Разработка экспертной системы, реализующей обратную цепочку рассуждений, в среде Visual Prolog 5.1

# Программная реализация

domains

spisok = string\*

predicates

main

input\_data(string,string) % Ввод данных

get\_sreda(string) % Возвращает значение переменной "среда"

get\_conc(string) % Возвращает значение переменной "концентрация"

get\_temp(string) % Возвращает значение переменной "температура"

poyasnenie(spisok)

ask(integer) % Объяснение рассуждений

why(spisok)

append(spisok,spisok,spisok) % Соединение двух списков

is(string,string,string) % Факты, используемые базой знаний

mater\_v(string,spisok) % Правило базы знаний

sreda(string,spisok)

futerovka(string,spisok)

mater\_f(string,spisok)

temp(string,string,spisok)

type\_o(string,spisok)

database - tmp

% Данные, введенные пользователем

sr(string) % Среда

conc(string) % Концентрация

t(string) % Температура

clauses

main:-

write("Выберите цель:"),nl,nl,

write("1. Материал ванны"),nl,

write("2. Материал футеровки"),nl,

write("3. Тип обогрева"),nl,

write("4. Выход"),nl,nl,

write("Ваш выбор > "),

readint(Num),nl,

ask(Num).

input\_data(ParamName,Dt):-

write("Введите значение параметра '"),

write(ParamName),

write("' > "),

readln(Dt).

get\_sreda(S):-

sr(S),!.

get\_sreda(S):-

input\_data("среда",S),

assert(sr(S)).

get\_conc(C):-

conc(C),!.

get\_conc(C):-

input\_data("концентрация",C),

assert(conc(C)).

get\_temp(T):-

t(T),!.

get\_temp(T):-

input\_data("температура",T),

assert(t(T)).

poyasnenie(List):-

write("\nРешение получено на основании следующих рассуждений: "),nl,

why(List),nl,nl,

ask(4).

ask(1):-

mater\_v(Y,List),

write("\nМатериал ванны: ",Y),nl,

poyasnenie(List).

ask(2):-

mater\_f(Y,List),

write("\nМатериал футеровки: ",Y),nl,

poyasnenie(List).

ask(3):-

type\_o(Y,List),

write("\nТип обогревателя: ",Y),nl,

poyasnenie(List).

ask(4):-

write("Теперь можно закрыть это окно!").

ask(\_):-

write("\nНеверный ввод.").

why([]).

why([X|Y]):-

write(X),

why(Y).

append([],L,L).

append([N|L1],L2,[N|L3]):-

append(L1,L2,L3).

is("кислота","агрессивная жидкость",F):-

F="\nкислота = агрессивная жидкость".

is("азотная кислота","кислота",F):-

F="\nазотная кислота = кислота".

is("сернистая кислота","кислота",F):-

F="\nсернистая кислота = кислота".

mater\_v("хромоникелевая сталь",List):-

sreda("концентрированная кислота",List1),

append(["\nматериал ванны = хромоникелевая сталь, т.к.\n"],List1,List2),

append(List2,["."],List).

mater\_v("углеродистая сталь",List):-

sreda("разбавленная кислота",List1),

append(["\nматериал ванны = углеродистая сталь, т.к.\n"],List1,List2),

append(List2,["."],List).

sreda("концентрированная кислота",List):-

sreda("кислота",List1),

get\_conc(C),

C=">70%",

append(["\nсреда = концентрированная кислота, т.к.\n"],List1,List2),

append(List2,["\n\nИ концентрация >70%"],List).

sreda("разбавленная кислота",List):-

sreda("кислота",List1),

get\_conc(C),

C="<70%",

append(["\nсреда = разбавленная кислота, т.к.\n"],List1,List2),

append(List2,["\n\nИ концентрация <70%"],List).

sreda("кислота",List):-

get\_sreda(S1),

is(S1,"кислота",S2),

List=["\nсреда = кислота, т.к.\n",S2,"\nсреда = ",S1,"."].

sreda("агрессивная жидкость",List):-

is(K,"агрессивная жидкость",S1),

get\_sreda(S2),

is(S2,K,S3),

List=["\nсреда = агрессивная жидкость, т.к.\n",S1,S3,"\nсреда = ",S2,"."].

futerovka("есть",List):-

sreda("агрессивная жидкость",List1),

append(["\nфутеровка = есть, т.к.\n"],List1,List).

mater\_f("свинец",List):-

futerovka("есть",List1),

get\_sreda(S),

S="сернистая кислота",

append(["\nматериал футеровки = свинец, т.к.\n"],List1,List2),

append(List2,["\n\nИ среда = сернистая кислота."],List).

mater\_f("винипласт",List):-

futerovka("есть",List1),

get\_sreda(S),

S="азотная кислота",

append(["\nматериал футеровки = винипласт, т.к.\n"],List1,List2),

append(List2,["\n\nИ среда = азотная кислота."],List).

temp("меньше 100 градусов","пароводяная рубашка",List):-

List=["\nтемпература = меньше 100 градусов."].

temp("больше 100 градусов","электронагреватель",List):-

List=["\nтемпература = больше 100 градусов."].

type\_o(Tp,List):-

get\_temp(T),

temp(T,Tp,List1),

append(["\nтип обогрева = ",Tp,", т.к.\n"],List1,List).

goal

main.