Московский Государственный Технический Университет имени Н. Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

Е.К. Пугачев

УТВЕРЖДАЮ
Зав. кафедрой ИУ6

д.т.н., проф. _____ Сюзев В.В.
"____" _____2013 г.

Язык программирования Prolog применительно к системам искусственного интеллекта.

Методические указания
по выполнению лабораторных работ
по дисциплине "Системы искусственного интеллекта".

Часть 1

Москва 2013

Введение.

Целью проведения лабораторных работ является приобретение навыков разработки и реализации основных элементов систем искусственного интеллекта. Первая часть методических указаний посвящена инструментальному средству создания систем обработки знаний языку программирования высокого уровня Prolog. Во второй части методических указаний рассматриваются основные методы, применяемые при создании экспертных систем.

Конкретно предлагается освоить Turbo-Prolog 2.0. Обусловлено это следующими факторами. С одной стороны данная версия широко используется, требует мало ресурсов и включает в себя богатый набор команд для выполнения различных операций. С другой стороны, синтаксис, большое количество встроенных предикатов и механизмов составляют основу других версий языка Prolog, например, Visual Prolog 5.2.

Предыстория языка Prolog.

Название "*Пролог*" произошло от словосочетания "*Программирование при помощи логики*" (PROgramming in Logic). Пролог был разработан и впервые реализован в 1973 г. *Алэном Колмероэ* и другими членами "группы искусственного интеллекта" Марсельского университета (Франция). Главной задачей группы было создание системы *для обработки естественного языка*.

Turbo-Prolog 2.0. реализован компанией *Borland International*. Он является *компиляторно - ориентированным* языком высокого уровня. Turbo-Prolog 2.0. особенно хорош для создания *экспертных систем, динамических баз данных, программ с применением естественно-языковых конструкций*.

Наряду с Turbo-Prolog 2.0. созданы еще несколько реализаций языка Prolog, например, *Arity Prolog, Wisdom Prolog, Micro Prolog*. Шотландский вариант C&M Prolog получил название в честь авторов *Уильяма Клоксина и Кристоффера Меллиша* классической работы "Программирование на Прологе" , которая считается неофициальным стандартом.

Одной из причин предпочтительности выбора Турбо-Пролога является то, что написанные на нем программы компилируются, в отличие от других версий Пролога, где программа интерпретируется. Преимущество откомпилированной программы является в том, что данная программа в исходном тексте не нуждается и работает на много быстрее.

Самым существенным отличием Турбо-Пролога от других версий языка является наличие в Турбо-Прологе *строгой типизации элементов данных*. Сделанные отступления позволили значительно увеличить скорость трансляции и счета программ.

Такие языки как *Паскаль, Бейсик и СИ* относятся к разряду императивных или процедурных. Программа, написанная на процедурном языке, состоит из последовательности команд, определяющих шаги, необходимые для достижения назначения программы.

Пролог является *декларативным языком*. Программа на декларативном языке представляет собой набор логических взаимосвязанных описаний, определяющих цель, ради которой она написана. Обозначения, используемые в Прологе для выражений логических взаимосвязей, унаследованы из логики предикатов.

Пролог освобождает программиста от составления программы в виде последовательности действий. В нем отсутствуют такие явные управляющие структуры, как DO WHILE, IF THEN и т.п. Пролог базируется на естественных для человека логических принципах.

Краткие теоретические сведения.

Основные понятия языка Турбо-Пролог (в дальнейшем просто Пролог).

Факт - это некоторое утверждение, определяющее отношение между объектами или описывающее свойства объекта. Общая форма записи факта имеет следующий вид:

<имя отношения>(*имя_объекта_1, имя_объекта_2, ... , имя_объекта_N*).

Необходимо соблюдать следующие правила :

- имена всех отношений и объектов должны начинаться со строчной буквы;
- сначала записывается имя отношения, затем через запятую записываются имена объектов , а весь список имен объектов заключается в круглые скобки;
- каждый факт должен заканчиваться точкой;
- имена объектов в скобках могут перечисляться произвольно, но по одному произвольному порядку.

Например, факт с двумя объектами может быть описан так:

likes(tom,computer).

На естественном языке вышеприведенный факт означает: "Тому нравится компьютер".

Атом - имя, число без знака или символ.

Аргумент - имя объекта в круглых скобках .

Объект - название отдельного элемента в конструкции Пролога.

Домен - диапазон и тип значений, определенные для базисного типа данных.

Предикат - утверждение о наличии связи между объектами посредством задания имени отношения перед круглыми скобками и доменов его аргументов.

Функтор - имя составного объекта (в Прологе функторы объявляются в разделе программы predicates).

База данных - в Прологе представляет собой совокупность фактов (утверждений).

Унификация - процесс, выполняющий попытки сопоставить цель и утверждение. Он обычно включает поиск, сопоставление и означивание.

Правило - утверждение о связи некоторого факта с другими фактами.

Общая форма записи правила имеет вид:

<заголовок правила>:- <тело правила>.

Заголовок представляет собой предикат. Тело состоит из термов, которые могут быть связаны между собой "," или ";". (","- означает И,";"- означает ИЛИ).

Между телом и заголовком стоит символ ":-", который означает ЕСЛИ.
Например, правило, состоящее из двух термов может описано следующим образом:

likes(tom,kathy) :- likes(kathy,computer), likes(kathy,apples).

На естественном языке это означает : "Тому нравится Кэти, если Кэти нравится компьютер и яблоки."

Структура программы и стандартные типы доменов языка Пролог.

Программа на Прологе может состоять из следующих разделов:

1. CONSTANS - раздел описания констант. Имена констант должны быть описаны строчными буквами. В качестве констант могут быть целые или вещественные числа, символы, строки. Например, col=17, x=3.62, c='W', st="Выход" .

2. DOMAINS - данный раздел содержит определения доменов, которые описывают различные классы объектов используемых в программе (определение типов данных).

Например, имеется факт

likes(mary,apples).

Здесь mary и appless являются объектами предиката likes . Пролог требует указания типов объектов для каждого предиката программы.

Для указания типа объекта Пролог имеет шесть следующих типов данных (типов доменов):

* symbol (символические имена) - это последовательность букв, цифр и знаков подчеркивания, которая начинается со строчной буквы или заключена в кавычки, например: flower, pay_check,"Prolog" и т.п..

* string (строки) - любая последовательность символов, которая заключена в кавычки (не более 250). Например: "today","123","ПРИВЕТ".

В Турбо-Прологе 2.0 ограничение 250 накладывается, если какой-либо переменной присваивается конкретное значение непосредственно в тексте программы. Например, пусть переменным S1 и S2 конкретизированы значениями

S1="111...1", (250 символов)

S2="222...2" (250 символов),

Далее, если соединить эти строки , то ничего не потеряется

concat(S1,S2,S)

- в итоге имеем переменную S, в которой 500 символов. Длина может достигать строки в таких случаях может достигать 64 Кбайт.

* char (символы) - отдельный символ, заключенный в апострофы, например: 'A','3','a','\13'.

* integer (целые числа) - можно задавать в диапазоне от -32768 до +32767.

* real (действительные числа) - диапазон от +1E-307 до +1E308 .

* file (файлы) - допустимое в DOS имя файла.

Вернемся к факту lakes(mary,apples). С предикатом likes могут быть еще факты , такие ,например :

likes(tom,computer).

likes(kathy,computer).

Во всех этих фактах с предикатом lakes на первом месте стоят имена объектов, имеющие смысл "тот, кто любит", а на втором месте имена объектов - "вещь".

В данном примере эти два вида имен объектов имеют один и тот же тип symbol. Прежде чем показать, как выглядит раздел DOMAINS для нашего примера, рассмотрим раздел, который называется PREDICATES.

3. PREDICATES - данный раздел служит для описания используемых программой предикатов.

В нашем примере предикат likes не является встроенным предикатом, поэтому его необходимо объявить в разделе PREDICATES.

Например:

```
PREDICATES
    lakes(symbol,symbol)
```

Это описание означает, что оба объекта относятся к типу symbol. Такое описание предиката likes освобождает от необходимости использования раздела DOMAINS. Наш пример является очень простым, поэтому мы можем себе такое позволить. Реальные программы могут содержать несколько десятков предикатов, причем с различным количеством объектов. Поэтому, чтобы программа хорошо читалась - видам объектов присваивают имена, но при этом уже необходим раздел DOMAINS. Для нашего примера это будет выглядеть так:

```
DOMAINS
        person,thing = symbol
PREDICATES
        likes(person,thing)
```

где person - "тот кто любит" ,thing - "вещь".

4. CLAUSES - в данный раздел заносятся факты и правила. О содержимом этого раздела можно говорить как о данных, необходимых для работы программы. Для нашего примера этот раздел включает три факта :

```
CLAUSES
    likes(mary,apples).   likes(tom,computer).   likes(kathy,computer).
```

5. GOAL - данный раздел может располагаться перед разделом CLAUSES или после него. В этом разделе определяется цель. Цель может состоять из нескольких подцелей. Если программа предназначена для работы в пакетном режиме, раздел GOAL не может быть опущен.

В программе могут присутствовать еще два раздела, обеспечивающие определение глобальных доменов и предикатов.

6. GLOBAL DOMAINS - располагается после раздела DOMAINS.

7. GLOBAL PREDICATES - следует после раздела PREDICATES.

Определение типов данных и предикатов в этих разделах позволяет обеспечить межмодульный интерфейс.

8. DATABASE - следует перед разделом PREDICATES. Он содержит определения предикатов динамической базы данных. Если программа такой базы данных не требует, то этот раздел может быть опущен.

Турбо-Пролог обеспечивает возможность включения в программу комментариев, которые обрамляются символами /* ... */ . Также комментарием считается строка, которая начинается с символа % .

В итоге программа в законченном виде для нашего примера будет выглядеть так :

Пример 1.

```
/* НАЧАЛО */
Domains
    person,thing = symbol
Predicates
    likes(person,thing)
Goal
    likes(mary,apples),nl, write("Мэри любит яблоки").
Clauses
     likes(mary,apples).
     likes(tom,computer).
     likes(kathy,computer).
% КОНЕЦ
```

Предикаты и утверждения разных арностей.

Термин «арность» обозначает число объектов утверждения. Количество объектов в одном предикате может быть не более 50.

Ниже приведен пример предикатов и утверждений различных арностей.

```
Domains        s=string
Predicates     % раздел предикатов
        start          %  арность 0
        woman(s)       %  арность 1
        father(s,s)    %  арность 2
Clauses        % раздел утверждений
        start.         %  арность 0
        woman("Маша")      %  арность 1
        father("Петр Иванович","Маша")   %  арность 2
```

Переменные в языке Пролог.

Любое имя может быть переменной, если начинается с прописной буквы. Переменная позволяет отвечать на вопросы. Например, если необходимо узнать кому нравятся яблоки (см. пример 1), то в разделе GOAL должны написать следующее

likes(X,apples),nl, write(X," - любит яблоки").

Переменная X конкретизируется первым значением, который имеется в базе фактов с предикатом likes.

Использование правил.

В Прологе правила используются, когда необходимо сказать, что некоторый факт зависит от группы других фактов.

При описании правил часто применяют переменные. Например, правило с одной переменной и одним предикатом может выглядеть так

likes(tom,X) :- likes(X,wine).

На естественном языке это означает: «Тому нравится любой, кому нравится вино».

Правило, в котором используется одна переменная и два предиката может быть описано следующим образом

likes(tom,X) :- woman(X),likes(X,wine).

На естественном языке звучит так: «Тому нравится любая женщина, которой нравится вино».

Правило, в котором используются две переменные и три предиката может выглядеть так

is_sister(X,У) :- woman(X), parents(X,M,F), parents(Y,M,F).

X является сестрой Y, если X - женщина и имеет родителей M и F и Y имеет тех же родителей M и F.

Часто в правилах используются анонимные переменные. Анонимная переменная - это одиночный знак подчеркивания «_». Например, если нужно определить, является ли X вообще чьей-нибудь сестрой и неважно чьей, то после конкретизации X нужно записать:

... , is_sister(X,_), ...

Предыдущие примеры не учитывают факт существования нескольких вариантов, удовлетворяющих условию поиска, или просто возможность выдать содержимое всей базы.

Рассмотрим правило, позволяющее выдавать содержимое всей базы данных.

total:-woman(X),write(X),nl, fail.

В примере используется предикат fail , который осуществляет вынужденное неудачное завершение выполнения. Это один из способов организации циклов в Прологе (см. ниже), который позволяет просмотреть все факты с предикатом woman.

Ниже приведены два правила.

st:-consult("bd.pro"), ret.
ret:-retract(woman(_)), fail.

В теле первого правила с заголовком st используется два терма (предиката). С помощью встроенного предиката consult осуществляется загрузка базы данных с именем bd.pro, в которой содержатся факты с предикатом woman. Терм ret позволяет обратиться ко второму правилу с заголовком ret. Второе правило позволяет удалить все факты с предикатом woman из памяти.

Организация циклов.

В Прологе имеется два основных способа организации циклов, при организации которых необходимо учитывать встроенные «невидимые» механизмы языка Пролог.

В первом способе используется рекурсия. Общий вид правила, выполняющего рекурсию, следующий:

repetitive_rule :- %правило рекурсии
<предикаты и правила>,
repetitive_rule.

Во втором способе используется встроенный предикат fail, который вызывает откат. Данный способ называют повторением.

Откат - это механизм, который Пролог использует для нахождения дополнительных фактов и правил, необходимых при вычислении цели, если текущая попытка вычислить цель оказалась неудачной.

Общий вид правила, выполняющего повторение с помощью встроенного механизма, следующий

repetitive_rule :- %правило повторения
<предикаты и правила>,
fail.

Ниже приведен пример 2а, в котором для вычисления факториала используется рекурсия.

Пример 2а.

```
Domains r=real
Predicates
    start  calculate(r,r,r)
Goal start.
Clauses
    start:- write("Введите положительное целое число: "),
            readint(X),calculate(X,1,1).
    calculate(X,Y,Z) :- X>Y,Y1=Y+1,Z1=Z*Y1,calculate(X,Y1,Z1).
    calculate(X,Y,Z) :- X=Y,write(X,"!=",Z).
    calculate(X,Y,Z) :- X<Y,write(X,"!=1").
```

В данном примере используется символ «=». Данный символ применяется в двух случаях:

* чтобы конкретизировать переменную (присвоить конкретное значение);

* чтобы сравнить два значения.

Внешне эти два случая могут ничем не отличаться. Чтобы правильно понять смысл символа «=» в каком-то конкретном случае, необходимо проследить предыдущую цепь событий. В примере 2а в первом правиле с предикатом calculate вновь введенным переменным Y1 и Z1 присваиваются конкретные значения. Во втором правиле X и Y сравниваются, т.к. их значения передаются из заголовка правила (т.е. значения X и Y уже конкретизированы).

В примере 2в демонстрируется организация цикла без рекурсии с использованием предиката fail. Данный пример позволяет с помощью стандартных средств Пролога выбирать из каталога файлы и просматривать их содержимое.

Пример 2в.

```
Database     namefile(string) % для сохранения имени выбранного файла
Predicates   start andisk
Goal  start.
Clauses
start:-  makewindow(1,112,94," РЕЖИМ ПРОСМОТРА ТЕКСТОВЫХ ФАЙЛОВ
         (Esc-выход ...) ",0,0,25,80),    andisk.
andisk:- makewindow(2,45,47,"",1,1,4,78),dir("C:","*.*",Name),
         assert(namefile(Name)),removewindow,fail.
andisk:- not(namefile(_)),removewindow,exit,!.
andisk:- namefile(Name),concat("Содержимое файла ",Name,S),
         makewindow(3,73,27,S,1,1,23,78), file_str(Name,Str),
         display(Str),retractall(namefile(_)),removewindow,fail.
andisk:- andisk.
```

Ниже приведен пример 3, в котором с помощью предиката fail на экран выдается содержимое всей базы фактов с предикатом woman.

Пример 3.

```
Domains
        s=string
Predicates
        start
        woman(s)
```

```
Goal start.
Clauses
        start:- woman(Name),write(Name),nl,fail.
        % база фактов
        woman("Катя").  woman("Таня").  woman("Маша"). woman("Ира")
```

Кроме механизма отката в Прологе существует механизм отсечения, который можно организовать с помощью предиката cut (отсечение). В тексте программы предикат cut обозначается символом « ! ». Данный предикат устанавливает барьер, запрещающий выполнить откат ко всем альтернативным решениям текущей проблемы. Однако последующие подцели могут создать новые указатели отката и тем самым создать условия для поиска новых решений.

В примере 4 демонстрируется механизм отката. В отличии от примера 3, где на экран выдаются все факты, в данном примере, как только будет найдена женщина с именем Маша и выведено на экран, дальнейший поиск прекращается (т.е. происходит отсечение).

Пример 4.

```
Domains         s=string
Predicates      start
                woman(s)
                make_cut(s)
Goal            start.
Clauses
        start:- woman(Name),write(Name),nl,make_cut(Name),!,fail.
        make_cut(Name):-Name="Маша".
                % база фактов
woman("Катя"). woman("Таня"). woman("Маша"). woman("Ира").
```

Использование динамических баз данных.

Программы баз данных на Прологе есть частный случай систем управления базами данных. Можно выделить три основные модели организации базы данных это: иерархическая модель (данные хранятся в иерархии классов), сетевая модель (данные в виде связанных агрегатов, образующих сеть) и реляционная модель (данные в виде таблиц). Пролог ориентирован на создание баз данных на основе реляционной модели. Для работы с

базами данных существует достаточно большой набор встроенных предикатов. Некоторые из них продемонстрированы в примерах 5 и 6.

В Примере 5 происходит загрузка файла базы данных с именем «facts.bd», далее в случае существования объекта с именем Name все факты связанные с этим объектом уничтожаются, после чего база данных на диске обновляется.

Пример 5.

```
Domains
  i = integer  s = string
Database
        age(s,i)
        woman(s)
        man(s)
Predicates
        start
        conclusion(s)
Goal   start.
Clauses
start:- existfile("facts.bd"),consult("facts.bd"), % если база существует, то загружаем
        makewindow(1,27,57,"",0,0,25,80),  % создаем окно на весь экран
        /* где 1 - номер окна , 27 - атрибут экрана ( цвет символов )
                57 - атрибут рамки ,заголовок окна,
                0,0 -координаты верхнего левого угла окна (Y,X),
                25,80 -размеры окна по осям Y и X. */
write("Введите имя: "),readln(Name),nl, % вводим искомый объект
        conclusion(Name).   % переходим к правилу с одноименным заголовком
start.
conclusion(Name):-
        woman(Name),          % проверка существования объекта с именем Name
        retract(age(Name,_)), retract(woman(Name)),          % удаляем данные по объекту
        save("facts.bd"),         % сохраняем базу на диске
        write("Данные по объекту ",Name," удалены из базы !"),
        readchar(_).  % задержка экрана до нажатия любого символа
conclusion(Name):-
```

```
        not(woman(Name)),   %  если объект не существует, то выводим сообщение
        write("Объект ",Name," отсутствует в базе !"),readchar(_).
        /* содержимое файла "facts.bd"
age("Петя",20)   age("Таня",10)   age("Катя",18)
woman("Таня") woman("Катя")
man("Петя")
*/
```

Пролог позволяет создать несколько баз данных и дает возможность работать с ними как по отдельности, так и одновременно со всеми.

В примере 6 приведен фрагмент программы, который позволяет сохранить в базе данных на диске текущую дату с подтверждением пользователя. В данном примере используется база данных с именем da1. Обобщенный алгоритм программы такой:

* загружается файл базы данных с именем da1;
* если факты в базе отсутствуют, то в память заносится факт da("",1);
* с помощью системных предикатов считываем текущую дату;
* используя встроенный редактор пролога пользователь подтверждает (вводит) текущую дату;
* если формат даты правильный, то база данных с именем da1 обновляется.

Пример 6.

```
Domains         i = integer  s = string
Database - da1    % описание динамической базы данных с именем da1
        da(s,i)             % 1-й объект дата,
                            % 2-й объект - номер пользователя (активно не используется)
Predicates
        start   % предикат целевого утверждения
        dtt     % для ввода новой даты
        td(i,s) % для проверки правильности ввода с сохранением
Goal    start.
Clauses
start:- existfile("da.bd"),consult("da.bd",da1), % загрузка базы в память
        makewindow(1,0,0,"",0,0,25,80), dtt, removewindow, !.
start.
dtt:-not(da(_,_)),assert(da("",1)).        % если факты отсутствуют в базе, то
```

```
                                % добавляем в память начальный факт
dtt:-   da(H1,_),       % считываем последнюю дату работы из базы
          % определяем текущую дату и выводим ее в заголовке окна редактора
        date(G,M,D),str_int(G1,G),str_int(M1,M),str_int(D1,D),
        concat("Введите дату (Esc - отказ) [СЕГОДНЯ:",D1,S1),
        concat(S1,".",S2),concat(S2,M1,S3),concat(S3,".",S4),
        concat(S4,G1,S5),concat(S5," г.]",S6),
        makewindow(5,48,91,S6,10,10,4,58),
          % используем редактор Пролога
        editmsg(H1,H2,"F10 - выход с сохранением","","",0,"",C),
        removewindow,td(C,H2),!.
dtt:-dtt. % используем рекурсию для повторного ввода в случае ошибки
                % если дата введена правильно, то обновляем базу данных
td(C,H2):- C=0,str_len(H2,Ld),Ld<9,
        frontstr(2,H2,H3,H4),str_int(H3,Nd),Nd>0,Nd<32,
        frontstr(1,H4,K1,K2),K1=".", frontstr(2,K2,P1,P2),
        str_int(P1,Nm),Nm<13,Nm>0, frontstr(1,P2,J1,J2),J1=".",
        frontstr(2,J2,Q1,Q2),str_int(Q1,Ng),Ng>=0,
        da(_,Bb),  retractall(da(_,_)),assert(da(H2,Bb)),
        save("da.bd",da1),!.
td(C,H2):- C=1,!. % если пользователь ничего не ввел, то база не обновляется.
```

Использование списков.

Список является набором связанных объектов одного и того же доменного типа. Объектами списка могут быть целые числа, действительные числа, символы, символьные строки и структуры. Пролог позволяет выполнять со списком целый ряд операций. Их перечень включает:

* доступ к объектам списка;
* проверка на принадлежность к списку;
* разделение списка на два;
* слияние двух списков;
* сортировку элементов списка.

Основным встроенным механизмом Пролога для работы со списком является метод «разделения списка на голову и хвост». Ниже приведены примеры 7.1 и 7.2 . В одном из них показано, как можно разложить список на элементы, а в другом создать список из множества фактов и сохранить его в базе данных.

Пример 7.1. Программа демонстрации разделения списка на голову и хвост.

```
Domains         i=integer
                l=integer* %  список целых чисел
Predicates      split(l)
                sw
Goal            makewindow(1,1,7,"",0,0,25,80),sw.
Clauses
        sw:- S=[1,2,3,4,5],     % соэдаем список
                split(S).       % обращаемся к правилу разделения списка
                % разделение списка с помощью рекурсивного цикла
        split([N|S]):-write(N," ",S),nl,readchar(_),fail.
        split([]):-write("Список пуст").
        split([N|W]):-split(W).         % основная идея разделения списка
```

Пример 7.2. Программа собирает в список все объекты фактов с предикатом bn

```
Domains         i=integer
                l=integer*      % список целых чисел
Database        d(l) % база данных, где объектом в предиката является список
Predicates      sw      bn(i)   fb      sp(l)
Goal            makewindow(1,1,7,"",0,0,25,80),sw.
Clauses
        sw:-    assert(d([])),  % создаем в памяти факт БД с объектом типа "список"
                fb.             % переходим к правилу формирования списка
                % правила формирования списка с помощью бектрекинга
        fb:-bn(N),d(S),sp([N|S]),fail.
        sp(S):-retractall(d(_)),assert(d(S)),
                write(S," "),readchar(_).
                % база фактов непосредственно в тексте программы
                        bn(1). bn(2).
```

bn(3). bn(4).

Преобразование данных в Прологе.

Преобразование данных необходимо, если тип объектов встроенного предиката отличается от типа объектов предиката, определенного пользователем. Все предикаты преобразования данных содержат два объекта. Имена предикатов показывают тип выполняемого преобразования , а так же указывают и порядок следования объектов.

Особенностью является то, что предикаты преобразования имеют два направления. Основные виды преобразований представлены в примере 8.

Пример 8.

```
Predicates
        start  conv(char,string,integer,real)
Goal start.
Clauses
    start:-C='A',S="A",I=65,R=3.14,nl,conv(C,S,I,R).
    conv(C,S,I,R):-char_int(C,X),nl,write(X),fail.          %  X=65
    conv(C,S,I,R):-char_int(C1,I),nl,write(C1),fail.        %  C1=A
    conv(C,S,I,R):-str_int(S1,I),nl,write(S1),fail.%  S1=65
    conv(C,S,I,R):-str_char(S,C1),char_int(C1,I1),nl,write(I1),fail.   %  I1=65
conv(C,S,I,R):-str_real(S1,R),nl,write(S1),fail.         S1=3.14
```

Кроме встроенных предикатов преобразования данных пользователь может ввести свои. В примере 9 приведено нестандартное преобразование цифр в соответствующие им слова.

Пример 9.

```
Predicates
    start(string)  conv(string,integer)
Goal
    start("0123456789").
Clauses
start(S):- frontstr(1,S,S1,S2),  % расщепляем строку на элементы
        str_int(S1,K),conv(X,K),write(X),nl,start(S2).
```

```
        % База фактов преобразования
conv("ноль",0).     conv("один",1).     conv("два",2). conv("три",3).
conv("четыре",4).   conv("пять",5).     conv("шесть",6).    conv("семь",7).
conv("восемь",8).   conv("девять",9).
```

Логические возможности Пролога.

С помощью правил можно описать какой-либо процесс принятия решения или логический вывод. В этом случае в программе могут использоваться не только встроенные механизмы вывода или вывод, смысл которого сводится просто к поиску объекта в базе фактов, но и собственные пользовательские правила принятия решения.

Рассмотрим пример программы (см. пример 10), в которой определяется родственная связь на основе неявно заданных фактов, т.е. в базе отсутствуют факты типа «Объект_1 является_сестрой Объекта_2». В соответствии с примером, если ввести имя «Таня», то результатом вывода будет фраза: « Имеется брат Борис».

Пример 10. Определение родственной связи

```
Domains         s=string
Predicates
        start w(s) p(s,s,s) per(s) is_rs(s,s) m(s) wiw(s)
Goal    start.
Clauses
        start:- makewindow(1,52,37,"Определение родственной связи",0,0,25,80),
                write("  Введите имя : "),readln(X),nl,per(X).
        per(X):- is_rs(X,Y),X<>Y,wiw(Y),fail. %  основное правило проверки
        per(X):- nl,nl,write("Конец поиска."), readchar(_).
        % правило проверки общих родителей
        is_rs(X,Y):- p(X,M,F),p(Y,M,F).
        % уточнение и вывод родственной связи
        wiw(Y):-w(Y),write("  Имеется сестра ",Y),nl,!.
        wiw(Y):-m(Y),write("  Имеется брат ",Y,"."),nl,!.
  % база фактов женщин
w("Катя"). w("Света"). w("Таня").
  % база фактов мужчин
```

```
m("Дима").   m("Борис").   m("Вася").
  % база фактов родителей
p("Катя","Лена","Сергей").  p("Света","Галина","Сергей").
p("Таня","Зоя","Костя").              p("Дима","Лена","Сергей").
p("Вася","Лена","Сергей").  p("Борис","Зоя","Костя").
```

Реализация вспомогательных функций на Прологе.

В Турбо-Прологе имеется много встроенных предикатов для реализации вспомогательных функций, которые облегчают создание больших систем. Перечислим некоторые из них:

* вывод и просмотр файла (строки) со скроллингом;
* редактирование файла (строки);
* вывод текущего каталога;
* реализация многооконного интерфейса и т.д.

Однако, это не все функции, которые требуется реализовать при создании больших систем.

В примере 11 приведена программа реализации меню. Пункты меню можно выбирать с помощью клавиш стрелок или с помощью мышки.

Пример 11.

```
Constants
    cz0=63  % начальный цвет  пунктов меню
    col=28  % цвет пометки пунктов меню
    c1=63   % цвет фона главного окна
    c2=118   % цвет рамки
    kp=5     % количество пунктов в меню
Domains        i = integer  s = string  r=real
Database       ar(i,i) mn(i,i,i)
Predicates
        mkw(i,i,i,s,i,i,i,i)  rmw
        r_k(i)  c_k(i,i)   ko(i,s,i,i,i)  ord(i,i,i)
        bor  an(i,i)  start  menu(i,i)  ed_ar(i,i) uz(i,i,s)
        codm(i,i) ed_mn(i,i,i) my(i,i) pm(i,i) zn(i) initm
```

```
Goal    initm,% инициализация мыши
        start.% запуск меню
Clauses
start:- mkw(1,31,52," ОТДЕЛ ГРАФИЧЕСКИХ СИСТЕМ ",0,0,25,80),
        Xnt=22,mkw(2,0,0,"",8,Xnt,9,33),
        Xn=Xnt-2,mkw(3,c1,c2," Г Л А В Н О Е   М Е Н Ю ",7,Xn,9,33),
        menu(1,kp),pm(1,1),zn(Xn), ed_ar(80,1),bor,!.
 % Вывод пунктов меню с помощью рекурсивного правила
menu(X,Y):- ko(X,A,Y1,X1,C),cursor(Y1,X1),write(A),
                str_len(A,L),field_attr(Y1,X1,L,C),X2=X+1,X<Y,menu(X2,Y),!.
menu(X,Y):-X=Y,!.
 % создание окна
mkw(A1,A2,A3,S,B1,B2,B3,B4):-makewindow(A1,A2,A3,S,B1,B2,B3,B4,1,255,"r¬L-=¦").
rmw:-removewindow,!.  % удаление окна
% главный цикл опроса
bor:- keypressed,ar(_,R),r_k(K), an(K,R),fail.
bor:- codm(K,R),an(K,R),fail,!.
bor:- bor.
/*   основное правило   в  меню  */
an(Kod,R):- ko(R,A,Y,X,C),ord(R,Kod,Rs), pm(1,Rs),pm(0,R),ed_ar(Kod,Rs),!.
an(Kod,R):-uz(R,Kod,S),existfile(S),!.
an(Kod,R):-uz(R,Kod,S),mkw(26,112,67,"",4,10,3,52),cursor(0,5),
                write("  ФАЙЛ ",S," отсутствует !  "),readchar(_),rmw,!.
an(Kod,R):-R=5,Kod=13,mkw(10,7,0,"",0,0,25,80), exit,!.
an(Kod,R):-!.
% факты срабатывания пункта меню
% номер пункта, код срабатывания, имя файла
        uz(1,13,"form.exe").uz(2,13,"registr.exe").
        uz(3,13,"report.exe").uz(4,13,"servis.exe").
% модификация служебной базы
ed_ar(A,B):-retractall(ar(_,_)),assert(ar(A,B)),!.
% определение следующего пункта меню
ord(R,Kod,Rs):-R<kp,Kod=80,Rs=R+1.
```

```
ord(R,Kod,Rs):-R>1,Kod=72,Rs=R-1.
% факты пунктов меню
% ko(номер пункта,имя пункта,координата по Y, координата по X, цвет)
ko(1,"1. Формирование заказа",1,2,cz0).
ko(2,"2. Регистрация клиента",2,2,cz0).
ko(3,"3. Формирование отчета",3,2,cz0).
ko(4,"4. Сервисные функции",4,2,cz0).
ko(5,"5. Выход",5,2,cz0).
%Чтение (расширенного) кода
 r_k(Kod):- readchar(C), char_int(C,A),c_k(A,Kod),!.
c_k(A,Kod):- A<>0,Kod=A,!.
c_k(0,Kod):- readchar(C),char_int(C,Kod),!.
% пометка или разметка пунктов меню
pm(1,R):-ko(R,A,Y,X,_),str_len(A,L),field_attr(Y,X,L,col),!.
pm(0,R):-ko(R,A,Y,X,C),str_len(A,L),field_attr(Y,X,L,C),!.
pm(N,R).
 % Правила для мышки
initm:-  bios($33, reg(0,0,0,0,0,0,0,0),reg(_,_,_,_,_,_,_,_)),
         bios($33, reg(1,0,0,0,0,0,0,0), reg(_,_,_,_,_,_,_,_)),!.
initm.
codm(K,R):- bios($33, reg(3,0,0,0,0,0,0,0),reg(_,M,X,Y,_,_,_,_)),M=1,
                  my(Y,R),mn(R,Xn,Xk),X>=Xn,X<=Xk,       ar(_,Rr),pm(0,Rr),
                  pm(1,R),ed_ar(0,R),K=13,!.
my(72,1).my(80,2).my(88,3).my(96,4).my(104,5).
zn(O):- ko(R,A,Y,X,_),str_len(A,L),Xn=(O+X+1)*8,Xk=Xn+L*8-1,ed_mn(R,Xn,Xk),fail.
zn(O).
ed_mn(R,Xn,Xk):-retractall(mn(R,_,_)),assert(mn(R,Xn,Xk)),!.
```

Краткий список встроенных предикатов.

(полный список приведен в файле prolog.hlp)

Предикаты ввода.

readln(StringVariable) - читает строку.

readint(IntgVariable) - читает целое число.

readreal(RealVariable) - читает действительное числощ.

eadchar(CharVariable) - читает символ.

file_str(DosFileName,StringVariable) - читает строку из файла.

inkey(CharVariable) - читает ключ (символ).

keypressed - проверяет, нажата ли клавиша.

Предикаты вывода.

write(Variable|Constant *) - производит запись на текущее устройство вывода.

nl - перевод строки.

Предикаты для работы с файлами.

openread(SymbolicFileName,DosFileName) - открывает файл для чтения.

openwrite(SymbolicFileName,DosFileName) - открывает файл для записи.

openappend(SymbolicFileName,DosFileName) - открывает файл для добавления.

openmodify(SymbolicFileName,DosFileName) - открывает файл для чтения/записи.

readdevice(SymbolicFileName) - определяет или считывает символическое имя файла устройства ввода.

writedevice(SymbolicFileName) - определяет или считывает символическое имя файла устройства вывода.

filemode(SymbolicFileName,FileMode) - установка или чтение типа файла.

closefile(SymbolicFileName) - закрывает файл.

filepos(SymbolicFileName,FilePosition,Mode) - установка или чтение позиции указателя.

eof(SymbolicFileName) - проверка на конец файла

flush(SymbolicFileName) - очищает содержимое буфера.

existfile(DosFileName) - проверка существования файла.

deletefile(DosFileName) - удаляет файл.

renamefile(OldDosFileName,NewDosFileName) - переименовывает файл.

disk(DosPath) - устанавливает или показывает накопитель или путь.

Предикаты экрана.

scr_attr(Row,Column,Attr) - устанавливает или считывает атрибут.

field_str(Row,Column,Length,String) - записывает или читает строку.

field_attr(Row,Column,Length,Attr) - устанавливает или читает атрибут поля экрана.

cursor(Row,Column) - считывает или устанавливает позицию курсора.

cursorform(Startline,Endline) 0<Startline<14, 0<Endline<14

- считывает или устанавливает форму курсора.

attribute(Attr) - считывает или устанавливает цвет фона текущего окна.

Предикаты работы с окнами.

makewindow(WindowNo,ScrAtt,FrameAtt,Framestr,Row,Column,Height,Width)

- - создает окно.

shiftwindow(WindowNo) - меняет текущее окно или считывает номер текущего окна.

gotowindow(WindowNo)- -активизирует окно с заданным номером.

existwindow(WindowNo) - проверяет существование окна.

removewindow - удаляет текущее окно.

clearwindow - чистка окна.

window_str(ScreenString) - записывает (или считывает) строку в текущее окно

window_attr(Attribute) - определяет атрибуты текущего окна

scroll(NoOfRows,NoOfCols) - сдвиг содержимого текущего окна.

Предикаты для работы со строками.

frontchar(String,FrontChar,RestString) - разделяет заданную строку на первый символ и оставшуюся часть.

fronttoken(String,Token,RestString) - разделяет строку на лексему и остаток.

frontstr(Lenght,Inpstring,StartString,RestString) - разделяет строку на две части, количество первой части равно Lenght.

concat(String1,String2,String3) - String3 = String1 + String2.

str_len(String,Length) - определяет длину строки.

Предикаты преобразования данных.

char_int(CharParam,IntgParam) - преобразует символ в целое число или наоборот.

str_int(StringParam,IntgParam) - преобразует строку в целое число или наоборот.

str_char(StringParam,CharParam) - преобразует строку в символ или наоборот.

str_real(StringParam,RealParam) - преобразует строку в действительное число или наоборот.

upper_lower(StringInUpperCase,StringInLowerCase) - преобразует прописные буквы в строчные и наоборот.

Предикаты баз данных.

consult(DosFileName) - загружает или добавляет текстовый файл базы данных в ОП.

consult(DosFileName,InternalDatabaseName) - загружает в ОП группу поименованную группу фактов.

save(DosFileName) - записывает на диск все факты динамической БД.

save(DosFileName,InternalDatabaseName) - записывает на диск поименованную группу фактов БД.

assert(Term) - добавляет факт БД в ОП.

retractall(Term) - удаляет все факты с указанным термом.

retractall(_, InternalDbaseName) - удаляет все факты поименованной группы.

Предикаты для работы с редактором.

display(String) -) - показывает в текущем окне строку(до 64Кбайт).

edit(InputString,OutputString) - вызов редактора.

editmsg(InputString,OutputString,Headstr,Headstr2,Msg,Pos,Helpfilename,RetStatus) - вызов редактора с дополнительными возможностями.

Системные предикаты.

system(DosCommandString) - выполнение команд DOS.

dir(Path,Filespec,Filename) - выводит текущий каталог.

comline(LineBuffer) - читает параметры командной строки.

port_byte(PortNo,Value) - посылает байт в порт или читает его из порта.

ptr_dword(8086Ptr,Segment,Offset) - читает строку или адрес строки.

memword(Segment,Offset,Word) - запоминаетр или считывает слово по заданному адресу.

membyte(Segment,Offset,Byte) - запоминает или считывает байт.

bios(Interruptno,reg(AXi,BXi,CXi,DXi,SIi,DIi,DSi,ESi), reg(AXo,BXo,CXo,DXo,SIo,DIo,DSo,ESo)) - объявляет прерывания для вызова процедур BIOS/

exit - выход из программы.

storage(StackSize,HeapSize,TrailSize) - определяет размер имеющейся памяти.

sound(Duration,Frequency) - звуковой сигнал с параметрами.

beep - звуковой сигнал.

date(Year,Month,Day) - установка или считывание даты.

time(Hours,Minutes,Seconds,Hundredths) - устанавливает или считывает системное время.

findall(Variable, Atom, ListVariable) - собирает значения, возникающие в процессе бектрегинга, в список.

free(Variable) - проверяет свободная ли переменная.

bound(Variable) - проверяет связана ли переменная.

Арифметические операции.

+, -, *, /, mod, div

Операции отношения.

>, <, =, >=, <=, <>, ><

Логические операции.

not(Atom) - отрицание.

and , or

Функции.

sin, cos, tan, arctan, ln, log, exp, sqrt, round, trunc, abs

Порядок выполнения работы.

1. Ознакомиться с основными понятиями и механизмами языка Пролог.
2. Изучить работу основных встроенных предикатов и механизмов Пролога на конкретных примерах из методических указаний.
3. Сформулировать основные отличительные особенности Турбо-Пролога 2.0 относительно языка процедурного типа (Pascal, CI++ и т.п.).
4. Выполнить задание в соответствии с вариантом, полученным у преподавателя.

Отчет должен включать:

- Цель работы и конкретное задание.
- Текст отлаженной программы с комментариями в соответствии с заданием.

Порядок проведения защиты :

- Ответить на вопросы, предложенные преподавателем по теоретическому материалу.
- рассказать принцип работы программы, выполненной в соответствии с вариантом задания программы.

Задания к 1-й части лабораторных работ по СИИ.

Общие требования:

1. Кроме часто используемых служебных данных небольшого объема, все остальные данные должны подгружаться с диска.
2. Обеспечить своевременную чистку стека, т.е. программа не должна прерываться при большом количестве повторений.
3. Обеспечить приемлемую универсальность программы, т.е. в правилах необходимо использовать переменные, постоянные составляющие определять как факты и т.п.

Вариант 1. Дана строка (до 64 Кбайт) в текстовом файле. Создать динамическую базу предложений данной строки. По номеру предложения выдавать его на экран. Для создания интерфейса использовать стандартные средства.

Вариант 2. Дан текстовый файл. Создать новый файл, преобразовав старый файл путем перевода всех букв (латинского или русского алфавитов) в строчные или прописные по желанию пользователя. Обеспечить просмотр файлов с помощью стандартных средств.

Вариант 3. Дан текстовый файл. Создать новый файл путем удаления лишних пробелов между словами. Обеспечить просмотр файлов с помощью стандартных средств.

Вариант 4. Дана строка (до 64 Кбайт). Создать базу фактов слов данной строки. По номеру слова выдавать его на экран. Использовать стандартные средства для разработки интерфейса.

Вариант 5. Дан текстовый файл. Создать список слов данного файла и сохранить его как элемент динамической базы данных. Реализовать функцию проверки на вхождение какого-либо слова в сформированный список.

Вариант 6. Дана база фактов электрической принципиальной схемы однокаскадного усилителя. Программа должна считывать факты и в графическом режиме в соответствии с фактами выводить на экран схему с соответствующими комментариями.

Вариант 7. Создать базу данных записной книжки. Поля: ФИО, ТЕЛЕФОН, ДАТА РОЖДЕНИЯ. С помощью меню дать пользователю возможность выбрать одну из следующих функций: - просмотр БД; - поиск по фамилии; - добавление нового факта.

Вариант 8. Реализовать следующие функции: - выбор файла с помощью меню в текущем каталоге; - редактирование выбранного файла; - определять входит ли данное слово в выбранный файл.

Вариант 9. Создать базу данных сотрудников. Поля: ФИО, ДОЛЖНОСТЬ, ОБРАЗОВАНИЕ. Реализовать функции: - просмотр полей; редактирование полей.

Вариант 10. Создать базу данных студентов. Поля: ФИО, ГРУППА, СРЕДНИЙ ОЦЕНКА. Реализовать функции: - просмотр полей; - сортировка по ФИО.

Вариант 11. Создать многооконный интерфейс используя стандартные средства. В одном из окон пользователю задаются вопросы, в другом пользователь отвечает на них. В третьем окне отмечается пройденный путь диалога.

Вариант 12. Создать словарь перевода иностранных слов (язык выбрать самостоятельно) с помощью динамической базы данных. Реализовать функции: - перевод двунаправленный; - выдавать антонимы. Использовать стандартные средства диалога.

Вариант 13. Реализовать вывод на экран электрическую принципиальную схему эмиттерного повторителя. Обеспечить возможность выбора элементов с помощью «мышки» и вывода комментария к выбранному элементу. Использовать динамическую базу данных.

Вариант 14. Создать базу данных клиентов фирмы. Поля: ФИО, ФИРМА, ЗАКАЗ, ДАТА ОПЛАТЫ. Реализовать функции: - просмотр содержимого базы; - вывод клиентов по диапазону даты оплаты.

Вариант 15. Разработать меню для графического режима с использованием «мышки» и динамической базы данных.

Вариант 16. Дан текстовый файл. Реализовать его вывод на экран с пометкой всех слов начинающихся на гласную букву и длиной более N.

Вариант 17. Дана строка (до 64 Кбайт) в текстовом файле. Создать динамическую базу предложений данной строки. По введенным начальным фразам выдавать все предложение на экран. Для создания интерфейса использовать стандартные средства.

Вариант 18. Создать словарь перевода иностранных слов. Реализовать функции: - двунаправленный перевод; - выдавать синонимы. Использовать стандартные средства диалога и динамическую базу данных.

Вариант 19. Дан текстовый файл (объем > 64 Кбайт). Обеспечить возможность поиска в этом файле предложений по одному или более ключевым словам. Ключевые слова вводятся с клавиатуры, а найденные предложения должны сохраняться в динамической базе данных.

Вариант 20. Дана база фактов предложений на русском языке. Обеспечить вывод на экран данных предложений со скроллингом. Реализовать возможность перестановки двух предложений, номера которых пользователь может ввести с клавиатуры.

Вариант 21. Обеспечить возможность построения электрической принципиальной схемы на экране в графическом режиме.