

Логическое программирование



Урок 5

Курс “Парадигмы программирования и языки парадигм”






Цели семинара

-  Понять основные отличия между **логической парадигмой** и уже известными парадигмами
-  Научиться решать задачи в рамках **логической парадигмы**



План семинара


 Синтаксис Пролога

 Смотрим код

- 3 задачи

 Пишем код

- 3 задачи

 Подведение итогов



Синтаксис Пролога





Шпаргалка

- 📌 `predicate_name(X)` — предикат `predicate_name` истинен тогда, когда истинен `X`.
- 📌 `predicate_name(X, Y)` — предикат `predicate_name` истинен тогда, когда истинны и `X` и `Y` одновременно.
- 📌 С маленькой буквы пишутся значения (имена собственные).
- 📌 С большой буквы пишутся переменные (объекты, принимающие несколько значений).
- 📌 `[X|H]` — вертикальная черта означает разделение на “первую переменную `X`” и “`H` – всего что после `X`”.
- 📌 `[_|H]` — нижнее подчеркивание означает “любая переменная”, без имени.
- 📌 `reverse(List1, List2)` - встроенный метод, возвращающий `true`, когда список “развернутых” значений `List2` полностью совпадает с `List1`.
- 📌 В конце *высказывания* ставится точка. В начале *запроса* пишется знак вопроса и минус “`?-`”.

Больше информации по синтаксису языка Prolog - <https://www.swi-prolog.org>



Смотрим код





Регламент

1

Вы читаете код

2

Вы объясняете:

- Суть программы общими словами
- Построчный анализ кода
- Результат исполнения этой программы

3

Вместе обсуждаем решение



Палиндром

```
1 % Rules
2 palindrome(List) :-
3     reverse(List, List).
4
5 % Query 1
6 palindrome([1,2,3,2,1]).
7
8 % Query 2
9 palindrome([10, 35, 40]).
```

Суть фрагмента: ?

Построчно: ?

Результат исполнения: ?



Палиндром

```
1 % Rules
2 palindrome(List) :-
3     reverse(List, List).
4
5 % Query 1
6 palindrome([1,2,3,2,1]).
7
8 % Query 2
9 palindrome([10, 35, 40]).
```

Суть фрагмента: проверка, является ли список *List* палиндромом.

Построчно: в данном случае встроенный метод *reverse* - истинен, когда *List* “развернутой” версией себя.

Результат исполнения: [1,2,3,2,1] - true (является палиндромом), [10, 35, 40] - false (не является палиндромом)



Поиск: логический

```
1 % Rules
2 member(X, [X|_]).
3 member(X, [_|T]) :-
4     member(X, T).
5
6 % Query
7 ?- member(0, [1,2,3])
```

Суть фрагмента: ?

Построчно: ?

Результат исполнения: ?



Поиск: логический

```
1 % Rules
2 member(X, [X|_]).
3 member(X, [_|T]) :-
4     member(X, T).
5
6 % Query
7 ?- member(0, [1,2,3])
```

Суть фрагмента: проверка наличия элемента **X** в списке.

Построчно: предикат `member` истинен, если

- элемент **X** является первым элементом списка
- элемент **X** не является первым элементом списка и `member(X, T)` истинен для оставшейся части списка **T**

Результат исполнения: `false`, поскольку 0 не содержится в списке `[1,2,3]`.



len

```
1 % Rules
2 len([], 0).
3 len([_|T], Len) :-
4     len(T, Len1), Len is Len1 + 1.
5
6 % Query
7 len([1,2,3,4,5,6,7,8], Len)
```

Суть фрагмента: ?

Построчно: ?

Результат исполнения: ?



len

```
1 % Rules
2 len([], 0).
3 len([_|T], Len) :-
4     len(T, Len1), Len is Len1 + 1.
5
6 % Query
7 len([1,2,3,4,5,6,7,8], Len)
```

Суть фрагмента: вычисление длины списка.

Построчно: Для пустого списка - результат равен 0. Для непустых списков, **len** истинен для таких значений **Len**, для которых одновременно истинен **len(T, Len1)** и **Len** равен **Len1** + 1. Таким образом, решающий момент в рекурсии наступает, когда длина списка T равна 1, и Len1, соответственно равен 0.

Результат исполнения: 8, поскольку вычисляется длина списка [1,2,3,4,5,6,7,8].



Вопросы



Пишем код





Регламент

- 1 Вместе читаем условия задачи
- 2 Вы самостоятельно решаете задачу
- 3 Вместе обсуждаем решение



Генеалогия: практика



Генеалогия: практика

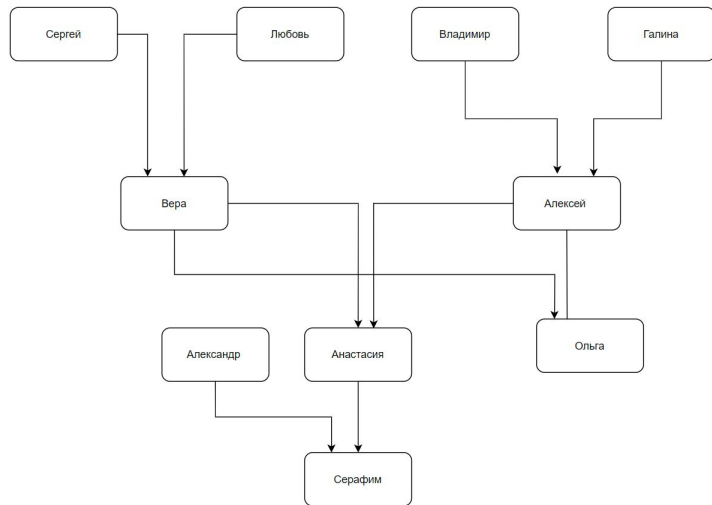
- **Контекст**

Продолжаем исследовать генеалогию с помощью логического программирования. Подобный пример мы уже разбирали на лекции. На этот раз ваша задача реализовать генеалогическое древо на языке Prolog самостоятельно.

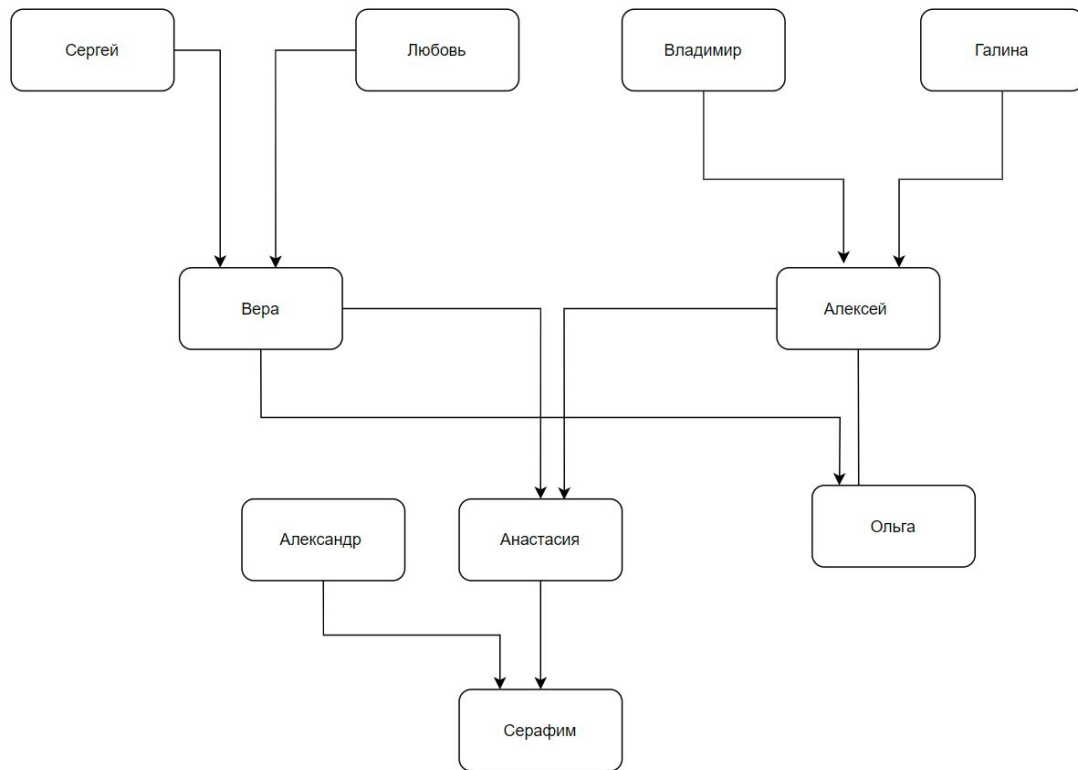
- **Ваша задача**

Реализовать изображённое генеалогическое древо на языке Prolog таким образом, чтобы про Алексея и Анастасию можно было узнать кто их: мать, отец, дедушки, бабушки, дети.

- Решение.. ?



Генеалогия: практика





Генеалогия: практика

- Решение:

```
1 female(lubov).
2 female(galina).
3 female(vera).
4 female(anastasiya).
5 female(olga).
6 male(sergey).
7 male(vladimir).
8 male(alexey).
9 male(alexander).
10 male(serafim).
11 parent(vera, lubov).
12 parent(vera, sergey).
13 parent(alexey, galina).
14 parent(alexey, vladimir).
15 parent(anastasiya, vera).
16 parent(anastasiya, alexey).
17 parent(olga, vera).
18 parent(olga, alexey).
19 parent(serafim, anastasiya).
20 parent(serafim, alexander).
21
22 father(X,Y):- parent(X,Y), male(Y).
23 mother(X,Y):- parent(X,Y), female(Y).
24 grandfather(X, Z):- parent(X, Y), father(Y, Z).
25 grandmother(X, Z):- parent(X, Y), mother(Y, Z).
26
27 child(X, Y):- parent(Y, X).
```



Максимальный элемент: логический



Максимальный элемент: логический

- **Контекст**
Данную задачу мы уже видели и решали с помощью императивной и декларативной парадигмы. Теперь давайте напишем её в логическом стиле.
- **Ваша задача**
Написать предикат на языке Prolog, который для входного списка возвращает максимальный элемент в этом списке.
- **Решение.. ?**



Максимальный элемент: логический

- **Контекст**

Данную задачу мы уже видели и решали с помощью императивной и декларативной парадигмы. Теперь давайте напишем её в логическом стиле.

- **Ваша задача**

Написать предикат на языке Prolog, который для входного списка возвращает максимальный элемент в этом списке.

- **Решение:**

```
1 % Rules
2 max_list([X], X).
3 max_list([H|T], Max) :-
4     max_list(T, Max1),
5     (H > Max1, Max is H; Max is Max1).
6
7 % Query
8 ?- max_list([1,2,3,4], X).
```



Удаление дубликатов



Удаление дубликатов

- **Контекст**
В предыдущем семинаре мы уже обсуждали важность выявления и удаления дубликатов в данных. В этот раз ваша задача реализовать именно удаление из списка с помощью логического программирования.
- **Ваша задача**
Реализовать логическую программу (предикат), которая принимает на вход массив и возвращает его версию без дубликатов.
- **Решение.. ?**



Удаление дубликатов

- **Контекст**

В предыдущем семинаре мы уже обсуждали важность выявления и удаления дубликатов в данных. В этот раз ваша задача реализовать именно удаление из списка с помощью логического программирования.

- **Ваша задача**

Реализовать логическую программу (предикат), которая принимает на вход массив и возвращает его версию без дубликатов.

- **Решение:**

```
1 % Rules
2 remove_duplicates([], []).
3 remove_duplicates([H|T], Result) :-
4     member(H, T),
5     remove_duplicates(T, Result).
6 remove_duplicates([H|T], [H|Result]) :-
7     \+ member(H, T),
8     remove_duplicates(T, Result).
9
10 % Query
11 ?- remove_duplicates([1,2,3,1,1], Result).
```



Итоги семинара





Итоги семинара



Смотрим код

- Проанализировали 3 скрипта на языке Prolog



Пишем код

- Решили 3 задачи по программированию



Подвели итоги



Домашнее задание

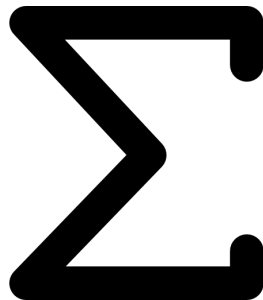




Сумма элементов списка

- **Контекст**
Мы уже видели множество решений этой задачи в различных стилях. Пришло время решить её с помощью логической парадигмы.
- **Ваша задача**
Написать программу на языке *Prolog* для вычисления суммы элементов списка. На вход подаётся целочисленный массив. На выходе - сумма элементов массива.
- Пример на языке Python в функциональной парадигме:

```
1 arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 sum(arr)
```





Конец семинара
Спасибо за внимание!

