(' Повітряне судно '),write(O),write(' компанія '),write(N),nl,write(' прибуття у '),write(I),write(' о '), write(K),write(' год. '),write(L), write(' хв.'),nl,write(' Час очікування '),((E=<U,R is (U-E)\*60+J-F,S is R//60,T is R mod 60);(E>U,R is (U-E)\*60+J-F,S is R//60+24,T is R mod 60)), write(S),write(' год. '),write(T), write(' хв.'), nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl,fail.

lemberg2.

Запит *lemberg*3 здійснює пошук прямих і непрямих рейсів зі Львова. -

lemberg3:-writeln('Введіть місто до якого плануєте подорож зі Львова прописними літерами і в одинарних лапках: '), read(I),lemb(I),lemb2(I), nl,fail.

lemberg3.

Система запропонує альтернативні поїздки. Альтернативи тісно пов’язані між собою, тому, що в процесі своєї роботи користуються одними і тими самими правилами вбудованими в систему.

Правило *lemberg*3 здійснює перепосилання на 2 предикати *lemb*(*I*) і *lemb*2(*I*), які, на відміну від попередніх мають аргумент *І*, де *І* – це місто, до якого планується подорож.

lemb(I):-flight('ЛЬВІВ',I,[time1(C:D),time2(E:F),\_,W,A,H]),write(' До '),write(I), write(' зі Львова прямим рейсом виліт об '), write(C), write(' год. '),write(D), write(' хв.'),write(' прибуття об '),write(E), write(' год. '),write(F), write(' хв.'),nl,write(' повітряним судном '),write(A),write(' компанії '),write(W),write(' у дні тижня: '),write(H),nl,write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'),nl.

lemb2(I):-flight('ЛЬВІВ','ВІДЕНЬ',[time1(C:D),time2(E:F),\_,W,A,\_]), flight('ВІДЕНЬ',I,[time1(U:J),time2(K:L),\_,N,O,\_]), write('З пересадкою: Виліт об '), write(C), write(' год. '),write(D), write(' хв.'),write(' прибуття у Відень об '),write(E), write(' год. '),write(F), write(' хв.'),nl,write(' повітряним судном '),write(A),write(' компанії '),write(W),nl,write(' Виліт з Відня об '),write(U), write(' год. '),write(J), write(' хв.'),write(' і прибуття у '),write(I),write(' о '), write(K),write(' год. '),write(L), write(' хв.'),nl,write(' повітряним судном '),write(O),write(' компанії '),write(N),write(' Час очікування у Відні '),((E=<U,R is (U-E)\*60+J-F,S is R//60,T is R mod 60);(E>U,R is (U-E)\*60+J-F,S is R//60+24,T is R mod 60)), write(S),write(' год. '),write(T), writeln(' хв.'),write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_').

Запити *lemberg*4 і *viden*4 здійснюють пошук всіх міст, до яких є рейси зі Львова та Відня.

lemberg4:-findall(City, flight('ЛЬВІВ', City,\_),List),write(List),fail.

lemberg4.

viden4:-findall(City, flight('ВІДЕНЬ', City,\_),List),write(List),fail.

viden4.

Додавання в базу даних проходить за правилом *add\_way*:

add\_way:-

write('Введіть пункт вильоту: '), read(P1),

write('Введіть пункт прильоту: '), read(P2),

write('Введіть час вильоту: '), read(C),

write('Введіть час прильоту: '), read(E),

write('Введіть номер рейсу: '), read(G),

write('Введіть компанію: '), read(W),

write('Введіть тип повітряного судна: '), read(A),

write('Введіть порядкові дні тижня у вигляді списку або alldays: '), read(H),

assert(flight(P1,P2,[time1(C),time2(E),G,W,A,H])),

writeln('Записано в БД').

add\_way.

В цьому пункті йде пряма взаємодія з файлом БД, при вводі даних аналізується кожен з пунктів для введення, покроково, і коли введена інформація вже міститься в БД, то інформація про них відкидається і БД залишається без змін. В протилежному випадку проходить весь процес-діалогу (відповідь на питання поставлені системою), результатом цього процесу є додавання нового, оригінального запису в БД, з предикатом *flight*. В процесі-діалозі потрібно дати відповіді на такі питання: пункт вильоту, пункт прильоту, час вильоту, час прильоту, номер рейсу, компанію, тип повітряного судна та порядкові дні тижня рейсу. Всі ці дані дуже важливі, і в подальшому будуть використовуватись при запиті до системи.

Приклад роботи процедури *add\_way* приведено на рисунку 3.1.

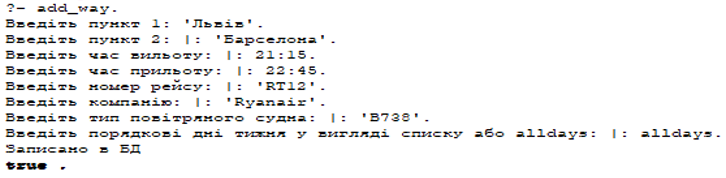


Рисунок 3.1 – Приклад роботи процедури *add\_way*

## 3.3 Розробка інтерфейсу системи. Тестування системи

Взаємодія між експертною системою і користувачем забезпечується системою користувацького інтерфейсу. Зазвичай ця взаємодія полягає у: обробці даних, отриманих з клавіатури, і відображенні вхідних і вихідних даних на екран; підтримці діалогу між користувачем і системою; розпізнаванні ситуації непорозуміння між користувачем і системою.

Система інтерфейсу з користувачем обробляє ввід і вивід, видає чітку і зрозумілу інформацію.

У ході етапу тестування оцінюється обраний спосіб представлення знань у ЕС в цілому. При створенні ЕС інженер по знаннях підбирає тестові приклади, які забезпечують перевірку всіх перспектив розробленої ЕС.

У роботі системи можуть бути наступні джерела невдач: невдалі тестові приклади, ввід-вивід, правила виводу, керуючі стратегії.

Найбільш очевидною причиною невдалої роботи ЕС є характерні тестові приклади. Вони можуть бути занадто однорідними і не охоплювати всю предметну область. Часто підготовці тестових прикладів проходить класифікація їх за проблемами предметної області, окремо виділяють стандартні випадки, визначаючи границі проблемних ситуацій.

Ввід-вивід характеризується даними, одержаними під час діалогу з експертом, і висновками, що робить ЕС у ході пояснень. Якщо задавалися неправильні запитання, або була зібрана не вся необхідна інформація, то методи отримання даних можуть не давати необхідних результатів. Крім того, запитання системи можуть бути важкими для розуміння, багатозначними і не відповідати знанням користувача. Помилки при введенні можуть виникати також через незручний для користувача інтерфейс.

Поширеним джерелом помилок у міркуваннях є правила виводу. Вони часто бувають помилковими, суперечливими і неповними. Якщо невірною є посилка правила, то це призведе до вживання правила в неналежному контексті. Якщо правило діє неправильно, то неправильним буде кінцевий результат.

## 3.3.1 Розробка інтерфейсу системи подорожі поїздом. Тестування системи

Інтерфейс системи розроблявся відповідно до основних функцій експертної системи. Загальне меню складається з семи пунктів (Рисунок 3.2).

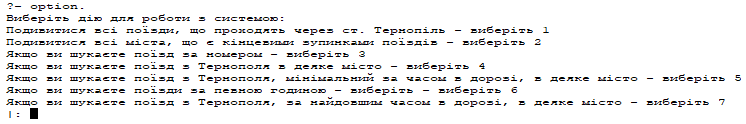


Рисунок 3.2 – Основне меню експертної системи

Дальше при введенні відповідної цифри йде перепосилання на процедури, які ми приводили в підрозділі 3.2.

Тестування розробленої ЕС проходило за всіма цифрами, що приведені на рисунку 3.2. Приведемо приклади для деяких цифр.

Тестування роботи ЕС при виборі цифр 3, 4, 5 і 10 приведено на рисунках 3.3-3.6 відповідно.

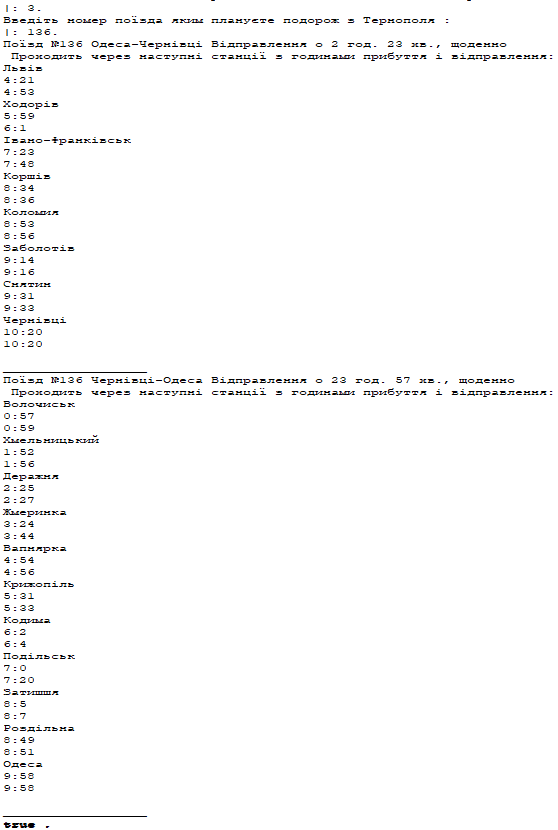


Рисунок 3.3 – Приклад роботи експертної системи при виборі цифри 3

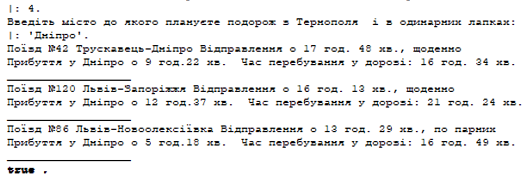


Рисунок 3.4 – Приклад роботи експертної системи при виборі цифри 4

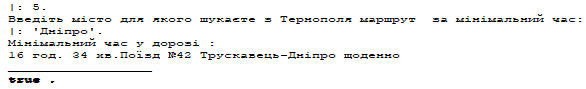


Рисунок 3.5 – Приклад роботи експертної системи при виборі цифри 5

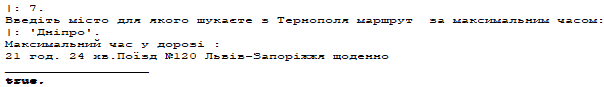


Рисунок 3.6 – Приклад роботи експертної системи при виборі цифри 7

Можна також в системі здійснити окремий виклик процедур і таким чином їх протестувати. Рисунок 3.7 демонструє приклад роботи процедури *passenger\_train*5, рисунок 3.8 – процедури *passenger\_train*6.

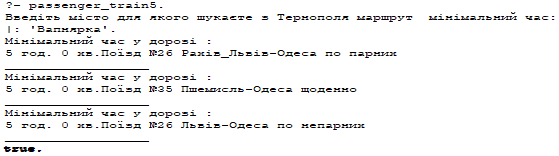


Рисунок 3.7 – Приклад роботи експертної системи при виборі процедури *passenger\_train*5

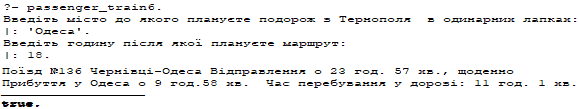


Рисунок 3.8 – Приклад роботи експертної системи при виборі процедури *passenger\_train*6

## 3.3.2 Розробка інтерфейсу системи подорожі автобусом. Тестування системи

На рисунку 3.9 приведено результат роботи команди *consulter*.

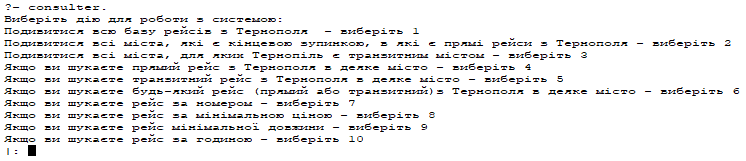
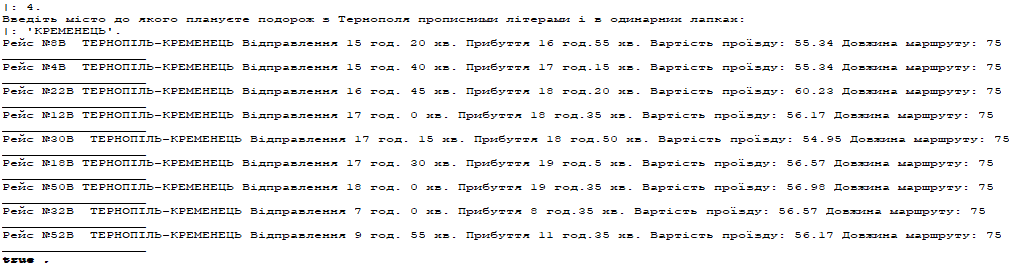


Рисунок 3.9 – Результат роботи команди consulter

Дальше при введенні відповідної цифри йде перепосилання на процедури підрозділу 3.1.

Тестування роботи програмного модуля при виборі цифри 4, 5 і 10 приведено на рисунках 3.10-3.12 відповідно.

 Рисунок 3.10 – Приклад роботи системи при виборі цифри 4

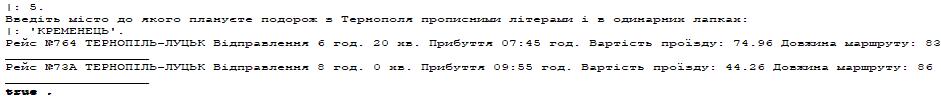
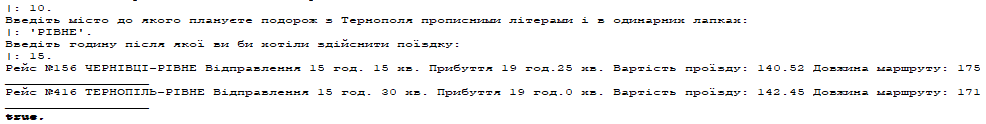


Рисунок 3.11 – Приклад роботи системи при виборі цифри 5

Рисунок 3.12 – Приклад роботи системи при виборі цифри 10

Окрім цього в системі передбачено окремий виклик процедур. Рисунки 3.13 – 3.16 демонструють їх виконання.

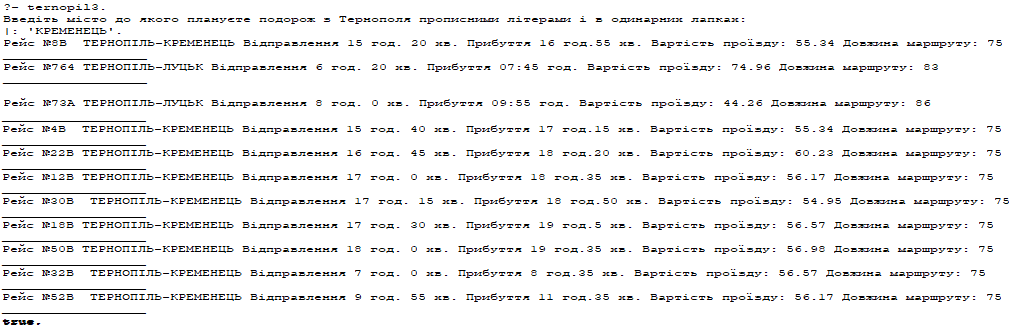
****

Рисунок 3.13 – Приклад роботи процедури *ternopil*3

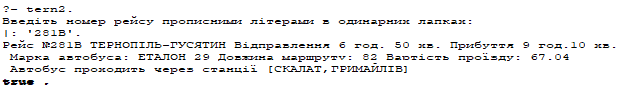
****

Рисунок 3.14 – Приклад роботи процедури *tern*2

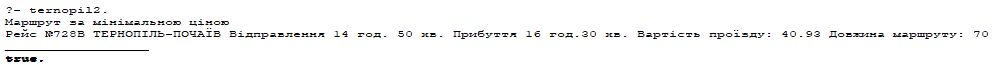


Рисунок 3.15 – Приклад роботи процедури *ternopil*2

В результаті роботи програмного модуля можна зробити висновок, що він має багато функцій, є ефективним, зручним в користуванні і при розширенні бази даних його можна використовувати працівниками Тернопільського автовокзалу.

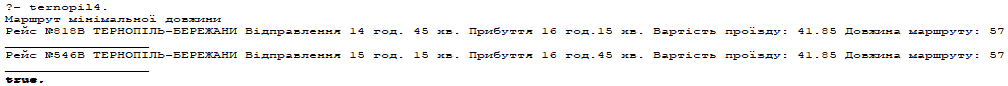


Рисунок 3.16 – Приклад роботи процедури *ternopil*4

## 3.3.2 Розробка інтерфейсу системи подорожі літаком. Тестування системи

Обробка даних проходить за допомогою сформованих нами правил, відповідно, до поставленого завдання.

Основне меню експертної системи приведено на рисунку 3.17.

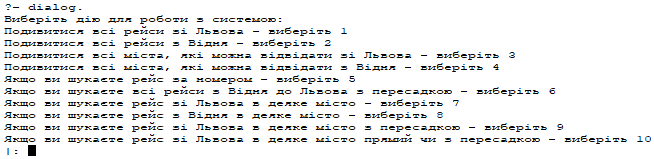


Рисунок 3.17 – Основне меню експертної системи

Дальше при введенні відповідної цифри йде перепосилання на процедури, які ми приводили в підрозділі 3.2.

Приклад роботи експертної системи при виборі цифри 3 і 10 приведено на рисунках 3.18-3.19 відповідно.

Окрім цього в системі передбачено окремий виклик процедур. Так як розроблена програмна система містить одинадцять таких процедур: *lemberg*, *lemberg*1, *lemberg*2, *lemberg*3, *lemberg*4, *viden*, *viden*1, *viden*2, *viden*4, *reys*, *deny*, то можна скористатися вибором однієї з них. Рисунок 3.20 демонструє приклад роботи процедури *reys*, рисунок 3.21 – процедури *lemberg*1.

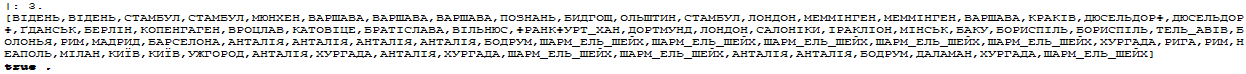


Рисунок 3.18 – Приклад роботи експертної системи при виборі цифри 3

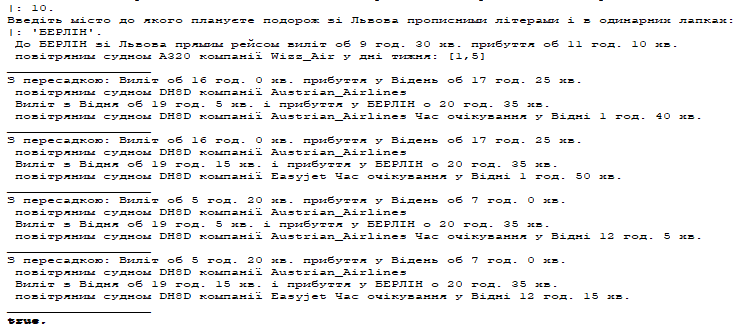


Рисунок 3.19 – Приклад роботи експертної системи при виборі цифри 10

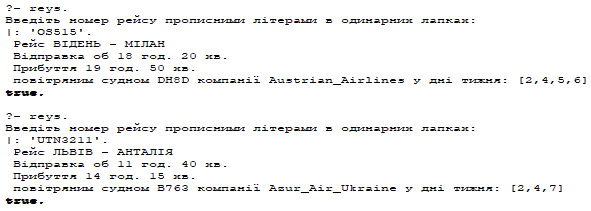


Рисунок 3.20 – Приклад роботи процедури *reys*

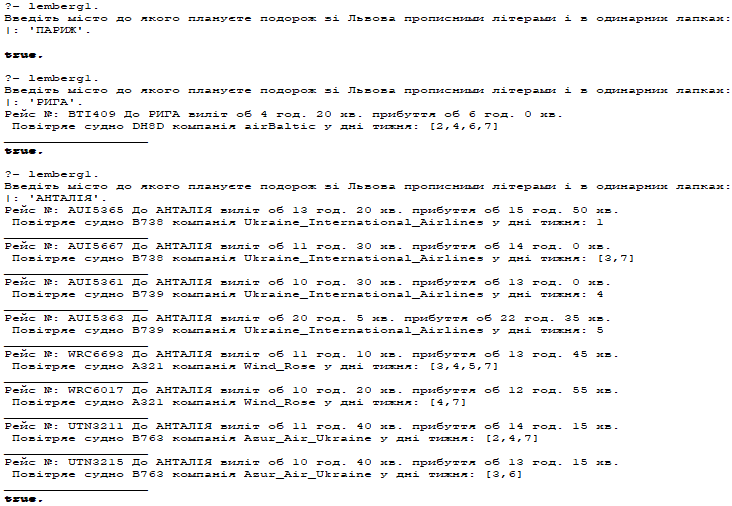


Рисунок 3.21 – Приклад роботи процедури *lemberg*1

В результаті роботи експертної системи можна зробити висновок, що вона має зручний для користувача інтерфейс, містить багато функцій, вирішує поставлені задачі, є ефективною, її можуть використовувати користувачі, що планують подорожі поїздом, автобусом, авіаподорожі, працівники Тернопільського залізничного вокзалу, Тернопільського автобусного вокзалу міжнародного аеропорту «Львів» імені Данила Галицького і Віденського міжнародного аеропорту «Швехат».

## 3.4 Висновки до розділу 3

В даному розділі приведені технічні вимоги до функціонування експертної системи, описана програмна реалізація бази знань та даних для подорожі поїздом, автобусом, літаком, розроблено інтерфейс системи, продемонстровано тестування системи.

# ВИСНОВКИ

1. Проаналізовано предметну область, зроблено огляд відомих моделей та мов представлення знань консультування пасажирів залізничного транспорту, здійснено аналіз відомих онлайн пошукових систем для консультування пасажирів залізничного транспорту, описані моделі представлення знань, зроблено огляд та обґрунтування вибору мови програмування для експертної системи.
2. Описано етапи розробки експертної системи, розроблено структуру експертної системи, описано базу знань, блок логічного виводу, блок взаємодії з користувачем, приведено алгоритм роботи експертної системи.
3. Програмно реалізовано компоненти експертної системи, розроблено інтерфейс системи, зроблено тестування системи.
4. Програмний модуль можуть використовувати користувачі, що планують поїздки, а також працівники Тернопільського залізничного вокзалу і Тернопільського автобусного вокзалу міжнародного аеропорту «Львів» імені Данила Галицького і Віденського міжнародного аеропорту «Швехат».

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Berezsky O., Berezska K., Batko Yu., Melnyk G. Vision-based medical expert system. 6th International Scientific and Technical Conference “Computer Sciences and Information Technologies”(CSIT'2011, Lviv, Ukraine, 16-19 November), 2011. p. 49-50.
2. Березький О. М., Мельник Г.М., Березька К. М., Дацко Т. В. Інтелектуальна система аналізу зображень ауто- та ксеногенних тканин. Науковий вісник НЛТУ України: зб. наук.-техн. праць. Львів: РВВ НЛТУ України. 2014. Вип. 24.11. С. 323-330.
3. Березький О. М. Березька К. М. Гібридні інтелектуальні системи медичного діагностування. Матеріали Міжнародної наукової конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту» (ISDMCI’2014), м. Залізний Порт, 28–31 травня 2014 р. С. 246–247.
4. Березький О. М., Березька К. М., Попіна С. Ю. Статистичне оброблення цитологічних зображень. Вісник Хмельницького національного університету: зб. наук.-техн. праць. Сер.: Технічні науки. 2012. № 5. С. 161–164.
5. Березький О. М., Мельник Г.М., Дацко Т.В., Вербовий С. О. База даних цитологічних та гістологічних зображень ауто- та ксеногенних тканин. Науковий вісник НЛТУ України: зб. наук.-техн. праць. Львів: РВВ НЛТУ України. 2014. Вип. 24.10. С. 338-345.
6. Березький О. М., Мельник Г. М., Березька К. М. Нечітка база знань інтелектуальної системи діагностування видів раку молочної залози. Вісник Хмельницького національного університету. Технічні науки. 2013. №6. С.284-291.
7. Березький О. М., Батько Ю.М., Мельник Г. М. Інформаційно-аналітична система дослідження та діагностування пухлинних клітин на основі аналізу їх зображень. Вісник Хмельницького національного університету. Технічні науки. 2008. №4. С.33-41.
8. Березький О. М., Батько Ю.М., Березька К.М. та ін. Методи, алгоритми та програмні засоби опрацювання біомедичних зображень. Тернопіль: Економічна думка, ТНЕУ, 2017. 330 с.
9. Berezsky O., Dubchak L., Batryn N., Datsko T., Berezska K., Pitsun O., Batko Y. Fuzzy System For Breast Disease Diagnosing Based On Image Analysis. Proceedings of the II International Workshop Informatics & Data-Driven Medicine (IDDM 2019). Lviv, Ukraine. 11-13 November, 2019.
10. Березский О. Н., Березская Е. Н. Количественная оценка качества сегментации изображений на основе метрик. Управляющие системы и машины. 2015. №6. С.59-65.
11. Александров Е.А. Основы теории эвристических решений. Подход к изучению естественного и искусственного интеллекта. М.: Сов. Радио, 1975. 256  с.
12. Експертні системи [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/ wiki/Експертні\_системи.
13. Джексон П. Введение в экспертные системы». М.: Вильямс, 2001. 397 с.
14. Рутецький Ю. О. Експертна система для пошуку оптимальних варіантів подорожей. *Інтелектуальні комп’ютерні системи та мережі* : тези доп. VІ Наук.-практ. конф. молодих вчених і студентів (10 лист. 2022 р.). Тернопіль : ЗУНУ, 2022.
15. Березький М. О., Рутецький Ю. О. Мови штучного інтелекту і машинного навчання. *Інтелектуальні комп’ютерні системи та мережі* : тези доп. VІ Наук.-практ. конф. молодих вчених і студентів (10 лист. 2022 р.). Тернопіль : ЗУНУ, 2022.
16. Березький О. М., Дубчак Л. О., Мельник Г. М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма – «Комп’ютерна інженерія». Тернопіль : ЗУНУ, 2021. 32 с.
17. Загальні рекомендації з підготовки, оформлення, захисту й оцінювання випускних кваліфікаційних робіт здобувачів вищої освіти першого бакалаврського і другого магістерського рівнів / за ред. доц. М.І. Шинкарика. Тернопіль: ТНЕУ, 2018. 60 с.
18. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.
19. Онлайн-система E-TRAVELS [Електронний ресурс]. – Режим доступу: https://e-travels.com.ua/ua.
20. Залізничний транспорт [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/.
21. Повітряний транспорт України [Електронний ресурс]. – Режим доступу: https://uain.press/articles/718438-718438.
22. Порівнюй ціни. Бронюй найкраще. Подорожуй вигідно [Електронний ресурс]. – Режим доступу: https://tickets.ua/uk.
23. Онлайн-система E-TRAVELS [Електронний ресурс]. – Режим доступу: https://e-travels.com.ua/ua.
24. BOOKING.UZ [Електронний ресурс]. – Режим доступу: https://booking.uz.gov.ua/.
25. BUSFOR [Електронний ресурс]. – Режим доступу: https://busfor.ua/.
26. TripMyDream [Електронний ресурс]. – Режим доступу: <https://tripmydream.ua/flights>.
27. Искусственный интеллект: в 3-х кн. / Справочник / Под ред. Д.А. Поспелова. М.: Радио и связь, 1990. Кн. 2: Модели и методы. 304 с.
28. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. Спб.: Питер, 2000. 384 с.
29. Пролог – язык программирования будущего: пер с англ. / Доорс Дж. и др. М.: Финансы и статистика, 1990. 144 с.
30. Мова логічного програмування Пролог [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Пролог.
31. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG: Изд. 3-е / пер. с англ. М.: Издательский дом «Вильямс», 2001. 640 с.
32. Братко И. Программирование на языке Пролог для искусственного интеллекта / пер. с англ. М.: Мир, 1990, 560 с.
33. Практичне програмування мовою Visual Prolog : навч. посіб. / [Дейнега Л. Ю., Камінська Ж. К., Левада І. В., Сердюк С. М] Запоріжжя : ЗНТУ, 2016. 236 с
34. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог / пер. с англ. М.: Мир, 1990. 225 с.
35. Ин Ц., Соломон Д. Использование Турбо-Пролога / пер. с англ. М.: Мир, 1993. 608 с.
36. Месюра В. І., Яровий А. А., Арсенюк І. Р. Експертні системи. Частина 1. Навчальний посібник. Вінниця: ВНТУ, 2006. 114 с.
37. Баклан І.В. Експертні системи. Курс лекцій. Навчальний посібник. К.: НАУ, 2012. 132 с.
38. Моделі представлення знань [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Моделі\_представлення\_знань
39. Шаров С.В. Інтелектуальні інформаційні системи: навч. посіб. / С.В. Шаров, Д.В. Лубко, В.В. Осадчий. Мелітополь: Вид-во МДПУ ім. Б. Хмельницького, 2015. 144 с.
40. Шаховська Н. Б. Системи штучного інтелекту: навч. посібник / Н. Б. Шаховська, Р. М. Камінський, О. Б. Вовк. Львів: Видавництво Львівської політехніки, 2018. 392 с.
41. Березький О. М., Піцун О.Й., Лящинський П.Б., Лящинський П.Б., Мельник Г.М. Інтелектуальна система автоматизованої мікроскопії аналізу гістологічних та цитологічних зображень. Штучний інтелект, Київ, 2017. №2 (76). С. 128-140.
42. Berezsky O., Melnyk G., Datsko T., Verbovy S. An Intelligent System for Cytological and Histological Image Analysis. Proceedings of the 13 th International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» CADSM 2015, 24-27 February 2015, Polyana-Svalyava (Zakarpattya), Ukraine. 2015. Р. 28-31.
43. Berezsky O., Melnyk G., Batko Yu. Biomedical image search and retrieval algorithms. Computing. 2008. Vol. 7, Issues 1. P.108-113.
44. Berezsky O., Datsko T., Verbovyy S. The intelligent system for diagnosing breast cancers based on image analysis. Proceedings of Information Technologies in Innovation Business (ITIB), 7-9 October, 2015, Kharkiv, Ukraine. Р. 27-30.
45. Berezsky О. М., Berezska К.M., Melnyk G.M., Batko Y.M. Design of computer systems for biomedical image analysis. Proceedings of the X th International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» CADSM 2009, 24-28 February 2009, Lviv-Polyana, Ukraine. Lviv: Publishing House Vezha&CoС. Р.186-191.
46. Березький О. М., Батько Ю.М., Мельник Г. М. Комп’ютерна система аналізу біомедичних зображень. Вісник Національного університету «Львівська політехніка». Комп’ютерні науки та інформаційні технології. 2009. № 570. С. 84-89
47. Лорьєр Ж-Л. Системы штучного интеллекта / пер. с франц. М.: Мир, 1991. 568 с.
48. Таунсенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ / пер. с англ. М.: Финансы и статистика, 1990. 320 с.
49. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: Навчальний посібник. Запоріжжя: ЗНТУ, 2008. 341 с.
50. Глибовець М.М., Отецький О.В. Штучний інтелект: Підруч. для студ. ВНЗ, що навчаються за спец. „Комп. науки” та „Прикл. математика”. К.: Вид. дім „КМ Академія”, 2002. 366 с.