**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Новгородский государственный университет имени Ярослава Мудрого»**

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

Кафедра ИТиС

Отчёт по

лабораторной работе

«ЭКСПЕРТНЫЕ СИСТЕМЫ НА ПРОЛОГЕ»

дисциплина:

«Системы искусственного интеллекта»

Выполнил: ст. гр. 8091

\_\_\_\_\_\_\_ \_\_\_\_Шаклеин В.В.

Проверил: Преподаватель

\_\_\_\_\_\_\_\_\_\_\_Михайлов Д.В.

Великий Новгород

2021

# Формулировка цели и задач

Целью работы является изучение принципов построения и организации экспертных систем, базирующихся на логике и правилах.

# Данные, использованные в ЭС

Предметная область для ЭС – компьютерные вирусы

Вирусы я разбил на группы в зависимости от их основанного предназначения: вредительство и вымогательство.

Таблица . Описание предметной области

|  |  |  |
| --- | --- | --- |
| **Группа** | **Название вируса** | **Свойства** |
| **вредительство** | **Morris Worm** | отказ в обслуживании  самораспространение  повреждение данных |
| **ILOVEYOU** | отказ в обслуживании  самораспространение  заражение без участия пользователя |
| **SQL Slammer** | отказ в обслуживании  самораспространение  избирательность |
| **Stuxnet** | отказ в обслуживании  самораспространение  шифрование данных |
| **CryptoLocker** | отказ в обслуживании  самораспространение  повышение прав |
| **Conficker** | отказ в обслуживании  самораспространение  самозащита |
| **вымогательство** | **Sasser** | самораспространение  повреждение данных  избирательность |
| **Melissa** | самораспространение  повреждение данных  шифрование данных |
| **Zeus** | самораспространение  повреждение данных  повышение прав |
| **Storm Trojan** | самораспространение  повреждение данных  самозащита |
| **Michaelangelo** | самораспространение  повреждение данных  заражение без участия пользователя |
| **Code Red** | заражение без участия пользователя  избирательность  шифрование данных |

# Описание характеристик разработанных ЭС

В зависимости от способов классификации и размещения информации Базы Знаний различают: продукционную, сетевую и фреймовую модели представления знаний.

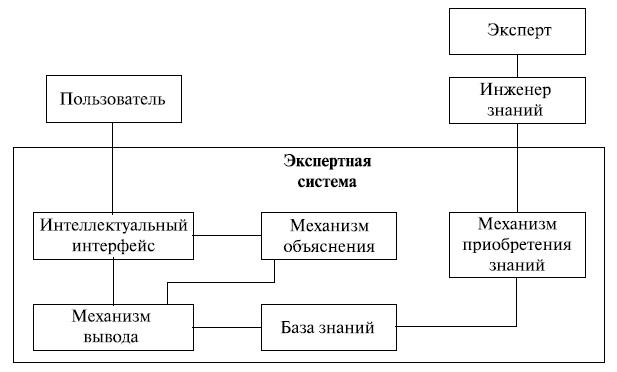
Сетевая модель основана на представление знаний в виде сети, вершины которой соответствуют понятиям, а дуги — отношениям между ними.

В основе фреймовой модели лежит логическая группировка атрибутов объекта, при этом для хранения и обработки логические группы описываются во фреймах.

ЭС, базирующаяся на правилах, позволяет проектировщику строить правила, которые естественным образом объединяют в группы связанные фрагменты знаний.

При реализации ЭС, базирующейся на логике, БЗ представляется совокупностью утверждений в виде фактов. Вывод экспертного заключения при этом строится на основе стандартных средств работы со списками.

# Диаграммы потоков данных

****

# Тестовые данные

**Для Экспертной Системы на правилах:**

Входные данные: Да, Нет, Да, Да, Да, Да   
Результат: Вирус: Sasser.

Входные данные: Нет, Да, Нет, Да, Да, Да  
Результат: Вирус: Code Red.

Входные данные: Да, Нет, Нет

Результат: Информация об интересующем Вас вирусе отсутствует в БЗ.

**Для Экспертной Системы на логике:**

Подается на ввод: вредительство  
Входные данные: Да, Да, Да

Результат: Вероятнее всего Ваш вирус: Morris Worm.

Подается на ввод: вымогательство

Входные данные: Да, Да, Да

Результат: Вероятнее всего Ваш вирус: Sasser.

Подается на ввод: вредительство

Входные данные: Нет

Результат: Информация об интересующем Вас вирусе отсутствует в БЗ.

# Описание механизмов вывода

Написание на Visual Prolog-е ЭС, базирующейся на правилах, начинается с декларации БД. БД хранит ответы пользователя на вопросы СПИ. Эти данные являются утвердительными или отрицательными ответами. Далее строятся продукционные правила, описывающие фрагменты фактического знания.

Запросы пользователя трансформируются в форму, сопоставимую с формой правил БЗ. Механизм вывода инициализирует процесс сопоставления, начиная с «верхнего» правила. Обращение к правилу называется «вызовом». Вызов соответствующих правил в процессе сопоставления продолжается до тех пор, пока не произошло сопоставление или не исчерпана вся БЗ, а значение не найдено. Если МВ обнаруживает, что можно вызвать более одного правила, то запускается процесс разрешения конфликта. При разрешении конфликта приоритет отдается обычно тем правилам, которые более конкретны, либо правилам, которые учитывают больше текущих данных.

Экспертная Система, базирующаяся на логике, содержит в БЗ предложения, которые управляют поиском и сопоставлением. Интерпретатор сопоставляет эти предложения с элементами данных в БД.

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

Система получает результаты унификационного процесса автоматически, поэтому они направляются на нужное (логическое) устройство вывода информации.

Так же, как и в ЭС, базирующейся на правилах, данный циклический процесс является процессом распознавание-действие.

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

# Вывод

Экспертные Системы базирующиеся на правилах проще в разработке и первоначальном наполнение знаниями, Экспертные Системы базирующиеся на логике в свою очередь требуют больших усилий для изначальной разработки и наполнения, однако позволяют более гибко осуществлять добавление новых записей.

# Приложение

## ЭС базирующаяся на правилах

clauses

/\* Консультация \*/

     do\_consulting:-

           virus\_is(X),!,

           concat("Вирус : ",X,Temp),

           concat(Temp,".",Result),

           dlg\_Note("Экспертное заключение : ",Result),

           clear.

     do\_consulting:-

           dlg\_Error("Информация об интересующем Вас вирусе отсутствует в БЗ."),

           clear.

     ask(X,Y):-

           concat("Вопрос : ",X,Temp),

           concat(Temp," ",Temp1),

           concat(Temp1,Y,Temp2),

           concat(Temp2,"?",Quest),

           Reply1=dlg\_Ask("Консультация",Quest,["Да","Нет"]),

           Reply=Reply1+1,

           remember(X,Y,Reply).

/\* Механизм вывода экспертного заключения \*/

     positive(X,Y):-

           xpositive(X,Y),!.

     positive(X,Y):-

           not(negative(X,Y)),!,ask(X,Y).

     negative(X,Y):-

           xnegative(X,Y),!.

     remember(X,Y,1):-!,

           assertz(xpositive(X,Y)).

     remember(X,Y,2):-!,

           assertz(xnegative(X,Y)),fail.

/\* Продукционные правила \*/

/\*

          positive("имеет", "отказ в обслуживании"),

          positive("имеет", "самораспространение"),

          positive("имеет", "повреждение данных"),

          positive("имеет", "заражение без участия пользователя"),

          positive("имеет", "избирательность"),

          positive("имеет", "шифрование данных"),

          positive("имеет", "повышение прав"),

          positive("имеет", "самозащита"),

          \*/

     virus\_is("Morris Worm"):-

           it\_is("вредительство"),

           positive("имеет", "отказ в обслуживании"),

           positive("имеет", "самораспространение"),

           positive("имеет", "повреждение данных"),!.

     virus\_is("ILOVEYOU"):-

           it\_is("вредительство"),

           positive("имеет", "отказ в обслуживании"),

           positive("имеет", "самораспространение"),

           positive("имеет", "заражение без участия пользователя"),!.

     virus\_is("SQL Slammer"):-

           it\_is("вредительство"),

           positive("имеет", "отказ в обслуживании"),

           positive("имеет", "самораспространение"),

           positive("имеет", "избирательность"),!.

     virus\_is("Stuxnet"):-

           it\_is("вредительство"),

           positive("имеет", "отказ в обслуживании"),

           positive("имеет", "самораспространение"),

           positive("имеет", "шифрование данных"),!.

     virus\_is("CryptoLocker"):-

           it\_is("вредительство"),

           positive("имеет", "отказ в обслуживании"),

           positive("имеет", "самораспространение"),

           positive("имеет", "повышение прав"),!.

     virus\_is("Conficker"):-

           it\_is("вредительство"),

           positive("имеет", "отказ в обслуживании"),

           positive("имеет", "самораспространение"),

           positive("имеет", "самозащита"),!.

     virus\_is("Sasser"):-

           it\_is("вымогательство"),

           positive("имеет", "самораспространение"),

           positive("имеет", "повреждение данных"),

           positive("имеет", "избирательность"),!.

     virus\_is("Melissa"):-

           it\_is("вымогательство"),

           positive("имеет", "самораспространение"),

           positive("имеет", "повреждение данных"),

           positive("имеет", "шифрование данных"),!.

     virus\_is("Zeus"):-

           it\_is("вымогательство"),

           positive("имеет", "самораспространение"),

           positive("имеет", "повреждение данных"),

           positive("имеет", "повышение прав"),!.

     virus\_is("Storm Trojan"):-

           it\_is("вымогательство"),

           positive("имеет", "самораспространение"),

           positive("имеет", "повреждение данных"),

           positive("имеет", "самозащита"),!.

     virus\_is("Michaelangelo"):-

           it\_is("вымогательство"),

           positive("имеет", "самораспространение"),

           positive("имеет", "повреждение данных"),

           positive("имеет", "заражение без участия пользователя"),!.

     virus\_is("Code Red"):-

           it\_is("вымогательство"),

           positive("имеет", "заражение без участия пользователя"),

           positive("имеет", "избирательность"),

           positive("имеет", "шифрование данных"),!.

     it\_is("вредительство"):-

           positive("цель вируса","вредительство"),!.

     it\_is("вымогательство"):-

           positive("цель вируса","вымогательство"),!.

/\* Уничтожение в базе данных всех ответов yes (да) и no (нет) \*/

     clear:-retract(xpositive(\_,\_)),retract(xnegative(\_,\_)),fail,!.

     clear.

## ЭС базирующаяся на логике

predicates

  task\_win\_eh : EHANDLER

     rev(char\_list,char\_list,char\_list).

     reverse(char\_list,char\_list).

     nondeterm rule(rule\_number,category,category,conditions)

     nondeterm cond(cond\_number,condition)

     nondeterm topic(condition)

     assert\_database

     do\_consulting

     nondeterm info

     erase

     clear

     do\_answer(cond\_number,integer)

     keyword(category)

     first\_keyword\_in\_sentence(word\_list,category)

     member(char,char\_list).

     nondeterm symbol\_counter(string,integer).

     nondeterm del\_front\_space(string,string).

     nondeterm fronttoken\_cyr(string,string,string).

     nondeterm convers(string,word\_list)

     nondeterm upper\_lower\_cyr(string,string).

     nondeterm upper\_lower\_cyr\_convers(char\_list,char\_list).

     nondeterm str\_char\_list(string,char\_list).

     pack(char\_list,string).

     goes(string,category)

     nondeterm go(history,category)

     nondeterm check(rule\_number,history,conditions)

     ask\_question(cond\_number,condition)

constants

%BEGIN Task Window, CreateParms, 21:26:22-22.12.2007, Code automatically updated!

  task\_win\_Flags = [wsf\_SizeBorder,wsf\_TitleBar,wsf\_Close,wsf\_Maximize,wsf\_Minimize,wsf\_ClipSiblings]

  task\_win\_Menu  = res\_menu(idr\_task\_menu)

  task\_win\_Title = "Экспертная система, базирующаяся на логике"

  task\_win\_Help  = idh\_contents

%END Task Window, CreateParms

clauses

/\* База знаний. \*/

/\* Размещение в резидентной БД информации из утверждений БЗ ЭС \*/

     assert\_database:-

           rule(Rule\_number,Category,Type\_of\_virus,Conditions),

           assertz(d\_rule(Rule\_number,Category,Type\_of\_virus,Conditions)),fail.

     assert\_database:-

           cond(Cond\_number,Condition),

           assertz(d\_cond(Cond\_number,Condition)),fail.

     assert\_database:-

           topic(Condition),

           assertz(d\_topic(Condition)),fail.

     assert\_database:-!.

/\* Условия-характеристики различных вирус.\*/

     cond(1,"цель вредительство").

     cond(2,"цель вымогательство").

     cond(3,"отказ в обслуживании").

     cond(4,"самораспространение").

     cond(5,"повреждение данных").

     cond(6,"заражение без участия пользователя").

     cond(7,"избирательность").

     cond(8,"шифрование данных").

     cond(9,"повышение прав").

     cond(10,"самозащита").

/\* Данные о типах вирусы \*/

     topic("вымогательство").

     topic("вредительство").

/\* Данные о конкретных вирусах \*/

     rule(1,"вирус","вымогательство",[1]).

     rule(2,"вирус","вредительство",[2]).

     rule(3,"вредительство","Morris Worm",[3,4,5]).

     rule(4,"вредительство","ILOVEYOU",[3,4,6]).

     rule(5,"вредительство","SQL Slammer",[3,4,7]).

     rule(6,"вредительство","Stuxnet",[3,4,8]).

     rule(7,"вредительство","CryptoLocker",[3,4,9]).

     rule(8,"вредительство","Conficker",[3,4,10]).

     rule(9,"вымогательство","Sasser",[4,5,6]).

     rule(10,"вымогательство","Melissa",[4,5,7]).

     rule(11,"вымогательство","Zeus",[4,5,8]).

     rule(12,"вымогательство","Storm Trojan",[4,5,9]).

     rule(13,"вымогательство","Michaelangelo",[4,5,10]).

     rule(14,"вымогательство","Code Red",[5,6,7]).

     do\_consulting:-

           goes(\_,First\_keyword),

           go([],First\_keyword),!.

     do\_consulting:-

           not(dummy),

           dlg\_Error("Информация об интересующем Вас вирусе отсутствует в БЗ."),

           clear.

/\* Выдача подсказки \*/

     info:-

           findall(Virus\_type,topic(Virus\_type),Virus\_type\_list),

           term\_str(slist,Virus\_type\_list,Virus\_type\_list\_str\_repr),

           str\_char\_list(Virus\_type\_list\_str\_repr,[\_|Virus\_type\_list\_char\_repr]),

           reverse(Virus\_type\_list\_char\_repr,[\_|Virus\_type\_list\_char\_repr\_rev]),

           reverse(Virus\_type\_list\_char\_repr\_rev,Virus\_type\_list\_char\_repr\_rev1),

           pack(Virus\_type\_list\_char\_repr\_rev1,Virus\_type\_list\_str\_repr1),

           dlg\_Note("База знаний содержит информацию о типах вирусов : ",Virus\_type\_list\_str\_repr1),

           assertz(dummy).

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

     ask\_question(Virus\_cond,Text):-

           concat("Вопрос : ",Text,Temp),

           concat(Temp," ",Temp1),

           concat(Temp1,"?",Quest),

           Response1=dlg\_Ask("Консультация",Quest,["Да","Нет"]),

           Response=Response1+1,

           do\_answer(Virus\_cond,Response).

/\* Предикаты ЕЯ-интерфейса \*/

/\* Реверсирование списка \*/

     rev([],Init,Init).

     rev([H|T],Init,Res):-

        rev(T,[H|Init],Res).

     reverse(Arg,Res):-

        rev(Arg,[],Res).

/\* Принадлежность элемента списку \*/

     member(Head,[Head|\_]):-!.

     member(Elem,[\_|T]):-

             member(Elem,T).

/\* Подсчет символов в строке до конца строки, либо ближайшего пробела,

   символа возврата каретки, перевода строки, !,",#,$ \*/

     symbol\_counter(Str,0):-

             frontchar(Str,'\32',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\10',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\13',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\33',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\34',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\35',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\36',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\40',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\41',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\44',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\45',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\46',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\59',\_),!.

     symbol\_counter("",0).

     symbol\_counter(Str,Number):-

             frontchar(Str,\_Char,Rest\_of\_string),

             symbol\_counter(Rest\_of\_string,Number1),

             Number=Number1+1.

/\* Выделение подстроки до первого разделителя \*/

     fronttoken\_cyr(Str,Token,Rest\_of\_string):-

             symbol\_counter(Str,Number),

             frontstr(Number,Str,Token,Rest\_of\_string).

/\* Удаление разделителя в начале строки \*/

     del\_front\_space("","").

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\32',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\10',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\13',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\33',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\34',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\35',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\36',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\44',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\40',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\41',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\45',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\46',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\59',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Arg):-frontchar(Arg,Char,\_),

                               not(member(Char,['\32','\10','\13','\33',

                                                '\34','\35','\36','\40','\41',

                                                '\44','\45','\46','\59'])).

/\* Модифицированное правило преобразования строки в список

   слов \*/

     convers("",[]):-!.

     convers(Str,[Head1|Tail]):-

             fronttoken\_cyr(Str,Head,Str2),

             upper\_lower\_cyr(Head,Head1),

             del\_front\_space(Str2,Str1),

             convers(Str1,Tail).

/\* Предикат upper\_lower для кирилицы Windows \*/

     upper\_lower\_cyr(InString,OutString):-

             str\_char\_list(InString,Char\_List\_for\_InString),

             upper\_lower\_cyr\_convers(Char\_List\_for\_InString,Char\_List\_for\_OutString),

             pack(Char\_List\_for\_OutString,OutString).

     upper\_lower\_cyr\_convers([],[]).

     upper\_lower\_cyr\_convers([Char|Char\_List],[Char1|Char\_List1]):-

             char\_int(Char,ASCII\_code),

             ASCII\_code>=192,ASCII\_code<=223,!,

             ASCII\_code\_new=ASCII\_code+32,

             char\_int(Char1,ASCII\_code\_new),

             upper\_lower\_cyr\_convers(Char\_List,Char\_List1).

     upper\_lower\_cyr\_convers([Char|Char\_List],[Char1|Char\_List1]):-

             char\_int(Char,ASCII\_code),

             ASCII\_code=168,!,

             ASCII\_code\_new=ASCII\_code+16,

             char\_int(Char1,ASCII\_code\_new),

             upper\_lower\_cyr\_convers(Char\_List,Char\_List1).

     upper\_lower\_cyr\_convers([Char|Char\_List],[Char1|Char\_List1]):-

             char\_int(Char,ASCII\_code),

             ASCII\_code>=65,ASCII\_code<=90,!,

             ASCII\_code\_new=ASCII\_code+32,

             char\_int(Char1,ASCII\_code\_new),

             upper\_lower\_cyr\_convers(Char\_List,Char\_List1).

     upper\_lower\_cyr\_convers([Char|Char\_List],[Char|Char\_List1]):-

             upper\_lower\_cyr\_convers(Char\_List,Char\_List1).

/\* Преобразование строки в список символов \*/

     str\_char\_list("",[]).

     str\_char\_list(Word,[Char|Char\_List]):-

             frontchar(Word,Char,WordRest),

             str\_char\_list(WordRest,Char\_List).

/\* Превращение списка символов в строку \*/

     pack([],"").

     pack([H|T],Res):-

             str\_char(Str\_H,H),

             pack(T,Res1),

             concat(Str\_H,Res1,Res).

/\* Проверка правильности ключевого слова \*/

     keyword(Keyword):-

             rule(\_,Keyword,\_,\_),!.

     keyword(Keyword):-

             rule(\_,\_,Keyword,\_),!.

/\* Поиск первого ключевого слова в высказывании пользователя \*/

     first\_keyword\_in\_sentence([Head|\_],Head):-

             keyword(Head),!.

     first\_keyword\_in\_sentence([\_|Tail],First\_keyword):-

             first\_keyword\_in\_sentence(Tail,First\_keyword).

     first\_keyword\_in\_sentence([],\_):-!,info,fail.

/\* Ввод запроса на естественном (русском) языке \*/

     goes(Mygoal,First\_keyword):-

           Mygoal=dlg\_GetStr("Консультация","Введите Ваш запрос : ","Текст Вашего запроса"),

           convers(Mygoal,Word\_list),

           first\_keyword\_in\_sentence(Word\_list,First\_keyword),!.

/\* Механизм вывода \*/

/\* Начальное правило механизма вывода \*/

     go(\_,Mygoal):-

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

           concat("Вероятнее всего Ваш вирус : ",Mygoal,Temp),

           concat(Temp,".",Result),

           dlg\_Note("Экспертное заключение : ",Result).

     go(History,Mygoal):-

           rule(Rule\_number,Mygoal,Type\_of\_virus,Conditions),

           check(Rule\_number,History,Conditions),

           go([Rule\_number|History],Type\_of\_virus).

/\* Сопоставление входных данных пользователя со списками атрибутов

   отдельных типов вирусов \*/

     check(Rule\_number,History,[Virus\_cond|Rest\_virus\_cond\_list]):-

           yes(Virus\_cond),!,

           check(Rule\_number,History,Rest\_virus\_cond\_list).

     check(\_,\_,[Virus\_cond|\_]):-

           no(Virus\_cond),!,fail.

     check(Rule\_number,History,[Virus\_cond|Rest\_virus\_cond\_list]):-

           cond(Virus\_cond,Text),

           ask\_question(Virus\_cond,Text),

           check(Rule\_number,History,Rest\_virus\_cond\_list).

     check(\_,\_,[]).

     do\_answer(Cond\_number,1):-!,

           assertz(yes(Cond\_number)).

     do\_answer(Cond\_number,2):-!,

           assertz(no(Cond\_number)),fail.

/\* Исключение данных из базы знаний

   после завершения цикла "Распознавание-действие" \*/

     erase:-retract(\_),fail.

     erase.

/\* Уничтожение в базе данных всех ответов yes (да) и no (нет) \*/

     clear:-retract(yes(\_)),retract(no(\_)),fail,!.

     clear.