МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»

Кафедра системного анализа и управления

В.В. ТУГОВ, Т.В. ГАИБОВА, Н.А. ШУМИЛИНА

# ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ И ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ЛАБОРАТОРНЫМ РАБОТАМ

Рекомендовано к изданию Редакционно-издательским советом
государственного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»

Оренбург 2006

УДК 004.89 (076.5)
ББК 32.813в73
Т 81

Рецензент кандидат технических наук, доцент А.М. Черноусова

**Тугов В.В.**

Т 81 **Интеллектуальные технологии и представление знаний [Текст]: методические указания / В.В. Тугов, Т.В. Гаибова, Н.А. Шумилина. – Оренбург: ГОУ ОГУ, 2006.- 21 с.**

Методические указания предназначены для выполнения лабораторных и самостоятельных работ по дисциплине «Интеллектуальные технологии и представление знаний» для студентов направления 220100 Системный анализ и управление. Могут быть использованы студентами других специальностей и направлений при изучении курса «Системы искусственного интеллекта».

ББК 32.813в73

© Тугов В.В.,
Гаибова Т.В.,
Шумилина Н.А., 2006
© ГОУ ОГУ, 2006

# *Содержание*

# 1 Цель работы

Ознакомиться с основами синтаксиса и семантики Prolog-программ.

# 2 Краткие теоретические сведения

Язык программирования Пролог предполагает получение решения задачи при помощи логического вывода из ранее известных фактов. Программа на языке Пролог не является последовательностью действий – она представляет собой набор фактов и правил, обеспечивающих получение логических заключений из данных фактов. Поэтому Пролог считается декларативным стилем программирования. Программируя в декларативном стиле, программист должен описать, что нужно решать.

Противоположным ему стилем программирования является императивный, в котором программа представляет последовательность операторов (команд, выполняемых компьютером). Программируя в императивном стиле, программист должен объяснить компьютеру, как нужно решать задачу.

Соответственно и языки программирования делят на императивные и декларативные.

К императивным языкам относятся такие языки программирования как Паскаль, Бейсик, Си и т.д. В отличие от них, Пролог является декларативным языком.

Теоретической основой Пролога является раздел математической логики исчисления предикатов. Само название языка есть сокращение – ПРОграммирование в терминах ЛОГики (PROgrammation in LOGique). Идея использовать логику в качестве языка программирования возникла впервые в начале 70-х годов. Одними из первых исследователями, разработавшими эту идею, были Р. Ковальский и Маартен ван Эмден (Эдинбург), А. Колмероэ (Марсель). В 1980 году Кларк и Маккейб в Великобритании разработали версию Пролога для персональных ЭВМ. Особенно широко Пролог применяется при решении задач искусственного интеллекта, в частности в такой области ИИ, как экспертные системы.

Пролог базируется на фразах (предложениях) Хорна, являющихся подмножеством формальной системы, называемой логикой предикатов. Пролог использует упрощенную версию синтаксиса логики предикатов, он прост для по-

нимания и очень близок к естественному языку. Пролог имеет механизм вывода, который основан на сопоставлении образцов. С помощью подбора ответов на запросы Пролог извлекает хранящуюся информацию. Пролог пытается ответить на запрос, запрашивая информацию, о которой уже известно, что она истинна.

Одной из важнейших особенностей Пролога является то, что он ищет не только ответ на поставленный вопрос, но и все возможные альтернативные решения. Вместо обычной работы программы на процедурном языке от начала и до конца, Пролог может возвращаться назад и просматривать все остальные пути при решении всех частей задачи. При программировании на Прологе усилия программиста направлены на описание логической модели фрагмента предметной области решаемой задачи в терминах объектов предметной области, их свойств и отношений между собой, а не детальной программной реализации.

Пролог очень хорошо подходит для описания взаимоотношений между объектами. Поэтому Пролог называют реляционным языком. Причем «реляционность» Пролога значительно более мощная и развитая, чем «реляционность» языков, используемых для обработки баз данных. Часто его используют для создания СУБД, где применяются очень сложные запросы, которые довольно легко записать на Прологе.

В Прологе очень компактно, по сравнению с императивными языками, описываются многие алгоритмы. По статистике, строка исходного текста программы на языке Пролог соответствует четырнадцати строкам исходного текста программы на императивном языке, решающем ту же задачу. Пролог-программу легко писать, понимать и отлаживать. Это приводит к тому, что время разработки приложения на языке Пролог во многих случаях на порядок быстрее, чем на императивных языках.

### 2.1 Основные понятия языка Пролог

Пролог - язык логического программирования (ПРОграммирование в ЛОГике), используемый для представления и манипулирования знаниями в системах ИИ.

Логическое программирование - программирование, основанное на использовании механизма доказательства теорем в логике, который позволяет выяснить, является ли противоречивым некоторое множество логических формул. При этом программа рассматривается как набор логических формул, описывающих предметную область, совместно с теоремой, которая должна быть доказана. Логическое программирование избавляет программиста от необходимости

определения точной последовательности шагов выполнения вычислений.

Программа на языке Пролог - набор утверждений, составляющих базу фактов и базу правил, к которым допустимо обращение с запросами, касающимися их содержимого. Запросы называются также целевыми утверждения.

Утверждение языка Пролог - линейная конструкция из термов, заканчивающаяся точкой.

Терм языка Пролог - это либо константа, либо **переменная**, либо **структура**. Константами являются атомы и числа.

Константы используются для обозначения (именования) конкретных объектов предметной области и конкретных отношений между ними.

Атом языка Пролог - имя, число без знака или символ, обязательно начинающаяся со строчной буквы.

Переменная языка Пролог - последовательность букв, цифр и знака "подчеркивание", обязательно начинающаяся с прописной буквы. Область известности (лексический диапазон) переменных - одно предложение. Поэтому одно и то же имя в двух предложениях обозначает две разные переменные.

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".

Факты образуют базу данных Пролог-программы.

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 быть 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 языка Пролог - конструкция из функтора и компонент, имеющая следующий вид:

<функтор>(<компонент-1>,<компонент-2>, ... ,<компонент-n>),

где в качестве функтора должен выступать атом, а компонентом может быть любой терм (в том числе и структура).

База фактов в языке Пролог - последовательность утверждений, описывающих факты предметной области в виде структур, функторами которых являются атомы - имена отношений (предикатные буквы), а компонентами - предметные константы. Каждый факт представляет собой элементарную формулу (предикат) исчисления предикатов первого порядка и является **дизъюнктом Хорна**, состоящим из одного (положительного) литерата. При описании фактов переменные не используются.

База правил - совокупность правил в программе на языке ПРОЛОГ.

Правило представляет собой дизьюнкт Хорна, содержащий один положительный литерал и несколько отрицательных, и записывается следующим образом

<структура-0>:-<структура-1>, ... ,<структура-N>.

Здесь каждая структура представляет собой предикат, областью действия переменных является все правило.

Правило может трактоваться следующим образом: предикат, являющийся заголовком правила доказан (удовлетворен), когда доказан каждый предикат тела правила.

В качестве предикатов, составляющих тело правила, могут выступать:

- предикаты, фигурирующие в базе фактов;
- предикаты, совпадающие с заголовком других правил;
- встроенные предикаты систем программирования Пролог.

Встроенный предикат - предикат, выводимость (согласованность) которого устанавливается непосредственно системой программирования Пролог.

Запрос на языке Пролог - утверждение, рассматриваемое в качестве целевого, имеющее следующий вид:

<структура-1>, ...,<структура-N>.

Здесь каждая структура представляет собой предикат, возможно, содержащий переменные. Причем областью действия переменной является все утверждение в целом, т. е. одна и та же переменная в пределах утверждения означает один и тот же объект.

Конкретизация переменной - связывание переменной языка Пролог с конкретным значением. Конкретизация переменной обеспечивает возврат искомых значений переменных по запросам.

Операция сопоставление берет два терма и пытается сделать их идентичными, подбирая соответствующую конкретизацию переменных в обоих термах. Сопоставление, если оно завершается успешно, в качестве результата выдает наиболее общую конкретизацию переменных.

## 2.2 Семантические модели Пролога

В Прологе наиболее часто используются две семантические модели: декларативная и процедурная. Семантические модели предназначены для объяснения смысла программы.

В декларативной модели рассматриваются отношения, определенные в программе. Она определяет, является ли целевое утверждение истинным, исходя из данной программы, и если оно истинно, то для какой конкретизации переменных. Для этой модели порядок следования предложений в программе и условий в правиле не важен.

Процедурная модель рассматривает правила как последовательность шагов, которые необходимо успешно выполнить для того, чтобы соблюдалось отношение, приведенное в заголовке правила. Это процедура достижения списка целей в контексте данной программы. Процедура выдает истинность или ложность списка целей и соответствующую конкретизацию переменных. Процедура осуществляет автоматический возврат для перебора различных вариантов.

Множество предложений, имеющих в заголовке предикат с одним и тем же именем и одинаковым количеством аргументов, трактуются как процедура. Для процедурной модели важен порядок, в котором записаны предложения и условия в предложениях. Поэтому порядок может повлиять на эффективность программы, неудачный порядок может даже привести к бесконечным рекурсивным вызовам.

Имея декларативно правильную программу, можно улучшить ее эффективность путем изменения порядка предложений и целей при сохранении ее декларативной правильности. Переупорядочивание - один из методов предотвращения зацикливания.

### 2.3 Структура программы языка Пролог

Программа на Прологе может состоять из следующих разделов:

- директивы компилятора;
- CONSTANS – раздел описания констант;
- DOMAINS – раздел описания доменов;
- DATABASE – раздел описания предикатов внутренней базы данных;
- PREDICATES – раздел описания предикатов;
- CLAUSES – раздел описания предложений;
- GOAL – раздел описания внутренней цели.

В программе не обязательно должны быть все приведенные разделы. Так, например, она может состоять из одного описания цели:

```
GOAL
write (“hello”), readchar (_).
```

Эта программ выведет сообщение (с помощью стандартного предиката write) и будет ожидать нажатия пользователем любой клавиши (стандартный предикат readchar читает символ).

Однако, как правило, программа содержит, по меньшей мере, разделы PREDICATES и CLAUSES.

В программе может быть несколько разделов описаний DOMAINS, PREDICATES, DATABASE, CLAUSES. Однако разделов GOAL не может быть в программе более одного.

Порядок разделов может быть произвольным, но при этом константы, домены и предикаты должны быть определены до их использования. Однако в разделе DOMAINS можно ссылаться на домены, которые будут объявлены поз-

же.

Рассмотри разделы немного подробнее.

### 2.3.1 Директивы компилятора

В самом начале программы можно расположить одну или несколько директив компилятора, которые дают компилятору дополнительные инструкции по обработке программы.

Для примера рассмотрим несколько наиболее широко используемых директив компилятора.

Директива *trace* применяется при отладке программы для трассирования. Этот процесс немного похож на пошаговое выполнение императивной программы с отслеживанием значений переменных. Трассировка позволяет пользователю наблюдать за ходом выполнения программы. Если после ключевого слова *trace* указаны имена предикатов через запятую, то трассировка идет только по этим предикатам. В противном случае — по всем предикатам программы. После завершения отладки трассировку нужно выключить.

Во время исполнения программы при включенной трассировке в специальном окне трассировки будет отображаться следующая информация:

- после слова «CALL» будет указано имя выполняемого предиката (текущая подцель) и его параметры;

- после слова «FAIL» будет выводиться имя текущей подцели, которая не была достигнута;

- после слова «RETURN» будет выводиться результат вычисления текущей подцели, в случае успеха. При этом если у подцели есть еще альтернативы, к которым возможен возврат, то перед именем предиката высвечивается звездочка («*»);

- слово «REDO» перед именем предиката указывает на то, что произошел возврат и происходит вычисление альтернативного решения.

Переход от подцели к подцели вызывается нажатием функциональной клавиши *F10.* При этом в окне редактирования выполняющуюся подцель указывает курсор, она также отображается в окне трассировки с параметрами и дополнительной информацией.

Директива *nowarnings* используется для подавления предупреждения системы о том, что какая-то переменная встречается в предложении только один раз. Эту директиву стоит использовать только в хорошо отлаженных программах. Как правило, для подавления такого предупреждения *(«WARNING: The variable is only used once»)* достаточно заменить переменную, которая встретилась только один раз, на анонимную переменную.

С помощью директивы *include* при компиляции в исходный текст можно вставить содержимое некоторого файла.

Заметим, что многие директивы компилятора могут быть не только расположены в тексте программы, но и установлены в меню среды разработки Турбо Пролога *(Options->Compiler Directives).* Значение директивы компилятора, указанное в тексте программы, имеет более высокий приоритет, чем значение, установленное в меню.

### 2.3.2 Раздел описания констант

Раздел, озаглавленный зарезервированным словом *CONSTANTS,* предназначен для описания констант. Объявление константы имеет вид:

<имя константы>=<значение>

Имя константы должно быть идентификатором, то есть оно может состоять из английских букв, цифр и знака подчеркивания, причем не может начинаться с цифры. Каждое определение константы должно размещаться в отдельной строке. Например, col=17, x=3.62, c=’W’, st="Выход".

В разделе описания констант можно использовать в качестве первого символа имени константы прописные символы, потому что в этом разделе прописные и строчные символы не различаются. Однако при использовании констант в разделе описания предложений нужно задействовать в качестве первого символа имени константы только строчные символы, чтобы Пролог-система не восприняла константу как переменную.

Разделов описания констант может быть несколько, но каждая константа должна быть определена до ее первого использования.

### 2.3.3 Раздел описания доменов

Раздел описания доменов является аналогом раздела описания типов в обычных императивных языках программирования и начинается с ключевого слова 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.

В Турбо Прологе имеются стандартные домены, которые не нужно указы-

вать в разделе описания доменов. Основные стандартные домены — это:

- symbol (символические имена) - это последовательность букв латинского алфавита, 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. Например: "today","123","ПРИВЕТ".

- char (символы) - отдельный символ, заключенный в одиночные 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.

Объявление домена имеет следующий вид:

<имя домена>=<определение домена>

или

file=<имя файлового домена1>;...;<имя файлового доменаN>

Удобно использовать описание доменов для сокращения имен стандартных доменов. Например, чтобы не писать каждый раз *integer,* можно написать следующее:

```
DOMAINS
i=integer
```

и далее использовать вместо ключевого слова integer односимвольное обозначение i.

Из доменов можно конструировать составные или структурные домены (структуры). Структура описывается следующим образом:

<имя структуры>=<имя функтора>(<имя домена первой компоненты>,...,<имя домена последней компоненты>) [;<имя функтора>(...)]*

Каждая компонента структуры в свою очередь может быть структурой. Например, структура, описывающая точку на плоскости и имеющая две компо-

ненты (координаты точки)

point= p(integer,integer)

может входить в качестве компоненты в более сложную структуру, описывающую треугольник:

triangle=tr(point, point, point)

В описание структуры могут входить альтернативы, разделенные символом *«;»* или ключевым словом *«or»*.

Так, структуру, описывающую точку и на плоскости, и в пространстве, можно задать следующим образом:

point = p(integer, integer);p(integer, integer, integer)

Описание файлового домена имеет вид:

file = <символическое имя файла 1>;...; <символическое имя файла N>

Для представления данных в Турбо Прологе, в отличие от стандартных алгоритмических языков программирования, используются не массивы, а списки. Списковый домен задается следующим образом;

<имя спискового домена>=<имя домена элементов списка>*

Например, список целых чисел описывается так:

list_of_integer=integer*

### 2.3.4 Раздел описания предикатов внутренней базы данных

Начинается раздел описания предикатов внутренней базы данных с зарезервированного слова DATABASE и описываются в нем те предикаты, которые можно в процессе выполнения программы добавлять во внутреннюю базу данных или удалять оттуда. Описываются предикаты базы данных аналогично предикатам в разделе описания предикатов PREDICATES, который рассматривается ниже. Если программа такой базы данных не требует, то этот раздел может быть опущен.

### 2.3.5 Раздел описания предикатов

В разделе, озаглавленном зарезервированным словом PREDICATES, содержатся описания определяемых пользователем предикатов. В традиционных языках программирования подобными разделами являются разделы описания заголовков процедур и функций. Описание n-местного предиката имеет следующий вид:

<имя предиката>(<имя домена первого аргумента>,..., <имя домена n-го аргумента>).

Домены аргументов должны быть либо стандартными, либо объявленными в разделе описания доменов. Имя предиката в Турбо Прологе должно быть идентификатором, т.е. оно должно состоять только из английских букв, цифр и символа подчеркивания, причем не может начинаться с цифры. Кроме этого, нельзя использовать пробел, знак «минус», звездочки, или наклонную вправо черту в названиях (именах) предиката.

Например, предикат, описывающий отношение «мама», может быть описан следующим образом:

```
PREDICATES
mother(string, string)
```

Это описание означает, что у предиката два аргумента, причем оба строкового типа. Данный пример является очень простым. 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.

Один предикат может иметь несколько описаний. Это используется, когда

необходимо, чтобы предикат работал с аргументами различной природы. При этом возможны несколько вариантов использования этого предиката. Первый аргумент может быть целым или вещественным числом, символом или строкой, второй аргумент, соответственно, списком целых или вещественных чисел, или списком, элементами которого являются символы, или списком, состоящим из строк. При этом процедура, реализующая этот предикат в разделе описания предложений, будет единственной.

Кроме того, при описании предиката можно указать, будет он детерминированным или недетерминированным. Детерминированный предикат возвращает только одно решение, а недетерминированный предикат при помощи поиска с возвратом может давать много решений. Детерминированные предикаты менее требовательны к оперативной памяти и выполняются быстрее.

Для того чтобы указать, что предикат является детерминированным (недетерминированным), нужно перед его именем поместить зарезервированное слово *determ (nondeterm)*. Если ни *determ,* ни *nondeterm* при описании предиката не использовались, то, по умолчанию, предикат считается детерминированным.

В Турбо Прологе имеется директива компилятора *check_determ,* которая принудительно включает проверку предикатов на детерминированность.

### 2.3.6 Раздел описания предложений

В данный раздел заносятся факты и правила. О содержимом этого раздела можно говорить как о данных, необходимых для работы программы. Этот раздел можно считать основным разделом программы. Все предикаты, которые применяются в этом разделе и не являются стандартными предикатами, должны быть описаны в разделе описания предикатов или в разделе описания предикатов базы данных. Начинается этот раздел со служебного слова CLAUSES.

Предложения, у которых в заголовке указан один и тот же предикат, должны идти друг за другом. Такой набор предложений называется процедурой. Программу на Прологе принято оформлять по следующим правилам:

- между процедурами пропускается пустая строка;
- тело правила записывается со следующей строки, после строки, в которой был заголовок, с отступом;
- каждую подцель записывают на отдельной строке, одну под другой.

Эти правила не являются обязательными, но они делают программу более «читабельной».

### 2.3.7 Раздел описания внутренней цели

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. Если этот раздел отсутствует, то после запуска программы Пролог-система выдает приглашение вводить вопросы в диалоговом режиме (внешняя цель). При выполнении внешней цели Пролог-система ищет все решения, выводя все возможные значения для переменных, участвующих в вопросе. Если же выполняется внутренняя цель, то осуществляется поиск только первого решения, а для получения всех решений нужно предпринимать дополнительные действия.

Программа, компилируемая в исполняемый файл, который можно запускать независимо от среды разработки, обязательно должна иметь внутреннюю цель. Внешнюю цель обычно применяют на этапе отладки программы.

Далее рассмотрим некоторые наиболее употребляемые стандартные предикаты для ввода и вывода.

### 2.3.8 Предикаты ввода-вывода

Турбо Пролог имеет отдельные предикаты для чтения с клавиатуры или из файла данных целого, вещественного, символьного и строкового типа. Рассмотрим чтение из стандартного устройства ввода информации (клавиатуры) и, соответственно, запись на стандартное устройство вывода информации (монитор).

Предикат *readln* считывает строку с текущего устройства ввода и связывает ее со своим единственным выходным параметром.

Предикат *readint* читает с текущего устройства целое число и связывает его со своим единственным выходным параметром.

Предикат *readreal* отличается от предиката *readint* тем, что он считывает не целое, а вещественное число.

Для чтения символа с текущего устройства ввода используется предикат *readchar.* Есть еще предикат *inkey,* который так же, как и *readchar,* читает символ со стандартного устройства ввода. Разница между ними в том, что предикат *readchar* приостанавливает работу программы до тех пор, пока не будет введен символ, а предикат *inkey* не прерывает выполнение программы. Если нужно просто проверить, нажата ли клавиша, можно воспользоваться предикатом *keypressed,* не имеющим аргументов.

Предикат *readterm* предназначен для чтения сложных термов. У него два параметра: первый входной указывает имя домена, второй параметр конкретизируется термом домена, записанного в первом параметре. Если считанная этим предикатом строка не соответствует домену, указанному в его первом парамет-

ре, предикат выдаст сообщение об ошибке.

Для записи данных в текущее устройство записи служит предикат *write*. Он может иметь произвольное количество параметров. Кроме того, в Турбо Прологе есть еще и предикат *writef,* который служит для форматного вывода данных.

Для осуществления перехода на следующую строку (возврат каретки и перевод строки) применяется предикат *nl,* не имеющий параметров.

Описанная ниже группа предикатов служит для преобразования типов.

Предикат *upper_lower* имеет два аргумента и три варианта использования. Если в качестве первого аргумента указана строка (или символ), а второй аргумент свободен, то второй аргумент будет означен строкой (символом), полученной из первого аргумента преобразованием к нижнему регистру. Если в исходной строке были прописные английские буквы, то они будут заменены строчными. Если же, наоборот, первый аргумент свободен, а второй аргумент — это строка (или символ), то первый аргумент получит значение, равное строке (символу), полученной из второго аргумента преобразованием к верхнему регистру. Если в строке, находящейся во втором аргументе, были строчные английские буквы, то они будут заменены прописными. И, наконец, имеется третий вариант использования. Если и первый, и второй аргументы связаны, то предикат будет истинным только в том случае, если во втором аргументе находится строка (символ), которая получается из строки, находящейся в первом аргументе, путем замены всех прописных английских букв на строчные. В противном случае предикат будет ложным.

Также имеют два параметра и три варианта использования предикаты *str_int, str_real.* Первый преобразует строку в целое число и наоборот. Второй служит для превращения строки в вещественное число или вещественного числа в строку.

Предикат *st_shar* имеет те же параметры использования и применяется для преобразования односимвольной строки в один символ и наоборот.

Немного по-другому работает предикат *char_int.* Он позволяет переходить от символа к его ASCII-коду и обратно.

Хотя Пролог — не самый лучший инструмент для выполнения большого объема вычислений, в нем имеются стандартные средства для реализации обычных вычислений. При этом можно использовать четыре арифметических операции (сложение (+), вычитание (-), умножение (*) и деление (/)), а также целочисленное деление (div) и взятие остатка от деления одного целого числа на другое *(mod).* Для сравнения чисел можно воспользоваться операциями рав-

но (=), не равно (<>), больше (>), больше или равно (>=), меньше (<), меньше или равно (<=).

Кроме того, можно использовать обычные математические функции, такие как: логарифмы натуральный *(1n)* и десятичный (log), квадратный корень *(sqrt),* модуль (abs), экспонента *(exp).* Тригонометрические функции: синус (sin), косинус (cos), тангенс *(tan),* арктангенс *(arctan).* Величины углов указываются в радианах.

Функция *trunk* отбрасывает дробную часть своего параметра, а функция *round* округляет вещественное число до ближайшего целого.

Для вычисления псевдослучайных чисел имеется два варианта предиката *random.* Первый вариант имеет один выходной параметр, в который помещается сгенерированное вещественное число, лежащее в промежутке межу нулем и единицей. Второй вариант этого предиката — двух-аргументный. В качестве первого входного аргумента указывается целое число. Второй аргумент означивается целым числом, лежащим между нулем и первым аргументом.

Нуль-местный предикат *true* всегда истинен, а нуль-местный предикат *fail* — всегда ложен. Предикат *fail* часто используется для организации поиска с возвратом. Причем размещение какой-либо подцели в теле правила после предиката *fail* бессмысленно, поскольку в связи с тем, что этот предикат всегда терпит неудачу, цель никогда не будет достигнута.

Одноместный предикат *free* истинен, если его аргументом является свободная переменная, и ложен в противном случае. Предикат *bound,* наоборот, истинен, если его аргумент — это связанная переменная, и ложен, если его аргумент свободен.

В Турбо Прологе любой текст, находящийся между символами /* и */, рассматривается как комментарий. Кроме того, любой текст между символом % и концом строки также воспринимается как комментарий. Комментарий отличается от остального текста тем, что он игнорируется компилятором Турбо Пролога. Соответственно, комментарии пишутся не для компилятора, а для человека, для того, чтобы сделать программу более легкой для понимания.

## 3 Задания к работе и порядок ее выполнения

3.1 Создать свою директорию - D/student/группа/фамилия.

3.2 **Запуск Visual Prolog 2.0** производится из главного меню:

**Пуск - Программы –Visual Prolog Personal Edicion – Vip 32.**

3.3 **Открытие окна Редактора Visual Prolog.**

Чтобы создавать новое окно редактирования, Вы можете использовать команду меню **File - Nev**. Появится новое окно редактор с заголовком "NONAME".

Чтобы проверить, что ваша система установлена правильно, в окне редактора наберите следующий текст:

**GOAL write("Hello world"),nl.**

Это - то, что называется ЦЕЛЬЮ в терминологии Пролога, и этого достаточно, чтобы быть программой, которая может быть выполнена.

Чтобы выполнять ЦЕЛЬ (GOAL), Вы должны или активизировать пункт меню **Project – Test GOAL,** или пиктограмму **G,** или нажать клавиши **Ctrl + G**. Если ваша система установлена правильно, на экране появится

**Hello world**

**Yes**

Результат выполнения будет находиться в отдельном окне, которое Вы должны закрыть прежде, чем Вы будете проверять другие ЦЕЛИ.

3.4 Анализ программы на Visual Prolog.

1) Загрузка примеров с диска.

Открыть файл (команда **File – Open)** Vip/Doc/Examples/CH02E01.PRO (см. приложение А).

2) Найти в программе переменные, константы, структуры, предложения, комментарии.

3) Определить тип данных.

4) Указать факты, правила, цели.

5) Составить структуру программы

6) Запустить файл CH02E01.PRO на выполнение:

Команда **Project – Test Goal.**

7) Составить алгоритм выполнения программы.

8) Сделать запрос программе, заменив предложение цели

**likes(bill, tennis).**

9) Перевести программу CH02E01.PRO на русский язык, оставив на английском языке лишь описание разделов и типов.

**Замечание**: Для переключение на кириллицу необходимо активизировать пиктограмма (Font) **F** – диалоговое окно **Шрифт – Набор символов – Кириллица**

10) В предложении цели заменить второй параметр, перейдя от атома к переменной. Проанализируйте, что измениться в программе.

3.5 Составить на русском языке свою программу на Visual Prolog, содержащую набор правил, фактов и целей.

3.6Анализ программы, содержащей арифметические вычисления.

1) Открыть файл (команда **File – Open)** Vip/Doc/Examples/CH03E01.PRO (см. приложение Б).

2) Проанализировать тип данных.

3) Перевести на русский язык.

4) Составить цель из нескольких предикатов.

5) Составить свою программу, содержащую арифметические операции.

3.7 Составить отчет по проделанной работе, в которой должны войти следующие разделы:

- название и цель работы;
- краткие теоретические сведения;
- программы и результат их выполнения для заданий.

## 4 Вопросы для самоконтроля

4.1 Введите понятия:

- объекты данных;
- атом, число, переменная, структура, терм;
- заголовок, тело, предикат;
- функтор, арность, главный функтор терма;
- утверждение, факт, правило, цель, предложение;
- унификация, конкретизация, наиболее общая конкретизация.

4.2 Синтаксис чисел, атомов, переменных, структур, предложений, фактов, правил, запросов, комментариев.

4.3 Структура программы на Visual Prolog.

4.4 Форма записи арифметических операций.

# Список использованных источников

1. **Шрайнер, П.А.** Основы программирования на языке Пролог [Текст]: курс лекций: учебное пособие для вузов / П.А. Шрайнер.- М.: Интернет-Ун-т Информ. Технологий, 2005. – 176 с.

2. **Братко, И.** Программирование на языке Пролог для искусственного интеллекта [Текст]: пер. с англ./ И Братко. - М.: Мир, 1990.-560 с.

3. **Доорс, ДЖ.** Пролог – язык программирования будущего [Текст]: пер. с англ./ Дж. Доорс, А.Р. Рейнблейн, С. Вадера. - М.: Финансы и статистика, 1990. – 144 с.

4. **Малпас, Дж.** Реляционный язык Пролог и его применение [Текст]: пер. с англ./ Дж. Малпас, под редакцией В.Н. Соболева. – М.: Наука. Гл. ред. физ.-мат. лит., 1990. – 464с.

5. **Янсон, А.** Турбо-Пролог в сжатом изложении [Текст]: пер. с нем./ А. Янсон. - М.: Мир, 1991. – 94 с.

# Приложение А
(обязательное)

```
/*********************************************

        Copyright (c) 1984 - 1998 Prolog Development Center A/S

 Project:
 FileName: CH02E01.PRO
 Purpose:
 Written by: PDC
 Modifyed by: Eugene Akimov
 Comments:
*********************************************/

PREDICATES
  nondeterm likes(symbol,symbol)

CLAUSES
  likes(ellen,tennis).
  likes(john,football).
  likes(tom,baseball).
  likes(eric,swimming).
  likes(mark,tennis).
  likes(bill,Activity):-
 likes(tom, Activity).

 GOAL
  likes(bill, baseball).
```

# Приложение Б
(обязательное)

```
/*********************************************

        Copyright (c) 1984 - 1998 Prolog Development Center A/S

 Project:
 FileName: CH03E01.PRO
 Purpose:
 Written by: PDC
 Modifyed by: Eugene Akimov
 Comments:
*********************************************/

DOMAINS
  product,sum = integer

PREDICATES
  add_em_up(sum,sum,sum)
  multiply_em(product,product,product)

CLAUSES
  add_em_up(X,Y,Sum):-
Sum=X+Y.
  multiply_em(X,Y,Product):-
Product=X*Y.

 GOAL
  add_em_up(32,54,Sum).
```