

Практическое занятие №1

Знакомство с Visual Prolog. Структура программы на Прологе

Краткие теоретические сведения

1. ИНТЕРФЕЙС СРЕДЫ РАЗРАБОТКИ VISUAL PROLOG

















Для запуска среды Visual Prolog следует нажать иконку  на рабочем столе либо выбрать Пуск - Программы-Visual Prolog 5.2 -Visual Prolog 32.exe, после чего появится окно с главным меню и панелью инструментов:



Рис.1. Главное меню и панель инструментов Visual Prolog

- | | |
|---|-------------------------------------|
|  | Создание нового файла. |
|  | Открытие файла |
|  | Сохранение файла. |
|  | Отмена действия (Undo). |
|  | Возврат отмененных действий (Redo). |
|  | Вырезать. |
|  | Копировать. |
|  | Вставить. |
|  | Компиляция проекта. |
|  | Построение проекта. |
|  | Запуск проекта на выполнение. |
|  | Тестирование секции программы Goal. |
|  | Просмотр кода проекта. |
|  | Изменение шрифта. |
|  | Просмотр файла помощи. |

Прежде, чем приступить к изучению основ логического программирования необходимо ознакомиться с интерфейсом Visual Prolog:

- окно редактора (рис.2) предназначено для набора кода программы;



Рис.2. Окно редактора Visual Prolog

- окно сообщений (рис.3), в котором можно проследить операции, выполняемые средой в целом;

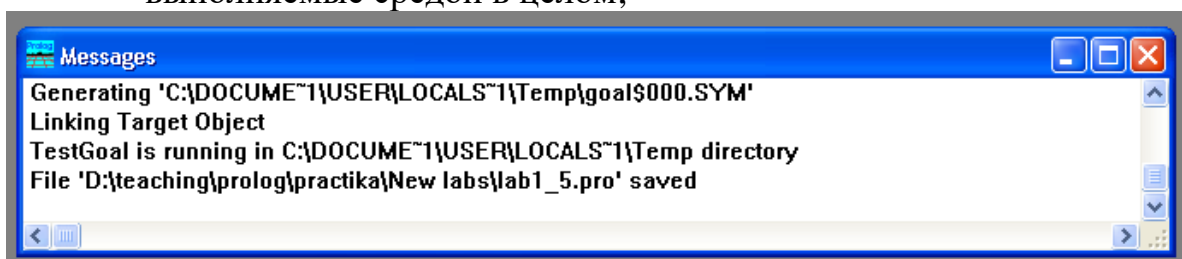


Рис.3. Окно сообщений Visual Prolog

- окно выдачи ошибок (рис.4) (двойной щелчок на ошибке позволяет перевести курсор в коде программы в ту позицию, где была допущена ошибка);

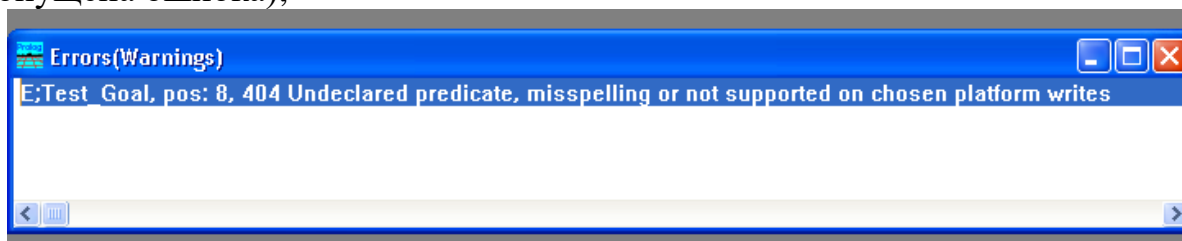


Рис.4. Окно информирования об ошибках Visual Prolog.

- после исправления ошибки получим результат работы программы в окне выдачи результатов (рис.5).

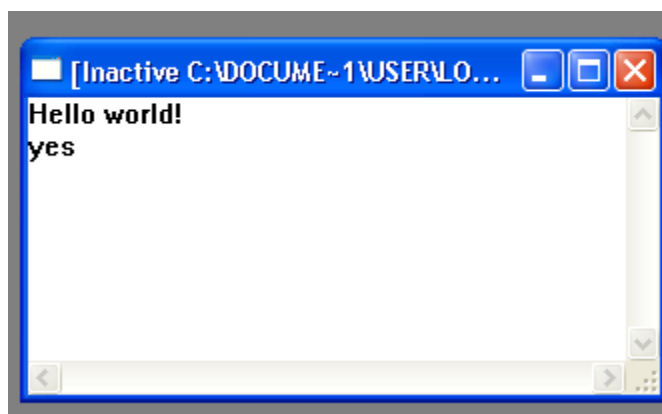


Рис.5. Окно выдачи результатов выполнения команды Test Goal

2. СТРУКТУРА ПРОГРАММЫ VISUAL PROLOG

Обычно программа Visual Prolog включает три или четыре основных раздела. Это раздел выражений clauses, раздел описания предикатов predicates, раздел доменов domains и раздел цели goal.

2.1. Раздел *clauses*

В разделе выражений *clauses* программист размещает все включаемые в программу факты и правила.

Выражения, относящиеся к определенному предикату, должны размещаться в разделе *clauses* вместе. Последовательность определяющих предикат выражений называется ПРОЦЕДУРОЙ.

При попытке удовлетворения цели Visual Prolog работает с самого начала раздела *clauses*, просматривая в процессе поиска последовательно каждый факт или правило. По мере прохождения раздела *clauses* Visual Prolog устанавливает внутренние указатели после каждого выражения, которое удовлетворяет текущей подцели. Если такое выражение не является частью ведущего к решению логического пути, то Visual Prolog возвращается к установленному указателю и ищет другое соответствие. Такой процесс называется поиском с возвратом (backtracking).

Фактом называют отношение или свойство, о котором известно, что оно имеет значение истина. Например:

```
pred1(1).  
pred2(3,"Start").  
pred3(computer(asus)).
```

Правилom же является конструкция, содержащая некоторые условия:

```
pred4(Arg1,Arg2,...,ArgN) if  
pred5(...) and pred6(...) and ... predN(...).
```

или:

```
pred4(Arg1,Arg2,...,ArgN) :- pred5(...), pred6(...), ... , predN(...).
```

где «:-» соответствует «if», а «,» соответствует «and».

Другими словами, правило – это связанное отношение. Правила позволяют Прологу логически выводить одну порцию информации из другой. Правило принимает значение «истина», если доказано, что заданный набор условий является истинным.

2.2. Раздел *predicates*

Если программист определяет в разделе *clauses* свой собственный предикат, то он **ДОЛЖЕН** объявить его в разделе *predicates*. В противном случае Visual Prolog не будет знать, о чем идет речь. Когда объявляется предикат, Прологу сообщается о том, к каким доменам принадлежат аргументы этого предиката.

Предикаты определяются фактами и правилами. В разделе *predicates* просто перечисляется каждый предикат с указанием доменов аргументов.

Имя предиката должно начинаться с буквы; после этой буквы могут следовать буквы, цифры и символы подчеркивания. Общий вид определения предиката:

```
pred(dom1,dom2,...,domN)
```

pred – имя предиката (имя отношения) (формально оно относится к типу symbol), dom – тип данных конкретного аргумента (всего аргументов в предикате N – это число аргументов предиката, его называют арностью предиката).

Например:

Predicates

run

sum(real,real,real).

parent(string,string).

student(string).

В этом же разделе можно задать тип детерминизма предиката, вставляя перед объявлением предиката ключевые слова *procedure*, *determ*, *failure* или *erroneous*. С другой стороны, можно определить недетерминированный предикат – вставляя перед его объявлением ключевые слова *nondeterm* или *multy*. Если предикат объявляется как детерминированный, то компилятор выдает предупреждение или ошибку, если найдет недетерминированные предложения для этого предиката. Режимом детерминизма для предикатов по умолчанию является *determ*.

- ***nondeterm***. Ключевое слово *nondeterm* определяет недетерминированные предикаты, которые могут совершать откаты назад и генерировать множественные решения. Предикаты, объявленные с ключевым словом *nondeterm*, могут быть неуспешны.
- ***procedure***. Ключевое слово *procedure* определяет предикаты, называемые *процедурами*, которые имеют всегда одно и только одно решение (но возможны ошибки во время исполнения). Процедуры всегда успешны и не порождают точек отката. Большинство из встроенных Visual Prolog предикатов внутренне объявлены как процедуры.
- ***determ***. Ключевое слово *determ* определяет детерминированные предикаты, которые могут быть успешными или неуспешными, но не могут порождать точек отката. Таким образом, предикат, объявленный с ключевым словом *determ*, имеет не более одного решения.
- ***multi***. Ключевое слово *multi* определяет недетерминированные предикаты, которые могут совершать откаты назад и генерировать множественные решения. Предикаты, объявленные с ключевым словом *multi*, не могут быть неуспешны.
- ***erroneous***. Предикат, объявленный с ключевым словом *erroneous*, всегда успешен и не порождает решения. Его обычно используют для управления ошибками. Visual Prolog предоставляет встроенные *erroneous* предикаты *exit* и *errorexit*.
- ***failure***. Предикат, объявленный с ключевым словом *failure*, не порождает решения, но может завершаться неуспешно. Visual Prolog предоставляет встроенный *failure* предикат *fail*. Неуспешные предикаты обычно используются для принуждения поиска с возвратом к ближайшей точке отката.

2.3. Раздел domains

Домены в Прологе подобны типам в Паскале. В программе Visual Prolog объекты в отношении (аргументы предиката) принадлежат доменам;

это могут быть домены стандартные или специальные, определяемые программистами.

Раздел *domains* служит двум очень важным целям. Во-первых, можно определить для доменов осмысленные имена, причем даже в том случае, если внутренне они совпадают с именами уже существующих доменов. Во-вторых, объявления специальных доменов используются для объявления структур данных, которые стандартными доменами не определяются.

Иногда целесообразно объявить домен тогда, когда возникает потребность более четкого выделения каких-либо частей раздела *predicates*. Объявление программистом своих собственных доменов помогает документировать предикаты, которые определяются путем задания в качестве типа аргумента удобного и понятного имени.

Например:

domains

selector = integer % тип selector для целых чисел

list_str = string* % список со строковыми данными

computer = name(string,list_sel,selector,integer) % описание структуры

2.4. Раздел *goal*

В Visual Prolog предусмотрен раздел *goal*, который должен включаться в программу.

Важно отметить то, что содержание раздела *goal* аналогично правилу и представляет собой список подцелей. Но между разделом *goal* и правилом есть два отличия:

1. После ключевого слова *goal* не следует знак *:-* (если).
2. При запуске программы на выполнение Visual Prolog отрабатывает цель автоматически.

Visual Prolog как бы вызывает цель (обращается к разделу *goal*), а программа выполняется, пытаясь удовлетворить тело целевого правила. Если достигаются все подцели раздела *goal*, то программа успешно завершается. Если же в процессе выполнения программы какая-либо подцель не достигается, то и программа заканчивает работу неудачно.

2.5. Другие разделы программы

2.5.1 Раздел *facts*

Программа на Visual Prolog представляет собой совокупность фактов и правил. Иногда в процессе выполнения программы может возникнуть потребность видоизменения (модификации, удаления или добавления) некоторых фактов, с которыми работает программа. В таком случае факты образуют **ДИНАМИЧЕСКУЮ** или **ВНУТРЕНнюю** базу данных (БД); она может изменяться в процессе выполнения программы. В Visual Prolog для объявления в программе фактов, которые должны стать частью динамической (или изменяющейся) базы данных, предусмотрен специальный раздел - *facts*.

Такой раздел базы данных объявляется с помощью ключевого слова *facts*, куда включаются объявления фактов, предназначенных для

организации динамической базы данных. В Visual Prolog имеется несколько встроенных предикатов, существенно облегчающих использование динамической БД.

2.5.2 Раздел **constants**

В программе на Visual Prolog можно объявить и использовать символические константы. Раздел объявления констант начинается ключевым словом **constants**, после которого следуют сами объявления с соблюдением следующего синтаксиса:

<Идентификатор> = <Макроопределение>

<Идентификатор> – это имя константы, а <Макроопределение> – это то, что этому имени соответствует. Каждое <Макроопределение> заканчивается символом новой строки, так что в одной строке может размещаться только одно описание константы. На объявленные таким образом константы можно затем ссылаться в программе.

Рассмотрим следующий пример:

constants

zero = 0

one = 1

two = 2

pi = 3.141592653

Перед компиляцией программы Visual Prolog заменит каждую константу действительной строкой, которую она представляет.

На использование констант накладываются следующие **ограничения**:

- определение константы не может ссылаться само на себя;
- в программе может быть несколько разделов constants, но константы должны объявляться до их использования;
- идентификаторы констант являются глобальными и могут объявляться только один раз. Несколько объявлений одного и того же идентификатора приведут к выдаче сообщения Constant identifier can only be declared once (Идентификатор константы может быть объявлен только один раз).

2.5.3 Разделы **global**

Visual Prolog позволяет объявить в программе некоторые домены, предикаты и выражения **ГЛОБАЛЬНЫМИ** (в отличие от **ЛОКАЛЬНЫХ**). Это можно сделать, сформировав в самом начале программы отдельные разделы **global domains**, **global predicates** и **global facts**.

3. Директивы компилятора. Директива **include**

Visual Prolog поддерживает несколько *директив компилятора*, которые можно добавлять в программу для сообщения компилятору специальных инструкций по обработке программы при ее компиляции.

Так, для того чтобы избежать многократного набора повторяющихся процедур, можно использовать директиву *include*.

Например:

1. Создается файл (например, MY.PRO), в котором объявляются наиболее часто используемые предикаты (с помощью разделов *domains* и *predicates*) и дается их описание в разделе *clauses*.

2. Пишется исходный текст программы, которая будет использовать эти процедуры.

3. В «допустимых областях» исходного текста программы размещается строка:

```
include "my.pro"
```

(«Допустимые области» – это любое место программы, в котором можно расположить декларацию разделов *domains*, *facts*, *predicates*, *clauses* и *goal*).

При компиляции исходных текстов программы Visual Prolog вставит содержание файла MY.PRO прямо в окончательный текст файла для компиляции.

Директиву *include* можно использовать для включения в исходный текст (практически любого) часто используемого фрагмента. Кроме того, любой включаемый в программу файл может, в свою очередь, включать другой файл (однако каждый файл может быть включен в программу только один раз).

4. Стандартные домены Visual Prolog

В Visual Prolog есть несколько встроенных стандартных доменов. Их можно использовать при декларации типов аргументов предикатов без описания в разделе *domains*. Основные стандартные домены приведены в таблицах 1 и 2.

Таблица 1. Основные стандартные домены


Домен	Описание и реализация		
short	Короткое, знаковое, количественное.		
	All platforms	16 bits, 2s comp	32768 .. 32767
ushort	Короткое, беззнаковое, количественное.		
	All platforms	16 bits	0 .. 65535
long	Длинное, знаковое, количественное.		
	All platforms	32 bits, 2s comp	-2147483648 .. 2147483647
ulong	Длинное, беззнаковое, количественное.		
	All platforms	32 bits	0 .. 4294967295
integer	Знаковое, количественное, имеет платформу-зависимый размер		
	16bit platforms	16 bits, 2s comp	-32768 .. 32767
	32bit platforms	32 bits, 2s comp	-2147483648 .. 2147483647
unsigned	Беззнаковое, количественное, имеет платформу-зависимый размер		
	16bit platforms	16 bits	0 .. 65535
	32bit platforms	32 bits	0 .. 4294967295
byte			
	All platforms	³ 8 bits	0 .. 255
word			
	All platforms	16 bits	0 .. 65535
dword			
	All platforms	32 bits	0 .. 4294967295

Таблица 2. Основные стандартные домены

Домен	Описание и реализация
char	Символ, реализуемый как беззнаковый byte. Синтаксически это символ, заключенный между двумя одиночными кавычками
real	<p>Число с плавающей запятой, реализуемое как 8 байт в соответствии с соглашением IEEE; эквивалентен типу double в C. Синтаксически числа с необязательным знаком (+ или -), за которым следует несколько цифр DDDDDDD, затем необязательная десятичная точка (.) и ещё цифры DDDDDDD, за которыми идет необязательная экспоненциальная часть (e(+ или -)DDD):</p> <p><+ -> DDDDD <.> DDDDDDD <e <+ -> DDD></p> <p>Примеры действительных чисел (real):</p> <p>42705 9999 86.72</p> <p>9111.929437521e238 79.83e+21</p> <p>Здесь 79.83e+21 означает 79.83 x 10²¹, как и в других языках.</p> <p>Допустимый диапазон чисел: 1*10⁻³⁰⁷ to 1*10³⁰⁸ (от 1e-307 до 1e+308). При необходимости, целые автоматически преобразуются в real.</p>
string	<p>Последовательность символов, реализуемых как указатель на байтовый массив, завершаемый нулем, как в C. Для строк допускается два формата:</p> <ol style="list-style-type: none"> 1. Последовательность букв, цифр и символов подчеркивания, причем первый символ должен быть строчной буквой. 2. Последовательность символов, заключенных в двойные кавычки. <p>Примеры строк:</p> <p>telephone_number "railway ticket" "Dorid Inc"</p> <p>Строки, которые находятся в программе, могут достигать длины в 255 символов, в то время как строки, которые система Visual Prolog считывает из файла или строит внутри себя, могут достигать (теоретически) до 4 Гбайт на 32-битных платформах.</p>
symbol	Последовательность символов, реализуемых как указатель на вход в таблице идентификаторов, хранящей строки идентификаторов. Синтаксис – как для строк.

Практические задания и методические указания по их выполнению

1. Создайте новый документ Visual Prolog:


- нажмите  на панели инструментов либо выберите пункт главного меню File-New.

В результате появится окно, изображенное на рис.2. Наберите в появившемся окне следующий текст:

Goal

write(“ Hello world! ”), nl.

(Предикат **nl** переводит курсор в начало следующей строки при выводе результатов).

Для проверки работы программы следует выбрать пункт меню Project-Test Goal или нажать кнопку  на панели инструментов. Сразу же появится окно рис.5, если не были допущены ошибки, иначе вы увидите окно рис.4. Ответ системы «Yes» после выдачи основных результатов означает, что поставленная цель в разделе Goal была успешно удовлетворена, в противном случае Visual Prolog выдаст результат «No». Такие ответы связаны, прежде всего, с тем, что в основе Visual Prolog находится язык логики предикатов первого порядка, который, как известно, работает с логическими значениями предикатных предложений такими как «True» или «False» («Истина» или «Ложь» соответственно).

2. Наберите в окне редактора следующую программу:

```
predicates  
father(symbol,symbol)  
mother(symbol,symbol)
```

```
clauses  
father(nikolai,ivan).  
mother(nina,ivan).
```

```
goal  
father(X,ivan), nl,  
mother(Y,ivan), nl.
```

Запустите на выполнение и проанализируйте полученные результаты.

3. Напишите программу, содержащую набор фактов:

```
родители(николай,нина,иван).  
родители(петр,галина,андрей).  
родители(виктор,надежда,мария).
```

Каждый из фактов трактуется таким образом. Первые два аргумента предиката являются родителями лица, которое определено третьим аргументом. Получите результаты следующих запросов:

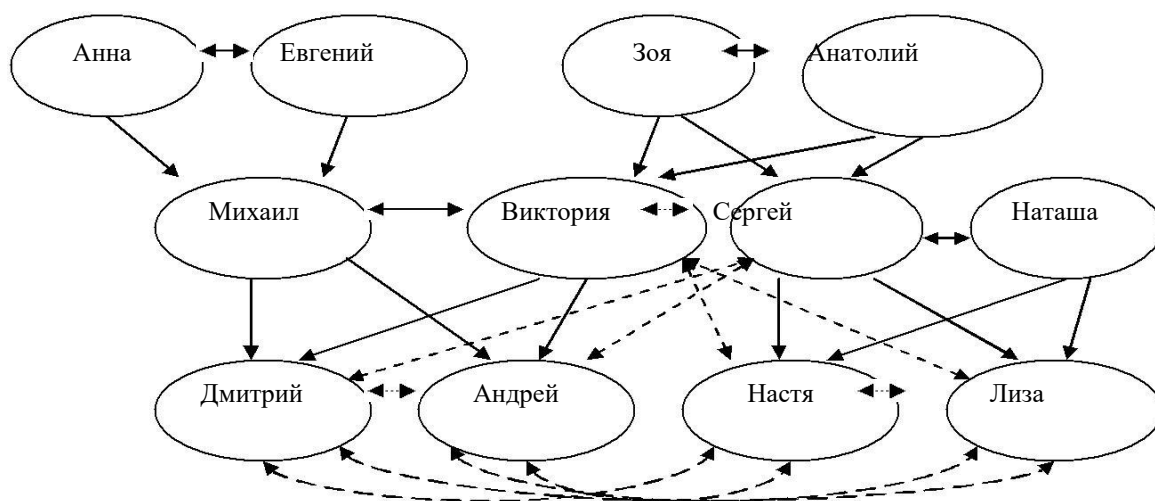
Goal: родители(X,Y,_);

Goal: родители(_,_,X).

4. Напишите программу, описывающую дерево Ваших родственных отношений и выполняющую различные запросы.

Пример выполнения задания

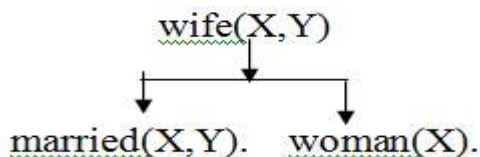
Задан следующий граф родственных отношений:



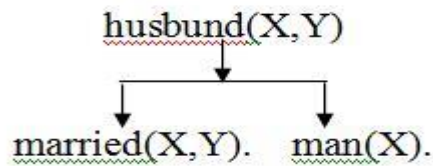
Условное обозначение	Отношение
↔	состоит в браке
↓	вертикальная стрелка – родитель, ребенок, сын/дочь
↔---	дядя/тетя, племянник/племянница
↔---	брат/сестра
↔---	- двоюродный брат/сестра

Представление программы в виде дерева

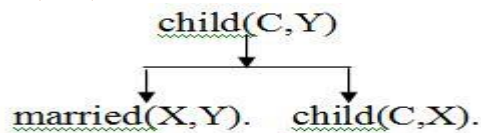
А) правило «жена»:
wife(X,Y):-
married(X,Y),woman(X).



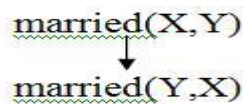
Б) правило «муж»:
 husband(X,Y):-
 married(X,Y),man(X).



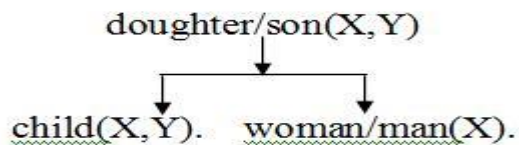
В) правило «ребенок»:
 child(C,Y):-married(X,Y),child(C,X).



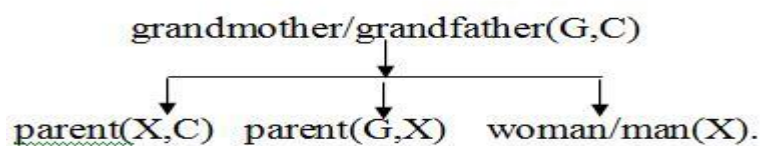
Г) правило «состоит в браке»:
 married(X,Y):-
 married(Y,X).



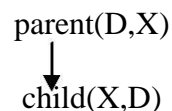
Д,Е) правило «дочь/сын»:
 doughter(X,Y):-
 child(X,Y),woman(X). son(X,Y):-
 child(X,Y),man(X).



Ж,З) правило «бабушка/дедушка»:
 grandmother(G,C):-parent(X,C),parent(G,X),woman(G).
 grandfather(G,C):-parent(X,C),parent(G,X),man(G).

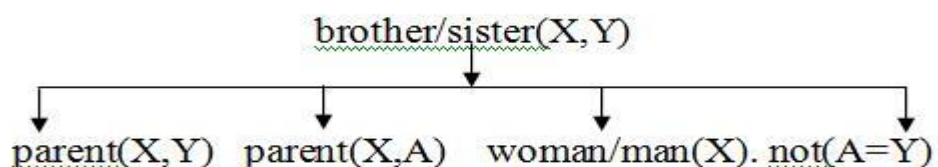


И) правило
 «родитель»: parent(D,X):-
 child(X,D).



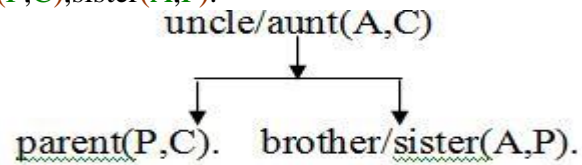
К,Л) правило «брат/сестра»:

brother(A,Y):-parent(X,Y),parent(X,A),man(A),
 not(A=Y). sister(A,Y):-
 parent(X,Y),parent(X,A),woman(A) not(A=Y).



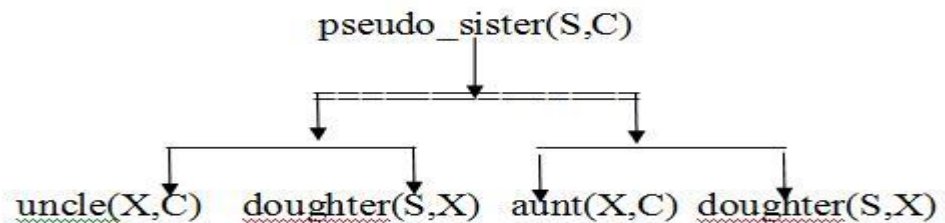
М,Н) правило «дядя/тетя»:

```
uncle(A,C):-  
parent(P,C),brother(A,P).  
aunt(A,C):-  
parent(P,C),sister(A,P).
```



О) правило «двоюродная сестра»: pseudo_sister(S,C):-

```
uncle(X,C),daughter(S,X).  
pseudo_sister(S,C):-  
aunt(X,C),daughter(S,X).
```



Пример программы с конъюнкцией нескольких вопросов: Миша является родителем Дмитрия И Андрей является братом Дмитрия И Анна является бабушкой Андрея И Евгений является дедушкой Дмитрия:

```

predicates
nondeterm man(symbol)
nondeterm woman(symbol)
nondeterm wife(symbol,symbol)
nondeterm husband(symbol,symbol)
nondeterm married(symbol,symbol)
nondeterm child(symbol,symbol)
nondeterm daughter(symbol,symbol)
nondeterm son(symbol,symbol)
nondeterm grandmother(symbol,symbol)
nondeterm grandfather(symbol,symbol)
nondeterm parent(symbol,symbol)
nondeterm brother(symbol,symbol)
nondeterm sister(symbol,symbol)
nondeterm uncle(symbol,symbol)
nondeterm aunt(symbol,symbol)
nondeterm pseudo_sister(symbol,symbol)

clauses
man(evgeniy).man(misha).man(dmitry).man(andrey).man(anatoly).man(sergey).
woman(anna).woman(viktoria).woman(zoia).woman(natasha).woman(nastia).woman(liza).
married(anna,evgeniy).married(viktoria,misha).married(zoia,anatoly).married(sergey,natasha).
child(dmitry,viktoria).child(andrey,viktoria).child(misha,anna).child(sergey,zoia). child(viktoria,zoia).
child(nastia,natasha). child(liza,natasha).child(C,Y):-married(X,Y),child(C,X).
wife(X,Y):-married(X,Y),woman(X).
husband(X,Y):-married(X,Y),man(X).
daughter(X,Y):-child(X,Y),woman(X).
son(X,Y):-child(X,Y),man(X).
grandmother(G,C):-parent(X,C),parent(G,X),woman(G).
grandfather(G,C):-parent(X,C),parent(G,X),man(G).
parent(D,X):-child(X,D).
brother(A,Y):-parent(F,Y),parent(F,A),man(A).
sister(X,Y):-child(X,F),child(Y,F),woman(X).
uncle(U,C):-parent(F,C),brother(U,F).
aunt(A,C):-child(C,F),sister(A,F).
pseudo_sister(S,C):-uncle(X,C),daughter(S,X). pseudo_sister(S,C):-aunt(X,C),daughter(S,X).

goal
parent(misha,dmitry), brother(andrey,dmitry), grandmother(anna,andrey), grandfather(evgeniy,dmitry).

```

В результате работы программы получим результат - Yes, т.е. Миша является родителем Дмитрия И Андрей является братом Дмитрия И Анна является бабушкой Андрея И Евгений является дедушкой Дмитрия.

Опишите предикат, позволяющий определить братьев и сестер. Выведите результат на экран.