МИНОБРНАУКИ РОССИИ

**«Логическое программирование на языке Prolog»**

Лабораторная работа №5

по предмету «Программирование»

Вариант №2

Работу выполнил: студент группы. \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_(ФИО)

Город

2023

Задания

1. Определите предикат my\_flatten(NestedList, FlattenedList), "расплющивающий" вложенный список NestedList.

?- my\_flatten([a, [[b], c], [[d]]], X). => X = [a, b, c, d]

Решение. Используем предикат list из lists.pl

%проверка, что параметр - список

list([]).

list([H|T]) :- list(T).

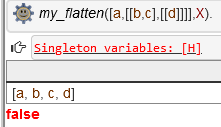
my\_flatten([], []). %пустой не меняется

my\_flatten([X|Y], Z):-list(X),my\_flatten(X,X1),

my\_flatten(Y,Y1),append(X1,Y1,Z).

my\_flatten([X|Y], [X|Y1]):-not(list(X)),my\_flatten(Y,Y1).

Проверим для запроса ? my\_flatten([a,[[b,c],[[d]]],X).

****

2. Код Грея для n бит -- список всех возможных списков 0 и 1 длины n, два соседних списка в котором отличаются только в одном месте.

Решение. Используем вспомогательный предикат add\_first.

%добавляет N ко всем спискам, заданным 2-м параметром

add\_first(\_,[],[]).

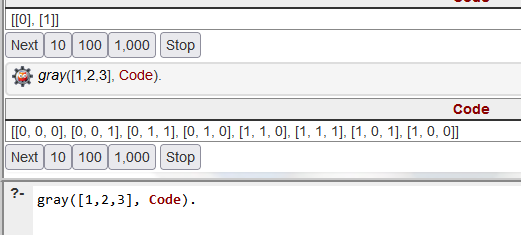
add\_first(N,[X|R],[[N|X]|R1]):-add\_first(N,R,R1).

%код Грея

gray([\_], [[0],[1]]).

gray([\_|R], C):-gray(R,A),reverse(A,B),add\_first(0,A,A1),

add\_first(1,B,B1),append(A1,B1,C).

****

1.1. Постройте деревья вызова для запросов:

?предок\_потомок(алексей, сергей)

Правила :

1) предок\_потомок(Предок, Потомок) :-

родитель\_ребёнок(Предок, Потомок).

2) предок\_потомок(Предок, Потомок) :-

родитель\_ребёнок(Предок, Предок1),

предок\_потомок(Предок1, Потомок).

успех

?предок\_потомок(алексей, сергей)

успех

2. Предок=алексей,

Потомок=сергей

неудача

1. Предок=алексей,

Потомок=сергей

родитель\_ребёнок(настя, сергей).

родитель\_ребёнок(алексей, сергей).

родитель\_ребёнок(алексей, Предок1),

предок\_потомок(настя, сергей).

Предок1=Настя

успех

1. Предок=настя,

Потомок=сергей

родитель\_ребёнок(алексей, настя)

1.2. Постройте деревья вызова для запросов:

my\_member(List, 1).

успех, List=[\_?3,\_?2,1|\_?1]

|  |
| --- |
| 1)my\_member([H|T], H). |
|  |

2) my\_member([\_|T], X) :- my\_member(T, X).

успех, List=[\_?2, 1|\_?1]

?my\_member(List, 1).

1) T=[1|T3], H=1

успех, List=[\_?2,1|\_?1]

1) T=[1|T1], H=1

2) List=[\_|T], Х=1

1) List=[1|T], H=1

…

успех, T=[1|\_?1]

дальше

успех, List=[1|\_?]

успех, List=[1|\_?]

my\_member([1|T3], 1).

my\_member([1|T1], 1).

my\_member(Т, 1).

my\_member([1|T], 1).

дальше

T=[\_,1|\_?3]

2) T=[\_|T2], Х=1

my\_member(T, 1).

успех, T=[1|\_?3]

В результате вызова my\_member(List, 1) будут последовательно выдаваться результаты:

List=[1|\_?],

List=[\_?1, 1|\_?],

List=[\_?1,\_?2, 1|\_?],

List=[\_?1,\_?2, \_?3,1|\_?],

… до бесконечности (до исчерпания ресурсов компьютера).

где \_?x –произвольные неопределенные (несвязанные) переменные.

2. Задайте предикат ordered(List), который успешен, если элементы List - числа, расположенные в порядке возрастания:

?- ordered([1,2,3]) => Yes

?- ordered([3,2]) => No

?- ordered([1,a]) => No

**Решение**.

%пустой упорядочен

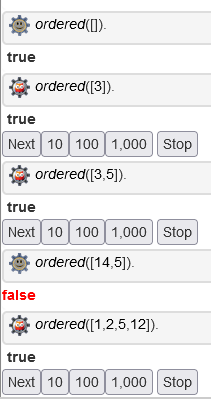
ordered([]).

%один элемент упорядочен

ordered([X]):-number(X).

ordered([X,Y|T]):-number(X),number(Y),X=<Y,ordered([Y|T]).

Без предиката number запрос ordered([1,a]) выдает не false, а ошибку «Arithmetic: `a/0' is not a function».



3. Задайте предикат sublist(List1, List2), успешный, если List2 - отрезок List1.

Воспользуемся предикатом my\_append из примеров lists.pl.

Получим:

my\_append([], List, List).

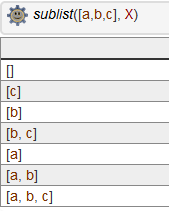
my\_append([H|T1], List2, [H|TResult]) :- my\_append(T1, List2, TResult).

sublist([],[]).

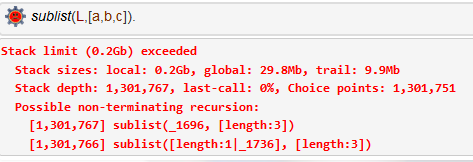
sublist([\_|T],L):-sublist(T,L).

sublist([H|T],[H|L]):-my\_append(L,\_,T).

Проверим для запроса ?sublist([a,b,c],X).



Проверим поведение при запросе ?sublist(L,[a,b,c]).



Произошла ошибка – исчерпание ресурсов компьютера (стека). Это объясняется тем, что существует бесконечно много списков L, в которых был бы отрезок [a,b,c].

**Lists2.pl**

%проверка, что параметр - список

list([]).

list([H|T]) :- list(T).

%предикат "расплющивающий" вложенный список

my\_flatten([], []). %пустой не меняется

my\_flatten([X|Y], Z):-list(X),my\_flatten(X,X1),

my\_flatten(Y,Y1),append(X1,Y1,Z).

my\_flatten([X|Y], [X|Y1]):-not(list(X)),my\_flatten(Y,Y1).

%добавляет N ко всем спискам, заданным 2-м параметром

add\_first(\_,[],[]).

add\_first(N,[X|R],[[N|X]|R1]):-add\_first(N,R,R1).

%код Грея

gray([\_], [[0],[1]]).

gray([\_|R], C):-gray(R,A),reverse(A,B),add\_first(0,A,A1),

add\_first(1,B,B1),append(A1,B1,C).

%ordered(L) - упорядочены ли числа в списке L по возрастанию (неубыванию).

%пустой упорядочен

ordered([]).

%один элемент упорядочен

ordered([X]):-number(X).

ordered([X,Y|T]):-number(X),number(Y),X=<Y,ordered([Y|T]).

%соединение двух списков

my\_append([], List, List).

my\_append([H|T1], List2, [H|TResult]) :- my\_append(T1, List2, TResult).

%sublist(L,X). - X отрезок списка L

sublist([],[]).

sublist([\_|T],L):-sublist(T,L).

sublist([H|T],[H|L]):-my\_append(L,\_,T).