Laborator 2

Grup facebook: https://www.facebook.com/groups/1404674159791963/

- Copiati urmatorul program Prolog intr-un fisier cu extensia .pl;
- Setati *trace –ul :*

  ?- trace.
- Urmariti cum se efectueaza procesul de backtracking in urma apelarii:

  ?- floare(crin, X,Y,Z).

```
culoare(rosu).
culoare(galben).
culoare(alb).
frunze(alterne).
frunze(mari).
frunze(nervuri_paralele).
petale(multepetale),
petale(petalegalbene).
petale(grupate_trompeta).

fl(trandafir, rosu, alterne, multepetale).
fl(floarea_soarelui, galben, mari, petalegalbene).
fl(crin, alb, nervuri_paralele, grupate_trompeta).


floare(A, X, Y, Z):- culoare(X), frunze(Y), petale(Z), fl(A,X,Y,Z).
```

- Puteti elimina *trace* – ul astfel:

  ?- notrace.

➢Predicatul cut (!)

Daca Prologul gaseste predicatul **cut** intro regula, nu va mai efectua backtracking.
Toate deciziile luate vor ramane finale. De exemplu, in programul de mai jos, **cut**-ul impiedica Prologul sa reinstantieze variabila X:

```
a(X, Y) :- b(X), !, c(Y).
b(1).
b(2).
b(3).

c(1).
c(2).
c(3).
```

Interogare:

?- a(Q, R).
Q = 1
R = 1 ;
Q = 1
R = 2 ;
Q = 1
R = 3 ;
No

• Cut rosu: modifica corespondenta dintre semnificatia declarativa si cea procedurala a programului Prolog;

• Cut verde: ajuta la cresterea eficientei programului.

```
min1(X, Y, X):- X=<Y,!.          % cut verde
min1(X, Y, Y):-X>Y.

min2(X, Y, X):- X=<Y,!.          % cut rosu
min2(X, Y, Y).
```

Daca se schimba ordinea clauzelor pentru min2, solutiile rezultate pot fi gresite:

```
min2(X, Y, Y).
min2(X, Y, X):- X=<Y,!.          % cut rosu
```

➢Predicatul fail

• ***fail*** intr-o conjunctie de scopuri (de obicei la sfarsit), forteaza intrarea in procesul de backtrackning;
• folosit dupa predicatul ***cut*** determina Prolog-ul sa nu mai efectueze backtracking-ul.

Copiati programul de mai jos intr-un fisier cu extensia .pl si interogati-l ca mai jos. Observati efectul predicatului *fail* .

```
getX('X1').
getZ('Z1').
getX('X2').
getZ('Z2').
wr(_):- getX(X), getZ(Z), wrXZ(X, Z), fail.
wrXZ(X, Z):-  write(X), write('  '), write(Z), write('  '), nl, fail.
```

Interogare:

?- wr(_).

• ***nl*** determina programul sa sara la linie noua.
• **write(X)** afiseaza pe ecran valoarea cu care a fost instantiat X-ul

➢ **Diferenta dintre '*is*' si '=':**

1
```
min2(X, Y, X):- X=<Y,!.
min2(X, Y, Y).
```

2
```
min3(X, Y, Z):- X=<Y, Z is X, !;   Z is Y.
```

3
```
min4(X, Y, Z):- X=<Y, Z = X, !;    Z = Y.
```

4
```
return_min_plus_one1(X, Y, Z):- X=<Y, Z is X+1, !; Z is Y + 1.
```

5
```
return_min_plus_one2(X, Y, Z):- X=<Y, Z = X+1, !; Z = Y + 1.
```

6
```
getH('Solutia este').
app(X,C):- getH(H), C = [H,X], !.
min5(X, Y, Z):- X=<Y, app(X,C), Z = C, !; app(Y,C),  Z = C,!.
```

7
```
getH('Solutia este').
app(X,C):- getH(H), C is [H,X], !.
min6(X, Y, Z):- X=<Y, app(X,C), Z is C, !; app(Y,C),  Z is C,!.
```

- Implementati reguli Prolog care sa:
  - determine daca anumite perechi de coordonatele formeaza:
    - un triunghi;
    - un patrat;
    - o linie:
      - verticala;
      - orizontala.

- Implementati un program Prolog care sa determine daca un numar dat este prim.

- Implementati un program Prolog care sa calculeze cel mai mare divizor comun a doua numere. Verificati daca doua numere sunt coprime (daca au cmmdc = 1).

- Implementati un program Prolog care sa calculeze factorialul unui numar.

- Implementati un program Prolog care sa extraga cifrele din numere si sa afiseze pe ecran perechi de forma:

  Exemplu: 71 -> (7, sapte); (1, unu).

➢Liste

• O lista este o secventa de oricate articole separate prin virgula.

[1,2,3,4,5,a,b,c]

• Lista vida se noteaza [].

• O lista nevida se poate imparti in cap si coada [A|B]. Capul A este un singur element. Coada B este lista. Putem pune in evidenta mai multe elemente la inceputul listei:

[A,B,C,…|T].

•Exemplu:

[1,2,3]=[1|[2,3]]=[1,2|[3]]=[1,2,3|[]].

•Putem colecta intr-o lista elemente ce satisfac o anumita proprietate cu ajutorul a 3 predicate predefinite:

1) **bagof(X,P,L)** pune in lista L elementele X ce satisfac P. Daca nu exista nici un astfel de element raspunsul este no.

2) **setof(X,P,L)** la fel ca bagof dar elimina duplicatele iar lista rezultata este sortata

3) **findall(X,P,L)** daca nu exista nici un element care sa satisfaca P rezultatul este yes iar L$\in \emptyset$. Nu tine cont de variabilele care apar in P si nu apar in X.

(Le vom studia in detaliu in laboratorul urmator – exemple pe pagina urmatoare)

•Adaugarea unui element in lista:

```
add_list([], L, L).
add_list(X, L, [X|L]).
```

•Concatenarea a doua liste:

```
lconcat([H|Tail], List2, [H|TailRez]):-lconcat(Tail, List2, TailRez).
lconcat([], L, L).
```

• Stergerea unui element din lista:

```
elimina(_,[],[]).
elimina(X,[X|Tail],T):-elimina(X,Tail,T),!.
elimina(X,[H|Tail],[H|T]):-elimina(X,Tail,T).
```

- ❑ Sa se elimine duplicatele dintr-o lista.
- ❑ Sa se elimine toate elementele mai mari decat un dumar dat dintr-o lista.
- ❑ Calculati reuniunea, intersectia si diferenta a doua multimi.