Logikai programozás

7.

TÖBBSZÖRÖS MEGOLDÁS

Problémafelvetés:

```
diak(jani, 3.3, pecs).
diak(laci, 3.7, kaposvar).
diak(matyi, 4.1, pecs).
diak(kati, 2.3, barcs).
diak(jeno, 4.2, szekszard).
```

```
?- segelytKap(Diak).
Diak = laci ;
Diak = matyi ;
Diak = jeno ;
Diak = laci ;
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).
kor(jozsi,4).
kor(judit,7).
kor(jani,6).
kor(marci,6).
kor(gizi,6).
kor(jeno,3).
kor(mari,5).
kor(jani,6).
kor(gizi,6).
kor(jakab,4).
kor(eva,6).
kor(jani,5).
kor(dezső,3).
```

```
bagof(Nev, kor(Nev,6),L).
setof(Nev, kor(Nev,6),L).
findall(Nev, kor(Nev,6),L).

bagof(Nev, kor(Nev,2),L).
setof(Nev, kor(Nev,2),L).
findall(Nev, kor(Nev,2),L).

bagof(Nev,kor(Nev,Ev),L).
setof(Nev,kor(Nev,Ev),L).
findall(Nev,kor(Nev,Ev),L).

setof(Nev, kor(Nev,_),L).
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:

```
setof(Nev, Ev^kor(Nev,Ev), Nevek),
```

A ^ jelentése: van olyan.
Vagyis az
Ev^kor(Nev,Ev) jelentése:
van olyan Ev, amelyre kor(Nev,Ev) igaz, azaz van olyan Ev, amelyre a kor(Nev,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' fájllal:

```
:- 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(_,_)).

Ha nincs törlendő:

eredmény: false

retractall(szereti(jani,_)).

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 futtató-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

Szúrjunk 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.

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

```
:- dynamic diak/1.

diak(jani).
diak('béla').
diak(mari).
```