МИНОБРНАУКИ РОССИИ

**«Логическое программирование на языке Prolog»**

Лабораторная работа №7

по предмету «Программирование»

Вариант №2

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

Город

2023

Задания

1. Чтобы разобраться с различиями операторов =, ==, =:= и is, для каждого запроса сначала попытайтесь понять, как должен ответить Пролог, а потом проверьте, правильны ли ваши предположения.

?- X = 1+2. %унификация термов, в результате X = 1+2, но не Х=3

?- 3 = 1+2. % результат false, термы 3 и 1+2 не унифицируются

?- 2+1 = 1+2. % false, термы не унифицируются

?- X == 1+2. % идентичность, результат false (переменная идентична, переменной, связанной с тем же.

?- 3 == 1+2. %false константа и выражение (структура) не идентичны

?- 2+1 == 1+2. % false, не идентичны

?- X =:= 1+2. %численно равны (оба выражения должны быть известны), результат – ошибка, т.к. Х не имеет численного значения

?- 3 =:= 1+2. %true

?- 2+1 =:= 1+2. %true

?- X is 1+2. %переменная связывается с выражением, или проверяется равенство числа и значения выражения.

?- 3 is 1+2. %true, число совпадает со значением выражения

?- 2+1 is 1+2. %false, слева не число

5. Определите предикат prime\_factors(Num, Factors), находящий все простые делители числа Num и их кратность.

Решение.

%число 1 не имеет простых делителей

prime\_factors(1,L,L).

% добавить очередной делитель

prime\_factors(N, [factor(M,K)|T],F):- N>1,N mod M =:=0,

K1 is K+1, N1 is N div M,

prime\_factors(N1, [factor(M,K1)|T],F).

prime\_factors(N, [factor(M,0)|T],F):- N>1,N mod M =\=0,

M1 is M+1, prime\_factors(N, [factor(M1,0)|T],F).

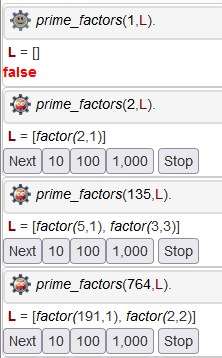
prime\_factors(N, [factor(M,K)|T],F):- N>1,K>0,N mod M =\=0,

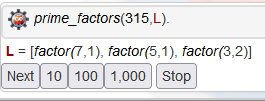
M1 is M+1,prime\_factors(N, [factor(M1,0),factor(M,K)|T],F).

prime\_factors(1, []).

prime\_factors(Num, Factors):-Num>1,

prime\_factors(Num,[factor(2,0)],Factors).

****

****

Но если обязательно нужно, чтобы множители в списке шли в порядке возрастания, то дополнительно применяем предикат naive\_rev из lists.pl ЛР 6. Тогда программа выглядит так:

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

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

my\_append(T1, List2, TResult).

naive\_rev([], []).

naive\_rev([H|T], Rev) :- naive\_rev(T, RevT),

my\_append(RevT, [H], Rev).

%число 1 не имеет простых делителей

prime\_factors(1,L,L).

% добавить очередной делитель

prime\_factors(N, [factor(M,K)|T],F):- N>1,N mod M =:=0,K1 is K+1,

N1 is N div M,

prime\_factors(N1, [factor(M,K1)|T],F).

prime\_factors(N, [factor(M,0)|T],F):- N>1,N mod M =\=0, M1 is M+1,

prime\_factors(N, [factor(M1,0)|T],F).

prime\_factors(N, [factor(M,K)|T],F):- N>1,K>0,N mod M =\=0, M1 is M+1,

prime\_factors(N, [factor(M1,0),factor(M,K)|T],F).

prime\_factors(1, []).

prime\_factors(Num, Factors):-Num>1,prime\_factors(Num,[factor(2,0)],F),

naive\_rev(F, Factors).

2. Определите предикат polynomize(Expr, Poly), который выполняется, если Expr -- арифметическое выражение (включающее операции +, \*, ^ в константную степень), а Poly -- многочлен в нормальной форме (см. 1 вариант 3 задание), получающийся в результате его упрощения.

?- polynomize((x + x)\*(x + 1), Poly).

Poly = 2\*x^2 + 2\*x

Решение.

Листинг программы

%длина списка

len([],0).

len([\_|T],N) :- len(T,N1), N is N1+1.

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

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

my\_append(T1, List2, TResult).

naive\_rev([], []).

naive\_rev([H|T], Rev) :- naive\_rev(T, RevT),

my\_append(RevT, [H], Rev).

%суммирование списков

sum\_list([],L2,L2).

sum\_list([H|T],[],[H|T]).

sum\_list([H1|T1],[H2|T2],[H|T]):-H is H1+H2,sum\_list(T1,T2,T).

%коэффициент при степени N

get\_coef([],\_,0).

get\_coef([A|\_],0,A).

get\_coef([\_|T],N,A) :-N>0, N1 is N-1, get\_coef(T,N1,A).

%ппоизведение кэффициентов

pcoef(P1,P2,I,J,C) :- get\_coef(P1,I,A), get\_coef(P2,J,B), C is A\*B.

%новые коэффициенты

newcoef(\_,\_,N,I,0) :- I>N.

newcoef(P1,P2,N,I,S) :- I=<N, J is N-I, pcoef(P1,P2,I,J,C), I1 is I+1,

newcoef(P1,P2,N,I1,S1), S is S1+C.

%произведение полиномов, заданных списками

poly\_mult(P1,P2,I,[]) :- len(P1,N1), len(P2,N2), N is N1+N2-2, I>N.

poly\_mult(P1,P2,I,[Q|T]) :- len(P1,N1), len(P2,N2), N is N1+N2-1,I<N,newcoef(P1,P2,I,0,Q), I1 is I+1, poly\_mult(P1,P2,I1,T).

polymult(P1,P2,R) :- poly\_mult(P1,P2,0,R).

%выражение в список коэффициентов

poly2list(X, [0,1]):-atom(X).%переменная

poly2list(X, [X]):-number(X).%число

poly2list(X+Y, L):-poly2list(X,L1),poly2list(Y,L2),sum\_list(L1,L2,L).

poly2list(X\*Y, L):-poly2list(X,L1),poly2list(Y,L2),polymult(L1,L2,L).

poly2list(X^1, L):-poly2list(X,L).

poly2list(X^N, L):-N mod 2 =:= 0, N1 is N div 2,

poly2list((X\*X)^N1, L).

poly2list(X^N, L):-N>1,N mod 2 =\= 0, N1 is N - 1,

poly2list(X\*X^N1, L).

%список коэффициентов в выражение

list2polinom([H],0,H).

list2polinom([0,H|T],I,X):-I1 is I-1,list2polinom([H|T],I1,X).

list2polinom([1|T],1,x+X):-list2polinom(T,0,X),X\=0.

list2polinom([1|T],I,x^I+X):-I>1,I1 is I-1,list2polinom(T,I1,X),X\=0.

list2polinom([H|T],1,H\*x+X):-H>0,H=\=1,list2polinom(T,0,X),X\=0.

list2polinom([H|T],I,H\*x^I+X):-H>0,H=\=1,I>1,

I1 is I-1,list2polinom(T,I1,X),X\=0.

%если ноль

list2polinom([1|T],1,x):-list2polinom(T,0,0).

list2polinom([1|T],I,x^I):-I>1,I1 is I-1,list2polinom(T,I1,0).

list2polinom([H|T],1,H\*x):-H>0,H=\=1,list2polinom(T,0,0).

list2polinom([H|T],I,H\*x^I):-H>0,H=\=1,I>1,

I1 is I-1,list2polinom(T,I1,0).

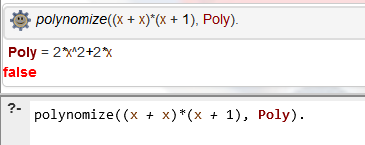
%полином в нормальный вид

polynomize(N,N):-number(N).

polynomize(X,Poly):-not(number(X)),poly2list(X,L),naive\_rev(L,R), len(R,K), K1 is K-1,list2polinom(R,K1,Poly).

Алгоритм работы

Сначала при помощи poly2list преобразуем выражение в список коэффициентов полинома, затем при помощи list2polinom произведем обратное преобразование в многочлен в нормальном виде. При этом нужно учесть множитель 1, слагаемое 0 и возведение в степень 1.

****

1. Определите предикат fib(N, NthFib), который выполняется, если NthFib -- N-ное число Фибоначчи (начиная с 1).

?- fib(3, X).

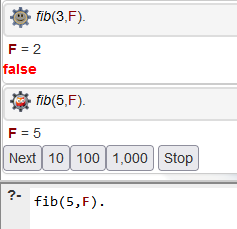
X = 2

**Решение**.

fib(1,1).

fib(2,1).

fib(N,F):-N>2,N1 is N-1, N2 is N-2,fib(N1,F1),fib(N2,F2),F is F1+F2.



1. Определите предикат mean(List, Mean), который выполняется, если Mean -- арифметическое среднее списка List.

?- mean([1,2,3], Mean).

Mean = 2.0

**Решение**.

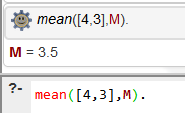
%сумма и длина

sum\_len([], 0,0).

sum\_len([H|T], S,L):-sum\_len(T, S1,L1), L is L1+1, S is S1+H.

%среднее арифметическое

mean(List,Mean):-sum\_len(List,S,L),Mean is S / L.



1. В числе операторов, определённых в Prolog, есть /\ (и), \/ (или), + (не) и -> (следует). Определите предикат dnf(Formula), который выполняется, если Formula - формула в дизъюнктивной нормальной форме (т.е. дизъюнкция ("или") конъюнкций переменных и их отрицаний, в качестве переменных используем атомы).

?- dnf((x /\ (\+ y) /\ z) \/ ((\+ x) /\ z)).

Yes

?- dnf((x \/ (\+ y)) /\ (z \/ (\+ x) \/ y)).

No

**Решение**.

%ДНФ

%переменная или отрицание

dnfvar(X):-atom(X),not(number(X)).

dnfvar(\+X):-atom(X),not(number(X)).

%конъюнкция

dnfand(X/\Y):-dnfand(X),dnfvar(Y).

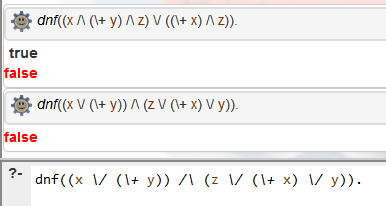
dnfand(X):-dnfvar(X).

%дизъюнкция

dnfor(X\/Y):-dnfor(X),dnfand(Y).

dnfor(X):-dnfand(X).

dnf(X):-dnfor(X).



1. Определите предикат eval\_logic(Formula, Values), который выполняется, если Formula -- логическая формула, Values -- список термов вида true(V) и false(V), где V -- атомы, которые используются в Formula, и в результате подстановки этих значений формула становится истинной.

?- eval\_logic(\+ (x -> (\+ y)), [true(x), false(y)]).

No

**Решение**.

my\_member([H|\_], H).

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

%логическая формула

eval\_logic(X, V):- atom(X),my\_member(V,true(X)).

eval\_logic(\+X, V):- not(eval\_logic(X,V)).

eval\_logic(X \/ \_, V):- eval\_logic(X,V).

eval\_logic(X \/ Y, V):- not(eval\_logic(X,V)),eval\_logic(Y,V).

eval\_logic(X /\ Y, V):- eval\_logic(X,V),eval\_logic(Y,V).

eval\_logic(X -> \_, V):- not(eval\_logic(X,V)).

eval\_logic(X -> Y, V):-eval\_logic(X,V), eval\_logic(Y,V).

