# Практическая РАБОТА №4

# Возможности внутренних баз данных.

# Построение экспертных систем

# 1 Цели и задачи

Целями работы являются:

– изучение возможностей по модификации внутренних баз данных языка Пролог, освоение принципов использования «статических переменных» в языке Пролог;

* изучение возможности создания экспертных систем на языке Пролог, освоение принципов формирования полноценных приложений, которые могут взаимодействовать с пользователем для сбора дополнительной информации.

Для достижения поставленных целей требуется решить следующие задачи:

1) Изучить теоретический материал.

2) Выполнить задание с номером варианта, равным номеру бригады.

3) Представить на проверку файл текста программы на языке Пролог, решающей поставленную задачу, и файл базы знаний экспертной системы. Номер варианта и текст варианта задания должны быть представлены в форме комментариев в тексте программы. Номер группы и номер варианта должны присутствовать в имени файла с текстом программы.

# 2 Основные теоретические сведения

assert(X) – добавляет факт X в программу.

retract(X) – удаляет факт X из программы.

Добавление имеет две модификации assertz – добавить в конец программы, asserta – добавить в начало программы.

Пример использования:

?- assertz(data(1)).

yes

?- data(X).

X = 1

yes

?- listing.

data(1).

yes

?- retract(data(\_)).

yes

?- listing.

yes

Для добавления правил в программу их необходимо поместить в *дополнительные скобки*: assertz((Правило)).

Пример:

?- assertz(man(socrat)), assertz((mortal(X):-man(X))).

yes

?- mortal(socrat).

yes

?- mortal(X).

X = socrat

yes

?- listing.

% file: user\_input

mortal(A) :- man(A).

man(socrat).

yes

Предикат abolish(Имя\_предиката/Арность\_предиката) удаляет все вхождения предиката с данным именем и данной арностью.

Напишем аналогичный предикат удаления retractAll(X) через retract:

retractAll(X) :- retract(X), retractAll(X).

retractAll(\_).

Он удаляет все вхождения Х в нашу программу.

Статические переменные в Прологе можно определить с использованием assert и retrtact.

Способ работы:

init(ИмяПеременной, Значение) – инициализация статической переменной заданным значением.

set(ИмяПеременной, Значение) – установка значения в переменную.

get(ИмяПеременной, Значение) – получение значения из переменной.

Решение:

init(Var, Val) :- assertz(variables(Var,Val)).

set(Var, Val) :- retract(variables(Var, \_)), assertz(variables(Var,Val)).

get(Var, Val) :- variables(Var, Val).

Пример использования:

?- init(t, 3), init(v, 4).

yes

?- set(t, data), get(t, T), get(v, V).

T = data

V = 4

yes

Естественно, повторная инициализация приведет к неправильной работе программы.

# 3 Примеры выполнения заданий

**3.1 Пример 1**

Распространенный вариант использования внутренних баз данных — это повышение эффективности программ за счет добавления уже вычисленных фактов в базу данных. При попытке вычислить предикат сначала проверяется, нет ли в базе данных уже вычисленного значения, и если оно там уже есть, то просто берется это значение. Если же ответа еще нет, он вычисляется обычным способом, после чего добавляется в базу данных для повторного использования. Эта техника еще называется *мемоизация* или *табулирование*.

Давайте разработаем табулированную версию предиката, вычисляющего число Фибоначчи по его номеру.

Мы уже рассматривали предикат, вычисляющий числа Фибоначчи. Выглядел он следующим образом:

fib(0,1).

fib(1,1).

fib(N,V) :- N1 is N-1, N2 is N-2, fib(N1,V1), fib(N2,V2), V is V1+V2.

Чем плох этот вариант предиката, вычисляющего числа Фиббоначи? Получается, что при вычислении очередного числа происходит многократное перевычисление предыдущих чисел Фибоначчи, что не может не приводить к замедлению работы программы.

Изменим нашу программу следующим образом: добавим в нее раздел описания предикатов *внутренней базы данных*.

Для корректного обращения к динамическому предикату его следует определить как динамический. Для этого в программе следует вызвать dynamic(Имя\_предиката/Арность\_предиката), но некоторые версии Пролога могут обходиться без данного определения.

:- dynamic(fibon/2).

fib(0, 1).

fib(1, 1).

fib(N, V) :- N1 is N - 1, N2 is N - 2, (fibon(N1, V1); fib(N1, V1)), (fibon(N2, V2); fib(N2, V2)), V is V1 + V2, asserta(fibon(N,V)).

При такой реализации решения количество рекурсивных вызовов fib существенно уменьшается. Почему вместо fib используется fibon для хранения данных? fibon используется, т.к. fib определен как статический предикат, а Пролог не позволит вносить изменения в статические предикаты. Изменения можно вносить только в динамические предикаты!

**3.2 Пример 2**

Создать программу универсальный англо-русский словарь, которая переводит слова и динамически запоминает перевод новых для неё слов. Пользователь вводит слово. Если программа знает его перевод, то выводит перевод на экран, иначе спрашивает пользователя о том, какой перевод у незнакомого слова (и запоминает его).

:- dynamic(variables/2).

get(X,Y) :- variables(X,Y).

set(X,Y) :- assertz(variables(Y,X)), assertz(variables(X,Y)).

translate(X) :- get(X, Y), write(Y),!;

write('Not such word, give translation: '),

read\_atom(Z), set(X, Z).

# 4 Перечень заданий

Реализуйте экспертную систему, на языке Пролог, согласно одному из предложенных вариантов в соответствии со своим номером варианта.

Вопросы, задаваемые пользователю, не должны повторяться (дублироваться).

Базу знаний (начальные факты) требуется придумать самостоятельно (не менее 20 фактов).

Желательно, чтобы база знаний сохранялась в файле (и читалась при запуске программы).

В ответах/фактах из нескольких слов вместо пробелов следует писать символ "\_".

Желательно, чтобы факты, вопросы и интерфейс были написаны на русском языке или на транслите.

1. Электронный терапевт.

Имеется список болезней и характерных для них симптомов (в виде фактов базы знаний). (например, "грипп": кашель, высокая температура, головная боль). Требуется по введенным пользователем симптомам определить, чем он заболел (м.б. варианты).

1. Прогноз погоды.

Имеется набор признаков (база знаний) о погодных приметах. Программа узнает от пользователя признаки (задавая вопросы вида "облачно? ветрено?"), и на основе полученных сведений выдает прогноз погоды на следующий день.

1. Консультант по сотовой связи.

Имеется база знаний о моделях телефонов и их характеристиках/функциях. Также имеются факты о том, какие функции телефона нужны для конкретных целей (общение, выход в Интернет, мобильный офис...). Программа должна задать пользователю (покупателю) ряд вопросов о том, для чего ему нужен телефон, и предложить те модели телефонов, которые удовлетворяют его запросам.

1. Сервисный центр.

Имеется набор признаков неисправности компьютера (например, "нет картинки на мониторе и "при загрузке издаются звуки: 1 длинный и 2 коротких" => "неисправна видеокарта").Требуется, задавая пользователю вопросы о признаках неисправности, вынести решение о том, что неисправно в компьютере (возможно, в виде предположений/нескольких вариантов).

1. Ресторанный гид.

Имеется база знаний о том, блюда каких стран могут предложить рестораны (итальянская, французская, испанская кухня и т.п.). Требуется спросить пользователя, какие кухни он предпочитает (может быть несколько) и предложить ему список всех ресторанов, содержащих как минимум один из вариантов.

1. Помощник кинолога.

Имеется набор фактов (признаков), и породы собак, обладающих этими признаками. Требуется задать пользователю вопросы с вариантами ответов да/нет - чтобы угадать, какую породу собаки (из представленных) он загадал.

1. Библиотекарь-консультант.

Имеется база данных книг и жанры, которым они соответствуют. Жанры представлены иерархически (например, Фантастика / Научная фантастика / Футурология / ...). Программа спрашивает пользователя, какой жанр ему интересен (если есть поджанры, то уточняет какой именно из них и т.д.) и выдает список соотв. книг.

1. Сказка про репку.

В программу последовательно вводятся имена персонажей сказки, в результате чего после каждого ввода данных программа рассказывает сказку "Репка" (что за кого тянет).

1. Сказочные персонажи.

В базе знаний в виде фактов хранятся характеристики некоторых известных сказочных персонажей. Программа выводит список их имен. Пользователь загадывает одного из них. Программа задает вопросы о признаках загаданного персонажа, пытаясь его угадать.

1. Переводчик фраз.

Задание, аналогичное Примеру 2, но с возможностью перевода фраз (из двух и более слов).