

## *Logikai programozás*

7.

### TÖBBSZÖRÖS MEGOLDÁS

Problémafelvetés:	?- segelytKap(Diak).
diak(jani, 3.3, pecs).	Diak = laci ;
diak(laci, 3.7, kaposvar).	Diak = matyi ;
diak(matyi, 4.1, pecs).	Diak = jeno ;
diak(kati, 2.3, barcs).	Diak = laci ;
diak(jeno, 4.2, szekszard).	Diak = kati ;
	Diak = jeno.

segelytKap(Diak) :- diak(Diak, Atlag, \_), Atlag > 3.5.  
segelytKap(Diak) :- diak(Diak, \_, Hely), Hely \== 'pecs'.

Állapítsuk meg, hogy kik kapnak segélyt!

## TÖBBSZÖRÖS MEGOLDÁS

1. Megoldás:

listaba(L) :- listaba([],L), !.

listaba(Akt, L) :- segelytKap(Diak) ,  
                  \+member(Diak,Akt),  
                  listaba([Diak|Akt], L).

listaba(L,L).

2. Megoldás:

A továbbiak ismeretében találja ki önállóan.

## TÖBBSZÖRÖS MEGOLDÁS

kor(jani,6).	bagof(Nev, kor(Nev,6),L).
kor(jozsi,4).	setof(Nev, kor(Nev,6),L).
kor(judit,7).	findall(Nev, kor(Nev,6),L).
kor(jani,6).	
kor(marci,6).	bagof(Nev, kor(Nev,2),L).
kor(gizi,6).	setof(Nev, kor(Nev,2),L).
kor(jeno,3).	findall(Nev, kor(Nev,2),L).
kor(mari,5).	
kor(jani,6).	bagof(Nev, kor(Nev,Ev),L).
kor(gizi,6).	setof(Nev, kor(Nev,Ev),L).
kor(jakab,4).	findall(Nev, kor(Nev,Ev),L).
kor(eva,6).	
kor(jani,5).	setof(Nev, kor(Nev,_),L).
kor(dezső,3).	findall(Nev, kor(Nev,_),L).

## TÖBBSZÖRÖS MEGOLDÁS

bagof(Valtozo, feltetel, Lista):

Lista-ba gyűjti a Valtozo feltetel-nek megfelelő értékeit. Ahányszor megtalál egy értéket, annyiszor belerakja a listába. Ha nincs a feltételnek megfelelő elem,  $\Rightarrow$  false.

setof(Valtozo, feltetel, Lista):

Lista-ba gyűjti a Valtozo feltetel-nek megfelelő értékeit. Minden értéket csak egyszer rak a listába, majd növekvő sorrendbe rendezi a listát.

setof(Valtozo, feltetel, Lista) :- bagof(Valtozo, feltetel, Lista1),  
sort(Lista1, Lista).

## TÖBBSZÖRÖS MEGOLDÁS

findall(Valtozo, feltetel, Lista):

Lista-ba gyűjti a Valtozo feltetel-nek megfelelő értékeit. Ahányszor megtalál egy értéket, annyiszor belerakja a listába. Ha nincs a feltételnek megfelelő elem,  $\Rightarrow$  üres lista.

## TÖBBSZÖRÖS MEGOLDÁS

Lényeges különbség a bagof-setof és a findall között:

Ha a feltételben van szabad változó, akkor

a/ a bagof-setof behelyettesít oda egy lehetséges értéket, majd kigyűjti a Valtozo ilyen feltételnek megfelelő értékeit.

b/ a findall minden lehetséges értéket behelyettesít a szabad változó helyére, és úgy gyűjti ki a Valtozo megfelelő értékeit.

## TÖBBSZÖRÖS MEGOLDÁS

Ez megoldható a bagof-setof-fal is:

$\text{setof}(\text{Nev}, \text{Ev}^{\text{kor}}(\text{Nev}, \text{Ev}), \text{Nevek}),$

$A^{\wedge}$  jelentése: van olyan.

Vagyis az

$\text{Ev}^{\text{kor}}(\text{Nev}, \text{Ev})$  jelentése:

van olyan Ev, amelyre  $\text{kor}(\text{Nev}, \text{Ev})$  igaz, azaz van olyan Ev, amelyre a  $\text{kor}(\text{Nev}, \text{Ev})$  szerepel az adatbázisban.

## TÖBBSZÖRÖS MEGOLDÁS

Más céllal, de térjünk vissza a kor/2 adatokat tartalmazó adatbázisra.

```
forall(kor(Nev,6), writef('%w\n',[Nev])).  
forall(kor(Nev,Ev), writef('%w\t%w\n',[Nev,Ev])).  
forall(kor(Nev,_), writef('%w\n',[Nev])).
```

```
forall(kor(Nev,Ev), barmi(Nev,Ev)).
```

## TÖBBSZÖRÖS MEGOLDÁS

Általánosan:

```
forall(feltetel, akcio).
```

**FIGYELEM!!**

Ha a forall akció része hamis, akkor az egész forall is az!!

## MÁR MEGINT KOCSMÁZÁS

kocsmazok1.pl:

szereti(pal,sor).

...

mernek(sarok,bor).

...

jar(Diak,Kocsma):-szereti(Diak,Ital), mernek(Kocsma,Ital).

kocsmabaJarok :- writeln('A kocsmába járó diákok: '), ...

## MÁR MEGINT KOCSMÁZÁS

kocsmazok2.pl:

jar(Diak,Kocsma):-szereti(Diak,Ital), mernek(Kocsma,Ital).

kocsmabaJarok :- writeln('A kocsmába járó diákok: '), ...

Működik, ha konzultálunk az 'adatok.pl' fájjal:

:- consult('kocsmaadatok.pl').

vagy

start :-

consult('kocsmaadatok.pl'), ...

## ADATBÁZISKEZELÉS

Adatok: a tényállítások

consult : hatására az adatok bekerülnek a memóriába

Lekérdezés:

Ahogy eddig – pl.:

szereti(jani, sör).

szereti(Ki, Mit).

mernek(Hol, bor).

## ADATBÁZISKEZELÉS

Beszúrás: assert/1

pl.: assert(szereti(jani, bor)).

Törlés: retract/1, retractall/1

pl.: retract(szereti(jani, bor)).

retract(szereti(jani, \_)).

retract(szereti(\_, \_)).

retractall(szereti(jani, \_)).

Ha nincs törlendő:

eredmény: false

Ha nincs törlendő:

eredmény: true

## ADATBÁZISKEZELÉS

Az előző műveletek végrehajtódnak, ha pl. a kocsmazok2.pl fájl futásakor adjuk ki a parancsokat,

de hibaüzenettel elszáll a kocsmazok1.pl futása közben:

```
assert/1: No permission to modify static_procedure
`szereti/2'
```

A predikátumok alapértelmezetten statikusként fordulnak.

Dinamikussá tétel:	Figyelem!
:- dynamic szereti/2.	Ez deklaráció, nem predikátum, vagyis nincs zárójel!

## ADATBÁZISKEZELÉS

Összes adat kilistázása: listing

pl.: listing(szereti/2).

Az adatok addig „élnek”, amíg ki nem lépünk a futató-környezetből.

Fájlba mentés mint eddig, de a listing-gel egyszerűsíthető:

pl.: fajlba:- tell('fajlnev'), listing(szereti/2), told.

Hatására a dynamic deklaráció is a fájlba kerül – saját adatfájlnál is célszerű ide írni.



## ADATBÁZISKEZELÉS - PÉLDA

Szűrjünk be diákokat egy adatbázisba!

start :- beolvas, kiir.

beolvas :- write('A diák neve: '), read(Nev),  
          assert(diak(Nev)),  
          write('Van még? (i/n)'), read(i),  
          beolvas.

beolvas.  :- dynamic diak/1.

kiir :- tell('fajlnev.txt'), listing(diak/1), told.      diak(jani).  
  diak('béla').  
  diak(mari).