АЛГОРИТМИЧЕСКИЙ ЯЗЫК "ПРОЛОГ"

ВВЕДЕНИЕ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

НАЗНАЧЕИЕ ЛАБОРАТОРНЫХ РАБОТ

Лабораторные работы предназначены для закрепления на практике теоретических знаний, полученных на лекциях по дисциплинам "Базы данных и экспертные системы" и "Информационные системы".

Полученные практические навыки используются студентами в дальнейших дисциплинах и дипломном проектировании.

СОДЕРЖАНИЕ И ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Перед началом выполнения лабораторной работы необходимо ознакомиться с ее описанием и получить задание. Затем:

1. Выполнить проектирование прикладной программы и базы данных в соответствии с полученным заданием.
2. В инструментальной среде алгоритмического языка "Пролог" создать файл, содержащий текст требуемой программы.
3. Отладить программу.
4. Выполнить типовые запросы к программе.
5. Оформить отчет и защитить лабораторную работу.

Отчет должен содержать:

- название лабораторной работы;
- цель работы;
- краткий конспект теоретической части;
- текст программы;
- результаты выполнения типовых запросов;
- выводы.

ОСНОВНЫЕ СВЕДЕНИЯ О СРЕДЕ TURBO - PROLOG .

СЛОВАРЬ СПЕЦИАЛЬНЫХ ТЕРМИНОВ

- -Автономные программы. - программы, которые должны быть запущены из ДОС независимо от Turbo Prolog.
- -Анонимная переменная. - переменная, обозначаемая именем " -", используемая вместо обычной переменной в том случае, когда значение обычной переменой, с которой она связана, не важно.
- -Аргументы. - обобщенное название всех имен переменных и объектов в отношении.
- -Атом. - отношение, содержащее, возможно, объекты или переменные.
- -Атрибут. - целое положительное число, отражающее характеристики отображения в данном окне, включая цвет, мерцание, инверсное изображение.

- -Внешняя цель. цель, которая вносится в диалоговое окно пользователем и задается для программы, находящейся в настоящее время в рабочем файле.
- -Внутренняя цель - цель, содержащаяся в секции целей программы.
- -Вызов подцели (либо предиката) - выражение, означающее, что Turbo Prolog в настоящее время пытается согласовать конкретную подцель (принадлежащую данному предикату).
- -Глобальный - спецификатор, используемый для разрешения доступа более чем одной программе к конкретным доменам и предикатам.
- -Действительное число (real) - десятичное число в диапазоне от +/- 1.0 e ^ - 308 до +/- 1.0 e ^ +308.
- -Дерево целей - диаграммное представление возможных альтернатив, которые могут быть выполнены при оценке подцелей, входящих в состав целей.
- -Директивы компилятора - инструкции для компилятора Turbo Prolog для выполнения специальных действий.
- -Домен - определяет типы значимых объектов, которые могут вступать в отношение.
- -Заголовок списка - первый элемент списка.
- -Имя - любая жесткая последовательность букв, цифр и символов " -", которая начинается с буквы нижнего регистра клавиатуры, либо символа " -".
- -Имя файла - символьное имя файла, начинающееся с буквы нижнего регистра клавиатуры и появляющееся в правой части описания файлового домена, либо одно из предварительно определенных символьных файловых имен принтера, экрана, клавиатуры и устройства связи.
- -Инфиксная запись - запись арифметического выражения с операторами между двумя значениями или выражениями, по которым они должны действовать.
- -Итеративный метод - метод, представляющий собой повторение одних и тех же основных действий до тех пор пока цель не будет достигнута (согласована).
- -Конец списка - список, который сохраняется при удалении первого элемента данного списка (вместе с его разделителем - запятой).
- -Лексема - имя, действительное или целое число, либо непустой символ.
- -Множественные описания предиката - отдельно взятый предикат может иметь несколько описаний, каждое из которых содержит различные доменовые спецификации для аргумента (аргументов) релевантного отношения.
- -Модуль - программа Turbo Prolog с глобальными описаниями, формирующими часть проекта.
- -Неуспех - подцель Turbo Prolog не может быть согласована (достигнута).

- -Объект - имя отдельного элемента конкретного типа.
- -Ограничение конца рекурсии - действие, выполняемое в системе Turbo Prolog для ограничения рекурсии по времени и месту в правилах.
- -Отношение - способ связи, при котором подобранные объекты (либо выполнение ссылок и переменных на объекты) соответствовали бы друг другу.
- -Отсечение (ИЛИ !) - отсечение позволяет Turbo Prolog использовать все возможности выборов в оценке предиката, содержащего отсечение. Отсечение может рассматриваться, как подцель, которую Turbo Prolog не может пройти при переборе с возвратами.
- -Параметр - общее имя для имен объектов и переменной в отношении.
- -Переменная - имя, начинающееся с прописной буквы для представления (возможно, неизвестного) значения конкретного объекта.
- -Подобъект - один из объектов в сложном объекте.
- -Подцель - отношение, возможно включающее объекты или переменные, которое должен согласовать Turbo Prolog.
- -Поиск с возвратом - встроенный в Turbo Prolog механизм, обеспечивающий после завершения оценки заданной подцели возвращение Turbo Prolog к предыдущей подцели с тем, чтобы попытаться согласовать ее другим путем.
- -Поле . - определенная последовательность отображаемых символьных позиций, появляющихся в одной и той же строке на экране дисплея.
- -Поточный вариант - если предикат ассоциируется с несколькими различными шаблонами, то для каждого шаблона будет реализована самостоятельная внутренняя программа, относящаяся к данному предикату. Эти различные действия называются поточными вариантами предиката.
- -Правило - связь между "событием" и списком подцелей, которые должны быть согласованы для того, чтобы "событие" было исполнено.
- -Предикаты базы данных - предикаты, для которых факты могут добавляться, либо исключаться из системы Turbo Prolog во время исполнения.
- -Предложение (утверждение) - событие или правило для определения предиката, завершающееся точкой.
- -Принцип поиска - одно из четырех основных правил, которые должен соблюдать Turbo Prolog с тем, чтобы достичь цели.
- -Приоритет оператора - иерархия, определяющая порядок, в котором выполняются операторы в арифметическом выражении.
- -Проект - программа Turbo Prolog, содержащая один или более модулей.
- -Рекурсия - способ, при котором объект определяется в собственных терминах.
- -Свободная переменная - переменная, не связанная в настоящее время ни с одним значением.

- -Связанная переменная - переменная, имеющая ссылку к известной величине.
- -Связывание переменных - состояние (свободное или связанное) одной или более переменных.
- -Символ (char) - произвольный символ, заключенный в одинарные кавычки.
- -Система типов - средство, с помощью которого все объекты в отношении или все переменные, используемые как аргументы в отношении, ограничиваются принадлежностью к конкретным доменам, использующимся в описании соответствующего предиката.
- -Сложная цель - цель, состоящая не менее чем из двух подцелей.
- -Сложный объект - объект, состоящий из функтора и списка объектов, разделенных запятыми и заключенных в скобки.
- -Согласование подцелей - процесс, при котором Turbo Prolog выбирает значения для любых несвязанных переменных (если это возможно) таким образом, чтобы подцель являлась истинной в соответствии с заданными утверждениями для передающего предиката.
- -Список - специальный вид объекта, содержащий набор элементов, заключенных в квадратные скобки и отделенных запятыми.
- -Строка - произвольная строка символов, заключенных в двойные кавычки.
- -Текущее устройство ввода - установленное в данным момент времени устройство ввода, из которого стандартные предикаты по умолчанию берут входные данные.
- -Текущее устройство вывода - установленное в данный момент времени устройство вывода, в которое стандартные предикаты по умолчанию посылают выходные данные.
- -Терм - любой объект одного из доменов стандартного типа, список, переменная, либо сложный терм, т.е. функтор, сопровождающийся списком термов, заключенных в круглые скобки и разделенных запятыми.
- -Трассировка - осуществление пошагового отчета о выполнении программы, показывающее все происходящие изменения.
- -Указатель - механизм, с помощью которого Turbo Prolog сохраняет запись следующего места в своей базе данных событий или правил, к которой он обращается при переборе с возвратом.
- -Унификация - процесс, при котором Turbo Prolog пытается сопоставить подцели с событиями и левой частью правил, либо для достижения цели, либо для определения одной или более подцелей, необходимых для оценки первоначальных целей.
- -Факт - отношение между объектами.
- -Функтор - имя для сложного объекта.
- -Целое число - целое число в диапазоне от -32768 до +32767.

- -Цель - набор подцелей, которые должен согласовать (достичь) Turbo Prolog.
- -Элемент списка - может быть как объектом, так и другим списком.
- -Экспертная система - компьютерная система, которая может давать экспертную оценку в конкретной (очень узкой) области.

Главное меню содержит следующие опции:

Files - Управление файлами (загрузка, сохранение, создание и т.д.), манипуляция каталогами (вывод, измене ние), вызов DOS, выход из системы.

Edit - Создание и редактирование исходных файлов с помощью строенного текстового редактора.

Run - Автоматическая компиляция и запуск программы из среды разработки.

Options - Установка опций компилятора (таких, как контроль переполнения, информация для отладки, размеры памяти) и компоновщика; обеспечение возможности редактирования файла описания проекта (файла с расширением .prj).

Setup - Установка цветов и размеров окон в Turbo Prolog; установка каталога для файлов исходного текста, объектных, исполняемых и прочих; установка конфигурации клавиатуры и текст строки помощи; со хранение в .sys - файле опций компилятора; за грузка существующего .sys - файла. Выбирать элементы меню можно несколькими способами:

1. Выделенная буква. Для выбора элемента непосредственно из еню нажать выделенную цветом в названии данного элемента букву ("F" для "File", "L" для "Linker Options" и т.д.).
2. Выделенная полоса. С помощью клавиш перемещения курсора передвинуть выделенную цветом полосу на требуемый элемент и нажать клавишу "Enter".
3. "Горячая клавиша". Для элементов главного меню можно воспользоваться "горячей клавишей": нажать клавишу "Alt" и первую букву выбранного элемента главного меню. Например, Alt-O переводит систему в режим Options из любого состояния и т.д.

Для выхода из меню и возврата в предыдущее положение необходимо нажать клавишу "Esc". Из среды редактирования в главное меню можно перейти нажатием Ctrl K D или F10.

Наиболее часто используемые элементы меню Turbo Prolog "связаны" с "горячими клавишами", нажав на которые можно вызвать соответствующий элемент меню , исключив необходимость прохождения через всю систему меню. Список "горячих клавиш" можно получить в любой момент времени, нажав Alt-R .

"Гор. клавиша" И Эквивалент меню

Выход из Пролога	Alt-X	F/Quit
Временный выход в DOS	Alt-D	F/Os Shell
Вывести "гор. клавиши"	Alt-H	S/M/K/Hot Keys
Вывести версию Пролога	Alt-F10	-
Загрузить файл	F3	F/Load
Указать файл	Фде-А3	А.Зшсл
Сохранить файл	F2	F/Save
Компилировать в память	F9	C/Memory
Компилировать в.OBJ	Shift-F9	C/OBJ File
Компилировать в .EXE	Ctrl-F9	C/EXE File
Компилировать проект	Alt-F9	C/Project

СИСТЕМНЫЕ ОКНА И СТРОКИ ПОМОЩИ

Окно редактирования

Если предварительно в редактор не был загружен какой-либо файл, то Turbo Prolog открывает окно редактирования, создавая при этом новый (пустой) текстовый файл с именем WORK.PRO.

Окно редактирования состоит из непосредственно самого окна, в котором можно наблюдать и корректировать строки исходного текста, и статусной строки, предоставляющей пользователю информацию о редактируемом тексте.

Статусная строка содержит полное имя исходного файла (устройство, каталоги, имя файла и расширение), координаты расположения курсора и текущие режимы редактирования.

Строка помощи в последней строке экрана перечисляет некоторые доступные в редакторе команды и "горячие" клавиши.

Существующий файл можно загрузить двумя способами:

1. Из меню : Files/Load (Файлы/Загрузка).
2. С помощью "горячей" клавиши F3.

Для сохранения исходного текста можно практически в любой момент нажать клавишу F2. Из главного меню это же можно сделать, выбрав пункт Files/Save .

Окно трассировки

В окне трассировки (Trace) можно следить за тем, как Turbo Prolog выполняет программу. Для этого необходимо откомпилировать программу с включенной опцией трассировки. Включить данную опцию можно либо из меню Options/Compiler Directives, либо, введя в исходный текст директивы trace или shorttrace. Можно также (нажав Alt-P) задать Прологу дублирование данных, вводимых в окно трассировки, на принтере или в файле. (Данная "горячая" клавиша активизирует специальное меню регистрации).

Окно сообщений

В процессе работы Turbo Prolog выводит различные сообщения в окно под названием Messages (Сообщения). Например, Turbo Prolog может:

- сообщить о чтении или сохранении файла;
- выводить имена предикатов компилируемой в данный момент программы;
- при обращении пользователя к стандартному или глобальному предикату с недопустимым шаблоном потока - отображать вариант потока и пр.

Диалоговое окно

Диалоговое окно является средством среды разработки, с помощью которого в систему Turbo Prolog передаются запросы пользователя. Для перехода в это окно достаточно нажать Alt-R .

Если в текущий момент в редакторе имеется исходный текст программы, причем в этой программе нет секции "Goal", то вначале Turbo Prolog скомпилирует ее (что необходимо для добавления в систему предикатов программы), а затем активизирует диалоговое окно. Если же в редакторе нет исходного текста программы, то Turbo Prolog активизирует диалоговое окно без какой-либо компиляции.

Затем можно вводить по одному вопросу (цели) за один раз. Клавиша F8 используется для вызова последней из ранее введенных целей.

Помощь при ошибках

Если в процессе компиляции система Turbo Prolog обнаружит ошибку, то на экран будет выведен ее номер и соответствующее сообщение. Для многих из этих сообщений в системе имеется дополнительная информация, объясняющая, чем могла быть вызвана Даная ошибка.

Строка помощи указывает, что для получения разъяснений можно нажать клавишу F1, а для продолжения компиляции - F10.

АЛГОРИТМИЧЕСКИЙ ЯЗЫК "ПРОЛОГ"

ЛАБОРАТОРНАЯ РАБОТА № 1

"ПРОЕКТИРОВАНИЕ И СОЗДАНИЕ БАЗЫ ДАННЫХ НА ЯЗЫКЕ "ПРОЛОГ"

Цель работы - приобретение практических навыков проектирования прикладной базы данных, ее создание и визуализация на языке "Пролог".

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

ОСНОВЫ ПРОГРАММИРОАНИЯ НА ПРОЛОГЕ.

Все Пролог-программы записываются с использованием литер, которые делятся на четыре категории:

A B C D E ...
a b c d e ...
0 1 2 3 ... 9
+ - * / : . ( ) ...

Т. е алфавит языка состоит из прописных и строчных букв латинского алфавита, цифр и специальных знаков.

Кроме того, в данные строкового типа могут входить прописные и строчные буквы русского алфавита.

СТРУКТУРА ПРОЛОГ-ПРОГРАММЫ

Пролог-программа состоит из термов. Терм - это либо константа, либо переменная, либо структура. Терм записывается, как последовательность литер, входящих в алфавит языка.

В целом Пролог-программа состоит из ряда секций, которые определяются с помощью ключевого слова и следуют в строго определенном порядке. Их описание приведено в таблице:

Название секции	Пояснение
domains	0 или более описаний доменов
global domains	0 или более описаний доменов
database	0 или более описаний предикатов базы данных
predicates	0 или более описаний предикатов
global predicates	0 или более описаний предикатов
goal	0 или 1 цель
clauses	или более утверждений (фактов и правил)

Нет необходимости включать все секции в Пролог-программу.

Например, можно опустить секцию "goal" - "цель" (или database).

С другой стороны, программа может состоять лишь из одной секции цели: goal readint(X), Y=X+3, write(" X+3=",Y).

Однако, как правило, требуется, по крайней мере, наличие предикатов и утверждений. Для больших программ использование секций доменов позволяет контролировать типы переменных и констант.

Секции глобальных переменных и доменов используются для реализации модульного программирования на Прологе. Их описания действуют глобально для ряда модулей.

При составлении Пролог-програм следует учитывать ряд ограничений:

1. секции следуют в строго установленном порядке, и каждой секции предшествует собственное ключевое слово;
1. в программе может быть максимум одна цель;
2. все утверждения, описываемые одним и тем же предикатом, должны группироваться (следовать одно за другим).

ИМЕНА В ПРОЛОГЕ

Имена служат для обозначения термов в Пролог-программе (символьных констант, доменов, переменных, предикатов). Имя состоит из букв, цифр и символа подчеркивания " -". Необходимо помнить:

1. имена символьных констант начинаются с маленькой (строчной) буквы;
2. имена переменных начинаются с большой (прописной) буквы или с символа " -";
3. одиночный символ " -" означает анонимную переменную;
4. максимальная длина имени - 256 символов;
5. существуют зарезервированные имена, которые нельзя использовать в прикладных целях:

and	clauses	findall	if	predicates
asserta	database	free	include	readterm
assertz	domains	global	not	retract
bound		fail	goal	or

6. существуют ключевые слова, которые имеют в Прологе специальный смысл и их нежелательно использовать в других целях: beep, bios, clearwindow, cos, cursor, display, exp, eof, in и другие.

СТАНДАРТНЫЕ ТИПЫ ДОМЕНОВ

Аналогично другим языкам программирования (например, Паскалю) Пролог позволяет описывать и манипулировать сложными структурами данных, образованными (в конечном счете) из простых стандартных типов данных.

Turbo Prolog обеспечивает следующие стандартные типы данных (домены):

1. integer;
2. real;
3. char;
4. symbol;
5. string;
6. file.

При этом описание доменов имеет вид:
name = d или name1,name2,...nameN = d

Подобное описание указывает, что объект с указанным именем может принимать значения, не выходящие из множества значений домена соответствующего типа.

Рассмотрим подробнее стандартные домены.

Тип	Примечание	Примеры
integer	Целые числа от -32768 до 32767	1, 2, 3, 45, -90
real	Вещественные числа в диапазоне +/- e ^ -307 до +/- e ^ +308	0.23, -7.458
char	Символ, заключенный в '	'a' , 'F'
string	Любая последовательность символов, заключенная между двумя кавычками (двойными)	"Моя первая программа"
symbol	Имеет два формата: 1. последовательность букв, цифр и " -", начинающаяся со строчной буквы; 2. последовательность символов, заключенных в кавычки "..." С точки зрения пользователя symbol и string идентичны. Отличие - в их внутреннем .	 phone -number "Моя программа"

Следует особо обратить внимание на объекты, принадлежащие типам char или string и содержащие символ "\". Они имеют специальное значение:

\Number - символ со значением кода (Number) ASCII;
\n - символ перехода на новую строку;
\t - символ табуляции.

Пример:

```
write ('\13').
write ('\n').
```

Системные ограничения на простые типы доменов:

1. константа типа string может состоять максимум из 250
1. символов.
2. переменная типа string может иметь в качестве значения строку максимум из 64К символов.
3. максимальное количество имен доменов - 250.

НАЧАЛА ПРОГРАММИРОВАНИЯ В СИСТЕМЕ ПРОЛОГ

СТРУКТУРА ПРОЛОГ-ПРОГРАММЫ

```
/* Программа 1 */
domains
  person, activity - symbol
```

```
predicates
  likes ( person^ activity )
clauses
  likes ( ellen, tennis ).
  likes ( john, football ).
  likes ( john, swimming ).
%  ....................................
  likesbt ( bill, X ) if likes ( tom, X ).
```

Итак, мы видим, что простая программа на Прологе состоит из трех секций.

Секция предложений ("клозов") содержит набор фактов и правил и описывает базу данных некоторой предметной области (ПО) (в нашем случае - людей и любимых ими видов спорта). То есть, рассматривая соответствующие предложения, мы можем установить, что

Элен любит играть в теннис.

Джон любит футбол и плавание и т.д.

Отметим, что в данном случае нет информации типа

likes (bill, football),

Но отсутствие этого факта в базе данных не означает, что этот факт отсутствует в действительности. Модель ПО рассматривается всегда, как некий "срез" реального мира без обсуждения (в лабораторных работах) ее соответствия действительности.

При запуске системы в окне диалога появится сообщение: Goal: - , в ответ на что пользователь может задавать различные вопросы в системе:

likes (ellen, tennis)

True - ответ системы;

likes (ellen, swimming)

No solutions - ответ системы (Нет решения);

Обычно перечисление всех фактов утомительно, поэтому, как правило, в ряде случаев используют переменные.

ПЕРЕМЕННЫЕ

Переменные служат для обозначения неизвестных заранее (изменяющихся) объектов

Goal: likes (john, X)

X = football

X = swimming

2 solutions

Goal: -

Имена переменных в Прологе должны начинаться с прописной буквы. Важна только первая буква, далее имя может состоять из любой последовательности букв, цифр и символов " -":

My -first -correct -name,

S -10 -15 -93.

Неверные имена:

1sAB -S,

name,

'name'.

Целесообразно выбирать имена переменных таким образом, чтобы они имели семантический смысл и удобно читались.

ОБЪЕКТЫ И ОТНОШЕНИЯ

Каждое предложение в Прологе состоит из отношения, которое связывает один или более объектов, например:

likes (tom, baseball), где

likes - отношение,

tom - объект,

baseball - объект.

Пользователь имеет полную свободу выбора имен отношений и объектов, но должен помнить:

1. Имя объекта должно начинаться со строчной буквы, за которой следует любая последовательность букв, цифр и " -".
2. Имя отношения может содержать любое количество букв, цифр и " -".

Следует отметить важность порядка следования объектов в отношении: "Том любит бейсбол", но не "бейсбол любит тома".

ПРИМЕР БАЗЫ ДАННЫХ НА ЯЗЫКЕ ПРОЛОГ

```
/* Пример базы данных для автомобилей
     brand   - марка автомобиля;
     mileage - пробег;
     age     - срок эксплуатации;
     color   - цвет;
     price   - цена.
*/
  domains
    brand, color = symbol
    age, price = integer
    mileage = real
  predicates
    car(brand, mileage, age, color, price)
    truck(brand, mileage, age, color, price)
  clauses
    car(chrysler, 130000, 3, red, 12000).
    car(ford, 90000, 4, grey, 25000).
    car(datsun, 8000, 1, red, 30000).
    truck(ford, 80000, 6, blue, 8000).
    truck(datsun, 50000, 5, orange, 20000).
    truck(toyota, 25000, 2, black, 25000).
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислить основные секции Пролог-программы, правила их построения и размещения.
2. Пояснить принципы формирования имен в Пролог-программах.
3. Объяснить назначение стандартных доменов.
4. Пояснить назначение переменных в Пролог-програмах.
5. Перечислить правила формирования отношений.

АЛГОРИТМИЧЕСКИЙ ЯЗЫК "ПРОЛОГ"

ЛАБОРАТОРНАЯ РАБОТА № 2

"ИСПОЛЬЗОВАНИЕ ПРАВИЛ ПРИ ПРОЕКТИРОВАНИИ БАЗЫ данных НА ЯЗЫКЕ ПРОЛОГ"

Цель работы - приобретение практических навыков использования правил при проектировании базы знаний.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Правила позволяют выводить из одних фактов выводить другие. Иначе, правило - это заключение, для которого известно, что оно истинно, если одно или несколько других ранее введенных или найденных заключений или фактов являются истинными. Ниже представлены правила, соответствующие связи "likes".

Cindy likes everything that Bill likes. ("Синди любит все, что любит Билл").
Caitlin likes everything that is green. ("Кейтлин любит все зеленое").

Для задания этих правил в Прологе Нужно лишь незначительно изменить синтаксис:

```
likes("Синди",Something) if likes("Билл", Something).
likes("Кейтлин",Something) if green(Something).
```

Кроме того, правило можно интерпретировать, как процедуру. Другими словами, правила могут означать следующее:

"Для того, чтобы доказать, что Синди любит что-либо, нужно доказать, что то же самое любит Билл".

"Для того, чтобы доказать, что Кейтлин любит что-либо, нужно доказать, что это - зеленого цвета".

С такой "процедурой" точки зрения правила выполняют действия, не являющиеся доказательствами, например, вывод на экран.

Правила соответствуют зависимым связям: они позволяют Прологу выводить один "элемент" информации из другого. Правило становится истинным, если можно доказать, что истинно заданное множество условий.

Каждое правило зависит от доказательства истинности соответствующих условий.

Все правила в Прологе имеют три части:

- заголовок;
- связка if;
- тело.

Заголовок - это факт, который должен быть истинным, если истинно некоторое количество условий. Для обозначения заголовка правила могут также использоваться термины "заключение" и "зависимая связь".

Связка "if" отделяет заголовок правила от его тела. Он может заменяться совокупностью символов ":-".

Тело правила - это множество условий (или список фактов), которые должны быть истинны для того, чтобы Пролог мог доказать истинность заголовка этого правила. Тело правила состоит из одной или более подцелей. Подцели разделяются запятыми (или словами "and"), а последняя подцель правила завершается точкой.

Каждая подцель выполняет вызов соответствующего предиката Пролога. После осуществления этого вызова, проверяется истинность данного предиката, и, если результат утвердительный,- работа продолжается, но уже для следующей подцели. Если же в процессе работы достигается точка, все правило считается истинным.

Для успешного разрешения правила Пролог должен разрешить все его подцели и создать последовательный список переменных, "связав" их должным образом. Если же хотя бы одна из подцелей ложна, Пролог возвращается назад для поиска альтернативы предыдущей подцели, а затем вновь перемещается вперед, но уже с другими значениями переменных. Этот процесс называется "поиск с возвратом".

Определив для Пролога множество фактов, можно (в соответствии с этими фактами) задавать вопросы. Это называется "запросами" к системе Пролога. Пролог всегда ищет решение, начиная с первого факта, и просматривает их до самого последнего.

Если на основании всех известных фактов не находится такого сочетания значений переменных, когда все подцели истинны,- использование правил не приводит к успешному завершению процедуры и Пролог-программа формирует отрицательный ответ на запрос.

Рассмотрим следующий пример программы:

```
predicates
  can -buy (symbol, symbol)      /* может купить */
  person (symbol)                /* человек      */
  car (symbol)                   /* машина       */
  likes (symbol, symbol)         /* нравится     */
  for -sale (symbol)             /* продается    */
clauses
```

```
person (kelly).
person (peter).
car (lemon).
car (pontiac).
likes (kelly, pontiac).
likes (peter, pizza)
for -sale (pizza).
for -sale (lemon).
for -sale (pontiac).
can -buy (Name, Model) if person (Name)
                and car (Model)
                and likes (Name, Model)
                and for -sale (Model).
```

На приглашение Пролога "Goal:" можно задать следующие вопросы:

```
can -buy (Who,What).
can -buy (peter,What).
can -buy (kelly,lemon).
can -buy (Who,pontiac).
```

Когда выполняется процедура поиска с возвратом, осуществляется поиск другого решения целевого утверждения. Это осуществляется путем возврата к последней основной, имеющей альтернативное решение, подцели, реализации альтернативы и очередного возврата.

Можно использовать поиск с возвратом для выполнения повторяющихся процессов.

Следующая программа иллюстрирует использование поиска с возвратом для выполнения повторяющихся операций:

```
predicates
  country (symbol)
  print -countries
clauses
  country (england).
  country (france).
  country (germany).
  country (denmark).

print -countries :-
    country(X),
    write(X),           /* записать значение X */
    nl,                 /* начать новую строку */
    fail.
```

Предикат countries составляет список названий различных стран.

Предикат print -countries отпечатывает все найденные решения.

Внутренний предикат fail всегда вызывает режим "неудачное завершение", но можно вызвать режим "поиска с возвратом", обратившись и к таким целевым утверждениям, как, например: 5 = 2 + 2 или country (shangry - la).

Первый проход выполняется до конца, X присваивается значение "england", которое выводится на печать. Затем Пролог-программа возвращается в начало, и при отсутствии другого маршрута к выполнению nl или write(X) переходит к поиску следующего решения предиката country(X).

При этом, выполняя country(X) повторно, программа освобождает переменную X, находит альтернативное решение предиката country(X) и присваивает X новое значение. После этого обработка продолжается, и название другой страны выводится на печать.

В конце концов, первое предложение проверится для всех альтернатив. Остается выполнить только второе предложение без каких-либо дополнительных сложностей. Целевое утверждение print -countries успешно завершится следующим результатом:

```
england
france
germany
denmark
Yes.
```

Если бы второго предложения не было, то целевое утверждение print - countries не выполнилось, и последнее сообщение было бы No, не говоря о результате, который был бы таким же.

КОНТРОЛЬНЫЕ ВОПРОСЫ.

1. Каково назначение правил Пролога?
2. Описать структуру правил в Прологе и их оформление.
3. Как в Прологе можно организовать циклический процесс?
4. Объяснить специфику использования механизма возврата.

АЛГОРИТМИЧЕСКИЙ ЯЗЫК "ПРОЛОГ"

ЛАБОРАТОРНАЯ РАБОТА № 3

"ИСПОЛЬЗОВАНИЕ АРИФМЕТИЧЕСКИХ И ЛОГИЧЕСКИХ ОПЕРАЦИЙ В ПРОЛОГЕ. БАЗОВЫЕ ПРОЦЕДУРЫ ВВОДА И ВЫВОДА".

Цель работы - приобретение практических навыков по использованию арифметических и логических операций и операций ввода-вывода на языке Пролог.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ, ВВОД И ВЫВОД В ПРОЛОГЕ.

АРИФМЕТИЧЕСКИЕ ОПЕРАТОРЫ

Turbo Prolog выполняет все базовые операции, присущие другим языкам программирования, между целыми и вещественными числами.

Таблица соответствия типов операндов и результата по операндам:

Операнд 1	Операнд 2	Оператор	Результат
integer	+ - *	integer	integer
real	+ - *	integer	real
integer	+ - *	real	real
real	+ - *	real	real
Integer Или real	/	Integer Или real	real

Для записи арифметического выражения используется стандартный предикат " = ":

A = (N - 1) / (B - 8)

Значение арифметического выражения может быть выполнено лишь тогда, когда все переменные, стоящие в правой части, являются конкретизированными.

Между различными константами и выражениями языка допускаются следующие бинарные операции:

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

Кроме того, Пролог допускает сложные логические выражения с использованием встроенных предикатов:

, или AND ,
; или OR ,
NOT .

Примеры:

X + 5 < 6 - T ,
"AAAB" > "AAAC" ,
'a' < 'b' ,
((X + 4) < 6) AND ((X + 10) > 7) .

Арифметические выражения вычисляются с использованием встроенных предикатов (встроенный предикат - процедура, входящая в систему программного обеспечения Пролога).

Предикат "=" (is) - также является инфиксным (то есть, по сути является операцией), в котором:

1-й аргумент - неконкретизированная переменная,

2-й аргумент - арифметическое выражение, не содержащее неконкретизированных переменных.

Предикат "=" унифицирует свободную переменную с результатом арифметического выражения.

Примеры:

1) Y = 10 * 5 + 3
   (Ответ: Y = 53) ;
2) 20 = 10 + 5 + 5
   (Ответ: Yes) ;
3) X = 3, Y = 2, X = Y
   (Ответ: No) .

Предикат "=" является детерминированным предикатом (то есть, при обратном ходе он повторно не выполняется).

Составные объекты могут сравниваться с использованием предиката "равенство" следующим образом:

```
domains
   d = pair(integer, integer);
     single (integer);
     none
predicates
   equal (d,d)
clauses
   equal (X,X).
```

Задание соответствующих целей приводит к следующим ответам:

```
equal (single(1),pair(1,2)).      - Ложно;
equal (pair(3,4),pair(3,4)).      - Истинно;
equal (none,none).                - Истинно.
```

Однако, при записи цели: equal(5,4) последует сообщение об ошибке из-за несоответствия доменов.

Для достижения последней цели необходимо добавить следующее описание предиката equal :

```
equal (integer,integer).
```

Тогда:

equal (5,4). - Ложно;
equal (5,5). - Истинно.

При сравнении объектов типа real нужно быть особенно осторожным, так как точное сравнение во многих случаях невозможно.

Однако, можно, зачастую, применить следующий вариант предиката equal :

equal (X , Y) :- X/Y < 1.1 AND X/Y >0.9.

Кроме того, в Прологе допускаются следующие встроенные математические функции и предикаты:

Предикат	Описание
X mod Y	Остаток от деления X на Y
X div Y	Результат деления нацело X на Y
abs(X)	Модуль X
cos(X)	cos(X)
sin(X)	sin(X)
tan(X)	tg(X)
arctan(X)	arctg(X)
exp(X)	e^x
ln(X)	ln(X)
log(X)	lg(X)
sqrt(X)	Квадратный корень из X
random(X)	Устанавливает X, как случайное число с псевдоравномерным распределением от 0 до 1

Пример:

```
test1 (Xreal, Answer ) :-
    Answer = ln(exp(sin(sqrt(Xreal)))).
```

БАЗОВЫЕ ПРЕДИКАТЫ ВВОДА-ВЫВОДА В ПРОЛОГЕ

Для вывода используется стандартный предикат

write (Arg1, Arg2, ... argN) ,

где Argi - либо константа, либо переменная (но с установленным заранее значением), либо терм другого вида.

Стандартный предикат nl (new line) часто используется для пропуска заданного количества строк при редактировании потока вывода.

Пример:

```
write("----------"),
nl,
nl,
write("Результат = ", Res).
```

a. Для ввода в Turbo Prolog имеется ряд стандартных предикатов для чтения:

2. символьных строк;
3. целых, вещественных чисел и данных типа char с дисплея;
4. файлов с диска.

Рассмотрим эти предикаты подробнее.

readln(Line) - домен для переменной line должен быть либо типа string, либо типа symbol. Line - должна быть свободной переменной. Длина строки фиксируется нажатием клавиши Enter.

readint(X) - X - свободная переменная из домена целого типа. Если возможно преобразование, то оно осуществляется, иначе неудача и возврат.

readreal(X) - X - свободная переменная типа real (аналогично readint(X)).

readchar(CharP) - считывается единственный символ, и CharP принимает значение этого символа.

ПРИМЕР ПРОГРАММЫ НА ЯЗЫКЕ ПРОЛОГ

```
/* Пример программы, реализующей арифметические операции.
      value1, value2 - операнды
      code - операция                  */
domains
   code = symbol
   value1, value2 = integer
predicates
   oper ( value1, value2, code ).
clauses
   oper (X, Y, "+") :- R=X+Y, nl, write "Результат - ", R).
   oper (X, Y, "-") :- R=X-Y, nl, write "Результат - ", R).
   oper (X, Y, "*") :- R=X*Y, nl, write "Результат - ", R).
   oper (X, Y, "/") :- R=X/Y, nl, write "Результат - ", R).

/* Пример программы, реализующей логические операции.
      value1, value2 - границы интервала
      value3 - проверяемое число        */
domains
   value1, value2, value3 = integer
predicates
```

```
    include (value1, value2, value3).
clauses
    include (X, Y, Z) :- X AND Y, nl, write ("Число",Z,
        "лежит в интервале от ", X, "до ", Y).
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислить основные арифметические, логические операции и операции отношения.
2. Как определяется тип результата арифметической операции?
3. Классифицировать множество предикатов ввода-вывода.
4. Назвать основные стандартные математические функции и предикаты.
5. Объяснить специфику унификации в Прологе.

АЛГОРИТМИЧЕСКИЙ ЯЗЫК "ПРОЛОГ"

ЛАБОРАТОРНАЯ РАБОТА № 4

"РЕАЛИЗАЦИЯ ОКОННОГО ИНТЕРФЕЙСА В СРЕДЕ ""TURBO PROLOG"

Цель работы - приобретение практических навыков по реализации оконного интерфейса в среде Turbo Prolog.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ
РАБОТА С ОКНАМИ В ПРОЛОГЕ

Система Turbo Prolog позволяет реализовать многооконный интерфейс с разнообразной цветовой гаммой окна и его рамки.

Атрибуты цвета для конкретного окна устанавливаютсяс помощью значений соответствующего параметра.

Значение цвета выбирают из соответствующей таблицы:

Фон		Буквы	
Цвет	Код	Цвет	Код
черный	0	черный	0
серый	8	синий	1
синий	16	зеленый	2
голубой	24	бирюзовый	3
зеленый	32	красный	4
бирюзовый	48	лиловый	5
красный	64	коричневый	6
коричневый	96	белый	7
белый	112		

Для установки значения соответствующего атрибута необходимо:

1. Выбрать цвет буквы и фона.
2. Найти соответствующие коды, представленные в таблице и сложить их.

3. Для осуществления режима мерцания добавить 128.
4. Для получения интенсивного (светлого) цвета букв добавить 8.

Пример.

Для получения черно-белого изображения необходимо вычислить:

(0 + 7 = 7), 7 - задать в качестве атрибута.

В целом Turbo Prolog обеспечивает с помощью стандартных предикатов работу с окнами, то есть, с областями экрана, где непосредственно осуществляется ввод и вывод информации:

makewindow (...) - создание окна;
shiftwindow (...) - смена окна;
removewindow (...) - удаление окна;
clearwindow (...) - очистка окна;
window -attr (...) - установка атрибута окна.

Примечание: существуют еще предикаты считывания данных из окна в добавление к базовым предикатам ввода-вывода.

Рассмотрим подробнее работу с предикатами.

Предикат makewindow (...) .

Стандартный предикат makewindow (...) имеет следующие параметры:

makewindow (Who, Scrattr, Frameattr, Reader,
Row, Col, Height, Width).

Здесь

Who - номер окна (служит для выбора окна в многоокон ном режиме).
Scrattr - атрибут окна (устанавливает цвет букв и фона).
Frameattr - атрибут рамки - определяет цвет рамки окна (и заголовка).
Reader - заголовок ("" - заголовок не выводится).
Row, Col - строка и столбец для левого верхнего угла окна.
Height, Width - высота и ширина окна.

Параметр заголовка должен иметь тип string или symbol, остальные параметры обязательно - целые.

Координаты и размеры окна должны учитывать фактические размеры окна.

Пример:

makewindow (1, 7, 7, "Окно 1", 5, 4, 10, 20).

Если имеется рамка, то она также занимает две строки и два столбца. это необходимо учитывать при задании координат и размеров окна.

Правила использования предиката.

1. Требуется задание конкретных значений для координат окна и его описателей (отдельные переменные недопустимы или все параметры переменные).

2. Все базовые предикаты ввода-вывода работают непосредственно в текущем окне при многооконной работе.
3. Текст в окне выравнивается по правому краю и может свертываться, как и в полноэкранном режиме.
4. Предикаты управления экраном работают только внутри активного окна.

Переключение окон осуществляется предикатом
shiftwindow (windowNo) ,
где WindowNo - номер активизируемого окна.

Система Turbo Prolog запоминает все созданные окна, но курсор все время возвращается в последнее активное окно.

Если WindowNo не существует, то есть, не было создано окна с подобным номером, то возникает ошибочная ситуация.

Предикат removewindow

Удаляет из памяти текущее окно. Дальнейшие ссылки на данное окно недопустимы и приводят к ошибкам.

Предикат clearwindow

Очищает текущее окно от содержимого. При этом курсор помещаетсяв левый верхний угол окна.

Предикат window -attr (attr)

При необходимости атрибут окна (то есть, цвет букв и фона) может быть изменен. Данный предикат изменяет атрибут текущего окна.

Предикат Cursor

Этот предикат устанавливает позицию курсора (Row, Col) в текущем окне, если параметры установлены, и возвращает координаты, если при вызове предиката переменные были свободными.

ПРИМЕР ПРОГРАММЫ НА ЯЗЫКЕ ПРОЛОГ

```
/* Программа, иллюстрирующая оконый интерфейс */
predicates
   set (integer)
   r -set
clauses
   set(0) :- removewindow,
          shiftwindow (1),
          removewindow, !.
   set(X) :- removewindow,
     makewindow (2, X, X, "", 8, 15, 7, 50),
     write ("Введите код цвета фона и символов: "),
```

```
        nl,
        write ("                    0 - выход "),
        nl,
        r -set.

r -set:- readint(X),set(X).

goal
        makewindow (1, 112, 0, "", 0, 0, 25, 80),
        makewindow (2, 23, 30, "", 8, 15, 7, 50),
        write ("Введите код цвета фона и символов :"),
        nl,
        write ("                    0 - выход"),
        write ("                          "),
        r -set.
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. 1. Как формируется цветовое сочетание фона и текста в Прологе?
2. 2. Назвать основные правила использования предиката makewindow.
3. Как осуществляются переходы между окнами и их очистка?

АЛГОРИТМИЧЕСКИЙ ЯЗЫК "ПРОЛОГ"

ЛАБОРАТОРНАЯ РАБОТА № 5

"СОСТАВНЫЕ ОБЪЕКТЫ (СТРУКТУРЫ) И ИХ ОПИСАНИЕ"

Цель работы - приобретение практических навыков работы с составными объектами (структурами) в среде Turbo Prolog.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Turbo Prolog позволяет иметь дело с объектами, содержащими другие объекты. Такие "сложные" объекты называются составными объектами или структурами. В ряде случаев использование составных объектов существенно упрощает программы.

Составные объекты состоят из функтора (он определяет имя всей структуры)и подобъектов или компонент.

functor(object -1, object -2, ..., object -n).

Функтор без объектов записывается в виде

functor() или functor.

Из этого можно сделать вывод, что простой объект, есть частный случай составного объекта.

Описание домена в случае составного объекта имеет вид:

```
MyCompDom = f1(d1 -1, d1 -2, ..., d1 -n);
            f1(d2 -1, d2 -2, ..., d2 -n);
                . . . . . .
            f1(dM -1, dM -2, ..., dM -n);
```

где **fi** - функтор **i** - ой альтернативы;
(di -j) - множество компонент **i** -ой структуры.

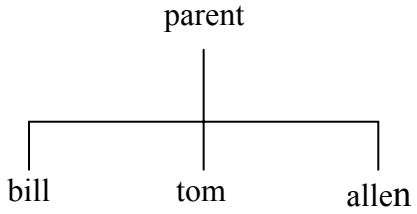
Различные альтернативные варианты представляются соответствующими функторами. Один вариант от другого отделяется символом ";". Любая из компонент di -j также может быть структурой, определенной в другой декларации, либо быть одним из стандартных типов - real, integer, symbol и пр.

Таким образом, необходимо помнить:

1. Альтернативы разделяются символом ";".
2. Альтернативаесть описание конкретной структуры и состоит из функтора и компонент.
3. Максимальное количество альтернатив в одном описании до мена - 250.

ПРИМЕРЫ СТРУКТУР НА ЯЗЫКЕ ПРОЛОГ

1. parent (bill, tom, allen); - то есть, обычный предикат также является структурой:

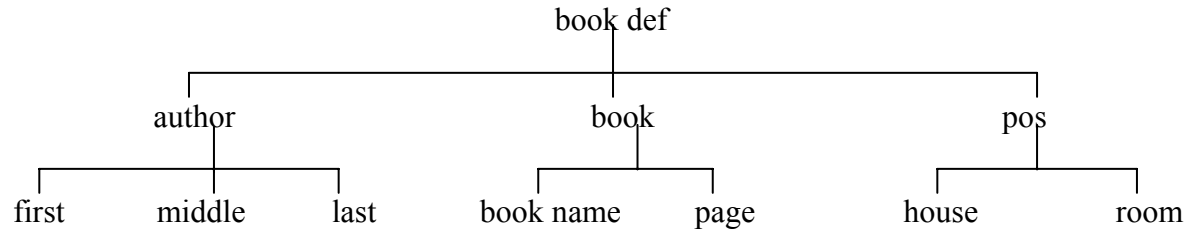


2. крточка-указатель - это структура, содержащая несколько компонент (сведения об авторе, название книги, дату издания, место хранения книги в библиотеке и т.д.):

```
domains
   author = name (first, middle, last)
   first, middle,last = symbol
   book = book -status (book -name, page)
   book -name = symbol
   page = integer
   pos = place (house, room)
   house, room = integer
predicates
   book -def (author, book, pos)
```

```
clauses
   book -def (name ("Иван", "Иванович", "Иванов"),
         book -status ("Сибирские рассказы", 150),
         place (3, 5).
```

То есть, имеем следующую структуру:



Максимальный уровень вложенности структур не оговаривается.

Правила, определяющие: подходит ли некоторое утвржение для огласования с целевым утверждением, имеют следующий вид:

1. Неконкретизированная переменная соответствует любому объекту. Этот объект становится значением переменной.
2. Целое число или атом соответствуют только самим себе.
3. Между структурами можно установить соответствие только в том случае, когда они имеют одинаковый функтор, одинаковое число параметров и соответствующие параметры имеют одинаковые домены.

ПРИМЕРЫ ПРОГРАММ НА ЯЗЫКЕ ПРОЛОГ, СОДЕРЖАЩИХ ТРУКТУРЫ

```
  /*      ПРОГРАММА  1    */
domains
   name = person (symbol, symbol, symbol)
     /* фамилия, имя, отчество */
   address = ad -ress (symbol, symbol, symbol, integer)
     /* город, округ, улица, дом */
predicates
   list (name, adress)
   get -list -address
   convert -location (symbol, integer)
   write -person (name, symbol, integer)
clauses
   get -list -address :-
     makewindow (1, 119, 119, "", 0, 0, 25, 80),
     makewindow (2,  30, 31, "Введите номер округа", 7, 27, 8, 30),
        write ("Северный    -- 1\n"),
        write ("Южный       -- 2\n"),
        write ("Восточный   -- 3\n"),
```

```
        write ("Западный    -- 4\n"),
        write ("Центральный -- 5\n              "),
        readint (Value),
        removewindow(),
        makewindow (3, 30, 31, " Список ", 0, 0, 25, 80),
        write (
  "\n ФАМИЛИЯ \t   ИМЯ   \t ОТЧЕСТВО  \t УЛИЦА  \t ДОМ\n\n"),
        list (Person, ad -ress ( -, Okrug, Street, House )),
        convert -location (Okrug, Value),
        write -person (Person, Street, House), fail.
    get -list -address :-
        write ("\n\n Нажмите любую клавишу: "),
        readchar( -).
    write -person (person (First -name, Second -name, Last -name),
                Street, House :-
          write ("   ", First -name, "\t   ", Second -name, "\t",
                Last -name, "\t  ", Street, "\t", House), nl.
    list (person ("Степанов", "Петр", "Васильевич"),
              ad -ress ("Москва", "Северный", "1-я улица", 146)).
    list (person ("Куликов", "Андрей", "Петрович"),
              ad -ress ("Москва", "Южный", "2-я улица", 16)).
    list (person ("Петров", "Иван", "Васильевич"),
              ad -ress ("Москва", "Восточный", "3-я улица", 14)).
    list (person ("Сидоров", "Андрей", "Геннадьевич"),
              ad -ress ("Москва", "Западный", "4-я улица", 46)).
    list (person ("Шпачилова", "Анна", "Сергеевна"),
              ad -ress ("Москва", "Южный", "999-я улица", 34)).

    convert -location ("Северный", 1).
    convert -location ("Южный", 2).
    convert -location ("Восточный", 3).
    convert -location ("Западный", 4).
    convert -location ("Центральный", 5).

goal
    get -list -address, exit.

        /*   ПРОГРАММА  2    */
domains
   name = person (symbol, symbol, symbol)
     /* фамилия, имя, отчество */
   birthday = b -date (symbol, integer, integer)
         /* месяц, день, год */
   ph -num = symbol     /* номер телефона */
predicates
```

```
        phone -list (name, symbol, birthday)
        get -months -birthdays
        convert -month (symbol, integer)
        check -birthday -month (integer, birthday)
        write -person (name, symbol)
    clauses
        get -months -birthdays :-
          makewindow (1, 119, 119, "", 0, 0, 25, 80),
          makewindow (2, 30, 31, "", 7, 27, 3, 30),
          write (" Введите номер месяца "),
          readint (Value),
          removewindow (),
          makewindow (1, 30, 31, "Список именинников данного месяца", 0, 0, 25,
80),
          write (              "\n  ФАМИЛИЯ  \t    ИМЯ    \t ОТЧЕСТВО    \t
ТЕЛЕФОН\n\n),
          phone -list (Person, Tel, Data),
          check -sirthday -month (Value, Date),
          write -person (Person, Tel), fail.
        get -months -birthdays :-
          write ("\n\n Нажмите любую клавишу: "),
          readchar ( -).
        write -person (person (First -name, Second -name, Last -name), Tel) :-
          write ("  ", First -name, "\t    ", Second -name, "\t",
                Last -name, "\t\t ", Tel), nl.
        check -birthday -month (Mon, b -date (Month,  -,  -)) :-
          convert -month (Month,Month1),
          Mon = Month1.
        phone -list (person ("Степанов", "Петр", "Васильевич"),
                        "767-8463", b -date ("январь", 3, 1955)).
        phone -list (person ("Куликов", "Андрей", "Петрович"),
                        "438-8400", b -date ("февраль", 5, 1985)).
        phone -list (person ("Петров", "Иван", "Васильевич"),
                        "555-5653", b -date ("март", 3, 1935)).
        phone -list (person ("Котов", "Михаил", "Петрович"),
                        "438-8400", b -date ("июнь", 17, 1980)).
        phone -list (person ("Жуков", "Лев", "Васильевич"),
                        "767-8463", b -date ("июнь", 20, 1986)).
        phone -list (person ("Иванов", "Артем", "Андреевич"),
                        "555-5653", b -date ("июль", 16, 1981)).
        phone -list (person ("Кириллов", "Степан", "Федорович"),
                        "767-2223", b -date ("август", 10, 1981)).
        phone -list (person ("Милованова", "Анна", "Николаевна"),
                        "438-8400", b -date ("сентябрь", 25, 1981)).
        phone -list (person ("Шапкина", "Наталья", "Кирилловна"),
```

```
                "438-8400", b -date ("октябрь", 20, 1952)).
    phone -list (person ("Кузнецов", "Виктор", "Михайлович"),
                "438-8400", b -date ("декабрь", 31, 1981)).
    convert -month ("январь", 1).
    convert -month ("февраль", 2).
    convert -month ("март", 3).
    convert -month ("апрель", 4).
    convert -month ("май", 5).
    convert -month ("июнь", 6).
    convert -month ("июль", 7).
    convert -month ("август", 8).
    convert -month ("сентябрь", 9).
    convert -month ("октябрь", 10).
    convert -month ("ноябрь", 11).
    convert -month ("декабрь", 12).
goal
    get -months -birthdays, exit.
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Объяснить назначение составных объектов.
2. Как описываются составные объекты?
3. Что такое "функтор"?
4. Каковы правила согласования утверждений?

ЛИТЕРАТУРА ПО ЯЗЫКУ PROLOG.

1. И.Братко. Программирование на языке Пролог для искусственного интеллекта.- М.: Мир, 1990.- 560 с.
1. Дж. Малпас. Реляционный язык Пролог и его применение.-М.: Наука, 1980.- 464 с.
2. А.Янсон. Турбо-Пролог в сжатом изложении.- М.: Мир, 1991.- 94 с.
3. Л.Стерлинг, Э.Шапиро. Искусство программирования на языке Пролог.- М.: Мир, 1990.- 235 с.
4. У.Клоксин, К.Меллиш. Программирование на языке Пролог.- М.: Мир, 1987.- 336 с.