

Deklaratív Programozás gyakorlat  
Prolog programozás  
2013.10.31.

Hasznos tudnivalók:

"univ" azaz =.. /2 beépített eljárás

Hívási módok:

+Kif =.. ?Lista

-Kif =.. +Lista

Jelentése: igaz, ha

o Kif = Str(A1,...,An) és Lista = [Str,A1,...,An],

ahol Str egy névkonstans és A1,...,An tetszőleges kifejezések; vagy

o Kif = C, és Lista = [C], ahol C egy (név- vagy szám-)konstans.

atom\_concat/3 beépített eljárás

Hívási módok:

atom\_concat(+A1, +A2, ?A)

atom\_concat(?A1, ?A2, +A)

Jelentése: az A1 és A2 atomok összefűzése az A atom.

select/3 eljárás (lists könyvtár)

Hívási módok:

select(?X, +L1, ?L2)

select(?X, ?L1, +L2)

Jelentése: X eleme az L1 listának és az L2 listát kapjuk, ha L1-ből elhagyjuk X egy előfordulását.

A következő feladat példa arra, hogy a Deklaratív Programozás vizsga Prolog részében milyen "mitírki" jellegű feladatokat kell megoldani. A GY típusú feladatra 7 pont kapható a Prolog vizsgarészen megszerezhető 35 pontból.

### 1. GY (gyakorló) feladat

Tekintse az alábbi Prolog programot és döntse el mindegyik célról, hogy hogyan fut le:

- sikerül,
- megghiúsul, vagy
- hibát jelez!

Sikeres futás esetén adja meg az összes olyan változó értékét, amelynek neve nem aláhúzás-jellel (\_) kezdődik. Ha egy cél többféleképpen is sikerülhet, akkor adja meg az összes lehetséges behelyettesítést pontosvesszőkkel elválasztva!

Mindegyik célt a Prolog interpreternek önmagában adjuk oda, azaz futásának kezdetén a célban előforduló változóknak nincs értéke. Feltételezzük, hogy a 'lists' könyvtár be van töltve.

```
p(1).
p(2).
p(X) :- X > 1.
```

```
m(2, 0).
m(1, 2).
m(1, 3).
m(2, 1).
m(_, 4).
```

```
q(X) :- m(_, X), p(X).
```

- (a) ?- select(1, [2,X,3], L).
- (b) ?- atom\_codes(abc, [\_|L]), atom\_codes(X, L).
- (c) ?- \+ \+ X = 1, X = 2.
- (d) ?- m(1, X).
- (e) ?- q(X).

A következő feladatok bemutatják, hogy a Deklaratív Programozás vizsga Prolog részében milyen programozási feladatokat kell megoldani. A P típusú vizsgafeladat két részből áll: egy segédeljárás megírásából és a teljes feladat megoldásából. Ez utóbbiban célszerű, de nem kötelező a segédeljárás felhasználása. A teljes feladatra 21 pont kapható (a Prolog vizsgarészen megszerezhető 35 pontból), ha csak a segédeljárást írja meg, arra legfeljebb 11 pont adható.

### P (programozási) feladat

Egy A atom egy N nemnegatív egésszel vett szorzatán az A atom N-szeri egymás után fűzésével kapott atomot értjük.

### 2. Segédeljárás (az átlagosnál könnyebb):

Írjon Prolog nyelven egy eljárást amelynek bemenete egy N nemnegatív egész és egy A atom, kimenete pedig N és A szorzata! Adjon jobbrekurzív megoldást!

% szorzata(N, A, NA): NA az N nemnegatív egész és az A atom szorzata.  
% :- pred szorzata(int::in, atom::in, atom::out).

Példák:

```
| ?- szorzata(2, korte, NA).          ----> NA = kortekorte ? ; no
```

```
| ?- szorzata(0, korte, NA).          ----> NA = ' ' ? ; no
```

### 3. Teljes feladat:

A szorzata/3 eljárás segítségével, könyvtári eljárások használata nélkül készítsen el egy Prolog eljárást, amely egy számokból és atomokból a + és \* operátorokkal felépített ún. atomkifejezés értékét kiszámítja! Ha A atomot, N nemnegatív egészet jelöl, akkor az AKif-fel jelölt atomkifejezéseket a következő nyelvtani szabállyal jellemezhetjük:

AKif ---> A | N\*AKif | AKif\*N | AKif+AKif.

A + operátor a konkatenálást jelöli, a \* operátor pedig a segédfeladatban definiált szorzást.

% erteke(OpKif, Ertek): Az OpKif operátoros ábrázolású  
% fent definiált alakú kifejezés értéke Ertek.  
% :- pred erteke(univ::in, atom::out).

Példák:

```
| ?- erteke(alma, E).
E = alma ? ; no
```

```
| ?- erteke(ba*2, E).
E = baba ? ; no
```

```
| ?- erteke(2*ba, E).
E = baba ? ; no
```

```
| ?- erteke(alma+'_korte'+2*'_barack', E).
E = alma_korte_barack_barack ? ; no
```

P (programozási) feladat

Egy Prolog kifejezést szimmetrikusnak mondunk, ha a szimmetrikus pozíciójú argumentumai azonosak (azaz egy n-argumentumú kifejezés szimmetrikus, ha az első és n-edik, 2. és (n-1)-ik.,..., i.-k és (n+1-i).-ik argumentuma azonos). Az argumentum nélküli kifejezéseket szimmetrikusaknak tekintjük.

4. Segéd eljárás

Írjon Prologban egy olyan eljárást, amely egy tetszőleges Prolog kifejezésről megállapítja, hogy az szimmetrikus-e! Az azonosság eldöntésére az `==/2` beépített eljárást használhatja. Vigyázat, a kifejezés tartalmazhat változót, és maga is lehet változó!

```
% szimmetrikus(Kif): igaz, ha Kif szimmetrikus.
% :- pred szimmetrikus(univ::in).
```

Példák:

```
| ?- szimmetrikus(f(g(X),h(a,a),j(3,4,3))).      ----> no
| ?- szimmetrikus(f(g(a,b),h(2,4),g(a,b))).    ----> yes
| ?- szimmetrikus(f(1)).                       ----> yes
| ?- szimmetrikus(1-X).                        ----> no
| ?- szimmetrikus(X-X).                       ----> yes
| ?- szimmetrikus(X).                          ----> yes
```

5. Teljes feladat:

A szimmetrikus/1 predikátum segítségével írjon egy olyan Prolog eljárást, amely egy tetszőleges Prolog kifejezés összes szimmetrikus részkifejezéseit felsorolja. A részkifejezések előfordulási sorrendjét őrizze meg! Vigyázat, a kifejezés lehet változó és benne is előfordulhatnak változók!

```
% szimmetrikus_resze(Kif, Resz): Resz a Kif-nek szimmetrikus részkifejezése.
% :- pred szimmetrikus_resze(univ::in, univ::out).
```

Példák:

```
| ?- szimmetrikus_resze(f(1,a,1), R).
R = f(1,a,1) ? ; R = 1 ? ; R = a ? ; R = 1 ? ; no

| ?- szimmetrikus_resze(f(2+1+1,3+3+3), R).
R = 2 ? ; R = 1 ? ; R = 1 ? ; R = 3+3 ? ; R = 3 ? ; R = 3 ? ; R = 3 ? ; no

| ?- szimmetrikus_resze(g(X,Y), R).
R = X ? ; R = Y ? ; no
```

-----  
Az alábbi két feladatot otthoni ill, szorgalmi munkára adjuk.  
-----

P (programozási) feladat

6. Segéd eljárás:

Írjon egy olyan Prolog eljárást, amely listák listáját kapja bemenetként, és eredményül adja egyrészt a listák fejének listáját, másrészt a listák farkainak listáját! Ha a bemenet valamelyik eleme üres lista, akkor az eljárás hiúsuljon meg!

```
% elsok(M, Elsok, Maradek): Elsok az M-ben szereplő listák fejének,
% Maradek pedig farkainak a listája.
% :- pred elsok(list(list(univ))::in,
%               list(univ)::out,
%               list(list(univ))::out).
```

Példák:

```
| ?- elsok([[1,2,3],
            [4,5,6],
            [7,8,9],
            [0,0,0]], E, M).
E = [1,4,7,0],
M = [[2,3],
     [5,6],
     [8,9],
     [0,0]] ? ; no

| ?- elsok([], E, M).
E = [], M = [] ? ; no

| ?- elsok([[1]], E, M).
E = [1], M = [[]] ? ; no

| ?- elsok([[[]],[[]]], E, M).
no
```

7. Teljes feladat:

Az `elsok/3` predikátum segítségével írjon olyan Prolog-eljárást, amely előállítja egy mátrix transzponáltját! Egy mátrixot sorok listájaként adunk meg, ahol egy sor a mátrix-elemek listája. A sorok mind azonos hosszúságúak.

Egy `a[i,j]` `n*m`-es mátrix transzponáltja az `a[k,l]` `m*n`-es mátrix, ahol `b[k,l] = a[l,k]`. Vigyázat, a mátrix üres (`[]`) is lehet!

```
% transzponalt(M, TM): TM az M matrix transzponáltja.
% :- pred transzponalt(list(list(univ))::in, list(list(univ))::out).
```

Példák:

```
| ?- transzponalt([[1,2,3],
                  [4,5,6],
                  [7,8,9],
                  [0,0,0]], T).
T = [[1,4,7,0],
     [2,5,8,0],
     [3,6,9,0]] ? ; no

| ?- transzponalt([], T).
T = [] ? ; no

| ?- transzponalt([[1]], T).
T = [[1]] ? ; no
```