predicat(*listă de variabile si constante*).

De fapt e: predicat(lista de termeni).

De exersat la laborator:

```
?- var(A).
true.

?- nonvar(A).
false.

?- var(_).
true.

?- nonvar(_).
false.

?- var('A').
false.

?- nonvar('A').
true.

?- var(a).
false.

?- nonvar(a).
true.

?- var(1).
false.

?- nonvar(1).
true.

?- var(p(X)).
false.

?- nonvar(p(X)).
true.

?- var(-->).
false.

?- nonvar(-->).
true.

?- compound([]).
false.

?- compound(X+1).
true.
```

```
?- atom(a).
true.

?- atom(alta_constanta).
true.

?- atom(-#-->).
true.

?- atom(1).
false.

?- atom([]).
false.

?- atom(p(X)).
false.

?- atom(X).
false.

?- atomic(a).
true.

?- atomic(alta_constanta).
true.

?- atomic(-#-->).
true.

?- atomic(1).
true.

?- atomic([]).
true.

?- atomic(p(X)).
false.

?- atomic(X).
false.
```

```
?- number(a).
false.

?- number(10+2).
false.

?- number(12).
true.

?- number(12.5e2).
true.

?- integer(5).
true.

?- integer(-5).
true.

?- integer(5.0).
false.

?- float(5).
false.

?- float(5.0).
true.

?- float(-5.0e2).
true.

?- float(-5e2).
true.

?- float(5e0).
true.

?- float(-15.25e-2).
true.

?- float(-15.25E-2).
true.
```

(Operatori pentru unificare, respectiv testarea egalității, inegalității, literal identității: predicate binare infixate predefinite)

```
= testeaza egalitatea ca expresii; face si unificarea, in unele versiuni de Prolog
cu testarea ocurentelor variabilelor in termeni, in alte versiuni fara acest test
unify_with_occurs_check face unificarea cu testarea ocurentelor variabilelor in termeni
\= testeaza nonegalitatea ca expresii
=:= testeaza egalitatea ca valori de expresii aritmetice calculate
=\= testeaza nonegalitatea ca valori de expresii aritmetice calculate
== testeaza literal identitatea
\== testeaza literal nonidentitatea
\+ sau not reprezinta negatia
=< este mai mic sau egal
>= este mai mare sau egal
```

Si de facut:

```
concat([],L,L).
concat([H|T],L,[H|M]) :- concat(T,L,M).

inversa([],[]).
inversa([H|T],L) :- inversa(T,M), concat(M,[H],L).
```

Predicatul *cut* (!)

Observați, mai jos, predicatul zeroar (adică fără argumente) *fail*, care întotdeauna eșuează, adică întotdeauna e evaluat la **false**, și predicatul unar predefinit *not* sau \+, care primește drept argument un predicat și întoarce **true** când acel predicat eșuează și **false** când acel predicat e satisfăcut.

După cum am văzut mai sus, pentru satisfacerea unui scop, faptele și regulile sunt aplicate în ordinea în care sunt scrise în baza de cunoștințe, printr–un backtracking încorporat în interpretorul Prologului.

Predicatul predefinit *cut* ("!") are funcția de a tăia backtrackingul executat pentru satisfacerea unui scop dintr-o interogare, astfel că, după ce scopul respectiv e satisfăcut prin aplicarea regulii în care apare *cut*, nu se mai caută alte satisfaceri ale acelui scop.

Exemplu (din fișierul *testcut.pl*)

Predicatele următoare testează apartenența unui element la o listă:

```
membru(_,[]) :- fail. %%% corect si: not(membru(_,[])).
membru(H,[H|_]).
membru(X,[_|T]) :- membru(X,T).

apartine(_,[]) :- fail. %%% corect si: not(apartine(_,[])).
apartine(H,[H|_]) :- !.
apartine(X,[_|T]) :- apartine(X,T).
```


De facut la curs: