ЛР №6 Списки на прологе

Цель работы: изучить работу со списками на языке Пролог.

Задание: составить стандартные алгоритмы работы со списками на Прологе. Представить в отчете скриншоты работающих программ.

1. Длина списка

2. Сумма элементов списка

3. Поиск элемента в списке 1

4. Поиск элемента в списке 2

5. Конкатинация списков Сп1 + Сп2

6. Конкатинация списков Сп2 + Сп1

7. Добавление элемента в список add

8. Добавление в начало списка push

9. Добавление в конец списка add

10. Последний элемент last, last2

11. neighbors, neighbors2

12. rev(22), revers(21)

13. палиндром

14. четные - не четные

15. номер элемента

16. удаление всех вхождений элемента из списка

17. удаление первого вхождения.

ЛР №7 Списки на прологе.

1. Множества

2. Сортировка списков

ЛР №6. Реализация стандартных алгоритмов.

1.

leng([],0).

leng([\_|T],X) :-

leng(T,XT),

X is XT + 1.

---

2.

summ([],0).

summ([H|T],X) :-

summ(T,XT),

X is XT + H.

---

3-4.

member([], \_, no ).

member([Z|\_], Z, yes) :- !.

member([\_|T], Z, Y ) :-

member(T,Z,Y).

---

member2(X, [X|\_]) :-

write("true"),nl.

member2(X, [\_|T]):-

member2(X,T).

---

member3(X,[X|\_]).

member3(X,[Y|T]) :-

not(X = Y), member3(X,T).

---

member4(X, [X|\_]) :-

write("true"),nl, !.

member4(X, [\_|T]):-

member4(X,T).

---

5.

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

view([]).

view([H|T]) :-

write(H),write(" "),

view(T).

----

6.

conc(L,[],L).

conc(L1,[H|L2],[H|L3]):-

conc(L1,L2,L3).

view([]).

view([H|T]) :-

write(H),write(" "),

view(T).

---

7.

add([],X,[X]).

add([H|T], X, [H|L]) :-

add(T,X,L).

---

add([],[X]) :-

read(X).

add([H|T], [H|L]) :-

add(T,L).

---

add([X|A]) :-

read(X),not(X=0),

add(A).

add([]).

---

8.

push(A,X,[X|A]).

---

push(A,B):-

read(X),

conc([X],A,B).

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

9.

add(A,B):-

read(X),

conc(A,[X],B).

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

10.

последний

last(A,X):-

conc(\_, [X], A).

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

предпоследний

last(A,X,Y):-

conc(\_, [X,Y], A).

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

предпоследний

last(A,X):-

conc(\_, [X|[\_|[]]], A).

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

last2([X],X).

last2([\_|T],X):-

last2(T,X).

---

еще раз member c conc

member5(A,X) :-

conc(\_, [X|\_], A),

write("true"), !.

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

11.

neighbors(A,X,Y) :-

conc(\_, [X,Y|\_], A),

write(X),write(" "),write(Y),!.

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

neighbors2(A,X,Y) :-

conc(\_, [X,Y|\_], A),

write(X),write(" "),write(Y);

conc(\_, [Y,X|\_], A),

write(Y),write(" "),write(X).

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

12.

reverse([],[]).

reverse([X|T],Z) :-

reverse(T,Y),

conc(Y,[X],Z).

conc([],L,L).

conc([H|L1],L2,[H|L3]):-

conc(L1,L2,L3).

---

rev([H|T],L1,L2):-

rev(T,[H|L1],L2).

rev([],L,L).

reverse2(L1,L2) :-

rev(L1,[],L2).

---

13.

rev([H|T],L1,L2):-

rev(T,[H|L1],L2).

rev([],L,L).

palindrom(L) :-

rev(L,[],X), L=X,

write("true").

---

rev([H|T],L1,L2):-

rev(T,[H|L1],L2).

rev([],L,L).

palindrom(L) :-

rev(L,[],L),

write("true").

---

14.

chet([],[],[]).

chet([H|T],[H|L1],L2) :-

H mod 2 =:= 0,

chet(T,L1,L2).

chet([H|T],L1,[H|L2]) :-

H mod 2 =:= 1,

chet(T,L1,L2).

---

chet([],[],[]).

chet([H|T],[H|L1],L2) :-

H mod 2 =:= 0,

chet(T,L1,L2).

chet([H|T],L1,[H|L2]) :-

chet(T,L1,L2).

---

15. Номер с начала списка

n\_elem([],-1,\_,no) :- !.

n\_elem([X|\_], 1, X,true) :- !.

n\_elem([\_|T], N, X,Z) :-

n\_elem(T,N1,X,Z), N is N1 + 1.

---

n\_elem([],\_,\_) :-

write("НЕТ"),nl,fail.

n\_elem([H|\_], H , 1).

n\_elem([\_|T], Key, X) :-

n\_elem(T, Key, X1),

X is X1 + 1.

---

n\_el(A,Key,Y) :-

n\_elem(A,Key,0,Y).

n\_elem([], \_, X, X):- !.

n\_elem([H|T], H, \_, Y) :-

n\_elem(T, H, 1, Y).

n\_elem([\_|T], H, X, Y) :-

X1 is X + 1,

n\_elem(T, H, X1, Y).

---

16.

delete\_all(\_, [], []).

delete\_all(X, [X|L], L1):-

delete\_all(X, L, L1).

delete\_all(X, [Y|L], [Y|L1]):-

not(X=Y),

delete\_all(X, L, L1).

---

17.

delete\_one(\_, [], []).

delete\_one(X, [X|L], L):- !.

delete\_one(X, [Y|L], [Y|L1]):-

not(X=Y),

delete\_one(X, L, L1).