С.М. Кравченко
НПУ ім. М.П.Драгоманова

Інформаційне моделювання на прикладі використання Турбо-Прологу в навчальному процесі

Одна з тенденцій розвитку вищої педагогічної освіти – її орієнтація на широке впровадження сучасних інформаційних технологій в навчальному процесі. Так, останнім часом в курсі інформатики з'явилась тема "Штучний інтелект. Експертні системи". Новизна цієї теми не тільки в шкільному курсіінформатики, а і в педагогічному вузі, відсутність достатньо добре і однозначно розробленої методичної системи її вивчення, приводять, як правило, тільки до викладення загальних питань з проблем штучного інтелекту. Тому, важко не погодитись з деякими дослідниками в цій галузі, що формування знань, умінь і навичок з початків штучного інтелекту повинне йти не тільки на теоретичному рівні, а й на практичному. Оскільки, вивчення предметної галузі для розв'язування практичної задачі та створення її інформаційної моделі на наш погляд дозволяє студентам оволодіти навичками формування на практиці бази знань, простежувати процеси побудови моделі структурної програми і практично створити прикладну експертну систему. Тобто відбувається певне виконання студентами ролі інженера знань при вивченні методів подання знань у конкретній предметній галузі у поєднанні з придбанням нових знань шляхом проведення досліджень при застосуванні інструментальних моделюючих експертних систем. Це дозволяє не тільки значно інтенсифікувати навчальний процес, а й підвищити теоретичний рівень та практичну значущість результатів навчання, зокрема з інформатики в педагогічному вузі. В тісному зв'язку із застосуванням студентами оболонки експертної системи для створення прикладних моделюючих експертних систем, проведенням комп'ютерних експериментів за допомогою створених експертних систем і аналізу їхніх результатів для одержання нових знань у проблемній галузі, поповненням студентами бази знань новими знаннями, надбаними в процесі досліджень, інформаційне моделювання сприяє кращому розумінню й засвоєнню навчального матеріалу. Крім того одержання знань у предметній галузі за допомогою методів інженерії знань є не тільки ефективним, а й має на меті використовуватись для одержання професійних знань підготовки фахівців у різних проблемних галузях, зокрема на прикладі підготовки фахівців з

інформатики в педагогічному вузі.

Подолати вище згадані недоліки можна з використанням в навчальному процесі декларативних мов програмування, а саме – Турбо-Прологу (ТП). Використовуючи ТП, замість детального розписування кроків алгоритму розв'язування задачі програміст записує умову задачі (мовою формальної логіки), а

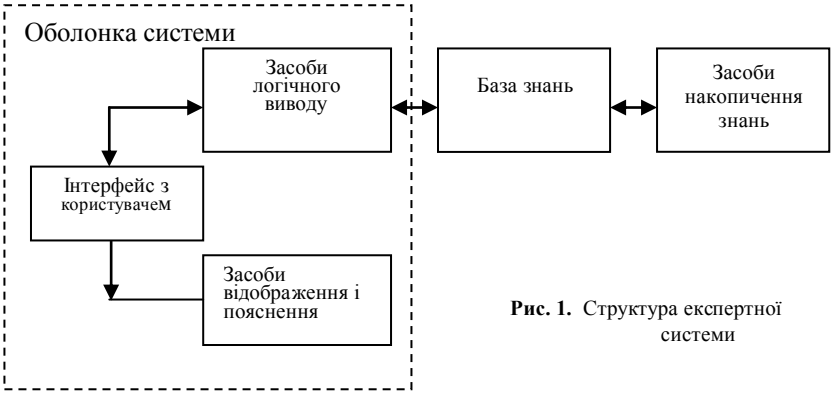

Рис. 1. Структура експертної системи

потужні внутрішні уніфікаційні процедури, самостійно шукають розв'язок поставленої задачі. Цю особливість вдало використовують при створенні експертних систем та експертних оболонок – галузі застосування ТП, що найбільш швидко розвивається та найкраще відображає всі його можливості. Тому наступний матеріал, на наш погляд, є доречним для засвоєння деяких питань і розширення рамок вивчення теми початків штучного інтелекту.

Експертна система (ЕС) - це обчислювальна система, яка здатна використовувати знання про деяку предметну галузь і приймати рішення в рамках цієї галузі знань на рівні експерта-професіонала. Такий ефект досягається завдяки тому, що експертна система у своїй роботі відтворює приблизно ту ж саму схему міркувань, яку використовує людина-експерт, аналізуючи проблему. Проблемна спрямованість і можливість розв'язувати широкий клас задач зробили ЕС незамінними в

багатьох галузях промисловості, виробництва, науки, в навчальному процесі.

Як системи штучного інтелекту, вони використовують знання висококваліфікованих спеціалістів-експертів для розв'язування задач в порівняно вузьких проблемних галузях. Ці знання використовує інженер знань при проектуванні конкретної експертної системи, що знаходить відображення у побудові моделі предметної галузі, виборі ефективних способів подання знань і механізмів виведення. Структуру ЕС подано на рис. 1.

Розглянемо принцип дії найпростішої експертної системи, що відповідала б вище згаданій схемі і була створена засобами ТП.

Подання знань. Усяка предметна (проблемна) галузь діяльності може бути описана у вигляді сукупності відомостей про структуру цієї галузі, основні її характеристики, процеси, що протікають у ній, а також про способи розв'язування задач, що тут виникають. Усі ці відомості й утворять знання про предметну галузь. Щоб мати можливість розв'язувати задачі, необхідно зібрати потрібні знання про предметну галузь і створити її *інформаційну модель.*

Процес логічного виводу – це одна з форм того, як людина приходить до певного висновку, використовуючи систему знань про деяку предметну галузь. Тому в інтелектуальних системах прагнуть відобразити основні особливості людських міркувань, професійні уміння фахівців, досвід яких поки не цілком доступний штучним системам.

Для швидкого і ефективного доступу до знань в пам'яті

комп'ютера їх, як правило, подають у вигляді деякої схеми або моделі. Використовують чотири основні моделі знань: семантичні мережі, фрейми, логічні системи, продукції. На сьогодні найбільш популярні ЕС, що базуються на правилах або продукціях "якщо – то". Представлення знань в такий спосіб дає можливість природним чином об'єднувати в групи зв'язні фрагменти знань, що в свою чергу дозволяє розширювати базу знань, "навчати" експертну систему, демонструвати штучний інтелект.

Припустимо, що необхідно представити базу знань про якусь предметну галузь. При цьому припускають, що будь-який об'єкт (*object*) може належати до деякого класу об'єктів (тобто наслідувати властивості цього класу) і мати позитивну чи негативну відповіді на деякі твердження (*property*), тобто мати чи не мати свої власні властивості.

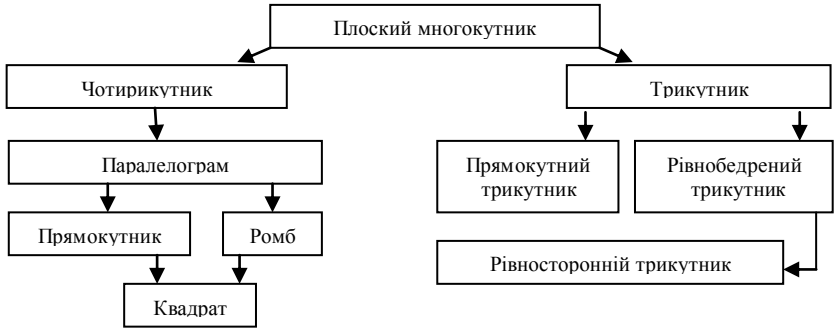

Рис. 2

Створення бази знань проілюструємо на прикладі визначення видів плоских многокутників, з числом кутів не більше за чотири. Деревовидна (ієрархічна) структура бази знань експертної системи подана на рис. 2.

Тепер спробуємо описати базу знань засобами ТП. Для цього введемо предикати:

object ("назва") – предикат визначення об'єкту (перелік властивостей, що дають можливість однозначно ідентифікувати об'єкт);

property (“властивість”, “позитивність”) – предикат, що має певне значення “позитивності” по відношенню до об’єкта – істинне чи хибне.

Продемонструємо на прикладах, як базу знань на рис. 2 можна подати за допомогою вищенаведених предикатів. При її створенні будемо дотримуватись наступної вимоги – спочатку будемо описувати об’єкти, які стоять на нижчому рівні ієрархії. Тобто, спочатку квадрат → ромб → прямокутник → паралелограм → чотирикутник, що пов’язано з методами пошуку розв’язку засобами ТР.

```
object(”квадрат”) :-
object (”ромб”),
object (”прямокутник”).

object (”ромб”) :-
object (”паралелограм”),
property (”має всі сторони рівні”, ”істинне”).

object (”прямокутник”) :-
object (”паралелограм”),
property (”має всі кути рівні”, ”істинне”).

object (”прямокутний трикутник”) :-
object (”трикутник”),
property (”один з кутів дорівнює 90 градусів”,
”істинне”).

object (”рівносторонній трикутник”) :-
object (”рівнобедрений трикутник”),
property (”всі сторони рівні”, ”істинне”).

object (”рівнобедрений трикутник”) :-
object (”трикутник”),
property (”дві сторони рівні”, ”істинне”).

object (”паралелограм”) :-
object (”чотирикутник”),
property (”об’єкт має паралельні протилежні
сторони”, ”істинне”).

object (”трикутник”) :-
not (object (”чотирикутник”)).

object (”чотирикутник”) :-
property (”об’єкт є многокутником, у якого чотири
сторони і чотири кути”, ”істинне”).
```

Засоби логічного виводу. Розглянемо ідентифікацію досліджуваного об’єкта щодо нашої бази знань на прикладі прямокутного трикутника. Будемо вважати, що наш трикутник не є одночасно рівнобедреним і прямокутним, оскільки для опрацювання такого випадку необхідно було б додати наступне правило:

```
object (”прямокутний, рівнобедрений трикутник”) :-
object (”прямокутний трикутник”),
```

object (”рівнобедрений трикутник”).

Для ідентифікації ТР спочатку звертається до першого предиката *object* в базі знань. В даному випадку це правило визначення квадрата. Квадрат визначається через ромб, а той в свою чергу через правило опису чотирикутників. Оскільки в резидентній базі відповідь на питання “об’єкт є многокутником, у якого чотири сторони і чотири кути?” відсутня, тому користувачеві пропонується дати на нього відповідь. Прямокутний трикутник не має наведену властивістю, тому в резидентній базі знань запам’ятовується твердження “*об’єкт є многокутником, у якого чотири сторони і чотири кути”* з позначенням позитивності “*хибне*”. Що автоматично виключає з ідентифікації всі об’єкти, пов’язані з чотирикутниками.

Отже, наступним предикатом, до якого звернеться ТР, буде предикат визначення прямокутного трикутника. Оскільки поняття трикутника вже є визначеним відповіддю на попереднє питання, тому ТР одразу переходить до другого питання “один з кутів дорівнює 90 градусів?**”**. Отримавши позитивну відповідь знайдено об’єкт, властивості якого відповідають вимогам користувача – тобто об’єкт повністю буде ідентифіковано.

Далі управління програмою надається предикату *пояснення*, що виводить на екран список питань заданих користувачеві з зазначенням позитивності відповідей на них.

Якщо в базі знань не буде знайдено об’єкту з властивостями, на які вказували б відповіді користувача, виводиться повідомлення про неможливість ідентифікування.

Отже, отримання користувачем певного результату робиться на основі співставляння зафіксованого в резидентній базі даних списку відповідей на питання з існуючою базою знань. Список відповідей створюється в залежності від характеру відповіді на питання за допомогою вбудованого предиката ТР *assertz(“текст”)*. Цей предикат дозволяє зберігати факти в оперативній пам’яті. Використаємо його для запам’ятовування твердження з позначенням його позитивності чи негативності по відношенню до об’єкта. Для цього використаємо предикати, що забезпечуватимуть процес виводу:

list_answer (“твердження”, ”позитивність”) – використовується для співставлення отриманих даних з існуючою базою знань (його оголошення відбувається в розділі *DATABASE)*;

add_ list_answer (“твердження”, “відповідь”, “позитивність”) – призначений для додання до списку тверджень, отриманих від користувача, з поміткою відповіді на дане твердження “так” чи “ні”.

ask (“твердження”, “позитивність”) – виводить на екран запитання про істинність чи хибність твердження, після чого передає управління *add_list_answer*.

Наведемо приклад використання вищезгаданих предикатів.

```
ask ( Текст, Позитивність ) :-
write ( Текст, ' ','?' ),
readln ( Відповідь ),
add_ list_answer ( Текст, Відповідь, Позитивність).
add_ list_answer (Текст, так, "істинне") :-
assertz ( list_answer  ( Текст, "істинне" ) ).
add_ list_answer (Текст, ні, "істинне") :-
assertz ( list_answer  ( Текст, "хибне" ) ), fail.
add_ list_answer (Текст, так, "хибне") :-
assertz ( list_answer  ( Текст, "істинне" ) ) , fail.
add_ list_answer (Текст, ні, "хибне") :-
assertz ( list_answer  ( Текст, "хибне" ) ).
```

Розглянемо дію предиката *add_ list_answer*. В залежності від того, як користувач відповідає на питання, до списку відповідей додається твердження за таким правилом: властивість разом з її "позитивністю" – при умові збігу відповіді з "позитивністю" твердження, що визначена експертом при складанні бази знань; властивість із запереченням "позитивності" в протилежному випадку.

Використовуючи вище наведений матеріал організуємо відповідну експертну систему (див. лістинг 1). Самостійно наповнивши базою знань з будь-якої теми за вище наведеними прикладами, читач може перевірити її в роботі.

Лістинг 1.

```
Database
  list_answer(symbol,symbol)

Predicates
  start
пояснення
  object(symbol)
  property(symbol,symbol)
  del_list_answer
  add_list_answer (symbol,symbol,symbol)
  ask(symbol,symbol)

Goal
  start.

Clauses

  start:-
        object(X),!,
        clearwindow,
        write("Ваш об'єкт ",X, " оскільки"),
        пояснення.

   /* дії на випадок не співпадання відповідей
користувача з жодним об'єктом бази знань */

  start:-
        clearwindow,
        write("Ідентифікація неможлива"),
        del_list_answer.
```

```
/* перевірка існування в динамічній базі знань
            відповіді на питання */

    property(Текст,"істинне"):-
list_answer(Текст,"істинне").
    property(Текст,"істинне"):-
list_answer(Текст,"хибне"),!,fail.
    property(Текст,"хибне"):-
list_answer(Текст,"хибне").
    property(Текст,"хибне"):-
list_answer(Текст,"істинне"),!,fail.
    property(Текст,Позитивність):-
not(list_answer(Текст,Позитивність)),
ask(Текст,Позитивність),!.

 ask(Текст,Позитивність):-
       clearwindow,
       write(Текст,"?"),
       readln(Відповідь),
 add_list_answer (Текст,Відповідь,Позитивність).

 /* виведення на екран пояснень отримання
             результатів */
пояснення:-
       list_answer(Текст,"Істинне"),
       write("Істиним є твердження: ",Текст),nl,
       fail.
 пояснення:-
       list_answer(Текст,"Хибне"),
       write("Хибним є твердження: ",Текст),nl,
       fail.
 пояснення:-
       del_list_answer.

 /* формування резидентної бази знань */

 add_list_answer (Текст,так,"істинне"):-
       assertz(list_answer (Текст,"істинне")).
 add_list_answer (Текст,ні,"істинне"):-
       assertz(list_answer(Текст,"хибне")),fail.

 add_list_answer (Текст,так,"хибне"):-
       assertz(list_answer
(Текст,"істинне")),fail.
 add_list_answer (Текст,ні,"хибне"):-
       assertz(list_answer (Текст,"хибне")).
 /* вилучення з динамічної бази знань списку
             відповідей */

 del_list_answer:-
       retract(list_answer(_,_)),fail.

       /* далі повинна йти база знань */
```

Основною перешкодою на шляху створення і поширення експертних систем є проблема набуття і подання знань. Значна кількість інформації, її слабка структурованість, труднощі експерта при спробі пояснити послідовність своїх міркувань при прийняті певного рішення призводять до того, що процес передавання знань від експерта до системи здійснюється з великими труднощами, методом “спроб, помилок і виправлень”. Виникає потреба у забезпеченні більш ефективної роботи

експерта при роботі з базою знань, яка використовується експертною системою. На допомогу приходять експертні оболонки.

Експертна оболонка (ЕО) – середовище, основою якого є механізм виводу і порожня база знань, а також деякі сервісні програми (редактор бази знань, засоби відображення, трасування). Інколи експертні оболонки ще називають експертними системами другого покоління.

Прикладом найпростішої ЕО, створеної засобами ТР, може виступати лістинг наступної програми:

Лістінг 2.

```
Domains
    sp=integer*
Database
    dquestion(integer,symbol,symbol)
    dobject(symbol)
    question(integer,symbol)
    object(symbol,sp)
Predicates
    start
    menu(char)
    define
    add_list_answer(integer,symbol,symbol)
    ask(sp)
    next(symbol,symbol)
    пояснення
Goal
    start.
Clauses

    start:-
          retract(dquestion(_,_,_)),
          fail.

    start:-
          retract(dobject(_)),
          fail.
    start:-
          clearwindow,!,
          write("1-ідентифікація"),nl,
          write("2-зчитування бази даних"),nl,
          write(”3-добавити питання”),nl,
  write("натисніть відповідну клавішу або Enter для
виходу:"),
          readchar(Key),
          menu(Key).

          /* правила для опрацювання меню */
    menu('1'):-
          !,clearwindow,
          define,
          start.

  /* зчитування з файлу в динамічну базу знань  */
    menu('2'):-
          clearwindow,
          write("введіть ім'я файлу: "),
          readln(File),
          consult(File),
```

```
        start.

    /* опрацювання натиснення клавіші Enter */
    menu(_):-exit.

    /* у разі неуспіху звернення до наступного
              предикату object */
    define:-
           object(Name,List),
           not(dobject(Name)),
           ask(List).

  /* у разі відсутності досліджуваного об'єкту в
       БД пропонується його добавити */
    define:-

    findall(Number,dquestion(Number,_,"y"),List),
           not(List=[]),
           not(object(_,List)),
           write("ідентифікувати неможливо, невідомий
об'єкт"),nl,
           readchar(_),
           write("введіть назву об'єкту"),nl,
           readln(Text),
           assertz(object(Text,List)),
           start.

        /* дії на випадок порожньої БД */
    define:-
           write("далі ідентифікувати неможливо"),nl,
           readchar(_),
           start.


 /* пошук питань на які не була дана відповідь та
          вивід їх на екран */
    ask([First|List]):-
            dquestion(First,_,"y"),
            ask(List).

    ask([First|_]):-
           not(dquestion(First,_,"n")),
           not(dquestion(First,_,"y")),
           question(First,Text),
           write(Text,"?"),
           readln(Answer1),
           add_list_answer(First,Text,Answer1).

    ask([First|_]):-
            dquestion(First,_,"n"),
            fail.

    ask([]):-fail.

  /* створення в ДБД списку відповідей на питання
                        */
    add_list_answer(Number,Text,"y"):-
           assertz(dquestion(Number,Text,"y")),

    findall(Number1,dquestion(Number1,_,"y"),List),
           object(Name,List),
           not(dobject(Name)),
```

```
            write("Ваш об'єкт ",Name,", оскільки"),nl,
            пояснення,
            write("продовжити ідентифікацію? "),
            readln(Answer2),
            next(Answer2,Name).

      add_list_answer(Number,Text,"n"):-
            assertz(dquestion(Number,Text,"n")),fail.

   /* правила для вибору продовження ідентифікації
                              */
      next(Answer,_):-
            Answer="n".
      next("y",Name):-
            assert(dobject(Name)),
            fail.
      /* виведення на екран пояснень отриманого
                  результатів */
пояснення:-
            dquestion(_,Text,_),
            write("Істинним є твердження: ",Text),nl,
            fail.
      пояснення.
```

Після запуску на виконання цієї програми ТР починає роботу з опрацювання предиката *start*, зазначеного в розділі цілі, в результаті виконання якого на екрані з'являється перелік пунктів меню доступних для користувача:

```
1-ідентифікація
2-зчитування бази даних з файлу
натисніть відповідну клавішу або Enter для виходу:_
```

Вище наведена експертна оболонка не містить в собі базу знань (як ЕС на Лістингу 1), а користується динамічною базою знань (ДБЗ). Тому необхідно спочатку звернутись до пункту 2 для її наповнення. Отримавши вказівку у вигляді символу '2', ТР опрацьовує її за допомогою серії предикатів *menu*. При знаходженні правила *menu*('2') на очищеному екрані з'являється запит:

```
введіть ім'я файлу:_.
```

Необхідно ввести ім'я файлу, що містить базу знань. Така база знань може бути створена будь-яким текстовим редактором під DOS та повинна містити предикати:

question ("номер", "текст_питання") – містить номер під яким буде зберігатись в ДБЗ питання та його текст, що з'являтиметься на екрані перед користувачем;

object ("назва", "список_номерів_питань") – предикат визначення *назви* об'єкту з зазначенням переліку властивостей (у вигляді списку номерів питань), що дають можливість однозначно ідентифікувати об'єкт.

Якщо вказаний файл існує на диску, його вміст буде доданий до ДБЗ за допомогою стандартного предикату ТР *consult* (ім'я_файлу). Після чого управління знову буде передано

предикату *start* і програма готова до проведення експертної консультації.

Процес виводу починається з опрацювання правил для предикату *define*. До якого передається список питань з першого факту *object*. Почергово питання виводиться на екран. При цьому відповідь на кожне питання обробляється предикатом *add_list_answe*.

У разі позитивної відповіді до ДБЗ додається задане питання з зазначенням позитивності. Після чого відбувається перевірка списку позитивних відповідей, що створюється за допомогою предиката *findall*, з списками об'єктів в базі знань. Якщо знайдено відповідність, назва об'єкту виводиться на екран. Щоб ЕО була універсальною і могла ідентифікувати об'єкти та їх підмножини. Наприклад, якщо маємо наступну базу знань:

```
question(1,"є многокутником, що має чотири кути")
question(2,"протилежні сторони паралельні")
object("чотирикутник",[1])
object("паралелограм",[1,2])
```

Після позитивної відповіді на питання під номером 1, експертна оболонка виведе повідомлення "Ваш об'єкт чотирикутник". Але "паралелограм" теж задовольняє позитивна відповідь на перше питання. Саме для відшукання підмножин об'єктів виводиться запитання "продовжити ідентифікацію?". За опрацювання цієї події відповідає предикат *next*. Він у разі погодження додає до ДБЗ факт *dobject("назва")*, що повинен виключити з наступного пошуку вже знайдений об'єкт.

Якщо ж користувач задоволений знайденим результатом і відмовляється продовжувати ідентифікацію, ЕО повертається до початкового стану. На екрані з'являється вище наведене меню, але перед цим ТР вилучить з ДБЗ всі питання та виключені з ідентифікації об'єкти.

Таблиця 1.

для вилучення питання:	для вилучення правила:

`Menu('5'):-`
`clearwindow,`
`write("введіть номер питання, що треба вилучити:"),`
`readint(Number),`
`retract(Number),`
`start.`
	`menu('6'):-`
	`clearwindow,`
	`write("назву об'єкту, що треба вилучити:"),`
	`readln(Name),`
	`retract(object(Name,_)),`
	`start.`

для добавлення питання	для добавлення правила

```
Menu('7'):-
clearwindow,
write("введіть текст
питання"),
readln(Text),
write("введіть номер під
яким зберегти питання"),
readint(Number),
assertz(question(Number,T
ext)),
start.
```

для запису БД на диск:

```
Menu('3'):-
clearwindow,
write("введіть ім'я
файлу: "), readln(File),
save(File),
start.
```

для перегляду БД

```
Menu('4'):-
clearwindow,
question(Number,Text),
write(Number,"
",Text),nl,
fail.

menu('4'):-
object(Text,List),
write(Text," ",List),nl,
fail.

menu('4'):-
readchar(_),nl,
start.
```

```
menu('8'):-
clearwindow,
write("введіть назву об'єкта"),
readln(Text),
spisok(Text,[]).

spisok(Text,SP):-
clearwindow,!,
write("введіть номер питання,
позитивна відповідь на яке влаштовує
об'єкт:"),
readint(Number),
write("будете продовжувати поповнювати
список відповідей?"),
readchar(Answer),
proverka(Answer,Text,[Number|SP]).

proverka(Answer,Text,SP):-
Answer='y',!,
spisok(Text,SP).

proverka( ,Text,SP):-
обернений(SP,[],SP1),
assert(object(Text,SP1)),
start.

обернений([],L,L).

обернений([X|Y],Z,L):-
обратный(Y,[X|Z],L).
```

Щоб перевірити роботу програми, створіть файл, що міститиме деяку базу знань. Так БЗ, зображена на рис. 2, може бути подана наступним чином:

```
question(1,"об'єкт є многокутником, у якого чотири кути").
question(2,"об'єкт є многокутником, у якого три кути").
question(3,"має протилежні паралельні сторони").
question(4,"дві сторони рівні").
question(5,"всі сторони рівні").
question(6,"всі кути рівні").
question(7,"один з кутів дорівнює 90 градусів").

object("чотирикутник",[1]).
object("паралелограм",[1,3]).
object("ромб",[1,3,5]).
object("прямокутник",[1,3,6]).
object("квадрат",[1,3,5,6]).
object("трикутник",[2]).
object("прямокутний трикутник",[2,7]).
object("рівнобедрений трикутник",[2,4]).
object("рівносторонній трикутник",[2,5]).
object("рівносторонній трикутник",[2,6]).
```

Щоб розширити можливості ЕО, слід додати в експертну оболонку пункти меню, що відповідають за збереження та редагування ДБЗ, та на практиці переглянути їх роботу. Для

цього можна використати фрагменти програми, подані в таблиці 1.

ЛІТЕРАТУРА:

1. Толковый словарь по искусственному интелекту / Авторы – составители А.Н.Аверкин, М.Г.Гаазе-Рапопорт, Д.А.Поспелов. – М.: Радио и связь, 1992. – 256 с.
2. Ин Ц., Соломон Д. Использование Турбо-Пролога: Пер с англ.. – М.: Мир, 1993. – 608 с.