

С. Григорьев

Йошкар-Ола

## Программирование на Прологе-Д

Программа на Прологе представляет собой базу знаний и следующий за ней вопрос. Построению базы знаний посвящена данная статья. Здесь рассматриваются структура программы и некоторые приемы технологии программирования, связанные с использованием встроенных предикатов, рекурсии, графики, специфичные для Пролога-Д. Как и прежде во всех случаях, где это необходимо, будут оговариваться различия Пролога-Д MSX и Пролога-Д БК-0010.

### Факты и правила

Особенностью языка Пролог-Д, отличающей его от других языков, используемых для работы на ЭВМ, является его применение не для программирования, но главным образом для описания данных и правил их обработки. Построение базы знаний на Прологе-Д означает выявление множества исследуемых объектов и связей между ними, совокупность которых описывает явление или процесс: иными словами, создание информационно-логической модели описываемого на языке Пролог-Д явления или процесса.

Первый шаг построения базы знаний состоит в выявлении объектов и соотношений между ними, отвечающих на вопрос «что дано?». Такую информацию целесообразно представлять в виде совокупности фактов. Классическим примером фактографии служит англо-русский словарь. Записанный средствами Пролога-Д, он выглядит так:

```
русангл(мама, mammy) ;  
русангл(небо, sky) ;  
русангл(солнце, sun) ;  
русангл(мальчик, boy) ;
```

и т.д. Подобные отношения представляют собой грамматическую конструкцию Пролога-Д, называемую фактом. Факт задается в виде функционала: имя и совокупность аргументов. В данном примере «русангл» - это имя; оно определяет информацию, записываемую в факте. Русские и английские слова мама, mammy, небо, sky, солнце, sun, мальчик, boy представляют собой аргументы фактов, определяющих взаимно-однозначное соответствие между русскими и английскими словами. Необязательно, чтобы факт имел два аргумента. Например, факт

```
мужчина(Николай) ;
```

имеет один аргумент Николай, а факт

```
родился(Петров, Иван, 10, сентябрь, 1979) ;
```

имеет пять аргументов. Однако с точки зрения синтаксиса языка Пролог-Д необходим хотя бы один аргумент. Если факт в базе знаний имеет имя и не имеет аргументов, то система выдаст сообщение о синтаксической ошибке.

Приведенный пример, по сути дела, уже является базой знаний. Перед этой базой знаний можно ставить различные вопросы:

```
?русангл(у, х) ;
```

если необходимо узнать все слова, хранящиеся в базе знаний;

```
?русангл(мама, х) ;
```

чтобы узнать, как по-английски «мама»;

?русангл(x, sky);

чтобы узнать, что значит слово sky.

Для описания всего множества информации достаточно фактов. Однако если можно задать некоторые связи и отношения между объектами, то удастся сократить число фактов и тем самым сделать базу знаний более лаконичной. Связи и отношения между объектами задаются правилами. При построении правил выделяется совокупность отношений, отвечающих на вопрос «что известно?». Правило можно построить, пользуясь известным принципом разделения исходной задачи на более простые, которые тоже, в свою очередь, могут быть разделены. Этот процесс известен под названием декомпозиции задачи. Декомпозиция заканчивается в тот момент, когда отношения связывают зафиксированные в базе знаний объекты. Например, в задаче о построении родственных отношений можно определить следующие правила:

бабушка(x, y) ← -мама(x, z), мама(z, y);

бабушка(x, y) ← -мама(x, z), папа(z, y);

дедушка(x, y) ← -папа(x, z), папа(z, y);

дедушка(x, y) ← -папа(x, z), мама(z, y);

Левая часть правила называется головой; разделенные запятыми выражения в правой части - целями; последние образуют тело правила.

Процесс декомпозиции не обязательно однозначен. Даже простой пример о родственниках допускает и иную трактовку. Если ввести правило, определяющее понятие «родитель»

родитель(x, y) ← -мама(x, y);

родитель(x, y) ← -папа(x, y);

то бабушку и дедушку можно определить проще:

бабушка(x, y) ← -мама(x, z), родитель(z, y);

дедушка(x, y) ← -папа(x, z), родитель(z, y);

Если к только что записанным правилам добавить несколько фактов, определяющих мам и пап, то получается база знаний «семья»:

мама(Саша, Петя);

папа(Сережа, Петя);

мама(Оля, Саша);

папа(Коля, Саша);

мама(Люда, Сережа);

папа(Петя, Сережа);

родитель(x, y) ← -мама(x, y);

родитель(x, y) ← -папа(x, y);

бабушка(x, y) ← -мама(x, z), родитель(z, y);

дедушка(x, y) ← -папа(x, z), родитель(z, y);

В данном примере для определения понятия родитель(x, y) потребовалось более одного правила. По сути дела, здесь использовано недетерминированное ветвление, дающее альтернативное определение этого отношения и используемое системой после того, как было применено первое отношение. Следует подчеркнуть, что в определении участвуют оба правила. В общем случае число правил не ограничено.

Упражнения.

1. В электронике известно понятие «стандартный ряд номинальных значений сопротивлений»: 10, 11, 12, 13, 15, 16, 18, 20, 22, 24, 27, 30, 33, 36, 39, 43, 47, 51, 56, 62, 68, 75, 82, 91. Любое сопротивление, выпущенное промышленностью, имеет номинал, кратный элементам указанного ряда. Напишите базу знаний, в которой описывается этот ряд.
2. Опишите на языке Пролог-Д состав своей семьи.
3. Составьте базу знаний, описывающую республики, входящие в состав СССР.
4. Напишите на языке Пролог-Д таблицу умножения чисел от 1 до 10. Какое количество предложений требуется для записи этой базы знаний?

## Арифметические и другие встроенные предикаты в Прологе-Д

Системы логического программирования, к числу которых относится и Пролог-Д, не предназначены для вычислений. Традиционный для Пролога-Д подход при выполнении арифметических действий дан в упражнении 4 из предыдущего раздела. Однако для определения таким образом всех математических действий памяти компьютера будет явно недостаточно. Поэтому традиционные действия, связанные с выполнением арифметических операций, осуществляются посредством специальных, так называемых встроенных предикатов.

В системе Пролог-Д **БК-0010** для выполнения арифметических действий предусмотрен один встроенный арифметический предикат:

`ВЫЧ(Арг1, Арг2, Арг3, Арг4)`

В системе Пролог-Д **MSX** для выполнения арифметических действий предусмотрены два встроенных арифметических предиката:

`УМНОЖЕНИЕ(Арг1, Арг2, Арг3, Арг4)`

`СЛОЖЕНИЕ(Арг1, Арг2, Арг3)`

Встроенный предикат `ВЫЧ` (в **MSX** `УМНОЖЕНИЕ`) имеет четыре аргумента (целых; переменных, конкретизированных целыми; неконкретизированных переменных). `ВЫЧ` (в **MSX** `УМНОЖЕНИЕ`) обеспечивает реализацию формулы

$$\text{Арг1} * \text{Арг2} + \text{Арг3} = \text{Арг4}$$

Предикаты предусматривают обратимость аргументов и полностью покрывают арифметические операции в области целых чисел, предусмотренных синтаксисом входного языка ( $0 \leq \langle \text{число} \rangle \leq 65535$  «Электроника **БК-0010**» и  $-32766 \leq \langle \text{число} \rangle \leq 32766$  в системе **MSX**). Если результат дробный, он округляется до целого. Оба предиката могут быть использованы только в качестве цели в предложении.

Предикат `СЛОЖЕНИЕ(Арг1, Арг2, Арг3)` обеспечивает реализацию формулы

$$\text{Арг1} + \text{Арг2} = \text{Арг3}$$

Следующая база знаний на языке Пролог-Д показывает, как можно описать любые арифметические операции в системе Пролог-Д **БК-0010**:

`сложение(x, y, z) :- ВЫЧ(1, x, y, z);`

`вычитание(x, y, z) :- ВЫЧ(1, x, z, y);`

`умножение(x, y, z) :- ВЫЧ(x, y, 0, z);`

`деление(x, y, z) :- ВЫЧ(y, z, 0, x);`

Во всех четырех случаях  $X$ ,  $Y$  - операнды операций, а  $Z$  - результат. Например, СЛОЖЕНИЕ ( $X, Y, Z$ ) реализует арифметическую операцию  $Z=X+Y$ .

Аналогично в системе Пролог-Д **MSX**:

вычитание ( $x, y, z$ ) <- УМНОЖЕНИЕ ( $1, x, z, y$ );

умнож ( $x, y, z$ ) <- УМНОЖЕНИЕ ( $x, y, 0, z$ );

деление ( $x, y, z$ ) <- УМНОЖЕНИЕ ( $y, z, 0, x$ );

Обратите внимание, вычитание можно определить в MSX и иначе:

вычитание ( $x, y, z$ ) <- СЛОЖЕНИЕ ( $y, z, x$ );

Более подробное описание синтаксиса встроенных арифметических предикатов ВЫЧ, УМНОЖЕНИЕ, СЛОЖЕНИЕ приведено в технических описаниях трансляторов.

*Пример 1.* На Прологе-Д опишем вычисление площади прямоугольника, имеющего стороны длиной  $a$  и  $b$ :  $S=a*b$ . Соответствующий предикат должен иметь три аргумента - длины сторон и величину площади. Имя предиката должно отражать его назначение, этому критерию удовлетворит имя «площадь». В системе Пролог-Д **БК-0010**:

умножение ( $x, y, z$ ) <- ВЫЧ ( $x, y, 0, z$ );

площадь ( $a, b, S$ ) <- умножение ( $a, b, S$ );

Первый предикат «умножение» потребовалось определить для наглядности записи. Необходимо отметить, что предикат «площадь» обратим, это означает, что, пользуясь этим описанием, можно вычислить не только площадь по заданным сторонам, но и любую (одну) сторону по другой стороне и площади. Перед базой знаний можно поставить вопрос:

?площадь ( $10, 20, S$ );

Ответ системы Пролог-Д:  $S=200$

?площадь ( $a, 20, 100$ );

Ответ системы Пролог-Д:  $a=5$

В системе Пролог-Д **MSX**:

умнож ( $x, y, z$ ) <- УМНОЖЕНИЕ ( $x, y, 0, z$ );

площадь ( $a, b, S$ ) <- умнож ( $a, b, S$ );

*Пример 2.* На Прологе-Д необходимо описать вычисление объема параллелепипеда высотой  $h$ , в основании которого прямоугольник, имеющий стороны длиной  $a$  и  $b$ :  $V=a*b*h$ . Соответствующий предикат должен иметь четыре аргумента - длины сторон  $a$ ,  $b$ , высоту  $h$  и объем  $V$ .

В системе Пролог-Д **БК-0010**:

умножение ( $x, y, z$ ) <- ВЫЧ ( $x, y, 0, z$ );

объем ( $a, b, h, V$ ) <- умножение ( $a, b, S$ ), умножение ( $S, h, V$ );

Как и прежде, предикат «объем» обратим, это означает, что, используя это описание, можно вычислить не только объем по заданным сторонам и высоте, но и любую (одну) сторону или высоту по высоте, стороне и объему.

Можно иначе записать объем, если воспользоваться формулой  $V=Sz$ . Эту базу знаний предлагается написать самостоятельно.

Перед данной базой знаний можно поставить вопрос:

?объем(10,20,5,V);

Ответ системы Пролог-Д: V=1000

В системе Пролог-Д **MSX**:

```
умнож(х,у,z)<-УМНОЖЕНИЕ(х,у,0,z);
объем(а,b,h,v)<-умнож(а,b,s),умнож(s,h,v);
```

Наряду с арифметическим предикатом в системе БК-0010 существуют два предиката БОЛ и НЕ.

Встроенный предикат БОЛ (Арг1,Арг2) предназначен для сравнения двух целых констант или переменных. Он имеет два аргумента (целых или переменных конкретизированных целыми). Оба аргумента к моменту выполнения должны быть определены. Если эти требования не выполнены, то появится сообщение об ошибке: «Функция не может быть выполнена (ошибка 34)». Предикат выполнен, если  $Арг1 > Арг2$ , иначе не выполнен.

Несмотря на то что предикат БОЛ один, его достаточно для описания всех возможных предикатов для сравнения числовой информации: равенство - «равно», меньше - «меньше», меньше и равно - «мир» и т. д. Это показывает база знаний, приведенная ниже.

```
равно(х,х);
меньше(х,у)<-БОЛ(у,х);
мир(х,у)<-НЕ(БОЛ(х,у));
```

В последнем предложении использован встроенный предикат НЕ, его синтаксис:

НЕ(Арг1);

Он имеет один аргумент, который обязательно должен быть предикатом. Предикат НЕ выполнен тогда и только тогда, когда предикат-аргумент не выполнен.

А теперь несложный пример, иллюстрирующий применение БОЛ и НЕ.

*Пример 3.* Опишите на языке Пролог-Д вычисление функции Хевисайда, определяемую формулой:

```
h(x)=0, если x<=0;
h(x)=1, если x>0.
```

База знаний должна содержать описание предиката «меньше и равно», который выше уже был описан; предикат, выполняющийся при вычислении функции Хевисайда, будет называться «хевисайд». Этот предикат будет иметь два аргумента: первый - аргумент функции, второй - ее значение. Предикат «хевисайд» определяется через два альтернативных описания для обоих значений х.

```
мир(х,у)<-НЕ(БОЛ(х,у);
хевисайд(х,0)<-мир(х,0);
хевисайд(х,1)<-БОЛ(х,0);
```

Перед этой базой знания можно поставить различные вопросы.

?хевисайд(20,х);

Ответ системы Пролог-Д: X=1

В системе Пролог-Д **MSX** существует полный набор предикатов сравнения, их имена БОЛЬШЕ, МЕНЬШЕ, РАВНО.

Еще один, последний, встроенный предикат - «отсечение», предназначенный для управления логическим выводом. Этот предикат потребуется для решения следующих проблем:

- ограничения количества найденных решений;
- нахождения некоторого особенного решения задачи;
- ограничения объема поиска с целью повышения эффективности работы системы.

Предикат «отсечение» обозначается восклицательным знаком `!`. Это традиционное обозначение отсечения в системах логического программирования. Если данный предикат использовать в качестве цели в предложении, то полученный при этом эффект можно проиллюстрировать дверью, через которую можно пройти только слева направо, но нельзя вернуться назад через эту дверь. Роль двери выполняет символ `!`. Как известно, система Пролог-Д будет пытаться выполнять цели предложения в порядке просмотра слева направо начиная от символа `<-`, от первой до последней цели. Если какая-либо цель оказывается невыполненной, то осуществляется возврат и делается попытка найти альтернативные решения. Отсечение ограничивает возможность поиска альтернатив с того момента, как была просмотрена цель, обозначенная символом `!`. Например, если не выполнены цели А,Б,В, возврат для нахождения альтернативных решений в предложении

пример `<-А,Б,В,!,Г,Д,Е;`

возможен, а если не выполнены цели Г, Д или Е, то уже нет. Рубикон при движении слева направо перейден.

Необходимо отметить важность этого предиката, особенно при описании задач, допускающих множественные решения.

Иллюстрация предиката «отсечение» на примере базы знаний «мама». Действительно, у человека не может быть две матери, поэтому, определив для данного человека имя матери, необходимо прекратить дальнейшие поиски.

```
мама(Наташа,Петя)<-!;  
мама(Наташа,Ваня)<-!;  
мама(Оля,Лена)<-!;  
мама(Катя,Даша)<-!;  
мама(Люда,Сереза)<-!;  
мама(Лена,Костя)<-!;
```

Перед базой знаний может быть поставлен вопрос

```
?мама(х,Петя);
```

Ответ системы Пролог-Д:

```
х=Наташа  
ДРУГИХ РЕШЕНИЙ НЕТ
```

После отыскания первого решения поиск альтернатив не производится. Что будет, если убрать во всех предложениях предикаты отсечения?

*Упражнения.*

1. Опишите на языке Пролог-Д вычисление площадей геометрических фигур: трапеции, треугольника, параллелограмма.
2. Опишите вычисление площади круга и длины окружности. Какова точность вычислений этих величин? Можно ли вычислить радиус круга по длине окружности?

3. На языке Пролог-Д напишите базу знаний, в которой определяется функция, заданная соотношением:

$F(x)=x^2$ , если  $x<1$ ,

$F(x)=x+1$ , если  $-1\leq x<1$ ,

$F(x)=x^2$ , если  $x>1$ .

4. Какие сложности могут возникнуть в базе знаний о мамах, если у двух мам дети будут тезками?

5. Предположим, что дана программа на Прологе-Д:

```
ff(xx);
```

```
ff(x)←pp(xx),ff(x);
```

Каким должен быть предикат pp(x), чтобы система нашла одно решение; бесконечно много решений?

## Рекурсия

Существует огромное количество задач, в которых отношения между объектами можно определить, только используя сами определяемые соотношения. При этом получаются правила, называемые рекурсивными. Применение рекурсии для списания задач при работе с системами логического программирования - широко распространенный прием.

Рекурсия будет проиллюстрирована несколькими примерами построения программ, как вычислительных, так и логических.

Первым примером будет пример вычисления наибольшего общего делителя (НОД) двух чисел. Предикат, который выполняется, если найден НОД двух данных чисел, будет иметь имя «нод» и три аргумента: числа a, b и значение НОД - c. Для описания вычисления НОД используются следующие соображения:

если  $a=b$ , то  $c=a=b$ ;

если  $a>b$ , то необходимо вычислить НОД для чисел b и  $a-b$ ;

если  $b>a$ , то необходимо вычислить НОД для чисел a и  $b-a$ .

Эти три утверждения естественным образом могут быть записаны на Прологе-Д в системе **БК-0010**.

```
нод(a,a,a);
```

```
нод(a,b,c)←БОЛ(a,b),вычитание(a,b,d),нод(b,d,c);
```

```
нод(a,b,c)←БОЛ(b,a),вычитание(b,a,d),нод(a,d,c);
```

```
вычитание(x,y,z)←ВЫЧ(1,y,z,x);
```

Если перед этой базой знаний поставить вопрос

```
?нод(10,15,x);
```

то система Пролог-Д ответит:

```
x=5
```

```
ДРУГИХ РЕШЕНИЙ НЕТ
```

Предикат «нод» оказывается обратимым.

В системе Пролог-Д **MSX** последняя строка базы знаний запишется иначе:

```
вычитание(x,y,z)←СЛОЖЕНИЕ(y,z,x);
```

а предикат БОЛ необходимо заменить на БОЛЬШЕ. Других изменений нет.

В качестве второго примера рассмотрим задачу о вычислении элементов последовательности 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... известной как последовательность Фибоначчи. Каждый ее элемент определяется следующими правилами:

$$\begin{aligned} f(0) &= 0, \\ f(1) &= 1, \\ f(n) &= f(n-1) + f(n-2) \end{aligned}$$

Первая формула утверждает, что значение нулевого элемента последовательности равно нулю. Это можно записать в виде факта

$$\PhiБ(0, 0);$$

Вторая утверждает, что первый элемент равен 1. На Прологе-Д это можно записать так:

$$\PhiБ(1, 1);$$

Третья формула представляет собой запись рекурсивного соотношения, переписываемого так:

$$\PhiБ(N, X) \leftarrow \text{БОЛ}(N, 1), \text{ВЫ}(N, 1, M), \text{ВЫ}(N, 2, K), \PhiБ(M, Y), \PhiБ(K, Z), \text{СЛ}(Y, Z, X);$$

«ВЫ» и «СЛ» - имена предикатов «вычитание» и «сложение», определенных с помощью правил

$$\begin{aligned} \text{СЛ}(X, Y, Z) &\leftarrow \text{ВЫЧ}(1, X, Y, Z); \\ \text{ВЫ}(X, Y, Z) &\leftarrow \text{ВЫЧ}(1, Y, Z, X); \end{aligned}$$

в системе **БК-0010**. Использование таких обозначений объясняется тем, что длина одной строки в редакторе системы Пролог-Д БК-0010 не должна превышать 64 символов.

Данные предложения представляют собой базу данных на языке Пролог-Д, позволяющую вычислять значения элементов последовательности. На вопрос

$$?\PhiБ(8, X);$$

система Пролог-Д ответит:

$$\begin{aligned} X &= 21 \\ \text{ДРУГИХ РЕШЕНИЙ НЕТ} \end{aligned}$$

Поиск числа Фибоначчи с порядковым номером более 8 невозможен в БК-0010 из-за ограниченной памяти, поэтому рекомендуется первым предложением базы знаний написать

$$\PhiБ(N, Y) \leftarrow \text{БОЛ}(N, 8), !;$$

Необходимо сказать, что такой путь решения данной задачи не самый лучший. Для нахождения  $N+1$  числа Фибоначчи требуется  $2(N+1)-1$  рекурсивное обращение. Однако этого можно избежать, если перейти к другой базе знаний, в которой предикат с именем «Фиб» определен как имеющий три аргумента, включающий в себя в качестве аргумента значения  $N$ -го и  $N-1$ -го элементов последовательности. Запишем эту базу знаний.

1. Нулевой элемент последовательности равен нулю, а «минус первый» не определен:

$$\PhiИБ(0, X, 0);$$

в данном случае значение  $X$  может быть любым.

2. Первый элемент последовательности равен 1, а нулевой - 0:

$$\PhiИБ(1, 0, 1);$$

3. Для остальных элементов:



$$\text{Фиб}(N, F, f) \leftarrow \text{БОЛ}(N, 1), \text{вы}(N, 1, M), \text{Фиб}(M, \Phi, F), \text{сл}(\Phi, F, f);$$

Обращение к этой базе знаний будет иметь вид:

$$?\text{Фиб}(10, F, f);$$

Ответ системы Пролог-Д:

$$F=55,$$

$$f=34$$

ДРУГИХ РЕШЕНИЙ НЕТ

При работе с этой базой знаний для вычисления N-го числа Фибоначчи необходимо всегда лишь N рекурсивных обращений.

Для системы Пролог-Д характерна особенность, проявляющаяся при работе с рекурсивными программами. В общем случае порядок предложений в базе знаний не имеет значения. Однако в нижеследующем примере это не так.

$$\text{родитель}(X) \leftarrow \text{родитель}(Y), \text{отец}(Y, Z);$$

$$\text{родитель}(\text{Коля});$$

$$\text{отец}(\text{Коля}, \text{Петя});$$

$$\text{родитель}(\text{Петя});$$

В первом предложении голова имеет то же имя, что и одна из целей - «родитель». В процессе поиска ответа в этой базе знаний будет применено правило «предложение, стоящее первым, будет и применено первым», известное как принцип поиска в глубину. Это приведет к тому, что система будет обращаться только к первому предложению базы знаний и ответ на вопрос

$$?\text{родитель}(\text{Петя});$$

не будет найден никогда. Вместе с тем небольшое изменение базы знаний, перестановка двух предложений, приводит к удачному поиску решения:

$$\text{родитель}(\text{Коля});$$

$$\text{родитель}(X) \leftarrow \text{родитель}(Y), \text{отец}(Y, X);$$

$$\text{отец}(\text{Коля}, \text{Петя});$$

$$?\text{родитель}(\text{Петя});$$

Неограниченно-повторное обращение к предложению может быть и более замаскированным, как это получается в примере:

$$\text{выше}(A, B) \leftarrow \text{ниже}(B, A);$$

$$\text{ниже}(B, A) \leftarrow \text{выше}(A, B);$$

$$\text{выше}(\text{Коля}, \text{Петя});$$

$$?\text{ниже}(\text{Петя}, \text{Коля});$$

Однако если третье предложение стоит на первом месте, то повторного обращения не произойдет и ответ будет найден. Такая ситуация называется петлей.

При вычислении элементов последовательности Фибоначчи может появляться бесконечная петля при исполнении программы. В самом деле, если вопрос имеет вид

$$?\text{Фиб}(0, x, y);$$

то первый возможный результат

$x=0$ ,  
 $y=1$

Далее в попытке отыскать следующее решение возникает бесконечная петля так как будет отыскиваться

Фиб(-1,x,y), Фиб(-2,...),...

Для контроля за подобной ситуацией необходима модификация базы знаний.

Первые два предложения должны быть записаны в виде

Фиб(0,x,1) <- ! ;

Фиб(1,1,1) <- ! ;

тогда при поиске альтернативного решения после получения ответа на вопрос:

?Фиб(0,x,y) ;

$x=0$

$y=1$

будет получен результат: БОЛЬШЕ РЕШЕНИЙ НЕТ. Данный пример иллюстрирует первое возможное использование предиката «отсечение».

Еще одно, чисто эстетическое предложение. База знаний на Прологе-Д будет выглядеть лучше, если предложения с одинаковыми именами расположены в одном месте. Для сравнения приводятся две базы знаний:

1.

мама(Таня,Надя) ;

бабушка(x,y) <-~мама(x,z),мама(z,y) ;

мама(Надя,Катя) ;

2.

мама(Таня,Надя) ;

мама(Надя,Катя) ;

бабушка(x,y) <-~мама(x,z),мама(z,y) ;

*Упражнения.*

1. Написать базу знаний, описывающую вычисление факториала.
2. Написать базу знаний, описывающую вычисление суммы чисел натурального ряда.
3. Написать базу знаний, описывающую вычисление суммы квадратов чисел натурального ряда.
4. Описать вычисление наименьшего общего кратного.
5. Написать базу знаний, описывающую известную притчу «У попа была собака, он ее убил. Она съела кусок мяса, он ее убил, на могиле написал...» и т. д. Как сделать, чтобы эта база знаний не содержала петли?
6. Написать базу знаний, описывающую сказку о репке, которую тянут-потянут, а вытянуть не могут.