Содержание

Введение

Экспертные системы – это перспективная технология, актуальная в наше время. Основная задача ЭС - создание программ (ПО), которые решают невыполнимые для человека (специалиста-эксперта) задачи. Зачастую, полученный благодаря ПО результаты намного точны, чем результаты, полученные специалистом-экспертом. ощутимые преимущества ЭС это постоянная работа, передача и воспроизведение и стабильность результатов. Подробнее об Экспортной Системе, а также и о его реализации на SWI-Prolog мы с вами рассмотрим в следующих главах проекта.

1.ПОНЯТИЕ ЭКСПОРТНОЙ СИСТЕМЫ

1.1. Экспертная Система(ЭС)

К началу 1980-ых годов знание в сфере искусственного интеллекта было достаточно, чтобы для него появилось отдельное направление. Оно получило название "экспертная система" (ЭС). Основная задача ЭС - создание программ (ПО), которые решают невыполнимые для человека (специалиста-эксперта) задачи. Зачастую, полученный благодаря ПО результаты намного точны, чем результаты, полученные специалистом-экспертом. ЭС применяют в основном для решения так называемых «неформализованных» задач. [2] Для них характерны такие признаки как:

Нечисловая форма записи;

отсутствие алгоритма решения задач;

если присутствует алгоритм решения, то его нельзя использовать;

ресурсы ограничены (ограничение в памяти или во времени).

Кроме всего перечисленного, эти задачи склонны к ошибочности, неполноте, неоднозначности и т. п. К тому же возникают противоречия в начальных данных задачи. Экспертная система – это система, применяющие знания для обеспечения качественного решения задач в определённой области.

1.2 Структура ЭС

1) База знаний (БЗ). (накапливается во время работы ЭС)

2) Механизм вывода (МВ).

3) Система пользовательского интерфейса (СПИ).

База знаний – центральная часть экспертной системы. Она содержит правила, описывающие отношения или явления, методы и знания для решения задач из области применения системы.[3]

Механизм вывода - специальный инструмент работающий с базой знаний. Она содержит в себе множество принципов и правил работы с БЗ. Выводит заключения (выводы) из информации.[3]

Интерфейс – специальное программное средство, которое взаимодействует с пользователем. Известно, что пользователь мало имеет представлений о базе знаний и о её организации. В этом поможет интерфейс. [3]

Доступность знаний – отличительная черта ЭС от обычных программ.

Определения основных свойств ЭС:

1) Использование в решении трудностей, связанных с опытом более высокого класса. Это уровень для более квалифицированных специалистов. Решения задач более точены и эффективны. [2]

2) Возможность строить прогнозы, благодаря чему ЭС выводит ответы не только для одного случая, но и даёт возможность увидеть, как меняется ответ в других случаях и указывает причины этих изменений.

3) Полезное качество как память и входящий в ЭС базы знаний, созданный группой специалистов, представляет собой правило для этих людей. Набор знаний и мнений становиться источником более объективных и успешных стратегий и методов. Если даже по какой-нибудь причине специалист покидает организацию, его опыт остаётся и им пользуются другие специалисты или новички.

4) ЭС также применяют для набора опыта работников. Благодаря этому, обеспеченный богатым багажом знаний и опытом новый сотрудник может применить его на практике при разработки новых стратегий и т п.

1.3. Работа экспертных систем

В построении и работе ЭС относятся:

1) Экспертная Система.

2) Эксперты

3) Инженеры

4) Инструменты создания ЭС.

5) Пользователи.

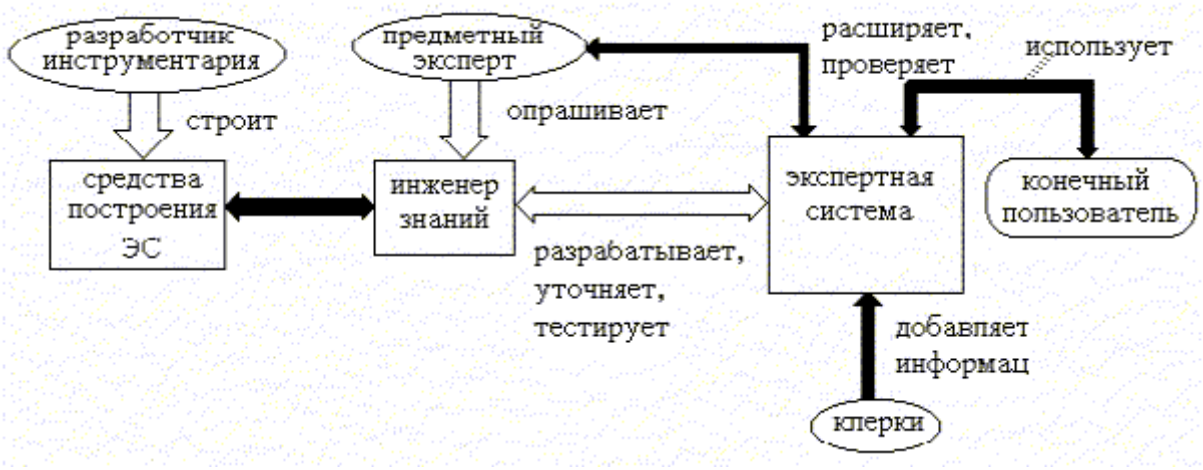

Рис. 0. Структура ЭС.

Экспертная система - программное средство, использующее методы, стратегии экспертов, для качественного и эффективного решения задач в интересующей пользователя предметной области. ЭС это не только программа, но и система, включая в себя БЗ, средство поддержки и т. п Также, есть некоторые компоненты помогающие пользователю работать с программой. [2]

Эксперт – человек-специалист, умеющий правильно делится мыслями и давать правильные решения проблем. Эксперт благодаря таким качествам как ухищрённость и находчивость создаёт стратегию для ЭС.

Инженер знаний - человек, знающий информатику, работу с искусственном интеллектом и имеющий опыт в строительстве ЭС. Помощник программиста в написании программ. [2]

Средство построения ЭС - ПО, применяемое инженером знаний или программистом для построения ЭС. Отличие от простых ПО в том, что они удобны в представлений понятий высокого класса.

Пользователь – человек (или юзер в современном понимании) пользующий готовым ЭС. Профессия пользователя может быть разной: от юриста до студента, изучающий ЭС. [2]

1.4. Преимущества ЭС

Можно выделить ощутимые преимущества ЭС:

Постоянная работа. Человек не всегда может работать качественно и перерывы в работе могут негативным образом отразится на него.

Передача и воспроизведение. Этот процесс на много лёгок и удобен для искусственного интеллекта чем для специалисту-эксперту передающий решение другому человеку.

Стабильность результатов. Специалист-эксперт может исказит результаты при неблагоприятных для него условиях и т. п. На ЭС не влияют подобные факторы и благодаря этому результаты остаются без изменений.

2. PROLOG – ЯЗЫК РАЗРАБОТКИ

2.1 Понятие Prolog

Любой язык программирования всегда основан на определенные типы задач, при решении которых он наиболее эффективен. Существуют задачи, связанные с построением систем для искусственного интеллекта – экспертные системы(ЭС), программы занимающийся переводом слов и т. п. Всё это характерно для языка PROLOG. [1]

К языку программированию Prolog интерес, то возникал, то затухал: энтузиазму, присущий ему, противодействовал жёсткое неприятие. Пик популярности Prolog приходился на 80-е годы XX века в Японии. Японские учёные и программисты верили, что за этим языком будущее. [1]

Работа над языком началась в 1970 г. французскими программистами Аланом Кулмероэ и Филиппом Русселом в университете Марселя. Цель создания Prolog – получения языка, который на основе полученного текста делать логические заключения. Название Prolog - это аббревиатура от выражения PROgrammation en LOGique (PROLOG) и первая реализация этого языка с использованием компилятора Николауса Вирта "AlgolW" была закончена в 1972 году.

С момента появления языка Prolog и до наших дней появились множество разновидностей этого языка. Среди основных разновидности выделяют SWI-Prolog, JIProlog, YAP-Prolog и т. д. Среди них особый интерес у нас вызывает SWI-prolog. Рассмотрим его поподробнее.

2.2 SWI-Prolog – популярная реализация Prolog.

Чем же привлекло нас внимание эта разновидность Prolog? SWI-Prolog – стабильная и удобная стандартная реализация Prolog, которая ориентирована в первую очередь на науку (а именно на научные исследования) и образование. Также возможное использование и в коммерческой деятельности. Для SWI-Prolog доступны версии под известными нами ОС (Windows, Linux и прочее разновидности ОС Unix). Интересно что с 2009 по 2012 годы количество скачиваний в среднем более 10 тыс. в месяц (рис. 1). Это говорит о его популярности и легкодоступности этой версии Prolog.[1]

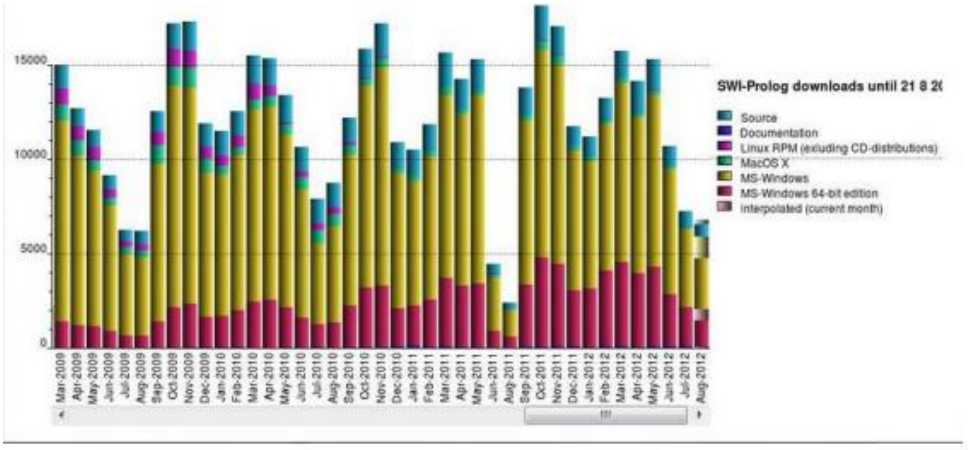

Рис.1 Количество скачек SWI-Prolog за 2009-2012 годы.

Если посмотреть ещё и за период 2001-2005 годы, то можно убедится, что популярность этого языка увеличилась в 2 раза. И этот рост популярности был на фоне высказываний типа «Prolog умер» и т.п.

С 1987 года (т. е. с момента появления SWI-Prolog) по сегодняшний день язык развивается и продвигается по всему миру. Создатель языка - Ян Вилмекер. Название SWI произошло от названия научной группы Sociaal-Wetenschappelijke Informatica («Social Science Informatics»), Амстердамского университета, где научным сотрудником является Вилмекер (в скорее переименованный в HCS (Human-Computer Studies)).

Наименование	Усл. компиля-ция	Сокеты	Многопоточ-ность	HTTP клиент	HTTP сервер	HTML парсер	Адрес сайта в Интернете
LPA-Prolog		+		+	+	+	www.lpa.co.uk/win.htm
SICStus Prolog	+	+	+				www.sics.se/projects/sicstus-prolog-leading-prolog-technology
SWI-Prolog	+	+	+	+	+	+	www.swi-prolog.org/
Visual Prolog	+	+	+	+		+	www.visual-prolog.com/
XSB		+	+	+			xsb.sourceforge.net
YAP-Prolog	+	+	+				www.dcc.fc.up.pt/~vsc/Yap/

Рис.2. Сравнений характеристик SWI-Prolog с другими разновидности языка Пролога

2.3 Среда разработки PIE и система SWI-Prolog

Среда PIE (Prolog Inference Engine) — пример программы, сделанный на VP, которая входит в поставочный дистрибутив. Также PIE является классическим интерпретатором для языка Prolog. С его помощью изучают Prolog и экспериментируют, не заботясь о разных возможностях VP. [1]

PIE представляет собой простую и легкую в использовании систему, которая имеет простой, стандартный для Windows, стиль многооконного пользовательского интерфейса, а также встроенный текстовый редактор с подсветкой синтаксиса (рис. 3).

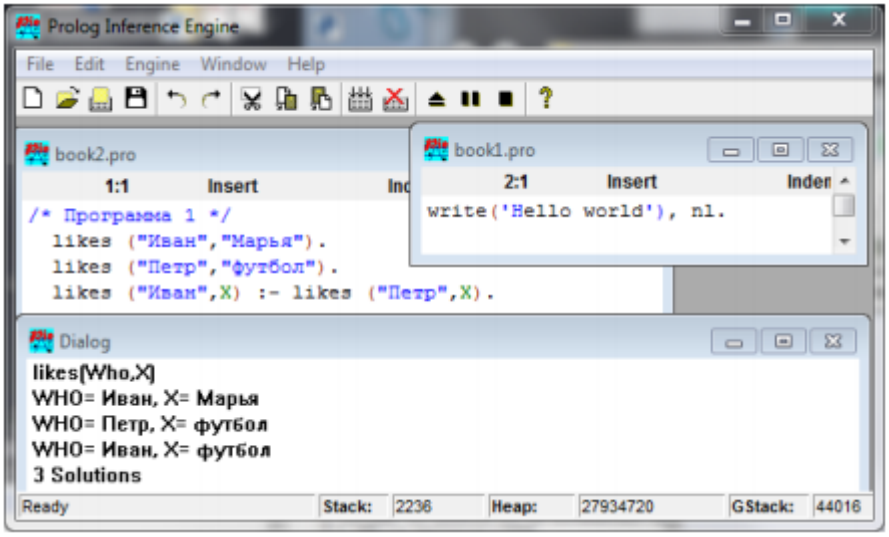

Рис.3. Графический редактор среды PIE.

SWI-Prolog позволяет разрабатывать различные приложения, включая Web-приложения и параллельные вычисления. Сам SWI-Prolog — это консольное приложение, но в его дистрибутив входят GUI тулкит XPCE и встроенный в среду текстовый редактор PceEmacs. Он поддерживает автоматические отступы, подсветку и проверку синтаксиса, отладку и многое другое.[1]

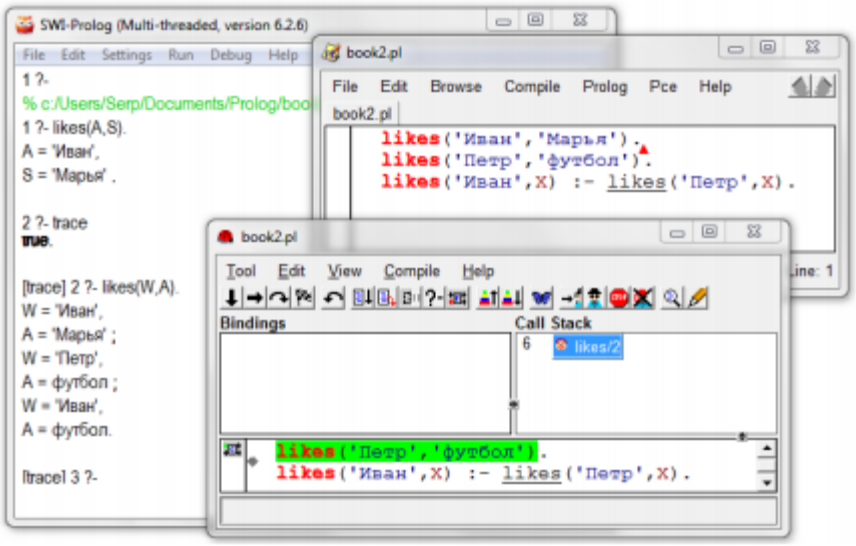

Рис. 4. Консоль SWI-Prolog. Окна редактора и отладчика интерфейса PceEmacs.

Однако на начальном этапе работы с Prolog представляется более целесообразным работа в более простой среде SWI-Prolog-Editor. Это среда программирования для SWI-Prolog, включающая редактор программ с подсветкой синтаксиса, интерпретатор и отладчик программ (рис. 5). Основным назначением этой среды является обучение логическому программированию на языке Prolog. [1]

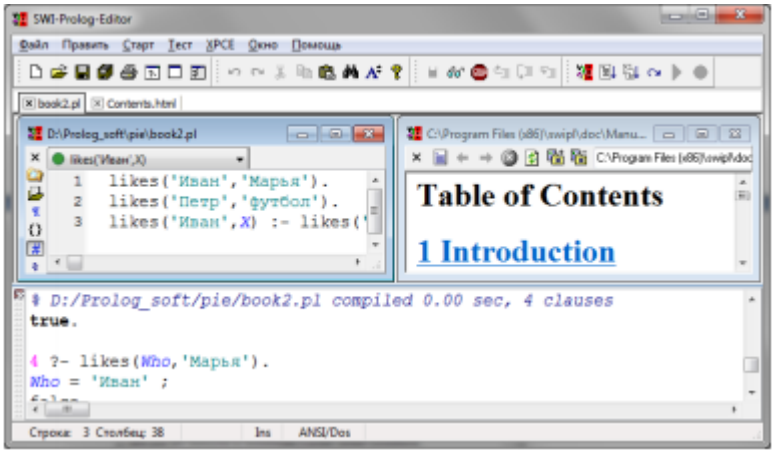

Рис. 5. Графический интерфейс среды SWI-Prolog-Editor.

2.4 Помощь в среде SWI-Prolog

Одной из главных причин популярности этой среды языка Prolog, особенно в учебных и образовательных целях, является то, что он достаточно хорошо документирован. В Интернете имеется большое количество учебных курсов, учебников и примеров программ. Правда, основное их число на английском языке, но и на русском языке за последние три года существенно увеличилось. Говоря о хорошей документированности SWI-Prolog и его клона SWI-Prolog-Editor, естественно, в первую очередь мы имеем в виду

наличие внутренней справки. Она поставляется вместе с версией продукта и органично связана со средой программирования. [1]

Рис. 6. Окно Help в SWI-Prolog-Editor.

Если интересует нужный нам предикат или конструкция языка Prolog, то лучше использовать опцию меню Помощь –-> Поиск (F1), которая позволит найти те разделы справки, где есть ссылки на эту конструкцию. Получить нужную информацию можно, если открыть соответствующий файл.

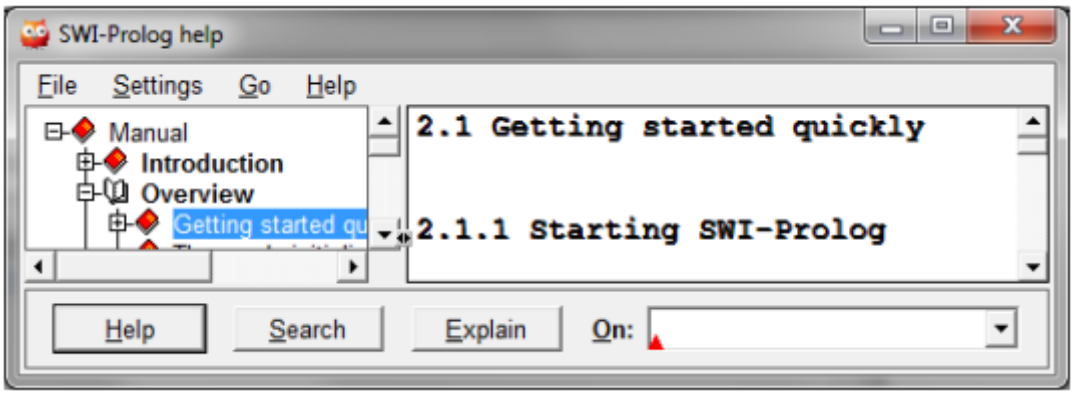

Рис. 7. Окно справки SWI-Prolog.

В случаи если вам нужна помощь вы можете ввести help(предикат) или более удобнее help. Появится окно справки (рис. 7) в котором можно найти информацию нужной конструкции SWI-Prolog.

3 РАЗРАБОТКА ЭКСПЕРТНЫХ СИСТЕМ В PROLOG

Одно из наиболее быстро продвигающего направления Prolog построение Экспертных Систем. В настоящее время использование языка Prolog (в том числе их подвидов: SWI-Prolog, Visual Prolog) в ЭС весьма популярна.

Кратко возвращаемся к понятию экспертных систем.

Экспертная система – это компьютерная программа, которая, как и эксперт-специалист обладает степенью познания в конкретной области. Число приложений с ЭС большое. В это число входит выявление болезней, определение марки автомобилей, воздушное сообщение и т.д.[3]

Существует два типа реализации ЭС : ЭС, построенную на правилах, и ЭС, построенную на логике. Рассмотрим по подробнее ЭС построенную на логике.

3.1 Экспертная Система, базирующаяся на логике.

База Знаний (БЗ) содержит в себя утверждения. В свою очередь утверждения представлены в виде предложений логики предикатов. При этом часть утверждений описывают объекты, другая часть — условия или атрибуты, которые характеризуют различные объекты. Точность классификации зависит от признаков: чем больше признаков, тем выше степень точности.[3]

Интерпретатор внутри системы выполняет свои функции на основе следующей схемы:

1) Содержание в БЗ предложения, которые управляют поиском и сопоставлением. Интерпретатор сопоставляет эти предложения с элементами данных в БД.

2) Если существует возможность вызова более одного правила, то для разрешения конфликта система использует возможности механизма внутренней унификации Пролога.

3) Система получает результаты унифицированного процесса автоматически, поэтому они направляются на нужное (логическое) устройство вывода информации. [3]

Так же, как и в ЭС, базирующейся на правилах, в ЭС базирующийся на логике данный процесс является процессом «распознавание-действие».

Основное отличие структуры ЭС, базирующейся на логике, состоит в описании объектов и атрибутов в виде фактов:

Пример реализации:[3]

/* Условия-характеристики различных пород.*/

cond(l, «короткошерстная порода»).

cond(2, «длинношерстная порода»).

cond(3, «рост меньше 22 дюймов»).

cond(4, «рост больше 30 дюймов»).

cond(5, «свисающий хвост»).

cond(6, «длинные уши»).

cond(7, «хороший характер»).

cond(8, «вес более 100 фунтов»).

/* Данные о типах породы */

topic (« короткошерстная »).

topic(«длинношерстная»).

```
/* Данные о конкретных породах */

rule(l, «собака», «короткошерстная», [1]).

rule(2, «собака», «длинношерстная», [2]).

rule(3, «короткошерстная», «английский бульдог», [3,5,7]).

rule(4, «короткошерстная», «гончая», [3,6,7]).

rule(5, «короткошерстная», «датский дог», [5,6,7,8]).

rule(6, «короткошерстная», «американский фокстерьер», [4,6,7]).

rule(7, «длинношерстная», «коккер-спаниэль», [3,5,6,7]).

rule(8, «длинношерстная», «ирландский сеттер», [4,6]).

rule(9, «длинношерстная», «колли», [4,5,7]).

rule(10, «длинношерстная», «сенбернар», [5,7,8]).



/* Начальное правило механизма вывода */

go(_, Mygoal): —

not(rule(_, Mygoal, _, _)), !,

concat(«Рекомендуемая порода : », Mygoal, Temp),

concat(Temp, «.», Result),

dlg_Note(«Экспертное заключение : », Result).

go(History, Mygoal): —
```

```
rule(Rule_number, Mygoal, Type_of_breed, Conditions),

check(Rule_number, History, Conditions),

go([Rule_number|History], Type_of_breed).


/* Сопоставление входных данных пользователя со списками атрибутов
отдельных пород собак */

check(Rule_number, History, [Breed_cond|Rest_breed_cond_list]):-

yes(Breed_cond), !,

check(Rule_number, History, Rest_breed_cond_list).

check(_, _, [Breed_cond|_]): -

no(Breed_cond), !, fail.

check(Rule_number, History, [Breed_cond|Rest_breed_cond_list]):-

cond(Breed_cond, Text),

ask_question(Breed_cond, Text),

check(Rule_number, History, Rest_breed_cond_list).

check(_, _,[]).

do_answer(Cond_number, 1): —!,

assertz (yes (Cond _ number)).

do_answer(Cond_number, 2): — !,

assertz(no(Cond_number)), fail.
```

```
/* Запрос и получение ответов yes и no от пользователя */

ask_question(Breed_cond, Text): —

concat(«Вопрос : », Text, Temp),

concat(Temp, « », Temp1),

concat(Tempi, «?», Quest),

Responsel1=dlg_Ask(«Консультация», Quest, [«Да»,«Нет»]),

Response=Response1+1,

do_answer(Breed_cond, Response).
```

Достоинством ЭС, базирующейся на логике, является возможность хранения фактов Базы Знаний в утверждениях динамической Базы Данных. В свою очередь база данных может быть на внешнем носителе (usb-носитель, диск и т. д.) [3]

3.2 Описание экспертной системы для мед. учреждения

В этой главе мы описываем экспертную систему мед. учреждения. По легенде гипотетический пациент, обладающий рядом симптомов подходит в поликлинику. Перед тем, как пациент получит лечение, нужно выяснить какой болезнью он болеет. Для этого он проходит опрос от ЭС который определяет болезнь по симптомам. Для начала ЭС запрашивает имя пациента.

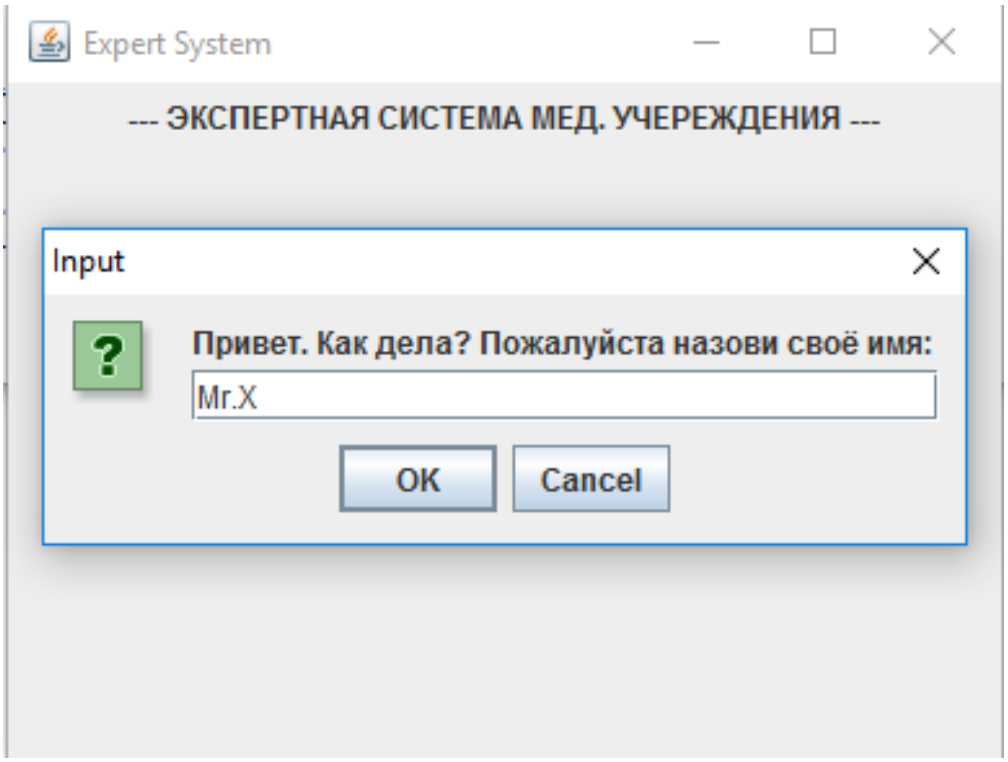

Рис.8 . Начало работы. ЭС запрашивает имя пациента.

Далее производится опрос пациента под именем «Mr.X» на наличии определённых симптомов. От Mr.X требуется подтверждения что он обладает такими симптомами, т.е. должен ответить да или нет (или y/n).

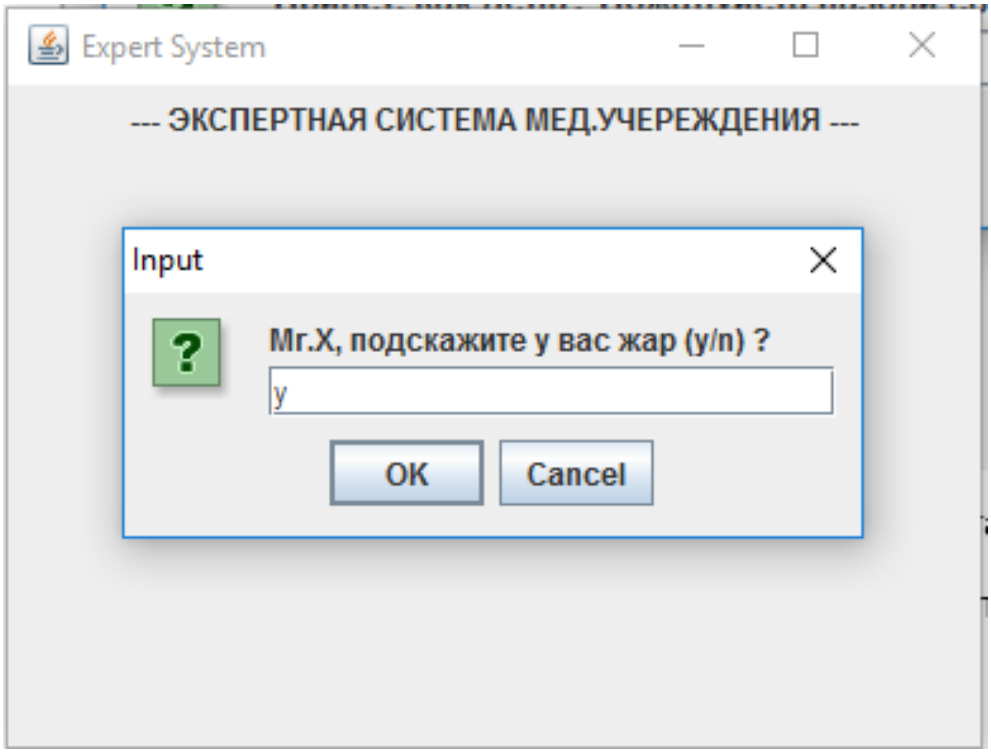

Рис. . Один из вопросов опроса пациентов. Пациент Mr.X подтверждает, что у него жар.

После того как Mr.X подтвердил, что у него есть такой симптом система ищет среди базе болезней заболевание для которого характерна этот синдром. Занимает это определённое время. Затем ЭС задаёт следующий вопрос пациенту. В случае если пациент не подтвердил данный синдром система просто переключается на другой вопрос. После прохождения опроса выводится результат.

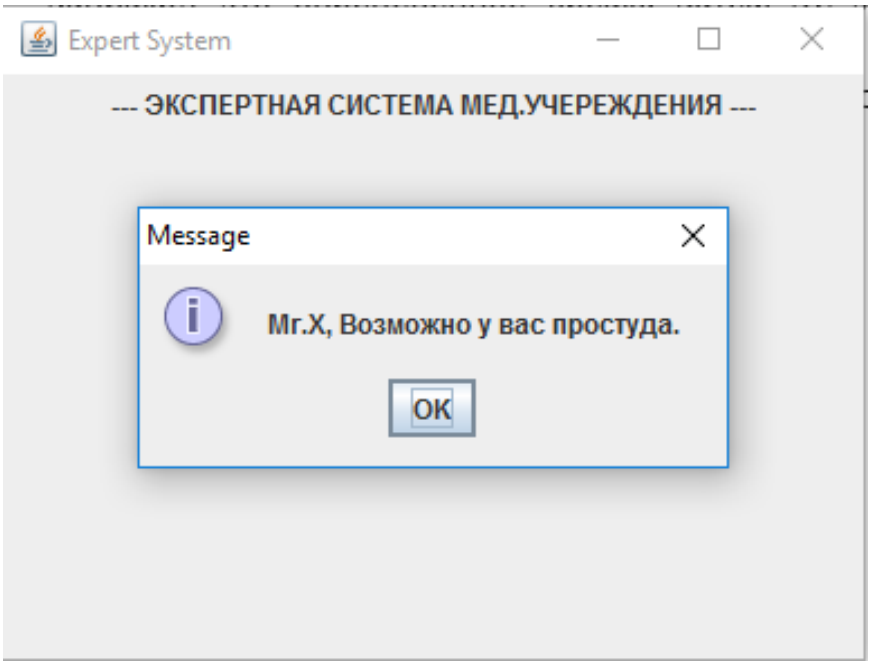

Рис.9 . Результат после проведённого опроса пациента. У Mr.X обнаружили простуду.

В случаи если система не могла найти базе болезни заболевание по симптомам пациента, то оповещает, что пациент болен неизвестным заболеванием.

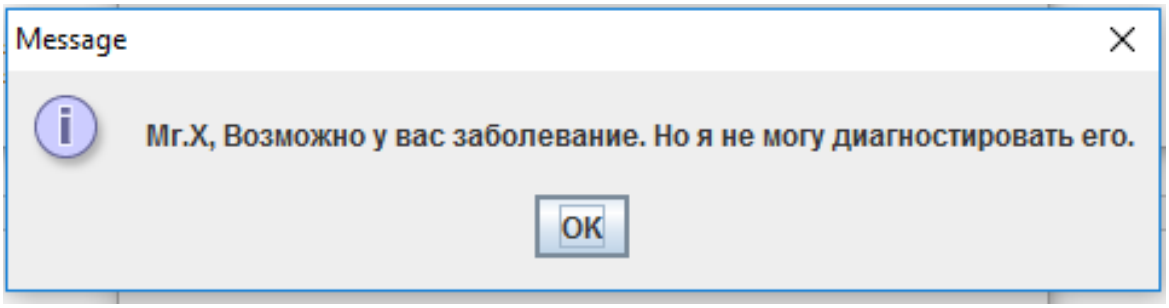

Рис.10 . Сообщение для Mr.X: вы больны неизвестной болезнью.

Данную проблему можно исправить если в базе болезней мы добавим новую болезнь, которую ей присуща определённый ряд симптомов. Данную процедуру добавлений новых заболеваний проводится человеком-экспертом по этому делу.

3.3 Листинг программы.

```
 :- use_module(library(jpl)).
start :-sleep(0.4),
                write('------------------------------------------------------------'),nl,
                sleep(0.4),

        write('**************************************************************
*********'),nl,
                sleep(0.2),
                write("##########|||          ЭКСПЕРТНАЯ          СИСТЕМА
|||##################"),nl,
                sleep(0.4),

        write('**************************************************************
*********'),nl,
                sleep(0.4),
                write('------------------------------------------------------------
'),nl,nl,nl,


   interface2.


   symptom(Patient,fever) :- verify(Patient," у вас жар (y/n) ?").

   symptom(Patient,rash) :- verify(Patient," у вас сыпь (y/n) ?").

   symptom(Patient,headache) :- verify(Patient," испытывайте ли головную
боль (y/n) ?").

   symptom(Patient,runny_nose) :- verify(Patient," течёт ли из носа (y/n)
?").

   symptom(Patient,conjunctivitis) :- verify(Patient," воспполены ли у вас
глаза (y/n) ?").

   symptom(Patient,cough) :- verify(Patient," вы кашлеете (y/n) ?").

   symptom(Patient,body_ache) :- verify(Patient," болит ли у вас тело (y/n)
?").
```

```
symptom(Patient,chills) :- verify(Patient," у вас есть озноб (y/n) ?").

symptom(Patient,sore_throat) :- verify(Patient," болит ли у вас горло (y/n) ?").

symptom(Patient,sneezing) :- verify(Patient," вы чихаете (y/n) ?").

symptom(Patient,swollen_glands) :- verify(Patient," раздуты ли у вас гланды (y/n) ?").



hypothesis(Patient, корь) :-
   symptom(Patient,fever),
   symptom(Patient,cough),
   symptom(Patient,conjunctivitis),
   symptom(Patient,runny_nose),
   symptom(Patient,rash).

hypothesis(Patient,краснуха) :-
   symptom(Patient,fever),
   symptom(Patient,headache),
   symptom(Patient,runny_nose),
   symptom(Patient,rash).

hypothesis(Patient,грипп) :-
   symptom(Patient,fever),
   symptom(Patient,headache),
   symptom(Patient,body_ache),
   symptom(Patient,conjunctivitis),
   symptom(Patient,chills),
   symptom(Patient,sore_throat),
   symptom(Patient,runny_nose),
   symptom(Patient,cough).

hypothesis(Patient,простуда) :-
   symptom(Patient,headache),
   symptom(Patient,sneezing),
   symptom(Patient,sore_throat),
```

```
    symptom(Patient,runny_nose),
    symptom(Patient,chills).

hypothesis(Patient,паротит) :-
    symptom(Patient,fever),
    symptom(Patient,swollen_glands).

hypothesis(Patient,ветрянка) :-
    symptom(Patient,fever),
    symptom(Patient,chills),
    symptom(Patient,body_ache),
    symptom(Patient,rash).

hypothesis(Patient,корь) :-
    symptom(Patient,cough),
    symptom(Patient,sneezing),
    symptom(Patient,runny_nose).

hypothesis(_,"заболевание. Но я не могу диагностировать его").

response(Reply) :-
    read(Reply),
    write(Reply),nl.

ask(Patient,Question) :-
      write(Patient),write(', подскажите'),write(Question),
      /*read(N),
      ( (N == yes ; N == y)
   ->
    assert(yes(Question)) ;
    assert(no(Question)), fail),*/

      interface(', подскажите',Patient,Question),
      write('Loading.'),nl,
      sleep(1),
      write('Loading..'),nl,
      sleep(1),
      write('Loading...'),nl,
      sleep(1),
nl.
```

```
:- dynamic yes/1,no/1.

verify(P,S) :-
  (yes(S)
   ->
   true ;
   (no(S)
    ->
    fail ;
    ask(P,S))).

undo :- retract(yes(_)),fail.
undo :- retract(no(_)),fail.
undo.
pt(Patient):-
                hypothesis(Patient,Disease),
                interface3(Patient,', Возможно у вас ',Disease,'.'),
      write(Patient),write(', возможно у вас есть
'),write(Disease),write('.'),undo,end.

end :-
                nl,nl,nl,
                sleep(0.7),
        write('**********************************************************
*********'),nl,
                sleep(0.4),
                write("################||| СПАСИБО ЗА ВНИМАНИЕ
|||#####################"),nl,
                sleep(0.4),

        write('**********************************************************
*********'),nl.
interface(X,Y,Z) :-
        atom_concat(Y,X, FAtom),
        atom_concat(FAtom,Z,FinalAtom),
        jpl_new('javax.swing.JFrame', ['Expert System'], F),
        jpl_new('javax.swing.JLabel',['--- ЭКСПЕРТНАЯ СИСТЕМА
МЕД.УЧЕРЕЖДЕНИЯ ---'],LBL),
        jpl_new('javax.swing.JPanel',[],Pan),
```

```
        jpl_call(Pan,add,[LBL],_),
        jpl_call(F,add,[Pan],_),
        jpl_call(F, setLocation, [400,300], _),
        jpl_call(F, setSize, [400,300], _),
        jpl_call(F, setVisible, [@(true)], _),
        jpl_call(F, toFront, [], _),
        jpl_call('javax.swing.JOptionPane', showInputDialog, [F,FinalAtom],
N),
        jpl_call(F, dispose, [], _),
        write(N),nl,
        ( (N == yes ; N == y)
    ->
     assert(yes(Z)) ;
     assert(no(Z)), fail).
interface2 :-
        jpl_new('javax.swing.JFrame', ['Expert System'], F),
        jpl_new('javax.swing.JLabel',['--- ЭКСПЕРТНАЯ СИСТЕМА МЕД.
УЧЕРЕЖДЕНИЯ ---'],LBL),
        jpl_new('javax.swing.JPanel',[],Pan),
        jpl_call(Pan,add,[LBL],_),
        jpl_call(F,add,[Pan],_),
        jpl_call(F, setLocation, [400,300], _),
        jpl_call(F, setSize, [400,300], _),
        jpl_call(F, setVisible, [@(true)], _),
        jpl_call(F, toFront, [], _),
        jpl_call('javax.swing.JOptionPane', showInputDialog, [F,'Привет. Как
дела? Пожалуйста назови своё имя: '], N),
        jpl_call(F, dispose, [], _),
        /*write(N),nl,*/
        (       N == @(null)
                ->      write('Вы вышли.'),interface3('Вы вышли. ','Спасибо
','за ','внимание.'),end,fail
                ;       write("Привет. Как дела пожалуйста назови своё имя:
"),write(N),nl,pt(N)
        ).
interface3(P,W1,D,W2) :-
        atom_concat(P,W1, A),
        atom_concat(A,D,B),
        atom_concat(B,W2,W3),
        jpl_new('javax.swing.JFrame', ['Expert System'], F),
```

```
        jpl_new('javax.swing.JLabel',['---  ЭКСПЕРТНАЯ  СИСТЕМА
МЕД.УЧЕРЕЖДЕНИЯ ---'],LBL),
        jpl_new('javax.swing.JPanel',[],Pan),
        jpl_call(Pan,add,[LBL],_),
        jpl_call(F,add,[Pan],_),
        jpl_call(F, setLocation, [400,300], _),
        jpl_call(F, setSize, [400,300], _),
        jpl_call(F, setVisible, [@(true)], _),
        jpl_call(F, toFront, [], _),
        jpl_call('javax.swing.JOptionPane', showMessageDialog, [F,W3], N),
        jpl_call(F, dispose, [], _),
        /*write(N),nl,*/
        (       N == @(void)
                ->      write('')
                ;       write("")
).
help :- write("Для запуска программы наберите слово 'start.' и нажмите
кнопку Enter. ").
```

Заключение:

В данном проекте мы рассмотрели работу экспертных систем (ЭС). Изучили его структуру и основы ЭС. Также подробнее узнали об их преимуществе. Кроме того, мы изучили его реализацию на языке SWI-prolog и просмотрели пример работы ЭС медицинского учреждения. В заключении можно добавить, что мир всё чаще стремится облегчить и усовершенствовать жизнь. И поэтому сегодня как никогда актуальны такие системы как ЭС. Точность результатов и бесперебойность работы – вот главные достоинства системы.

Литература:

1) Хабаров С.П. Интеллектуальные информационные системы. PROLOG-язык разработки интеллектуальных и экспертных систем: учебное пособие / С.П.Хабаров.- СПб. СПбГЛТУ, 2013.- 138 с.
2) Хабаров С.П. Введение в экспертные системы. Основные понятия и определения: лекции / С.П.Хабаров.- СПб. СПбГЛТУ
3) Разработка Экспертной Системы в Prolog/ http://itmu.vsuet.ru/Posobija/Predstavlenie_znan/htm/5_pr.htm
4) Wielemaker J. SWI-Prolog / VU University Amsterdam, 2010 – 30 с.