Seminar 3 – Heterogeneous Lists in Prolog

Numarul de aparitii al unui element intr-o lista (rezolvarea standard, fara variabila colectoare)

```
%nrAparitii(L:list, E:element, R:integer)
%model de flux (i,i,o), (i,i,i)
%L - lista in care calculam numarul de aparitii
%E - elementul ale carui aparitii le numaram in lista
%R - rezultatul, numarul de aparitii

nrAparitii([], _, 0).
nrAparitii([H|T], E, R):-
    H =:= E,
    nrAparitii(T, E, R1),
    R is R1 + 1.
nrAparitii([H|T], E, R):-
    H =\= E,
    nrAparitii(T, E, R).
```

Metoda variabilei colectoare

Variabila colectoare (variabila acumulator) este o variabila intermediara pentru rezultate partiale (permite determinarea rezultatului final pe parcurs, nu doar la iesirea din apelul recursiv).

Variabila colectoare se va reflecta printr-un parametru in plus, atat in modelul matematic, cat si in codul Prolog.

Acest parametru va fi un parametru de tip ***input*** (ea este initializata in cadrul primului apel).

```
%nrAparitiiCol(L:list, E:element, Col:integer, R:integer)
%model de flux (i,i,i,o), (i,i,i,i)
%L - lista in care calculam numarul de aparitii
%E - elementul ale carui aparitii le numaram in lista
%Col - variabila colectoare
%R - rezultatul, numarul de aparitii

nrAparitiiCol([], _, C, C).
nrAparitiiCol([H|T], E, C, R):-
    H = E,
    C1 is C + 1,
    nrAparitiiCol(T, E, C1, R).
nrAparitiiCol([H|T], E, C, R):-
    H \=E,
    nrAparitiiCol(T, E, C, R).
```

2. Determinarea inversului unui numar folosind metoda variabilei colectoare

```
%inv_numar(Nr: int, Ncv: int, Rez: int)
%Nr - numarul care trebuie inversat
%Ncv - variabila colectoare in care retinem rezultatul partial
%Rez - rezultatul
%model de flux (i, i, o), (i, i, i)

inv_numar(0, Ncv, Ncv).
inv_numar(N, Ncv, Rez):-
        N > 0,
        Cifra is N mod 10,
        NcvNew is Ncv * 10 + Cifra,
        NN is N div 10,
        inv_numar(NN, NcvNew, Rez).
```

Definirea unei liste eterogene in Prolog

Lista eterogena este o lista formata din atomi, numere si alte liste (subliste).
EG: [[2,4,6],1,4,a,x,[3,6]] este o lista eterogena.

In lucrul cu acetse liste eterogene, vom utiliza din nou mecanismul de a imparti o lista in Head and Tail cu |.
Dupa impartirea listei in H and T, vom verifica tipul Headului.
In contextul acesta, H poate sa fie numar, atom sau lista.

Verificarea se realizeaza utilizand predicatele implicite:

- is_list(H) – returneaza True daca H este o lista
- number(H) – returneaza True daca H este un numar
- atom(H) – returneaza True daca H este un atom (simbol).

3. Se dă o listă de numere și liste de numere. Se cere ca din fiecare sublistă să se șteargă numerele palindrome.

Avand in vedere ca avem deja predicatul care ne determina inversul unui numar, pentru o sublista, parcugem elementele din lista si daca este palindrom, stergem numarul.

```
%delSub(L:list, R: list)
%L - lista liniara din care eliminam numerele care sunt palindrom
%LR - lista rezultat
%model de flux (i, o), (i, i)

delSub([], []).
delSub([H|T], [H|TR]):-
        inv_numar(H, 0, HI),
        H =\= HI,
        delSub(T, TR).
delSub([H|T], TR):-
        inv_numar(H, 0, HI),
        H =:= HI,
        delSub(T, TR).
```

Ne trebuie apoi un Main, pentru a parcurge lista eterogena: daca gasim numar, mergem mai departe si daca gasim lista, intram in sublista si producem modificarea solicitata.

```
%mainHeter(L: lista, LR: list)
%L - lista eterogena initiala
%LR - lista rezultat
%model de flux (i, o), (i, i)

mainHeter([], []).
mainHeter([H|T], [H|LR]):-
        number(H),
        mainHeter(T, LR).
mainHeter([H|T], [H1|LR]):-
        is_list(H),
        delSub(H, H1),
        mainHeter(T, LR).
```

- Când lucrăm in SWI-Prolog cu liste lungi, s-ar putea ca Prolog să ne afișeze ca rezultat o listă de genul: R = [1, 2, [2, 1, 6], 6, [154, 11, 10], 7, 1, 0, [...|...]|...]. Pentru a avea lista întreagă, putem folosi predicatul write:
    - Dacă căutarea în SWI-Prolog nu s-a terminat de tot (adică SWI-Prolog așteaptă; ca să continue căutarea), apăsând tasta w, se va afișa lista completă.
    - Dacă căutarea s-a terminat, predicatul trebuie apelat din nou, adăugând și un apel la predicatul write: numepredicatapelat([1,2,3,4], R), write(R).

4. Se da o lista eteorgena (contine atomi, numere si subliste).
Sa se stearga toate secventele strict crestcatoare din fiecare sublista si sa se afiseze lista transformata.
Eg. prelucreaza([[1, 2, 4, 4, 7, 9, 0, 1, 1], 3,[], 24, [], a, [12, 4], [3, 5], b],R).

⇨ R = [[1], 3, [], 24, [], a, [12, 4], [], b]

```
% eliminaCresc(L:list, R:list)
% model de flux: (i,o) sau (i,i)
% L - lista din care eliminam secventele
% R - lista rezultat
eliminaCresc([], []).
eliminaCresc([H], [H]).
eliminaCresc([H1,H2], []):- H1 < H2.
eliminaCresc([H1,H2,H3|T], R):-
            H1 < H2,
            H2 < H3,
            eliminaCresc([H2,H3|T], R).
eliminaCresc([H1,H2,H3|T], R):-
            H1 < H2,
            H2 >= H3,
            eliminaCresc([H3|T], R).
eliminaCresc([H1,H2|T], [H1|R]):-
            H1 >= H2,
            eliminaCresc([H2|T], R).
```

```
%prelucreaza(L: lista, LR: list)
%L - lista eterogena initiala
%LR - lista rezultat
%model de flux (i, o), (i, i)

prelucreaza([], []).
prelucreaza([H|T], [H|LR]):-
        number(H),
        prelucreaza(T, LR).

prelucreaza([H|T], [H|LR]):-
        atom(H),
        prelucreaza(T, LR).

prelucreaza([H|T], [H1|LR]):-
        is_list(H),
        eliminaCresc(H, H1),
        prelucreaza(T, LR).
```