SZLÁVI PÉTER: A PROLOG NYELV „FELDOLGOZÁSA” (I. RÉSZ)

Adottak az árpádházi királyokról a következő adatok:

- ki kicsodának az apja,
- ki kicsodának a felesége,
- ki mikor uralkodott fejedelemként, ill.
- királyként.

1. ÁRPÁDHÁZI KIRÁLYOK „DIAGNOSZTIKÁJA” I.

Adjuk meg a Prolog szabályokat az alábbi kérdések megválaszolására:

- anyja-e valaki valakinek,
- szüleje-e valaki valakinek,
- nagyszüleje-e valaki valakinek,
- őse-e valaki valakinek,
- fejedelemné-e valaki,
- királyné-e valaki,
- testvére-e valaki valakinek,
- mostoha testvére-e (féltestvére-e) valaki valakinek,
- unokatestvére-e valaki valakinek,
- uralkodott-e egy adott érában,
- mikor uralkodott,
- azon uralkodó-e, aki fia is uralkodott,
- azon uralkodó-e, aki testvére is uralkodott,
- ...?

Tanács: az egyes szabályok felállításához célszerű először megfogalmazni függvényként azokat „matematikai formalizmussal”, rekurzívan, s csak ezután kódolni Turbo Prologban! Valahogy így:

```
anyja(Anya,X):=létezik Apa: apja(Apa,X) és felesége(Apa,Anya),
```

vagy

```
őse(Ki,Kinek):=szülője(Ki,Kinek) vagy
    létezik Valaki: szülője(Valaki,Kinek) és őse(Ki,Valaki)
```

Az „alap-adatbázis”:

```
/*
 Árpád-ház családfa részlet:
*/
Domains
/*
 A predikátumok (axiómák, szabályok) „értelmezési tartományai”,
 hagyományosan: típusdefiníciók:
/*
 szemely = string
 idopont = integer
```

```
Predicates
/*
 Az axiómák és szabályok értelmezési tartományának deklarálása;
 minden, később előkerülő axióma és szabály itt deklaráltatik:
*/
 apja(szemely,szemely)
 felesege(szemely,szemely)
 fejedelem(szemely,idopont,idopont)
 kiraly(szemely,idopont,idopont)
 anyja(szemely,szemely)
 szuloje(szemely,szemely)
 nagyszuloje(szemely,szemely)
 ose(szemely,szemely)
 kiralyne(szemely)
 testvere(szemely,szemely)
 feltestvere(szemely,szemely)
 uralkodott(szemely,idopont)
 fia_is_uralkodott(szemely)
 testvere_is_uralkodott(szemely)
Clauses
/*
 A „puszta” tények:
*/
apja(almos, arpad).
apja( arpad, levente).
apja( arpad, tarhos).
apja( arpad, solt).
apja( solt, taksony).
apja( taksony, geza).
apja( geza, istvan).
apja( istvan, imre).
apja( geza, judit).
apja( geza, "leany 1").
apja( geza, "leany 2").
apja( orseolo, peter).
apja( taksony, mihaly).
apja( mihaly, vazul).
apja( vazul, endre).
apja( endre, salamon).
apja( endre, adelheid).
apja( vazul, bela).
apja( bela, "I.geza").
apja( "I.geza", kalman).
apja( kalman, laszlo)
apja( kalman, "II.istvan")
apja( bela, "I.laszlo").
apja( bela, eufemia).
apja( vazul, levente).
apja( mihaly, laszlo).
```

```
felesege(solt,"menmarot lany").
felesege(geza,sarolta).
felesege(istvan,gizella).
felesege(orseolo,"leany 2").
felesege(laszlo,premiszlava).
felesege(endre,anasztazia).
felesege(bela,rixa).

felesege(salamon,judit).
felesege(vratislav,adelheid).
felesege("I.geza","görögnő").
felesege(kalman,buzilla).
felesege(kalman,eufemia).
```

```
fejedelem(almos,819,895).
fejedelem(arpad,889,907).
fejedelem(solt,907,947).
fejedelem(taksony,947,970).
fejedelem(geza,970,997).
fejedelem(istvan,997,1001).

kiraly(istvan,1001,1038).
kiraly(peter,1038,1041).
kiraly(peter,1044,1046).
kiraly(endre,1046,1060).
kiraly(bela,1060,1063).
kiraly(salamon,1063,1074).
kiraly("I.geza",1074,1077).
kiraly("I.laszlo",1077,1095.
kiraly(kalman,1095,1116).
kiraly("II.istvan",1116,1131).
```

Megoldás:

Előzetes, (Turbo) Prologra vonatkozó megjegyzések:

1. Az eljárásokhoz és függvényekhez szokottaknak idegen a Prolog azon lényegi sajátossága, hogy benne minden logikai értékkel rendelkező axióma, vagy szabály.

Így ha valamit hagyományosan a következő eljáráshívással lehetett megoldani:

```
b(Y,a(X)),
```

ahol

```
Eljárás b(YTípus,ZTípus)
```

```
és Függvény a(XTípus):ZTípus.
```

Ennek megfelel a Prologban:

```
a(X,Z) and b(Y,Z)
```

szabály,

ahol 'a' és 'b' szabályok.

2. Némely esetben elkerülhetetlen a rekurzió a szokványos iteratív ciklus megvalósítására (pl. őse):

a hagyományos iterációnak:

```
'Z:=h(X)
 Ciklus amíg f(X)
  Z:=g(X,Z); X:=i(X)
 Ciklus vége'
```

megfelel két szabály:

```
c(X,Z) if Z=h(X) and
ciklus(X,Z) or Z=h(X).
ciklus(X,Z) if f(X) and
     ZZ=g(X,Z) and
     XX=i(X) and
     ciklus(XX,ZZ).
```

3. A Turbo Prologban egyenértékűek a következők:

```
a) a(X) if b(X) or c(X).
```

```
b) a(X) if b(X).
   a(X) if c(X).
```

Ez esetben megszakítás nélkül kell, hogy egy csoportban legyenek az alternatívák!

4. A logikai műveletek közül a Prologban fölhasználhatókaz 'and', az 'or' (és a 'not'). Közöttük a megszokott precedenciaviszonyok állnak fönn. Zárójelezni a kifejezéseket nem szabad! Ennek a hiánynak pótlására alkalmazható a 3.-ban mutatott lehetőség. A 'not' nem valódi logikai művelet, hanem egy beépített predikátumtranszformátor, amely a paraméterként megadott predikátumot-függvényt (nem többtagú, vagy többtényezős!) negálja. E transzformátor volta tükröződik abban, hogy az argumentumot zárójelek közé kell zárni.

5. Ha egy szabály (axióma) valamely paramétere nem érdekes a cél szempontjából, akkor ezt az '_' (anonim-paraméter) jellel kell kifejezni!

A Prolog program:

```
/*
 A szabályok (a fenti axiómák után):
*/

anyja(X,Y) if felesege(Z,X) and apja(Z,Y).

/*
 Vegyük észre, hogy az axiómákból származtatható legkézenfekvőbb
 „anyaság” reláció valójában nem biztos, hogy genetikai rokonságot
 is takar. Ennek még lesznek pikáns következményei. Mindjárt az
 első, hogy az apa összes felesége anyja az összes gyermekének.
*/

szuloje(X,Y) if apja(X,Y) or anyja(X,Y).
nagyszuloje(X,Y) if szuloje(X,Z) and szuloje(Z,Y).
ose(X,Y) if szuloje(X,Y) or szuloje(X,Z) and ose(Z,Y).
kiralyne(X) if kiraly(Z,_,_) and felesege(Z,X).
testvere(X,Y) if apja(Z,X) and apja(Z,Y) and X<>Y.

/*
 A furcsa anyaságnak köszönhető, hogy a testvéséghez elegendő az
 apák azonossága. (Hisz ekkor garantált, hogy azonos az anyjuk,
 anyjaik!)
*/

feltestvere(X,Y) if /* X-nek apja, de Y-nak nem */
        anyja(ANYA,X) and anyja(ANYA,Y) and
        apja(APA,X) and not(apja(APA,Y)).

unokatestvere(X,Y) if szuloje(Z1,X) and szuloje(Z2,Y)
        and testvere(Z1,Z2).

uralkodott(X,MIKOR) if fejedelem(X,TOL,IG) and
        TOL<=MIKOR and MIKOR<=IG.

fia_is_uralkodott(X) if uralkodott(X,_) and
        apja(X,Y) and uralkodott(Y,_).

testvere_is_uralkodott(X) if uralkodott(X,_) and
        testvere(X,Y) and uralkodott(Y,_).
```

Próbák (Goal):

```
apja(almos,arpad)eredménye: Yes

apja(almos, levente)            eredménye: No

apja(peter, domonkos)           eredménye: No (de-de!)

felesege(kalman, Ki)            eredménye: 2 solution

  buzilla

  eufemia

anyja("II.istvan", Ki)          eredménye: 2 solution

  buzilla

  eufemia

szuloje(almos, arpad)           eredménye: Yes

szuloje(almos, levente)         eredménye: No

nagyszuloje(almos, levente)     eredménye: Yes
```

```
nagyszuloje(solt, geza)          eredménye: Yes
nagyszuloje(solt, levente)       eredménye: No
ose(almos, levente)              eredménye: Yes
ose(arpad, levente)              eredménye: Yes
ose(arpad, eufemia)              eredménye: Yes
ose(arpad, napoleon)             eredménye: No
fejedelem(almos, TOL, IG)        eredménye: 1 solution
  TOL=819, IG =895
fejedelem(geza, 971, IG)         eredménye: No (nem pontos a TOL!
  helyesen:TOL=970,IG=997)
uralkodott(geza, 971)            eredménye: Yes (TOL=970,IG=997)
uralkodott(napoleon, 971)        eredménye: No (TOL=???,IG=???)
testvere(geza, mihaly)           eredménye: Yes
testvere(levente, endre)         eredménye: Yes (levente: vazsoly
                                                  fia)
feltestvere(levente, endre)      eredménye: No (mert testvére)
unokatestvere(levente, endre)    eredménye: No (mert testvére)
unokatestvere(istvan, vazsoly)   eredménye: Yes (mert geza mihaly
                                                  testvére)
```

2. ÁRPÁDHÁZI KIRÁLYOK „DIAGNOSZTIKÁJA” II.

Válaszoljuk meg a következő kérdéseket:

- ki az, aki sem király sem fejedelem,
- ki az, akinek nincs gyereke?

Továbbá: gondoljuk meg az előzőekben írt szabályokat, alkalmasak-e olyan kérdés megválaszolására, amelyben éppen a személyt keressük!

Megoldás:

A szabályok „terve”:

```
nem_uralkodott(Valaki):=   nem létezik király(Valaki,Tól,Ig) és
          nem létezik fejedelem(Valaki,Tól,Ig).

magtalan(Szülő):=nem létezik Gyerek: szülője(Szülő,Gyerek)
```

Közös a szabályokban, hogy olyan valamit keresünk, ami nem valamilyen. Azaz „prologosítva”:

```
'a(X) if not(b(X)) and c(X).'
```

alakú a szabály.

A probléma: Ha X szabad paraméter (azaz maga a paraméter) kérdéses, akkor azt igényelnénk a Prologtól, hogy keressen olyan X-t, amelyre nem teljesül a 'b' szabály; s erre nem hajlandó. Mondván: „végtelen” halmazban kellene megfelelőt találnia! Ezt a halmazt végesre kell leszűkíteni egy további szabály segítségével, amely vagylagosan fölsorolja a halmaz elemeit (amin már véges lépésben végig lehet menni).

Tehát a megoldás: „változó paraméteres” -a Prolog terminológia szerint: szabad változós-kérdésfeltevés miatt az alaphalmazt kijelölő szabályt is föl kell venni (a deklarációs részben, ill. „alapaxiómaként” az „adatbázis” elé) akkor, ha van olyan szabály, amelyben a szabad paraméter illesztéséhez nincs elegendő információja a Prolognak (pl. tagadással indul, vagy a szabály azon pontjáig jutva marad szabad paraméter, pl.

```
a(x) if b(y) and not(c(x))... )

...

csaladtag(szemely)

nem_uralkodo(szemely)

magtalan(szemely)

...

csaladtag(X)

 if X=almos or X=arpad or X=levente or X=tarhos or X=solt
    or X=taksony or X=geza or X=istvan or X=imre or X=orseolo
    or X=peter or X=mihaly or X=vazul or X=bela or X=endre
    or X=salamon or X="I.geza" or X=kalman or X="I.laszlo"
    or X=levente or X=laszlo or X="II.istvan" or X=laszlo.

csaladtag(X)

 if X=sarolta or X=gizella or X="leany 1" or X="leany2"
    or X="menmarot lany" or X=premiszlava or X=anasztazia
    or X=rixa or X=judit or X=adelhaid or X="görög nő"
    or X=buzilla or X=eufemia.
```

A szabályok:

```
nem_uralkodo(VKi) if csaladtag(VKi) and not(kiraly(VKi,_,_))
            and not(fejedelem(VKi,_,_)).

magtalan(VKi) if csaladtag(VKi) and not(szuloje(VKi,_)).
```

Az eredeti szabályok változtatás nélkül használhatók a személyre vonatkozó kérdésekben.

Próbák:

```
apja(Ki,arpad) eredménye: 1 solution

            Ki=almos

apja(arpad, Ki) eredménye: 3 solutions

             Ki=levente

             Ki=tarhos

             Ki=solt

szuloje(Ki, istvan) eredménye: 2 solutions
             Ki=geza
             Ki=sarolta

nagyszuloje(almos, Kinek) eredménye: 3 solutions
               Kinek=levente
               Kinek=tarhos
               Kinek=solt

ose(Ki, levente) eredménye: 2 solutions
              Ki=arpad
              Ki=almos
```

```
ose(Ki, sarolta) eredménye: No

fejedelem(Ki, 997, IG) eredménye: 1 solution
                Ki=istvan,IG=1001

uralkodott(Ki, 971) eredménye: 1 solution
             Ki=geza

testvere(geza, Kinek) eredménye: Yes
               Kinek=mihaly

feltestvere(levente, Kinek) eredménye: No
     (csak testvére van)

nem_uralkodo(levente) eredménye: No

nem_uralkodo(Ki) eredménye: ? solutions
              Ki=?
              Ki=?
              Ki=?

magtalan(Ki) eredménye: ? solutions
              Ki=?

              Ki=?

              Ki=?
```

A PROLOG FELDOLGOZÁSA (II. RÉSZ)

1. „PORTIA LÁDIKÁJA” ELSŐ FELADATA:

Portiának van három ládikája. Férjhez akar menni, de intelligens férjet szeretne magának. Ezért mielőtt a kérőnek igent mondana, próbára teszi észbeli képességét.

Elrejti egy képét valamelyik ládikába, és mindegyikre egy-egy állítást ír föl, amely a kép hollétére vonatkozik. Majd hozzátesz segítségként egy "peremfeltételt", ami alapján már egyértelműen meg lehet találni a helyes választ.

A konkrét feladvány részletei:

Az arany ládika felirata: „ebben a ládikában van”.
Az ezüst ládika felirata: „nem ebben a ládikában van”.
Az ólom ládika felirata: „nem az arany ládikában van”.
A peremfeltétel: „az állítások közül legfeljebb egy igaz”.

Írjunk tehát a kérő megsegítésére olyan PROLOG programot, amely egyszerűen felkészíthető tetszőleges állítások és peremfeltétel melletti megoldás megtalálására!

Megoldások:

Első meggondolni való a támpont. Mi legyen a támpont, amelyből szabályt, vagy axiómát lehet csinálni? A válasz: a ládákra írt állítások. Mi érdekes bennünk, és hogyan fogalmazhatók meg a predikátumként? „Állítás-attributumok”: - melyik ládára van írva (hiszen ezt fölhasználjuk a „peremfeltételekben”), - az állítása szerint: melyik ládában (vagy ládákban) található a kép.

Második dolog, ami az előzőek után nem váratlan az az, hogy mivel előreláthető olyan predikátum léte, amely tagadással indul (s mint tudjuk ehhez nem tud a PROLOG mintát illeszteni), ezért kell egy axiómasorozat támpontul. Ezek arról "szólnak", hogy milyen paraméterek, azaz ládák lehetnek egyáltalán.

1. megoldás:

Így adódik az első lehetőség a predikátumra, ami egyben tényállítás is lehet (hisz az kétségtelen, hogy ez a ládára van írva; azt persze nem állítjuk, hogy igaz is):

```
állítás(melyiken,miben).

/*
 Portia ládikája 67a. kérdése:
*/

Predicates
 lada(Symbol) % ládaféleségek a tagadáshoz
 allitas(Symbol,Symbol) % a ládán lévő állítás
 mind_igaz(Symbol) % mindegyik felirat igaz
 legf_1_igaz(Symbol) % legfeljebb 1 igaz a feliratok közül
 legf_1_hamis(Symbol) % legfeljebb 1 hamis a feliratok közül
 mind_hamis(Symbol) % mindegyik felirat hamis

Clauses
 lada(X) If X="arany" or X="ezust" or X="olom".
```

```
  /* a szószerint vett állítás az arany ládán:
allitas(Ladan,Ladaban) If Ladan=arany and Ladaban=Ladan.

  /* ugyanez egyszerűbben: */
allitas(arany,arany).

  /* a szószerint vett állítás az ezüst ládán:
allitas(Ladan,Ladaban) If Ladan=ezust and Ladaban<>Ladan.

  /* ugyanez egyszerűbben: */
allitas(ezust,arany).
allitas(ezust,olom).

  /* a szószerint vett állítás az ólom ládán:
allitas(Ladan,Ladaban) If Ladan=olom and Ladaban<>arany.

  /* ugyanez egyszerűbben: */
allitas(olom,ezust).  /*
allitas(olom,olom).

  /* a lehetséges "peremfeltételek": */

mind_igaz(X) If allitas(arany,X) and allitas(ezust,X)
        and allitas(olom,X).

legf_1_hamis(X) If allitas(ezust,X) and allitas(olom,X).

legf_1_hamis(X) If allitas(arany,X) and allitas(olom,X).

legf_1_hamis(X) If allitas(ezust,X) and allitas(arany,X).

legf_1_igaz(X) % pontosan 1 igaz
  If allitas(ezust,X) and not(allitas(olom,(X)) and
     not(allitas(arany,X)) or allitas(arany,X) and
     not(allitas(ezust,(X)) and not(allitas(olom,X)) or
     allitas(olom,X) and not(allitas(arany,X)) and
     not(allitas(ezust,X)).

legf_1_igaz(X) % 1 sem igaz
  If mind_hamis(X).

mind_hamis(X) If lada(X) and not(allitas(arany,X)) and
     not(allitas(ezust,X)) and not(allitas(olom,X)).

Goal

  /* a konkrét kérdés célállítása: */

Write("A 67.a) rejtvény megoldása:") and nl and
     legf_1_igaz(X) and Write(X)
```

2. megoldás:

A két fenti "állításattributumot" másként is vissza lehet adni. Az elv, amit gyakran alkalmazunk a Prologban: az egyik attributumot relációként tekitjük, s általa nevezzük el a szabályt, a másik (a többi) pedig ennek paramétere(i).

```
attributum1(attributum2,...)
```

```
/*
Portia ládikája 67a. kérdése:
*/

Predicates
 lada(Symbol) % ládaféleségek a tagadáshoz
 arany_ladan(Symbol) % arany láda felirata
 ezust_ladan(Symbol) % ezüst láda felirata
```

```
 olom_ladan(Symbol) % ólom láda felirata
 mind_igaz(Symbol) % mindegyik felirat igaz
 legf_1_igaz(Symbol) % legfeljebb 1 igaz a feliratok közül
 legf_1_hamis(Symbol) % legfeljebb 1 hamis a feliratok közül
 mind_hamis(Symbol) % mindegyik felirat hamis

Clauses
 lada(X) If X="arany" or X="ezust" or X="olom".

   /* a szószerint vett állítás az arany ládán: */
 arany_ladan(X) If X=arany.

   /* ugyanez egyszerűbben: */
 arany_ladan(arany).

   /* a szószerint vett állítás az ezüst ládán: */
 ezust_ladan(X) If X<>ezust.

   /* ugyanez egyszerűbben: */
 ezust_ladan(arany).
 ezust_ladan(olom).

   /* a szószerint vett állítás az ólom ládán: */
 olom_ladan(X) If X<>arany.

   /* ugyanez egyszerűbben: */
 olom_ladan(ezust).
 olom_ladan(olom).

 /* a lehetséges "peremfeltételek": */

 mind_igaz(X) If arany_ladan(X) and ezust_ladan(X) and
          olom_ladan(X).

 legf_1_hamis(X) If ezust_ladan(X) and olom_ladan(X).

 legf_1_hamis(X) If arany_ladan(X) and olom_ladan(X).

 legf_1_hamis(X) If ezust_ladan(X) and arany_ladan(X).

 legf_1_igaz(X) % pontosan 1 igaz

   If ezust_ladan(X) and not(olom_ladan(X)) and
      not(arany_ladan(X)) or arany_ladan(X) and
      not(ezust_ladan(X)) and not(olom_ladan(X)) or
      olom_ladan(X) and not(arany_ladan(X)) and
      not(ezust_ladan(X)).

 legf_1_igaz(X) % 1 sem igaz

   If mind_hamis(X).

 mind_hamis(X) If lada(X) and not(arany_ladan(X)) and
       not(ezust_ladan(X)) and not(olom_ladan(X)).

 Goal

 /* a konkrét kérdés célállítása: */

 Write("A 67.a) rejtvény megoldása:") and nl and
 legf_1_igaz(X) and Write(X)
```

Futási eredmény: ezüst.

2. „PORTIA LÁDIKÁJA” MÁSODIK FELADATA

... most már „rafináltkodik” Portia. (Könnyűnek tűnt talán a fel-adványa.) Minden ládi-kára két állítást ír, amelyek vonatkozhatnak akár arra, hogy melyik ládában található a kép, de a kép festőjére is.

A konkrét feladvány részletei:

Az arany ládika feliratai:

- „nem ebben a ládikában van”,
- „velencei a festő”.

Az ezüst ládika feliratai:

- „nem az arany ládikában van”,
- „firenzei a festő”.

Az ólom ládika feliratai:

- „nem az ólom ládikában van”,
- „az ezüst ládikában van”.

A peremfeltétel: „az állítások közül ládánként legfeljebb egy hamis”.

Írjunk tehát a kérő megsegítésére olyan PROLOG programot, amely egyszerűen felkészíthető tetszőleges állítások és peremfeltétel melleti megoldás megtalálására!

Megoldás:

```
 /*
 Portia ládikája 68a. kérdése:
 */

Predicates
 lada(Symbol) % ládaféleségek
 festo(Symbol) % festők
 arany1(Symbol) % arany láda 1. felirata
 ezust1(Symbol) % ezüst láda 1. felirata
 olom1(Symbol) % ólom láda 1. felirata
 arany2(Symbol) % arany láda 2. felirata
 ezust2(Symbol) % ezüst láda 2. felirata
 olom2(Symbol) % ólom láda 2. felirata
 legf_1_hamis_arany(Symbol,Symbol)
 legf_1_hamis_ezust(Symbol,Symbol)
 legf_1_hamis_olom(Symbol,Symbol)
 kerdes_teljes(Symbol,Symbol) % ha festőre is kíváncsiak vagyunk
 kerdes(Symbol)

Clauses
 lada(X) If X="arany" or X="ezust" or X="olom".
 festo(X) If X="velence" or X="firenze".
 arany1(ezust).
 arany1(olom).
 arany2(velence).
 ezust1(ezust).
 ezust1(olom).
 ezust2(firenze).
 olom1(arany).
 olom1(ezust).
 olom2(ezust).
 legf_1_hamis_arany(X,Y) If arany1(X) or festo(Y) and arany2(Y).
 legf_1_hamis_ezust(X,Y) If ezust1(X) or festo(Y) and ezust2(Y).
```

```
 legf_1_hamis_olom(X,Y) If olom1(X) or olom2(X).
 kerdes_teljes(X,Y) If lada(X) and legf_1_hamis_arany(X,Y) and
 legf_1_hamis_ezust(X,Y) and legf_1_hamis_olom(X,Y).
 kerdes(X) If lada(X) and legf_1_hamis_arany(X,Y) and
     legf_1_hamis_ezust(X,Y) and legf_1_hamis_olom(X,Y).

 Goal
 /* a konkrét kérdés célállítása: */

 Write("A 68a. rejtvény megoldása:") and nl and
 kerdes(X) and Write(X)
```

Futási eredmény: ezüst.

3. SUBIDAM ÉS SUBIDU

Subidam és Subidu hazudósak. E tulajdonságuk nap fajtájától függően változik. Pl. Subidam csütörtökön, pénteken, szombaton és vasárnap mond igazat; Subidu hétfőn, kedden, szerdán és vasárnap.

Írjunk Prolog programot annak eldöntésére, hogy ma milyen nap van, ha kettejükkel találkozva a következők hangzanak el szájukból:

Egyik: „Subidam vagyok”.
Másik: „Subidu vagyok”.

Megoldás:

Mivel „egybehangzóan” nyilatkoznak, vagy mindketten igazat mondanak, vagy mindketten hazudnak.

```
/*
 Subidam és Subidu (majdnem az 51. feladat):
*/

Domains
 Nap = Symbol
 Kicsoda= Symbol

Predicates
 nap(Nap)
 subiduIgaz(Nap)
 subidamIgaz(Nap)
 mindkettenHazudnak(Kicsoda,Kicsoda,Nap)
 mindkettenIgazatmondanak(Kicsoda,Kicsoda,Nap)

Clauses
 nap(N) If N=hetfo or N=kedd or N=szerda or N=csutortok
        or N=pentek or N=szombat or N=vasarnap.

 subidamIgaz(Nap) If Nap=csutortok or Nap=pentek or Nap=szombat
        or Nap=vasarnap.

 subiduIgaz(Nap) If Nap=hetfo or Nap=kedd or Nap=szerda
        or Nap=vasarnap.

 mindkettenIgazatmondanak(subidam,subidu,Z) If subidamIgaz(Z) and
        subiduIgaz(Z).

 mindkettenHazudnak(subidu,subidam,Z) If nap(Z) and
        not(subidamIgaz(Z)) and not(subiduIgaz(Z)).

Goal
```

```
 mindkettenHazudnak(EloszorSzol,MasodszorSzol,Nap) or
 mindkettenIgazatmondanak(EloszorSzol,MasodszorSzol,Nap)
```

Futási eredmény:

```
ElőszörSzól =Subidam

MásodszorSzól=Subidu

Nap =vasárnap.
```

A PROLOG FELDOLGOZÁSA (III. RÉSZ)

Tételek „prologosítása” (általános feladat): az adott tételt fogalmazzuk meg először matematikai, függvényes jelöléssel, majd adjuk meg a Prolog programot is!

1. Összegzés tétel
2. Eldöntés tétel
3. Keresés tétel
4. Kiválogatás tétel
5. Megszámlálás tétele
6. Maximum kiválasztás tétele
7. Rendezés tétel
8. Logaritmikus "eldöntés" tétele (rendezettben benne van-e egy adott elem)

Megoldás:

A tételeket egészek vagy szimbólumok sorozatán értelmezzük. A tételek „hagyományos” értelemben vett eredménye (kimenő paraméterei) most többletparaméterként illesztjük a tételek predikátumaihoz. Amint ez az alábbiakban látszik:

```
1. Összegzés(EgészSorozat->Egész [összeg])
2. Eldönt(SzimbólumSorozat) [sikeresség]
3. Keres(SzimbólumSorozat->Egész [hányadik elem]) [sikeresség]
4. Kiválogat(SzimbólumSorozat->SzimbólumSorozat [T-tul. elemek])
5. Megszámlál(SzimbólumSorozat->Egész [T-tulajdonságúak száma])
6. Maximum(SzimbólumSorozat->Elem [legnagyobb])
7. Rendez(SzimbólumSorozat->SzimbólumSorozat [rendezve])
   (minimumkiválasztással)
8. LogEldönt(Bináris fa,Elem) [sikeresség]
```

Megjegyzések:

1. egy újdonságot rejt magába a logaritmikus eldöntés: a bináris fa rekurzív típusát. Erre azért van szükség, mert ilyenben képes valóban hatékonyan keresni a Prolog. Ezzel kapcsolatban fölmerül egy kiegészítő feladat: írjunk predikátumot, amely egy közönséges sorozatot bináris fává alakít át! (Ehhez lehet szükség pl. az 'AVLfa', a 'szintszám' és a 'max' predikátumokra.)
2. a maximumkiválasztásos rendezés segéd predikátumként igényel egy 'kihagy' nevűt.)

A tulajdonságot egy 'tul' predikátum fejezi ki. (Ez most konkrétan 'a szóközzel kezdődik'-t jelenti.)

```
Domains
 EgeszSorozat = Integer*
 Elem = Symbol
 SzimbolumSorozat= Elem*
 BinFa = bf(Binfa,Elem,Binfa); ures % bin.fa típus

Predicates
 osszeg(EgeszSorozat,Integer)
 eldont(SzimbolumSorozat)
```

```
 keres(SzimbolumSorozat,Integer)
 kivalogat(SzimbolumSorozat,SzimbolumSorozat)
 megszamlal(SzimbolumSorozat,Integer)
 maximum(SzimbolumSorozat,Elem)
 rendez(SzimbolumSorozat,SzimbolumSorozat)
 kihagy(SzimbolumSorozat,Elem,SzimbolumSorozat)
 logeldont(BinFa,Elem)
 fasit(SzimbolumSorozat,BinFa)
 beilleszt(BinFa,Elem,BinFa)
 AVLfa(BinFa,BinFa)
 szintszam(BinFa,Integer)
 max(Integer,Integer,Integer)
 tul(Elem)

Clauses

 tul(Elem)

 If frontchar(Elem,Elso,_) and Elso=' '. % ' '-zel kezdődő

 osszeg([],0).

 osszeg([Elem|Sorozat],S) If osszeg(Sorozat,T) and S=Elem+T.

 eldont([]) If fail.

 eldont([Elem|Sorozat]) If tul(Elem) or eldont(Sorozat).

 keres([],_) If fail.

 keres([Elem|Sorozat],Hol) If tul(Elem) and Hol=1.

 keres([Elem|Sorozat],Hol) If keres(Sorozat,NaHol) and
    Hol=NaHol+1 and ! /* '!' mivel csak az 1. kell! */.

 kivalogat([],[]).

 kivalogat([Elem|BeSorozat],[Elem|KiSorozat]) If tul(Elem) and
    kivalogat(BeSorozat,KiSorozat) and !.

 kivalogat([Elem|BeSorozat],KiSorozat) If kivalogat(BeSorozat,
    KiSorozat) and !.

 megszamlal([],0).

 megszamlal([Elem|BeSorozat],Db) If tul(Elem) and megszamlal(Be
Sorozat,Db1) and Db=Db1+1 and !.

 megszamlal([Elem|BeSorozat],Db) If megszamlal(BeSorozat,Db) and
    !.

 maximum([],_) If fail.

 maximum([Elem],Elem).

 maximum([Elem|Sorozat],Melyik) If maximum(Sorozat,TalanEz) and
    Elem>TalanEz and Melyik=Elem and ! or maximum(Sorozat,
    TalanEz) and Elem<=TalanEz and Melyik=TalanEz and !.

 rendez([],[]).

 rendez([Elem],[Elem]) If !. %nem kell több megoldás!

 rendez(BeSorozat,[Elem|KiSorozat]) If maximum(BeSorozat,Max) and
    Elem=Max and kihagy(BeSorozat,Max,UjSorozat) and
    rendez(UjSorozat,KiSorozat) and !.

 kihagy([],Nelkul,[]).

 kihagy([Nelkul|BeSorozat],Nelkul,BeSorozat) If !.
```

```
 kihagy([Elem|BeSorozat],Nelkul,[Elem|KiSorozat])
    If kihagy(BeSorozat,Nelkul,KiSorozat) and !.

 logeldont(ures,_) If fail.

 logeldont(bf(Bal,Elem,Jobb),Elem).

 logeldont(bf(Bal,Gyoker,Jobb),Elem) If Gyoker>Elem and
    logeldont(Bal,Elem) or Gyoker<Elem and logeldont(Jobb,Elem).

/* az alábbiak meggondolandók: */

 fasit([],ures).

 fasit([Elem],bf(ures,Elem,ures)).

 fasit([Elem|Sorozat],X) If fasit(Sorozat,Fa) and
    beilleszt(Fa,Elem,X) and !.

 beilleszt(ures,Elem,bf(ures,Elem,ures)).

 beilleszt(bf(ures,K,Jobb),Elem,bf(bf(ures,Elem,ures),K,Jobb))
    If Elem<K and !.

 beilleszt(bf(Bal,K,ures),Elem,bf(Bal,K,bf(ures,Elem,ures)))
    If Elem>=K and !.

 beilleszt(bf(Bal,K,Jobb),Elem,bf(UjBal,K,Jobb))
    If Elem<K and beilleszt(Bal,Elem,UjBal) and !.

 beilleszt(bf(Bal,K,Jobb),Elem,bf(Bal,K,UjJobb))
    If Elem>=K and beilleszt(Jobb,Elem,UjJobb) and !.

 szintszam(ures,0).

 ...

 AVLfa(ures,ures).

 AVLfa(..,..)
    If szintszam(bf(BB,BGy,BJ))>=szintszam(bf(JB,JGy,JJ))-1 or
    szintszam(bf(BB,BGy,BJ)<=szintszam(bf(JB,JGy,JJ))+1.

 szintszam(bf(Bal,_,Jobb,Magassag) If szintszam(Bal,BalMag) and
    szintszam(Jobb,JobbMag) and max(BalMag,JobbMag,MaxMag) and
    Magassag=MaxMag+1.

 max(X,Y,Max) If Max=X and X>Y or Max=Y and X<=Y.

*/
```

Futási eredmények (Goal):

```
osszeg([],0) eredménye: 0

osszeg([1,2,3,4],0) eredménye: 10

eldont([]) eredménye: No

eldont(["1","2","3"]) eredménye: No

eldont(["1"," bármi","3"]) eredménye: Yes

keres(["1"," bármi","3"],Hol) eredménye: 1 solution

             Hol=2

keres(["1","akármi","3"],Hol) eredménye: No

kivalogat(["1","a","3","B"],Mik) eredménye: No

kivalogat(["1"," a","3"," B"],Mik) eredménye: 1 solution
```

```
            Mik=[" a"," B"]
megszamlal(["1"," a","3"," B"],Db) eredménye: 1 solution
            Db=2
megszamlal(["1","a ","3","BB"],Db) eredménye: 1 solution
            Db=0
maximum(["BB"],Max) eredménye: 1 solution
            Max="BB"
maximum(["1","a ","3","BB"],Max) eredménye: 1 solution
            Max="a "
kihagy([a,b,c,abc,a,b,c],a,Nelkuli) eredménye: 1 solution
            Nélküli=[b,c,abc,b,c]
kihagy([abc,bc,ab,c],a,Nelkuli) eredménye: 1 solution
            Nélküli=[abc,bc,ab,c]
rendez(["a"],Rend) eredménye: 1 solution
            Rend=["a"]
rendez(["1","a ","3"],Rend) eredménye: 1 solution
            Rend=["1","3","a "]
rendez(["1","a ","3"],["1"|Foly]) eredménye: 1 solution
            Foly=["3","a "]
logeldont(bf(bf(ures,a,ures),c,ures),c) eredménye: Yes
logeldont(bf(bf(ures,a,bf(ures,c,ures)),b,ures),c) eredménye: No
    (ui. rossz a bf: nem rendezett)
fasit([a,b,c],BF) eredménye: 1 solution
          BF=bf(bf(bf(ures,a,ures),b,ures),c,ures)
fasit([b,a,c],BF) eredménye: 1 solution
          BF=bf(bf(ures,a,bf(ures,b,ures)),c,ures)
```

A PROLOG FELDOLGOZÁSA

(IV. RÉSZ)

Feladat: írjunk tanuló programot, amely képes ismerőseit megállapodás, ill. a szokásos üdvözlési formulák szerint üdvözölni! Azaz a következő elképzelés alapján működik:

1. Miután a kezdő tudását beolvasta fájlból (ez a saját nevével megegyező DBF-fájlból történik), azután
2. tisztázza, hogy hány óra van. Erre amiatt van szükség, mivel bizonyos esetekben a napszak függvényében kell köszönnie (pl. „Jó reggelt!", vagy „Jó éjszakát!").
3. Ezt követően a indul „normál" tanuló ciklus, vagyis kétféle események sora jön.
   3.1. Vagy ember jön, akit vagy ismer már vagy nem.
      3.1.1. Ha ismeri, akkor a megállapodás szerinti üdvözlést válaszolja az érkező köszöntésére;
      3.1.2. ha nem ismerné, akkor megismerkedik vele, s följegyzi az ismereteit (mi a neve, hány éves, milyen nemű, hogyan kell köszöntenie), majd köszönti is.
   3.2. Előfordulhat, hogy időállítás következik. Ekkor be kell kérnie az aktuális óraállást. (Ebből derül ki számára a napszak, vagyis a „Jó reggelt!"-stílusú köszöntésekre adandó visszaköszönés. (Nem muszáj kiröhögni az érkezőt, ha az reggel pl. „Jó estét"-tel köszön, de lehet.)

4. A ciklus 0 órakor ér véget. Ekkor rögzíti az aktuális tudását fájlba.

További lehetőségek:

- Az emberek neme ismert, tehát „formális" köszönő formulák esetén meglehet toldani azt „Hölgyem!", vagy „Uram!" megszólítással.
- Ha az ember szó nélkül bukkan föl, akkor szóvá lehet tenni az udvariatlanságot.
- Hosszabb párbeszédet lehet kezdeményezni a szituációtól függően.
- Reagálni lehet a megállapodástól eltérő köszönésre. (Pl. valakivel „Sziá"-ban állapodott meg, de egyszercsak az „Jó reggelt" mond.)

Tudnivalók a Turbo Prolog-hoz:

A belső adatbáziskezeléshez:

- az DATABASE szekcióba kell a PREDICATES helyett az adatbázisban előforduló predikátumokat deklarálni.
- az adatbáziskezelés műveletei:
  = fölvenni új predikátumot: Assert/Asserta (elejére)/Assertz (végére)
  = elhagyni egy predikátumot: Retract/RetractAll (az összes olyat)
  = adatbázis és fájl kapcsolata: Consult/Save
- általában a FÁJLokkal kapcsolatban:
  = ExistFájl
- az ablakszervezéshez:
  = MakeWindow(AS,Hat,Tin,Cim,BY,BX,Mag,Szel), ahol
  AS=ablaksorszám, Hat=háttérszín, Tin=tintaszín,
  Cim=címszöveg, BY=balfelső sarok y-, BX=bal felső sarok x-koordinátája,
  Mag=ablakmagasság, Szel=ablakszélesség
  = ExistWindow(AS)

= GotoWindow(AS)
= RemoveWindow(AS)

Egy „nyers” megoldás:

```
Domains
 Nev = Symbol
 Nem = Char
 Kor,Ora = Integer
 Napszak = Reggel or Delelott or Delutan or Este or Ejszaka
Database
 Szemely(Nev,Nem,Kor)
 En(Nev,Nem,Kor)
 Koszones(Nev,String)
 Ido(Ora)
Predicates
 Napszak(Ora,Napszak)
 MemoriaFeltoltes
 FájlMegnyitas(String)
 Memorizalas
 FoAblak(Ora)
 Ablak(Integer,Integer,Integer,String,Integer,Integer,Integer,
Integer)
```

Az alábbi predikátumokat kell a gyakorlaton önállóan megtervezni:

```
 Parbeszed(Ora)
 Jon(String,Ora)
 UjIdo(Ora)
 IdoCsere(Ora)
 Te(Nev)
 Mondd(Nev,String)
 Valasz(Nev,String)
 KiVagy(Nev,Nem,Kor)
 KiVagyok(Nev,Nem,Kor)
 Megszolitas(Nem,String)
```

Az itt kezdődő predikátumokat minta gyanánt előre mellékeljük:

```
Clauses
 FájlMegnyitas(FN) If Concat(FN,".dba",Fájl) and ExistFájl(Fájl)
    and Consult(Fájl) and Retract(en(_,_,_)).
 FájlMegnyitas(_).
 Ablak(A,Hat,Tin,Cim,BY,BX,Mag,Szel) If ExistWindow(A) and
    GotoWindow(A) and ClearWindow and ! or MakeWindow(A,Hat,Tin,
    Cim,BY,BX,Mag,Szel).
```

```
KiVagyok(NaNev,NaNem,Kor) If Ablak(2,112,4," Ez vagyok én. "
   ,19,50,5,29) and Write("Mi a nevem?:") and Readln(Nev) and
   Upper_Lower(NaNev,Nev) and Write("Mi a korom?:") and
   Readint(Kor) and Write("Mi a nemem?:") and Readchar(Nem) and
   Upper_Lower(NaNem,Nem) and Write(NaNem) and UjIdo(O) and
   FoAblak(O).

MemoriaFeltoltes If RetractAll(_) and KiVagyok(Nev,Nem,Kor) and
FájlMegnyitas(Nev) and Asserta(en(Nev,Nem,Kor)).

Memorizalas If En(Nev,_,_) and Concat(Nev,".dba",Fájl) and
   Save(Fájl).

FoAblak(0) If MakeWindow(1,2,1," Köszöngetés ",0,0,25,80) and
   Ablak(7,112,4," Idő ",1,50,3,29) and Napszak(0,NSz) and
   Write("óra: ",0) and Write("\tNapszak: ",NSz) and
Ablak(6,2,3," Történés ",1,1,23,25).

FoAblak(O) If GotoWindow(7) and Napszak(O,NSz) and
   Write("\nóra: ",O) and Write("\tNapszak:",NSz) and
   GotoWindow(6) and !.
```

Az alábbi predikátumokat már önállóan kell megtervezi:

```
Megszolitas('F',"Uram").

Megszolitas('N',"Hölgyem").

Megszolitas(_,"\nZavarban vagyok, mi vagy: fiú vagy lány?").

napszak(O,ejszaka) If O>=22 and ! or O<4 and !.

napszak(O,reggel) If O>=4 and O<8 and !.

napszak(O,delelott) If O<12 and !.

napszak(O,delutan) If O<18 and !.

napszak(O,este) If O<22 and !.

Parbeszed(0) If !. %vége

Parbeszed(_) If Write("\nMi következik?\n (Valaki vagy az
   Idő)\n") and Readln(Mi) and Upper_Lower(NaMi,Mi) and
   Jon(NaMi,Mikor) and Parbeszed(Mikor).

Jon(Mi,Mikor) If Concat("IDO",Tobbi,Mi) and Str_Len(Tobbi,H) and
   H=0 /*"Idő"*/ and UjIdo(Mikor) or Concat("I",Tobbi,Mi) and
   Str_Len(Tobbi,H) and H=0 /* "I" */ and UjIdo(Mikor) or
   Str_Len(Mi,H) and H=0 and ido(Mikor) or Te(Mi) and ido(Mikor).

IdoCsere(O) If RetractAll(ido(_)) and Assertz(ido(O)) and ! or
   Assertz(ido(O)).

UjIdo(Mikor) If Ablak(3,2,5," óraállítás ",1,30,3,49) and
   Write("Hány óra van? (0..23, 0: Vége): ") and Readint(Mikor)
   and RemoveWindow /* le kell venni, itt két ablakis van */ and
   IdoCsere(Mikor) and FoAblak(Mikor).

Te(Ki) If Szemely(Ki,Nem,Kor) /*már ismerem*/ and Ablak(8,112,4,
   " Ez vagy Te. ",9,50,5,29) and Write("\nA neved:",Ki) and
   Write("\nA korod:",Kor) and Write("\nAnemed:",Nem) and
   Mondd(Ki,Mit) and Upper_Lower(NaMit,Mit) and Valasz(Ki,NaMit)
   and ! or KiVagy(Ki,Nem,Kor) /*ismerkedés*/ and
   Assert(szemely(Ki,Nem,Kor)) and Te(Ki) and !.
```

```
 Mondd(Ki,Mit) If Ablak(9,2,5," Te mondd! ",5,30,4,49) and
    Write(Ki,",mit mondasz? (Köszönés)\n") and Readln(Mit) and
    Ido(O) and FoAblak(O).

 Valasz(Kinek,Mit) If En(Magam,_,_) and Ablak(4,2,5,Magam,14,30,4,
    49) and Concat("JO",_,Mit) and Ido(O) and Napszak(O,NSz) and
    Write("Jó",NSz,"t,") and Szemely(Kinek,Nem,_) and
Megszolitas(Nem,Megsz) and Write(" ",Megsz,"!") and Ido(O) and
FoAblak(O) and ! or Str_Len(Mit,H) and H=0 and
 Write("Bunkó! Miért nem köszönsz?\n") and fail or
koszones(Kinek,Udv) and Write(Kinek,",",Udv,"!") and Ido(O) and
    FoAblak(O) and ! or Write("Hogy köszöntselek?") and
    Readln(Igy) and Upper_Lower(NaIgy,Igy) and
    Assert(koszones(Kinek,NaIgy)) and Write("\nHát akkor...") and
    Valasz(Kinek,Mit) and Ido(O) and FoAblak(O) and !.

 KiVagy(Ki,NaNem,Kor) If Ablak(5,2,5," Te mondd! ",5,30,4,49) and
    Write(Ki,", mi a korod?:") and Readint(Kor) and
    Write("Mi a nemed?:") and Readchar(Nem) and
    Upper_Lower(NaNem,Nem) and Write(NaNem) and
    Ido(O) and FoAblak(O).

Goal

 /* adatbázisföltöltés: */

 FoAblak(0) and MemoriaFeltoltes and

 /* "párbeszélgetés": */

 Ido(O) and parbeszed(O) and

 /* a szerzett tapasztalatok kivitele: */

 Memorizalas
```

A PROLOG FELDOLGOZÁSA
(TURBO PROLOG ÖSSZEFOGLALÓ)

PROGRAMSZERKEZET:

```
Constant
 konstans = predikátum(par1,par2,...)       konstansmegadás
/* elhagyható szekció */
Domains
/* a predikátumok értékhalmazainak  definiálása: */
Goal
 típus = típusdefiníció /* a célállítás ide irandó; ez a szekció
    el is hagyható, vagy a 'Predicates' elé is helyezhető */
Predicates
/* predikátumok értelmezési tartományának definiálása: */
 predikátum(típus1,típus2,...)
Database
/* a "dinamikusan" kezelendő tudás: szabályok */
 predikátum(típus1,típus2,...).
/* elhagyható szekció; de ha van, akkor nem lehet közös része a
   'Predicates'-szel */
Clauses
/* a predikátumok összefüggései: */
Constant
 pi = 3.141592653
 napok= [hétfő,kedd,szerda,csütörtök,péntek,szombat,vasárnap]
Domains
 személy = string
Clauses
 év,hó,nap= integer
/* fölsorolástípus: */
 király(én,d(55,8,6),d(93,5,1 ládák = arany; ezüst; )).
ólom /* sorozatszelekció:
/* direktszorzat; [1. elem,2.
d:="funktor": */ elem,... |
 dátum = d(év,hó,nap) maradék
/* unió: */ sorozat] */
```

```
 kiadvány = könyv(string); eldönt(Ki,[Ki|_]).
 újság(string) eldönt(Ki,[_|Többi]) if
/* sorozatképzés: '*'-jellel eldönt(Ki,Többi).
*/ keres(Ki,f(_,Ki,_)) if !.
 sor = személy* keres(Ki,f(B,VKi,J)) if
/* rekurzió: */ VKi>Ki and keres(Ki,B)
 fa = f(fa, személy, fa); or
 üres VKi<Ki and. keres(Ki,J).
Predicates ...
 Goal
király(személy,dátum,dátum) clearwindow and beep and
 eldönt(személy,sor) keres(1,f(
 keres(személy,sor) f(üres,1,üres),
/* "többszörös" 2,
paraméterezés: */ f(üres,3,üres)),
 keres(személy,fa) ) and
... write('Van') and nl and
Database readchar(_) and !
 ős(személy,személy) or write('Nincs').
```

LEGFONTOSABB KULCSSZAVAK

Rövidítések:

int :=integer
str :=string
etip:=elemtípus (tetszőleges, definiált típus)
tip :=típus-azonosító

1. Standard I/O:

inkey(char)
keypressed
readchar(char)
readint(int)
readln(str)
readreal(real)
readterm(tip,etíp)
clearwindow
makewindow(int,int,int,str,int,int,int,int)
cursor(int,int)
nl
write(etip,etip,...)

2. Grafika:

initgraph(int,int,int,int,str)
detectgraphint,int)
closegraph
bar(int,int,int,int)
drawpoly(int,int,...)
getmaxx(int)
...
rectangle(int,int,int,int)
...
...BGI...

3. Belső és külső adatbázis:

assert(etip)
asserta(etip)
assertz(etip)
retract(etip)
retractall(etip)
consult(str)
save(str)
...

4. Fájlkezelés:

openappend(fájl,str)
openread(fájl,str)
openmodify(fájl,str)
closefájl(fájl)
eof(fájl)
existfájl(str)

5. Függvények, operátorok:

char_int(char,int)
str_char(str,char)
str_int(str,int)
str_real(str,real)
abs(real)
arctan(real)
cos(real)
exp(real)
ln(real)
round(real)
sin(real)
sqrt(real)
tan(real)
trunc(real)
+, -, *, /, mod, div, >, >=, =, <=, <, <>

6. Egyebek:

fail
true
!
beep
sound(int,int)

date(int,int,int)
time(int,int,int,int)
random(real)
random(int,int)

Használati összefoglaló:

Pogramszerkesztésbe kezdeni: <ALT><E>

Futtatni a programot: <ALT><R>

A „futtató ablakban” az előző célállítás megismétlése: F8

Funkcióbillentyű +<SHIFT> +<CTRL>

F1: Help Help-fájl

F2: Save fájl Ugrás adott Ugrás adott pozícióra sorra

F3: Load fájl

Újrakeresés Keresés

F4: Keresés és csere Újrakeresés és -

F5: Zoomolás másolás

Blokkmásolás

F6: Ablakcsere Újramásolás

(ciklikusan) Nagy-kisbetű

F7: Blokkmásolás csere

fájlból blokkban

F8: Segédszerkesztő Visszacsinálás

ablak

F9: Fordítás

F10:Kilépés a Fordítás '.OBJ'- Fordítás '.EXE'- szerkesztőből ba be

Ablak- újraméretezés

Programszerkesztés: a Turbo-környezetben megszokott módon (pl. Turbo Pascal 5.0).

Nyomkövetés:

1. a forrásprogramba el kell helyezni 'Trace' (vagy 'ShortTrace') direktívát; ha csak bizonyos predikátumokat akarunk figyelni, akkor azt vagy azokat föl kell sorolni a 'Trace' után, ill. ennél "finomabb" nyomkövetés végezhető el a 'Trace(on)' és 'Trace(off)' beépített predikátumokkal,
2. fordítani és futtatni,
3. F10-zel lépésenként követni,
4. ha a további futás nyomkövetés nélkül mehet, akkor <ALT><T> és 'Trace off' beállítása (az <ENTER>-rel).