МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСТИТЕТ

ФАКУЛЬТЕТ КОМП’ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КУРСОВА РОБОТА

з дисципліни «Комп’ютерні системи штучного інтелекту»

на тему «Створення експертної системи для вибору клавіатур»

Виконав: студент гр. КСМм-51

Пилипчук Андрій

Перевірив: Березький О. М.

Тернопіль – 2013

**ЗМІСТ**

ВСТУП

1. Оглядова частина

1.1 Характеристики експертної системи

1.2 Структура експертної системи

1.3 Методи реалізації експертної системи

1.4 Огляд предметної області

2. ????????

2.1 Класифікація моделей представлення знань

2.2 Логічна модель

2.3 Продукційна модель

2.4 Модель семантичної мережі

2.5 Фреймова модель

3. Програмна реалізація експертної системи

3.1 База знань

3.2 Блок логічного виводу

3.3 Блок взаємодії з користувачем

ВИСНОВКИ

ПЕРЕЛІК ПОСИЛАНЬ

ДОДАТОК А

**ВСТУП**

Одним з направлень інформаційних технологій є царина інформатики, яка називається штучним інтелектом. Штучний інтелект — це така програмна система, що імітує на комп’ютері мислення людини. Для створення такої системи потрібно:

* вивчити процес мислення людини, яка розв’язує певні задачі або приймає рішення у певній ділянці знань,
* виділити основні позиції цього процесу і розробити програмні засоби, які б реалізували ці позиції за допомогою комп’ютера.

Інакше кажучи, штучний інтелект передбачає простий структурний підхід до розроблення складних програмних систем прийняття рішення.

При цьому ефективність таких програм залежить від системи знань, якими програма оперує, а не лише від схем виводу, які вона використовує, тобто, щоб зробити програму інтелектуальною, її потрібно наповнити множиною високоякісних спеціальних знань про певну предметну галузь.

Комплекс таких програм, кожна з яких є експертом у первинній предметній галузі, отримав назву експертних систем.

Технологію побудови експертних систем часто називають інженерією знань. Це процес взаємодії автора експертної системи (інженера знань) з одним або кількома експертами в певній предметній області. Інженер знань «видобуває» з експертів стратегії, процедури, правила, які вони використовують під час розв’язування задач, і вбудовує отримані знання в експертну систему.

**1 ОГЛЯДОВА ЧАСТИНА**

**1.1 Характеристики експертної системи**

Експертна система — комп’ютерна система, що здатна частково замінити спеціаліста-експерта у вирішенні певних задач.

Основою експертної системи є база знань, яка накопичується в процесі побудови експертної системи. Знання в такій базі організовані й відображені так, щоб спростити прийняття рішення.

Однією з характеристик експертної системи є те, що вона застосовує досвід мислення найкваліфікованіших експертів у даній ділянці знань, що приводить до точних, творчих та ефективних рішень.

Крім того, експерти можуть з часом мінятися, а їхній досвід — лишається незмінним. Так забезпечується нова характеристика — інституційна пам’ять.

Іншою характеристикою експертних систем є наявність у них прогностичних можливостей, тобто використання принципу  
«If… Then… Else…», одним словом, системи можуть видавати відповіді на поведінку в конкретній ситуації і показувати, як зміняться ці відповіді у нових ситуаціях. Це дасть змогу користувачеві оцінити можливий вплив нових фактів або інформації та зрозуміти, як вони пов’язані з рішенням.

Останньою характеристикою експертної системи є те, що її можна використовувати для навчання й тренування робітників та спеціалістів.

## 1.2 Структура експертної системи

У роботі з експертною системою беруть участь:

* сама експертна система;
* експерт;
* інженер знань;
* засіб побудови експертної системи;
* користувач.

Їхні ролі та взаємовідносини наведені на рисунку 1.1.

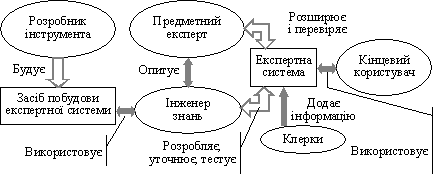


Рисунок 1.1 – Компоненти експертної системи та зв’язки між ними

Вище наголошувалося, що основою експертної системи є сукупність знань, структурована з метою спрощення процесу прийняття рішення експертною системою.

Знання для спеціалістів штучного інтелекту є інформацією, яка необхідна програмі, щоб бути «інтелектуальною». Ця інформація набуває вигляду фактів і правил, при цьому факти та правила можуть бути істинними, хибними та інколи може бути присутня непевність у достовірності факту або правила. У разі явно вираженої непевності під час викладення факту чи правила використовується коефіцієнт упевненості.

Деякі правила експертної системи є евристичними, тобто емпіричними правилами чи спрощеннями, що ефективно обмежують процес рішення. Такі правила не піддаються точному математичному або алгоритмічному розв’язанню. Алгоритмічний метод гарантує коректне чи оптимальне розв’язання задачі, а евристичний метод — дає прийнятне розв’язання в більшості випадків.

Знання в експертній системі організовані таким чином, щоб знання про предметну галузь були відокремлені від інших загальних знань. Відокремлені знання про предметну галузь називаються базою знань, а загальні знання — механізмом виводу. Усі програми, що працюють зі знаннями, які організовані за допомогою бази знань і механізму виводу, називають системами, що ґрунтуються на знаннях.

База знань — особливий вид бази даних, що використовуються в експертних системах та призначена для роботи зі знаннями (метаданими).

База знань експертної системи містить факти (дані) і правила (або інші знання), які використовують ці факти як основу для прийняття рішення. Механізм виводу містить інтерпретатор, що визначає, яким чином застосовувати новий порядок для виведення нових знань, і диспетчер, що встановлює порядок застосування цих знань.

Такі експертні системи одержали назву статичних експертних систем і мають структуру, аналогічну рисунку 1.2. Ці експертні системи використовуються в тих додатках, де можна не враховувати зміни навколишнього світу за час вирішення задачі.

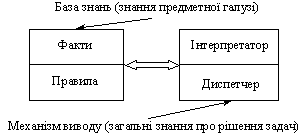


Рисунок 1.2 – Структура статичної експертної системи

Однак існує більш високий клас додатків, де потрібно враховувати динамічні зміни навколишнього світу за час виконання додатку. Такі експертні системи одержали назву динамічних, і їхня узагальнена структура буде мати вигляд, приведений на рисунку.1.3.

У порівнянні зі статичною експертною системою, у динамічну вводиться ще два компоненти:

* підсистема моделювання зовнішнього світу;
* підсистема сполучення з зовнішнім світом.

Динамічні експертні системи здійснюють зв’язок із зовнішнім світом через систему контролерів і датчиків. Крім того компонента БЗ і механізму виведення істотно змінюються, щоб відобразити тимчасову логіку подій, що відбуваються в реальному світі.

## 1.3 Методи реалізації експертних систем

Для реалізації експертних систем застосовуються мови програмування, такі як: LISP, PROLOG, PASCAL та інші процедурні мови програмування та мови інженерії знань. За своїм призначенням і функціональними можливостями інструментальні програми, які застосовуються для проектування експертних систем, можна поділити на чотири достатньо великих категорії.

**Оболонки експертних систем.** Системи цього типу створюються, як правило, на основі будь-якої експертної системи, яка достатньо непогано зарекомендувала себе на практиці. При створенні оболонки-прототипу видаляються компоненти, які є занадто специфічними для області її безпосереднього застосування, і залишаються ті, які не мають вузької спеціалізації. Прикладом може служити система EMYCIN, створена на основі системи MYCIN. В EMYCIN збережений інтерпретатор і всі базові структури даних — таблиці знань і зв’язаний з ними механізм індексації. Оболонка, доповнена спеціальною мовою, яка покращує читабельність програм, і засобами підтримки бібліотеки типових випадків і висновків, виконаних за ними експертною системою. Подальшим розвитком оболонки EMYCIN стали системи S.1 і M.4, у яких механізм побудови ланки зворотних міркувань, позичений в EMYCIN, поєднаний з фреймоподібною структурою даних і додатковими засобами управління шляхом міркувань.

**Мови програмування високого рівня.** Інструментальні засоби цієї категорії звільняють розробника від необхідності заглиблюватися в деталі реалізації системи — способи ефективного розподілення пам’яті, низькорівневі процедури доступу й маніпулювання даними.

Мови високого рівня в руках досвідченого програміста є потужним засобом швидкого створення прототипу експертної системи, дозволяють забезпечити гнучкість процесу розробки при одночасному зниженні матеріальних витрат і скороченні термінів виконання проекту. Як правило, середовище розробки таких мов забезпечує поєднання інтерфейсу розробки і часу виконання, що дозволяє поєднати вставку, редагування й тестування фрагментів програмного коду. Але інтерфейс користувача такого середовища поступається інтерфейсу оболонок, але це не заважає досвідченому програмісту швидко її освоїти.

Мови опису правил, що породжують, об'єктно-орієнтовані мови і процедурні дедуктивні системи представляють проектувальникові експертних систем значно більшу свободу дій, ніж оболонки. Особливо це стосується програмування процедур управління й обробки невизначеності. Як було відмічено вище, зазвичай оболонка має вмонтований режим управління й методи обробки невизначеності, які не можуть бути потім змінені в процесі побудови на її основі конкретної експертної системи. Та гнучкість, яку надають програмісту мови високого рівня, особливо важлива при створенні експериментальних систем, в яких з самого початку вибрати оптимальний режим управління неможливо.

Одним із найбільш відомих представників таких мов є OPS5. Ця мова дуже проста у вивченні і представляє програмісту більш широкий спектр можливостей, ніж типові спеціалізовані оболонки. Треба відмітити, що більшість подібних мов так і не було доведено до рівня комерційного продукту і представляють собою скоріш інструмент для досліджень.

**Середовище програмування, яке підтримує декілька парадигм.** Засоби цієї категорії включають декілька програмних модулів, що дозволяє користувачеві комбінувати в процесі розробки експертної системи різні стилі програмування.

**Додаткові модулі.** Засоби цієї категорії представляють собою автономні програмні модулі, призначені для виконання специфічних задач у рамках обраної архітектури системи рішення проблем. Спеціально створені модулі управління семантичною сіткою можна використовувати для розповсюдження внесених змін на всі компоненти системи.

**Архітектури рішення проблем,** такі як системи з дошкою об’яв, представляють собою об’єктно-орієнтовані оболонки, на основі яких можна створювати конкретні програми.

Для систем другого покоління характерно використання змішаної архітектури, яка об’єднує різні парадигми. Використання відповідних інструментальних засобів продемонструвало, що з їх допомогою можна створювати системи із різною архітектурою, яка підтримує різні способи представлення знань і механізми управління процесом.

Багатофункціональні програмні середовища дозволяють досвідченому програмістові експериментувати при рішенні нових класів проблем, вибираючи потрібні сполучення різних методів, які представлені в існуючому модульному наборі. Оскільки не існує єдиної універсальної мови представлення знань для довільної експертної системи, у розробників виникає бажання об'єднати декілька різних схем представлення, особливо на етапі створення прототипу. Хоча повної теорії таких гібридних систем і не існує, експерименти з різними схемами представлення і логічного виводу показали, що кожна з них має свої слабкі сторони. Тому зрозумілим є бажання об'єднати різні методики таким чином, щоб достоїнства одних компенсували недоліки інших.

Правила породження дозволяють представити в програмі емпірично виявлені зв'язки між умовами й діями, між спостереженнями та гіпотезами, але вони значно гірше підходять для представлення відношень між об'єктами предметної області, включаючи і такі важливі, як відношення множина/елемент або множина/підмножина. Структуровані об'єкти, наприклад, фрейми, виявилися більш зручним засобом для зберігання й маніпулювання описами об'єктів предметної області, але застосування таких знань потребує включення в програму фрагментів програмного коду (наприклад на мові LISP), які потім важко аналізувати. Раціональне зерно в перших спробах звести разом стилі, які основані на правилах і фреймах, було в тому, щоб об'єднати здатність представляти об'єкти, характерні для фреймів, із можливостями зв'язувати умови й дії за допомогою правил породження.

Одним із перших багатофункціональних середовищ штучного інтелекту є LOOPS, в якому в рамках єдиної архітектури обміну повідомленнями були об'єднані чотири парадигми програмування:

* + **Процедурно-орієнтоване програмування**. Ця парадигма була представлена мовою LISP, в якій активним компонентом є процедури, а пасивним — дані, незважаючи на те, що в LISP процедури самі по собі також є даними, оскільки мають вигляд списків. В рамках єдиного середовища процедури можуть бути використані для обробки зовнішніх даних, зокрема, зміни значень загальнодоступних змінних.
  + **Програмування, орієнтоване на правила**. Ця парадигма, аналогічна попередній, але роль процедур грають правила «умова – дія». В середовищі LOOPS набори правил самі собою є об'єктами, які можна рекурсивно вкладати один в інший. Таким чином, частина «дія» одного правила, у свою чергу, може активізувати підлеглий набір правил. З множинами правил пов'язуються керуючі компоненти, за допомогою яких у простій формі виконується розв'язання конфліктів.
  + **Об'єктно-орієнтоване програмування.** Структуровані об'єкти мають властивості і процедур, і даних, причому побічні ефекти звичайно, локалізуються в межах об'єкту.Обробка повідомлень, що надходять, призводить до передачі даних або зміни їх значень, але всі маніпуляції даними виконуються під керуванням того компонента, який звертався до об'єкту. При цьому об'єкт, який викликає, зовсім не цікавить, як зберігаються дані і як вони модифікуються усередині об'єкту.
  + **Програмування, орієнтоване на дані.** Доступ до даних та обновлення даних запускає визначені процедури, причому не має значення, чому змінено компонент даних, — чи то результат побічного ефекту, чи результат дії інших процедур. Зі змінними, в яких зберігаються значення даних, пов'язуються визначені процедури, подібно тому, як це відбувається в слотах фрейму, причому такі змінні часто називають активними величинами.В таких додатках, як моделювання, цей стиль програмування виявляється розповсюдити ефективні зміни якого-небудь компоненту на інші, із ним пов'язані.

В рамках основної об’єктно-орієнтованої парадигми модулі середовища, які підтримують різні стилі програмування, можна комбінувати. Звичайно, умови в правилах, які породжують, і логічні фрази пов'язуються зі значеннями слотів структурованих об'єктів, а правила модифікують значення цих слотів. Саме такий стиль об'єднання парадигм в теперішній час реалізований у мові CLIPS.

В системах KEE і LOOPS поведінка об'єктів описується в термінах множин правил, що породжують, як це зробив Ейкінс у системі CENTAUR. В середовищах KEE і Knowledge Craft до перерахованих вище парадигм додається і логічне програмування в стилі мови PROLOG. Нова версія KEE, відома під назвою KAPPA-PC, представляє в розпорядження програміста ще більш широкий набір стилів для комбінування правил, об'єктів і процедур.

До розряду динамічних середовищ розробки експертних систем відноситься сімейство програмних продуктів фірми Gensym Corp. (США). Один з таких продуктів система G2 — базовий програмний продукт, що представляє собою графічне, об'єктно-орієнтоване середовище для побудови і супроводу експертних систем реального часу, призначених для моніторингу, діагностики, оптимізації, планування і керування динамічним процесом.

**1.4 Огляд предметної області**

**2 ??????**

**2.1 Класифікація моделей представлення знань**

Одним з основних елементів в архітектурі експертної системи є база знань (БЗ). Відповідно до одного з принципів, висловлених Е. Фейгенбаум, потужність експертної системи залежить, в першу чергу, від потужності бази знань і можливості її поповнення. Що є вмістом БЗ? Це сукупність знань тієї предметної області, для якої створена експертна система. Причому ці знання представлені в інтерпретації експерта (або групи експертів).

Обробка даних на ЕОМ, як правило, пов'язана з використанням моделей систем, процесів і т.д. Те, що зберігається в БЗ, також не є знаннями в чистому вигляді. Мова йде, як правило, про моделі знань.

Для того щоб маніпулювати всілякими знаннями з реального світу за допомогою комп'ютера, необхідно здійснити їх моделювання.

При проектуванні моделі подання знань варто врахувати дві вимоги: однорідність подання; простоту розуміння.

Виконання цих вимог дозволяє спростити механізм логічного висновку і процеси набуття знань і управління ними, однак, як правило, творцям інтелектуальної системи доводиться йти на певний компроміс у прагненні забезпечити однакове розуміння знань і експертами, і інженерами знань, і користувачами.

Модель представлення знань повинна відбивати істотні характеристики задачі, що розв’язується і забезпечувати відповідною інформацією процедури, що виконують пошук рішень. При цьому вона повинна володіти необхідною виразною здатністю, щоб відбивати всі необхідні деталі предметної області задачі, а також бути досить ефективною з точки зору пошуку рішень, що розглядають як вивід на знаннях. Виразна сила й обчислювальна ефективність — дві основні характеристики моделей представлення знань. Багато моделей представлення знань, що володіють великими виразними можливостями, не піддаються ефективній реалізації. Тому пошук оптимального співвідношення між виразною силою моделі представлення знань, що використовується і ефективністю її реалізації — основне завдання розробника систем штучного інтелекту.

Моделі представлення знань можна умовно розділити на декларативні і процедурні. У декларативних моделях знання представляються у вигляді описів об’єктів і відносин між об’єктами без вказівки в явному вигляді, як ці знання обробляти. Такі моделі припускають відділення описів (декларацій) інформаційних структур від механізму вивід в, що оперує цими структурами. У процедурних моделях значення представляються алгоритмами (процедурами), що містять необхідний опис інформаційних елементів і одночасно визначають способи їх обробки.

Конкретні моделі, застосовувані на практиці, являють собою комбінацію декларативних і процедурних представлень. Найбільш розповсюдженими є наступні моделі представлення знань:

* логічні моделі;
* продукційні моделі;
* мережі моделі;
* фреймові моделі.

**2.2 Логічна модель**

Логічні моделі реалізуються засобами логіки предикатів. У цьому випадку знання про предметну область представляються у вигляді сукупності логічних формул. Тотожні перетворення формул дозволяють одержувати нові знання. Перевагами логічних моделей представлення знань є наявність чіткого синтаксису і широко прийнятої формальної семантики, а також теоретично обґрунтованих процедур автоматичного вивід в. Основним недоліком даних моделей є неможливість одержання висновків в областях, де вимагаються правдоподібні висновки, коли результат виходить з визначеною оцінкою впевненості в його істинності. Крім цього, такі моделі характеризуються монотонним характером вивід в, тобто в базу знань додаються тільки істинні твердження, що виключає можливість протиріч. На практиці часто зустрічаються немонотонні міркування, що важко реалізувати в рамках логічної моделі.

Проте, логічні моделі виступають у якості теоретичної основи опису самої системи представлення знань і поступово розширюють свої можливості. Тому надалі цим моделям приділяється значна увага.

**2.3 Продукційна модель**

У продукційних моделях знання представляються набором правил виду «Якщо А, то В», де умова правила А є твердженням про вміст бази фактів, а наслідок В говорить про те, що треба робити, коли дане продукційне правило активізоване.

Продукційні моделі представлення знань завдяки природній модульності правил, наочності і простоті їхнього створення широко застосовуються в інтелектуальних системах.

Повнота бази знань (бази правил) визначає можливості системи по задоволенню потреб користувачів. Логічний висновок в продукційних системах заснований на побудові прямого і зворотного ланцюжків висновків, утворених в результаті послідовного перегляду лівих і правих частин відповідних правил, аж до отримання остаточного висновку.

Механізм логічного висновку — невід'ємна частина системи, заснованої на знаннях (експертної системи), реалізує функції виводу (формування) умовиводів (нових суджень) на основі інформації з бази знань і робочої пам'яті.

Як випливає з визначення, для роботи механізму логічного висновку необхідна як «довготривала» інформація, що міститься в базі знань у вибраному при розробці експертної системи вигляді, так і «поточна» оперативна інформація, що надходить в робочу пам'ять після обробки в лінгвістичному процесорі запиту користувача. Таким чином, база знань відображає основні (довгострокові) закономірності, притаманні предметної області. Запит користувача, як правило, пов'язаний з появою якихось нових фактів і/або з виникненням потреби в їх тлумаченні.

Перед розглядом конкретних механізмів логічного висновку підкреслимо кілька важливих обставин:

* єдиного механізму логічного висновку для довільних експертних систем, не існує;
* механізм логічного висновку повністю визначається моделлю подання знань, прийнятою в даній системі;
* існуючі механізми логічного виведення не є строго фіксованими (узаконеними) для кожного типу експертних систем.

З усіх відомих механізмів виведення механізм логічного висновку є найбільш формалізованим (зумовленим).

Розрізняють два типи логічного висновку:

* прямий висновок (пряма ланцюжок міркувань);
* зворотний висновок (зворотний ланцюжок міркувань).

Сутність прямого логічного висновку в продукційних експертних системах полягає в побудові ланцюжка висновків (продукцій або правил), що зв'язують початкові факти з результатом виводу.

**2.4 Модель семантичної мережі**

Мережні моделі представлення знань розрізняються між собою типами зв’язків (відносин), що використовуються. Якщо в мережі використовуються ієрархічні зв’язки (клас-підклас, рід-вид і т.п.), то мережа називається класифікованою. Якщо зв’язки між інформаційними елементами представляються функціональними відносинами, що дозволяють обчислювати значення одних інформаційних елементів за значеннями інших, то мережі називають функціональними (обчислювальними). Якщо в мережі допускаються зв’язки різного типу, то її називають семантичною мережею.

Семантична мережа являє собою спрямований граф, в якому вершинам відповідають об’єкти (сутності) предметної області, а дугам – відносини, у яких знаходяться ці об’єкти. Вивід в семантичних мережах може виконуватися на основі алгоритмів співставлення, шляхом виділення підграфів з визначеними властивостями.

У семантичних мережах використовують три типи вершин:

* вершини-поняття (зазвичай це іменники);
* вершини-події (зазвичай це дієслова);
* вершини-властивості (прикметники, прислівники, визначення).

Дуги мережі (семантичні відносини) ділять на чотири класи:

* лінгвістичні (відмінкові, дієслівні, атрибутивні);
* логічні (І, АБО, НЕ);
* теоретико-множинні (безліч — підмножина, відносини цілого і частини, родовидові відносини);
* квантифікувати (визначувані кванторами спільності ∀ і існування  ∃).

До переваг семантичних мереж відносять: велику виразну здатність, наочність графічного представлення, схожість структури мережі до семантичної структури фраз природної мови. Недоліком представлення знань у вигляді семантичних мереж є відсутність єдиної термінології. Дана модель представлення знань знаходить різне втілення в різних дослідників.

**2.5 Фреймова модель**

Фреймові моделі (системи, мережі фреймів) у порівнянні із семантичними мережами дають більш формалізований і в той же час досить гнучкий «механізм» представлення знань. Випереджаючи строге визначення фрейму, помітимо, що фрейм можна розглядати як складний вузол в особливого виду семантичної мережі. У рамках фреймових моделей вдалося значною мірою об'єднати декларативні знання про об'єкти і процедурні знання про методи витягу і перетворення інформації для досягнення заданих цілей.

Термін фрейм (каркас, рамка) був запропонований у 1973р. М Мінським для мінімального опису явищ, понять, об'єктів. Відповідно до його визначення фрейми – це структури даних (знань), використовувані для представлення стереотипних ситуацій. У психології та філософії відоме поняття абстрактного образа чи прототипу. Наприклад, при проголошенні вголос слова «кімната» виникає її абстрактний образ (рамка чи фрейм), що включає житлове приміщення з чотирма стінами, підлогою, стелею, вікнами і дверима, площею від ≈ 6 до 20 м2. Цей опис мінімальний в тому розумінні, що з нього нічого не можна забрати. Так, забравши вікна, одержимо фрейм прикомірки чи комори і т.п. У цьому мінімальному описі опущені деталі (колір стін, кількість вікон і т.д.) Факт існування подібних деталей для конкретної кімнати враховується тим, що в згаданому описі маються слоти — вакантні клітини пам’яті. На рівні мінімального опису кімнати як прототипу деякого житлового приміщення ці слоти не заповнені. Вони заповнюються (чи прототип обрамляється) конкретними значеннями атрибутів кімнати (кількість вікон, колір стін, висота стель і ін.), якщо має місце деяка конкретна ситуація.

Абстрактний образ (прототип) деякого об'єкта з потенційною можливістю його обрамлення атрибутами називається фреймом. Він дозволяє сконцентрувати всі знання про даний об'єкт (чи клас об'єктів) у єдиній структурі даних (фреймі), а не розпорошувати їх між безліччю більш дрібних структур начебто логічних формул чи продукцій. Декларативні і процедурні знання про деяку сутність укладаються (пакуються) у єдину структуру (фрейм).

Фрейм складається з імені, що виконує роль ідентифікатора, і слотів. Ідентифікатор, що привласнюється фрейму, повинний мати унікальне ім’я, єдине в даній фреймовій (мережній) системі. Кожен фрейм включає довільне число слотів, при цьому деякі з них визначаються самою системою, а інші задаються користувачем.

Завдяки слотам-посиланням («a kind of», «is a», та ін.) фреймові системи утворять ієрархічні структури, що реалізують принцип спадкування інформації. Спадкування відбувається в напрямку «суперклас-підклас», «клас-екземпляр класу». Фрейм, у якому заповнені всі значення слотів, називається фреймом-екземпляром. Існують ще зразки чи фрейми-прототипи. Вони являють собою оболонку, у якій зазначені тільки імена слотів. Інкапсульовані в слоти чи приєднані процедури є важливою особливістю фреймових мереж, що істотно відрізняють їх від мереж семантичних. Ці процедури додають фреймовій системі можливість керування виводом, таким же способом, як це робиться на основі використання об’єктно-орієнтованих мов.

**3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЕКСПЕРТНОЇ СИСТЕМИ**

Для розробки експертної системи вибору клавіатури було обрано мову Prolog. В базі даних клавіатури будуть описані за допомогою наступних характеристик: № товару, виробник, інтерфейс, кількість основиних клавіш, колір, тип та вартість.

Експертна система буде складатись з наступних блоків:

* + Блок бази даних
  + Блок важливості коефіцієнтів
  + Блок прийняття рішень (логічного виводу)
  + Блок виводу запитань для вибору характеристик клавіатур
  + Блок виводу запитань для вибору характеристик клавіатур
  + Блок виводу результату пошуку з усіма характеристиками

**3.1 База знань**

База знань повинна містити відомості, які відносяться до предметної області, в тому числі такі інформаційні компоненти, які 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.

Прості бази знань можуть використовуватися для зберігання даних про організації: документації, інструкцій, статей технічного забезпечення. Головна мета створення таких баз — допомогти менш досвідченішим людям знайти існуючий опис способу вирішення якої-небудь проблеми предметної області.

База знань представлена у вигляді тверджень, які мають наступний вигляд:

keyboard(1,'A4 Tech','USB',104,'Чорний','Звичайна',157).

keyboard(2,'A4 Tech','USB',104,'Чорний','Звичайна',116).

keyboard(3,'A4 Tech','USB',104,'Сріблястий','Ігрова',182).

keyboard(4,'A4 Tech','USB',104,'Чорний','Ігрова',94).

keyboard(5,'A4 Tech','USB',104,'Білий','Звичайна',88).

keyboard(6,'A4 Tech','PS/2',104,'Чорний','Ігрова',50).

**3.2 Блок логічного виводу**

Додатковою властивістю, яке вимагається від експертної системи, є здатність функціонувати в умовах невизначеності і неповної інформації. Інформації про задачу, яка потребує вирішення, можуть бути визначені приблизно.

Процедура get\_all\_answer є блоком обробки відповідей і передає отримані відповіді у блок знаходження результату calculate\_matches.

За допомогою блоку виведення результату, яка реалізована у вигляді процедури result, виводиться найбільш точний результат.

get\_all\_answer(Index):-

CurIndex is 7-Index, ask(CurIndex), NewIndex is Index-1, get\_all\_answer(NewIndex).

get\_all\_answer(0).

**3.3 Блок взаємодії з користувачем**

Блок взаємодії з користувачем працює наступним чином. По черзі, запитуючи користувача кожну характеристику та отримуючи відповіді від користувача.

ask(Index):-

(Index==1, findall(Manufacturer,keyboard(\_,Manufacturer,\_,\_,\_,\_,\_),List), writeln('Виберіть виробника: ');

Index==2, findall(Interface,keyboard(\_,\_,Interface,\_,\_,\_,\_),List), writeln('Виберіть тип підключення: ');

Index==3, findall(Keys,keyboard(\_,\_,\_,Keys,\_,\_,\_),List), writeln('Виберіть кількість клавіш: ');

Index==4, findall(Colour,keyboard(\_,\_,\_,\_,Colour,\_,\_),List), writeln('Виберіть колір: ');

Index==5, findall(Type,keyboard(\_,\_,\_,\_,\_,Type,\_),List), writeln('Виберіть тип: ');

Index==6, writeln('Введіть ціновий діапазон'), write('Мінімальна ціна: '), read(Min), assert(answer(Index,Min)),

write('Максимальна ціна: '), read(Max), nl, assert(answer(Index,Max))),

(Index\=6, list\_to\_set(List,OutList), NewList=['Неважливо'|OutList],

print\_variant(NewList,1), get\_choice(NewList,RES),

assert(answer(Index,RES)), !; true, !).

Результат роботи експертної системи зображено на рисунку 3.1.

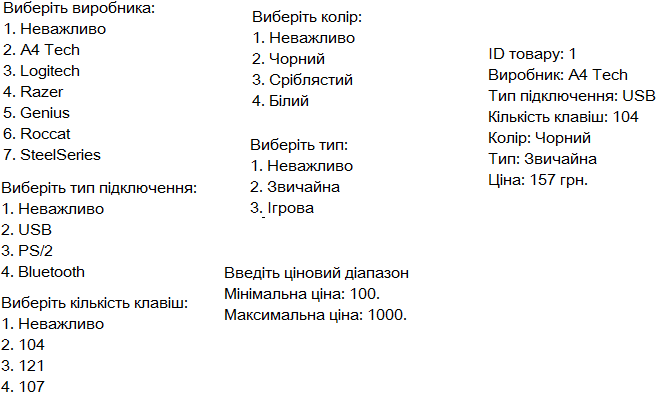


Рисунок 3.1 – Результат роботи експертної системи

**ВИСНОВКИ**

В курсовій роботі було розглянуто експертні системи, методи представлення знань та побудовано експертну систему для вибору клавіатур. Під час тестування експортної системи було доведено, що система працює вірно і є корисною для використання.

**ПЕРЕЛІК ПОСИЛАНЬ**

1. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб.: Питер, 2001. - 384 с.

2. Леоненков А.В. Нечёткое моделирование в среде MATLAB. - СПб : БХВ-Петербург, 2003. - 736 с.

3. Штовба С.Д. Введение в теорию нечетких множеств и нечеткую логику. - 2002.

4. Бакан Г.М. Вступ до теорії експертних систем та баз знань.-К.:ВПЦ «Київський університет», 2005.-90с.

5. Р. Дорф, Р. Бишоп Современные системы управления: пер. с англ.-М.: Лаборатория Базовых Знаний, 2002.-832с.

6. Каллан Р. Основные концепции нейронных сетей. – М.: Вильямс, 2001.-288с.

7. Ануфриев И.Е. Информатика. Пакет МаLab. Изд-во СПбГПУ. 2003. - 67с.

8. С.П. Иглин Математические расчеты на базе MATLAB.-СПб.: БХВ-Петербург, 2005.-640с.

**ДОДАТОК А**

Лістинг коду програми експертної системи для вибору клавіатур.

% Автор: Пилипчук Андрій

% Група: КСМм-51

% ТНТУ

% База знань:

keyboard(1,'A4 Tech','USB',104,'Чорний','Звичайна',157).

keyboard(2,'A4 Tech','USB',104,'Чорний','Звичайна',116).

keyboard(3,'A4 Tech','USB',104,'Сріблястий','Ігрова',182).

keyboard(4,'A4 Tech','USB',104,'Чорний','Ігрова',94).

keyboard(5,'A4 Tech','USB',104,'Білий','Звичайна',88).

keyboard(6,'A4 Tech','PS/2',104,'Чорний','Ігрова',50).

keyboard(7,'A4 Tech','PS/2',121,'Сріблястий','Звичайна',107).

keyboard(8,'A4 Tech','PS/2',104,'Чорний','Звичайна',182).

keyboard(9,'A4 Tech','PS/2',104,'Чорний','Звичайна',54).

keyboard(10,'Logitech','USB',104,'Білий','Звичайна',79).

keyboard(11,'Logitech','USB',121,'Чорний','Звичайна',94).

keyboard(12,'Logitech','USB',104,'Чорний','Звичайна',68).

keyboard(13,'Logitech','Bluetooth',107,'Сріблястий','Звичайна',73).

keyboard(14,'Logitech','USB',107,'Білий','Звичайна',68).

keyboard(15,'Logitech','USB',104,'Чорний','Ігрова',90).

keyboard(16,'Logitech','USB',104,'Білий','Звичайна',45).

keyboard(17,'Logitech','PS/2',121,'Сріблястий','Звичайна',74).

keyboard(18,'Logitech','PS/2',104,'Чорний','Ігрова',98).

keyboard(19,'Logitech','PS/2',107,'Білий','Звичайна',73).

keyboard(20,'Logitech','PS/2',104,'Чорний','Звичайна',90).

keyboard(21,'Razer','USB',104,'Сріблястий','Ігрова',578).

keyboard(22,'Razer','USB',104,'Білий','Ігрова',740).

keyboard(23,'Razer','USB',121,'Чорний','Ігрова',1068).

keyboard(24,'Razer','USB',104,'Чорний','Ігрова',578).

keyboard(25,'Razer','USB',107,'Сріблястий','Ігрова',820).

keyboard(26,'Razer','USB',107,'Чорний','Ігрова',743).

keyboard(27,'Razer','USB',104,'Білий','Ігрова',2418).

keyboard(28,'Razer','USB',107,'Сріблястий','Ігрова',1139).

keyboard(29,'Razer','USB',104,'Чорний','Ігрова',818).

keyboard(30,'Razer','USB',121,'Сріблястий','Ігрова',656).

keyboard(31,'Genius','USB',104,'Білий','Звичайна',75).

keyboard(32,'Genius','USB',107,'Чорний','Ігрова',247).

keyboard(33,'Genius','USB',104,'Білий','Звичайна',212).

keyboard(34,'Genius','USB',107,'Сріблястий','Звичайна',125).

keyboard(35,'Genius','USB',107,'Чорний','Звичайна',75).

keyboard(36,'Genius','Bluetooth',104,'Сріблястий','Звичайна',111).

keyboard(37,'Genius','Bluetooth',104,'Білий','Звичайна',76).

keyboard(38,'Genius','PS/2',107,'Чорний','Звичайна',134).

keyboard(39,'Genius','PS/2',107,'Сріблястий','Звичайна',103).

keyboard(40,'Genius','PS/2',107,'Білий','Звичайна',120).

keyboard(41,'Roccat','USB',121,'Чорний','Ігрова',949).

keyboard(42,'Roccat','USB',121,'Сріблястий','Ігрова',1155).

keyboard(43,'Roccat','USB',107,'Білий','Ігрова',495).

keyboard(44,'Roccat','USB',107,'Чорний','Ігрова',834).

keyboard(45,'Roccat','USB',121,'Сріблястий','Ігрова',371).

keyboard(46,'SteelSeries','USB',107,'Сріблястий','Ігрова',655).

keyboard(47,'SteelSeries','USB',104,'Білий','Ігрова',743).

keyboard(48,'SteelSeries','USB',107,'Чорний','Ігрова',990).

keyboard(49,'SteelSeries','USB',121,'Чорний','Ігрова',289).

keyboard(50,'SteelSeries','USB',104,'Чорний','Ігрова',902).

% Коефіцієнти( важливість тої чи іншої властивості):

koef(1,0.3). % manufacturer

koef(2,0.2). % interface

koef(3,0.1). % keys

koef(4,0.1). % colour

koef(5,0.2). % type

koef(6,0.2). % price

% Блок прийняття рішень:

select\_keyboard:-

retractall(answer(\_,\_)), get\_all\_answer(6),

create\_Matches, calculate\_matches(6),

max\_matches(0,0,ID), result(ID).

max\_matches(Max,IDM,Res):-

matches(ID,M), M>Max, max\_matches(M,ID,R), !, Res=R; Res=IDM.

create\_Matches:-

retractall(matches(\_,\_)), keyboard(ID,\_,\_,\_,\_,\_,\_),

assert(matches(ID,0)), fail.

create\_Matches.

calculate\_matches(0).

calculate\_matches(Index):-

CurIndex is 7-Index, NewIndex is Index - 1, (answer(CurIndex,Ans),!,

((CurIndex==1, keyboard(ID,Manufacturer,\_,\_,\_,\_,\_), Manufacturer==Ans;

CurIndex==2, keyboard(ID,\_,Interface,\_,\_,\_,\_), Interface==Ans;

CurIndex==3, keyboard(ID,\_,\_,Keys,\_,\_,\_), Keys==Ans;

CurIndex==4, keyboard(ID,\_,\_,\_,Colour,\_,\_), Colour==Ans;

CurIndex==5, keyboard(ID,\_,\_,\_,\_,Type,\_), Type==Ans;

CurIndex==6, keyboard(ID,\_,\_,\_,\_,\_,Price), answer(CurIndex,Max), Max>Ans, Price<Max, Price>Ans),

inc(ID,CurIndex), fail; (Ans=='Неважливо', incforall(50,CurIndex); true)), calculate\_matches(NewIndex),!).

incforall(N,Index):-

NN is N-1, inc(N,Index), incforall(NN,Index).

incforall(0,\_).

inc(ID,Index):-

matches(ID,M), retract(matches(ID,M)), koef(Index,K), NM is M+K, assert(matches(ID,NM)).

% Блок взаэмодії з користувачем.

get\_all\_answer(Index):-

CurIndex is 7 - Index, ask(CurIndex), NewIndex is Index-1, get\_all\_answer(NewIndex).

get\_all\_answer(0).

ask(Index):-

(Index==1, findall(Manufacturer,keyboard(\_,Manufacturer,\_,\_,\_,\_,\_),List), writeln('Виберіть виробника: ');

Index==2, findall(Interface,keyboard(\_,\_,Interface,\_,\_,\_,\_),List), writeln('Виберіть тип підключення: ');

Index==3, findall(Keys,keyboard(\_,\_,\_,Keys,\_,\_,\_),List), writeln('Виберіть кількість клавіш: ');

Index==4, findall(Colour,keyboard(\_,\_,\_,\_,Colour,\_,\_),List), writeln('Виберіть колір: ');

Index==5, findall(Type,keyboard(\_,\_,\_,\_,\_,Type,\_),List), writeln('Виберіть тип: ');

Index==6, writeln('Введіть ціновий діапазон'), write('Мінімальна ціна: '), read(Min), assert(answer(Index,Min)),

write('Максимальна ціна: '), read(Max), nl, assert(answer(Index,Max))),

(Index\=6, list\_to\_set(List,OutList), NewList=['Неважливо'|OutList],

print\_variant(NewList,1), get\_choice(NewList,RES),

assert(answer(Index,RES)), !; true, !).

% Блок виводу можливих варіантів:

print\_variant([H|T],Count):-

write(Count), write('. '), writeln(H), NewCount is Count +1, print\_variant(T,NewCount).

print\_variant([],\_).

get\_choice(List,RES):-

read(Choice), choice\_name(Choice,List,RES),!.

choice\_name(Choice,[H|T],RES):-

Choice>1, NewChoice is Choice-1, choice\_name(NewChoice,T,RES); RES=H.

% Блок виводу результату пошуку з усіма характеристиками:

result(ID):-

matches(ID,M), !, (M<0.6, writeln('Вибачте, але підходящого варіанту незнайшлось.\nМожемо запропонувати такий\n');

writeln('Найбільш підходящий варіант\n')), !,

keyboard(ID,Manufacturer,Interface,Keys,Colour,Type,Price),

write('ID товару: '), writeln(ID),

write('Виробник: '), writeln(Manufacturer),

write('Тип підключення: '), writeln(Interface),

write('Кількість клавіш: '), writeln(Keys),

write('Колір: '), writeln(Colour),

write('Тип: '), writeln(Type),

write('Ціна: '), write(Price), writeln(' грн.').