Programozási készségek fejlesztése az ImagineLogo programmal

(A helyes elrendezés megjelenítéséhez, kérjük a Word webes elrendezésének használatát.)

Logo bemutatása, telepítése

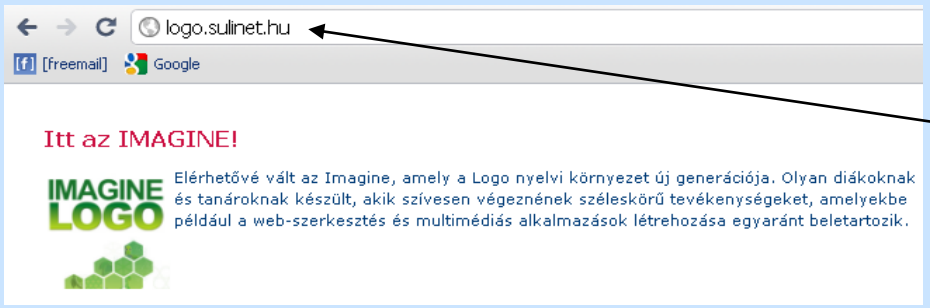
A Logo története 1966-ig nyúlik vissza, amikor két kutató SeymourPapert és Wallace Feuerzieg, mindketten a Massechusetts Institute of Technology munkatársai, a konstruktív, azaz felfedezésen alapuló tanulás segítésére megalkották a Logo programozási nyelvet. Papert öt évet töltött Piagetgenfi Genetikus Ismeretelméleti Központjában, s az itt szerzett tapasztalatok alapján kezdett saját kutatásaiba. Piaget fontosnak tartotta, hogy a tanuló maga építse fel a gondolati struktúrákat, vagyis ne azon legyen a hangsúly, amit a diáknak meg kell tanulnia, hanem a természetes spontán tanuláson. A Logo egy olyan környezetet, mikrovilágot biztosít a tanulók számára, ahol saját maguk tehetnek felfedezéseket, és sajátíthatnak el új ismereteket. A Logo nyelv fejleszti a gondolkodást, segíti a matematikai alapkészségek elsajátítását, a nyelvi eszközök fejlődését, és ráadásul szórakoztató módon segíti az ismeretszerzést, valamint kitartásra és önbizalomra nevel.

A számtalan Logo generáció közül, ami a 60-as évek óta a világon megjelent, az ImagineLogo vonzó lehet, hiszen ingyenesen elérhető és letölthető a magyar oktatási intézmények számára. A multimédiás eszközökben is jelentős korszerűsítést végeztek a készítők: Andrej Blaho, Ivan Kalas, Tomcsányi Péter és LubomírSalanci, a pozsonyi Comenius Egyetem munkatársai. Az ImagineLogo első magyar nyelvű változata Abonyi-Tóth Andor, Turcsányi-Szabó Márta és Windish József az ELTE TEAM Labor munkatársainak köszönhetően 2005-ben jelent meg.

Minimális rendszerkövetelményként Pentium II. 200 MHz-es processzor, 32MB RAM, 60MB szabad lemezterület, legalább 800x600-as felbontású grafikus kártya, 16 bites színmélységgel, CD meghajtó, egér (vagy más mutatóeszköz), és Windows 98/ME vagy Windows 2000/XP operációs rendszer szükséges.

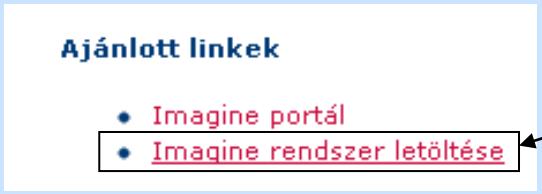
Az IMAGINE rendszerről több információt találhatunk a termék hivatalos weboldalán, az IMAGINE portálon (http://imagine.elte.hu/).

A letöltés a logo.sulinet.hu oldalról lehetséges:



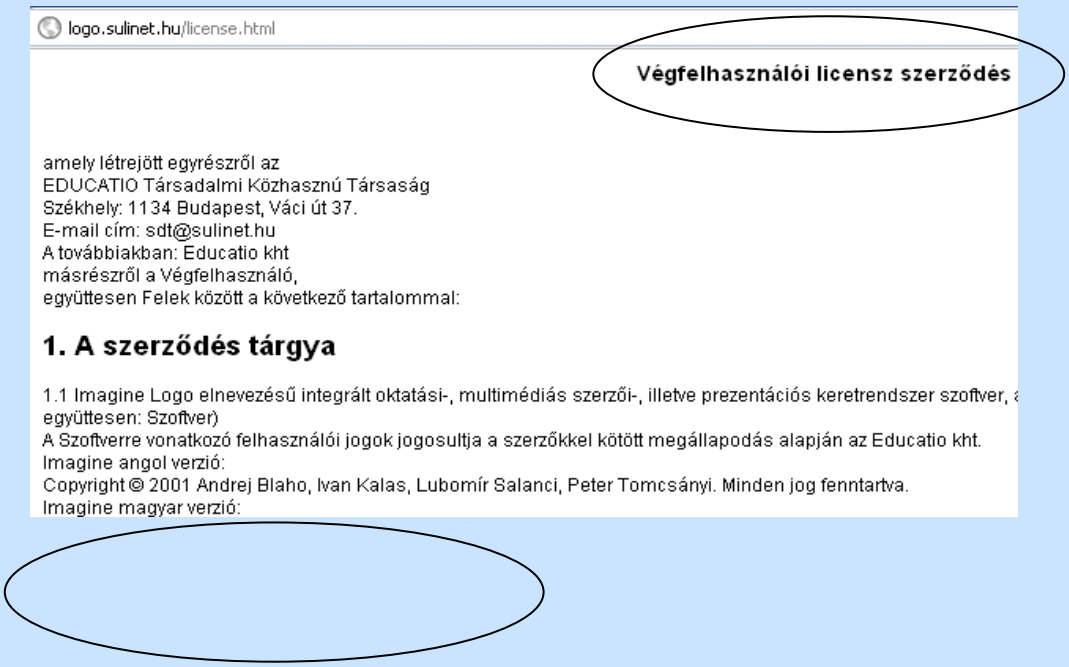
logo.sulinet.hu

A lap alján megtalálható a letöltésre mutató link:



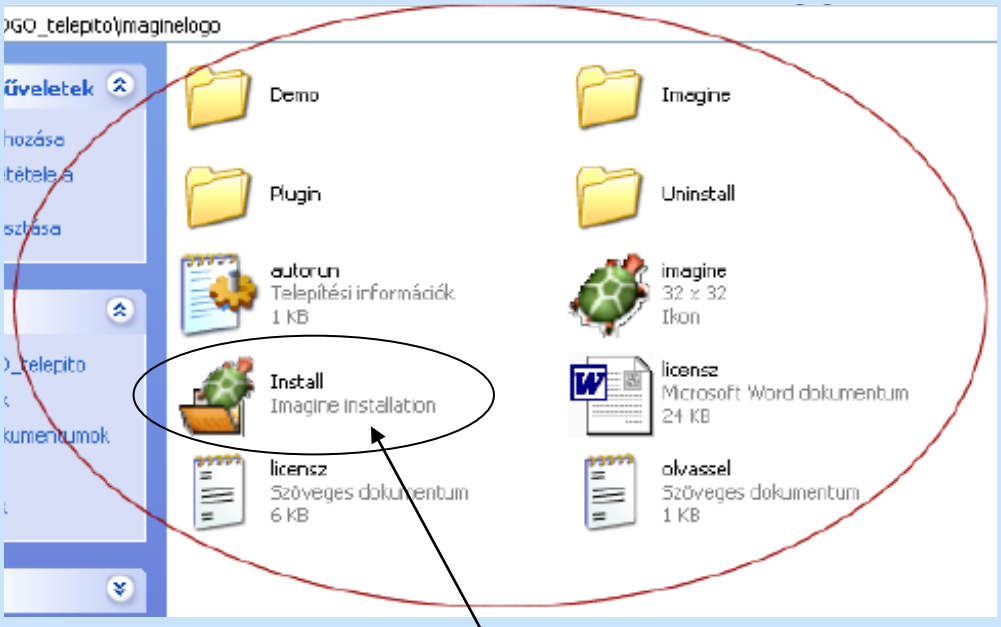
A letöltéshez kattintani!

El kell fogadni a végfelhasználói licensz szerződést:



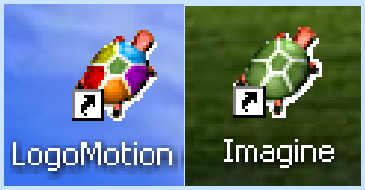
Az elfogadással egy **imaginelogo.zip** fájl letöltése válik lehetővé:

Kicsomagolás után ezeket a fájlokat kell látnunk:



Ide kattintva indul a telepítés!

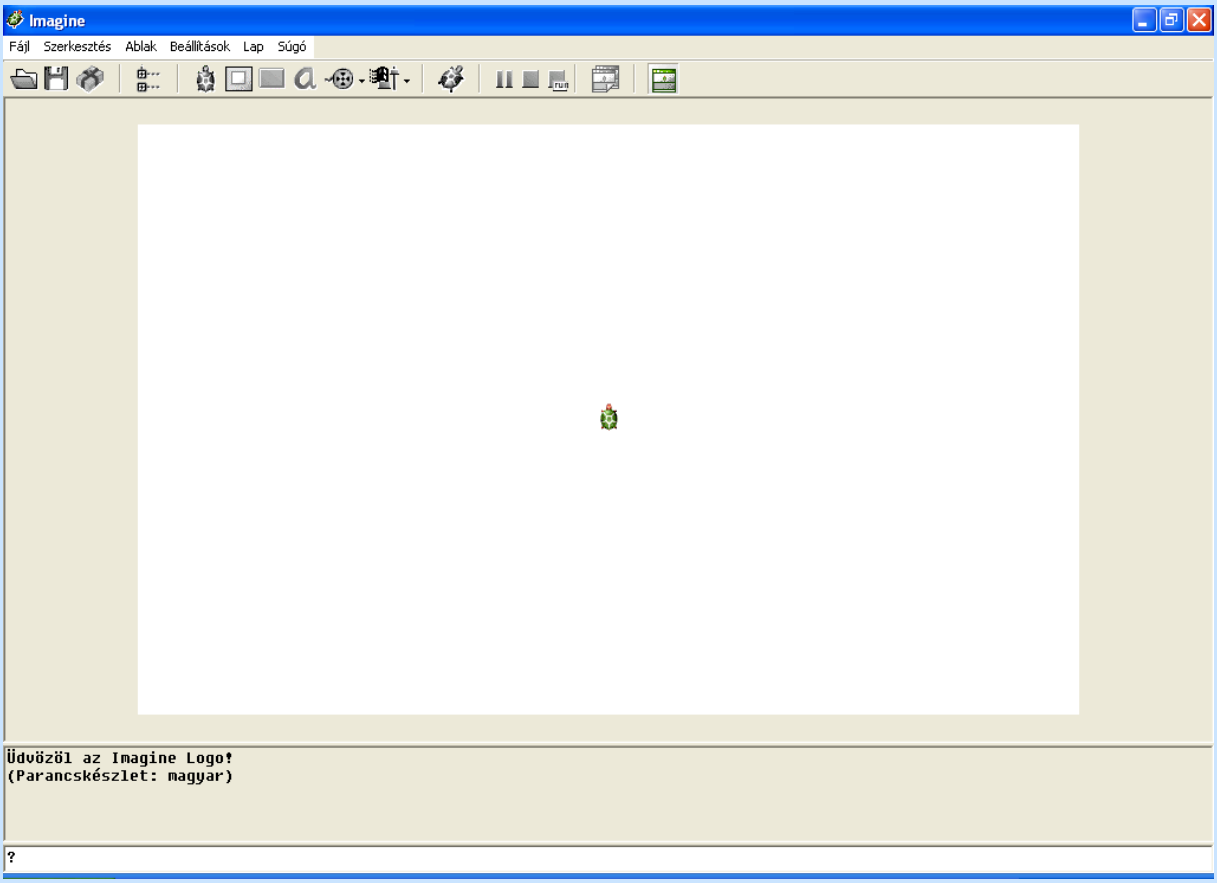
Ha jól dolgoztunk, a következő ikonok lesznek láthatóak az Asztalunkon:



Jó munkát a telepítéshez!

Rajzolási műveletek

A logo indításakor látható képernyő:



Az ImagineFőAblaka egy szokványos Windows alkalmazás, annak összes megszokott lehetőségével: van kerete, kontrollmenü ikonja a címsor baloldali végén, ikonizál és maximalizál (vagy helyreállít) gombja a címsor jobb kéz felőli végében, menüsora parancsokkal, van ikonsora számos olyan gombbal, amelyek felgyorsítják és leegyszerűsítik az alapvető feladatok végrehajtását, s végül van egy munkaterülete, amely vízszintesen ketté van osztva (alapértelmezés szerint) egy szöveges képernyőre (írólapra) és grafikus képernyőre (rajzlapra).

Az ikonsor gombokat tartalmaz projektek megnyitására és mentésére, bemutatóprojektek megnyitására, a Projektböngésző ablak megnyitására és bezárására, új teknőcök, panelek, gombok, szövegdobozok, multimédia, csúszkák, Web böngészők és Web linkek létrehozására, a Festő eszköztár megjelenítésére és elrejtésére, valamint folyamatok vezérlésére. Az ikonsor maradéka lapokkal foglalkozik – tartalmazza az Új lap gombot, és a projekt minden egyes lapjához egy ikont. Ezek közül az egyik ki van emelve: az reprezentálja az aktív lapot.

Az ikonsor az Eszközsor osztály egy példánya. Gombokból és eszközgombokból áll, amelyek a Gomb osztály és az Eszközgomb osztály példányai.

Az ikonsor minden gombja és eszközgombja (kivéve az aktív lapot reprezentáló ikont) szürke. Csak akkor válnak színessé és aktívvá, ha fölémozgatják az egérkurzort.

Mind az ikonsornak, mind pedig a Festő eszköztárnak van egy „igazítás” beállítása, amely alapértelmezésben „felülre”-re van állítva. De ez módosítható:

```
? főeszközsor'igazítás! "balra
```

vagy „jobbra” vagy „alulra”, akkor könnyedén a FőAblak bármely széléhez illeszthetők, Festő eszköztárnál a parancs:

```
? festőeszközsor'igazítás! "alulra
```

A Festő eszköztárt az ikonsor Mutat / Rejt festő eszköztár gombjára való kattintással, vagy az Ablak / Festő eszköztár parancs kiválasztásával lehet megnyitni (és elrejteni). A Festő eszköztár az aktuális háttér letörléséhez, betöltéséhez, elmentéséhez és szerkesztéséhez, négyszögletű terület kiválasztásához, az éppen kiválasztott színnel való kitöltéshez, szabadkézi rajzoláshoz, vonalak, téglalapok és kitöltött téglalapok (ha rákattintanak a Téglalap és az Ellipszis gombokra másodszor, azok Kitöltött téglalapra és Kitöltött ellipszisre változnak) rajzolásához, festékszóráshoz és törléshez tartalmaz gombokat. Ezen eszközök mellett van az a terület, amellyel kiválasztható a vonalvastagság és a vonal színe. Kiválasztható a 18 Imagine szín egyike, s mindhez beállítható a 12 különböző szintű fényerő egyike.

Ha kiválasztják a Aktuális szín opciót a Festő eszköztáron, akkor bekapcsolódik a Szín pipetta eszköz. Ekkor akárhova lehet kattintani a lapon vagy panelen, s ezzel választható ki az aktuális szín. Ha az nem az Imagine színeinek egyike, akkor eltűnik a Világosság beállítása csúszka, s az Aktuális szín megmutatja a fölcsippentett színt, a hozzá tartozó tippben pedig közli a szín RGB kódját is.

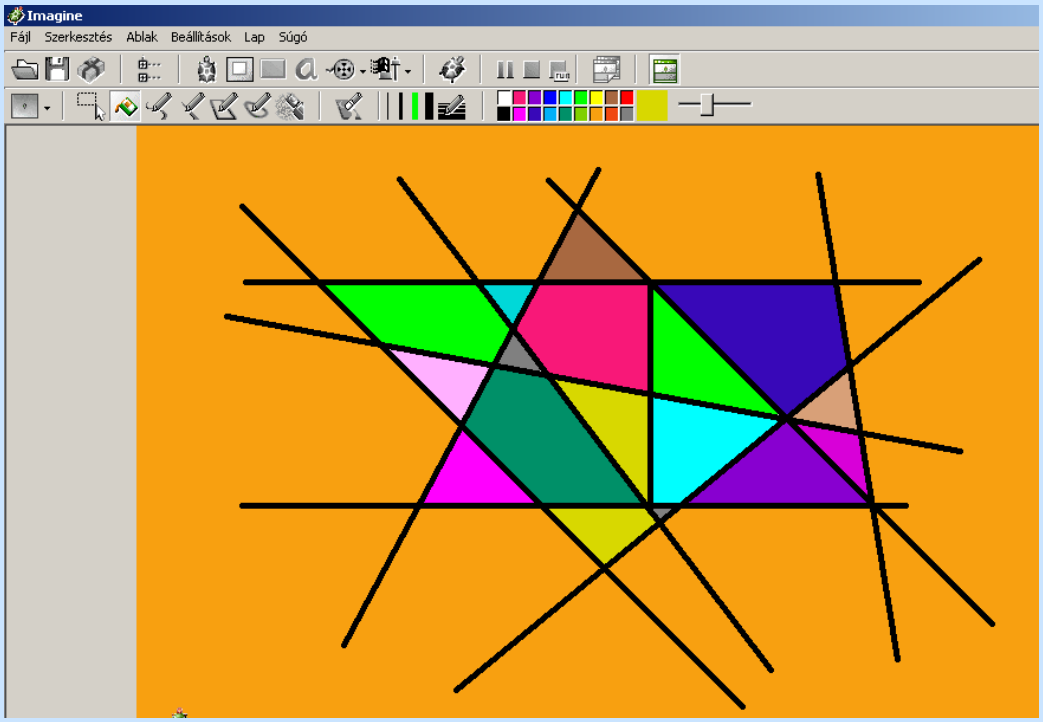
Amennyiben valamit elrontottunk, lehetőségünk van az utoljára végrehajtott rajzolási művelet visszavonására, helyreállítására a SZERKESZTÉS menü RAJZ VISSZAVONÁSA/HELYREÁLLÍTÁSA menüponttal.

Szabadkézi rajzolás:	CERUZA ikon. Egér bal gombjának lenyomásával, s a beállított vonalvastagságnak és színnek megfelelő vonalat rajzol, s az egér mozgatásával készítjük a rajzot.
Kitöltés:	FESTÉS ikon kiválasztásával, bal egérgombbal az alakzat belső pontjára kattintva a zárt vonallal határolt terület a kiválasztott színűre változik.
Egyenes vonal rajzolása:	VONAL ikon kiválasztásával, egér bal gombjának lenyomásával és nyomva tartásával az egér mozgatásával, adott helyen való elengedésével elkészül a vonal. Amennyiben a Shift billentyűt rajzolás közben nyomva tartjuk, akkor csak 45 fokonként elforgatott főirányokban állhat a vonal.
Téglalap rajzolása:	TÉGLALAP ikon. Egér bal gombjának lenyomásával kezdődik, s ha valahol elengedjük, akkor az lesz a másik csúcs (szemközti). Az oldalai vízszintesek és függőlegesek. Shift billentyű folyamatos lenyomásával négyzetet kapunk. A téglalap lehet kitöltetlen és kitöltött (kerete nincs, színe a beállított rajzolószín).
Ellipszis rajzolása:	Téglalaphoz hasonlóan készült kitöltött és kitöltetlen is lehet. ELLIPSZIS ikon kiválasztásával kezdjük a munkát, a Shift folyamatos lenyomásával kört rajzolhatunk.
Radírozás:	RADÍR ikont használjuk, működése a Ceruzához hasonló.
Festékszóró:	FESTÉKSZÓRÓ ikonnal használhatjuk, hatása a valódi festékszóróhoz hasonló, az egér mozgatásának gyorsaságával dönthetjük el a háttérre kerülő festékpöttyök számát.
Vonalvastagság:	Ez határozza meg, hogy az egyes rajzoló eszközök milyen vastag vonalat húznak. VONALVASTAGSÁG ikonnal történik a beállítása, az előre beállított mértéke 1, 2, 5 vagy 10. A

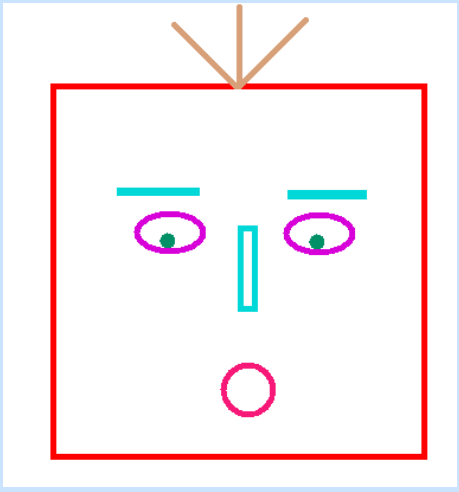
VONALVASTAGSÁGKIVÁLASZTÁSA ikonnal tetszőleges vastagságot is ki lehet választani.

Feladat:

1. Szabjuk „testre” a logo képernyőjét, állítsuk a nekünk kellően kézre eső pozícióba az ikonsort és a Festő eszköztárat!
2. Szabadkézi rajzolás használatával rajzoljunk egy virágot!
3. Egyenes vonal és kitöltés használatával rajzoljon az alábbi ábrához hasonlót!



4. Rajzoljon „Kockafejet” téglalap és ellipszis használatával:



5. Rajzoljon egy házat füstölő kéménnyel!

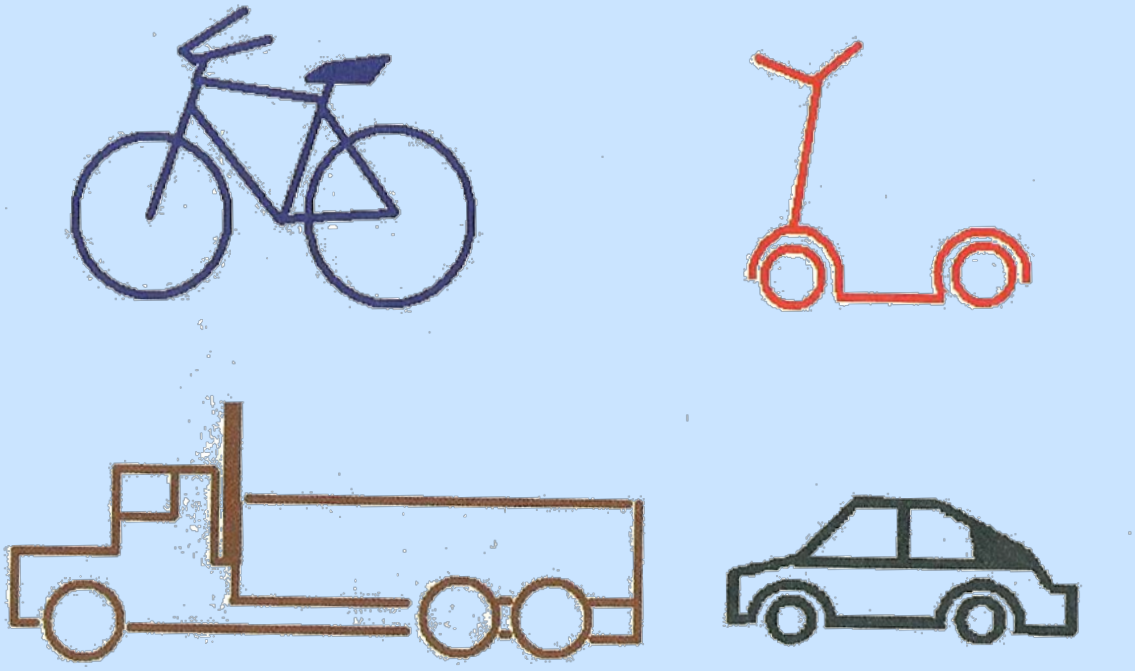
Megoldás:

1. Parancsok:

```
? főeszközsor'igazítás! "balra

? festőeszközsor'igazítás! "alulra
```

(jobbra, balra, alulra, felülre)

Gyakorlás:



Relatív teknőcutasítások

- Egy Imagine projekt tartalmazhat egy vagy több lapot. A lap a projekt összetevőinek alapvető munkaterülete.
- Elhelyezheted rajta a projekted szereplőit (vagyis teknőceit), amelyeket azután mozgathatsz, rajzoltathatsz, írathatsz, beszéltethetsz stb.
- Elhelyezhetsz egy képet a lap háttereként.
- Külön munkaterületeket – vagyis paneleket – is helyezhetsz el a lapon (Ezek lehetnek mozgathatók, egymást átfedők, átméretezhetők, és mindegyiknek lehet saját teknőce vagy más objektuma.)
- Irányítószerveket is elhelyezhetsz rajta. Ilyenek a gombok (akciókkal, hangokkal, zenével, stb.), csúszkák, eszközgombok, eszköztárak.
- Elhelyezhetsz egyéb vizuális objektumokat is, mint például szövegdobozokat, médialejátszókat, Web böngészőket, hálózati objektumokat stb.

Amikor megnyitjuk az Imagine környezetet, az kezdetben csak egy lapot tartalmaz, melynek Lap1 a neve. Ehhez egy teknőc tartozik. Természetesen létre lehet hozni további lapokat is. Ehhez a következőket teheted:

- Rákattintasz a Új lap gombra az Ikonsoron
- Az „újlap" parancsot használjuk bemenet nélkül vagy választható bemenetként az új lap nevével: újlap "név
- Az új objektumok létrehozására szolgáló standard parancsot használjuk: új "Lap […beállítások …] vagy újobjektum beállítás1 érték1 beállítás2 érték2

Ha egy projekt több lapot tartalmaz és szeretnéd megtudni az éppen látható lap nevét, egyszerűen az aktív laphoz tartozó ikon fölé (az ikonsoron) viheted az egérmutatót és leolvashatod a gombhoz tartozó leírásról.

Egy lap törléséhez a következőket teheted:

Rákattinthatsz a jobb egérgombbal a lapra, és a környezeti menüjéből kiválaszthatod a Lap törlése parancsot.

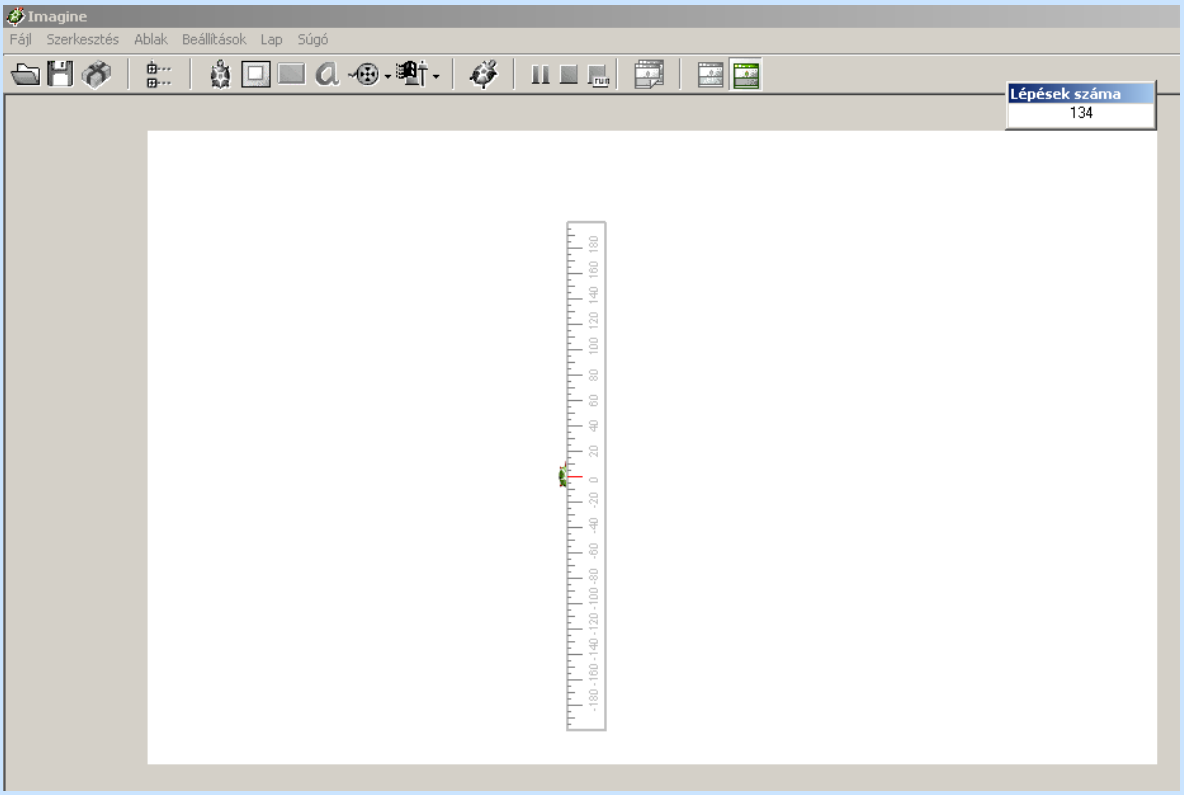
Használhatod az „töröllap” parancsot egy paraméterrel: a törlendő lap nevével. Ha csak egy lap van, az nem törölhető. (pl.: töröllap ”lap2)

Alapvető mozgások:

előre / e szám

Az éppen aktív teknőcöket elmozdítja a számmal megadott lépéssel (azaz pixellel) abba az irányba, amelybe néz. A teknőc tollának üzemmódjától függően vagy nem rajzol vonalat, vagy vonalat rajzol a beállított színnel, vastagsággal és mintával, vagy töröl egy vonalat, vagy invertál egy vonalat. Minden teknőc csak a saját helyén mozog és rajzol vonalat, és az aktuális tartományán belül. Megjegyzés: az előre 0 azt jelenti, hogy 0 lépés előre, azaz egy 1 egység méretű vonalat (pontot) rajzol. Az előre 1 már 1 lépést jelent előre, így egy 2 hosszúságú vonalat húz.

Ha az „e „vagy „h” utasítás után leütjük az F9 gombot, akkor igénybe vehetjük a vonalzó segítségét (vonalzón kattintunk, vagy a kis ablakba beírhatjuk a lépés nagyságát):



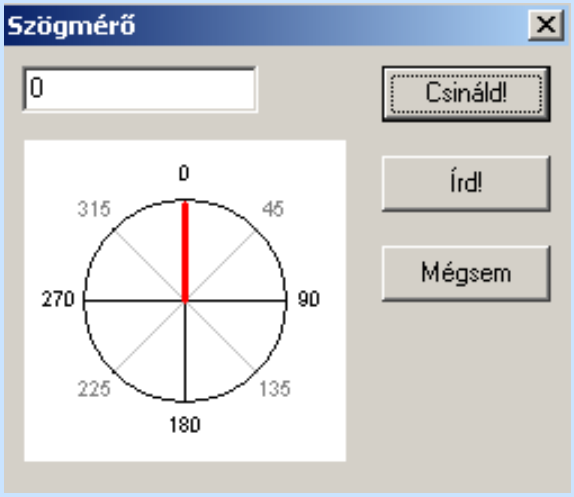
hátra / h szám — Minden épp aktív teknőcöt szám lépéssel (vagyis pixellel) elmozdít az ellenkező irányba, mint amerre néznek. Ugyanúgy, mint az előre parancsnál, a vonal végső alakja a toll üzemmódjától, stílusától és a tartomány stílusától függ.

Alapvető elfordulások:

balra / b szög — Hatására minden aktív teknőc balra fordul (vagyis az óramutató járásával ellenkező irányban) az adott szöggel (fokokban mérve). Az első aktív teknőc aktuális irányát az „irány” művelettel lehet lekérdezni. Ez a „jobbra” és a „irány!” használatával is megváltoztatható.
Megjegyzés: a balra szög ugyanaz, mint a jobbra -szög

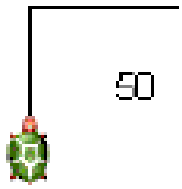
jobbra / j szög — Hatására minden aktív teknőc jobbra fordul (vagyis az óramutató járásával megegyező irányban) az adott szöggel (fokokban mérve).

Ha a „j” vagy „b” utasítás után leütjük az F9 gombot, akkor igénybe vehetjük a szögmérő segítségét (szögmérőn kattintunk, vagy beírhatjuk a szög nagyságát):

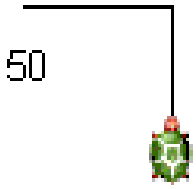


Feladat:

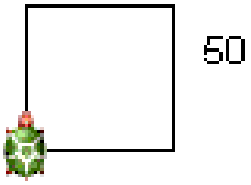
1.



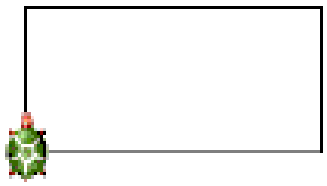
2.

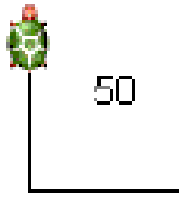


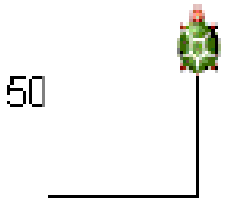
3

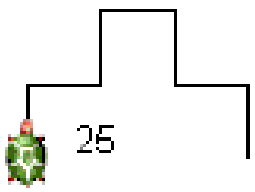


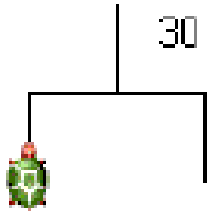
4.











5.

7.

Megoldások:

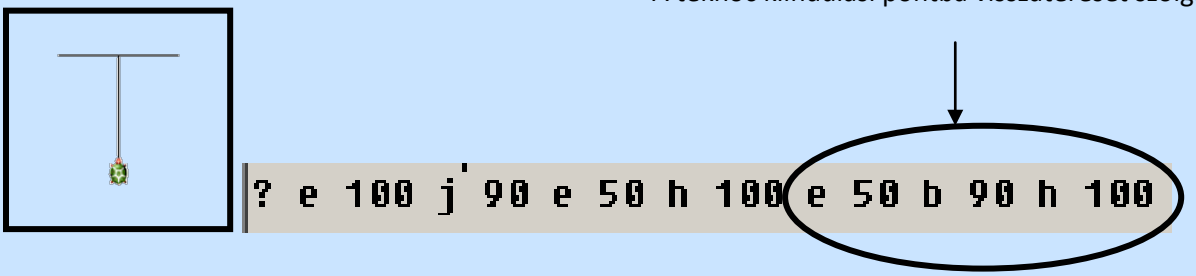
1. e 50
   j 90
   e 50
2. e 50
   b 90
   e 50
3. e 50
   j 90
   e 50
   j 90
   e 50
   j 90
   e 50
   j 90
4. e 50
   j 90
   e 100
   j 90
   e 50
   j 90
   e 100
   j 90
5. h 50
   j 90
   e 50
6. h 50
   b 90
   e 50
7. e 25
   j 90
   e 25
   b 90
   e 25
   j 90
   e 25
   j 90
   e 25
   b 90
   e 25
   j 90
   e 25
8. e 30
   j 90
   e 30
   b 90

```
e 30
h 30
j 90
e 30
j 90
e 30
```

A nagy teknőctétel

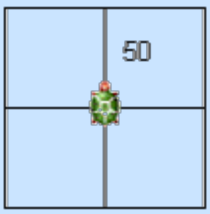
A nagy teknőctétel: bármilyen ábrát rajzolunk, annak elkészítése után a teknőc kezdő és végállapot jellemzői egyezzenek meg.

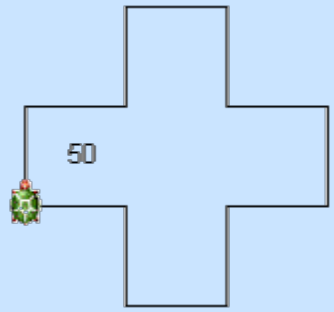
A teknőc kiindulási pontba visszatérését szolgálja!

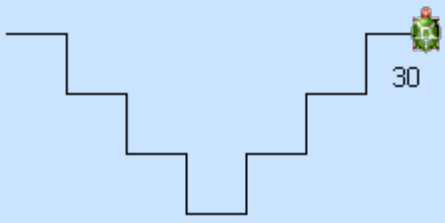


Feladat: Tartsuk be a nagy teknőctételt!

1. 

2. 

3. 

4. 

Megoldás:

1. e 50
   j 90
   e 50
   j 90
   e 50
   b 90
   e 50
   b 90
   e 50
   h 50
   j 90
   h 50
   j 90
   h 50
   b 90
   h 50
   b 90
   h 50

Vegyük észre, hogy a visszafelé vezető út az eredetihez hasonló, csak az „e” utasítás helyett a „h”-t a „jobbra” fordulás helyett pedig a „balra” parancsot használjuk és viszont.

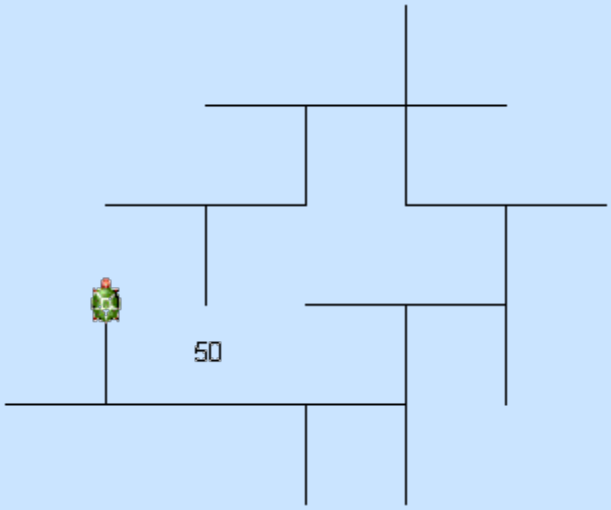
2. e 50 j 90 e 50 j 90 e 50 j 90 e 50 j 90
   j 90
   e 50 j 90 e 50 j 90 e 50 j 90 e 50 j 90
   j 90
   e 50 j 90 e 50 j 90 e 50 j 90 e 50 j 90
   j 90
   e 50 j 90 e 50 j 90 e 50 j 90 e 50 j 90
   j 90

Vegyük észre, hogy az uta
vannak ismétlődő részek,
későbbiekben még visszat

3. e 50 j 90 e 50 b 90 e 50 j 90
   e 50 j 90 e 50 b 90 e 50 j 90
   e 50 j 90 e 50 b 90 e 50 j 90
   e 50 j 90 e 50 b 90 e 50 j 90

4. b 90 e 30 b 90 e 30 j 90 e 30 b 90 e 30 j 90 e 30 b 90 e
   30 j 90 e 30
   j 90
   e 30 b 90 e 30 j 90 e 30 b 90 e 30 j 90 e 30 b 90 e 30 j
   90
   b 90 h 30 j 90 h 30 b 90 h 30 j 90 h 30 b 90 h 30 j 90 h
   30
   b 90
   h 30 b 90 h 30 j 90 h 30 b 90 h 30 j 90 h 30 b 90 h 30 j
   90 h 30 j 90

Gyakorlás



A toll beállításai

A teknőc jellemzői közül fontos csoportot alkotnak a rajzolással kapcsolatos beállítások.

A toll állapota:

Ezt az ábrát úgy érhetjük el, hogy a teknőc tollát felemeljük, ebben az esetben haladtában nem rajzol; a parancs formája a **tollatfel/tf**, ezzel ellentétes folyamat, mikor felemelt toll után újra rajzolni szeretnénk, akkor a tollat le kell tenni **tollatle/tl**. A parancsok ugyan nyelvtanilag helytelenek, de az Imagine-ben (hasonlóan a többi programozási nyelvhez) a parancsok csak egy szóból állnak.

A toll pillanatnyi állapotáról a **toll** függvény ad tájékoztatást:

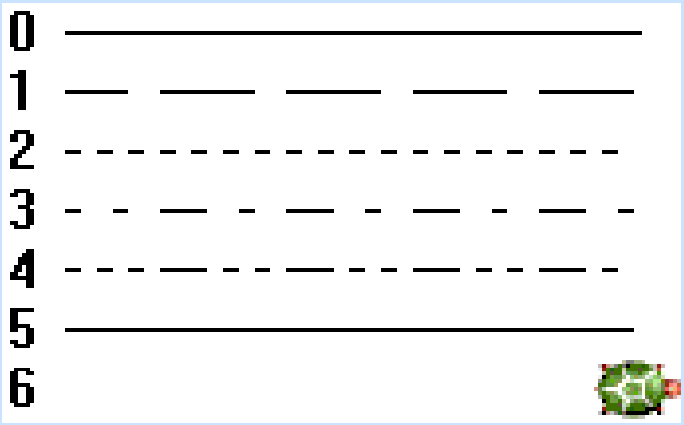
```
? ki toll
tollatle
```

tollradír / trd Hatására minden aktív teknőc úgy változtatja meg tollának állapotát, hogy ha mozog a helyén, akkor töröl (azaz a háttérszínre állít) minden olyan pontot (az aktuális tollvastagságot használva), amely fölött elhalad.

tollváltó / tvt Hatására minden aktív teknőc úgy változtatja meg tollának állapotát, hogy ha mozog a rajzterületén, akkor invertál (azaz megcseréli a tollszínt és a háttérszínt) minden olyan pontot (az aktuális tollvastagságot használva) amely fölött elhalad. Ezzel a toll-állapottal sok érdekes hatást lehet elérni, amelyek a tollszín és a háttérszín aktuális kombinációjától függnek.

Tollminta:

Toll- és háttérvonalhoz hét különböző minta áll rendelkezésre: 0-tól 6-ig számozva. Az alapértelmezett minta a 0. Beállítani vagy visszaállítani a tollminta! segítségével lehet. Megjegyzés: a 0-tól különböző minták használatának csak akkor van értelme, ha a teknőc tollvastagsága legalább 1. Megjegyzés: a 6-os minta egy üres vonalat ábrázol.



A használt tollminta értékét kaphatjuk meg a:

```
? mutat tollminta
0
```

vagy a

```
? ki tollminta
0
```

utasításokkal. Ha mi kívánjuk beállítani a tollmintát a **tm!***szám* utasítással tehetjük meg:

Tollvastagság

Bármely pozitív szám (minden 0-nál kisebb megadott szám automatikusan 1-re nő). Megjegyzés: az 5-nél vastagabb tollal rajzolt vonalak vége lekerekített.

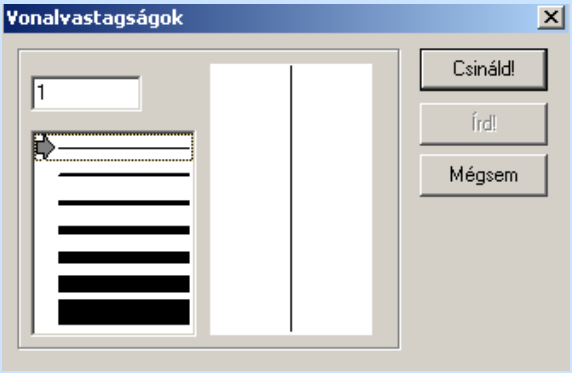
A **tollvastagság** röviden **tv** függvény értékeként kapjuk meg a toll vastagságát:

```
? ki            ? mutat
tv      vagy    tollvastagság
1               1
```

Parancskiadás: **tollvastagság!** *szám*

A parancskiadást könnyíti:

A vonalvastagságok kiválasztó ablak, melyet a tv! parancs begépelése és az F9 lenyomásával hívhatunk elő.



Tollszín

? mutat tollszín azzal a színnel tér vissza, amely az első aktív teknőc tollának aktuális színe. Amikor a teknőcöt létrehozzák, alapértelmezett tollszíne fekete.

A toll színének beállítása a tollszín! / tsz! paranccsal történik: **tsz!***szín*

Az alábbi lehet:

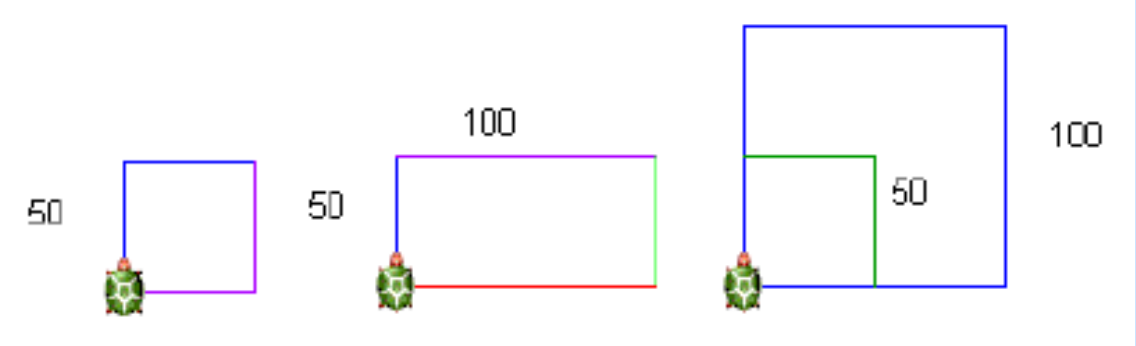
- Egy egész szám 0-tól 15-ig, amely a 16 színű alap-paletta egyszerű színeinek megjelölése:

- Egy Imagine szín, például piros, piros11, sötétkék2 stb.:

- Egy RGB szín, például [128 128 0].
- A tetsz szó, amely egy véletlen színt jelöl.

Feladat:

1. Készítse el a következő ábrákat:

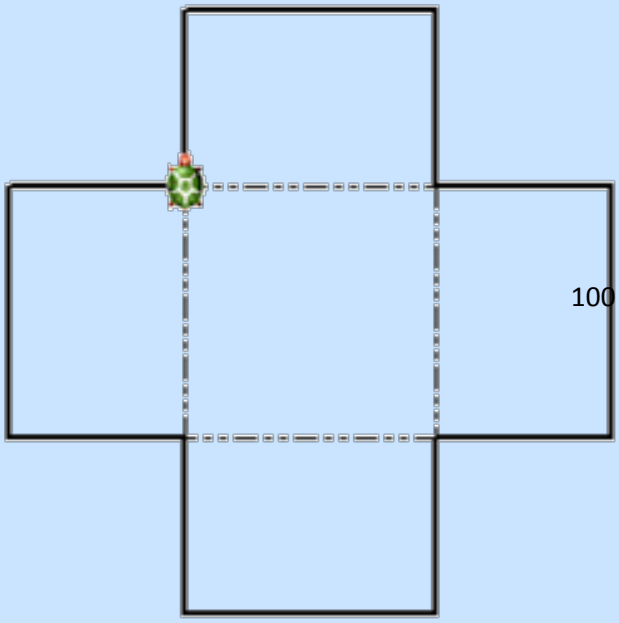


Megoldás:

`? tsz! "kék e 50 j 90 e 50 j 90`
`? tsz! "lila e 50 j 90 e 50 j 90`
	`? tsz! "kék e 50 j 90`
	`? tsz! "lila e 100 j 90`
	`? tsz! "zöld e 50 j 90`
	`? tsz! "piros e 100 j 90`
		`? tsz! "zöld e 50 j 90 e 50 j 90 e 50 j 90 e 50 j 90`
		`? tsz! "kék e 100 j 90 e 100 j 90 e 100 j 90`

		Ha nem a zöld négyzettel kezdjük:
		`? tsz! "kék e 100 j 90 e 100 j 90 e 100 j 90`
		`? tfe 50 j 90 tsz! "zöld tle 50 j 90 e 50 j 90 tf e 50 j 90`

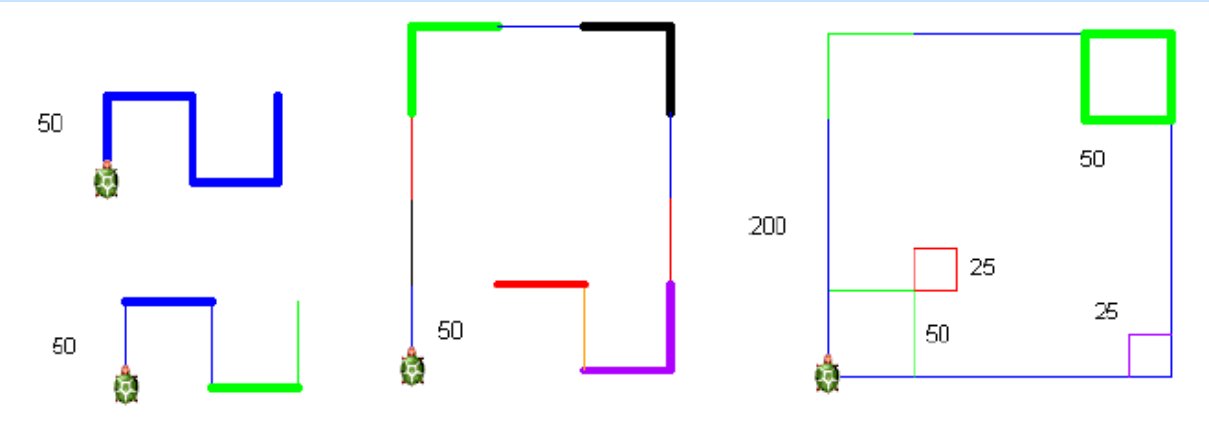
2. Készítse el a következő ábrát!

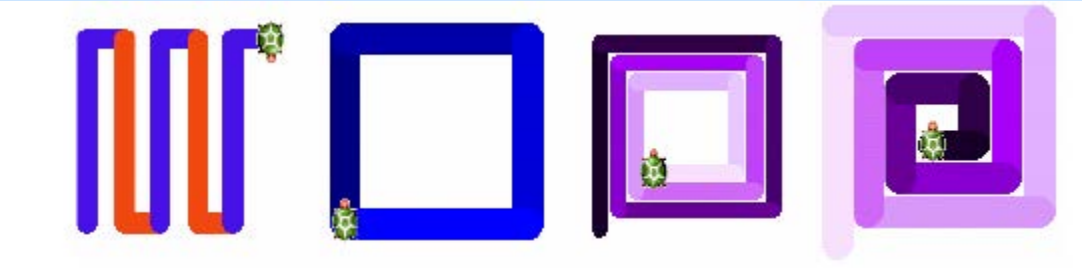


Megoldás:

```
? tv! 2 tsz! "fekete tm! 0
? e 70 j 90 e 100 j 90 e 70 b 90
? e 70 j 90 e 100 j 90 e 70 b 90
? e 70 j 90 e 100 j 90 e 70 b 90
? e 70 j 90 e 100 j 90 e 70 b 90
? tv! 1 tm! 4
? j 90 e 100
? j 90 e 100
? j 90 e 100
? j 90 e 100
```

Gyakorló feladatok:





Ismétlések

Az eddigiekben is láthattuk, hogy az utasításokban vannak olyan részek, melyek többször előfordulnak egymás után; ennek megvalósítására szolgál az
ismétlés*ismétlésszám* [*utasításlista*] parancs.

A relatív teknőcutasításoknál találkoztunk az alábbi feladatokkal:

A négyzet megvalósítására szolgáló utasítás a következő volt:

```
e 50
j 90
e 50
j 90
e 50
j 90
e 50
j 90
```

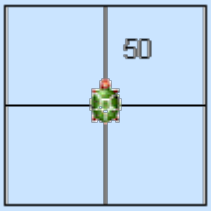
Észrevehetjük, hogy az e 50 j 90 utasítás 4-szer ismétlődik.

Ismétlés használatával egyszerűen megvalósítható: **`ism 4 [e 50 j 90].`**

A téglalap elkészítése:

```
e 50
j 90
e 100
j 90
e 50
j 90
e 100
j 90
```

Ismétlés használatával: **`ism 2 [e 50 j 90 e 100 j 90].`**

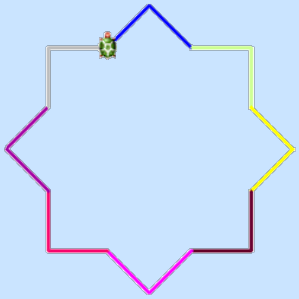



```
ism 4 [e 50 j 90 e 50 j 90 e 50 j 90 e 50 j 90 j 90]
```

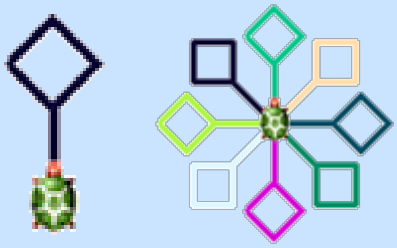

```
ism 4 [e 50 j 90 e 50 b 90 e 50 j 90]
```

Feladat:

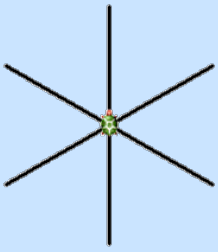
1. Készítsük el az alábbi ábrát ismétlés használatával:



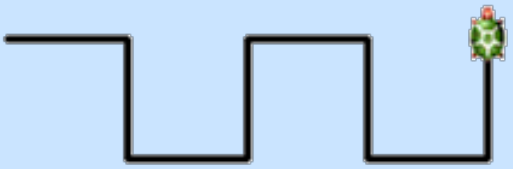
2. Az első ábra felhasználásával készítsd el a másodikat is!



3. Fedezzük fel az ismétlődő részt!



4. Vajon miből áll?

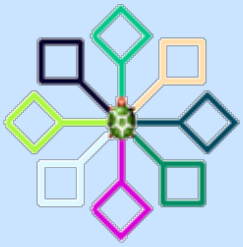


5. Ismétléses?

Megoldás:

1. `? j 45`
   `?ism 8 [tsz! tetsze 50 j 90 e 50 b 45]`
   `? b 45`
2. Az alap

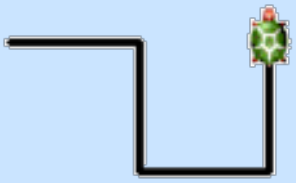
`e 30 b 45 ism 4 [e 20 j 90] j 45 h 30`



`ism 8 [tsz! tetsz e 30 b 45 ism 4 [e 20 j 90] j 45 h 30 j 45]`

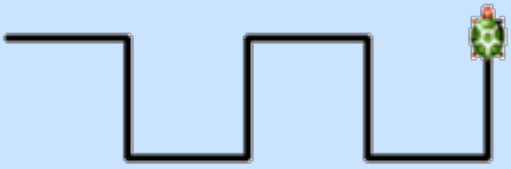
3. `ism 6 [e 100 h 100 j 60]`

4. Az alap:

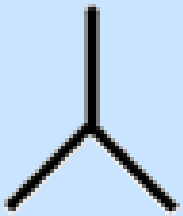


5. `ism 2 [j 90 e 50] ism 2 [b 90 e 50]`

A teljes ábra utasítása tehát:

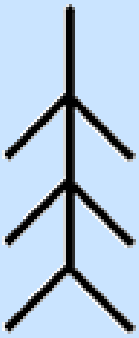


6. `ism 2 [ism 2 [j 90 e 50] ism 2 [b 90 e 50]]`
7. Az alapelem:



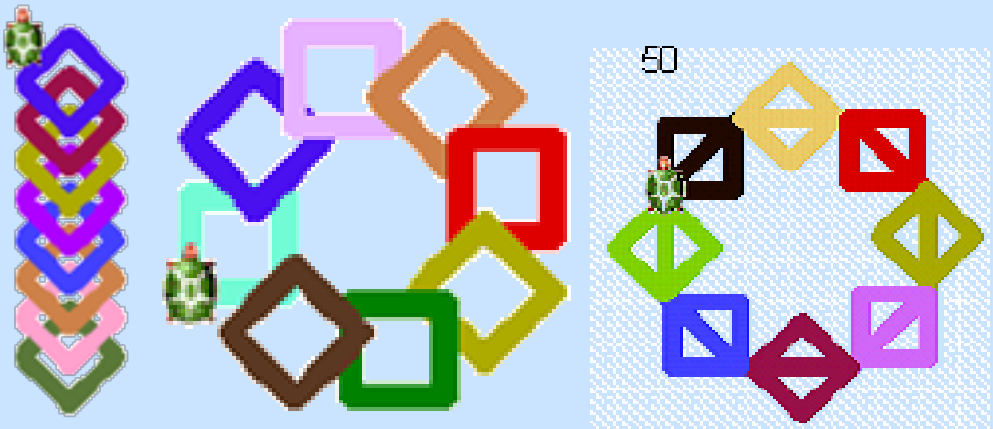
Utasítás: `j 45 h 30 e 30 j 90 e 30 h 30 b 135 e 30`

A teljes ábra:



`ism 3 [j 45 h 30 e 30 j 90 e 30 h 30 b 135 e 30]`

8. Gyakorlás



Eljárások - Sokszögek

A szabályos sokszög olyan sokszög, amelynek minden oldala és minden belső szöge egyenlő.

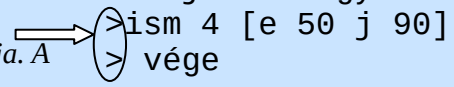
Az előzőekben már megtanultunk négyzetet készíteni; először a parancsok begépelésével, majd ismétléssel.


```
e 50
j 90
e 50
j 90
e 50
j 90
e 50
j 90
```

```
ism 4 [e 50 j 90]
```

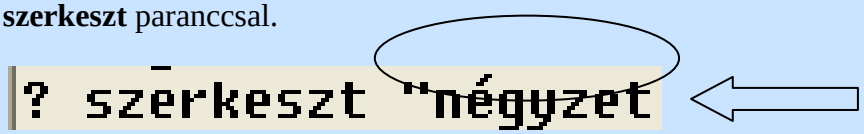
Jó lenne, ha a négyzet szó begépelésével az alakzat elkészülne. Ahhoz, hogy ez sikerüljön meg kell tanítanunk erre a programot. Erre szolgál az **eljárás** parancs.

A megváltozott sormutató jelzi, hogy a program a beírt parancsokat tárolja. A vége paranccsal zárjuk.



```
? eljárás négyzet
ism 4 [e 50 j 90]
vége
```

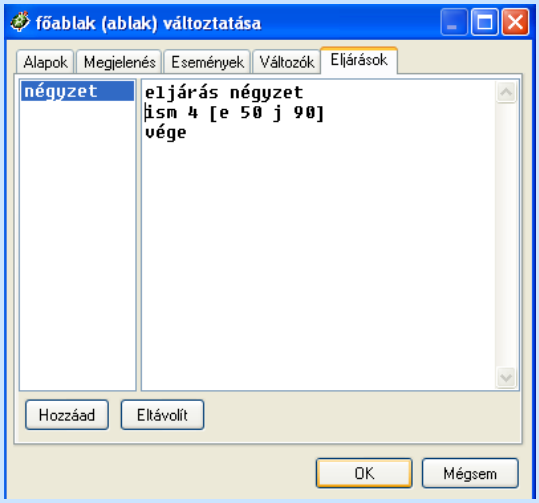
Ezzel a beírási móddal a korábban beírt sorok nem javíthatóak, ezért a megadás egyszerűbb a **szerkeszt** paranccsal.



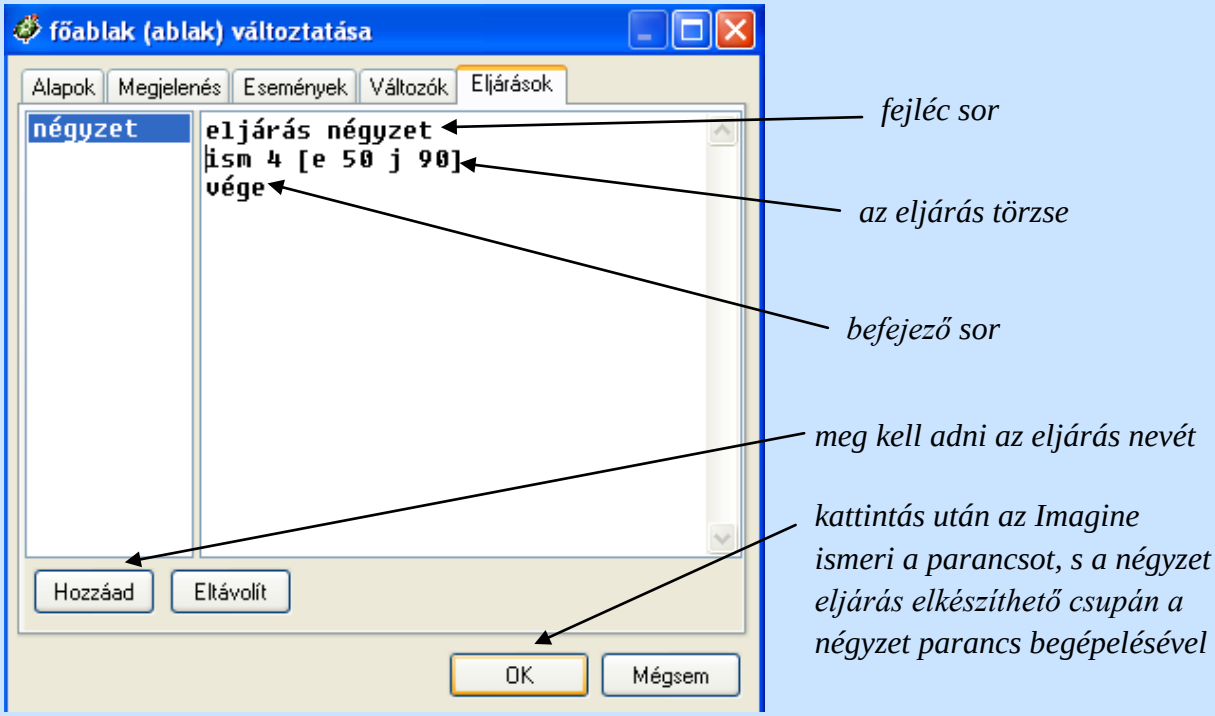
```
? szerkeszt "négyzet
```

Az eljárás neve

Ennek hatására megnyíló ablakban szerkeszthetjük az eljárásunkat.



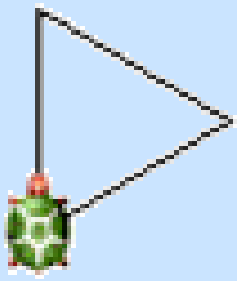
Ez az ablak jelenik meg akkor is, ha a Fő eszköztár Intéző ikonjára kattintunk, az Intézőből a Főablakot dupla kattintással megnyitjuk és az Eljárások fület választjuk.



Programjainkat összetartozó, általában egy részfeladatot megoldó kisebb részekből állítjuk össze, úgynevezett eljárásokba szervezzük.

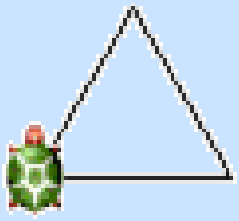
Készítsük el a szabályos háromszöget!

```
? eljárás háromszög
>ism 3 [e 50 j 120]
> vége
```

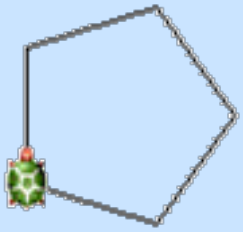


Amennyiben azt szeretnénk, hogy a háromszögnek legyen vízszintes oldala, akkor az eljárás a következőképpen alakul:

```
? eljárás háromszög
> j 30
>ism 3 [e 50 j 120]
> b 30
> vége
```

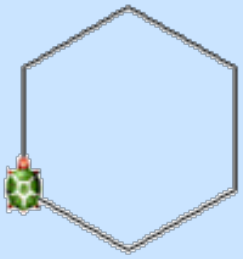


Szabályos ötszög:



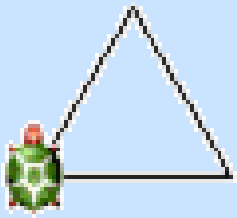
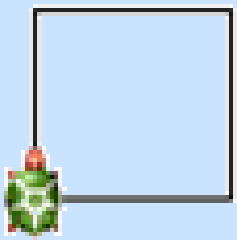
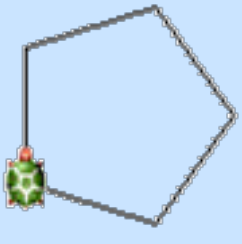
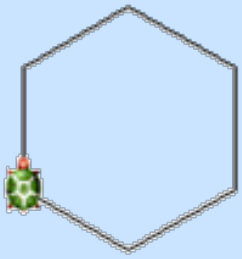
```
? eljárás ötszög
>ism 5 [e 50 j 72]
> vége
```

Szabályos hatszög:



```
? eljárás hatszög
>ism 6 [e 50 j 60]
> vége
```

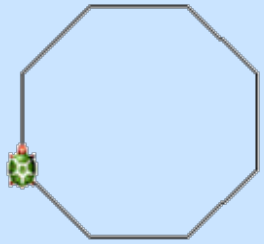
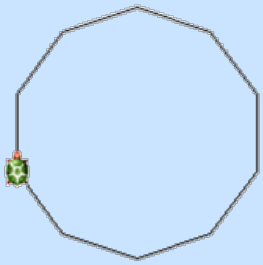
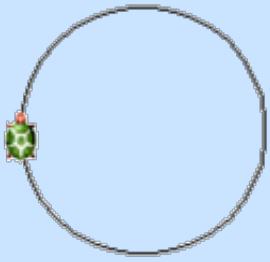
Próbáljunk meg valamilyen összefüggést felfedezni?

			
eljárás háromszög j 30 ism 3 [e 50 j 120] b 30 vége	eljárás négyzet ism 4 [e 50 j 90] vége	eljárás ötszög ism 5 [e 50 j 72] vége	eljárás hatszög ism 6 [e 50 j 60] vége
3*120=360	4*90=360	5*72=360	6*60=360

Feladat:

1. Készítsük el a szabályos nyolcszöget!
2. Készítsük el a szabályos tízszöget!
3. Készítsük el a szabályos harminchatszöget 10-es oldalhosszúsággal!

Megoldás:

`eljárás nyolcszög`
`ism 8 [e 50 j 45]`
`vége`
	`eljárás tízszög`
	`ism 10 [e 50 j 36]`
	`vége`
		`eljárás harminchatszög`
		`ism 36 [e 10 j 10]`
		`vége`

Alakzatok kitöltése

Az eddigiekben alakzatainkat a tollszín és tollvastagság megfelelő beállításaival készítettük el. Többször azonban szükség lehet arra, hogy a zárt alakzatokat valamilyen színnel valamilyen mintázattal kitöltsük. Erre szolgáló parancs a **tölt**.

Ahhoz azonban, hogy a kitöltésre vonatkozó parancsot használni tudjuk, be kell állítanunk a használandó **töltőszín**-t (tsz) valamint a **töltőminta**-t (tlm).

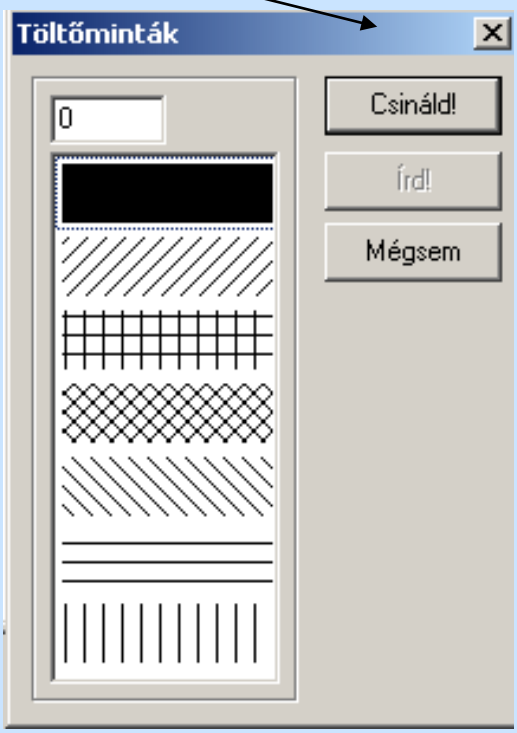
pontszín	A művelet a lap vagy panel azon pontjának színével tér vissza, amelyen az első aktív teknőc épp elhelyezkedik.
tölt	Parancs hatására minden aktív teknőc kitölti az őt körülvevő tartományt a kitöltőszínével vagy kitöltőmintájával, az aktuális kitöltési szabályt alkalmazva.
töltőszín / tlsz	A művelet azzal a kitöltőszínnel tér vissza, amely az első aktív teknőcnek épp be van állítva. Ha nem állítottak be semmilyen kitöltőszínt, akkor az alapértelmezett érték ugyanaz, mint a teknőc tollszíne.
töltőminta / tlm	A művelet azzal a kitöltőmintával tér vissza, amely az első aktív teknőcnek épp be van állítva. Ha nem állítottak be semmilyen kitöltőmintát, akkor az alapértelmezett érték 0, ami egyszerű (szolid) kitöltést jelent.

töltőszín! / tlsz!

Mint parancs beállítja minden aktív teknőc kitöltőszínét. Ezt a színt használja a tölt és a sokszög parancs is. (Az utóbbira a későbbiekben még visszatérünk). Ha nem állítottak be semmilyen színt, a teknőc az aktuális tollszínét használja. Ez a parancs lehetővé teszi, hogy különböző legyen a körvonal és a kitöltés színe.

töltőminta! / tlm!

Mint parancs beállítja minden aktív teknőc kitöltőmintáját. Ha nem állítanak be kitöltőmintát, a teknőc a 0 mintát használja. Ehhez az eljáráshoz tartozik egy kiválasztó is.

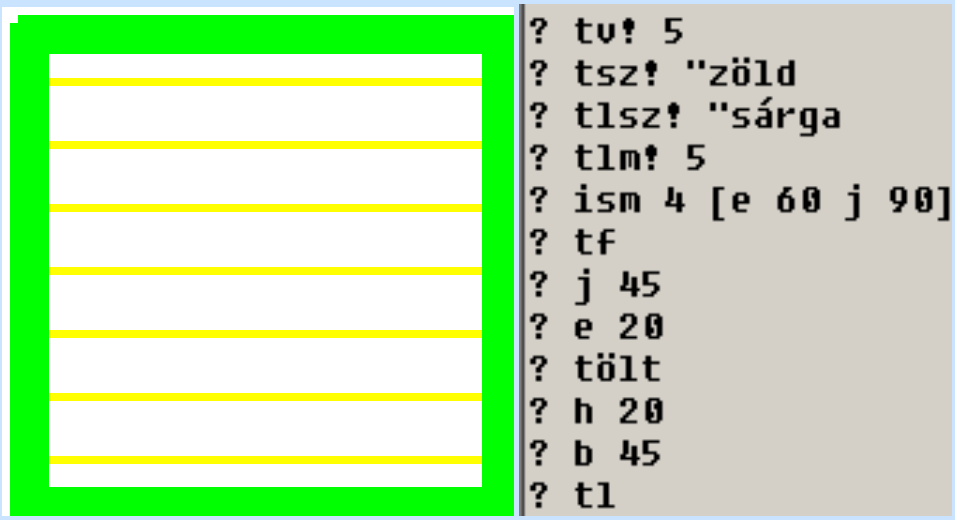



```
? ki pontszín
fehér
? ki töltőszín
fekete
? ki töltőminta
0
? (ki töltőszín töltőminta)
fekete 0
```

Alapértelmezés

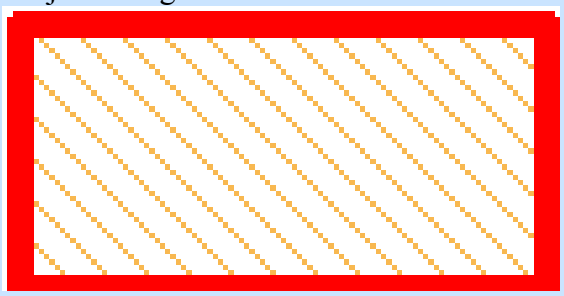
Ahhoz, hogy egy zárt alakzatot kitöltsünk, az alakzaton belül kell tartózkodnunk. Ezt úgy tudjuk megtenni, hogy a tollat felemeljük és „belemegyünk” (a zárt alakzat belső pontja fölé mozgatjuk a teknőst) az alakzatba, ott elvégezzük a kitöltést (ez felemelt tollal is megvalósítható), s a nagy teknőctétel értelmében visszatérünk a kezdő pozícióba, s itt letesszük a tollunkat (különben az ezt követő parancsoknál nem tudunk semmit rajzolni).

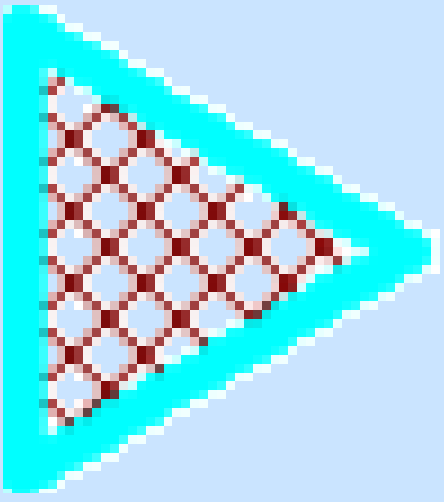
Készítsünk egy 5-ös tollvastagsággal zöld színű szegéllyel ellátott sárga vízszintesen csíkozott 60-as oldalhosszúságú négyzetet!



```
? tv! 5
? tsz! "zöld
? tlsz! "sárga
? tlm! 5
? ism 4 [e 60 j 90]
? tf
? j 45
? e 20
? tölt
? h 20
? b 45
? tl
```

Feladat: Rajzold meg a következő ábrákat!

1. 

2. 

Megoldás

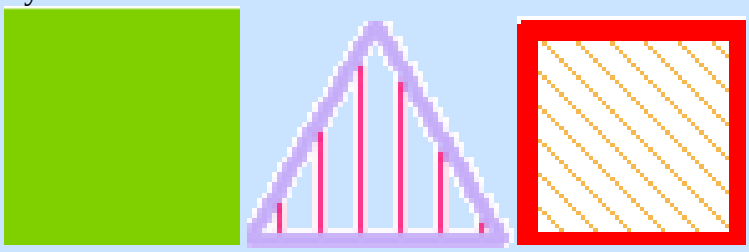
1.
```
tsz! "piros
tv! 5
tlm! 4
tlsz! "narancs7
```

2.
```
tv! 5
tsz! 11
tlm! 3
tlsz! 4
```

```
téglalap
tf
j 45
e 10
tölt
h 10
b 45
tl
```

```
ism 3 [e 50 j 120]
tf
j 45
e 10
tölt
h 10
b 45
tl
```

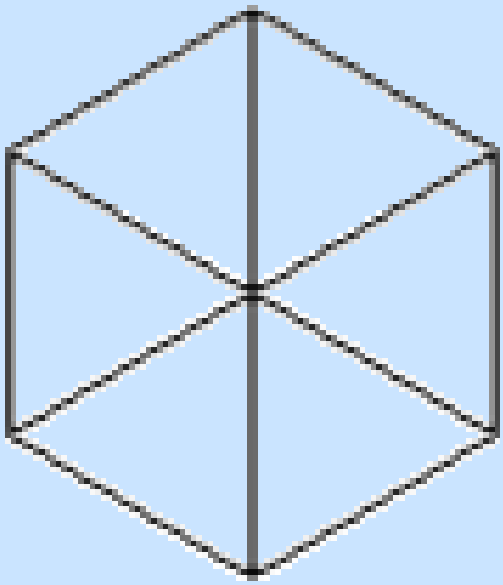
Gyakorlás



Építkezés alakzatokból

Az eddigiekben megtanultunk különböző alakzatokat megrajzolni; szabályos sokszögeket elkészíteni pedig már eljárásokkal is tudunk. Használjuk fel az eddig tanultakat és építkezzünk ezekből az alakzatokból.

Készítsünk egy szabályos hatszöget a szabályos háromszög felhasználásával.

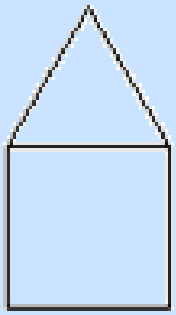


```
ism 3 [e 50 j 120] j 60
ism 3 [e 50 j 120] j 60
ism 3 [e 50 j 120] j 60
ism 3 [e 50 j 120] j 60
ism 3 [e 50 j 120] j 60
ism 3 [e 50 j 120] j 60

eljárás hatszög_3
ism 6 [háromszög j 60]
vége
```

Természetesen ehhez léteznie kell a háromszög eljárásnak:

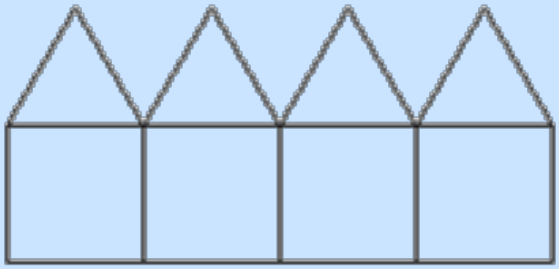
```
eljárás háromszög
ism 3 [e 50 j 120]
vége
```

Készítsünk egy ház alakzatot négyzet és a szabályos háromszög felhasználásával!



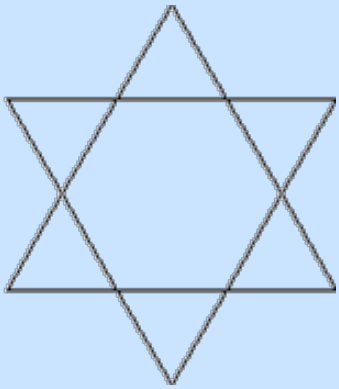
```
eljárás ház
négyzet e 50 j 30 háromszög
b 30 h 50
vége
```

Az előbb elkészített ház felhasználásával készítsünk el egy 4 házból álló házsort!

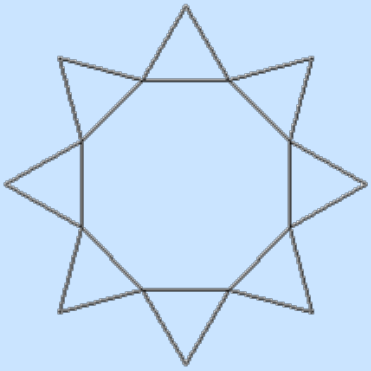


```
eljárás házsor
ism 4 [ház tf j 90 e 50 b 90
tl] b 90 e 200 j 90
vége
```

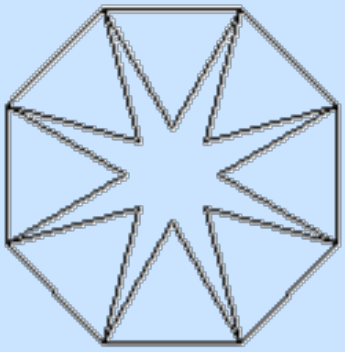
Feladatok:



1. hatszöges_3



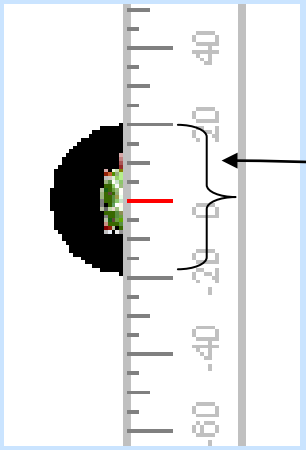
2. nyolcszöges_3

3.  nyolcas_belül

Színes pontok a Logo képekben

Az eddigi rajzolóeljárások során az alakzatok a teknőc mozgatásával kerültek a rajzlapra. Természetesen feltétel, hogy a toll beállításai megfeleljenek, tehát a teknőc tolla le legyen téve. Van olyan utasítás, melynek során a teknőcnek nem kell mozognia, csak az alakzat közepéről irányítja a rajzfolyamatot. Emiatt ezek az eljárások lényegesen gyorsabbak.

A pontméret szám megadásával az adott tollszínnel valamint a szám nagyságának megfelelő átmérőjű pont keletkezik a lapon.

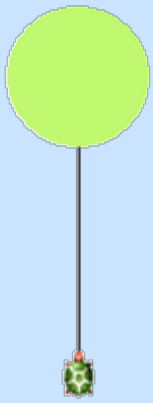


A kiadott utasítás a **pontméret 40** volt.

Készítsünk egy színsor!

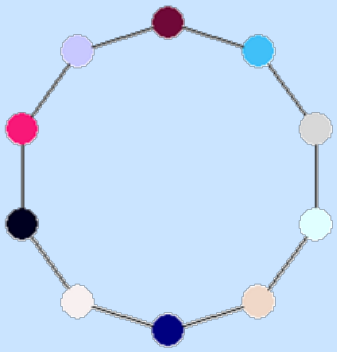
```
tsz! "narancs1
pontméret 50
e 35
tsz! "narancs2
pontméret 50
e 35
tsz! "narancs3
pontméret 50
e 35
tsz! "narancs4
pontméret 50
e 35
tsz! "narancs5
pontméret 50
e 35
tsz! "narancs6
pontméret 50
e 35
tsz! "narancs7
pontméret 50
e 35
tsz! "narancs8
pontméret 50
e 35
tsz! "narancs9
pontméret 50
e 35
tsz! "narancs10
pontméret 50
e 35
tsz! "narancs11
pontméret 50
e 35
tsz! "narancs12
pontméret 50
```

Készítsünk egy lufit!

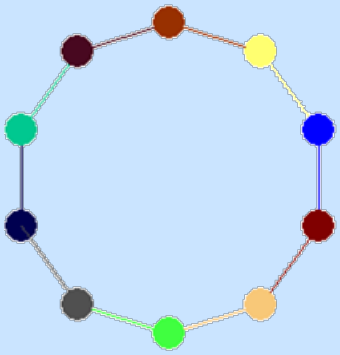


Feladat:

Készítsünk karkötőt (az alapja egy tízszög)!

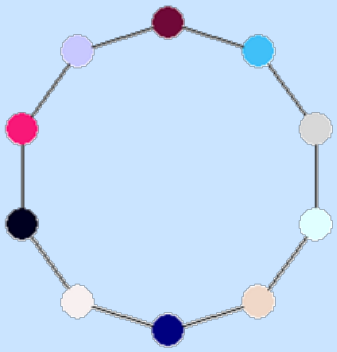
1. 

Itt a karkötő alapja fekete színű legyen!

2. 

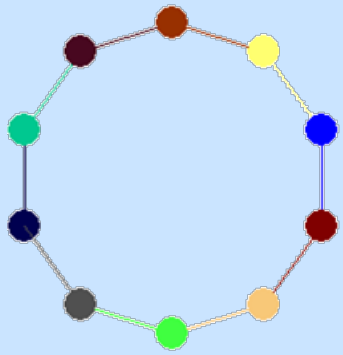
A karkötő alapja a pöttyök színével legyen megegyező a következő pöttyig!

Megoldás

1. 

```
eljárás karkötő2
tízszög
pöttyök
vége

eljárás tízszög
tsz! "fekete
ism 10[e 60 j 36]
vége

eljárás pöttyök
tf
ism 10[tsz! tetsz pontméret 20 e 60
j 36]
tl
vége
```

2.



```
eljárás karkötő
ism 10 [tsz! tetsz pontméret 20 e
60 j 36]
vége
```

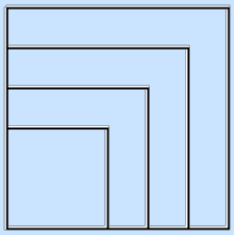
Paraméteres eljárások

Az eddigi eljárásaink parancsai állandó értéket kaptak bemenetként: pl.: a négyzet hossza 50 egység volt. Jó lenne, ha az eljárások is tudnának változtatható értéket fogadni. Így például a négyzet eljárásnak megadhatnánk, hogy milyen hosszú legyen az oldala. Ezután már ezzel a paraméternek nevezett értékkel rajzolhatjuk meg a négyzetet.

A paraméter nevét :-al elválaszt

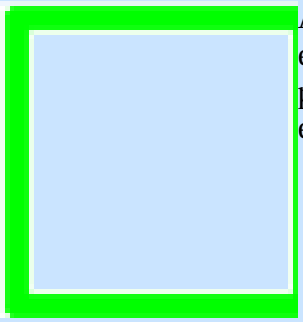
```
eljárás négyzet :hossz
ism 4 [e :hossz j 90]
vége
```

Konkrét érték helyett a paraméter nevét has



Az ábra a paraméteres négyzet eljárás többszöri meghívásával jött létre. Visszahíváskor a paraméter helyett a konkrét értéket írjuk be.

```
négyzet 50
négyzet 70
négyzet 90
négyzet 110
```



Az eljárást úgy készítsük el, hogy a négyzet oldalhossza mellett a szín és a tollvastagság is paraméteres legyen! Visszahíváskor a paramétereknek megfelelő sorrendben kell megadnunk a konkrét értékeket.

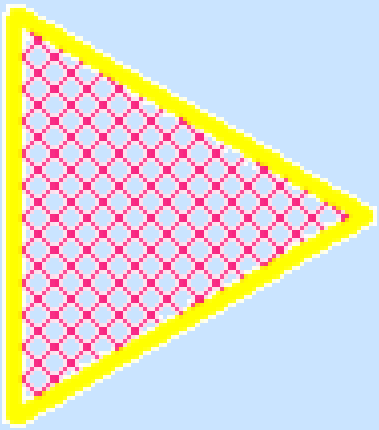
```
eljárás négyzet2 :hossz :szín :vasta
tsz! :szín tv! :vastagság
ism 4 [e :hossz j 90]
vége
```

Az eljárás visszahívása:

```
négyzet2 60 "zöld 5
```

Feladat

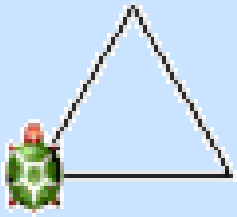
Készítsünk egy háromszög eljárás úgy, hogy az oldalhossz, a szín, a tollvastagság, a kitöltő szín, és a kitöltő minta is paraméterként beállítható legyen!



```
eljárás háromszög2 :hossz :szín
:vastagság :tszín :tminta
tl
tsz! :szín
tv! :vastagság
tlsz! :tszín
tlm! :tminta
ism 3 [e :hossz j 120]
tf j 45 e :hossz*0.2 tölt h :hossz*0.2
b 45 tl
vége
```

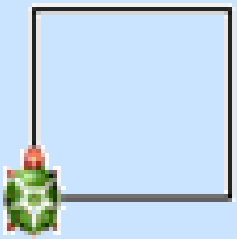
Visszahívása (az ábrán látható eredményért):
`háromszög2 100 "sárga 4 "bíbor 3`

Az eljárások témakörben találkoztunk az alábbi ábrával



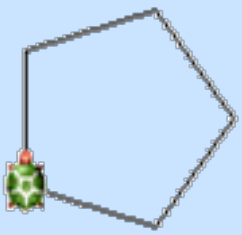
```
eljárás
háromszög
j 30
ism 3 [e 50 j
120]
b 30
vége
```

3*120=360



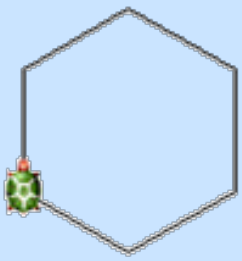
```
eljárás
négyzet
ism 4 [e 50 j
90]
vége
```

4*90=360



```
eljárás ötszög
ism 5 [e 50 j
72]
vége
```

5*72=360



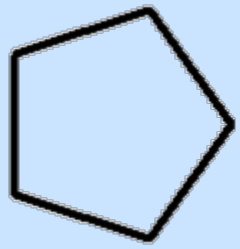
```
eljárás
hatszög
ism 6 [e 50 j
60]
vége
```

6*60=360

Ennek felhasználásával próbáljunk meg elkészíteni egy általános sokszög megrajzolására alkalmas sokszög2 nevű paraméteres eljárást. (Azért nem a sokszög szót használjuk, mert az egy Imagine alapszó.)

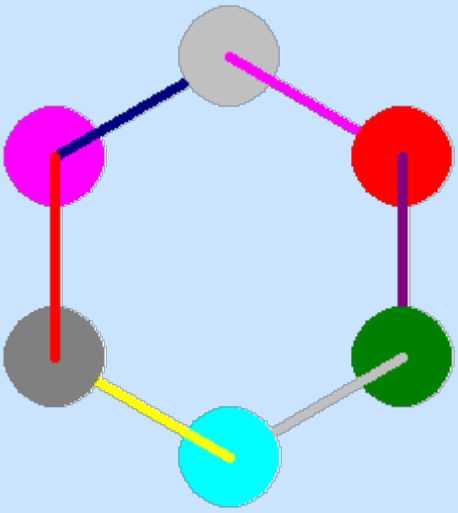
```
eljárás sokszög2 :n :hossz
ism :n [e :hossz j 360/:n]
vége
```

Ha ezt az eljárást visszahívjuk: `sokszög2 5 70`módon, akkor a következő ábrát kapjuk:



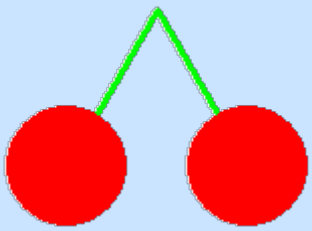
Feladat

Készítsük el a következő ábrát! A pontméretek közötti távolság a pontméret kétszerese legyen!

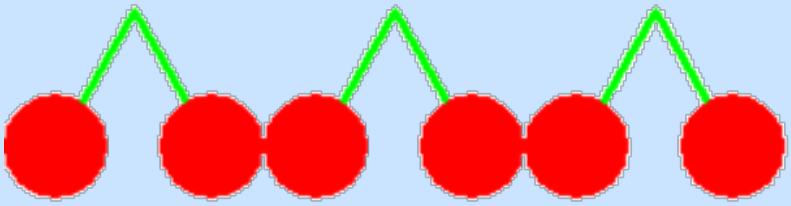


```
eljárás hatszögéskörök :a
törölkép
tv! 5
ism 6 [
tsz! tetsz
pontméret :a*0.5
tsz! tetsz
j 60 e :a]
vége
```

Gyakorlás



Az eljárás neve legyen: cseresznye :a.

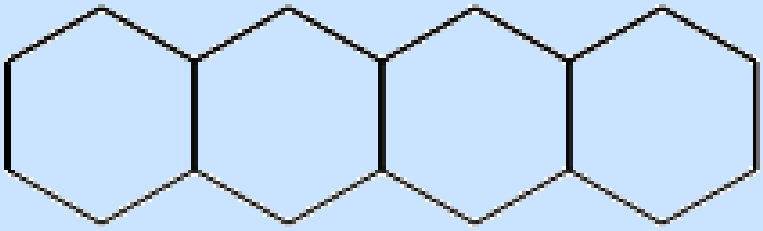


Az eljárás neve legyen: cseresznyesor :a :db.

Paraméteres eljárások gyakorlása

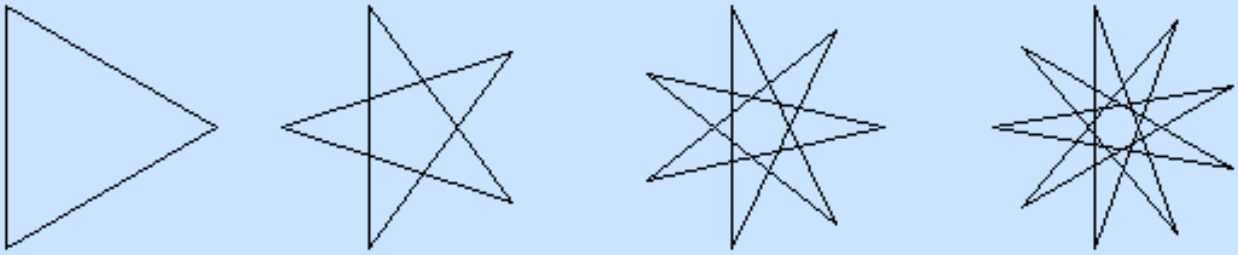
Hatszög sorozat

Paraméter a hatszögek darabszáma és oldalhossza.



Csillagok

Rajzoljunk páratlan oldalszámú csillag-sokszögeket! Paraméter az oldalak száma és hossza.

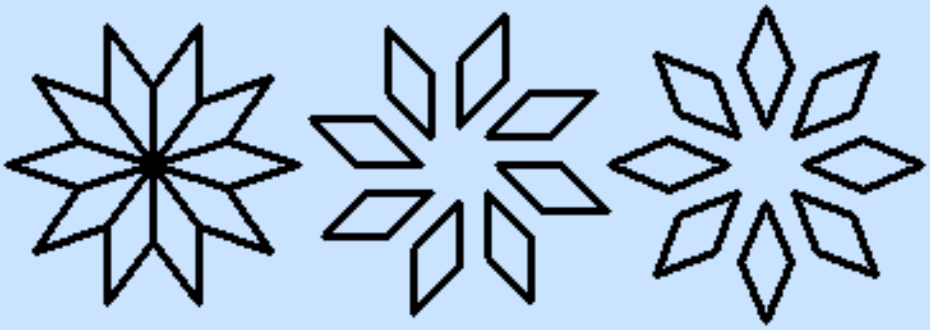


Rombusz

Írjunk rombusz rajzoló eljárást! A rombusz két paramétere az oldalhossz és az alsó és felső csúcsnál található szög (rombusz :m :sz).

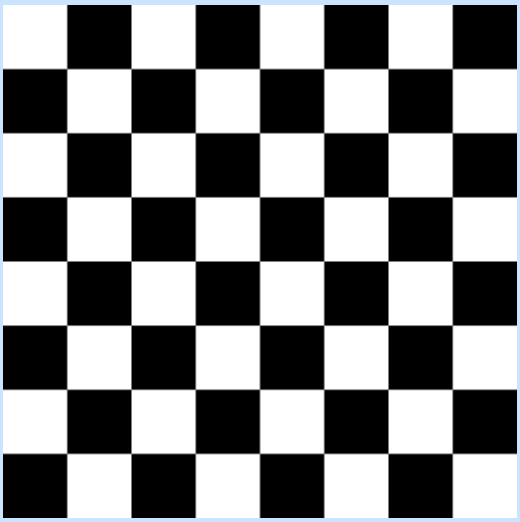
Forgók

Rombuszokból más forgókat rajzolhatunk.

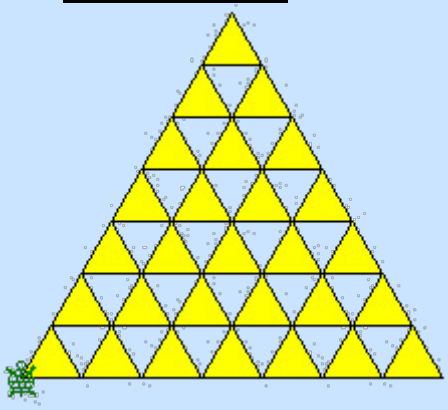


Sakk

Rajzoljunk sakktáblát!

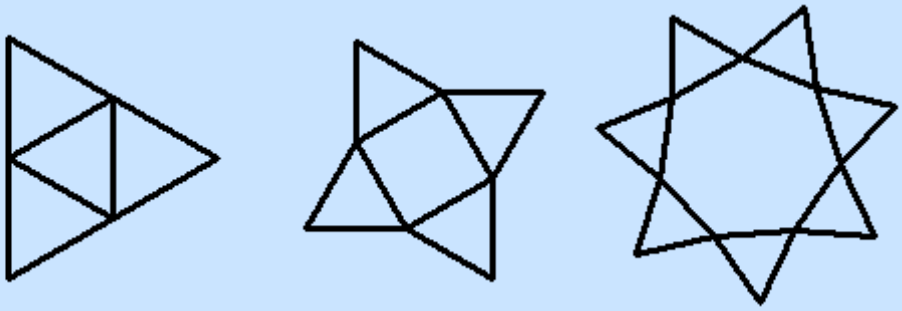


Színes piramis



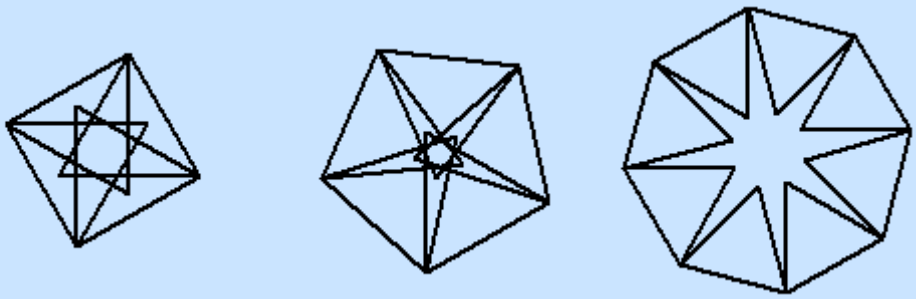
Háromszög oldalú sokszög

A szabályos sokszög minden oldalára rajzoljunk sokszöget!

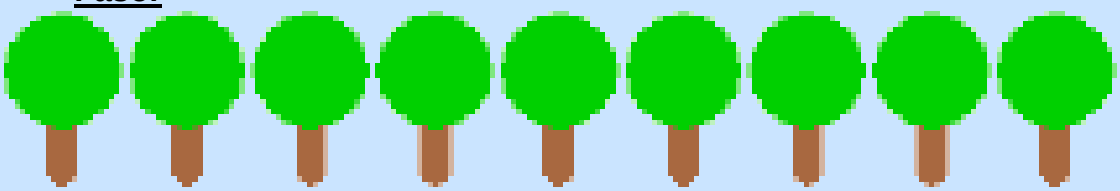


Befelé álló háromszög oldalú sokszög

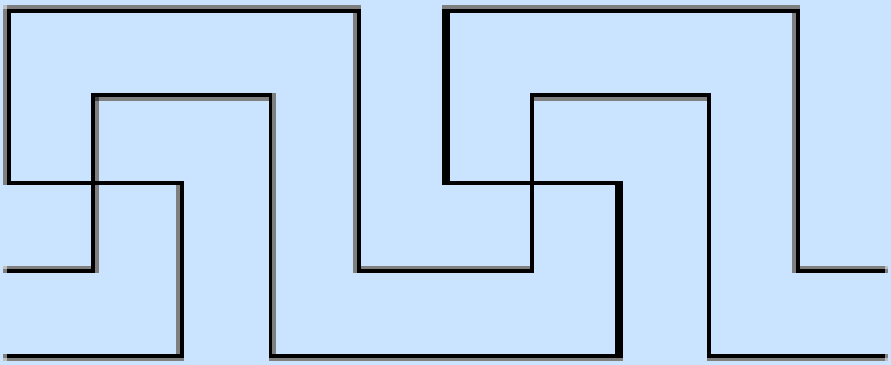
Készítsük el a sokszög mintákat úgy, hogy a szabályos háromszögek befelé állnak!



Fasor



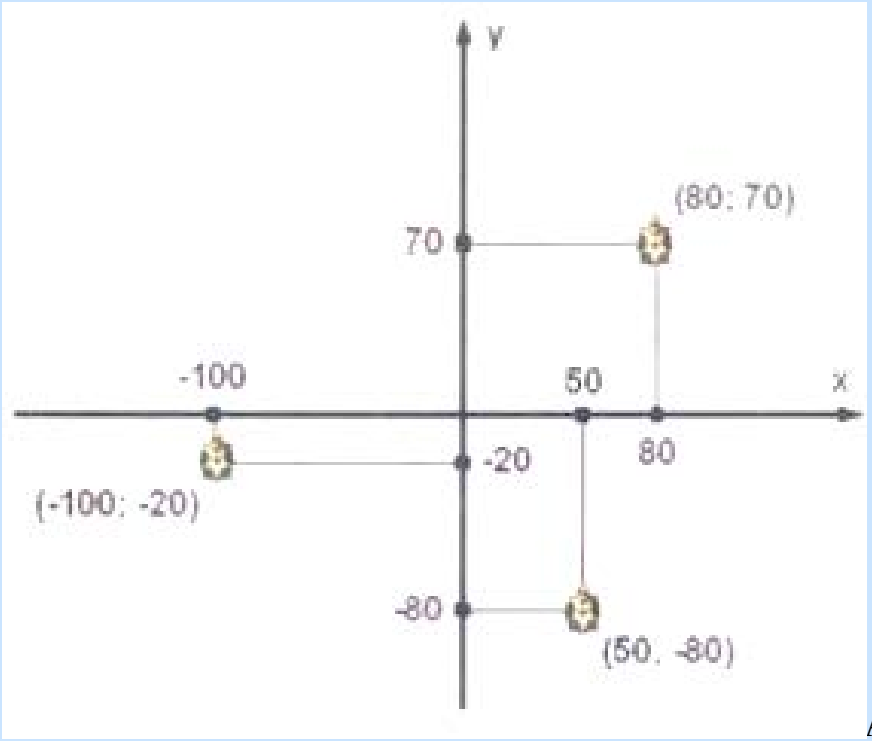
Görögös



Abszolút teknőcutasítás

Az eddigi feladatainkban a klasszikus teknőcgeometriai parancsokat használtuk. Az Imagine azonban lehetőséget biztosít másfajta teknőcpozíció meghatározására.

Az abszolút teknőcutasítások esetében a mozgás és a forgás nem viszonylagos. A teknőc helyét számpárral adjuk meg a Descartes-féle koordináta rendszerben, irányt az égtájakhoz igazítjuk.



Alapértelmezetten a lap vagy panel középpontja a koordináta-rendszer origója.

Pozíció megadása:

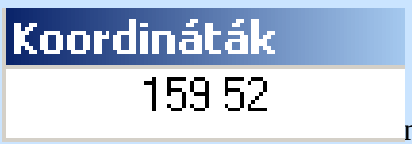
`? kipoz`utasítás hatására megkapjuk teknőc aktuális pozícióját
`0 0`lesz az eredmény (a teknőc az alappozícióban a lap közepén helyezkedik el)

A **poz!**parancs hatására a teknőc bárhol is volt az adott pozícióba mozdul.

A parancs használata:

`poz! [100 43]` (azaz koordináta megadása)

poz! után Enter vagy F9 leütésével megjelenik egy segítő:



melynek segítségével a kívánt koordináta könnyedén beállítható, s Enter lenyomásával a teknőc elhelyezkedik ebben a pozícióban. Ha a teknőc tolla le van téve (alapértelmezés), akkor az eredeti pozíciójától az új pozícióig vonalat is húz.

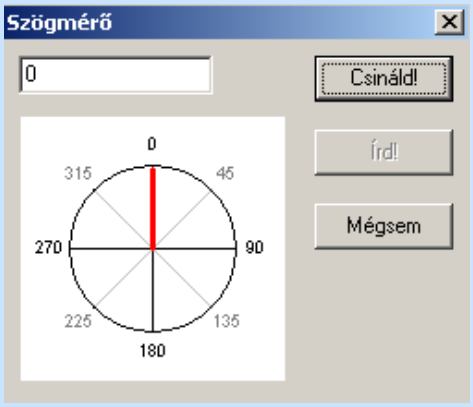
Irány megadása:

`? ki irány`utasítás hatására megkapjuk teknőc aktuális irányát
`0` lesz az eredmény (a teknőc az alappozícióban fölfelé (észak felé) néz)

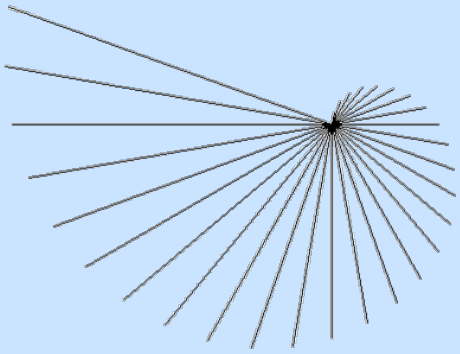
A parancs használata:

`irány! 120` (az irány megadása, ide fordul bármerre nézett is)

irány!után Enter vagy F9 leütésével megjelenik egy segítő (az alapvető fordulásoknál megismert):



Az irány parancs érdekessége:

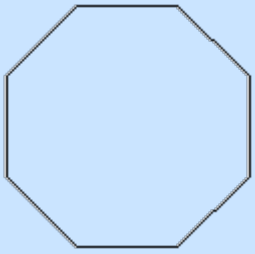


```
ism 30 [e irány h irány jobbra 10]
```

A **haza** parancs hatására a teknőc visszatér arra a helyre, ahol létrehozták, s abba az irányba néz, mint létrehozáskor; vonalat ebben az esetben akkor sem húz, ha a toll le van téve.
A Shift lenyomva tartása mellett a jobb egérgombbal a teknőc szabadon vonszolható. és közben vonalat még akkor sem húz, ha a tolla lent van.

Feladat:

1. Rajzoljunk egy szabályos nyolcszöget, s a csúcsoknál kérdezzük le a teknőc pozícióját és koordinátáit! Az eljárás neve nyolcszög2, s az oldal paraméterrel legyen megadva!

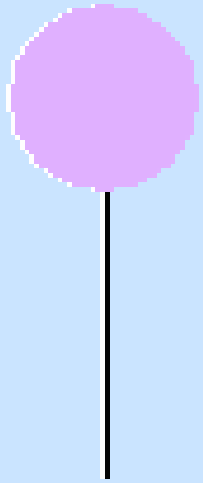


```
eljárás nyolcszög2
:a
ism 8 [(ki "poz:
poz "irány: irány)
e :a j 45]
vége
```

nyolcszög2 50 esetén:

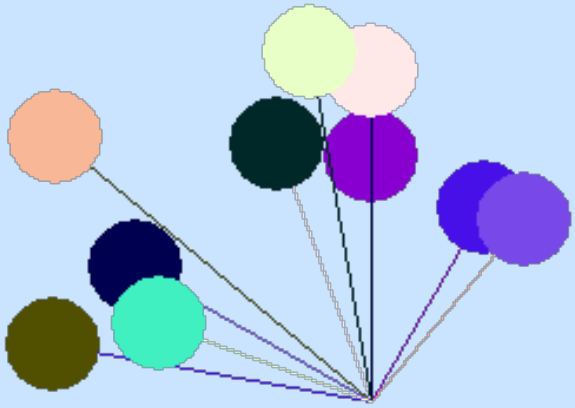
poz: 0 0 irány: 0
poz: 0 50 irány: 45
poz: 35.3553 85.3553 irány: 90
poz: 85.3553 85.3553 irány: 135
poz: 120.711 50 irány: 180
poz: 120.711 0 irány: 225
poz: 85.3553 -35.3553 irány: 270
poz: 35.3553 -35.3553 irány: 315

2. Készítsünk egy lufi eljárást paraméteresen! (A paraméter a lufi átmérője legyen! A lufi szára a lufi kétszerese!)



```
eljárás lufi :a
tsz! "fekete
e :a*2
tsz! tetsz
pontméret :a
tf
h :a*2
tl
vége
```

3. Készítsünk egy luficsokor eljárást paraméteresen! (A paraméter a lufi átmérője legyen! A lufi szára minimum a lufi átmérőjének kétszerese legyen valamint a lufik között a kiindulási helytől számítva 10 fokos fordulat!)



```
eljárás luficsokor :a
ism 10 [e :a*2+tetsz tsz!
tetszpontméret :a haza
irány! kiválaszt [270 280
290 300 310 320 330 340
 350 360 10 20 30 40 50
60 70 80 90]]
vége
```

Elemezzük az eljárás néhány részletét:

`e :a*2+tetsz` Ezzel biztosítjuk, hogy a szár a lufi átmérőjének minimum kétszerese legyen, mert a kétszereséhez még hozzáadunk egy tetszőleges számot, azaz egy véletlenszerű értéket. Értéke intelligens módon a parancstól függően határozza meg az értékkészletet és a tartományt! Vannak korlátai is, mivel nem használható minden parancs bemeneteként, valamint nem határozhatjuk meg az intervallumot, ahonnan az értéket választja.

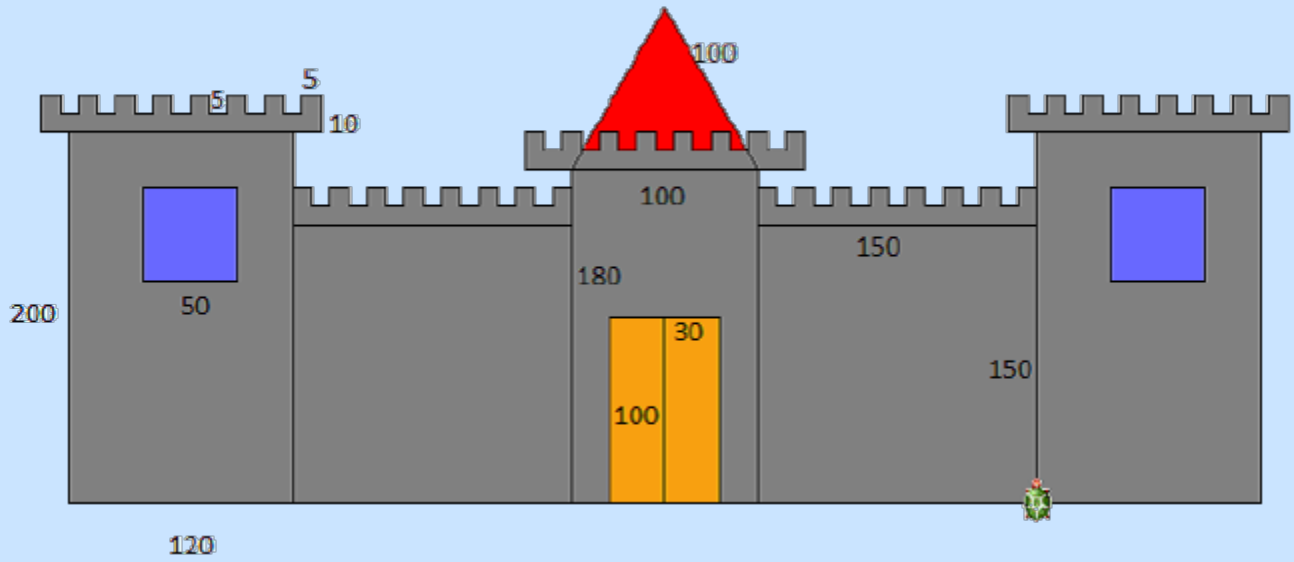
`kiválaszt [270 … 90]` A kiválaszt parancs, a paraméterként megadott lista véletlenszerűen kiválasztott elemével tér vissza: azaz ebben az esetben az irány lehet 270 foktól 90 fokig, azaz a felső félkör 10 fokonként megadott értékei közül lesz kiválasztva.

A tetszőleges és kiválaszt parancsok mellett jó, ha ismerjük a véletlenszám-ot is.

A **véletlenszám** / **vszám***szám* egy nulla és a szám közötti egész értéket ad vissza.

Bonyolultabb ábrák készítése I.

Ha egy összetett feladatot akarunk megoldani, a feladatunkat lépésekre kell bontani.



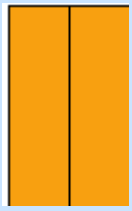
Nézzük, hogy milyen alapokból tevődik össze:

ablak

```
eljárás ablak
tl tsz! "fekete
ism 4 [e 50 j 90]
tf tsz! "kék8
j 45 e 20
tölt
h 20 b 45
tsz! "fekete
tl
vége
```

önálló

ajtó



```
eljárás ajtó
tl tsz! "fekete
ism 2 [e 100 j 90 e 30
j 90]
tf tsz! "narancs
j 45 e 20
tölt
h 20 b 45
tsz! "fekete tl
j 90 e 30 b 90
ism 2 [e 100 j 90 e 30
j 90]
tf
tsz! "narancs
j 45
e 20
tölt
```

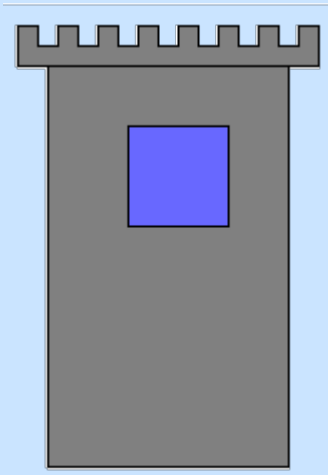
önálló

lőrés

```
h 20
b 45
tsz! "fekete
tl
vége
eljárás lőrés
e 20
j 90
ism 7 [ e 10 j 90 e 10
b 90
e 10 b 90 e 10 j 90]
e 10 j 90 e 20 j 90
vége
```

önálló

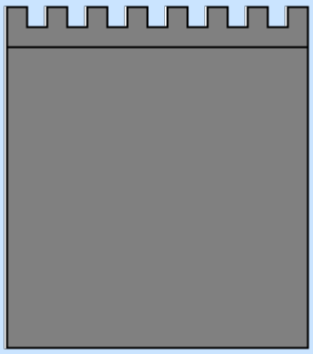
bástya



```
eljárás bástya
ism 2 [e 200 j 90 e
120 j 90]
e 200 b 90
e 15 j 90
lőrés
e 135 j 90
h 200 tf
j 90 e 10
b 90 e 20
tsz! "szürke tölt
e 190 tölt
h 210 b 90 e 10 j 90
tsz! "fekete
tltf
e 120 j 90
e 40 b 90
ablak
tf b 90
e 40 j 90
h 120
tl
vége
```

lőrés
ablak

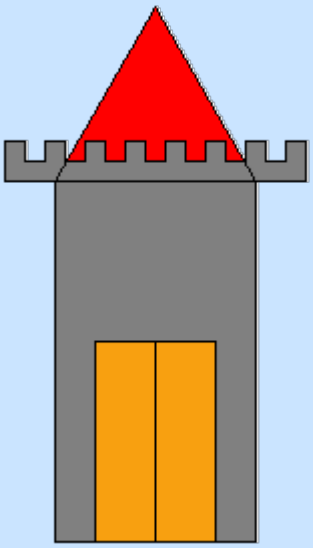
bástya2



```
eljárás bástya2
ism 2 [e 150 j 90 e
150 j 90]
e 150 lőrés
e 150 j 90
h 150 tf
j 90 e 5
b 90 e 20
tsz! "szürke tölt
e 132 tölt
h 152 b 90 e 5 j 90
tsz! "fekete tl
vége
```

lőrés

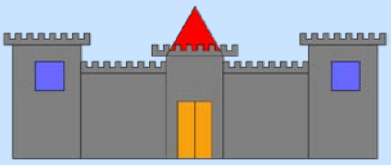
bástya3



```
eljárás bástya3
ism 2 [e 180 j 90 e
100 j
90]
e 180 b 90
e 25 j 90
lőrés e 125
j 90 h 180
j 90 e 20 b 90
ajtó
tf j 90 h 50 b 90 tl
tf j 90 e 5 b 90
e 20 tsz! "szürke tölt
e 164 tölt
h 184 b 90 e 5 j 90
tsz! "fekete
tl e 180 b 90
ism 3 [j 120 e 100]
j 90 h 180 tf
j 90 e 55 b 90 e 240
tsz! "piros tölt
h 240 b 90 e 55 j 90
tsz! "fekete tl
vége
```

ajtó

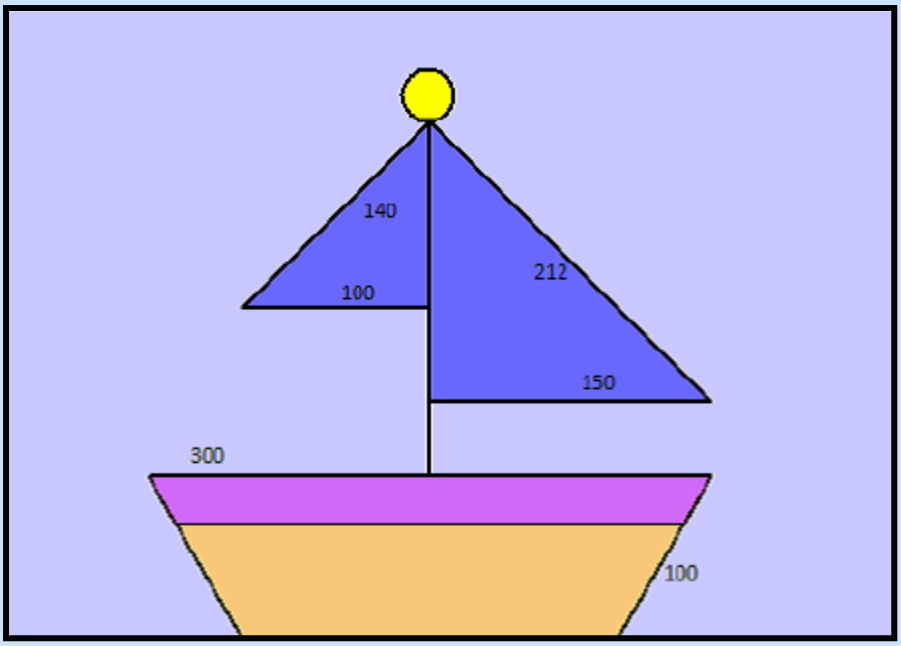
vár



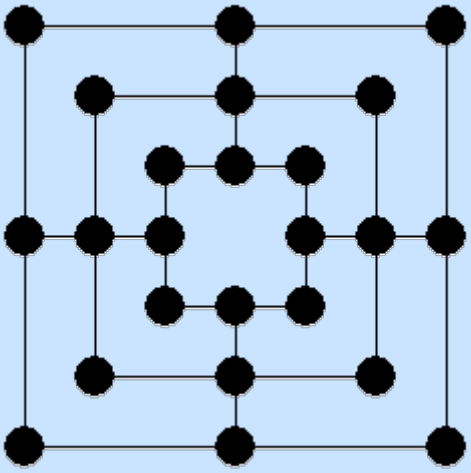
```
eljárás vár
tf h 100 b 90 e 300 j
90
tl
bástya
j 90 e 120 b 90
bástya2
j 90 e 150 b 90
bástya3
j 90 e 100 b 90
bástya2
j 90 e 150 b 90
bástya
vége
```

bástya
bástya2
bástya3

Gyakorlás

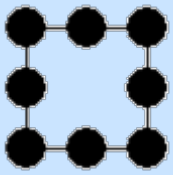

140
100
212
150
300
100

Bonyolultabb ábrák készítése II.



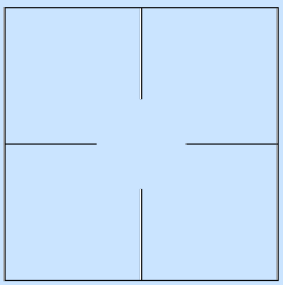
Bontsuk a feladatot részekre!

malomalap



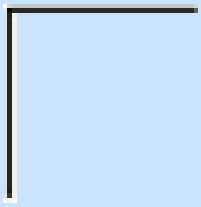
```
eljárás malomalap :meret
ism 4 [ism 2 [e :meret
pontméret 20 ] j 90]
vége
```

összekötő



```
eljárás összekötő :meret
ism 4 [
j 90
e :meret*3
b 90
e :meret*2
h :meret*2
j 90
e :meret*3
j 180]
vége
```

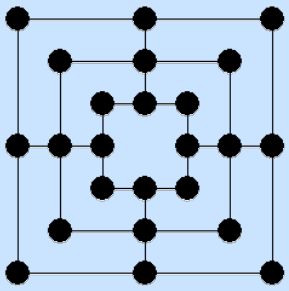
következő



(A művelet eredménye nem látható, de ez történik.)

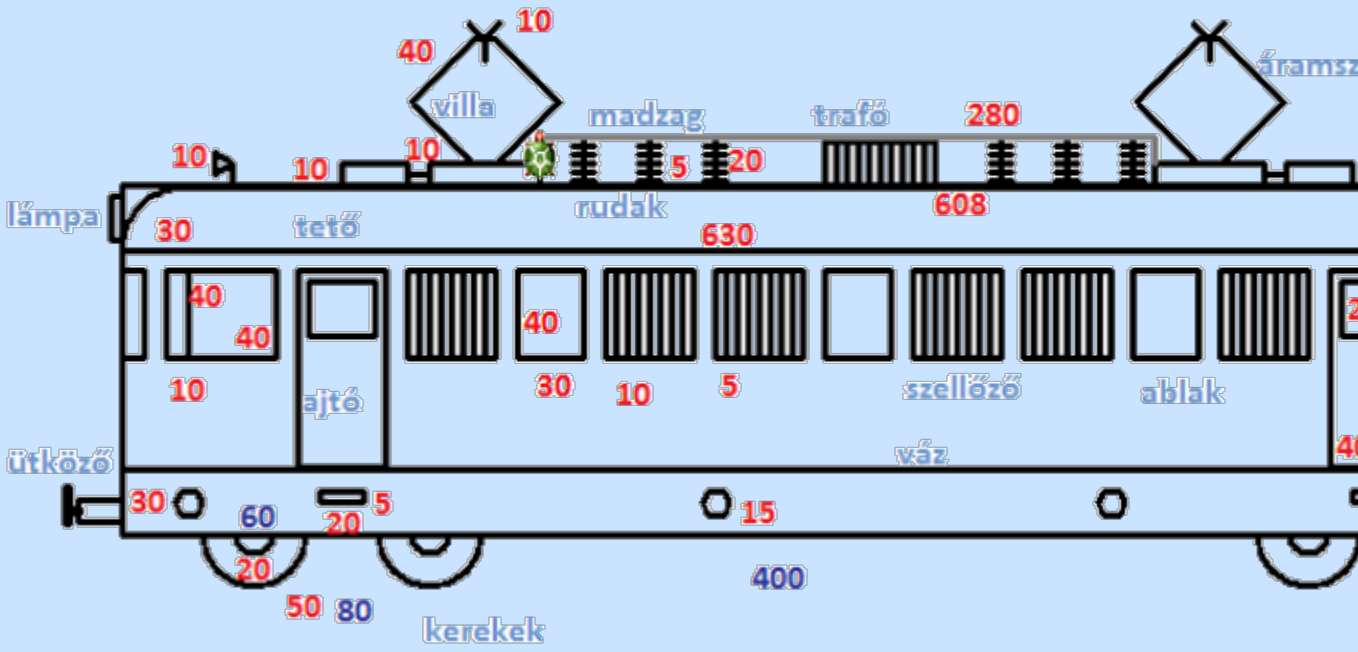
```
eljárás következő :meret
tf
b 90
e :meret
b 90
e :meret
j 180
tl
vége
```

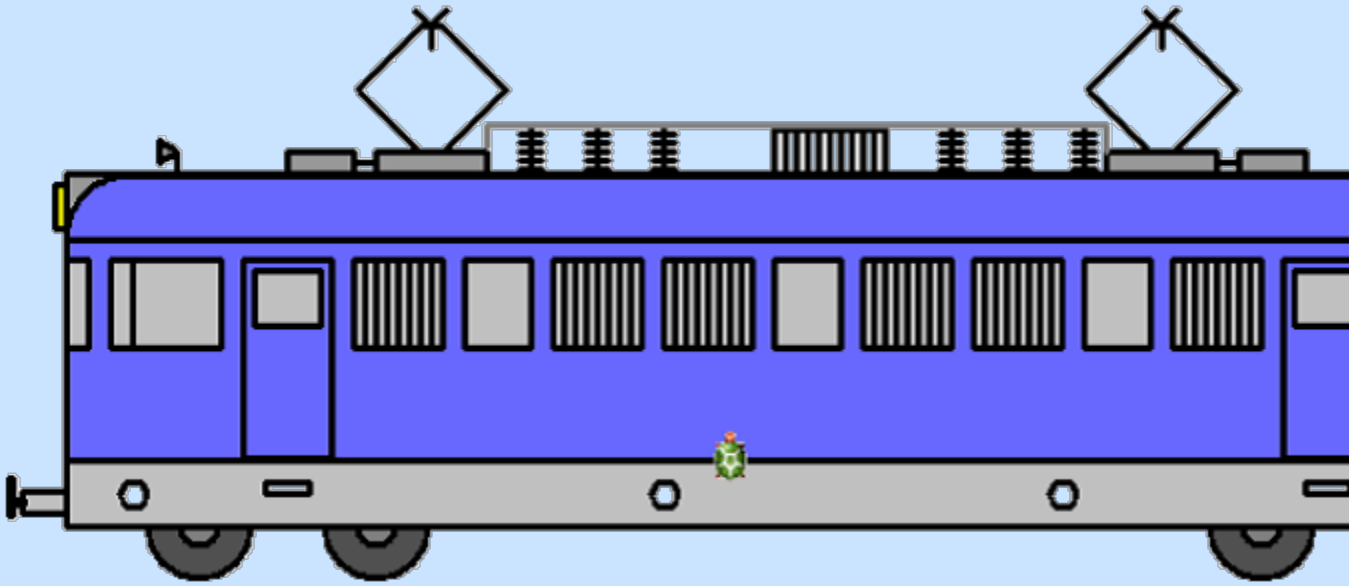
malom2



```
eljárás malom2 :meret
malomalap :meret
következő :meret
malomalap :meret*2
következő :meret
malomalap :meret*3
összekötő :meret
vége
```

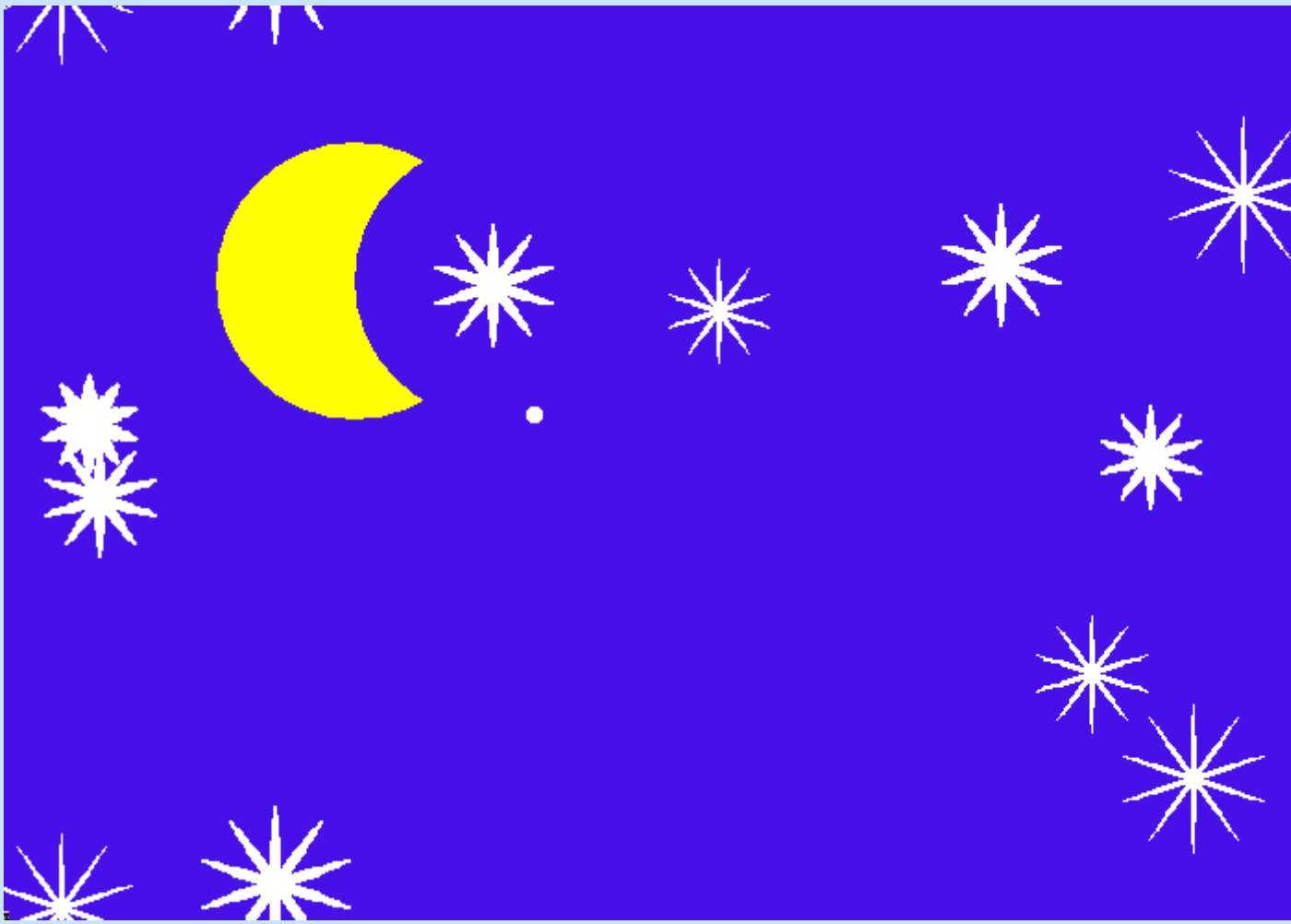
Gyakorlás





Ábrák részekre bontása

Éjszakai égbolt készítése:

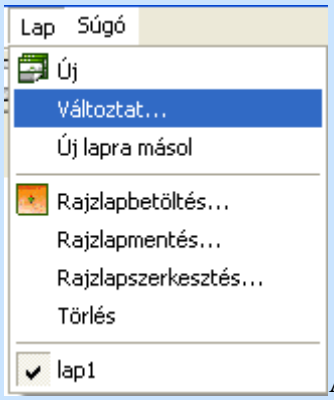


Bontsuk a feladatot részekre! Mit látunk a képen?

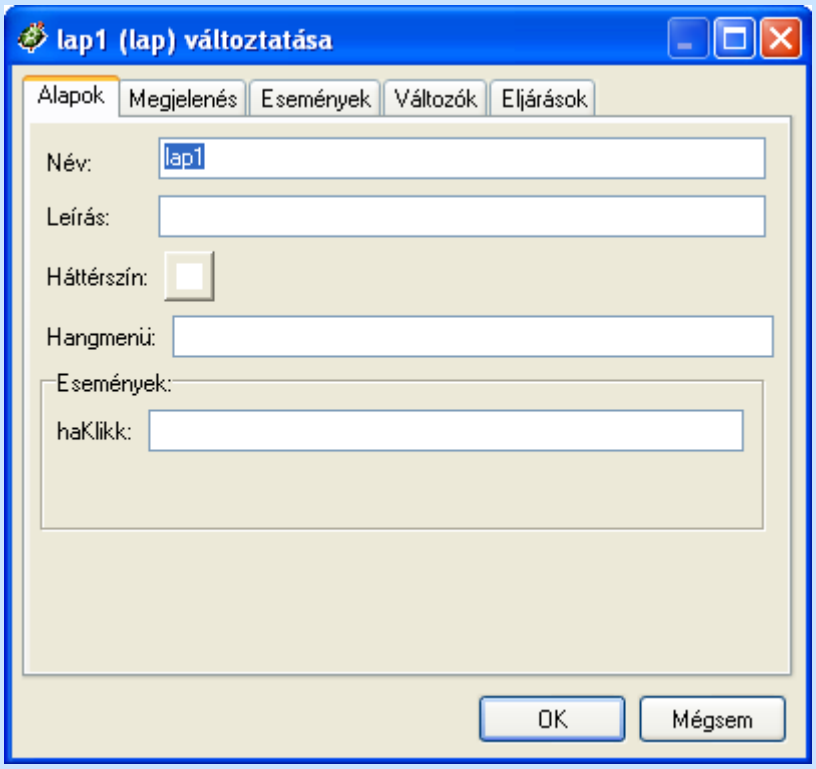
Csillagok, hold, háttér.

Kezdjük a háttérszín beállításával!

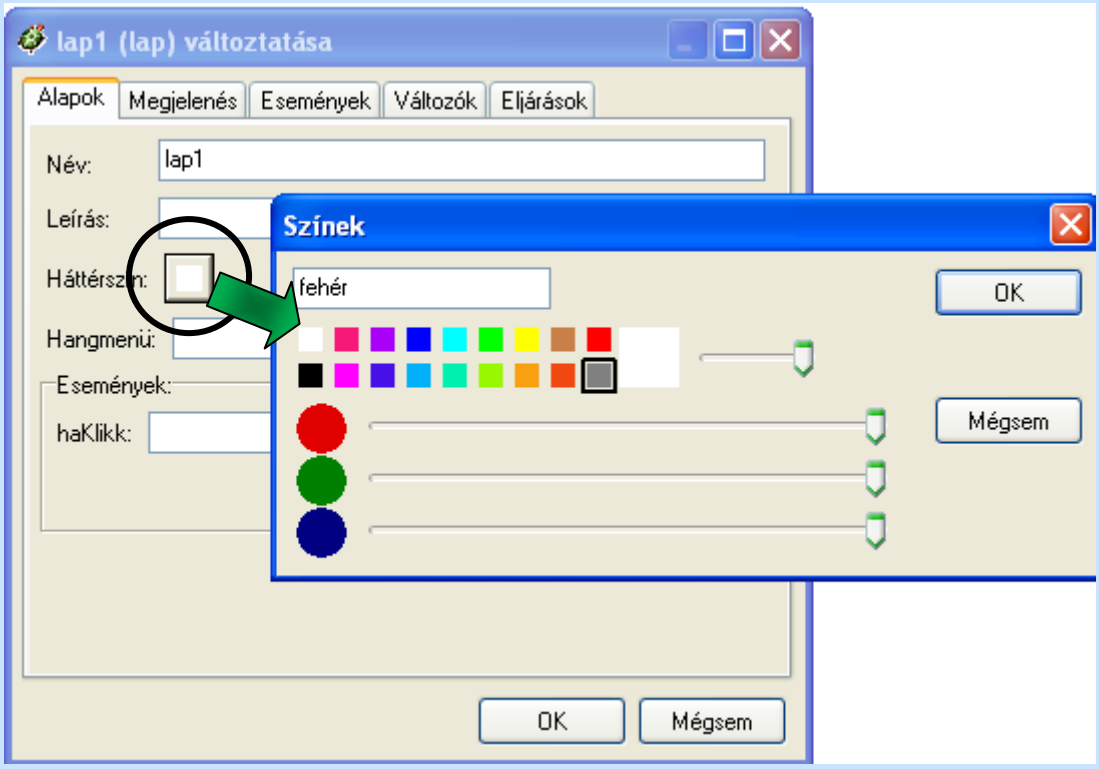
A háttérszín beállítására több lehetőségünk van:



A LAP menüben a Változtat pontra kattintva a következő ablak jelenik meg:



Az Alapok fülön a Háttérszín beállíthatjuk a kívánt színre (esetünkben ez sötétkék).



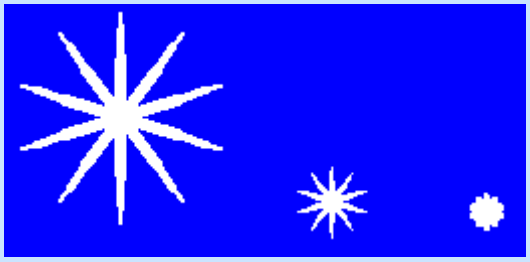
Természetesen a beállítás megtörténhet parancs segítségével is: **háttérszín! / hsz!** ”sötétkék.

Csillagok:

Egy egyszerű csillag eljárást készíthetünk egy egyre csökkenő tollvastagságú fehér vonal többszöri elforgatásával:

```
eljárás csillag
tl
tsz! "fehér
ism10[tv! 3 e 15 tv! 2 e 10
tv! 1 e 5
h 30 j 36]
vége
```

Ha viszont ezt tennénk rá az éjszakai égboltra, nem kapnánk túl meggyőző képet, mert minden csillag egyforma lenne.

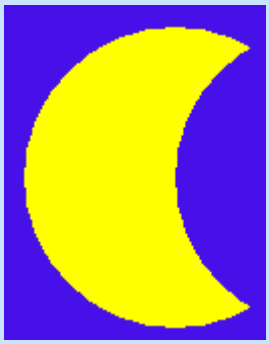


Készítsünk egy olyan paraméteres csillag eljárást, melyben felhasználjuk az előző csillagnál kialakított elvet, de a paraméter a vonal hosszúsága és a tollvastagság legyen:

```
eljárás csillag2 :a :v
tsz! "fehér
ism10[tv! :v*3 e :a tv! :v*2 e
:a/2
tv! :v e :a/4
h :a*7/4 j 36]
vége
```

```
csillag2 30 2
csillag2 10 1
csillag2 5 2
```

A hold elkészítése pontméret segítségével történhet:



```
eljárás hold
tsz! "sárga
pontméret 150
j 90 tf e 75
b 90
tsz! "sötétkék
pontméret 150
vége
```

Utolsó lépésként meg kell oldani, hogy az elkészített részek egy képet alkossanak.

```
eljárás égbolt2
törölkép
hsz! "sötétkék
poz! [-200 100] hold
tfismvszám 20 [ poz! tetsztl
csillag2 vszám 25 kiválaszt [1 2
3] tf]
vége
```

Elemezzük:

```
eljárás égbolt2
törölkép
hsz! "sötétkék
poz! [-200 100] hold
tfismvszám 20 [ poz! tetsztl csillag2 vszám 25 kiválaszt [1 2 3] tf]
vége
```

Képernyőtörléssel kezdjük!

Beállítjuk a háttérszín!

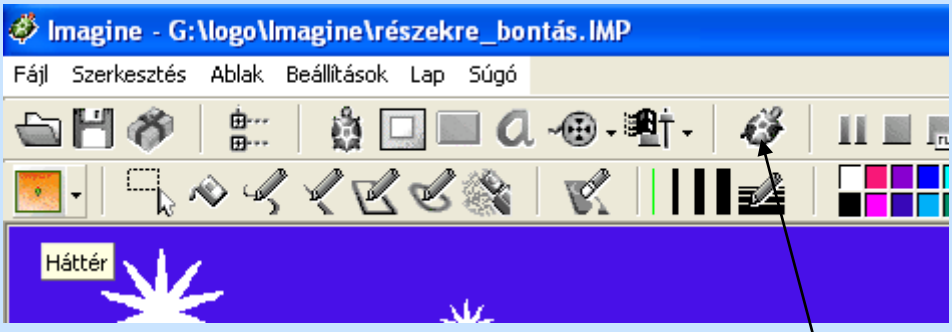
A holdat elhelyezzük adott pozícióba

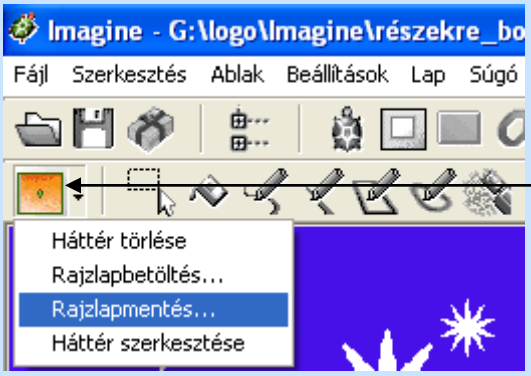
Ezzel korlátozzuk a csillagok számát (előfordulhat, hogy egy csillag sem lesz az égen)!

A csillagok pozíciója tetszőleges!

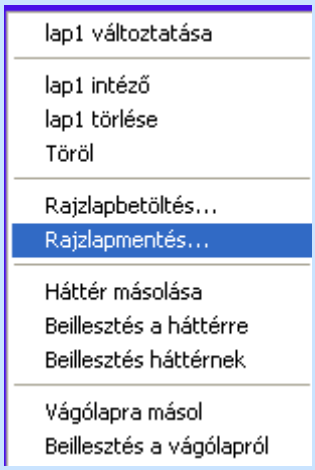
A csillag2 eljárás 2 paraméterrel rendelkezik: az eg
hossza, melyet egy véletlen számmal hívunk vissza
pedig a vonalvastagság volt melyet most tetszés sz
az 1 2 3 számok közöl. Ezzel érhetjük el, hogy a la
csillagok változatosságot mutassanak!

Az elkészített rajzunkat mentsük el képként!

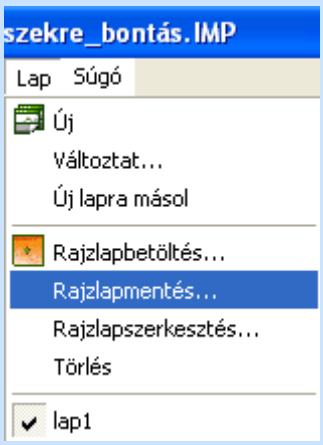




Festő eszköztár háttér ikonjára kattintva és a Rajzlapmentés pontot választva!



A helyi menü Rajzlapmentés pontját választva!



A lap menüben a Rajzlapmentést választva!

A rajz a **kepsor** mappába kerül BMP kiterjesztéssel!

Gyakorlás

Mutatás és elrejtés

Eddigi feladatainkban a munka során a teknőc mindig látható volt, néha azonban hasznos, ha nem mindig látható.

Teknőc nem látható	Teknőc látható

`elrejtteknőc`
`rejttek`
`látható! "hamis`
	`mutatteknőc`
	`mutattek`
	`látható! "igaz`

Ahhoz, hogy adott pillanatban a teknőc látható vagy nem a látható függvényt kell használni:

`? ki látható`
`igaz`esetén a teknőc látszik

A teknőcnek beállítható egy olyan téglalap alakú terület, ahol mozgás közben látszik. Rajzolni ezen kívül is rajzol de nem látszik.

Alapbeállítás esetén az egész lap vagy panel a teknőc látható munkaterülete.

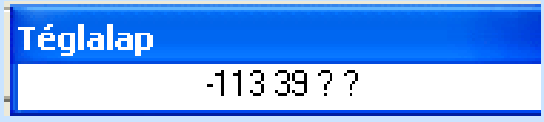
Az ábrán látható, hogy a teknőc már csak félig látható, mert éppen kilép a látható tartományból.

A látható tartomány beállítása a láthatótartomány! / láthtart! paranccsal, ami után meg kell adnunk a téglalap jellemzőit:

`? láthatótartomány! [-100 30 150 90]`

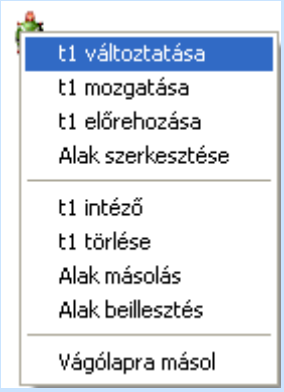
Az első számpár koordinátáival a téglalap bal felső sarkát jelöli, a második számpár pedig a szélességét és a magasságát adja.

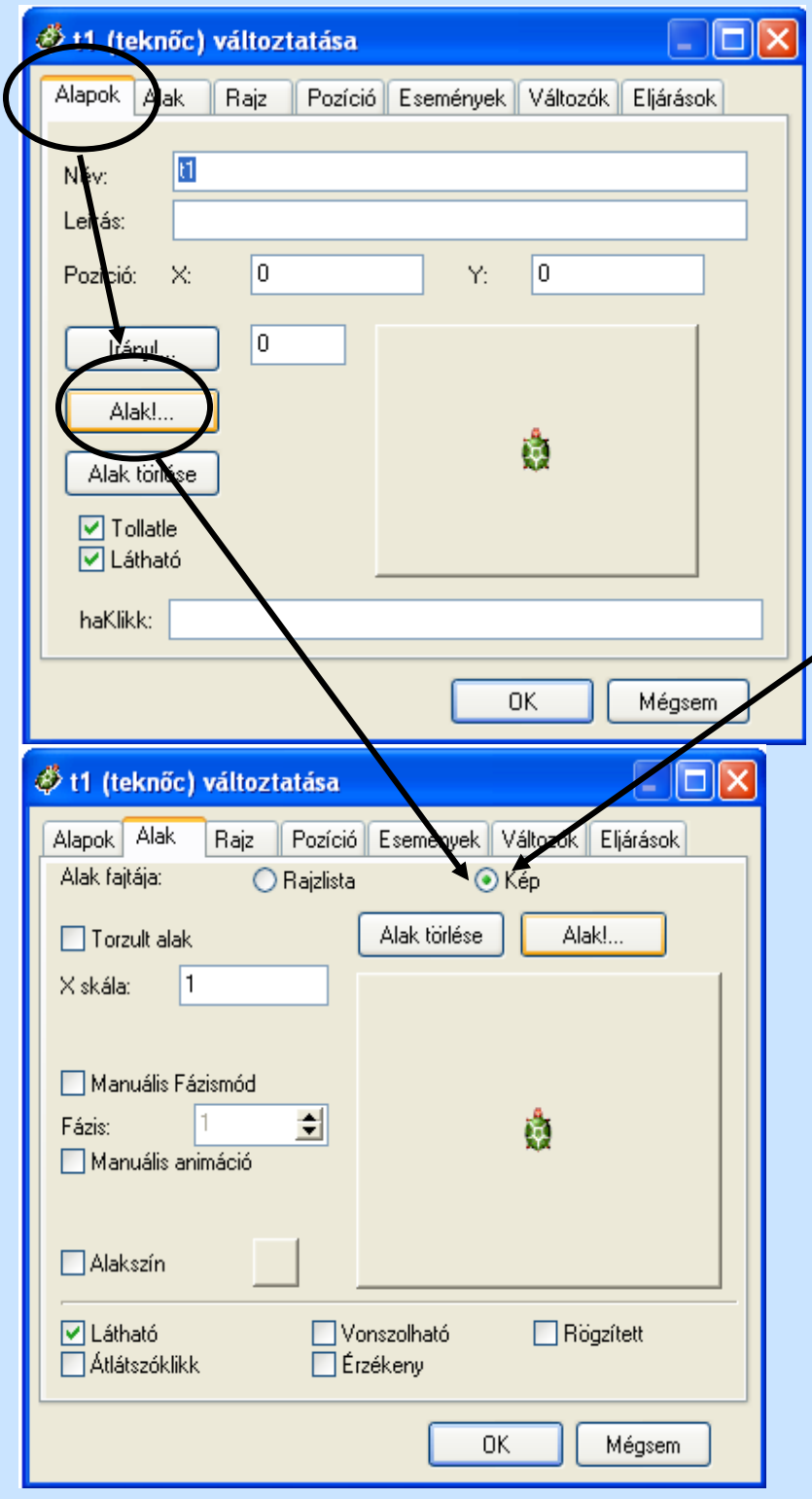
Ha a **`láthatótartomány!`** parancs után Enter-t nyomunk, az egérrel körbekeríthetjük a téglalap alakú területet:

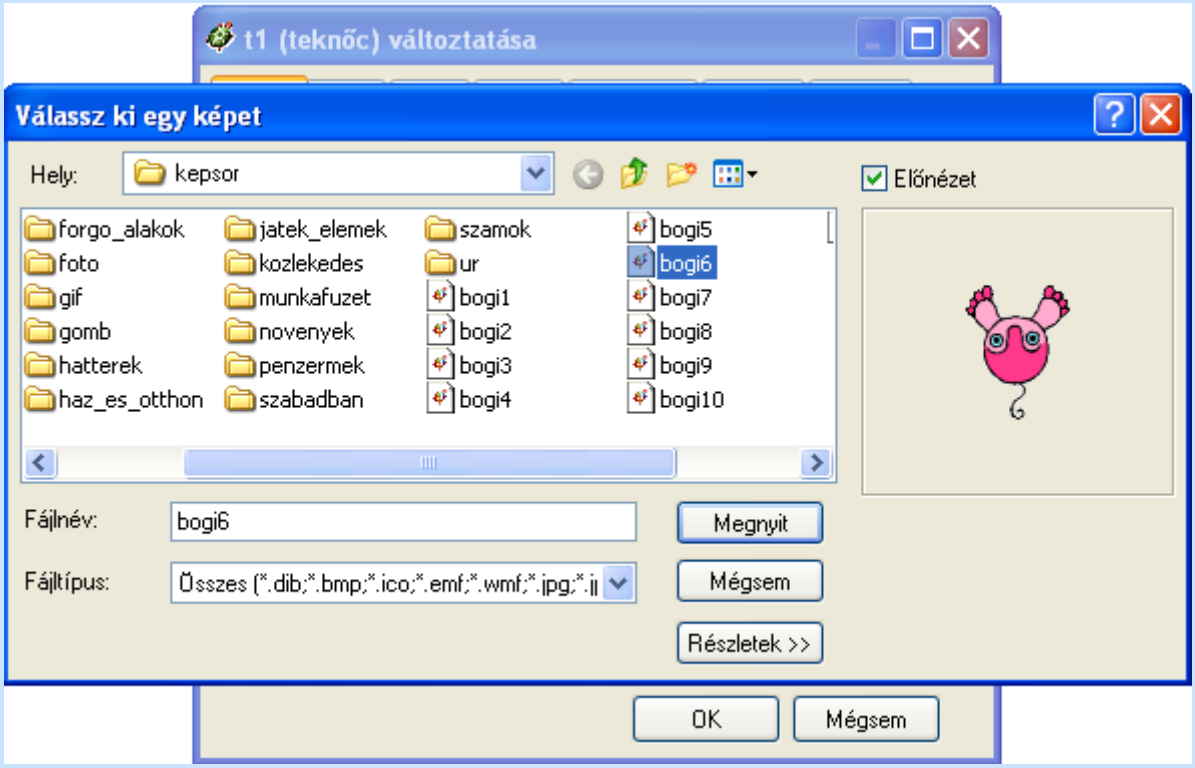


A teknőc alakja

A lapon a teknőc alapalakkal  jön létre. Természetesen ez megváltoztatható. A alak egyszerű rajztól kezdve iránynak megfelelően változó animáció is lehet. Az alak megváltoztatása a teknősre való jobb egérkattintással tehető meg:


t1 változtatása
t1 mozgatása
t1 előrehozása
Alak szerkesztése
t1 intéző
t1 törlése
Alak másolás
Alak beillesztés
Vágólapra másol


t1 (teknőc) változtatása
Alapok
Alak
Rajz
Pozíció
Események
Változók
Eljárások
Név:
t1
Leírás:
Pozíció:
X:
0
Y:
0
Alak!...
Alak törlése
Tollatle
Látható
haKlikk:
OK
Mégsem
t1 (teknőc) változtatása
Alapok
Alak
Rajz
Pozíció
Események
Változók
Eljárások
Alak fajtája:
Rajzlista
Kép
Torzult alak
Alak törlése
Alak!...
X skála:
1
Manuális Fázismód
Fázis:
1
Manuális animáció
Alakszín
Látható
Vonszolható
Rögzített
Átlátszóklikk
Érzékeny
OK
Mégsem



A képválasztó ablakból kiválaszthatjuk a teknőc alakjának szánt képet, mely bármilyen szabványos képformátumú kép lehet.

Amennyiben az előzőekben megadott lapokon a teknős előnézeti képére kattintunk, elindul a LogoMotion szerkesztő, ahol szerkeszthetjük a képünket:

A teknőcalak beállításait megtehetjük parancssoron keresztül az **alak!**parancs használatával.

```
alak! "bogi3
```

A bemenet egy szabványos Imagine által támogatott képfájl.

```
alak! [tlsz! "zöld sokszög [ism
6 [e 20 j 60]]]
```

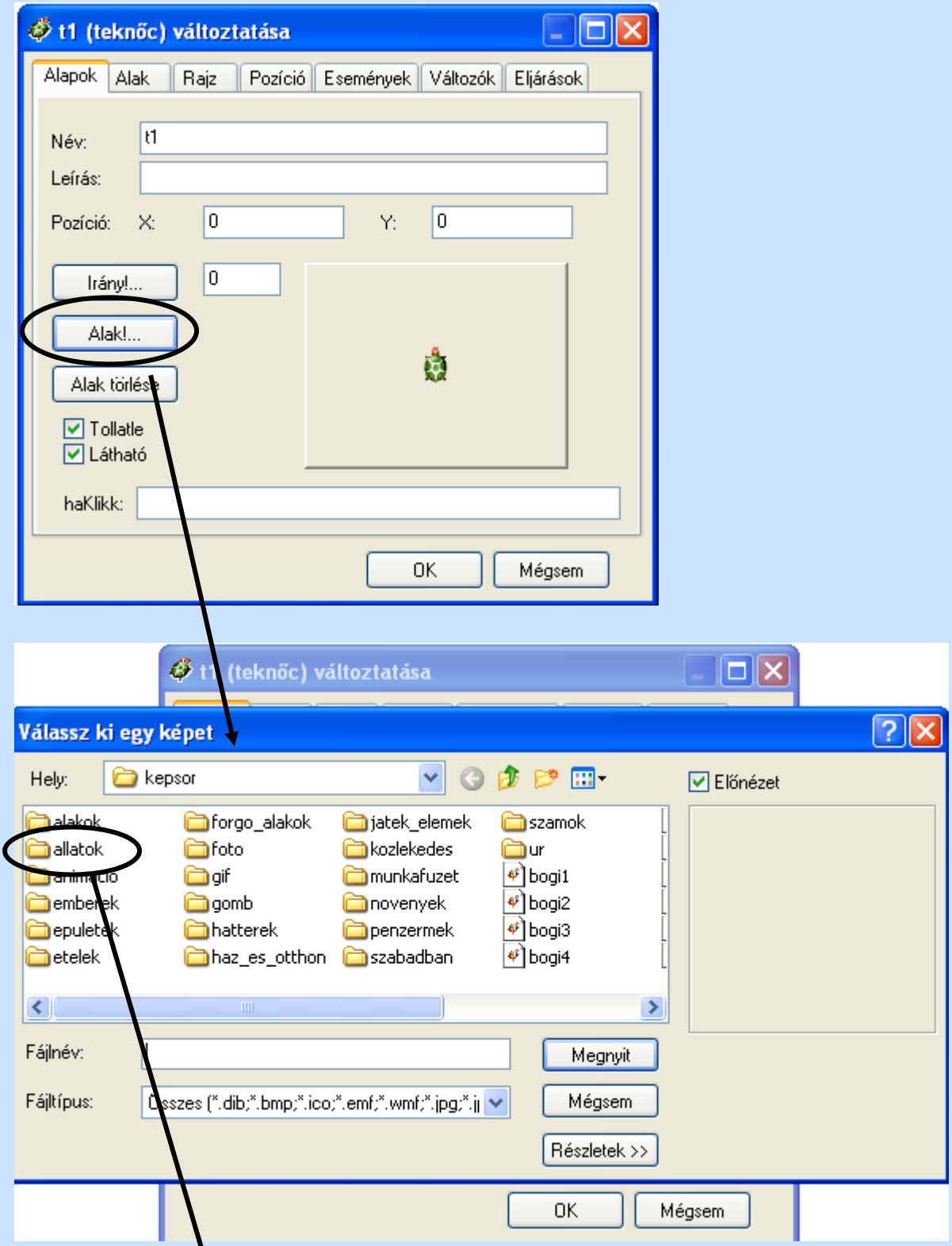
A bemenet egy szabványos Imagine által támogatott rajzlista.

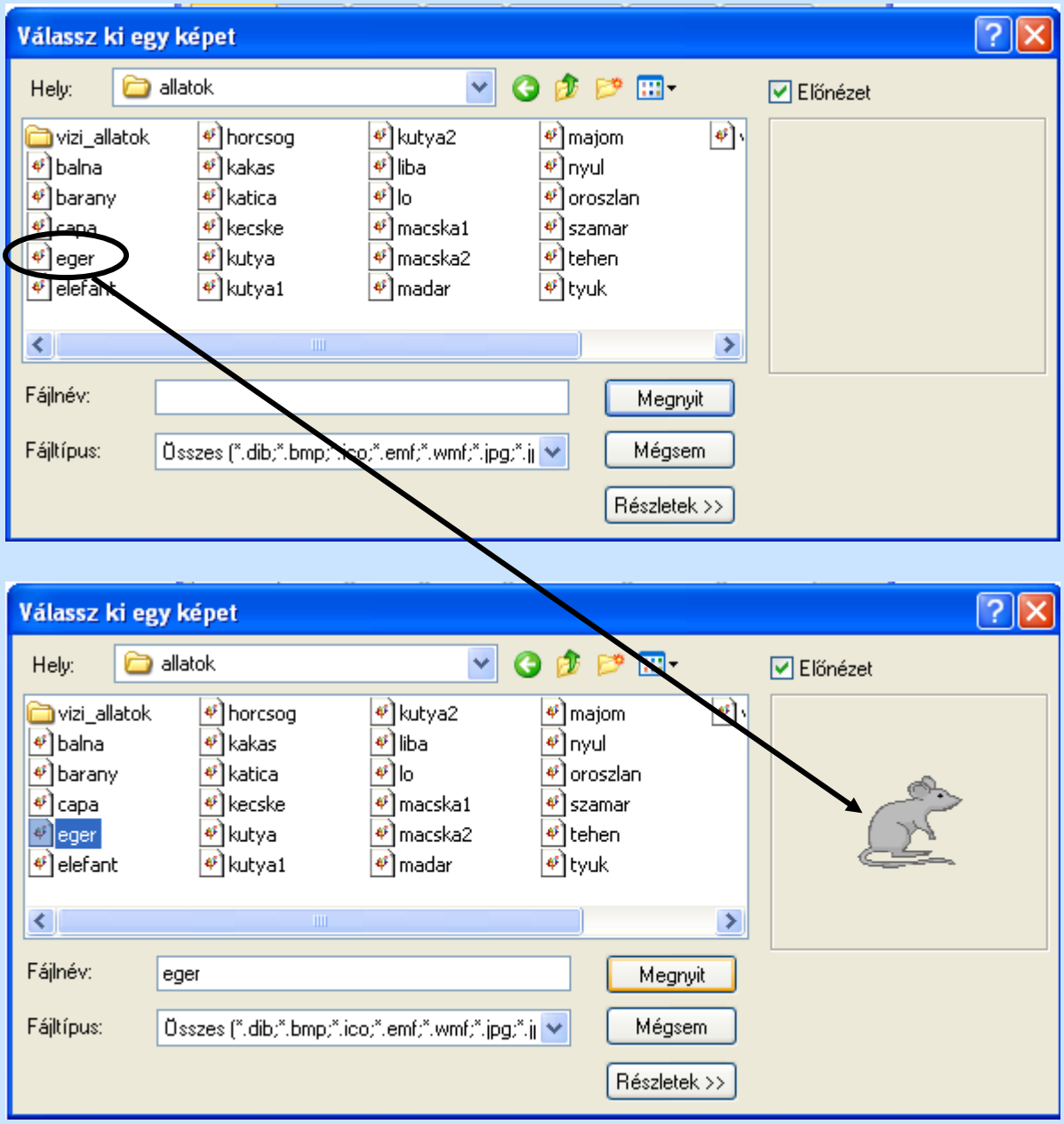
```
alak! []
```

A parancsot üres listabemenettel használjuk, akkor visszakapjuk az alapalakot.

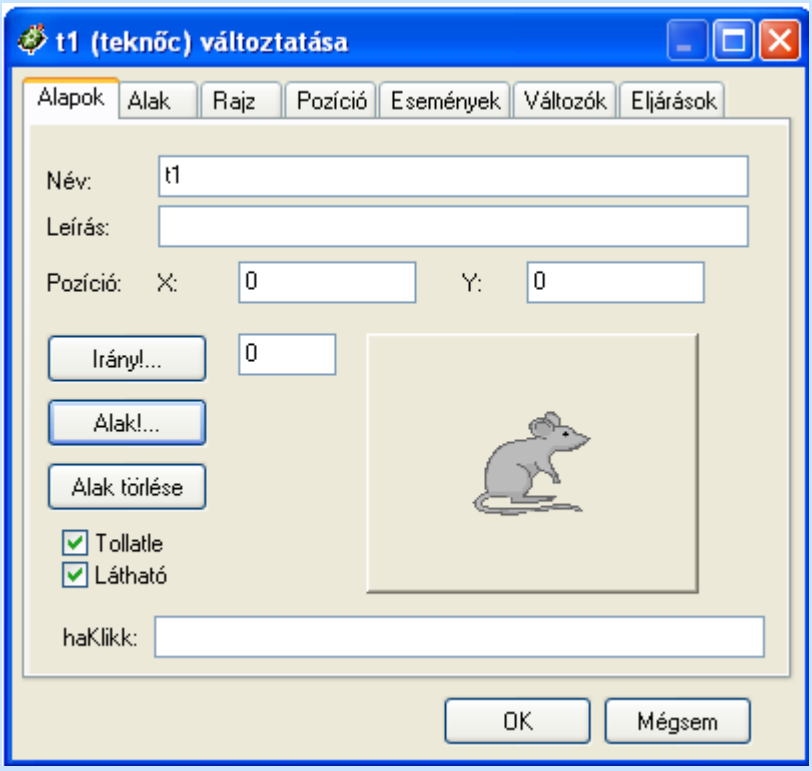
Feladat

1. Változtasd meg paranccsal a teknős alakját, az új alak legyen egy egér!

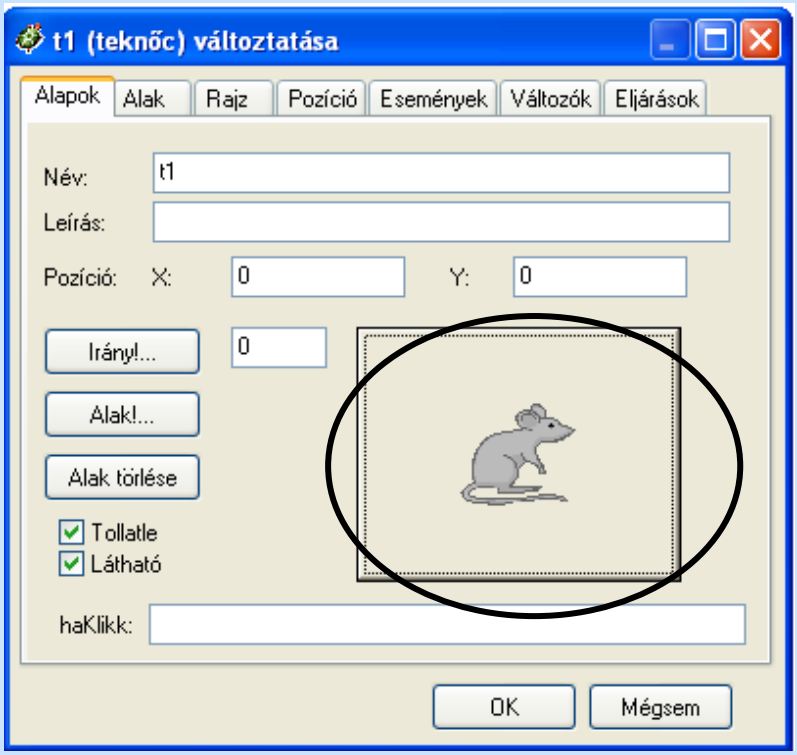




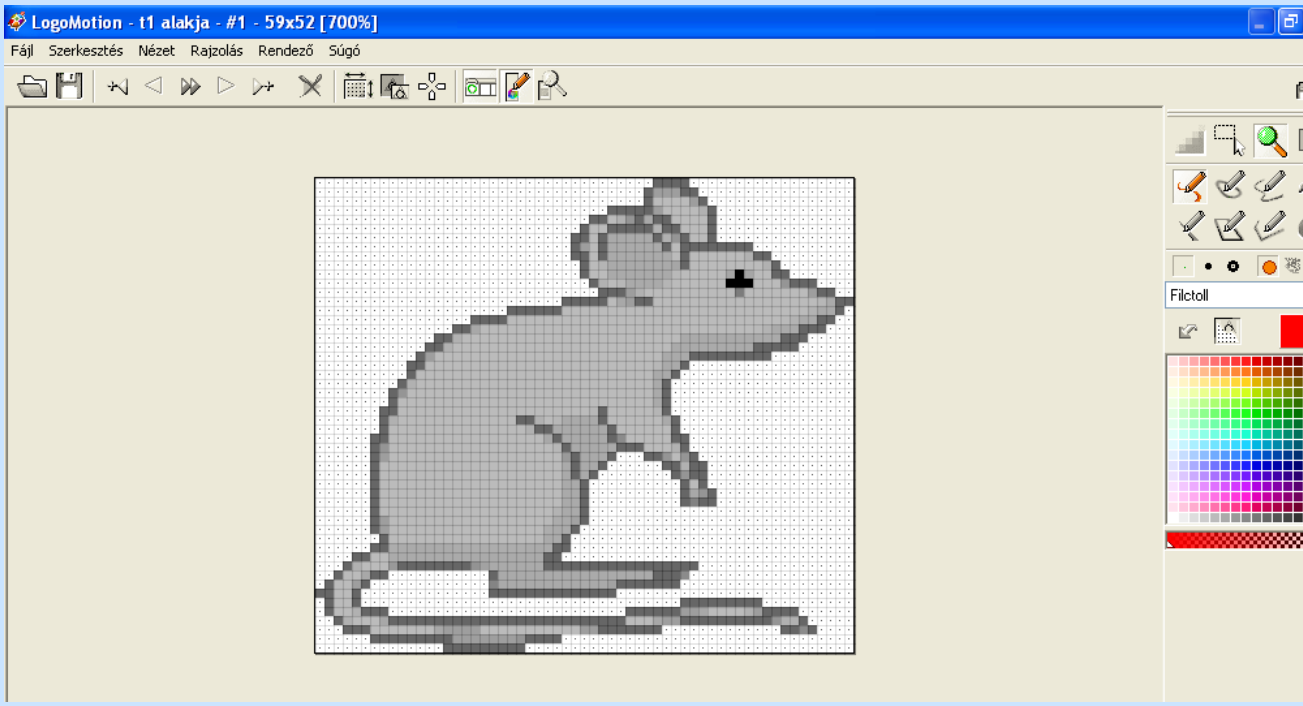
Elkészült az új alak:



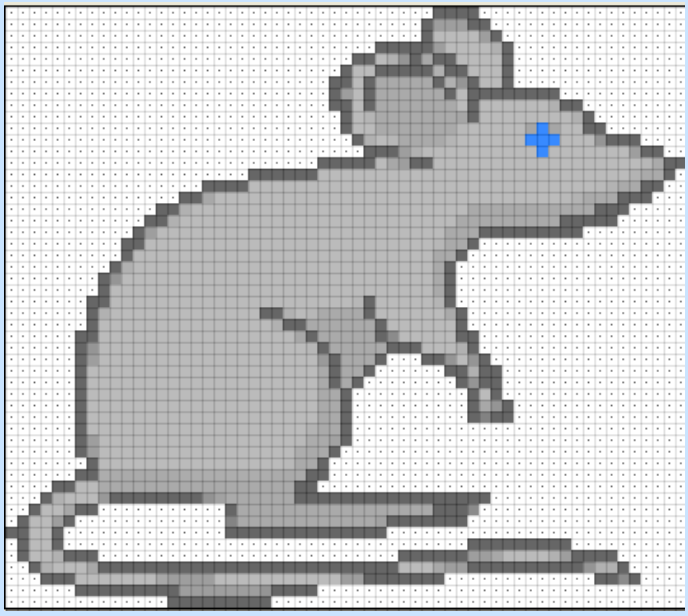
2. Az előzőekben beállított teknőc alak (jelenleg egér) szeme színe legyen kék! Kattintsunk az előnézetre:



Megnyílik a LogoMotion szerkesztő:

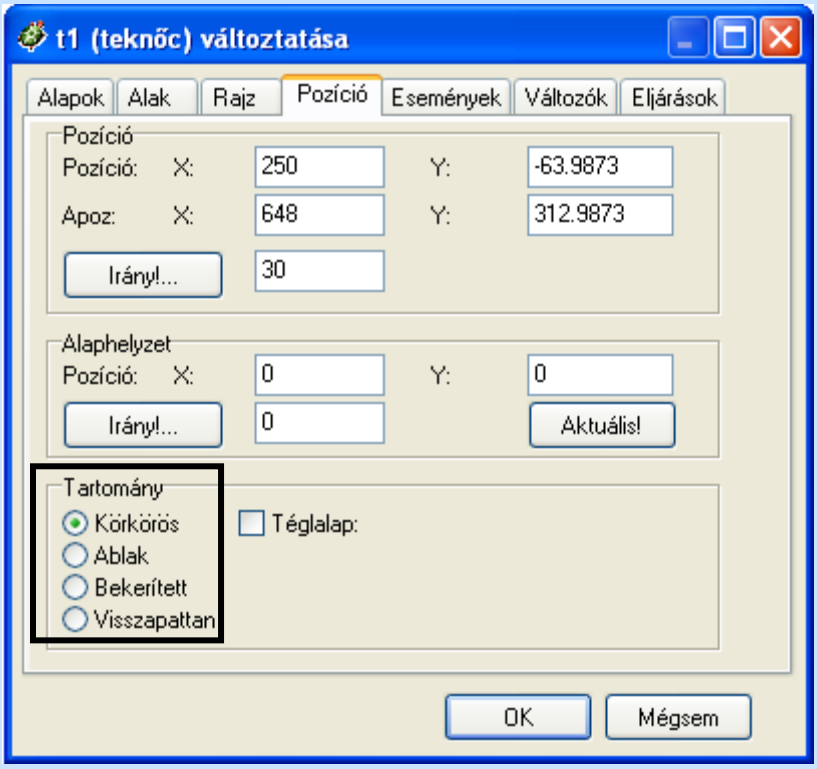


Válasszuk ki a kék színt és színezzük át az alak szemét:



Tartomány és tartománystílus

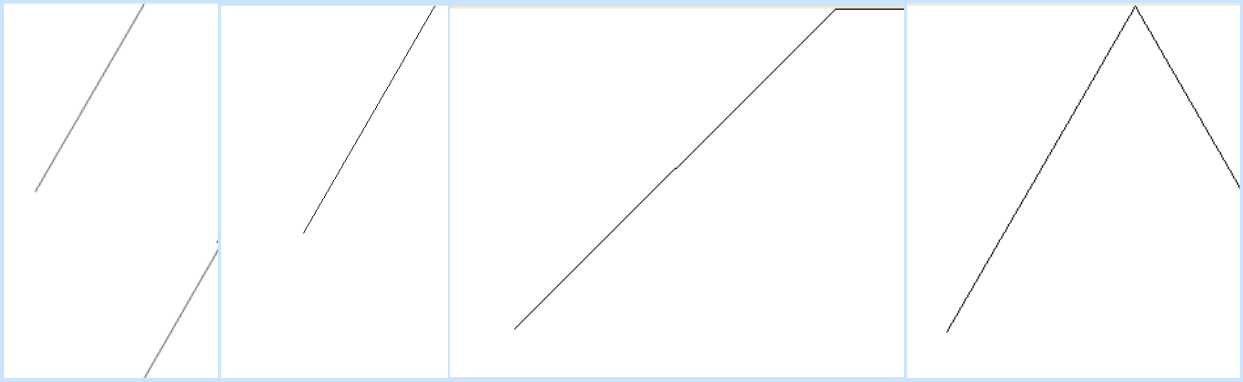
A lap vagy panel, melyen a teknősünk mozog egy téglalap alakú terület, ha a mozgás során eléri területének határát, beállításainak megfelelően viselkedik:



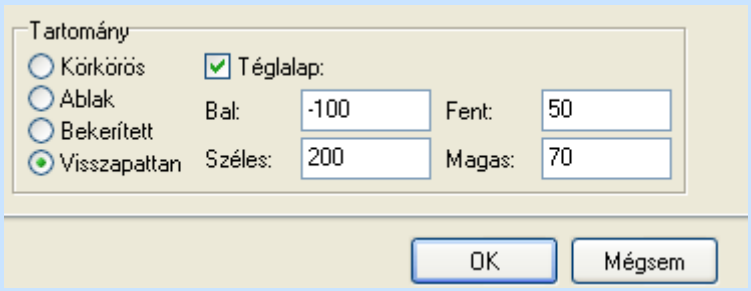
A tartománystílus azt mondja meg a teknősnek, hogy hogyan viselkedjen, ha eléri a tartományának határát.

körkörös stílus	a teknőc mindig a tartományán belül marad, mindegy mekkorákat mozog. Ha ilyen teknőc kimozdul a képernyőről, automatikusan “átgördül” és megjelenik a tartomány átellenes oldalán.
ablak stílus	a teknőc kimozdulhat a tartományának határain túlra is, de ott nem rajzol és nem is látszik maga a teknőc.
visszapattan stílus	a teknőc mindig visszapattan a tartományának határától, és mozgását a tartományon belül folytatja.
bekerített stílus	a teknőc nem keresztezi tartományának határát, nem is pattan vissza róla és nem is gördül az átellenes oldalra. A határ valódi palánkként működik a teknőc számára. A teknőc továbbmozog a fal mentén addig, míg el nem érné az elmozdulásának megfelelő pontot, vagy el éri a téglalap csúcsát.

körkörös stílus | ablak stílus | bekerített stílus | visszapattan stílus



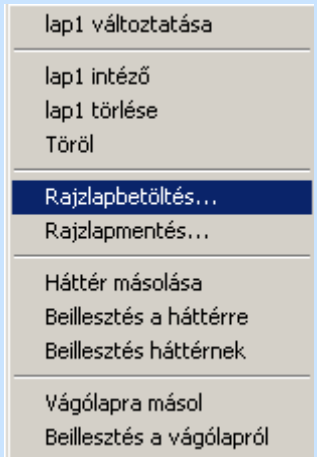
Alapértelmezésben a tartomány a lap vagy a panel. Bizonyos esetekben szükség lehet arra, hogy ezt a területet mi határozzuk meg:

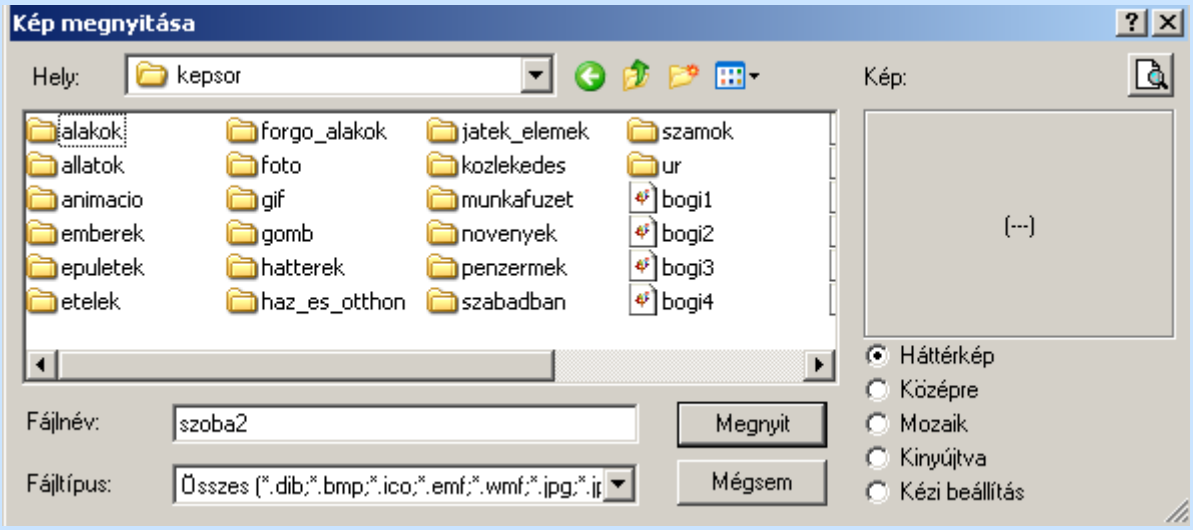


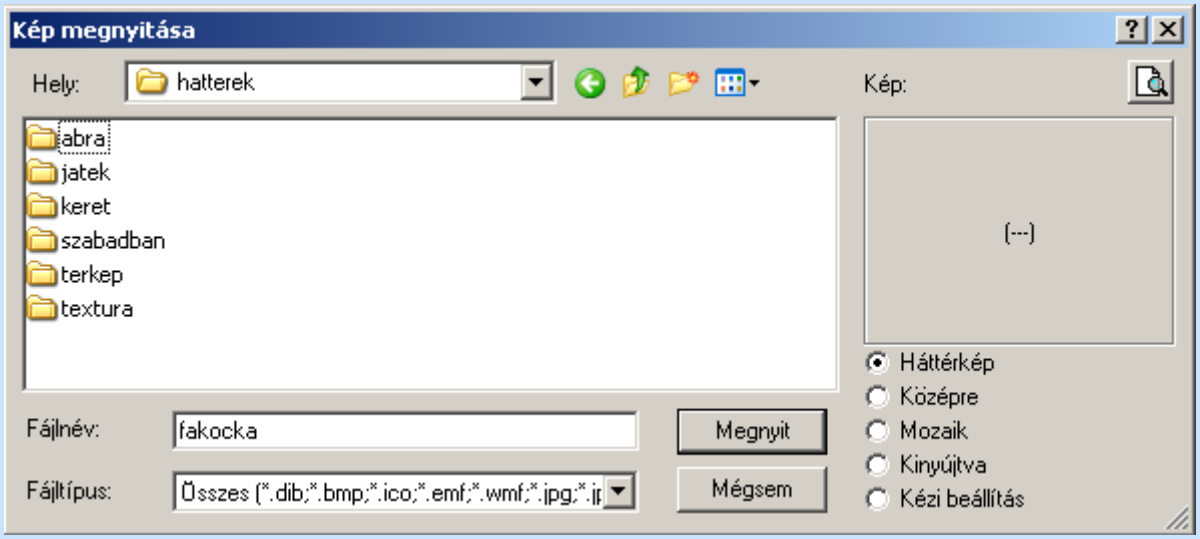
Feladat

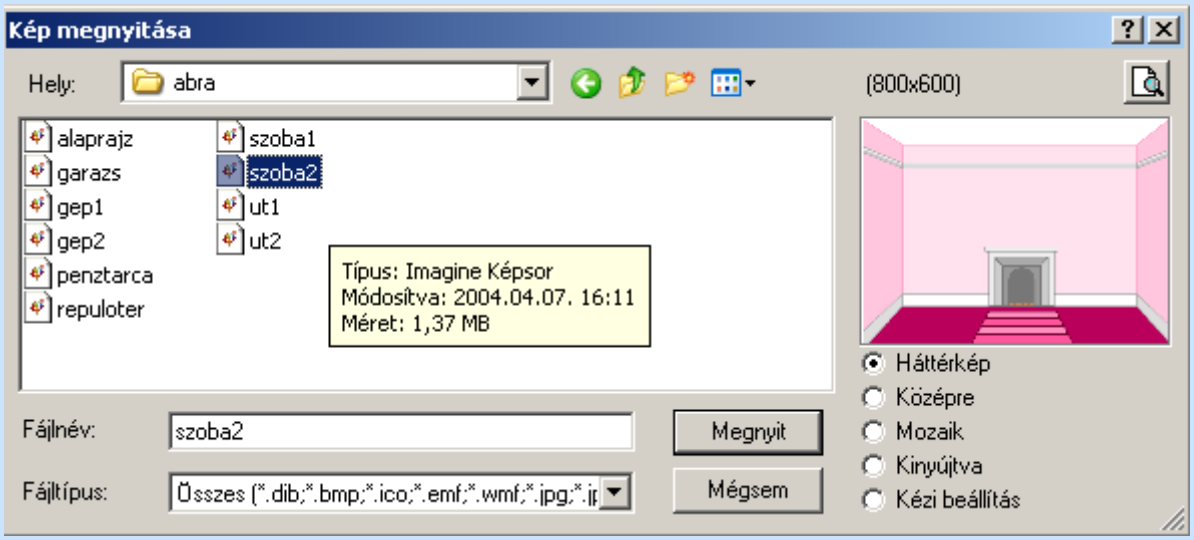
1. Készítsünk el egy kandallós szobabelsőt, melyben a kandalló két oldalán szobanövény is látható, a kandalló tetején legyen egy televízió. A kandallóba „beszorult egy kisegér, mely nem talál ki a kandallóból. Az egér csak akkor látható, ha a kandalló nyitott ajtaja előtt halad, de a kandalló égéstere természetesen szélesebb, mint az ajtaja, de ott mi már nem látjuk.

Töltsünk be egy alap hátteret:

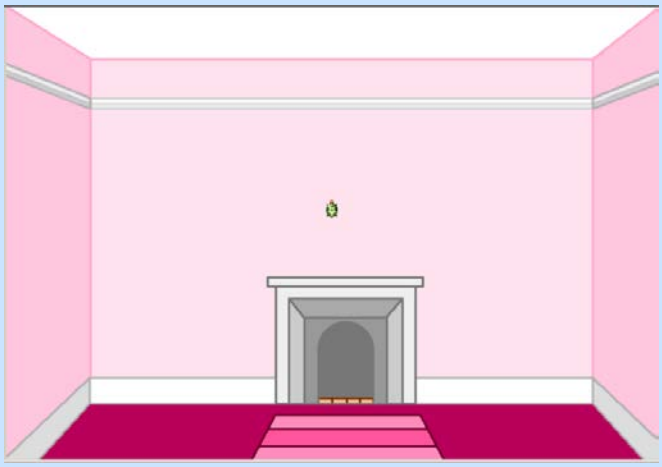




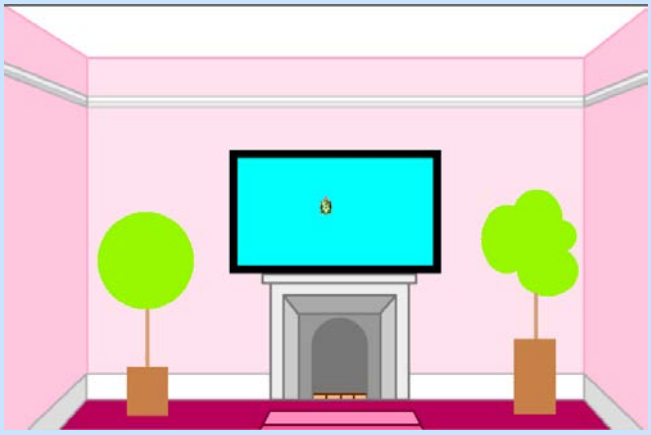




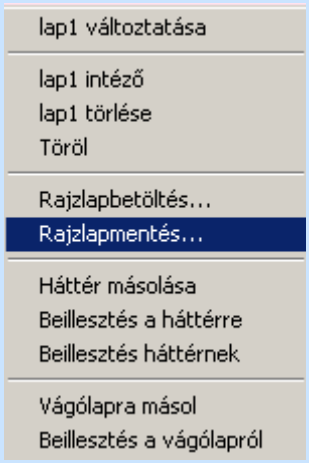
Ezzel ez lesz a háttérképünk.



Az eddigiekben megtanult rajzoló eszközökkel egészítsük ki a rajzunkat a feltételnek megfelelően.

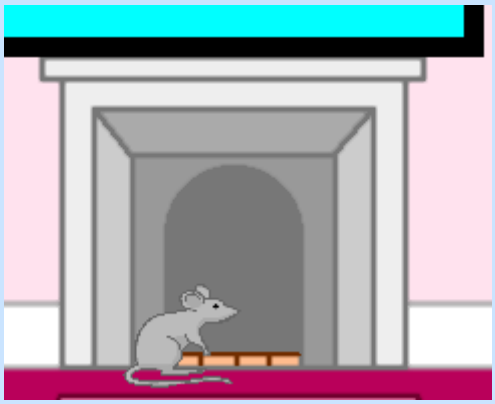


Majd az elkészült képet mentsük el:



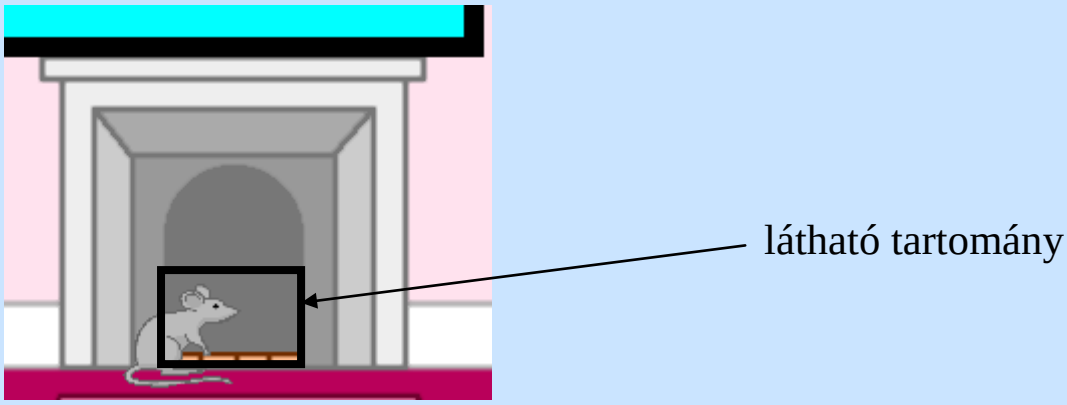
A hely alapértelmezésben a **kepsor** mappába kerül.

A teknős alakját változtassuk meg egér formájúra, s mozgassuk a kandalló elé.



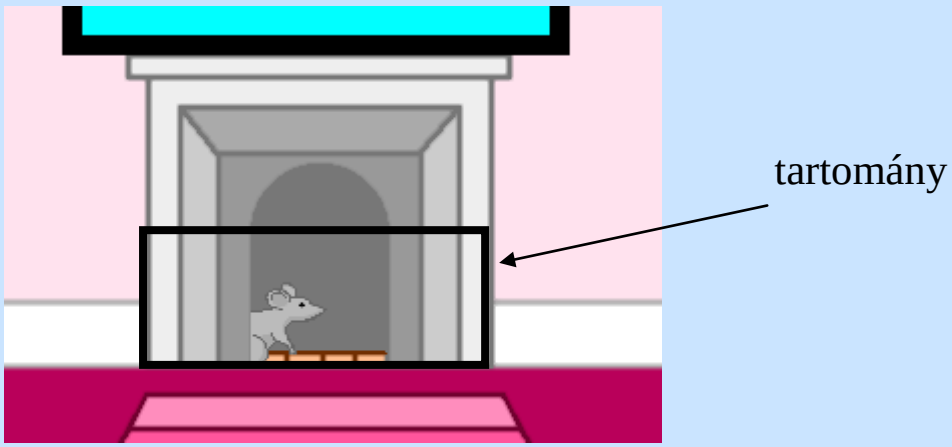
A látható tartomány kijelölését tegyük meg a megismert paranccsal:

láthtart! [-17 -184 69 53]

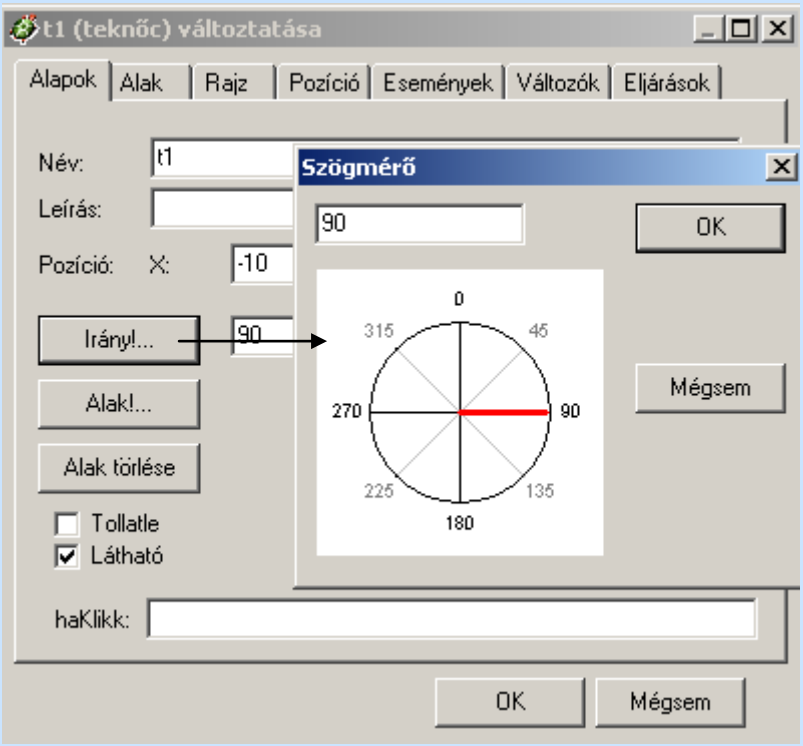


Ekkor még a lap teljes területén tudna mozogni az egér, csak nem látnánk.

Ezt a területet szűkíthetjük a **tartomány!**paranccsal, melynek működése a **láthatótartomány!**használatával azonos.



A teknős ne húzzon vonalat, s mozgásakor jobbra induljon.



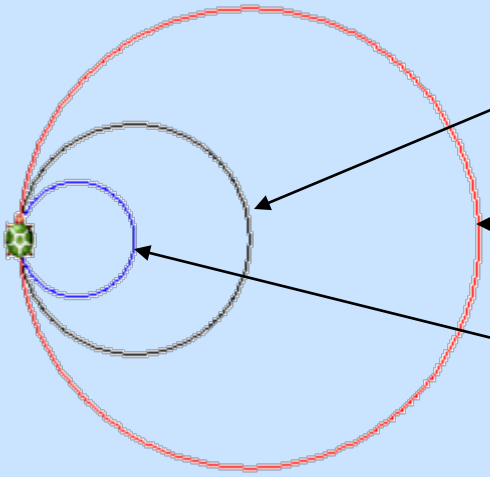
Az**`ism 50 [e 5 várj 80]`** parancs hatására az egér néhányszor elmegy a kandalló ajtaja előtt. Ez a mozgás a LogoMotion animációi segítségével finomítható. Ekkor az is megoldható lesz, hogy a tartomány széléről az egér „visszapattan” tartománystílus alkalmazásával a másik irányba induljon el.

Körök görbék I.

A teknősök a kezdőpozícióban vannak (ebben az esetben nem alkalmazzuk a nagy teknőctételt).

ism 360 [e 1 j 1] ism 90 [e 1 j 1] ism 180 [e 1 j 1] ism 90 [e 1 b 1] ism 270 [e 1 b 1]

Kör kicsinyítése és nagyítása:



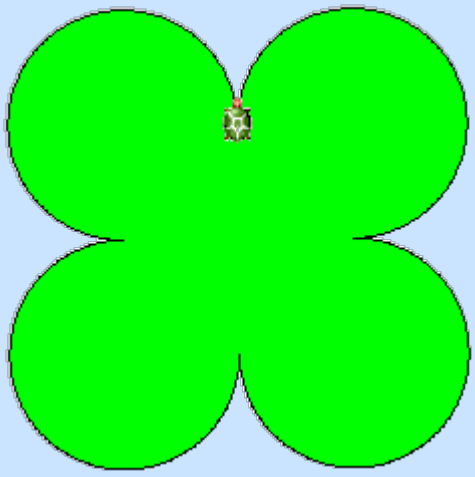
```
ism 360 [e
1 j 1]

tsz!
"piros
ism 360 [e
2 j 1]

tsz! "kék
ism 360 [e
1 j 2]
```

Nagyításnál a lépések nagyságát növeljük.

Kicsinyítésnél a fordulat nagyságát növeljük.

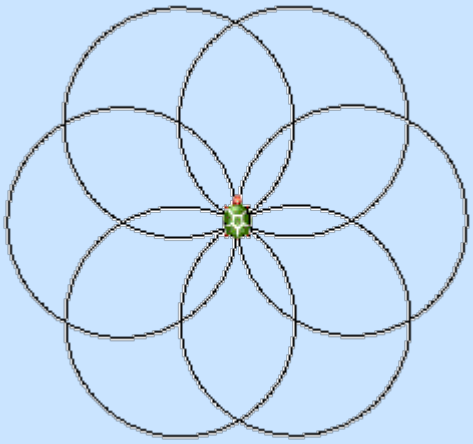
Készítsük el a következő ábrákat:

1.



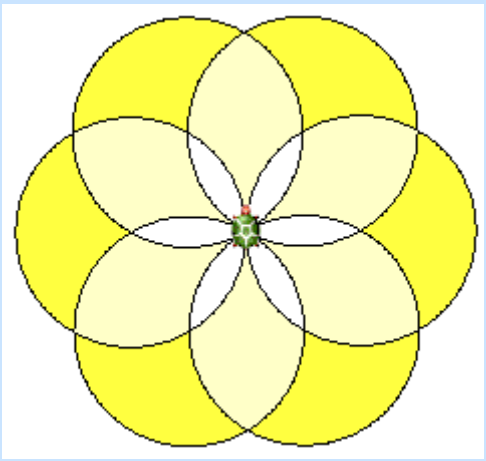
```
eljárás lóhere
tsz! "fekete
tlsz! "zöld
ism 4 [ism 270 [e 1 j 1] j 180]
tf h 30 tölt e 30 tl
vége
```

2.



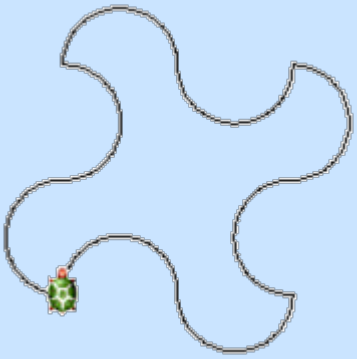
```
eljárás margaréta
ism 6 [ism 360 [e 1 j 1] j 60]
vége
```

3.



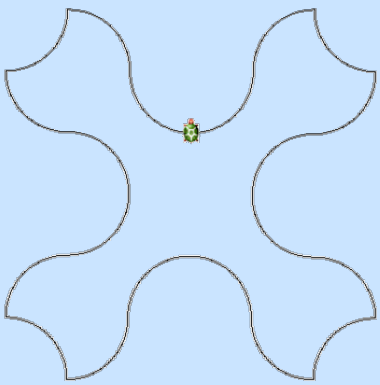
```
eljárás sz_margaréta
tl
tsz! "fekete
ism 6 [ism 360 [e 1 j 1] j 60]
tf
tlsz! "sárga11
ism 6 [e 60 tölt h 60 j 60]
tl
j 30
tf
tlsz! "sárga7
ism 6 [e 100 tölt h 100 j 60]
tl
b 30
vége
```

4.



```
ism 4 [ism 90 [e 1 j 2]
ism 90 [e 1 b 2]
b 90]
```

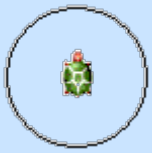
5.



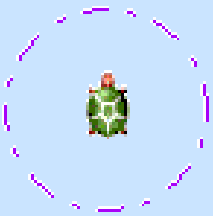
```
eljárás 4körív
j 90
ism 4 [
ism 90 [e 1 b 1]
ism 90 [e 1 j 1]
j 90
ism 90 [e 1 b 1]
j 90
ism 90 [e 1 j 1]
ism 90 [e 1 b 1]]
b 90
vége
```

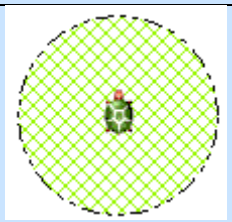
Körök, görbék II.

A **kör** a Logo beépített utasítása, segítségével adott átmérőjű kört tudunk rajzolni:


A teknős a kör középpontjában marad.

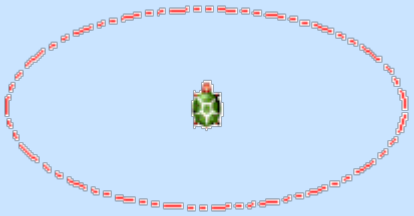
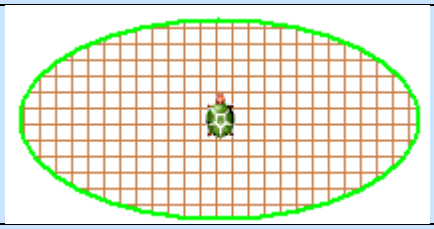
A **töltöttkör** utasítás hatása ugyanez, csak a kitöltésnél megtanultak alapján beállíthatjuk a toll színét, vastagságát, mintáját, valamint a töltőszínt és a töltőmintát is.

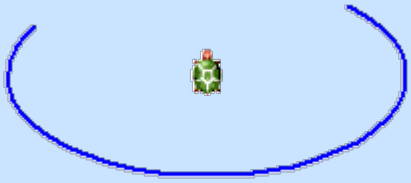
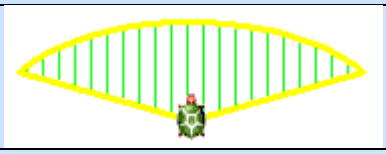
	`tsz! "lila`
	`tv! 1`
	`tm! 3`
	`kör 70`

	`tsz! "fekete`
	`tm! 4`
	`tv! 1`
	`tlsz! "zöldesbarna`
	`tlm! 3`
	`töltöttkör 100`

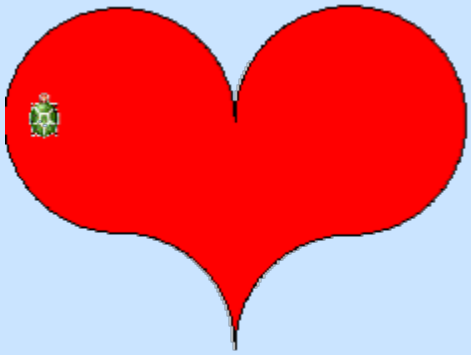
Az **ellipszis** utasítás hatására a teknőc ellipszist rajzol, melynek iránya teknős aktuális irányával megegyező.

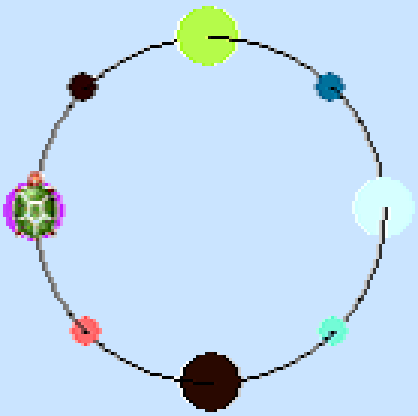
ellipszis [*nagytengely kistengely*] vagy **ellipszis** [*nagytengely kistengely szögtől szögig*] az utasítások **töltöttellipszis** esetén is ugyanígy használhatóak.

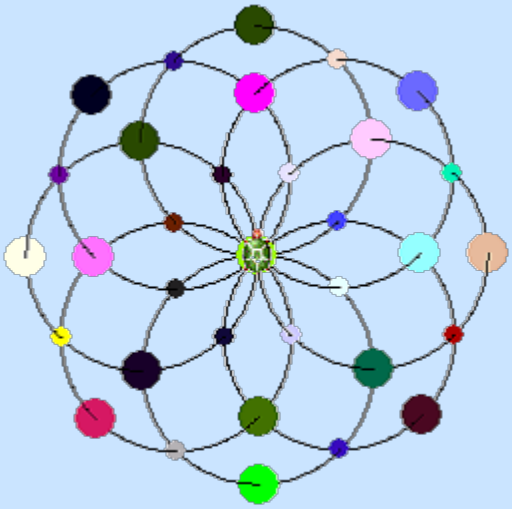
	`tsz! "piros tm! 4 ellipszis [200 100]`
	`tsz! "zöld tm! 3 tv! 2 tlm! 2 tlsz! "szürkésbarna töltöttellipszis [200 100]`

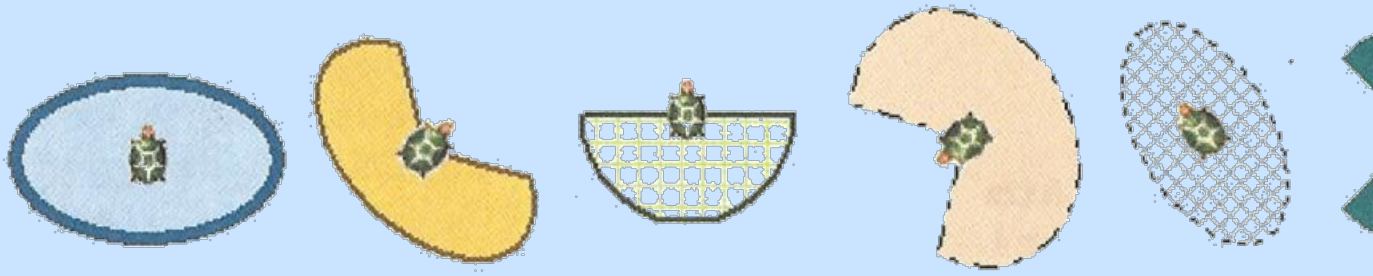
	`? tsz! "kék tm! 5 tv! 2 ellipszis [200 100 45 300]`
	`tsz! "sárga tm! 2 tv! 3 tlm! 6 tlsz! "zöld töltöttellipszis [200 100 300 60]`

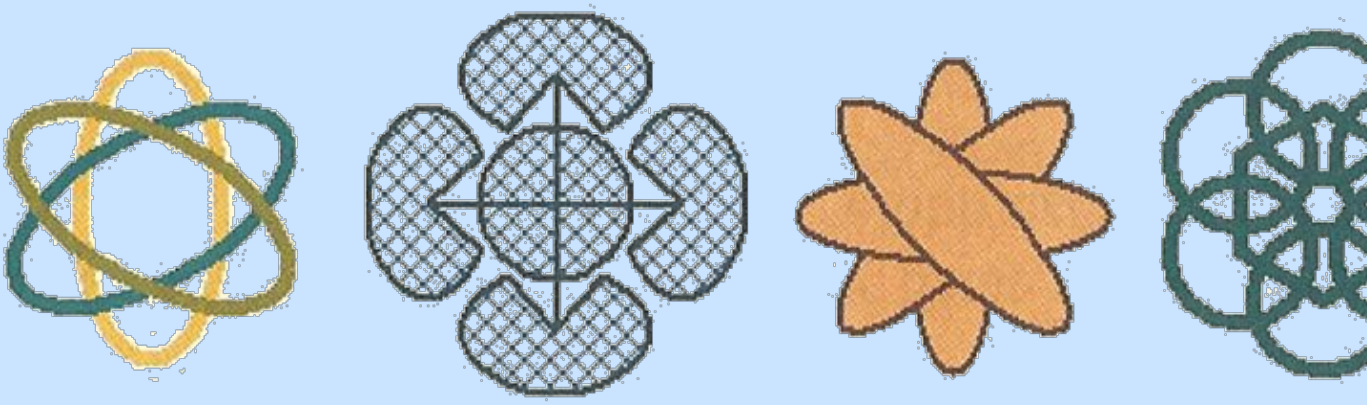
Gyakorlás









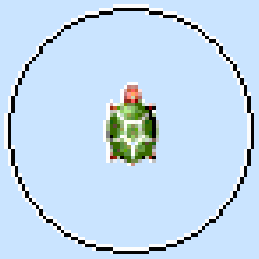


Sugaraskör

Az eddig megismert kör rajzolására vonatkozó parancsainknál vagy nem tudtuk figyelembe venni a kör sugarát, vagy megadhattuk a kör átmérőjét, de a teknős ilyenkor a kör középpontjában maradt. Előfordul azonban, hogy úgy kellene egy kört megrajzolnunk, hogy tisztában vagyunk sugarának nagyságával, de a körvonalon szeretnénk haladni vagy végig, vagy annak csak egy szakaszán.

Ismétlés

```
ism 360 [e 1 j 1]   ism 90 [e 1 j 1]   ism 180 [e 1 j 1]   ism 90 [e 1 b 1]   ism 270 [e 1 b 1]
```



```
kör 50
```

Ebben a feladatban az ív rajzolása jelenti a problémát:

A teknősnek egy lépéssel a kör kerületének 360-ad részét kell megtennie.
Egységsugarú körben az 1 fokhoz tartozó körívhossz: $\pi/180$.
Ennek felhasználásával el tudjuk készíteni a megfelelő eljárást:

```
eljárás sugaras :r
ism 360 [e :r *3.14159/180 j 1]
vége
```

ahol r a sugár.

Ha nem szeretnénk egy teljes kört megrajzolni, akkor alkalmazhatjuk az:

```
eljárás sugaras_ív :r :f
ism :f [e :r *3.14159/180 j 1]
tf
ism 360- :f [e :r *3.14159/180 j 1]
tl
vége
```

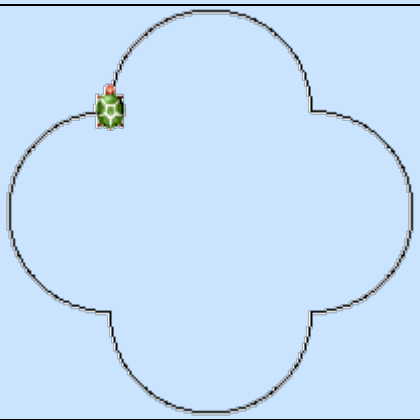
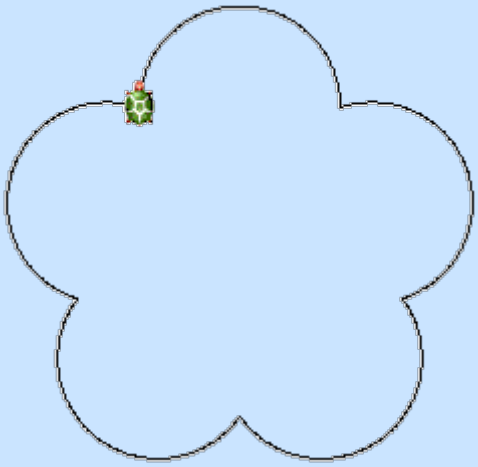
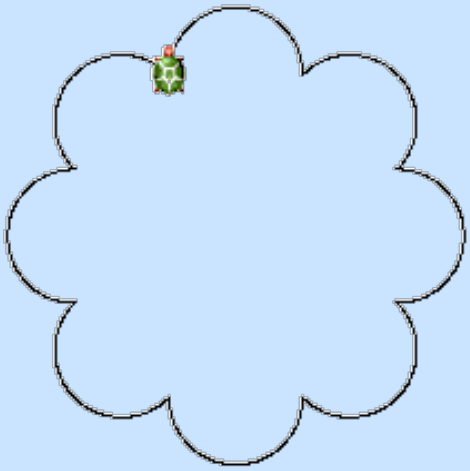
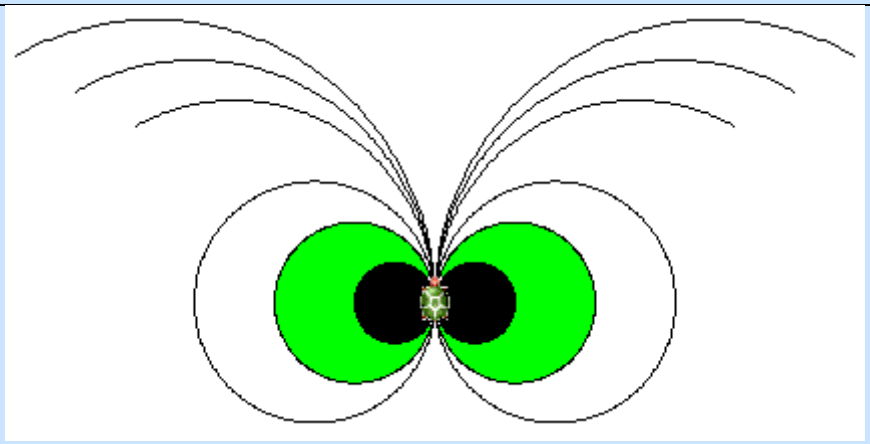
ahol r a sugár, f pedig annak a paramétere, hogy milyen szögnek megfelelő ívet rajzoljunk.

Az előző eljárások felhasználásával készíthetünk olyan eljárásokat, melyeknél a teknőc ugyan a kör középpontjában fog tartózkodni, de a körvonalon rajzolja meg a kört. (A kör eljárásnál csak a kör középpontjából irányítja a folyamatot.)

```
eljárás középpontos :r
tf e :r j 90 tl
ism 360 [e :r *3.14159/180 j 1]
tf b 90 h :r tl
vége
```

```
eljárás középpontos_ív :r :f
tf e :r j 90 tl
ism :f [e :r *3.14159/180 j 1]
tf
ism 360- :f [e :r *3.14159/180 j 1]
tl
tf b 90 h :r tl
vége
```

Feladat

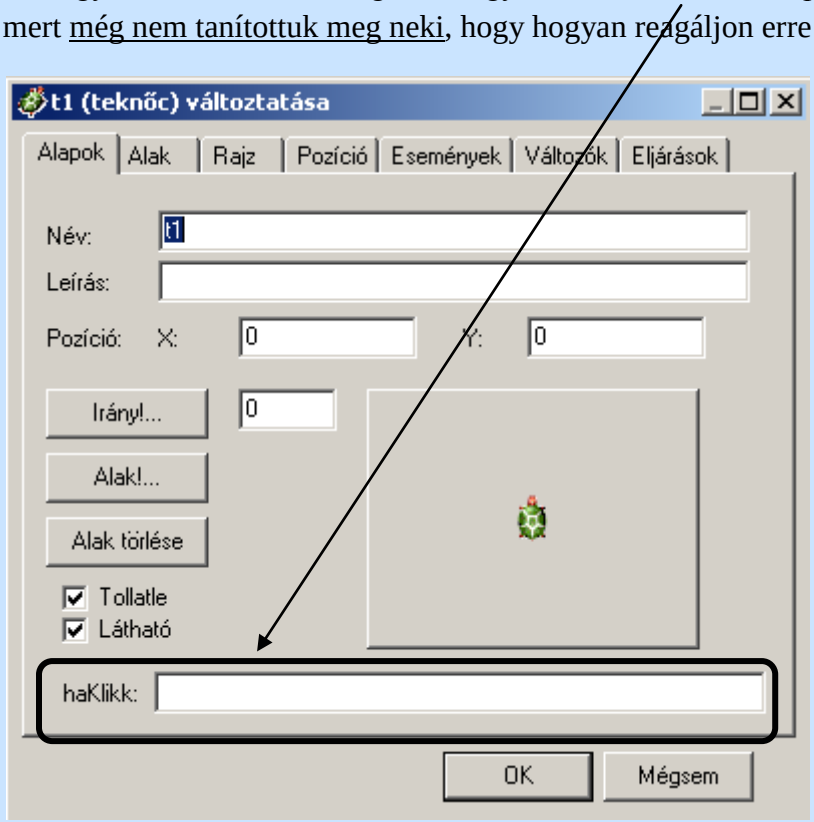
	eljárás lóhere :r tlism 4 [sugaras_ív :r 180 tf j 90 e :r*2 tl] vége
	eljárás lóhere5 :r tlism 5 [sugaras_ív :r 180 tf j 90 e :r*2 b 18 tl] vége
	eljárás lóhere8 :r tlism 8 [sugaras_ív :r 180 tf j 90 e :r*2 b 45 tl] vége
	eljárás bagoly :r törölkép tl szem :r szinez :r sugaras_ív :r*5 120 sugaras_ív :r*6 120 sugaras_ív :r*7 120 sugaras_ív_b

	:r*5 120 sugaras_ív_b :r*6 120 sugaras_ív_b :r*7 120 vége

Különleges események a teknős életében

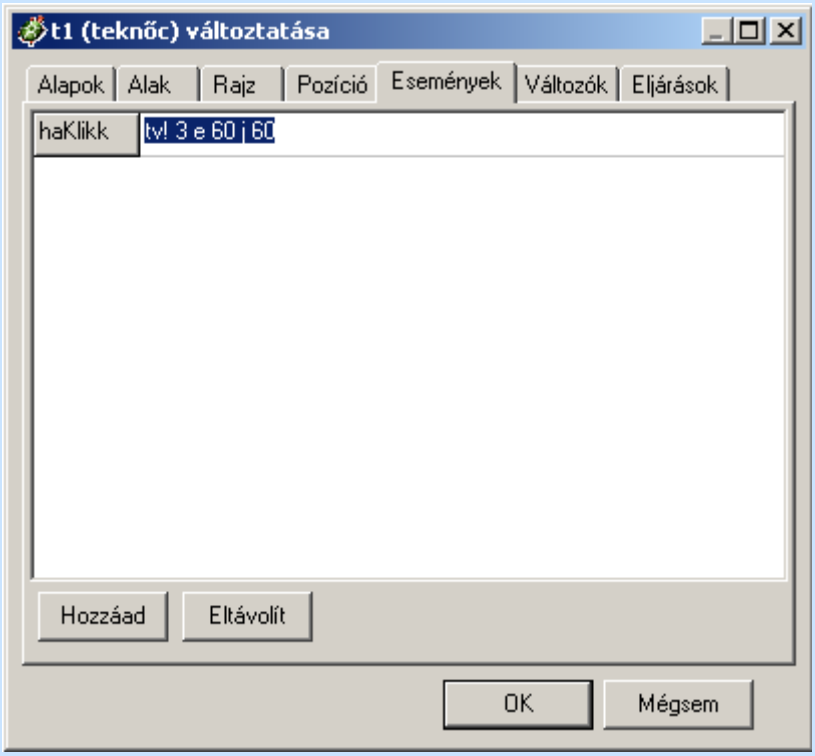
`haKlikk`

Az eddigiekben kattintottunk már a teknősünkön jobb egér gombbal, de még nem próbáltuk ki, hogy mi tesz, ha ballal végezzük ugyanezt a műveletet. Alapként nem történik semmi, mert még nem tanítottuk meg neki, hogy hogyan reagáljon erre a helyzetre.

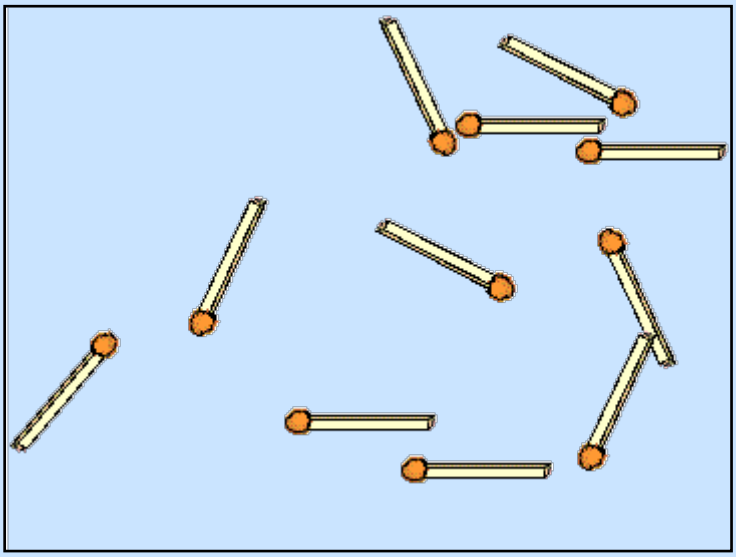


Ha ide beírjuk pl.: a **`tv! 3 e 60 j 60`**, akkor a teknősre való kattintással hatszöget is el tudunk készíteni:

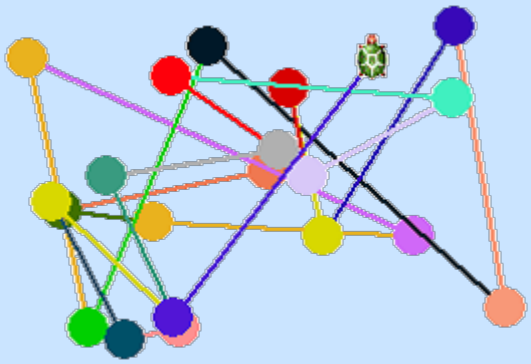
Erre a helyre beírt utasításunk megjelenik az Események között is:



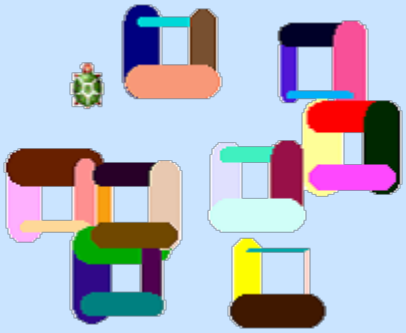
A gyufa alak kiválasztásával és a**`e 50 + tetsz irány! tetsz lenyomat`**utasítás haKlikk-ben való elhelyezésével juthatunk az alábbi ábrához:



Gyakorlás



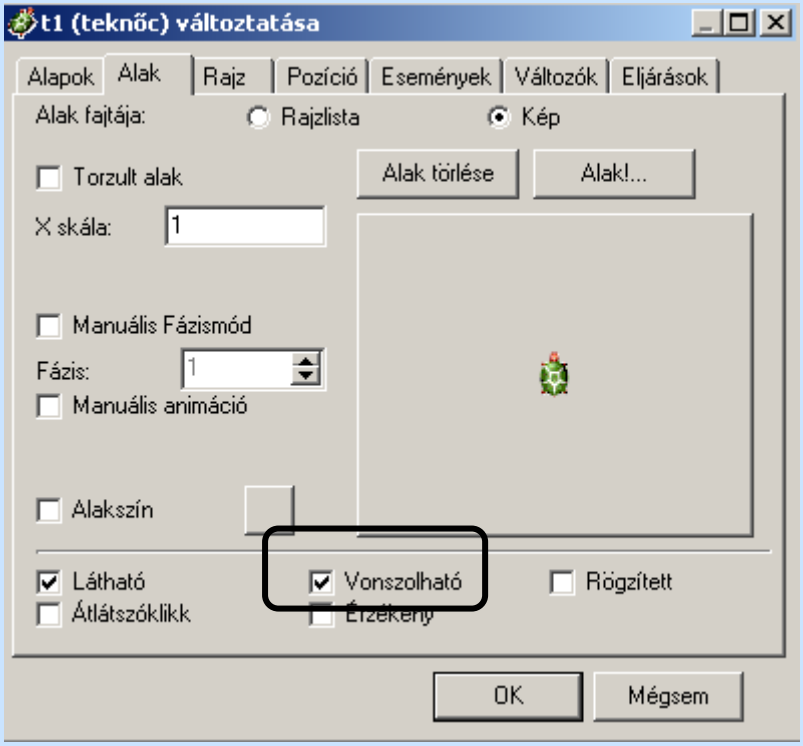
Teknőcre való kattintásra véletlenszerűen beállítja a toll színét, és rajzol egy **20**-as méretű pontot, majd leeresztett tollal tetszőleges helyre ugrik.



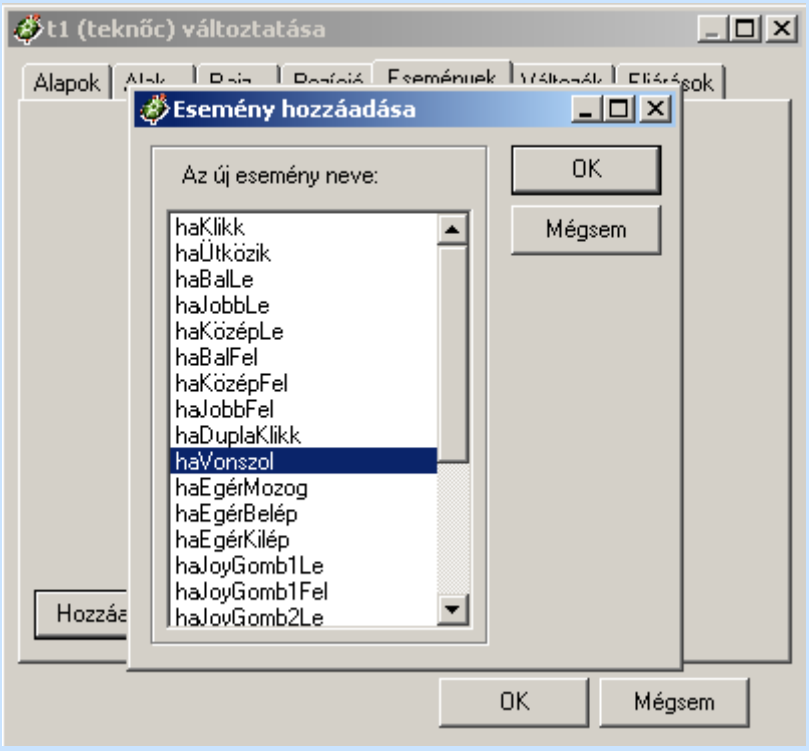
Teknőc rajzol egy 30 oldalhosszúságú négyzetet, amelynek minden oldala véletlenszerűen megválasztott színű és vastagságú, majd tetszőleges helyre ugrik.

haVonszol

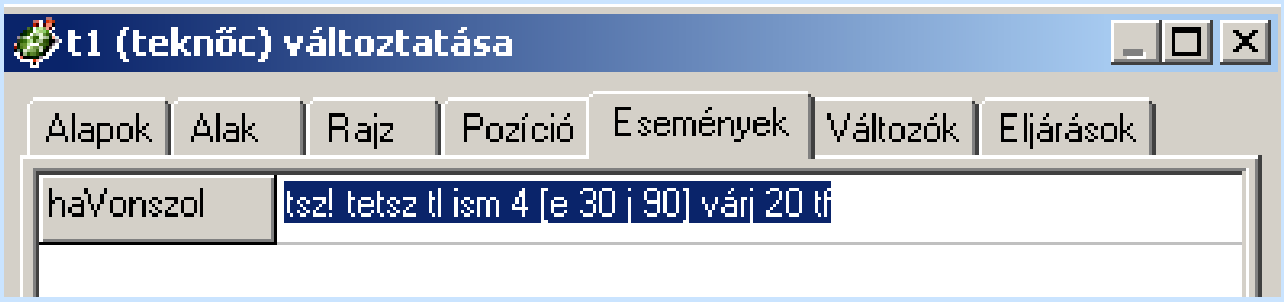
Érdekes rajzokat készíthetünk a teknőc vonszolásával, ehhez azonban a teknőcöt vonszolhatóvá kell tenni:



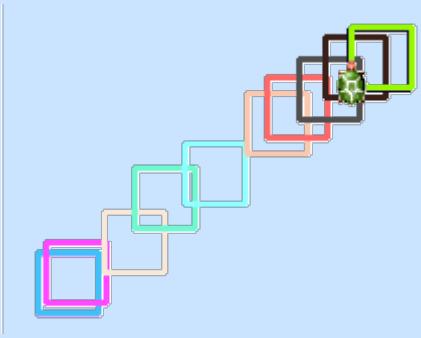
Az Események közül hozzáadjuk a haVonszol-t:



Beírjuk a végrehajtandó utasítást:



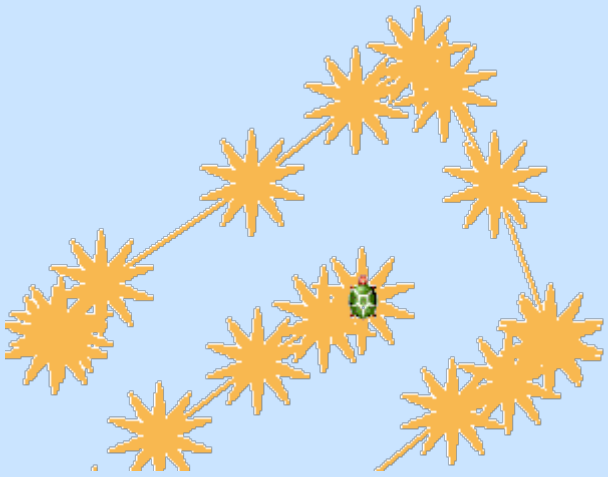
Eredmény:



Ha az Alak vonszolhatóságát beállítjuk, az Alapok oldalon pedig a Tollatle be van kapcsolva, akkor egyszerű rajzeszközként is használható.

Gyakorlás

A feladathoz az éjszakai égbolt csillag eljárását használhatjuk.

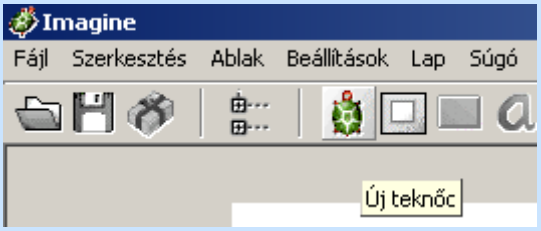


Több teknőc

A teknőc az Imagine legalapvetőbb objektuma. A projektek érdekesebbé tehetők, ha egyetlen teknőc helyett sokat használnak. Több teknőc párhuzamosan rajzolhat, párhuzamosan mozoghat, és segíthetnek komplex képek létrehozásában a képernyőn. Az Imagine-ben a teknőcök számának csak a felhasznált hardver szabhat korlátot.

Amikor az Imagine elindul, csak egy teknőc van: t1 (ha nem változtatják meg az Imagine alapértelmezett szabályát), mely lap1-en él. További teknőcöket az alábbi módszerek bármelyikével létre lehet hozni:

- Rákattintva az Ikonsor Új teknőc gombjára, majd a lapra vagy panelre



- Az új "teknőc [… beállítások …] utasítást használva
- Az újobjektum "Teknőc

Minden teknőc egy lapon vagy egy panelen él. Ezt hívják a teknőc helyének. A teknőc egész életére arra van korlátozva, hogy a helyén éljen (ott rajzolhat, mozoghat, írhat ki szövegeket stb.). A teknőcöknek egyedi nevük van az egész lapon – még akkor is, ha egy panelen élnek, amely maga a lapon él, vagy egy másik panelben lévő panelen, amely a lapon van… stb. Ennek a szabálynak köszönhetően a lap valamennyi teknőce közvetlenül megszólítható a nevével.

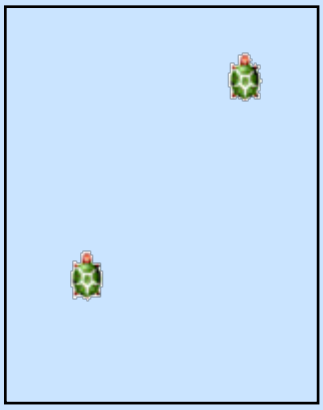
Teknőcöt vagy teknőcöket törölni így lehet:

```
törölobjektum "t22
törölobjektum [t2 t4 t20]
törölobjektum mindenteknőc
```

Egy teknőc minden időpillanatban **aktív** vagy **inaktív**. Az Imagine a teknőceiről két lista segítségével készít feljegyzést:

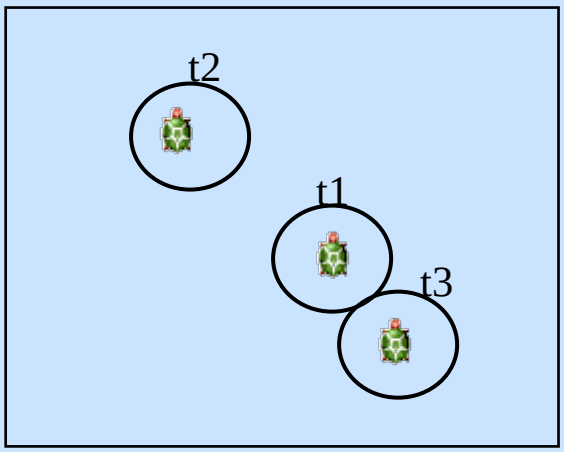
- mindenteknőc – az összes teknőc listája, melyek egy lapon vagy panelen élnek.
- kiaktív – az összes pillanatnyilag aktív teknőc listája.

Helyezzünk el egy új teknőcöt a lapon:

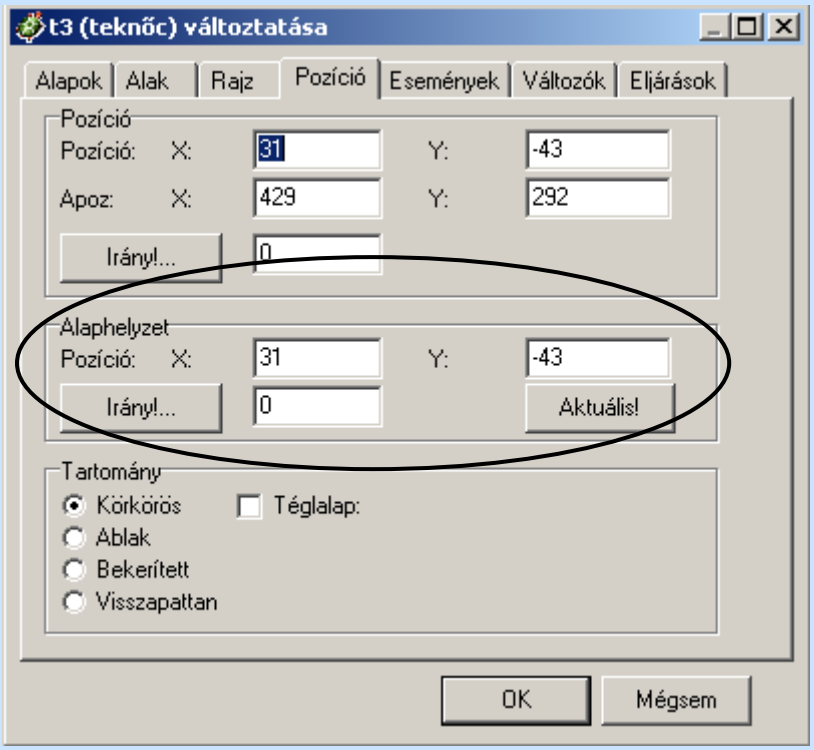
	`? ki kiaktív` `t1`	Az aktív teknőcök nevét írja ki. A kiaktív helyett a kifigyel függvény is használható.
	`? ki mindenteknőc` `t1 t2`	Megjeleníti az összes teknőc nevét. A mindenteknőc helyett a rövidebb mindtek parancs is használható. A teknőcök sorrendje az mindenteknőc listán alapértelmezés szerint megfelel annak a sorrendnek, amelyben a teknőcöket létrehozták. De ez a sorrend át is definiálható.

Feladat

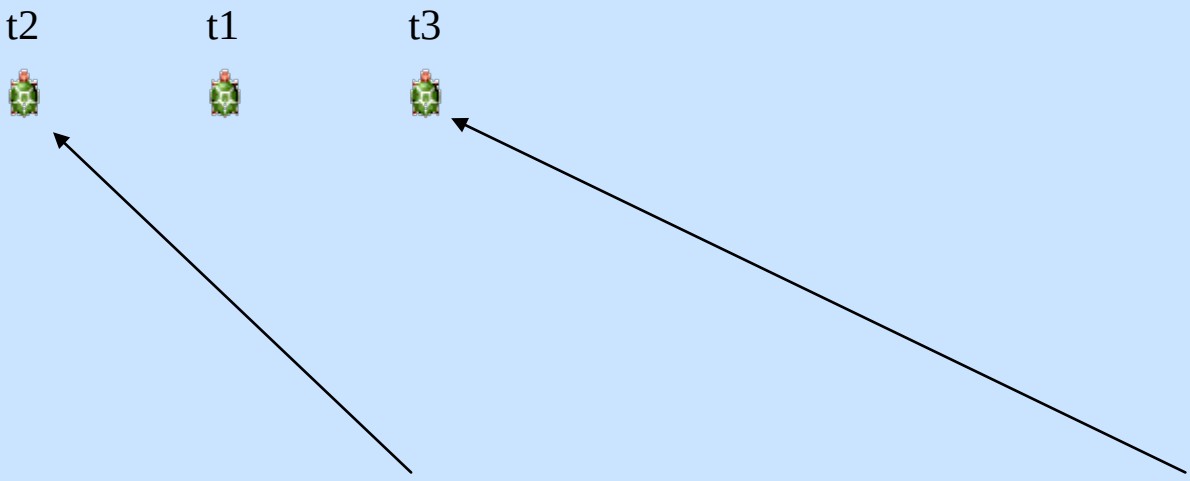
1. Hozzunk létre egy lapon 3 teknőcöt (különböző alakokkal), s állítsuk be, hogy képernyőtörlés után az általunk meghatározott pozícióban helyezkedjenek el.

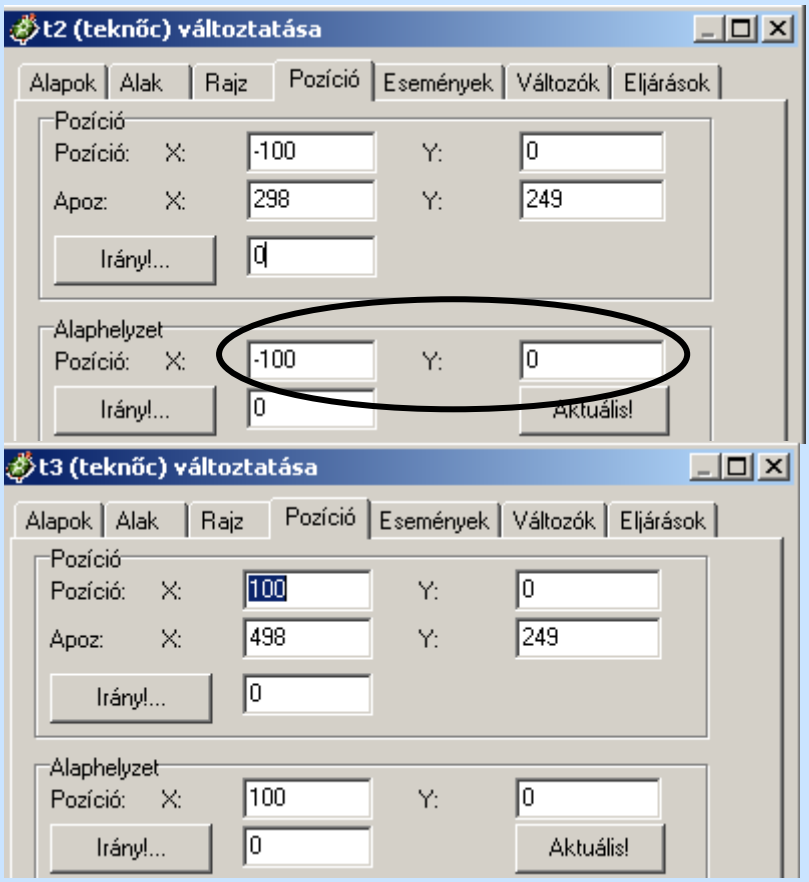


A teknősök helyi menüjébe állítsuk be az alakot, illetve a Pozíció fülön az alaphelyzetet:

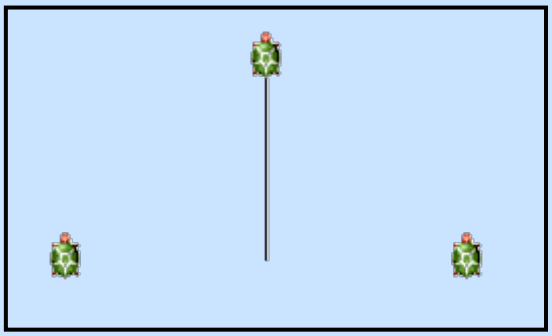


A teknőcök legyenek egy vonalban, egymástól 100 egység távolságra, a t1 teknőc pozícióját ne változtassuk meg:

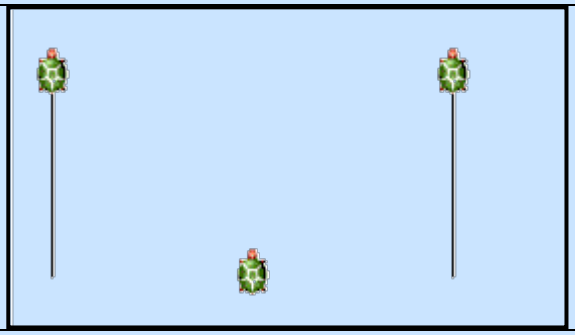




2. Egy utasítás kiadásával mozdítsuk előre a teknőcöket 100 lépéssel:
   Jelen pillanatban az e 100 hatására, csak a lapon eredetileg létező t1 teknőc teszi meg az utat.



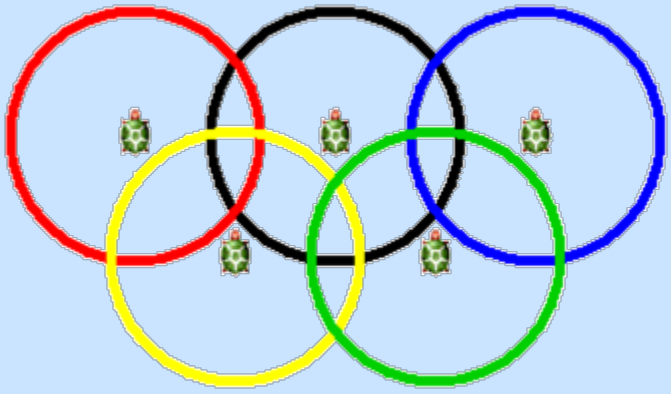
   Ahhoz, hogy minden teknőc teljesítse az utasításunkat aktívvá kell válniuk. Ez megtehető időlegesen vagy hosszabb időre.
   kér*teknőcök* [*utasításlista*]

? ki kifigyel t1 ? kér [t2 t3] [e 100] ? ki kifigyel t1 Látható, hogy az aktív teknőc mivolta nem változott, de a másik 2 teknőc végrehajtotta az utasítást, azaz átmenetileg aktívvá vált.	
? ki kifigyel t1 ? figyelj mindtek ? e 100 ? ki kifigyel t1 t2 t3 Látható, hogy ennek az utasításnak a hatására minden teknőc aktívvá vált. A feladatot párhuzamosan hajtotta végre.	

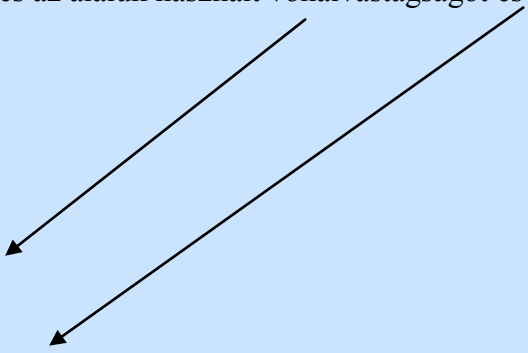
3. Oldjuk meg, hogy a feladatot egymás után hajtsák végre:

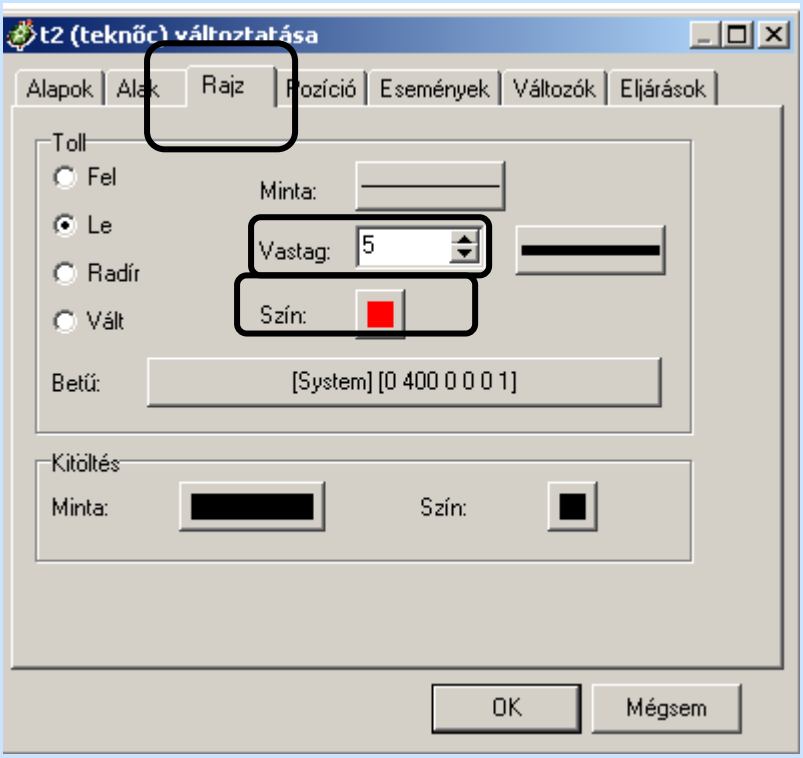
Az erre szolgáló parancs az egyenként [utasításlista], ez azonban jelen esetben olyan gyors, hogy nem vesszük észre ezért lassítanunk kell rajta a várj utasítás segítségével: egyenként [e 100 várj 500] a feladat végrehajtásának sorrendje a figyelj parancsban a teknőc neveinek megfelelőre való változtatásával oldható meg.

4. Készítsünk 5 teknős segítségével 5 különböző színű egymást metsző kört!



Állítsuk be a teknőcök pozícióját, és az álaluk használt vonalvastagságot és színt:





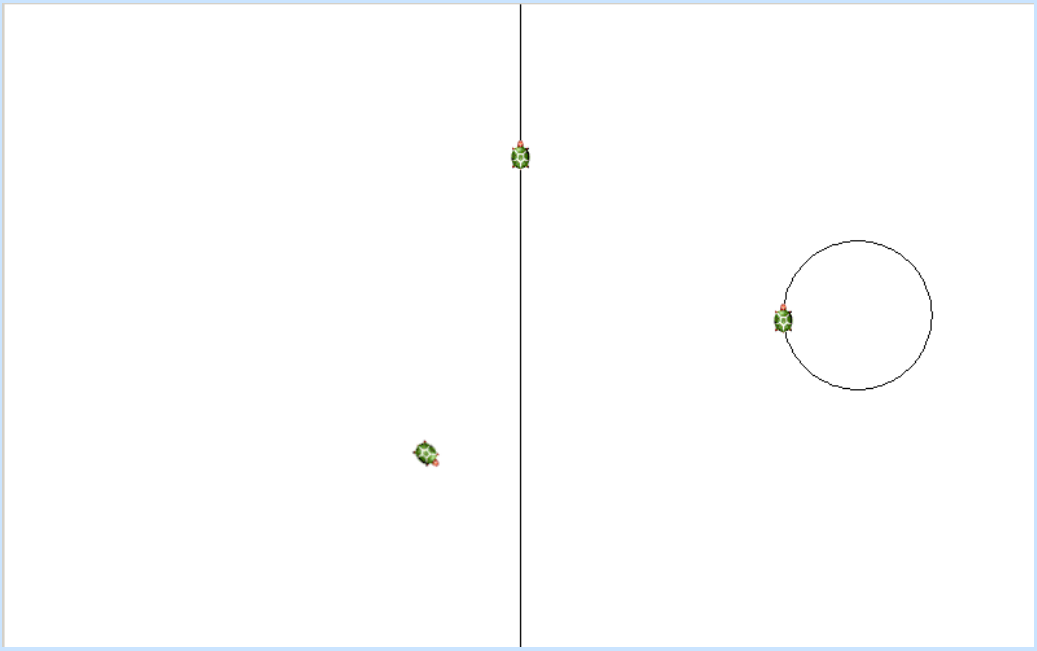
A teknőcök pozíciója:

t1: (-100, 0)
t2: (0, 0)
t3: (100, 0)
t4: (-50, -60)
t5: (50, -60)

A kiadott parancs pedig a **kör 130** volt.

Több teknőcös feladat

1. Helyezzünk el egy lapon 3 teknőcöt, melyek mindegyike a haKlikk-re meghatározott feladatot hajt végre:
   A teknőcök pozíciója tetszőleges, az egyik folyamatosan halad előre, a másik forog körbe saját középpontja körül, a harmadik pedig a kettő kombinációját teszi: kört rajzol!

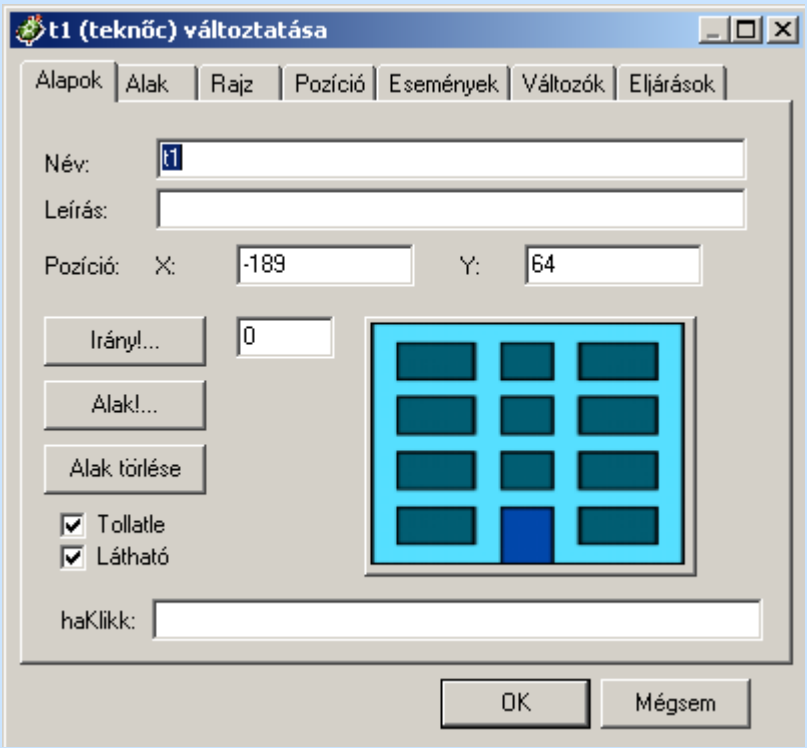


   Segítség:
   A mozgások leírásához használjuk a minden parancsot, melynek használati módja:
   `? minden időzítés_érték [parancsok] vagy` ha a folyamatnak nevet is akarunk adni, akkor
   `? ( minden időzítés_érték [parancsok] "folyamatnév).`
   Az időzítés értékét ezredmásodpercekben kell megadnunk.
   A példánkban levő teknőcök mozgását a következő utasításokkal írhatjuk le:

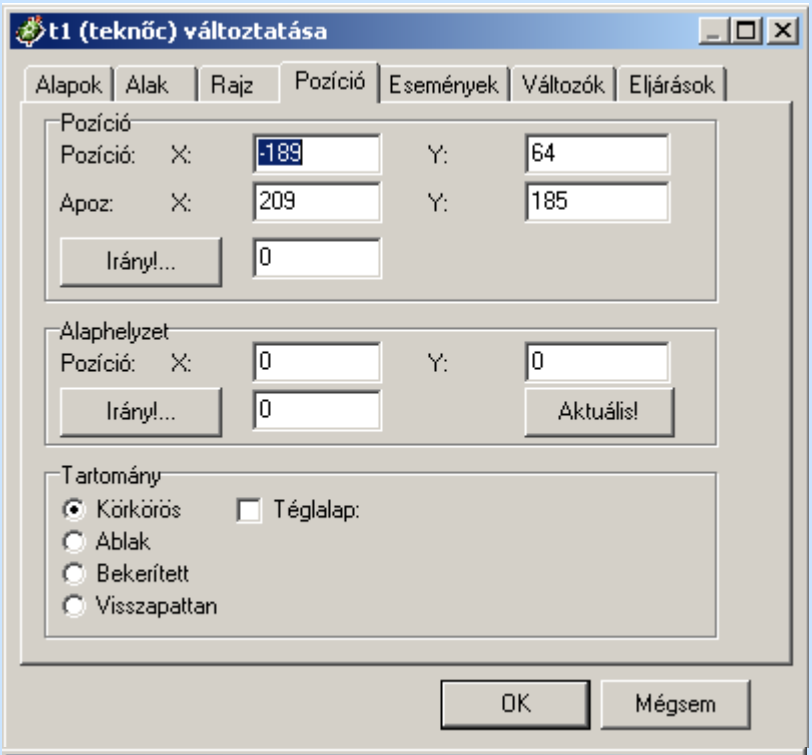
```
minden 100 [j 3]
minden 100 [e 3]
minden 100 [e 3 j 3]
```

2. Készítsünk egy Imagine projektet bringás néven, melyben több tekivel dolgozunk az ábrán láthatóhoz hasonlóan. Írjunk egy olyan eljárást, melyben a bringa alakú teknőc egy helyben teker, közben a házak jobbról balra mozognak!

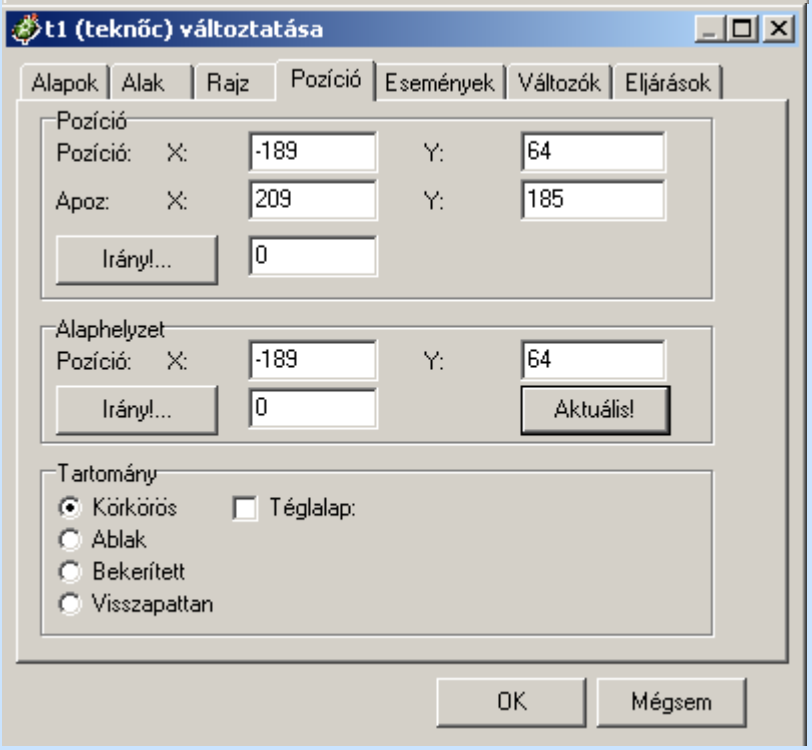
Segítség:
A t1 teknőc alakja a biciklist.lgf, melyet a **kepsor/forgo_alakok** között találunk.
A t2, t3, t4 teknőcök alakját az **epuletek** közül választhatjuk ki. Helyezzük el egymás mellé s a mozgatással meghatározott pozíciót állítsuk be alaphelyzetnek!



A teknőcöt eredeti pozíciójából elmozgattuk, melyet leolvashatunk az Alapok fül Pozíciójából.

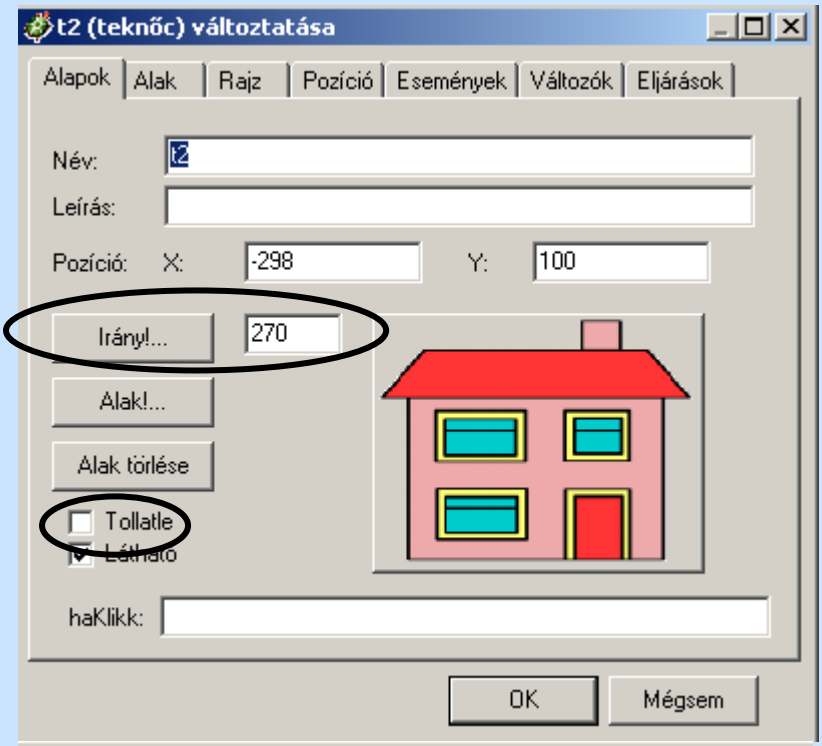


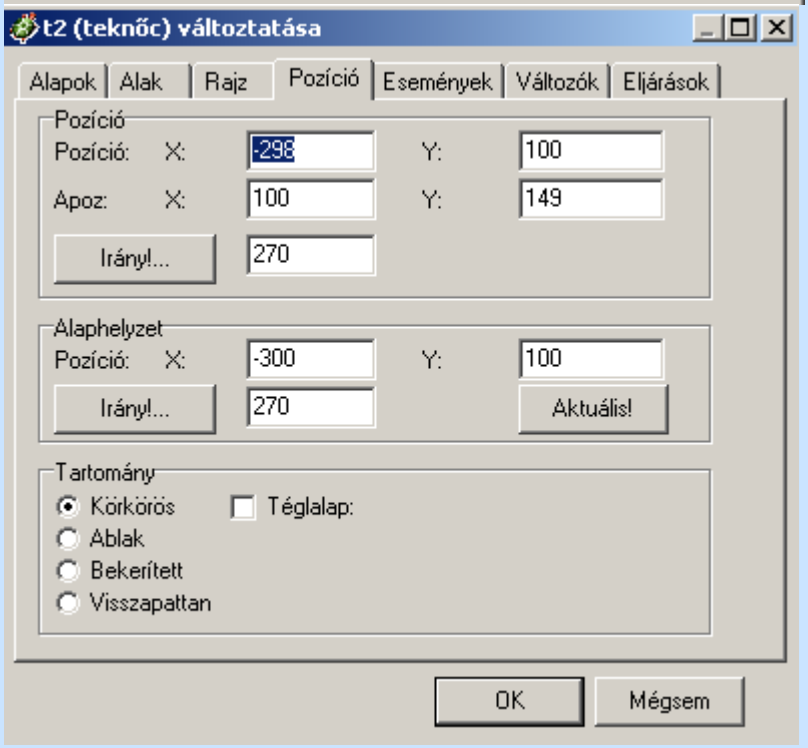
A Pozíció fülön leolvasható a teknőc alappozíciója, azaz az a koordináta ahol létrehoztuk.



Az Alaphelyzet-nél az Aktuális!-ra kattintva a teknőcnek az alappozíciója megegyezővé válik a mozgatás során beállított pozíciójával.

Ahhoz, hogy a házak mozogjanak jobbról balra, mindhárom teknőcnél (t2, t3, t4) be kell állítanunk a megfelelő irányt (Alaphelyzetre is állítsuk be), illetve a felemelt tollat, mert most nem kell, hogy a teknőcök rajzoljanak:





Az eljárás elkészítése:

```
eljárás mozog
ism 1000 [e 10 várj 100]
vége
```

Az eljárásnak a t1-re nem kell vonatkoznia, tehát a t2,t3, t4 teknőcök legyenek aktívak az eljárás meghívása előtt.

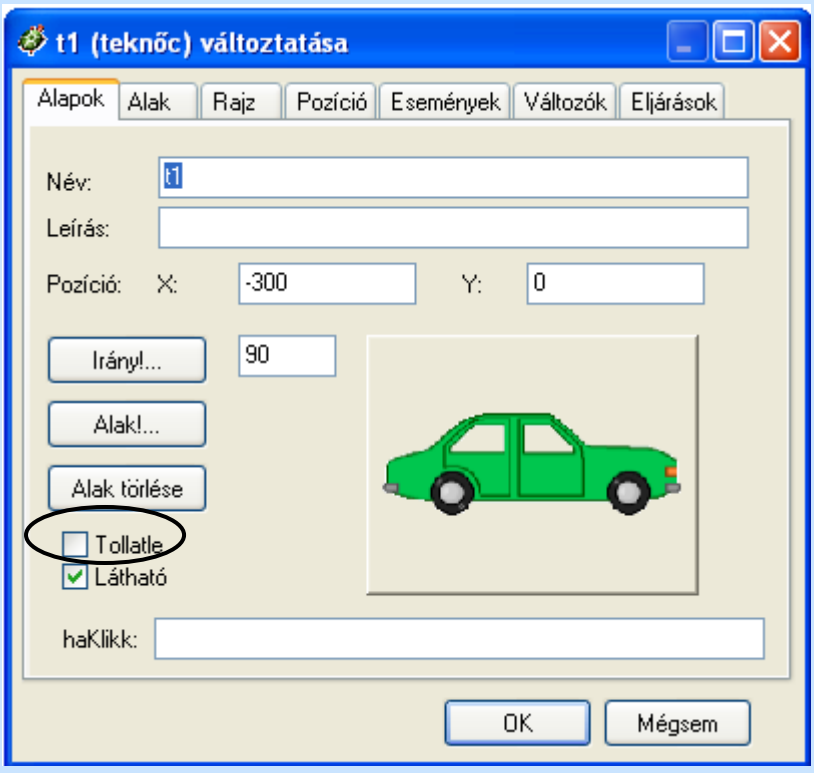
3. Készítsünk egy Imagine projektet karambol2 néven, melyben több tekivel dolgozunk az ábrán láthatóhoz hasonlóan. Írjunk egy olyan eljárást, melyben a zöld autó alakú teki (t1) jobbra haladva nekiütközik a tolató (azaz balra haladó) kék autó alakú tekinek (t2) (a tolató jármű lassabban halad, mint az egyenesen közlekedő), s az ütközés pillanatában megállnak!

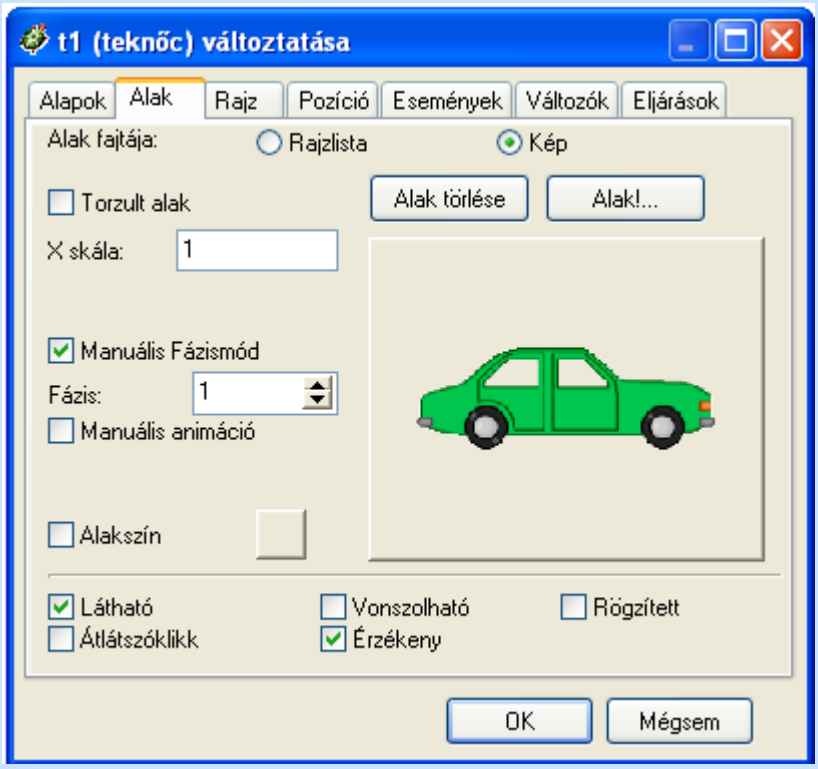
Ha azt akarjuk, hogy A teknőc reagáljon a B teknőccel (vagy hasonlóan több teknőccel) való ütközésre, akkor az alábbi pontoknak kell eleget tenni:·

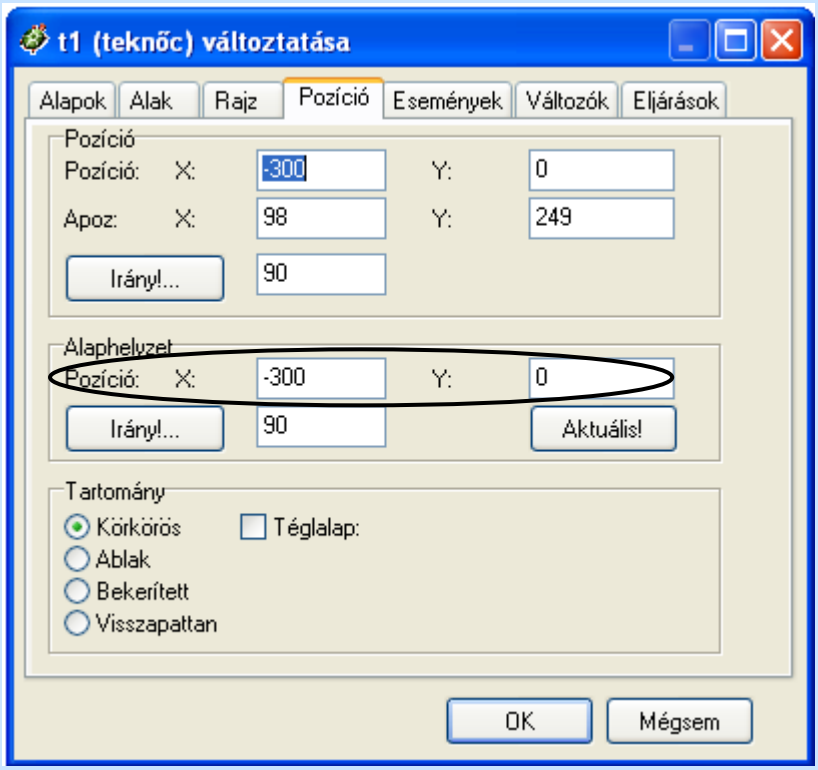
- A teknőcben definiálni kell a haÜtközik eseményt.
- A teknőcök érzékeny beállítását igaz-ra kell állítani.
- Az A teknőcnek mozognia kell, különben nem érzékelne ütközést.
- Az A és B teknőcnek is meg kell lenniük jelenítve.

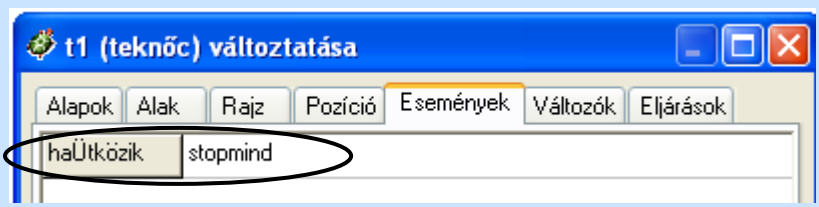
Ekkor, ha az A teknőc mozog és eltalálja a B teknőcöt, akkor lefut a haÜtközik eseménye.

t1 tulajdonságai:









Ezzel az utasítással az összes háttérfolyamatot leállítjuk.

A t2 teknőcre hasonló beállítások alkalmazhatóak.
Az alkalmazott eljárás:

```
eljárás karambol
ism 100 [figyelj "t2 e 4
figyelj "t1 e 7
várj 10]
vége
```

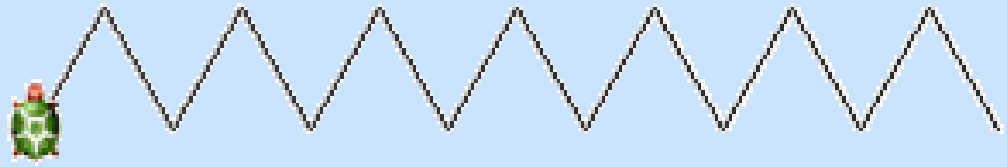
Sorminta, területminta

Sorminta

A sorminták mindig valamilyen alapelem felsorolásából állnak, azaz meg kell rajzolni egy alapelemet, el kell mozdítani a teknőcöt a következő alapelem kezdetéhez, majd ezt a tevékenységet kell megismételni a megfelelő számban.

Feladat:

1. Készítsük el a következő sormintát!



```
3szög_sor 40 7
```

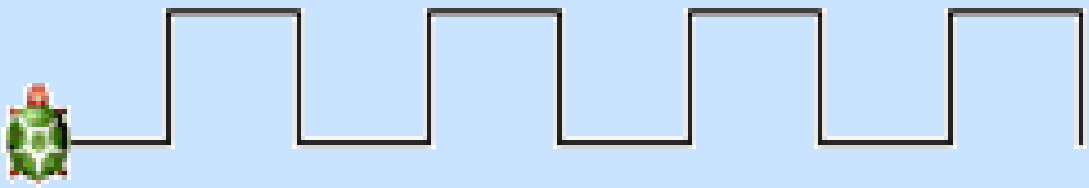
Megoldás:

Alapábra: 	`eljárás 3szög :a`
	`j 30`
	`ism 2 [e :a j 120]`
	`tf e :a j 120 tl`
	`b 30`
	`vége`

Következő alapelem kezdete	`eljárás elmozdul :a`
	`tf j 90 e :a b 90 tl`
	`vége`

A kész sorminta	`eljárás 3szög_sor :a :k`
	`ism :k [3szög :a elmozdul :a]`
	`tf b 90 e :a* :k j 90 tl`
	`vége`

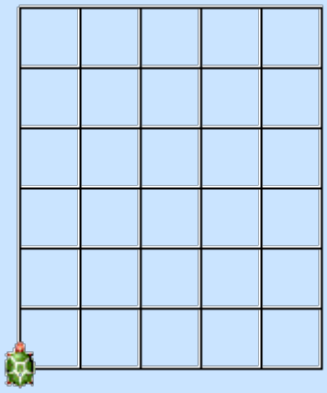
2. Készítsük el a következő sormintát!



Területminta

A területkitöltő mintázat (mozaik) tulajdonképpen adott darabszámú sormintából áll.

3. Írjon segédeljárások felhasználásával eljárást, amely négyzetekből mozaikot készít!



```
mozaik1 30 5 6
```

Segédeljárások:

- négyzet

```
eljárás négyzet :a
ism 4 [e :a j 90]
vége
```

- elmozdul

```
eljárás elmozdul :a
tf j 90 e :a b 90 tl
vége
```

- sorminta1

```
eljárás sorminta1 :a :n
ism :n [négyzet :a elmozdul :a]
vége
```

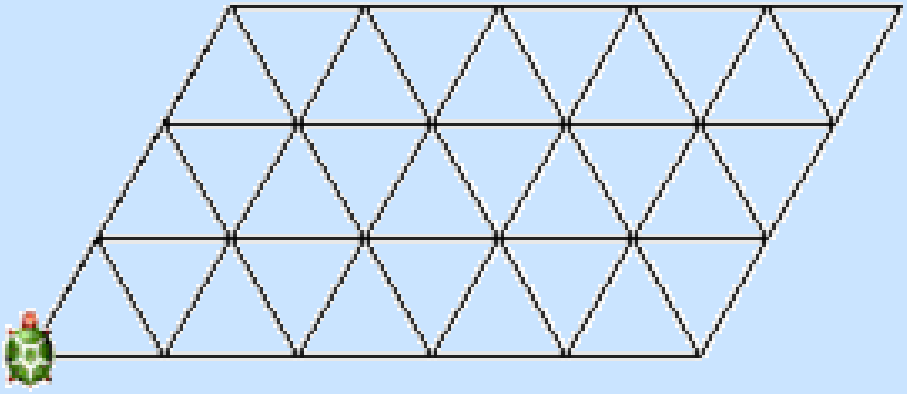
- újsor1

```
eljárás újsor1 :a :n
tf b 90 e :a *:n j 90 e :a tl
vége
```

- mozaik1

```
eljárás mozaik1 :a :n :k
ism :k [sorminta1 :a :n újsor1 :a :n]
tf h :a *:k tl
vége
```

4. Írjon segédeljárások felhasználásával eljárást, amely háromszövegekből mozaikot készít!



mozaik2 40 5 3

Segédeljárások:

- 3szög

```
eljárás 3szög :a
j 30 ism 3 [e :a j 120] b 30
vége
```

- dupla_3szög

```
eljárás dupla_3szög :a
ism 2 [3szög :a
j 30 e :a j 60 e :a j 90]
vége
```

- elmozdul

```
eljárás elmozdul :a
tf j 90 e :a b 90 tl
vége
```

- sorminta2

```
eljárás sorminta2 :a :n
ism :n [dupla_3szög :a elmozdul :a]
vége
```

- újsor2

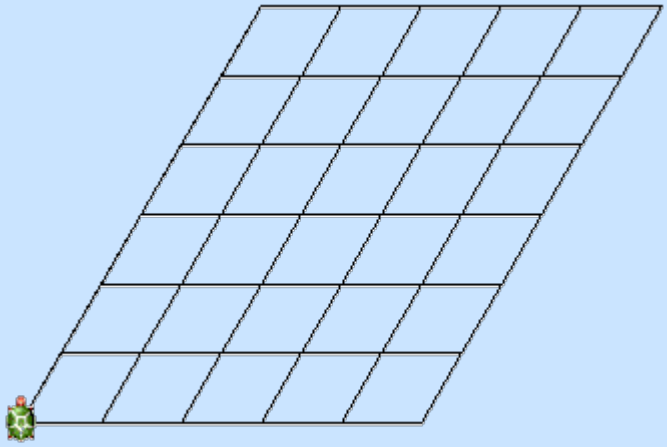
```
eljárás újsor2 :a :n
j 30 e :a b 120 e :a*:n j 90
vége
```

- mozaik2

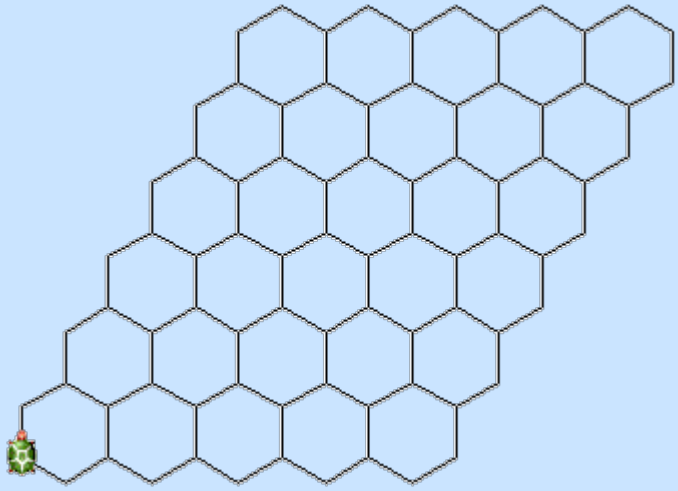
```
eljárás mozaik2 :a :n :k
ism :k [sorminta2 :a :n újsor2 :a :n]
j 30 h :a* :k b 30
vége
```

Gyakorlás

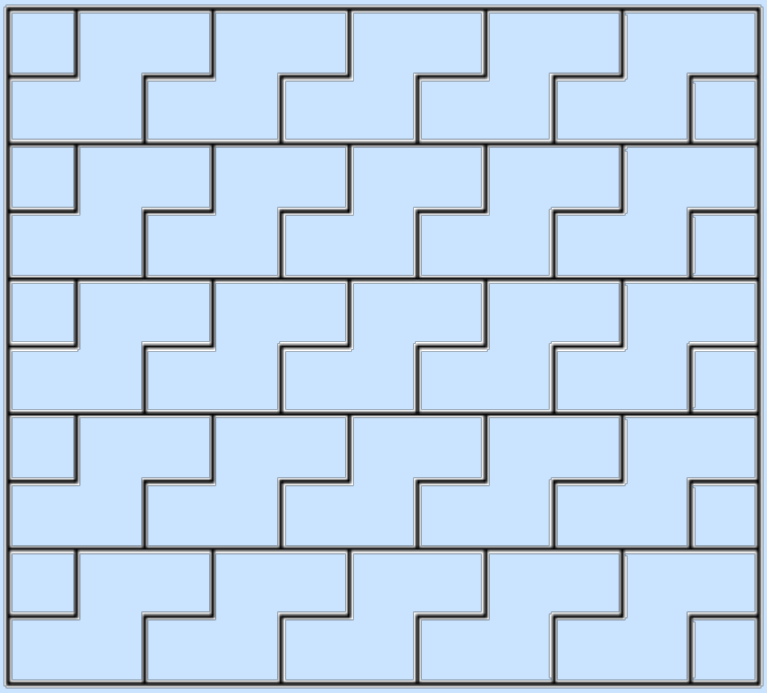
1. Írjon segédeljárások felhasználásával eljárást, amely rombuszokból mozaikot készít!

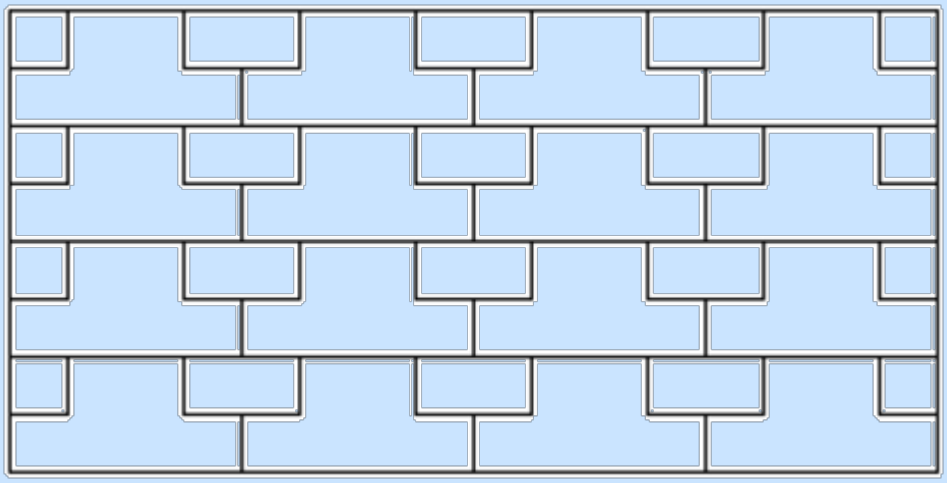


2. Írjon segédeljárások felhasználásával eljárást, amely hatszögekből mozaikot készít!

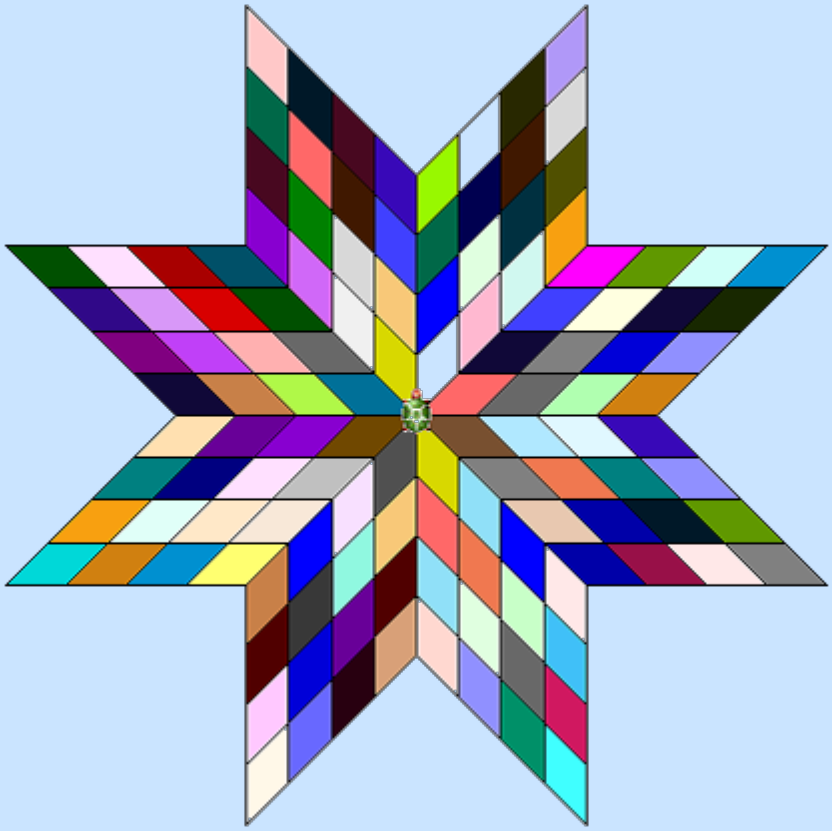


3. Készítsünk parkettát a következő alakzatokból!

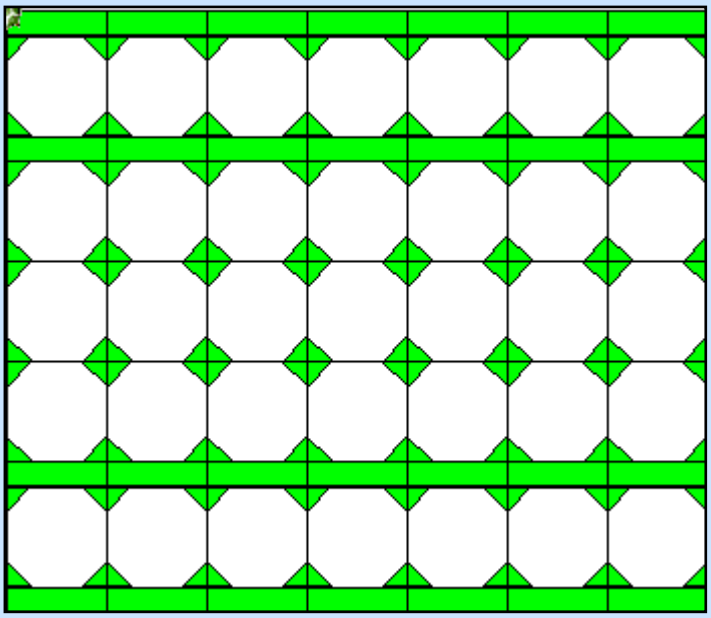




4. Lássuk színesben!



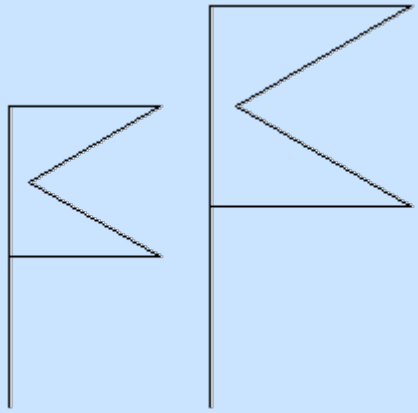
5. Végezetül egy igazán nehéz feladat! (Megoldás megtalálható!)
   A FAL :n :m :hossz eljárás :n csempesort rajzol, amely:m darab :hossz méretű csempéből áll. Az alsó és a felső csempesort mindkét oldaláról egy-egy zöld vonalsor határolja.



Geometriai transzformációk

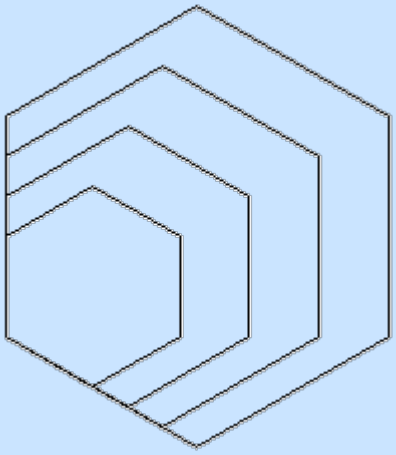
Hasonlósági tétel: Ha a végrehajtás során a szögeket változatlanul hagyjuk, de egy szorzóval beszorozzuk a lépéseket, akkor a szorzó arányának megfelelő alakzatot kapjuk.

Tulajdonképpen ezt alkalmaztuk a paraméteres eljárások esetében.



```
eljárás zászló :h
e :h j 90 e :h*0.5
j 150 e :h*0.5 b 120
e :h*0.5 j 150 e :h*0.5 j 90
h :h*0.5
vége
```

Készítsünk el egymáshoz hasonló hatszögeket egymásban!

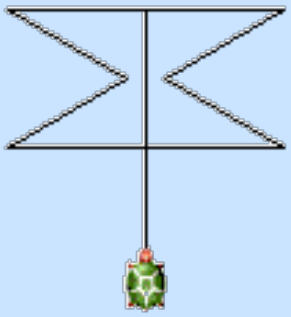


```
hatszög 50
hatszög 70
hatszög 90
hatszög 110
```

1. **tükrözési tétel:** Tetszőleges alakzat megrajzolása után, ha a lépéseket változatlanul hagyjuk, viszont minden fordulási parancsot egyenlő nagyságú, de ellentétes irányú fordulással helyettesítünk, akkor az alakzatot a teknőc kiinduló pontján átmenő, kiindulási irányba eső egyenesre tükrözzük. (balra helyett jobbra és viszont, vagy a paramétert ellentétes előjelűre cseréljük)
2. **tükrözési tétel:** Ha a fordulatokat változatlanul hagyjuk, de lépések hosszát ellentétesre változtatjuk, akkor a teknőc kiinduló pontjára tükrözzük az alakzatot
3. **tükrözési tétel:**Ha a fordulatok irányát és a lépések hosszát is ellentétesre változtatjuk, akkor az alakzatot a teknőc kiinduló pontján átmenő, kiindulási irányáras merőleges egyenesre tükrözzük.

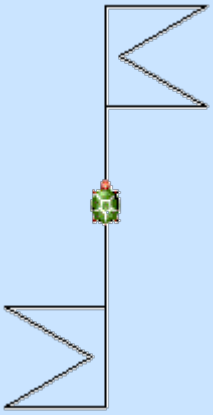
Alkalmazzuk a zászló eljárásunkra a tükrözési tételeket:

1. Tükrözési tétel



```
eljárás tükör1_zászló :h
e :h b 90 e :h*0.5
b 150 e :h*0.5 j 120
e :h*0.5 b 150 e :h*0.5 b 90
h :h*0.5
vége
```

2. Tükrözési tétel



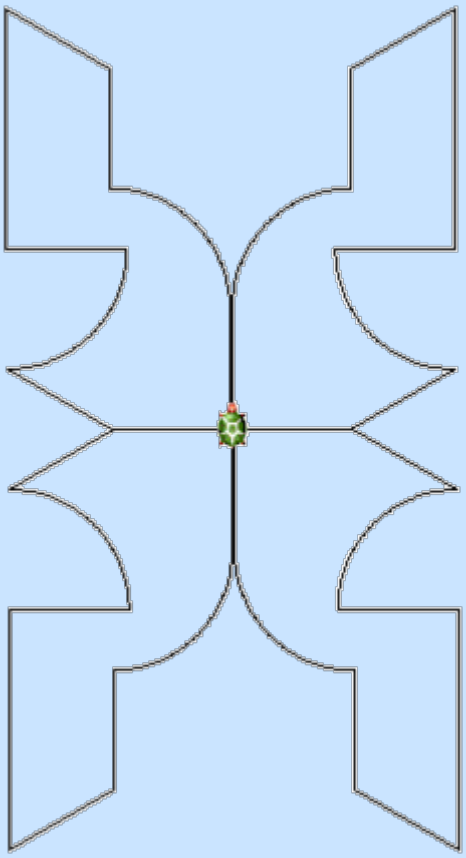
```
eljárás tükör2_zászló :h
e -:h j 90 e -:h*0.5
j 150 e -:h*0.5 b 120
e -:h*0.5 j 150 e -:h*0.5 j
90
h -:h*0.5
vége
```

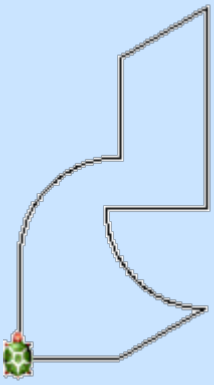
3. Tükrözési tétel

```
eljárás tükör3_zászló :h
 e -:h j -90 e -:h*0.5
j -150 e -:h*0.5 b -120
e -:h*0.5 j -150 e -:h*0.5 j
-90
h -:h*0.5
vége
```

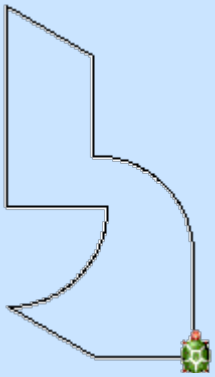
Feladat:

Tükrözési tételek használatával készítsük el a következő ábrát:

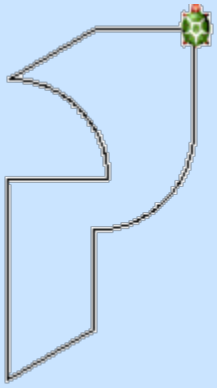




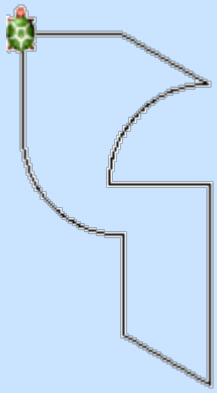
```
eljárás kezd :x
e :xism 90 [e :x/180*3.14 j 1]
 b 90 e :x j 60 e :x j 120
e :x*2 j 90 e :x b 90
ism 90 [e :x/180*3.14 b 1] j 90
 j 60 e :x j 30 e :x j 90
vége
```



```
eljárás tükör1 :x
e :xism 90 [e :x/180*3.14 j -1]
 b -90 e :x j -60 e :x j -120
e :x*2 j -90 e :x b -90
ism 90 [e :x/180*3.14 b -1] j -90
 j -60 e :x j -30 e :x j -90
vége
```

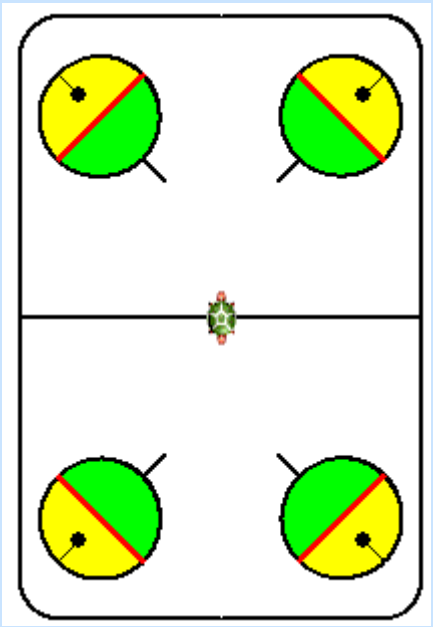


```
eljárás tükör2 :x
 e -:xism 90 [e -:x/180*3.14 j 1]
 b 90 e -:x j 60 e -:x j 120
 e -:x*2 j 90 e -:x b 90
ism 90 [e -:x/180*3.14 b 1] j 90
 j 60 e -:x j 30 e -:x j 90
vége
```



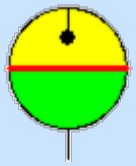
```
eljárás tükör3 :x
  e -:xism 90 [e -:x/180*3.14 j -1]
 b -90 e -:x j -60 e -:x j -120
 e -:x*2 j -90 e -:x b -90
ism 90 [e -:x/180*3.14 b -1] j -90
 j -60 e -:x j -30 e -:x j -90
vége
```

Készítsük el 4 teknőc segítségével és a tükrözés felhasználásával a magyar kártyák közül a nyarat szimbolizálót!

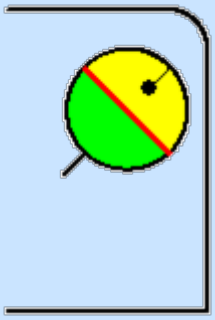


Bontsuk a feladatot részekre.

Ha négy teknőccel dolgozunk, akkor miden teki a kártya negyedét fogja elkészíteni, ebben az esetben elegendő csupán az első tükrözési tételt használni, mert ennek segítségével a lap felső része elkészül. Az alsó részen, ha a teknőcök iránya 180 fok, akkor a felső „negyedlapok” itt is használhatóak lesznek.



```
eljárás tök
tl
tsz! "fekete
e 15
sugaraskör 30
j 90
tf
e 30
j 90
e 30
tl
tv! 3
tsz! "piros
h 60
e 30
tv! 1
tsz! "fekete
tf b 90
e 30
tl
h 15
pontméret 8
tf h 10
tlsz! "sárga tölt
h 20
tlsz! "zöld tölt
tf h 30 tl
vége
```



```
eljárás negyedlap
tv! 2 tsz! "fekete
tf
e 150
j 90
tl
e 80
ism 90 [e 20 *3.14159/180 j 1]
e 130
j 90
e 100
j 90
tf
e 40
j 45
e 40
tök
tf
h 40
b 45
h 40
vége
```

```
eljárás tükör1
tv! 2 tsz! "fekete
tf
e 150
j -90
tl
e 80
ism 90 [e 20 *3.14159/180 j -1]
e 130
j -90
e 100
j -90
tf
e 40
j -45
e 40
tök
tf
h 40
b -45
h 40
vége
```

A teknőcök tulajdonságait be kell állítani:

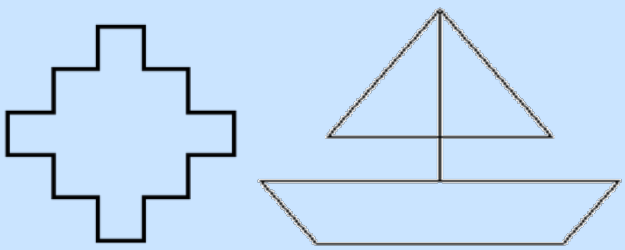
Minden teknőc alappozíciója az origó, azaz a [0,0], a t1, t2 iránya 0, a t3 , t4 iránya pedig 180.

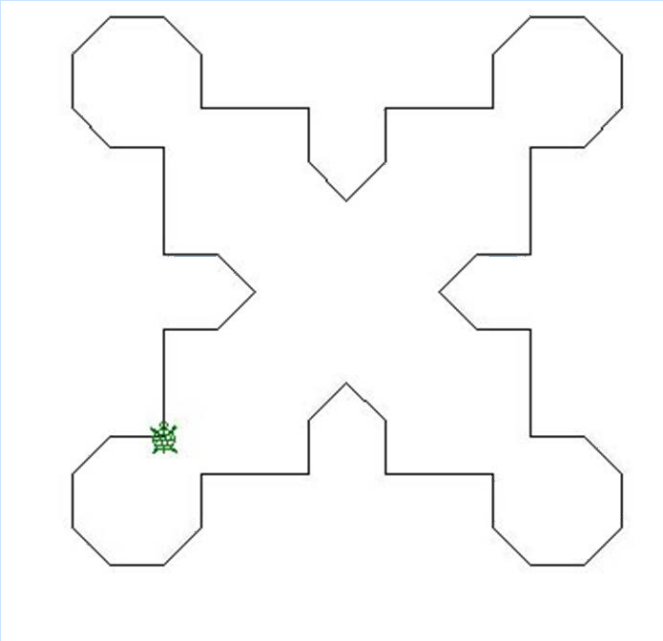
A kész eljárás:

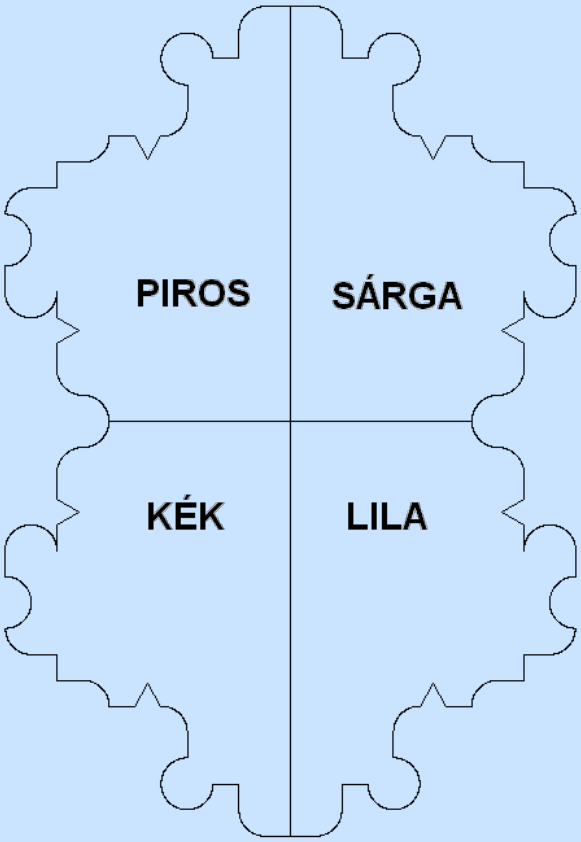
```
eljárás nyár
figyelj "t1 negyedlapvárj 500
figyelj "t2 tükör1 várj 500
figyelj "t3 negyedlap várj 500
figyelj "t4 tükör1 várj 500
vége
```

A várj utasítást nem fontos beépíteni, csak így látványosabb.

Gyakorlás



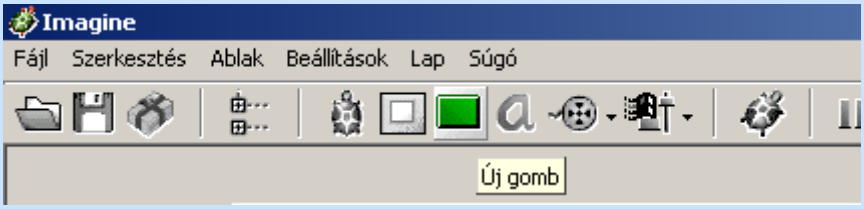


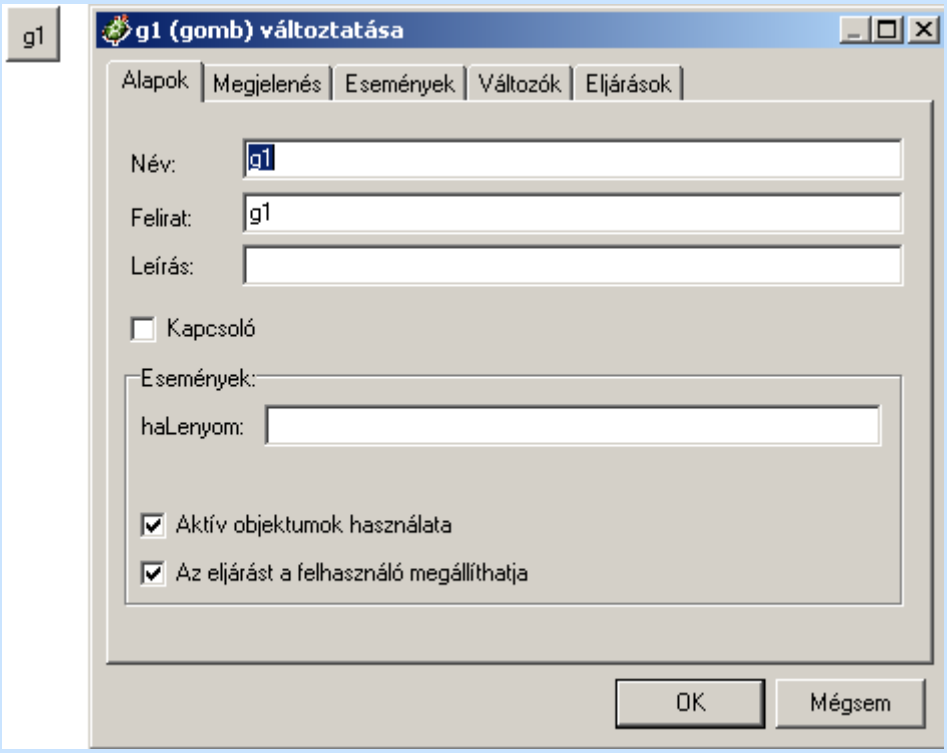

PIROS
SÁRGA
KÉK
LILA

Gombok létrehozása

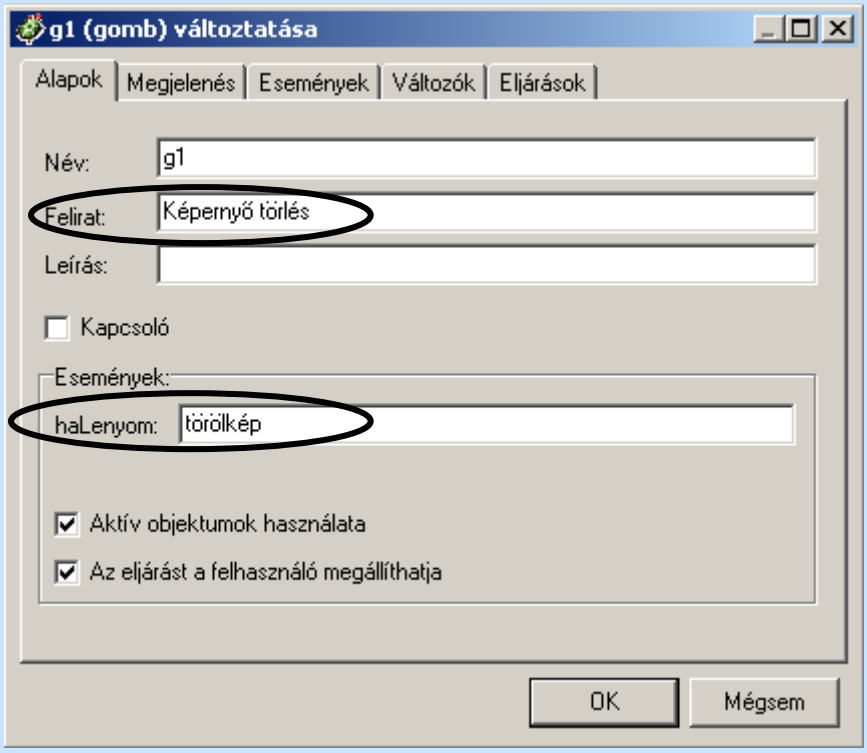
A gombok olyan grafikus objektumok a lapon vagy panelen, amelyeke keresztül az egérrel vezérelhetjük a programunkat.

Gombot létrehozni legegyszerűbben a Fő eszköztár Új gomb ikonjának megnyomásával tudunk. Ez közvetlenül nem használható, meg kell változtatni a beállításait.



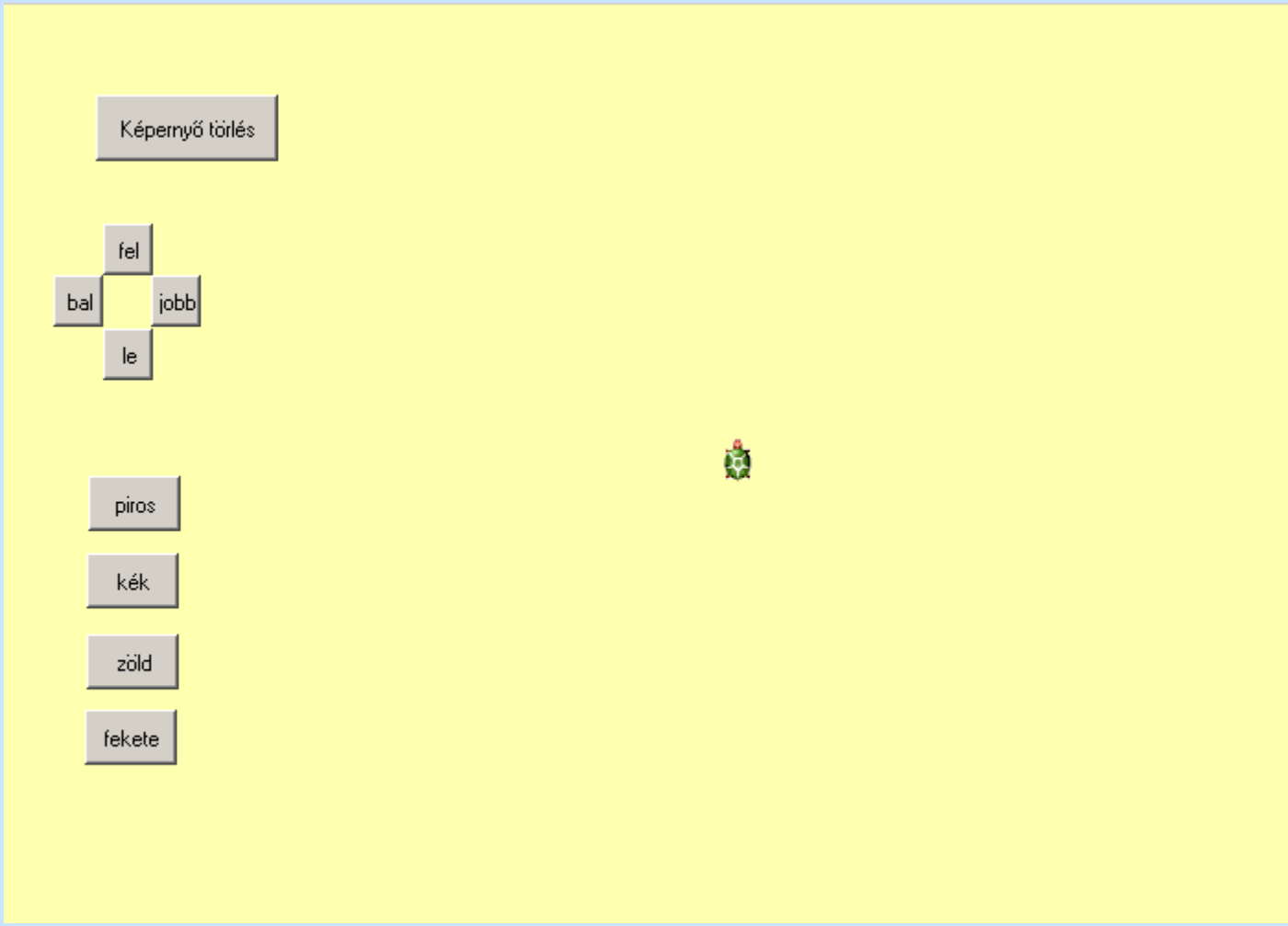


Először készítünk egy gombot a képernyő törlésére.

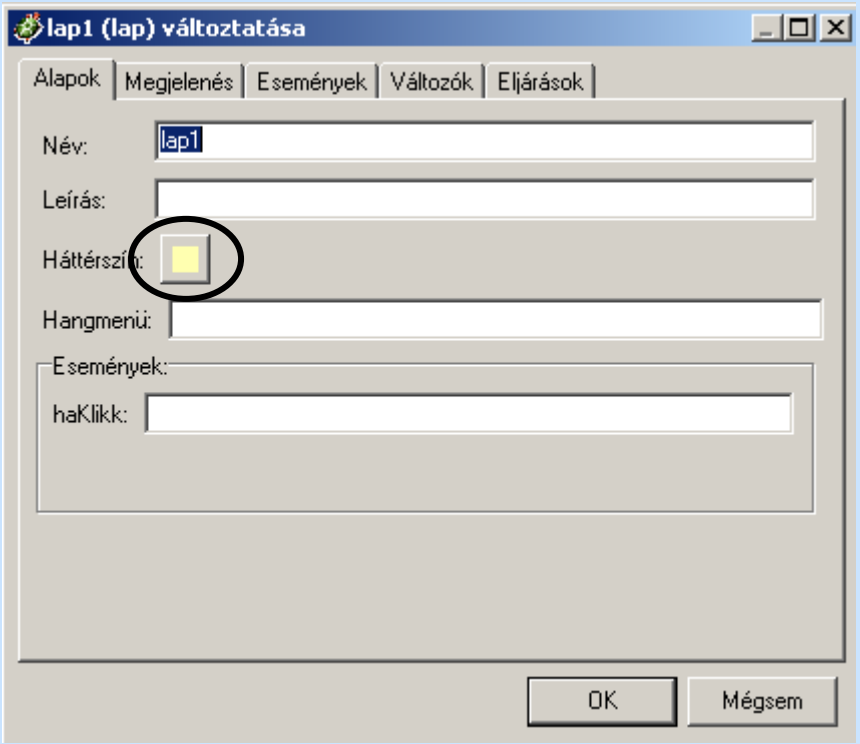


Méretének megváltoztatását a jobb egérgomb és a Ctrl együttes lenyomása mellett, jobb alsó sarkánál való húzással tehetjük meg.

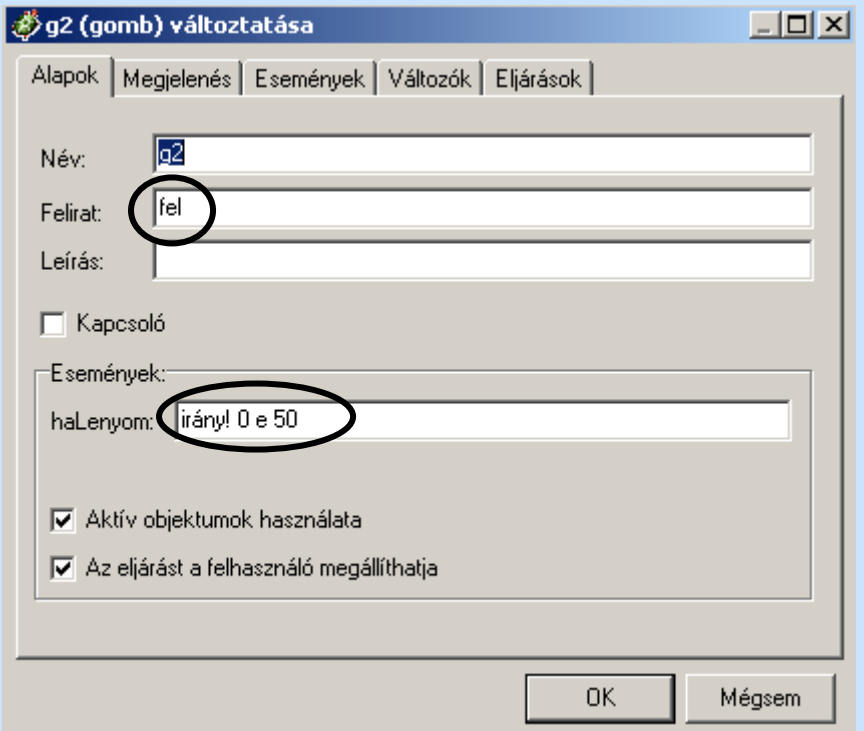
Készítsük el a következő projektet!

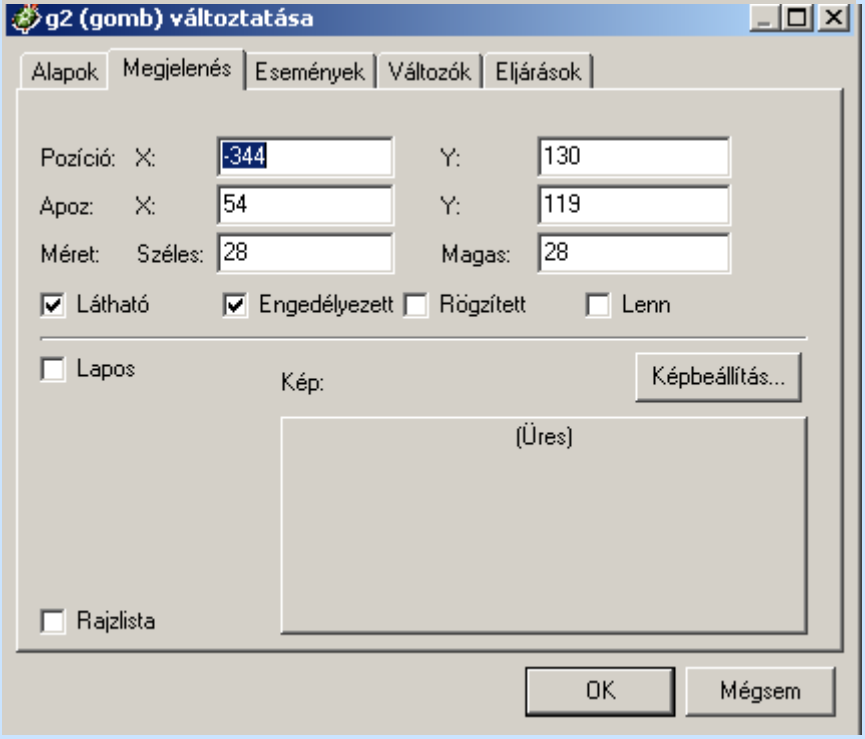


Állítsuk be a lap hátterét:



A fel, le jobb, bal gombok beállítása hasonló:





Természetesen az irány! megadása az adott irányoknak megfelelő:

fel: irány! 0

jobb: irány! 90

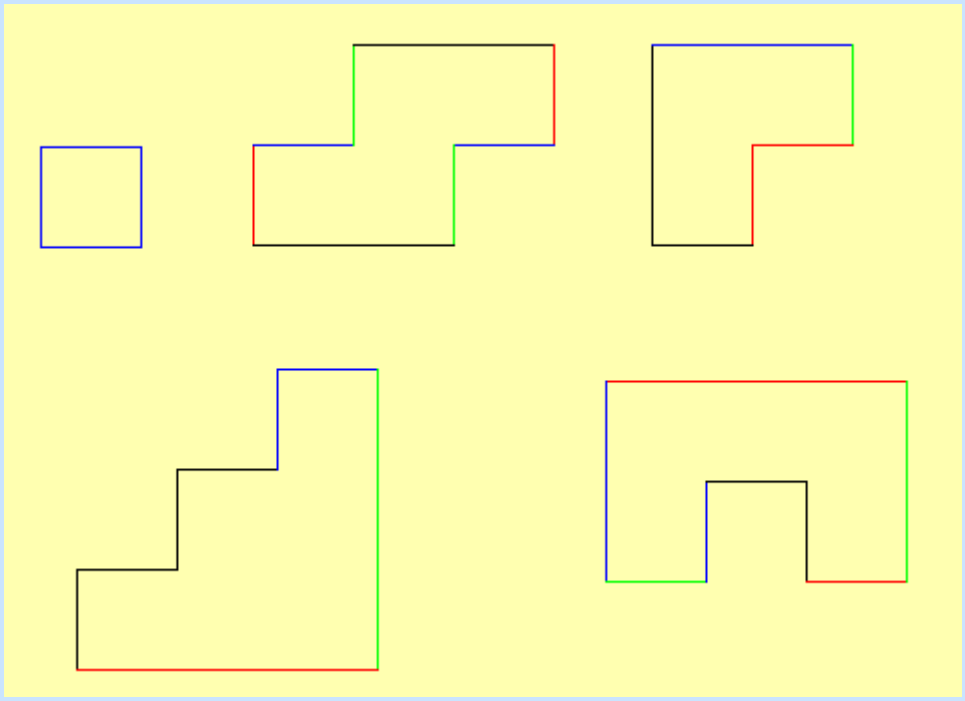
le: irány! 180

balra: irány! 270

A szín beállítása is ezekkel megegyezik, csak a haLenyom – nál az adott tollszín szerepel (tsz! ”piros)!

Ha kész indulhat a játék!

Készítsük el a gombok segítségével a következő ábrákat:

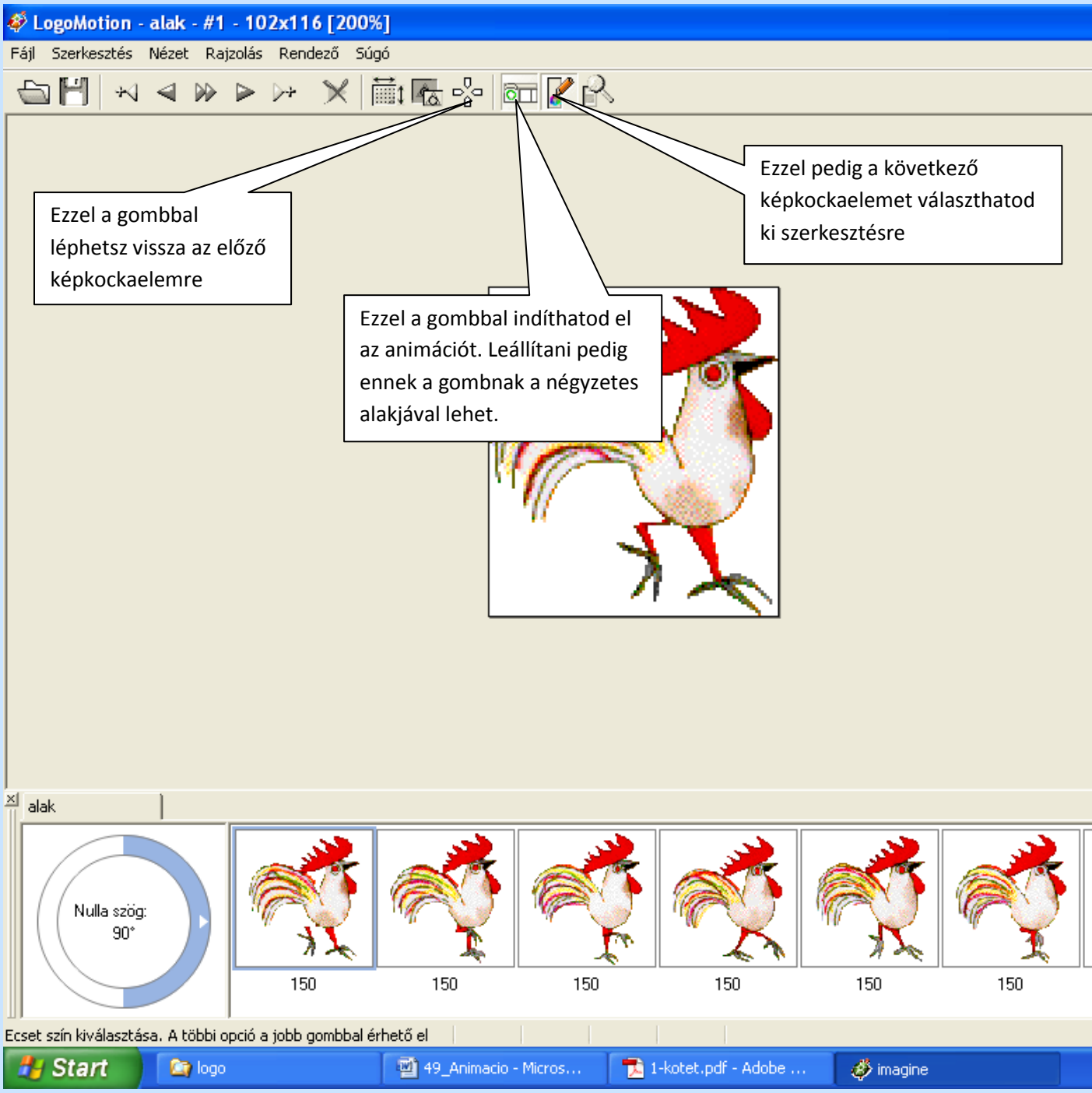


Animáció készítése

Az egyszerű képek megrajzolásán túl lehetőségünk van létrehozni mozgó képeket, animációkat. Az animáció elkészítéséhez a LOGOMotion szerkesztő programot fogjuk használni. Ezzel a programmal már találkozhattál, amikor a teknőc alakját kellett átalakítanod (egérke kék szeme).

Az animáció összeállítása előtt meg kell rajzolnunk az animációs fázisokat, ezeket az Imagine-ben*képkockaelemeknek* hívunk. Egy képsor több képkockából is állhat. Amikor a képkocka elemeit egymás után folyamatosan lejátsszuk, akkor tulajdonképpen magát az animációt látjuk. Minden képkockaelemhez rendelhetünk egy meghatározott *késleltetési idő*t, mellyel azt szabályozzuk, hogy mennyi ideig legyenek láthatóak a lejátszás során. Ezt az értéket ezredmásodpercben kell megadni, így minél kisebb értéket adunk meg, annál gyorsabb lesz az animáció.

Nézzünk meg egy már létező animációt, nyissunk meg a LOGOMotion-ban a *kakas.gif* képet



Az animáció fázisait a rendező ablakban látjuk. Az animáció a kakas lépegetését mutatja, 1 képkockából és 8 képkockaelemből áll. A képkockaelemek alatt látható a késleltetés mértéke, amelyet akár meg is változtathatunk (duplán kattintunk a beállított késleltetési időn, majd átírjuk a kívánt értékre). A rendező ablakban kiválasztott képet láthatjuk a szerkesztő ablakban nagy méretben, ekkor tudunk rajta módosításokat végrehajtani.

1. Változtassuk meg az időzítési értéket úgy, hogy a kakas lassabban sétáljon, majd mentsük le a saját könyvtárunkba **kakas2** néven. Az alap animációban a kakas 150 ezredmásodpercenként lép egyet. Állítsuk át a lépéseket 300-ra.

2. Módosítsuk az animációt úgy, hogy egyszer lassan, egyszer gyorsabban lépeget!

Arra nincs lehetőség, hogy egy képkockaelemhez két időértéket is beállítsunk, de bővíthetjük úgy az animációt, hogy a meglévő képkockaelemek mögé még egyszer odamásoljuk a már meglévő elemeket, így ezeknek már beállítható új késleltetés.

Kattintsunk a rendező ablakban az első elemre, majd a SHIFT billentyű nyomva tartása mellettkattintsunk az utolsó elemre is. Válasszuk ki a SZERKESZTÉS/MÁSOLÁS menüpontot (CTRL+C), álljunk az utolsó képkockaelemre, és válasszuk ki a SZERKESZTÉS/BEILLESZTÉS (CTRL+V), ennek eredményeként a kijelölt képkockaelemek az utolsó elem utáni helyre kerülnek. Most állítsuk be a késleltetési időket 150 illetve 300 ezredmásodpercre. Mentsük el `kakas_setal.gif` néven.

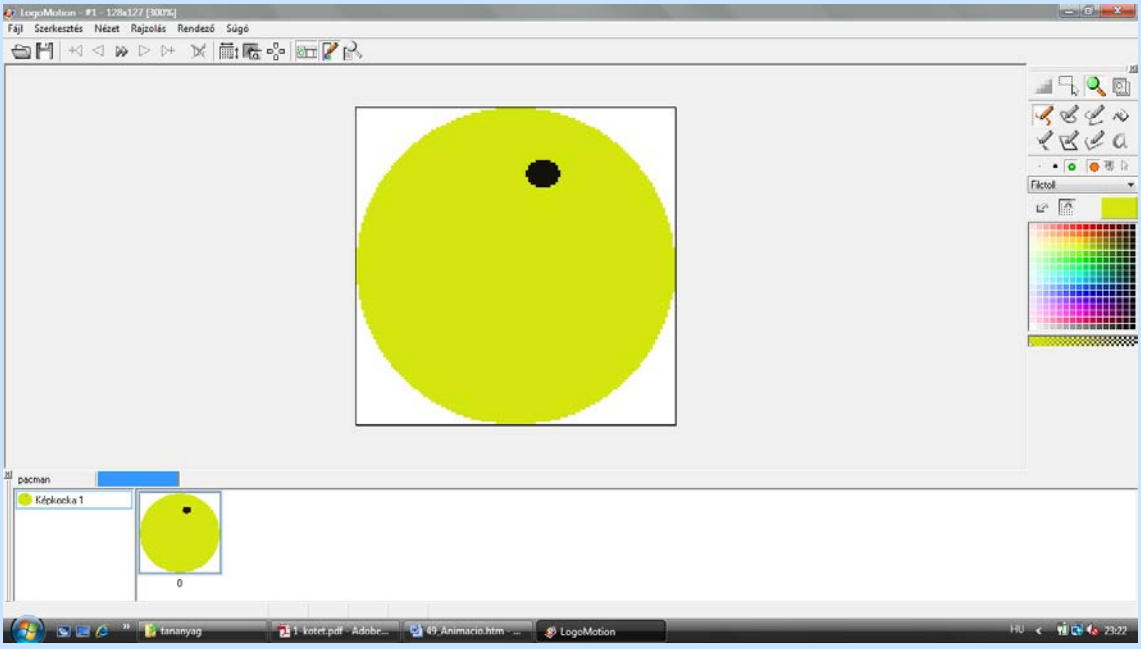
1. Készítsünk egy vészvillogós rendőrautót!

Rajzoljunk egy autót, SZERKESZTÉS/MÁSOLÁS segítségével készítsünk négy egyforma képkockaelemet, majd egyesével változtassuk őket úgy, hogy villogó, mozgó hatást keltsen. Különböző késleltetéssel lehet gyorsabb illetve lassabb az autónk. Mentsük el **`rendor_auto.gif`** néven.

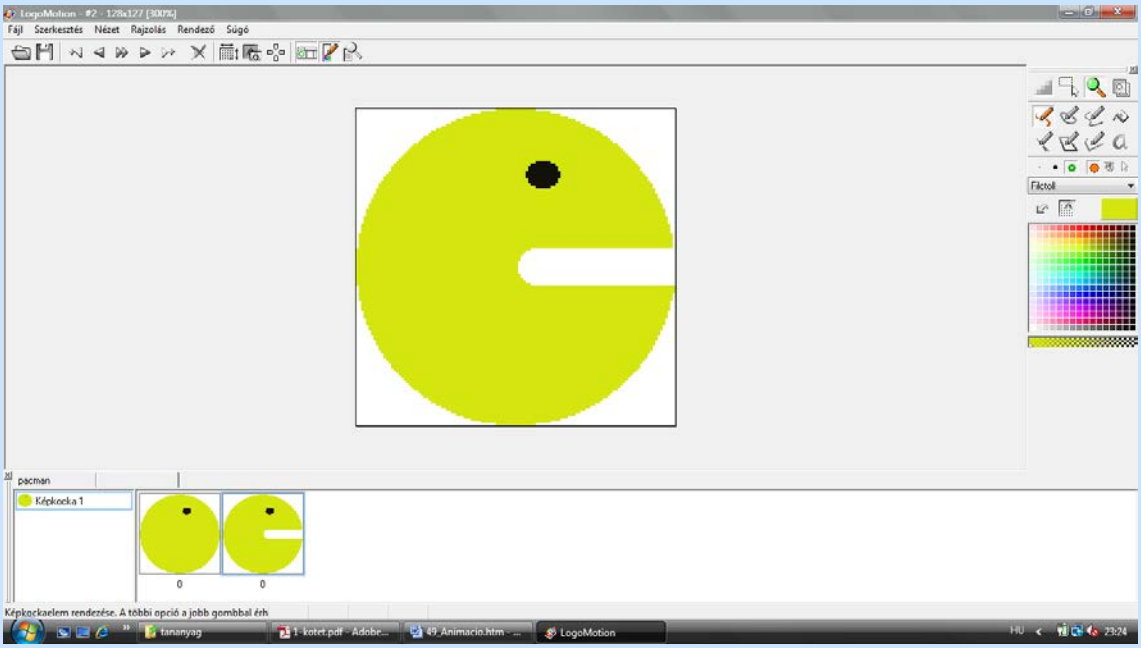
Irányfüggő és irány független animáció

2. Készítsünk el egy egyszerű animációt, a játékokból jól ismert Pac-man figurát.

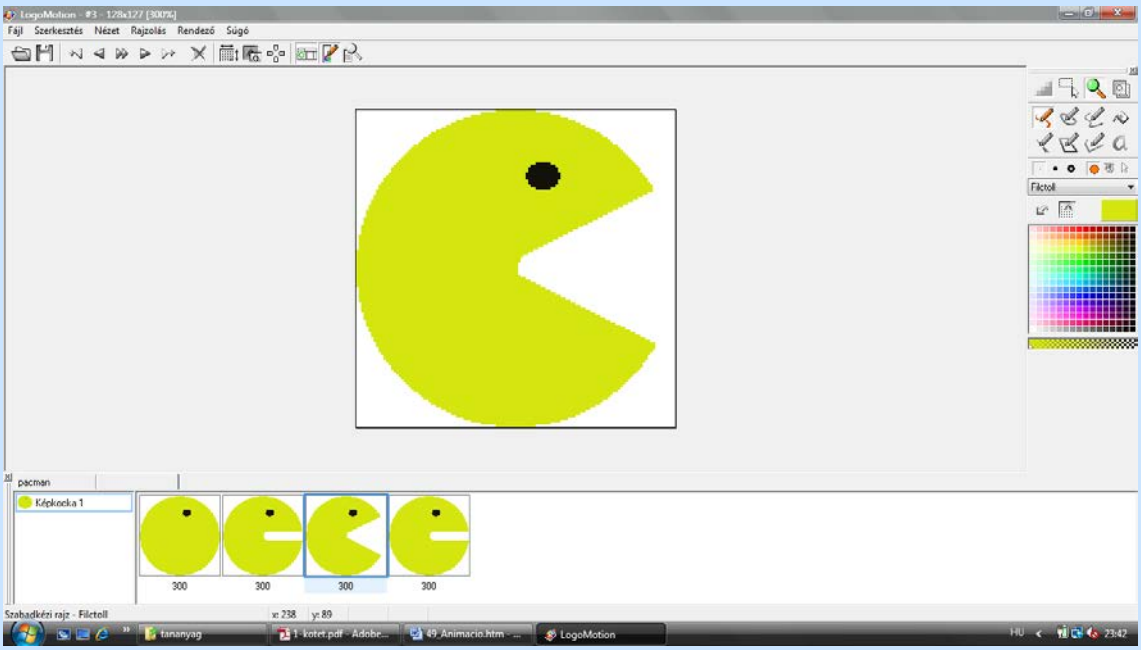
Első lépésként készítsünk el egy új képet! A RENDEZŐ/KÉPBEÁLLÍTÁSOK menüben állítsuk be, hogy a KÉPSTÍLUSA**Alak** legyen, így létrejön egy 128x128-as méretű kép átlátszó háttérrel. A kiinduló állapotban legyen az alak szája zárt, vagyis csak egy kitöltött sárga kört és egy fekete kört a szemet kell megrajzolnunk. Az animáció többi fázisát ebből alakítjuk ki.



Készítsünk egy másolatot az elkészült képkockaelemről a már megtanult módon. Ezután alakítsuk át úgy a képet, hogy azon Pac-man kicsit kinyissa a száját. Ezt legegyszerűbben egy nagyméretű ecset és a jobb egérgombos rajzolás segítségével érhetjük el. Amikor jobb egér gombbal rajzolunk, akkor mindaz átlátszóvá válik, amire az egérrel rámutatunk.



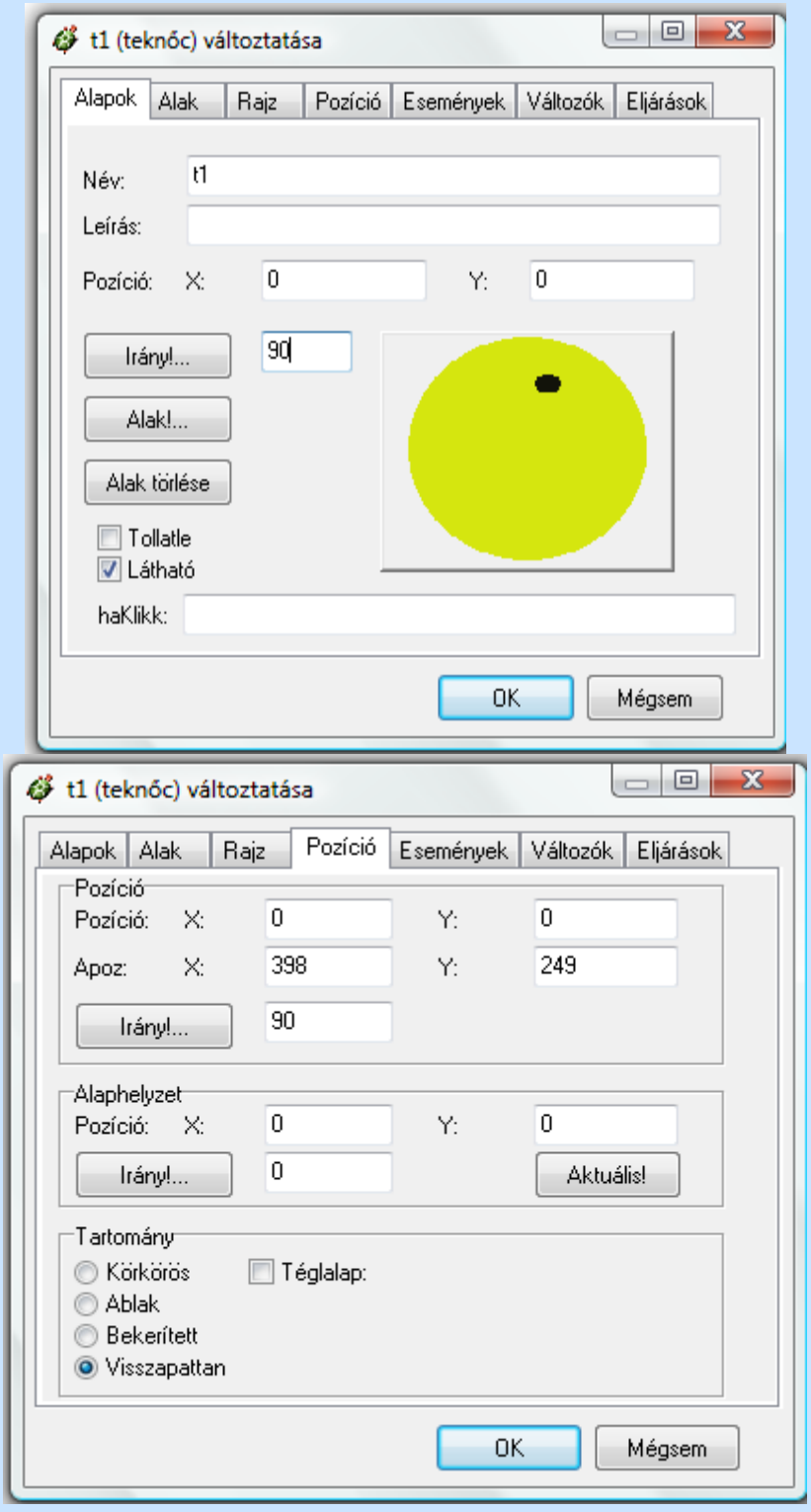
Most erről is készítsünk egy újabb másolatot és ezen teljesen nyissuk ki a száját a figuránknak! Állítsunk be 300-as késleltetést. Ha ezután lejátsszuk az animációt, akkor azt látjuk, hogy Pac-man hirtelen csukja be a száját, tehát még egy kicsit nyitott szájú elemre lesz szükségünk. Szerencsére ezt már nem kell elkészíteni, csupán a második elemet kijelölni és a végére beilleszteni.



Mentsük az animációt Pacmannéven.

Most próbáljuk ki az Imagine-ben. Hozzunk létre az ImagineLOGO-ban egy teknőcöt, nevezzük el Pacman-nak, majd állítsuk be a teknőc alakjának a előbb létrehozott animációt.

Állítsuk be az irányt 90°-ra, vegyük ki a pipát a Tollatle elől és állítsuk be a tartomány jellemzőknél a visszapattanást.



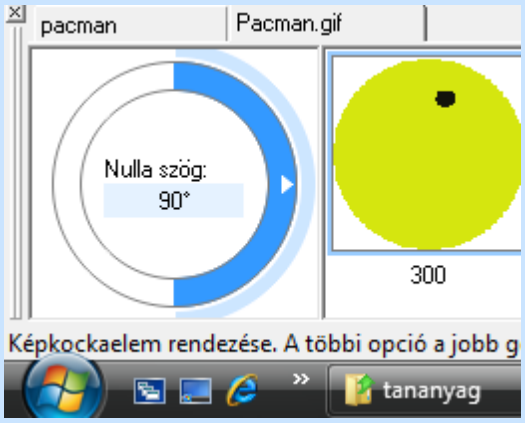
Adjuk ki a következő parancsot:

```
? minden 10 [előre 1]
```

Ennek hatására a beállítások alapján Pacman-ünk elindul jobbra előre 10 ezredmásodpercenként, majd amikor eléri a lap szélét, visszapattan. Sajnos továbbra is csak balra néz, ez így nem túl szép.

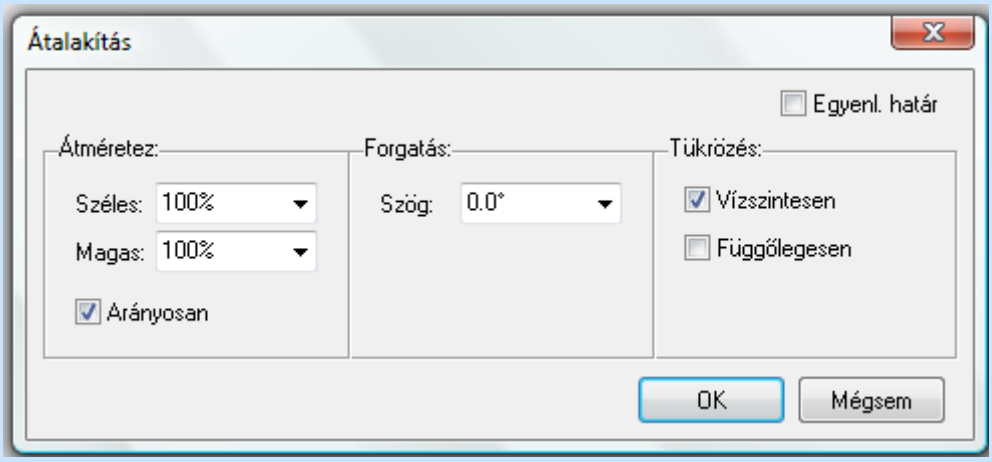
Változtassuk meg az animációt úgy, hogy más fázisok látszódjanak, ha Pacman balra vagy jobbra halad. Nyissuk meg az animációt a LOGOMotion-ban. A rendező ablak bal oldalán kattintsunk a jobb egérgombbal, válasszuk ki az IRÁNYTŰ MÓD menüpontot. Az iránytű nincs felosztva, ez azt jelenti, hogy csak egy képkockából áll az animáció.

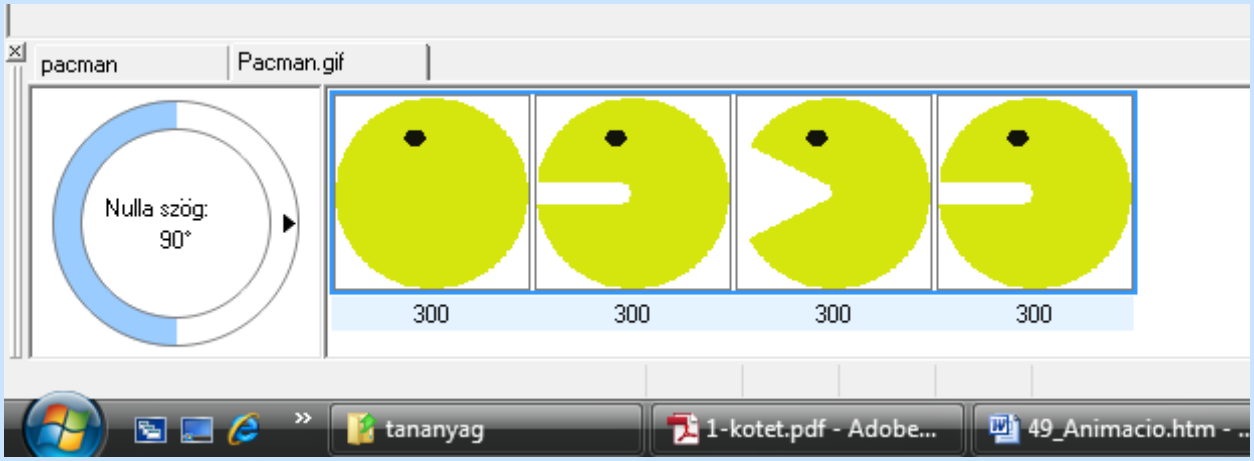
Új irány megadásához az iránytűn állva meg kell nyomnunk az INSERTbillentyűt, vagy jobb egérgombbal előhívjuk a SZERKESZTÉS/BESZÚR(MÖGÉ) menüpontot. Ennek hatására előjön egy új irány ami nem tartalmaz képkockaelemet. Kattintsunk az iránytű felfelé mutató részére. Megjelennek Pacman elkészített képkockaelemei. El kell forgatnunk az iránytűt jobbra 90°-ra. Ezt a nyíl forgatásával (ezt elég nehéz pontosan megtenni) vagy a Nulla szög mezőbe írva tehetjük meg.



Most készítsük el a balra haladó figurát!

Első lépésként jelöljük ki a jobbra néző változat összes elemét, kattintsunk az iránytű balra néző tartományára majd illesszük be a képkockaelemeket és töröljük az első üres elemet(kijelöljük és lenyomjuk a Delete billentyűt). Most végezzük el a tükrözést, hogy az alakok megforduljanak. Jelöljük ki az elemeket, kattintsunk a jobb egérgombbal és válasszuk ki a RENDEZÉS/ÁTALAKÍTÁS… . A megjelenő ablakban tegyünk egy pipát a **Vízszintes tükrözés** elé. Kattintsunk az OK gombra és el is készültünk az új iránnyal.





Mentsük el `Pacman2irany` néven, majd hozzunk létre egy újabb teknőcöt az előbbivel megegyező beállításokkal, alaknak pedig állítsuk az újonnan létrehozott kétirányú animációt és próbáljuk ki a működését!

Feladatok:

1. Készítsük el Pacman olyan változatát, ami nem csak jobbra és balra, hanem le és felfelé is más animációt használ!

2. Alakítsd át a kandallóba szorult egeret úgy, hogy mindig jó irányba álljon.

3. Készítsünk egy autót felülnézetből, melynek a méret 40x40 képpont legyen, majd készítsük el mind a 4 fő irányba elforgatott alakját.

4. Készítsük el egy megnövő és leeresztő léggömb animációját.

Program vezérlés

Az Imagine LOGO többféle programvezérlő utasítással rendelkezik. Ezeket három csoportba soroljuk:

1. Szekvencia

A parancsok végrehajtásának alapvető sorrendje a balról jobbra haladás, illetve a fentről lefelé való végrehajtás.

Ezek közé tartoznak az eddig megtanult parancsok:

Imagine LOGO	
parancs	**rövidítés**
előre	e
hátra	h
jobbra	j
balra	b
tollatle	tl
tollatfel	tf
tollszín	tsz
tollszín!	tsz!
tölt	
tollvastagság	tv
tollvastagság!	tv!
törölképernyő	törölkép
betölthátterkép	bethp
figyelj, aktív	

haza	
irány	
irány!	
írólapablak	ía
kér	
kiír	
elrejtteknőc	rejttek, elrejt
látható	
rajzlapablak	ra
eljárás	
törölháttérkép	törölhkép
törölképernyő	törölkép
törölobjektum	
várj	

Mielőtt ismertetnénk az egyes elágazási szerkezeteket, azelőtt röviden nézzük meg a feltételek megadásánál alkalmazható logikai műveleteket!

Logikai műveletek

Tagadás a **nem** paranccsal végezhető, a konjunkcióra az **és**, diszjunkcióraa **vagy**művelet áll rendelkezésre. Ezeken kívül a **kizárólagy** - röviden **xor** - is használható.

Nézzünk meg néhány egyszerű példát!

```
? mutat nem "hamis
igaz
? mutat (vagy "igaz "hamis)
igaz
```

```
? mutat (és "igaz "hamis)
hamis
? mutat (kizáróvagy "igaz "hamis)
igaz
? mutat (kizáróvagy nem "hamis "igaz)
hamis
? mutat (vagy nem "igaz "hamis)
hamis
? mutat (és nem "hamis nem "hamis)
igaz
```

Vegyük észre, hogy az **IGAZ** logikai értéket az "**igaz** szókonstans, a **HAMIS** logikai értéket a "**hamis** szókonstanssal tudjuk megadni.

A műveletek természetesen egymásba is ágyazhatóak, és több operandussal is elvégezhetőek, például:

```
? mutat (és "igaz (vagy "igaz nem (és "hamis "igaz)))
igaz
? mutat (kizáróvagy "igaz "igaz "igaz)
Igaz
```

2. Szelekció

A másképpen *elágazási* vagy *kiválasztási* szerkezeteknek nevezett vezérlő szerkezetek. Az elágazásoknak több különböző formáját használhatjuk aszerint, hogy mennyi kimeneti értéket szeretnénk.

Egyágú kiválasztás

A**ha** szerkezet első bemenete egy logikai értéket visszaadó kifejezés, a *feltétel,* amely ha **IGAZ** lesz, akkor a második bemenet utasításait végrehajtja, ha a *feltétel* **HAMIS**, akkor nincs hatása a műveletnek.

```
hafeltétel [utasításlista]
```

Nézzünk két egyszerű példát:

```
? ha 5>4 [mutat "helyes]
helyes
? ha 5=4 [mutat "helyes]
```

Utóbbi esetben nincs kiírás!

Kétágú kiválasztás

A **hakülönben** - röviden **hak** - szerkezet első bemenete egy logikai értéket visszaadó kifejezés, a *feltétel,* ami ha **IGAZ**, akkor a második bemenetének megadott utasításlistát hajtja végre, ha **HAMIS**, akkor a harmadik bemeneteként megadott utasításlistát. Fontos, hogy mindkét ágnak szerepelnie kell!

```
hakülönben feltétel
[utasításlista igaz esetén]
[utasításlista hamis esetén]
```

Nézzünk két egyszerű példát:

```
? mutathak 5>4 ["helyes][“helytelen]
helyes
? mutathak 5=4 ["helyes][“helytelen]
helytelen
```

Lehetőségünk van arra is, hogy az elágazási feltételt külön értékeljük ki, majd adjuk meg, hogy mi történjen ha igaz, vagy ha hamis. Ezt a megoldást teszi lehetővé a **teszt**,**halgaz**, **haHamis** parancshármas - röviden **hai** és **hah**. E szerkezet egy eljáráson belül használható, és nyilvánvalóan a **teszt** parancsnak meg kell előznie a **halgaz** és **haHamis** parancsokat, amelyekből több is lehet az eljáráson belül. Ez utóbbi gyakorlatilag azt jelenti, hogy az elágazási feltételt adó kifejezés többszöri kiértékelését lehet elkerülni, illetve mivel nem kötelező mindkét ág, így nem határozzuk meg előre, hogy egyvagy kétágú kiválasztást akarunk-e megvalósítani.

Példaként nézzük meg az egyszerű értékvizsgálatot bemutató **jellemez** eljárást.

```
eljárás milyen :szám
teszt :szám >= 0
halgaz [mutat "JA szám pozitívj ]
haHamis [mutat "|A szám negatív|]
halgaz [mutat "|vagy nulla|]
vége
```

Hívjuk meg az eljárást:

```
? milyen 8
A szám pozitív
vagy nulla
? milyen -2
```

```
A szám negatív
```

Amint látható, az igaz ág részekre bontható lett.

Többágú kiválasztás

A többágú kiválasztás az **elágazás** paranccsal valósítható meg. Itt a feltételként megadott *kifejezésnek* egy szóval kell visszatérnie, amelynek a lehetséges értékeit soroljuk fel az egyes ágak feltételeként. Fontos, hogy az értékek nem kerülnek kiértékelésre, azaz csak szókonstansok használhatóak esetként.

Az egyes ágak egymás után kerülnek vizsgálatra, és azt az ágat hajtja csak végre, amelynél először teljesül az egyezés. Ha nem adunk meg az utolsó ág előtt feltételt, akkor az hajtódik végre, mint különben ág. Ha ilyen nincs, és egyik ág esetei sem adtak egyezést, akkor nincs hatása a szerkezetnek.

```
elágazáskifejezés [
eset1 [utasításlista]
eset2 [utasitaslista]
eset3 [utasításlista]
[utasításlista egyébként(elhagyható) ]]
```

Például, ha egy beolvasott jelről szeretnénk eldönteni, hogy ékezetes vagy ékezet nélküli magánhangzó, vagy esetleg egyéb karakter, akkor ezt a következő módon tehetjük meg egy eljárásban megvalósítva.

```
eljárás magánhangzó
mutat "|Nyomj le egy billentyűt!|
lokért "jel olvasjel
(kiírérték "|A(z) | :jel "| |)
mutat elágazás :jel [
a e i o u ["|ékezet nélküli.)]
á é í ó ö ő ú ü ű ["ékezetes.]
["|egyéb jel.|]
vége
```

Hívjuk meg rendre a **magánhangzó** eljárásunkat az u, ö és a p megadásával:

```
? magánhangzó
Nyomj le egy billentyűt!
A(z) u ékezet nélküli.
? magánhangzó
Nyomj le egy billentyűt!
A(z) ö ékezetes.
```

```
? magánhangzó
Nyomj le egy billentyűt!
A(z) p egyéb jel.
```

Feladat: Alakítsuk át úgy a **magánhangzó** *eljárást úgy, hogy az a mássalhangzó esetén is helyes kiírást adjon!*

3. Iteráció

Az iterációnak, azaz az ismétlésnek számos formája létezik

Elsőként a mindenki által jól ismert, **ismétlés** parancs új arcát mutatjuk meg.

a.) Számlálásos ciklus, ciklusváltozó nélkül

```
ismétlésismétlésszám [utasításlista]
```

Az előre meghatározott számú ismétlés elvégzésére az **ismétlés** parancsot használhatjuk - röviden **ism.** A címből sokan talán rá sem ismernek, mivel az nem is teljesen pontos. A ciklus ugyanis rendelkezik ciklusszámlálóval, de ezt nem nevesíti, és a **hányadik** paranccsal lekérdezhető az értéke.

A működése jobb megértéséhez érdemes megfigyelni a következő utasításokat.

```
? ismétlés 1 [mutat "alma]
alma
? ismétlés 3 [mutat "alma]
alma
alma
alma
? ismétlés 2.5 [mutat "alma]
alma
alma
alma
? ismétlés 2.49999 [mutat "alma]
alma
alma
```

A számlálásos ciklusban alkalmazzuk ezután a hányadik eljárást is, ami tehát a ciklusszámláló szerepét tölti be, így 1-el indul, ha egyáltalán történik végrehajtás legalább egyszer.

```
? ismétlés 2.4 [kiír hányadik]
```

```
1
2
? ismétlés 2.5 [kiír hányadik]
1
2
3
```

De lássuk, hogy is lehet ezt kombinálni!

```
? ismétlés 2 [kiír hányadik "alma]
1
2
? ismétlés -1 [kiír hányadik "alma]
? ismétlés 0 [kiír hányadik "alma]
? ismétlés 0.5 [kiír hányadik "alma]
1 "alma
```

Az **ismétlés** parancsok egymásba is ágyazhatóak, de ekkor a **hányadik** parancs működése nehezebben nyomon követhető!

```
? ismétlés 3 [(kiírérték hányadik "| |) ismétlés 2 [kiír hányadik]]
1 1
2
2 1
2
3 1
2
```

Mint a példákból is látszik, ebben az esetben a **hányadik** mind a külső, mind a belső ciklus végrehajtásszámát megadja, azaz úgy viselkedik, mintha mind a kettőnek külön-külön ciklusszámlálója lenne.

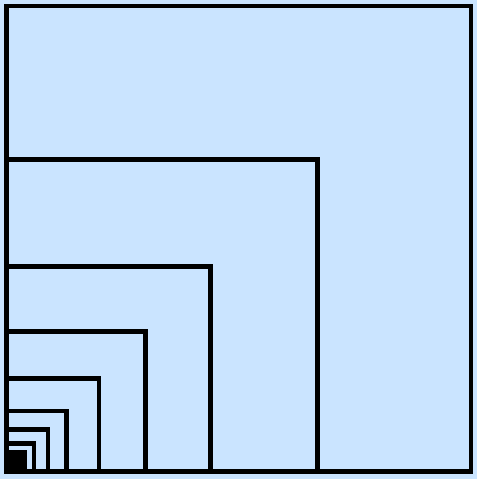
Rekurzió

A rekurzív problémamegoldás lényege, hogy a feladatot kisebb, az eredetihez hasonló részfeladatokra bontjuk, és azok (szintén rekurzív módon megkapott) megoldásából állítjuk össze az eredeti feladat megoldását. Ahhoz, hogy módszerünk eredményt adjon, kell egy olyan szint, ahol már nem kell tovább bontani a problémát, hanem az már direkt módszerrel megoldható.

A rekurzív eljárások olyan eljárások, amelyek meghívják önmagukat. Ha el akarjuk kerülni a végtelen ciklust, paraméterezzük a rekurzív eljárásokat, és a rekurzív hívást egy feltételtől tegyük függővé.

A rekurzív ábrák részként tartalmaznak egy, az egészhez hasonló alakzatot.

A paraméterezett négyzet eljárást alapul véve készítsük el az alábbi rajzot, melyben a négyzetek oldala egyre kisebb lesz.



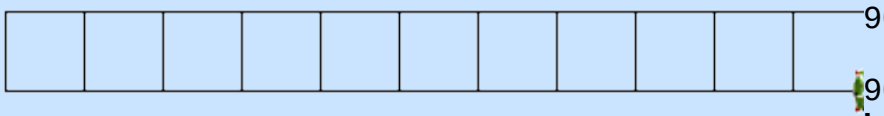
```
eljárás reknégyzet :h
ism 4 [e :h j 90]
ha :h>1 [reknégyzet
:h/1.5]
vége
```

A :h jelöli az oldal hosszát, amit az eljárás újrahívásakor másfélszeresére növelünk.

Módosítsd az eljárást úgy, hogy egy új változó a *:per* segítségével paraméterezve határozod meg a négyzet oldalméretének csökkenését.

(Megoldás)

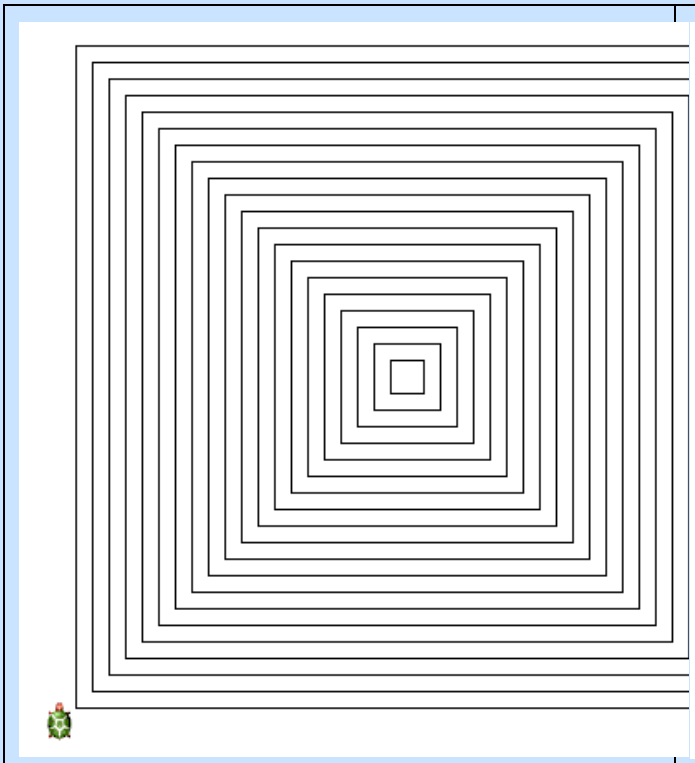
Ha egy alakzatot :n-szer akarunk kirajzolni, a rekurzió paraméterével tudjuk :n értékét szabályozni.



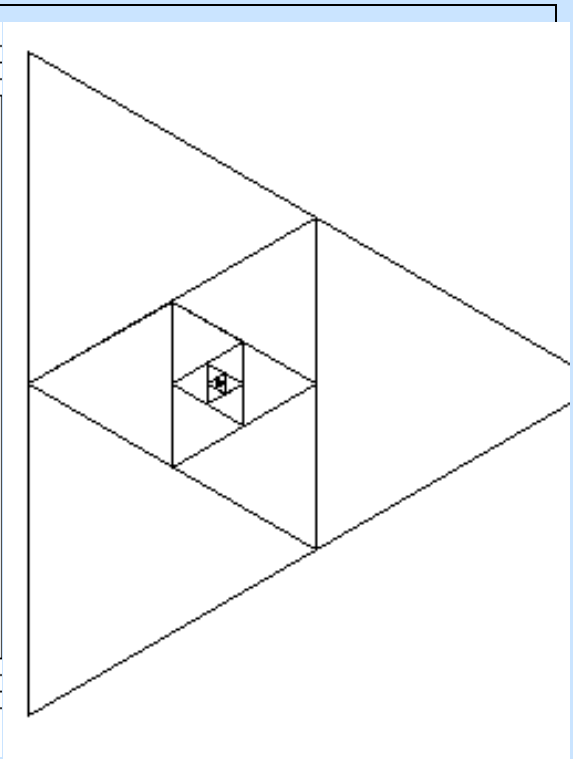
```
eljárás sornégyzet :n :h
ism 4 [ e :h j 90 ]
    j 90 e :h b 90
ha :n > 0 [sornégyzet :n-1 :h]
vége
```

Gyakorló feladatok:

1-2.

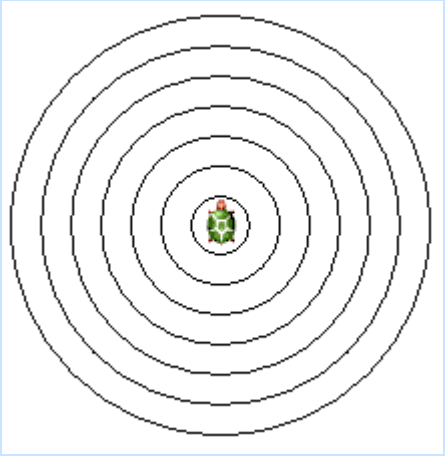


```
eljárás reknégyzet2 :h :db
 ha :db > 0 [
ism 4 [e :h j 90]
tf h 10 j 90 h 10 b 90 tl
   reknégyzet2 :h+20 :db-1 ]
vége
```



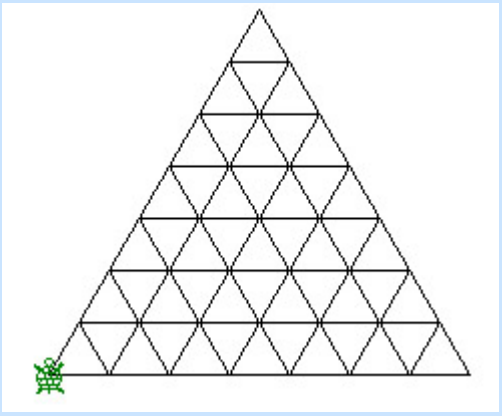
```
eljárás rekháromszög2 :h :db
 ha :db > 0 [
ism 3 [e :h j 120]
   e :h/2  j 60
   rekháromszög2 :h/2 :db-1 ]
vége
```

3. A céltábla elkészítéséhez a középpontból rajzolt :r sugarú kör eljárást használj! Ha elkészültél, színezd ki fehérrel és feketével, ahogyan azt az igazi céltáblán is láthatod!



CÉLTÁBLA :N :R :D
:N a körök száma, :R a legkisebb kör sugara, :D a szomszédos körök távolsága.

4. Vedd észre, hogy a piramis megrajzolásához nem kell minden látható háromszöget elkészítened!



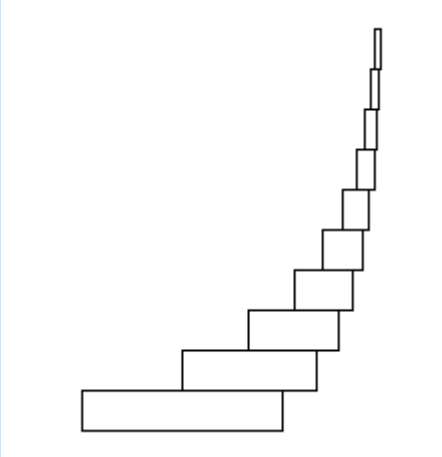
PIRAMIS :N :H
:N az emeletek száma, :H egy kisháromszög oldalának hossza

5. Készíts rekurzív (**lépcső :n :h :m** módon hívható) programot az alábbi ábra kirajzolásához!

A paraméterek jelentése:

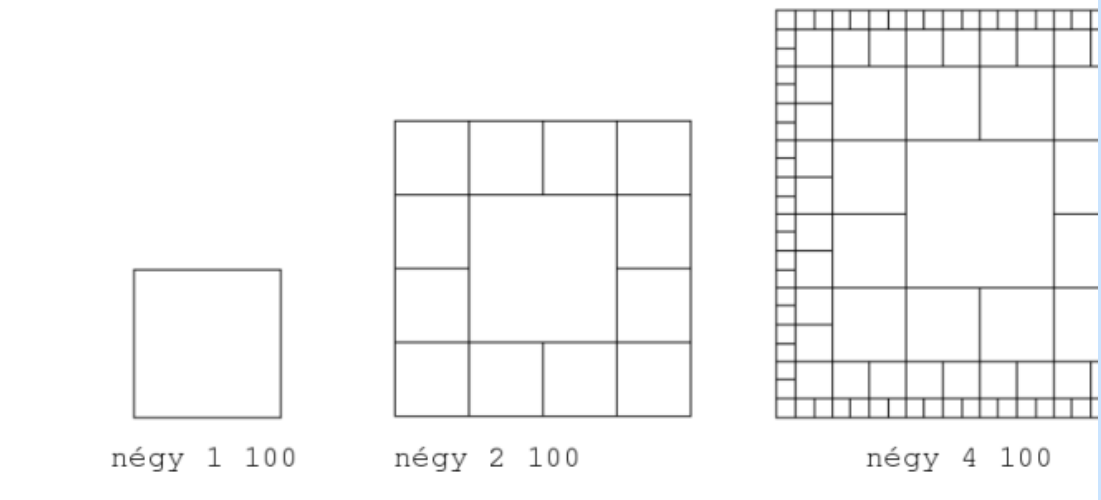
- **:n** a lépcsőfokok száma
- **:h** az első lépcsőfok hossza
- **:m** a lépcsőfokok magassága

Az új lépcsőfok mindig az előző felénél indul, és hossza az előző kétharmada.



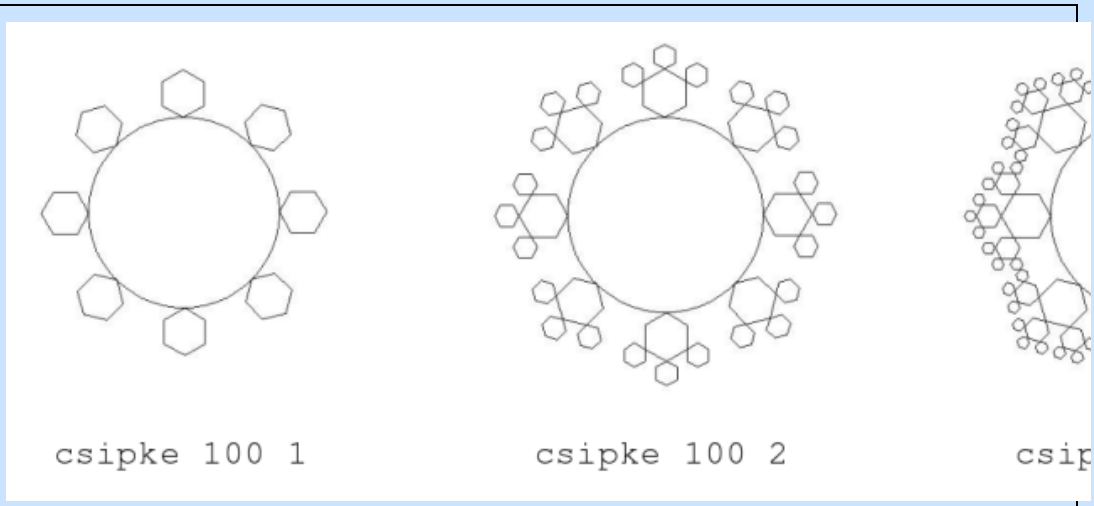
6. Készíts Logo eljárást (**négy :n :h**), amely négyszögeket rajzol egymás köré!

. Legyen **:n** darab négyzetsor egymás körül, a legbelső négyzet oldalhossza legyen **:h**, kifelé haladva a négyzete



k oldalhossza feleződjön!

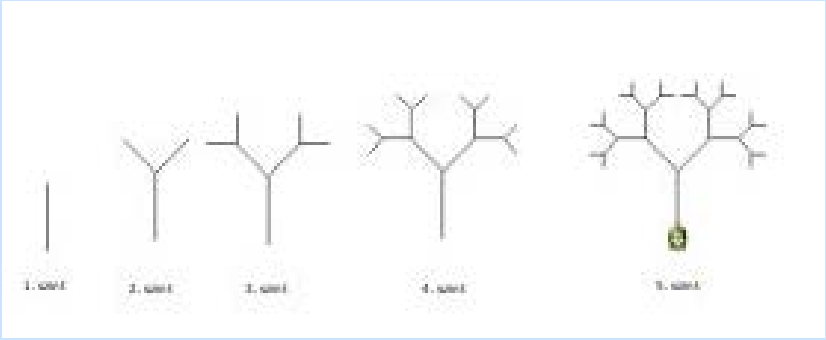
7. Készítsd el a következő **csipke :méret :sordb** eljárást, amely egy nyolcas szimmetriájú csipke terítőt rajzol:

amely**:sordb** sorból áll és az egyes csipkék mérete **:méret**. A csipke darabok hatszögekből állnak, és a következő sor, csak a hatszög „külső” három csúcsára illeszkedik.	

Rekurzív görbék - fraktálok

A fraktálok olyan alakzatok, melyek önhasonlóak, vagyis kisebb részük kinagyítva olyan, mint az eredeti. Ilyen például a természetben a fa is. Az alábbi kis rekurzív program egy ilyen fát rajzol. Látható az eljárásban, hogy mikor elágazáshoz érünk, akkor a `fa` eljárás az eredeti hossz 2/3 részével mindkét irányba újra meghívásra kerül, hiszen onnan és azokban az irányokba fog tovább nőni a fa. A szintszámot ilyenkor csökkentjük, és mikor az összes szintnek megfelelő farész is megrajzolásra került, befejeződik az eljárás.

```
Eljárás fa :szint :hossz
  e :hossz
  ha :szint>1
    [j 45 fa :szint-1 2/3*:hossz
     b 90 fa :szint-1
2/3*:hossz
     j 45 ]
  h :hossz
vége
```



Nevezetes görbék:

Koch görbe: Ezt a fraktálszerű görbét *Helge von Koch* svéd matematikus 1906-ban írta le először. A Koch-görbe előállítási lépései a következők:

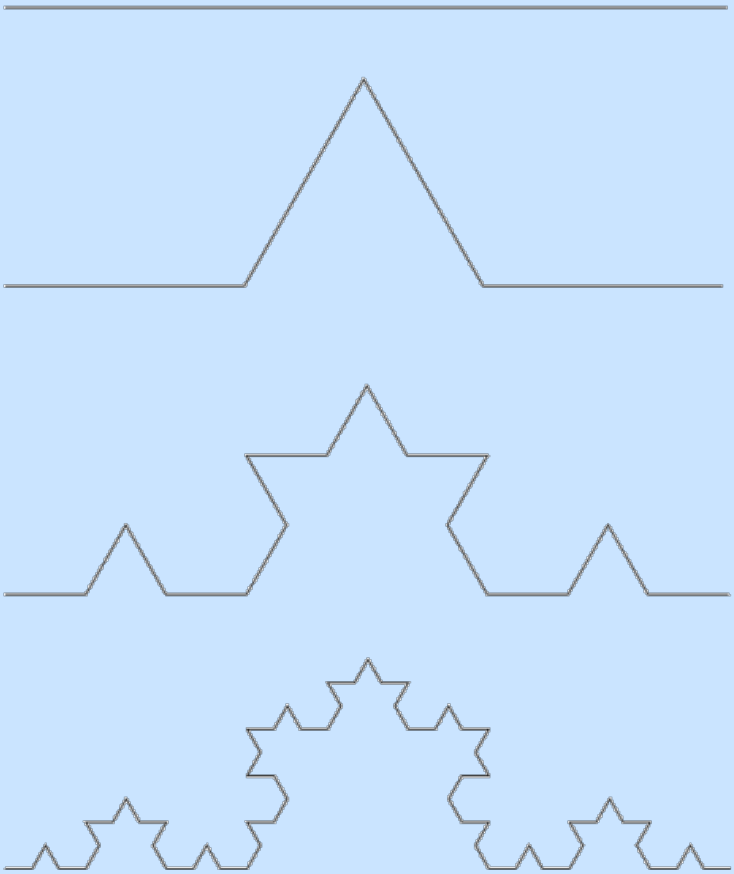
0. lépés: vegyünk egy szakaszt

1. lépés: a középső egyharmadát vágjuk ki, és emeljünk helyette "sátrat" két, egyharmad hosszú szakaszból, mintha azok egy szabályos háromszög két oldalát alkotnák

2. lépés: a keletkezett négy szakasszal tegyük ugyanezt

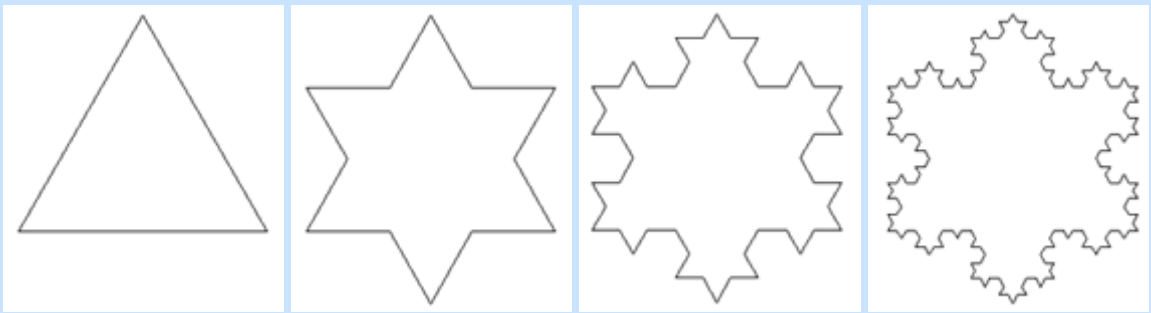
3. lépés: a keletkezett összes szakasszal tegyük ugyanezt

És így tovább. Amit végtelen lépés után kapnánk, az a Koch-görbe.

0. szint 1. szint 2. szint 3. szint		eljárás koch :h :szint HAK :szint = 1 [E :h] [koch :h/3 :szint-1 B 60 koch :h/3 :szint-1 J 120 koch :h/3 :szint-1 B 60 koch :h/3 :szint-1] vége

KOCH-HÓPEHELY

A Koch-hópehely vagy más néven Koch-sziget három darab Koch-görbéből áll, melyek szabályos háromszög-alakban kapcsolódnak egymáshoz. Tehát az előző Koch eljárást felhasználva Te is könnyedén megírhatod azt a kochpehely eljárást, melynek két paramétere ugyancsak a:lépés és a :hossz, és megrajzoltatja a három darab görbét a megfelelő szöggel elfordulva közöttük. Íme az eredmény:



kochpehely 0 200 **kochpehely 1 200** **kochpehely 2 200** **kochpehely 3 200**

Sierpinski háromszög

Ezt a fraktált 1916-ban mutatta be WaclawSierpinski lengyel matematikus. A Sierpinski-háromszög az eddigiekkel ellentétben nem egy fraktálgörbe, hanem olyan fraktál, melynek előállításakor szakasz helyett egy egyszerű síkidomból, egy szabályos háromszögből kell kiindulnunk. Az előállítási szabály ezúttal sem bonyolult:

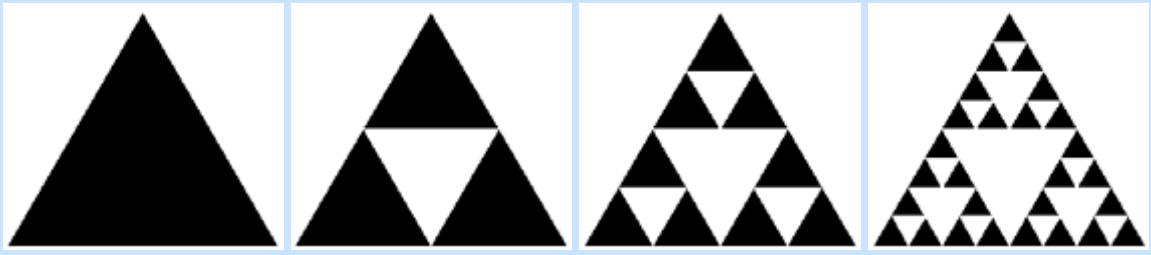
0. lépés: vegyünk egy szabályos háromszöglapot

1. lépés: kössük össze a háromszög oldalainak felezőpontjait egymással, így a háromszöget négy kisebb háromszögre osztottuk. A középső, csúcsára állított háromszöget vágjuk ki.

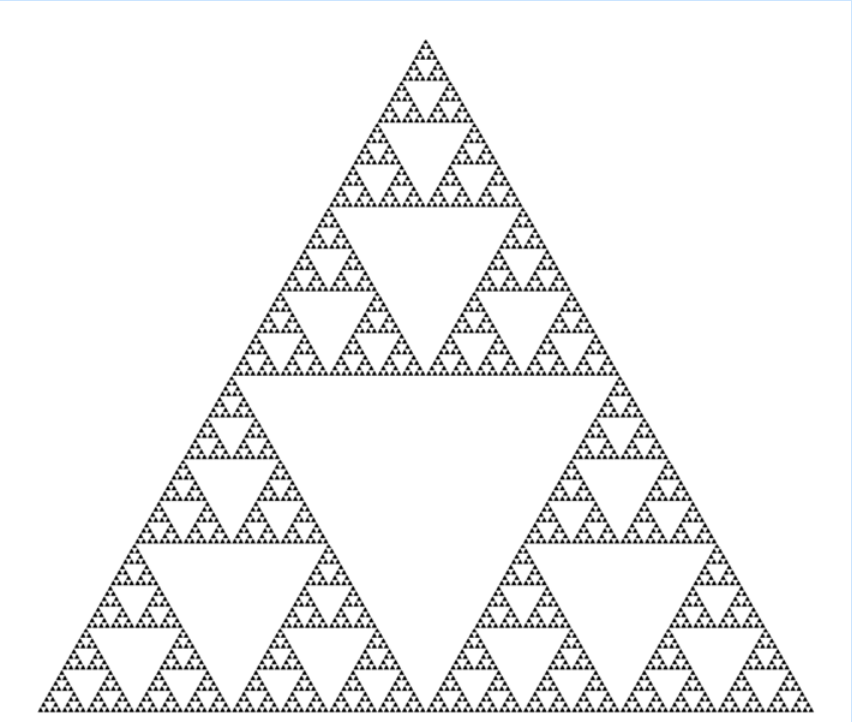
2. lépés: a maradék három kis háromszögnek ugyanígy vágjuk ki a közepét.

3. lépés: a keletkezett összes háromszöggel tegyük ugyanezt.

És így tovább. Amit végtelen lépés után kapunk, az a Sierpinski-háromszög.



0. lépés **1. lépés** **2. lépés** **3. lépés**



LOGO-ban ez az eljárás kicsit bonyolultabb, mint az eddigiek. Az alapötlet az, hogy rajzolunk egy fekete háromszöget, majd írunk egy rekurzív eljárást, mely egy-egy háromszög közepébe rajzol egy fehér, fejjel lefelé álló háromszöget. Ezen kívül lesz egy segédeljárásunk, mely egy adott színű háromszöget tud rajzolni.

Lássuk először ezt a segédeljárást. Nem tesz mást, mint rajzol egy szabályos háromszöget és kiszínezi azt. A színezéshez felemelt tollal kell bemennünk a háromszög belsejébe, majd a színezés után visszatérni a háromszög sarkába.

```
eljárás háromszög :hossz :szín
ismétlés 3 [előre :hossz jobbra 120]
töltőszín! :szín tollatfel jobbra 30 előre 3
tölt hátra 3 balra 30 tollatle
vége
```

Most nézzük azt eljárást, mely egyetlen fehér, fejjel lefelé álló háromszög megrajzolásáért felelős. Hatására a teknőc a nagy háromszög bal alsó sarkából indulva, a tollat felemelve elmegy az alap feléig, ott helyes irányba fordul, majd alsó csúcsától kezdve felfelé megrajzolja a fehér háromszöget, végül a tollat felemelve visszamegy a nagy háromszög bal alsó csúcsába:

```
eljáráskivágháromszög :kishossz
tollatfel jobbra 90 előre :kishossz balra 120 tollatle
háromszög :kishossz 15
```

```
tollatfel balra 60 előre :kishossz jobbra 90
vége
```

A háromszög eljárással fehér színű háromszöget szeretnénk rajzoltatni, ezért az eljárásnak paraméterként a 15-öt adjuk, mely a fehér szín kódja.

Most már jöhet az igazi rekurzív eljárás, mely minden szükséges fehér háromszöget megrajzol (egy-egy lépésben összesen hármat):

```
eljárássierpháromszög :lépés :hossz
ha :lépés > 0 [kivágháromszög :hossz
ismétlés 3 [sierpháromszög :lépés - 1 :hossz / 2
tollatfel balra 90 hátra :hossz * 2 tollatle balra
30]]
vége
```

Nincs más teendőnk, minthogy megírjuk a főeljárást, mely rajzol egy nagy fekete háromszöget, majd meghívja a fehér háromszögeket rajzoló eljárást (a tollszínt először feketére, majd fehérre, majd újból feketére váltja, hogy a háromszögek körvonalai is megfelelő színűek legyenek.):

```
eljárásfősierpháromszög :lépés :hossz
tollszín! 0 jobbra 30 háromszög :hossz*2 0 balra 30
tollszín! 15 sierpháromszög :lépés :hossz
tollszín! 0
vége
```

SIERPINSKI-SZŐNYEG

A Sierpinski-szőnyeg szintén síkbeli fraktál, előállítási ötlete ugyanaz, mint a Sierpinski-háromszögé, mivel megrajzolása több ismétlést és rekurziót igényel, a részletes eljárásokat nem ismertetjük. Bemutatjuk viszont az előállítás szabályát, és az első néhány lépés után a fraktál kinézetét. Az előállítás lépései:

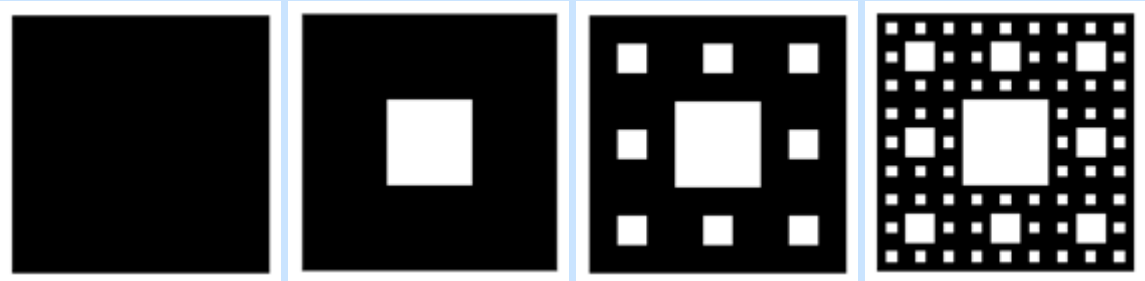
0. lépés: vegyünk egy négyzetet

1. lépés: kössük össze a négyzet oldalainak harmadolópontjait a szemközti harmadolópontokkal, így a négyzetet kilenc kisebb négyzetre osztottuk. A középső négyzetet vágjuk ki.

2. lépés: a maradék nyolc négyzetnek ugyanígy vágjuk ki a közepét.

3. lépés: a keletkezett összes négyzettel tegyük ugyanezt.

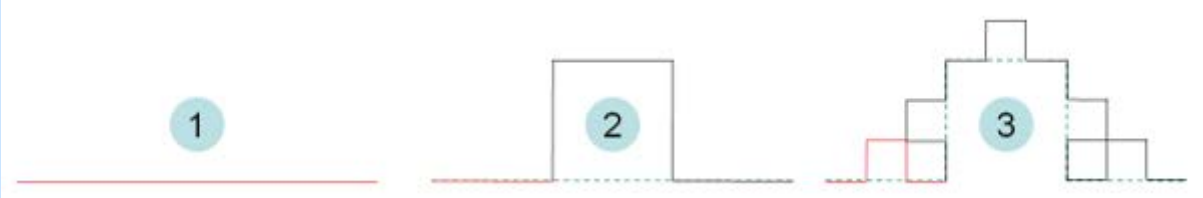
És így tovább. Amit végtelen lépés után kapnánk, az a Sierpinski-szőnyeg.



0. lépés **1. lépés** **2. lépés** **3. lépés**

Feladatok:

A Koch görbe eljárást alapján próbáld elkészíteni következő ábrát



Találj ki saját fraktálokat is! Találj egy olyan alakzatot, amire a szakaszokat minden lépésben kicseréled! Legjobb, ha papíron megrajzolod az első 1-2 lépést és utána írsz rá eljárást. Válaszd ki hozzá kedvenc tollszínedet!

Programkészítés alapjai

Az eddigi ismereteink alapján már használt **ismétlés**és **ha** utasítás mellett az Imagine számos más vezérlési szerkezetet is tartalmaz. Ezek alapos megismeréséhez azonban szükségünk lehet néhány új fogalom bevezetésére, új parancs megismerésére.

VÁLTOZÓ LÉTREHOZÁSA, ÉRTÉKADÁS

Az Imagine már tartalmazza a globális és lokális változó használatának lehetőségét.

Globális változók

A **globálisváltozó** - vagy röviden **globvál**- valamint a **név** paranccsal globális változó hozható létre. Az így létrehozott változók a program minden részében elérhetőek lesznek, de természetesen ha azonos nevű lokális változó is van egy adott eljáráson belül, akkor az általános szabályt kell alkalmaznunk. Erre később az eljárásoknál is látunk példát.

```
? név 15 "labda
? globvál "egérke 7
? mutat :labda
15
? mutat :egérke
7
```

Lokális változók

A lokális változó létrehozására szolgál a **lokálisváltozó** parancs – röviden **lokvál.** Ez csak magát a változót hozza létre, de értéket még nem ad neki.

```
? lokálisváltozó "egérke
? lokálisváltozó [erre arra]
? mutat :egérke
A(z) egérke változónak nincs értéke
Használd a NÉV parancsot a változó értékének megadásához!
```

A változó neve előtt " idézőjel szerepel, a : kettőspont pedig az értékének használatakor. Ez azért is fontos, mert az ***"egérke*** név vagy cím szerinti változó meghívást jelent, ahol a változó értéke módosulhat, míg az ***:egérke***érték szerinti meghívás, a változó értékét kiolvassuk, de nem módosítható. Értéket adni a már létrehozott változónak a **név** paranccsal is lehet.

```
? név 8 "egérke
? mutat :egérke
```

```
8
```

Figyeljünk arra, hogy a név paranccsal az értékadás használata nem a megszokott sorrendű, így elsőként az érték, utána a változó áll. Ha fordítva adnánk meg a sorrendet, akkor érdekes dolog alakul ki:

```
? név "egérke 8
? mutat :egérke
A(z) egérke változónak nincs értéke
Használd a NÉV parancsot a változó értékének megadásához!
```

Létrehoztunk viszont egy új, globális változót 8 néven, amit ellenőrizhetünk is. Az alábbi második parancs mutatja a különbséget is.

```
? mutat :8
egérke
? mutat "8
8
```

A lokális változók használata esetén célszerűbb azonnal az értékadást is ehetővé tevő **lokálisérték** - röviden **lókért** - parancsot használni.

```
? lokálisérték "labda 4
? mutat :labda
4
```

A lokális csak az adott eljárásban, míg a globális változó mindenhol látható. A létrehozott változókkal végezhető műveletek közül a **növel** igen hasznos. Ez alapértelmezetten 1-gyel való növelés, de megadhatunk növekményt is.

```
mutat :labda
15
? növel "labda
? mutat :labda
11
? növel "labda 2
Nem tudom mit csináljak a(z) 2-val
Nem mondtad meg, mit csináljak az eredménnyel.
```

hibaüzenet

A hiba oka, hogy alapértelmezetten csak a változó a **növel** bemenete, ezért a hiba a **növel** feldolgozása után lép fel, maga az értéknövelés megtörténik!

```
mutat :labda

12
```

A növekmény megadásánál ki kell jelölni a hatókört.

```
? (növel "labda 2)

? mutat :labda

14

? (növel "labda -5)

? mutat :labda
```

Teljesen hasonlóan használható a **csökkent** parancs is.

Próbáld ki!

ADAT BE- ÉS KIVITEL

Kiírás

A változók írólapon való megjelenítésére már használtuk a **mutat** parancsot. Ha több elemből áll, amit ki akarunk íratni, akkor hatókört kell kijelölni a zárójelek használatával.

```
? (mutat "|labda értéke:| :labda "|, egérke értéke:|
:egérke)

labda értéke: 12 , egérke értéke: 8

? mutat "Bendegúz mutat [a b c]

Bendegúz

[a b c]
```

Vegyük észre, hogy a **mutat** parancs a kiírt elemek közé automatikusan szóközt helyez el, ami felhasználói szempontból néha felesleges. A kiírás után új sor elejére teszi a kurzort, valamint a listát listaként jeleníti meg [] jelek között.

Létezik egy általánosabban használható parancs a **kiír** – röviden **ki** - parancs. Ez is átirányítható megjelenítést tesz lehetővé, azaz nem csak az írólapon, hanem a megadott kiíró eszközön is tud adatot megjeleníteni, viszont a listát szögletes zárójel nélkül adja vissza, különben használata megegyezik a **mutat** paranccsal.

```
? (kiír "|labda értéke:| :labda "|, egérke értéke:| :egérke)

labda értéke: 12 , egérke értéke: 8

? kiír "Lajos kiír [a b c]

Lajos
```

```
a b c
```

A **kiírérték**parancs a kiírt bemeneteket nem választja el szóközökkel, és a kurzort sem viszi a következő sorba, azaz alkalmas formázottabb kiírásokra, valamint átirányítás esetén is használható.

```
? (kiírérték "|labda értéke:| :labda "|, egérke értéke:|
:egérke)

labda értéke:l2, egérke értéke:8

kiírérték "Bendegúz kiírérték [a b c]

Bendegúza b c
```

Az írólapra kerülő kiírásokat a törölszöveg – röviden törölszöv – paranccsal tudjuk eltüntetni.

Adatok a programban

Minden programozási nyelv alapvető építőkövei az alkalmazható adattípusok és a rajtuk értelmezett műveletek. A LOGO nyelv elemi adata a karakter. A karakterekből előállhat például *szó,*ami értelmezhető *egész* vagy *valós számként,* illetve építhető belőle *mondat* vagy *lista.*

Egyszerű adatobjektumok, primitívek

Szó

A szó karakterek sorozata. A parancsszavaktól és egyéb elemektől megkülönböztetendő a szó írása " idézőjellel történik, ha önállóan áll. Így például *szó* lesz a ***"akác,*** de a ***"2.Vilmos***is. Néhány speciális karakter nem szerepelhet a szóban egyszerűen beírva, így a \ és / jel, vagy a műveleti jelek sem írhatók be. Éppen ezek miatt az Imagine-ben megalkották az *általános szóírási módot.* Ekkor a szó karaktereit | | jelek közé írjuk, és így például már szóköz is lehet a szó része, például a ***"|12/B tanulói*** | egy szónak számít. Célszerű ezt a szóírásmódot választanunk mindannyiszor, amikor betűkön kívül más jeleket is írnunk kell, ill. ha a szóközt akarjuk használni. Az üres szó alakja: **"|**

A **szó** parancs is az ImagineLOGO-ban.

Figyeljük meg, hogyan lesz a megadott elemekből egyetlen, felesleges szóköz nélküli kiírás!

```
? ciklusegyenként "i [kék sárga zöld] [(kiír szó hányadik ".
"szó elemszám :i "betűből "áll)]
1. szó 3 betűből áll
2. sző 5 betűből áll
3. szó 4 betűből áll
```

A **szó?** parancs viszont annak a megállapítására szolgál, hogy a kérdéses jelsorozat szó-e.

```
? mutat szó? "betűk
igaz
? mutat szó? 12356
igaz
? mutat szó? [Vilmos 2]
hamis
```

Mint látható a 12356 is szó!

Feladat:

1. Azonos jelsornak számít-e a "matrac és a "|matrac| karaktersorozat?

2. Vizsgáljuk meg, hogy a következő jelsorozatok szavak-e? Használjuk erre a **szó?** *parancsot!*

a) teki
b) "9
c) |7 csoda|
d) "|Holnap kirándulni megyünk a hegyekbe!|

A karakterekből alkotott nem üres szavak speciális sorozata a *szám.* A szám alapvetően számjegyek sorozata, például a 123, de szerepelhet a tizedespont és az előjel karakter is a tizedesek elválasztására és az előjel jelölésére: -34.6 is szám lesz. Ezen kívül lehet betű is a szám része, mert például szám a 3.5e-4 jelsor: normál alakban megadott.

Ne felejtsük el, hogy az Imagine számalakjában tizedespont van, és nem tizedesvessző!

3. Állapítsuk meg, hogy a következő jelsorozatok számok-e?

a)+6
b)-6.6E6
c) 6E+6
d) 3.4e5.6
e) 6,78

A **szám?**parancs a bemenetként megadott szóról állapítja meg, hogy értelmezhető-e a számként.

```
? mutat szám? 12356
igaz
? mutat szám? "betűk
hamis
? mutat szám? "123,56
hamis
```

Vegyük észre, hogy az utóbbi esetben a tizedesvessző okozza a hibát! A törtrész elválasztására a többi ország pontot használ, a vessző használatának következményét sajnos el kell fogadnunk, és nagyon figyelnünk kell rá.

Műveletek szavakkal

A *szó* tehát az általános adatobjektum, amelynek egy részhalmazát alkotják csak a *számok.* E sajátosságra tekintettel kell lennünk, ha műveleteket végzünk a számokkal. Elsőre például meglepő lehet, de a fentiekből következik, hogy a

```
? mutat 8 + első "|7. sor|
15
```

eredmény logikus, mivel a szó első karaktere számként is értelmezhető. Az alkalmazott **első**parancs a szó első karakterét adja vissza. Értelemszerűen az **utolsó**parancs az utolsó karaktert jeleníti meg.

```
? mutat első "labda
l
? mutat első "|

? mutat utolsó "labda
a
```

Mint látható az **első**parancs számára nem okoz hibát, ha az üres szóra alkalmazzuk, az üres karaktert adja vissza. A nem üres szóra alkalmazható még az **elsőnélküli**, **utolsónélküli**– röviden **en**és **un**- parancs is. Ezek a szó első vagy utolsó karakterét elhagyják.

```
? mutatelsőnélküli "borsó
orsó
? mutat utolsónélküli "borsó
bors
```

Az **első**paranccsal kombinálva hozzáférhetünk a szó második karakteréhez is.

```
? mutat első elsőnélküli "mogyoró
o
? mutat utolsó utolsónélküli "mogyoró
r
```

Az **elemnélküli**és az **elemsorszámnélküli**parancsok már a szó belsejében is ki tudják fejteni hatásukat. Mindkettő két bemenetű. Az **elemsorszámnélküli**- röviden **elemsn**- esetében az első bemenet egy szám, a második egy szó. A művelet eredménye egy olyan szó lesz, amely az eredeti szóból a szám helyén lévő karaktert nem tartalmazza.

```
? mutatelemsorszámnélküli 4 "körte
köre
```

Az **elemnélküli**első bemenete egy karakter, a második egy szó. A művelet eredménye egy olyan szó lesz, amelyből hiányzik az összes megadott karakter.

```
? mutat elemnélküli "e "elemes
lms
```

Ha a szóban nem fordul elő a megadott karakter, akkor természetesen a szó nem változik meg!

4. Egy általános szóírási móddal megadott szóból szedjük ki az esetlegesen előforduló szóközöket!

```
? ki elemnélküli "| "|el kel káposztásított alananított
átok|
elkelkáposztásítottalanítottátok
```

Igen hasznos lehet a két szó bemenetes **elemtől**parancs. A parancs balról megkeresi az első bemenet első előfordulását a második bemenetben, és a második bemenetnek azzal a részével tér vissza, amely attól az előfordulástól kezdődik. Ha az nem található meg benne, akkor az üres szóval tér vissza.

```
? mutat elemtől "me "mamut

? mutat elemtől "ma "mamut
```

```
mamut
? mutat elemtől "mu "mamut
mut
```

Az ugyancsak két bemenetű **elsőnek**és **utolsónak**parancsokkal az első bemenetként megadott szót illeszthetjük a második bemenetként megadott szó elé vagy után.

```
? mutat elsőnek "kör "lap
körlap
? mutat utolsónak "kör "lap
lapkör
```

A **szó**paranccsal hasonló eredményt érhetünk el, és ezt több bemenettel is használhatjuk. Ekkor viszont ki kell jelölni a parancs hatókörét.

```
? mutat (szó "kör "lap "szél)
körlapszél
```

Az **üres?** paranccsal megállapíthatjuk, hogy egy szó tartalmaz-e karaktert vagy sem. Akkor kapunk igaz értéket, ha a szó üres, különben hamis lesz a visszatérési értéke.

```
? ki üres? "|egy két|
hamis
? ki üres? "|
igaz
```

Az **elemszám**parancs a szó karaktereinek számát adja meg. Az üres szó nyilván 0 értékű lesz.

```
? ki elemszám "|egy két|
7
? ki elemszám"
```

A szó parancs mintájára lehetőségünk van különálló szavakból mondatot készíteni, Erre szolgál a mondat parancs.

? mutat (mondat „Jó” |LOGO-ban| „programozni!)

[Jó LOGO-ban programozni!]

A szó belsejében is keresgélhetünk. Az **eleme?**és**elem** parancsok a tartalmazást vizsgálják. Az eleme? a létezést, az eleme a tartalmazás helyét vizsgálja.

```
? mutat eleme? "e "szeles
igaz
?mutat eleme "e
2
```

Az első bemenet lehet egy szó is:

```
? mutat eleme? "les "szeles
igaz
?mutat eleme "les "szeles
3
```

A visszatérési érték akkor hamis, nem fordul elő a karakter a szóban, illetve a hely száma 0.

```
? mutat eleme? "x "zavaros
hamis
?mutat eleme "x "zavaros
0
```

Az első bemenet lehet egy szó is:

```
? mutat eleme? "fel "zavaros
hamis
?mutat eleme "fel "zavaros
0
```

Feladatok szavakkal

1. *Egy szó valamennyi magánhangzóját cseréljük ki az e betűre.*

A feladatot két eljárás segítségével oldjuk meg.

Az első eljárás segítségével meghatározzuk a magánhangzók halmazát:

```
Eljárás magánhangzó
    eredmény [ a á e é i í o ó ö ő u ú ü ű ]
Vége
```

Vizsgáljuk meg az eljárást:

```
? mutat eleme? "a magánhangzó

Igaz

? mutat eleme? "b magánhangzó

Hamis
```

A tényleges ***cserél***eljárásban végighaladunk majd a szó betűin, mindaddig, amíg üres szót nem kapunk. Ez lesz egyben az eredmény. Ha nem üres a szó, akkor viszont az első betűjéről el kell döntenünk, hogy eleme-e a magánhangzók halmazának, vagy sem. Ha igaz, akkor az eredménybe az e betűt írjuk helyette, ha hamis, akkor az eredeti betű kerül be, és az első betű elhagyásával kapott szót vizsgáljuk tovább.

```
eljárás cserél :szó

ha üres? :szó [eredmény :szó]

hak eleme? első :szó magánhangzó

[eredmény elsőnek "e cserél elsőnélküli :szó]

[eredmény elsőnek első :szó cserél elsőnélküli :szó]

vége
```

Vizsgáljuk meg az elkészített eljárást:

```
? mutat cserél "kitárul

keterel

? mutat cserél "stb

stb
```

A csere megvalósítására létezik a **csere** parancs is a LOGO nyelvben. Ha a **csere** első bemenet egy *szám,* akkor a **csere** a második bemenetével úgy tér vissza, hogy kicseréli annak szám-adik elemét a

harmadik bemenetre. Ha a *szám* kisebb, mint 1, akkor a harmadik bemenetet a második első elemeként szúrja be. Ha a *szám* nagyobb, mint a második bemenet elemeinek száma, akkor a harmadik bemenetet utolsó elemként szúrja be a másodikba.

```
? mutat csere 2 "kor "é
kér
? mutat csere 8 "kér "dés
kérdés
? mutat csere 0 "kér "fel
felkér
```

Ha a *szám* nem egész, akkor a csere eljárás kerekít és az így kapott helyen kerül sor a fenti műveletre.

```
? mutat csere 2.8 "kor "é
koé
? mutat csere 2.4 "kor "é
kér
```

Ha az első bemenet egy egyszerű *[szám1 szám2] szegmens,* azaz egy tartomány, akkor a művelet a második bemenetével úgy tér vissza, hogy kicseréli a szegmens által meghatározott összefüggő karaktereit a harmadik bemenetre. Ez persze túl is nyúlhat az eredeti bemeneten, de ekkor azt felülírja.

```
? mutat csere [2 2
]"bunda "or
Borda
? mutat csere [3 4] "banán "jonett
Bajonett
```

Figyeljünk arra, hogy ha az első bemenet egy szegmens, akkor a harmadik bemenetnek mindig meg kell egyeznie a másodikkal.

2. Készítsünk eljárást, amely megállapítja egy megadott szóról, hogy mély, magas vagy vegyes hangrendű-e!

Elsőként adjuk meg külön függvényekben a magas és mély magánhangzókat!

```
eljárás magas
eredmény [e é i í ö ő ü ű]
vége
eljárás mély
eredmény [a á o ó u ú]
vége
```

A bemeneti szóban megkeressük, hogy szerepel-e magas illetve mély magánhangzó a ***van.magas?*** és a ***van.mély?*** eljárások segítségével.

```
eljárásvan.magas? :szó
ha üres? :szó [eredmény "hamis]
ha eleme? első :szó magas[eredmény "igaz]
eredményvan.magas? elsőnélküli :szó
vége

eljárásvan.mély? :szó
ha üres? :szó [eredmény "hamis]
ha eleme? első :szó mély [eredmény "igaz]
eredményvan.mély? elsőnélküli :szó
vége
```

Ezután a szó betűit egyenként vizsgáljuk meg a ***hangrend***eljárásban. Ha van magas rendű, akkor megvizsgáljuk, hogy van-e mély is. Ha igen, akkor *vegyes,* ha nem, akkor tiszta *magas* hangrendű a bemeneti szó. Ha nem volt magas, de van benne mély magánhangzó, csak tisztán *mély* lehet, viszont ez a feltétel sem teljesül, akkor a bemeneti szóban *nem volt magánhangzó.*

```
eljárás hangrend :szó
havan.magas? :szó
[hakvan.mély? :szó
[eredmény "vegyes][eredmény "magas]]
hakvan.mély? :szó
[eredmény "mély][eredmény "|nincs benne magánhangzó|]
```

```
vége
```

Kész eljárásunkat vizsgáljuk meg a lehetséges eseteket!

```
? mutatvan.magas? "baba
hamis
? mutatvan.magas? "bébi
igaz
? mutatvan.magas? "stb
hamis
? mutatvan.magas? "ciklon
igaz
? mutatvan.mély? "ciklon
igaz
? mutat hangrend "baba
mély
? mutat hangrend "bébi
magas
? mutat hangrend "ciklon
vegyes
? mutat hangrend "stb
nincs benne magánhangzó
```

Bemeneti szavunkat magánhangzói hangrendje szerint kódolhatjuk is. A szó betűit egyenként megvizsgáljuk: Ha magas a magánhangzó, akkor ' jelet írunk, ha mély, akkor , jelet. Ha nem magánhangzó, akkor semmit.

```
eljárásrend.kód :szó
ha üres? :szó [eredmény :szó]
ha eleme? első :szó mély
[eredmény elsőnek "|,| rend.kódelsőnélküli :szó]
ha eleme? első :szó magas
```

```
[eredmény elsőnek "|'| rend.kódelsőnélküli :szó]
eredményrend.kódelsőnélküli :szó
vége
```

Az eljárás eredménye egy kódolt szó lesz, amely az eredeti szóban található magánhangzók számával azonos hosszú. Ezt a kódolt kimenetet vizsgáljuk meg, hogy azonos jeleket tartalmaz-e, vagy vegyesen fordulnak elő a magánhangzók, esetleg nincs is benne magánhangzó.

```
? mutatrend.kód "lapos
''
? mutatrend.kód "éléskamra
'',,
? mutatrend.kód "|kihagytál-e|
',,'
```

3. Készítsünk eljárást, amely a bemeneteként megadott szót megfordítja!

4. Írassuk ki egy mondatot betűnként megfordítva!

5. Egy mondat palindrom, ha a benne szereplő betűk a szóközöktől eltekintve azonos sorrendben helyezkednek el mindkét irányból. Készítsünk programot, amely eldönti egy mondatról, hogy palindrom-e!

Listakezelés alapjai

- **Elemi adat:** a karakter: 'k
  Speciálisan a számjegy elé nem szükséges idézőjel: 3

- **Összetett adat:**
  - elemi adatok sorozata a **szó**: 'kakukk vagy 1526
  - összetett adatok sorozata
    - azonos szerkezetű elemek **mondat** például

:

[
s
z
ó

s
z
ó
s
z
ó
]

m
o
n
d
a
t
s
o
r
o
z
a
t

p
é
l
d
á
u
l
:

[
[
e
l
s
ő

mondat]

[közép]

[utolsó mondat]]

- különböző szer

kezetű elemek[indulás [8 13] érkezés [10 22]]

A *lista* az ImagineLogo egyik legfontosabb összetett adatstruktúrája. A lista szavakból, képsorokból és egyé listákból állhat. Az *üres lista* (jele []) egy olyan speciális lista, melynek nincsen eleme. A mondat szintén fontos, szavakból álló szerkezet, mivel a szavakat szóközökkel választjuk el egymástól, ezért a mondat esetében is ezt használjuk elválasztóként. A mondat határait [] jelek jelölik. Üres mondatot nem adhatunk meg, mivel a [] jelekkel az üres listát jelöljük.

Műveletek listákkal

parancs	**leírás**
ELEMSZÁM	Megadja egy lista elemeinek számát. Az összetett objektumok (például listák) egy elemnek számítanak.
ELEME?	Eldönti, az első paraméter benne van-e a második paraméterben megadott listában.
ELEME	Megadja, hogy az első paraméter hányadik helyen szerepel (először) a listában. Ha nem szerepel, 0-t ad vissza.
ELSŐ	Egy nemüres lista első elemét adja meg.
UTOLSÓ	Egy nemüres lista utolsó elemét adja meg.
ELEM	Az adott indexű elemet adja vissza.
ELSŐNÉLKÜLI	Az első elem eltávolításával kapott listát adja vissza.
UTOLSÓNÉLKÜLI	Az utolsó elem eltávolításával kapott listát adja vissza.
ELSŐNEK	Az új elemmel az első helyen bővített listát adja meg.
UTOLSÓNAK	Az új elemmel az utolsó helyen bővített listát adja meg.
MONDAT	Két lista összefűzése.
EGYENLŐ?	Eldönti, hogy a két lista azonos-e.
ÜRES**?**	A lista ürességének vizsgálata.
ELEMNÉLKÜLI	A paraméterként megkapott érték összes előfordulását kiveszi a listából.

A lista adatszerkezet alapvetőbb műveleteivel foglalkozunk a következőkben.

Az első művelet a lista hozzárendelése egy változóhoz, ami a

```
lokálisérték :l [ e1 e2 e3 ... en-1 en]
```

paranccsal tehető meg például.

Az **első, elsőnek, elsőnélküli**a lista első, ei eleméhez fér hozzá, míg az **utolsó, utolsónak, utolsónélküli** az utolsó elemet, e„-t tudja kezelni. Az üres lista létrehozása történhet egyszerűen az értékadás utasítással is: a **lokért :l []** parancs egy :l nevű üres listát hoz létre, amely parancs egyben alkalmas például egy lista kiürítésére is. A lista elemszáma, üressége is vizsgálható. Erre szolgál az **elemszám** és az **üres?** parancs. A többi listaművelet közül kilóg az **elemnélküli,** amely a paraméterként megkapott érték összes előfordulását kiveszi a listából. Ez azért is kilóg a sorból, mert a listának általában az első vagy az utolsó elemét tudjuk kezelni, a belső elemek külön mutatókkal érhetők el. Az elemnélküli eljárás tulajdonképpen egy egyszerűsítés, amely a többi művelettel helyettesíthető is. Nézzünk meg néhány példát ezen parancsok működésére!

```
? mutat elemszám [2 14 5]
3
? mutat elemszám [2 [1 4 5]]
2
? mutat eleme 4 [2 1 4 5]
3
? mutat eleme? 6 [2 14 5]
hamis
? mutat elemnélküli 2 [2 [1 2 3]]
[1 [1 2 3]]
? mutat első [12 [12 3]]
12
? mutat utolsó [12 [1 2 3]]
[1 2 3]
? kiír utolsó [12 [12 3]]
1 2 3
```

Ez utóbbi két parancs ismételten példa két kiírás közti különbségre.

Lista legkisebb eleme

```
eljárás legkisebb :lista
 HAK (ELEMSZÁM :lista)=1
 [EREDMÉNY ELSŐ :lista]
 [HAK (ELSŐ :lista) > (UTOLSÓ :lista)
   [EREDMÉNY legkisebb ELSŐNÉLKÜLI :lista]
   [EREDMÉNY legkisebb UTOLSÓNÉLKÜLI :lista] ]
vége
```

Lista elemeinek összege

```
eljárás összegzés :lista
 HAK ÜRES? :lista
 [EREDMÉNY 0 ]
 [EREDMÉNY (ELSŐ :lista) +(összegzés ELSŐNÉLKÜLI :lista)]
vége
```

Feladatok listákkal

1. Egy beolvasott mondat szavai helyett adjuk meg rendre azok hosszát!

A feladat értelmezhető úgy, hogy ha a bemeneti mondat üres, akkor egy üres listát kell megadnunk, különben az első eleme helyett annak hosszát kell ími a végeredmény listába, majd ezután a maradék mondatra kell alkalmazni az eljárást.

```
eljárásszóhossz.lista :m
ha üres? :m [eredmény []]
eredmény elsőnek elemszám első :m szóhossz.lista en :m
vége
Nézzük meg, hogyan is működik az eljárásunk!
? mutatszóhossz.lista [Itt az idő!]
[3 2 4]
? mutatszóhossz.lista (szó "Itt "az "idő!)
[ 1 11111111 ]
```

Mint látható, a ! jel is karakter, a második esetben lista helyett szó a bemenet, amit nem listaként, hanem karakterenként vizsgál meg.

*2. Javítsuk ki a **szóhossz.lista**eljárást úgy, hogy szó bemenet esetén a szó hosszát adja vissza!*

A lista elemeinek sorrendjét is tudjuk változtatni, például ha a lista első elemét akarjuk a lista végére helyezni, akkor tulajdonképpen egy jobbra léptetést hajtatunk végre.

```
eljáráslép.jobbra :s
eredmény elsőnek utolsó :s utolsónélküli :s
vége
```

```
Nézzünk meg egy futási példát:
? mutatlép.jobbra "lakó
Ólak
```

Vegyük észre, hogy a listaműveletek kevés kivétellel azonosak a szóra alkalmazható műveletekkel.

*3. Készítsünk **lép.balra**nevű függvényt, amely a lista utolsó elemét teszi át a lista első helyére!*

*4. Készítsünk **egyelemű** nevű függvényt, amely akkor ad igaz eredményt, ha a paramétereként megadott lista vagy szöveg egy elemű!*

*5. Készítsünk**szerepel** nevű függvényt, amely megállapítja, hogy a paramétereként megadott listában hányszor szerepel egy érték!*

Programozási tételek I.

A programkészítés során a részekre bontott feladat megoldásának nagy része visszavezethető egyszerűen megfogalmazott kis algoritmusokra, ezeket nevezzük elemi algoritmusoknak, vagy más néven programozási tételeknek.

ÖSSZEGZÉS TÉTELE

A bemeneti sorozat vagy lista alapján egyetlen értéket kell meghatároznunk. Lényeges, hogy a kimenet meghatározásában a bemenet összes eleme szerepet játszik. A probléma alapján ezt a csoportot összegzésnek nevezzük. Adjuk össze egész számok sorozatának elemeit!

Példa: összegzés [4 3 5] Eredménye: 12

```
eljárás összegzés :a
 ha üres? :a [er0]
 eredmény (első :a) + összegzés en :a
vége
```

Összegzésre visszavezethető problémák:

1. Határozzuk meg egy tantárgyból kapott osztályzatok átlagát, ahol az osztályzatokat a :oszt lista tartalmazza.
2. Adjuk meg az első 10 természetes szám szorzatát.
3. Egyesítsük egyetlen szóvá a listában tárolt egymást követő elemeket.

ELDÖNTÉS TÉTELE

Ebbe a csoportba azokat a problémákat soroljuk be, amikor az a célunk, hogy megállapítsuk a bemenet minden eleme alapján legalább egy elemére meglévő tulajdonság meglétét vagy hiányát. A megvalósítás során azt várjuk, hogy IGAZ vagy HAMIS legyen a visszatérési érték a keresett tulajdonság teljesülése alapján.

1. Készítsünk eljárást, amely megválaszolja, hogy egy listában van-e 0-nál kisebb elem!

```
Eljáráseldöntés :lista

   Ha üres? :lista [eredmény „hamis]

   Hakelső :lista< 0

         [eredmény „igaz]

         [eredmény eldöntés elsőnélküli :lista]

   Vége
```

2. Döntsük el egy diák évvégi jegyei alapján, hogy kitűnő lett-e vagy sem!

3. Állapítsuk meg egy szóról, hogy magas hangrendű vagy sem!

KIVÁLASZTÁS TÉTELE

A kiválasztás esetén tudjuk, hogy van keresett tulajdonságú elem a bemenetben, de azt is szeretnénk meghatározni, hogy melyik az első eleme a bemenetnek, melyre teljesül ez a tulajdonság. Eredménynek várhatjuk magát az első megfelelő értéket, valamint a bemenetben elfoglalt helyének a sorszámát. Ennek megfelelően kétfajta megvalósítás is lehetséges.

1. Készítsünk eljárást, amely megadja egy listában az első 0-nál kisebb elem helyét!

```
Eljáráskiválasztás.hely :lista
  Hakelső :lista< 0
      [eredmény 1]
      [eredmény 1 + kiválasztás.helyelsőnélküli :lista]
  Vége
```

2. A hely mellett az értéket is visszaadhatjuk, most az első 0-nál kisebb értékre vagyunk kíváncsiak.

```
Eljáráskiválasztás.érték :lista
  Hakelső :lista< 0
      [eredmény első :lista]
      [eredmény + kiválasztás.értékelsőnélküli :lista]
  Vége
```

3 Adjuk meg egy összetett szám legnagyobb, vele nem egyenlő, valódi osztóját!

4. Adjuk meg egy összetett szám legkisebb valódi, nem 1-gyel egyenlő osztóját!

5. Adjuk meg egy legalább két szótagos szóban az első magánhangzó helyét!

Programozási tételek II.

MEGSZÁMLÁLÁS

A probléma csoportban azt határozzuk meg, hogy a bemenet elemei közül hány rendelkezik a keresett tulajdonsággal.

```
eljárás megszámlálás :lista
    ha üres? :lista [eredmény 0]
    hakelső :lista< 10
       [eredmény 1 + megszámlálás en :lista]
       [eredmény megszámlálás en :lista]
vége
```

1.. Egy osztály tanulóinak átlaga alapján adjuk meg a jeles rendűek számát!

2. Adjuk meg egy szó mássalhangzóinak számát!

3. Határozzuk meg egy mondat szavaira kiszámított magánhangzók és mássalhangzók arányát!

MINIMUM MEGKERESÉSE

A bemenet értékein, ha értelmezett egy rendezés, akkor problémaként felmerülhet, hogy melyik a legkisebb elem, azaz vagy hol helyezkedik el a bemeneten belül, vagy konkrétan mi az értéke. A bemenetről feltesszük, hogy legalább egy elemű.

```
eljárás minimum :l
   ha üres? en :l [eredmény első :l]
   hakelső :l> utolsó :l
        [eredmény minimum en :l]
        [eredmény minimum un :l]
vége
```

1. Készítsünk eljárást, amely a minimum első előfordulási helyét adja meg! Az eljárás minimális átalakításával megkaphatjuk a maximum meghatározását is!

MAXIMUM KERESÉSE

```
eljárás maximum :l
   ha üres? en :l [eredmény első :l]
```

```
hakelső :l< utolsó :l
        [eredmény maximum en :l]
        [eredmény maximum un :l]
vége
```

Vegyük észre, hogy az eljárások alkalmasak ebben a formában is egyetlen szó legnagyobb karakterkódú elemének meghatározására is.

1. Adjuk meg, hogy egy beteg napi hőmérséklet mérései alapján melyik nap volt a legmagasabb a testhőmérséklete!

2. Adjuk meg egy szám legkisebb és legnagyobb számjegyének összegét!

3. Keressük meg egy verseny első három legjobb eredményt elért indulóját!

Programozási tételek III.

LINEÁRIS KERESÉS

Ebben a problémában nem tudjuk biztosan, hogy létezik-e egyáltalán a bemenetben az adott tulajdonságú elem, de ha igen, akkor az első ilyen helyét vagy értékét várjuk visszatérő értékként. Ha az adott tulajdonságú elem nem létezik, akkor a hely keresése esetén a visszatérési érték 0 legyen! A konkrét megvalósításban egy ***valami*** nevű érték előfordulási helyét keressük a bemenetben!

1. Készítsünk eljárást, amely egy adott érték első előfordulási helyét határozza meg egy listában!

```
eljáráslin.keresés :valami :lista
    ha üres? :lista [eredmény 0]
    hakelső :lista = :valami
       [eredmény 1]
       [eredmény 1 + lin.keresés :valami en :lista]
vége
```

2. A fenti megvalósítás, ha nem eleme a keresett **:valami**a listának, akkor az elemszámot adja vissza! Javítsuk ki az eljárást!

3. Adjuk meg egy sorozat első olyan elemét és helyét, amely kisebb, mint az előtte álló!

4. Egy hónapon keresztül mérjük a reggeli hőmérsékleteket. A mérések alapján adjuk meg az első negatív értékű nap sorszámát!

5. Adjuk meg egy egész szám 1-től és önmagától különböző egyik osztóját!

KIVÁLOGATÁS

A probléma csoportba azokat soroljuk, amelyekben bemenet elemeiből egy új adatszerkezetbe kigyújtjuk az összes adott tulajdonságú elemet. E feladat esetén is lehetséges, hogy nem magukat az értékeket, hanem csak a bemeneti sorozatban elfoglalt helyüket kell kigyűjteni.

1. Itt most a bemeneti lista 10-nél kisebb elemeit gyűjtsük ki.

```
eljárás kiválogatás :lista
  ha üres? :lista [eredmény []]
  hakelső :lista< 10
     [eredmény elsőnek első :lista kiválogatás
  elsőnélküli :lista]
     [eredmény kiválogatás elsőnélküli :lista]
```

vége

2. Készítsük el azt az eljárást, amely nem az adott tulajdonságú elemeket, hanem azok helyét gyűjti ki egy listába!

3. Gyűjtsük ki egy osztály jeles tanulóinak nevét!

Programozási tételek alkalmazása

A funkcionális LOGO használatának igen fontos területe, hogy felismerjük, milyen programozási tételek bújnak meg a megoldandó problémában.

MEDIÁNMEGHATÁROZÁSA

1. Határozzuk meg egy lista mediánját!

Egy sorozat mediánján a nagyság szerint rendezett elemei közül a középső helyen állót értjük. Ha a sorozat elemszáma páros, akkor vehetjük a két középső közül bármelyiket vagy éppen a két középső elem átlagát. Mi ez utóbbit értjük most alatta. A megoldás ötlete hasonlít a minimum megkereséséhez. Ha a sorozat egy elemű, ez az eleme maga a médián. Ha a sorozat kételemű, akkor a két elem átlaga, különben pedig elhagyjuk a legkisebb és a legnagyobb elemet, azaz kettesével csökkentjük a bemenet hosszát.

```
eljárás medián :lista
    ha üres? en :lista [eredmény első :lista]
    ha esz :lista = 2 [eredmény ((első :lista) + utolsó
  :lista)/2]
    eredmény medián en un :lista
vége
```

Figyeljünk arra, hogy a bemeneti sorozatnak rendezettnek kell lennie, éppen ezért vagy rendezetten adjuk meg, vagy használjuk a **rendez** parancsot.

```
? mutat medián rendez [1 6 2 3 4]
3
? mutat medián rendez [16 2 3 4 5]
3.5
```

2. Készítsük el úgy a **medián** meghatározását, hogy nem rendezzük a bemeneti sorozatot! Vegyük észre, hogy a rendezett sorozat legkisebb és legnagyobb elemét hagyjuk el, ami megoldható a **minimum** és a **maximum** elhagyásával is, ha kettőnél több elemű! Figyeljünk arra, hogy csak egy-egy ilyen értékű elemet szabad elhagynunk!

3. Vizsgáljuk meg, hogy nagy elemszám esetén a rendezés vagy a minimum és maximum megkeresése hatékonyabb-e!

PRÍMTÉNYEZŐKRE BONTÁS

Egy természetes számot prímtényezőire bonthatjuk, ha a szám összetett. Az 1 nem összetett szám, prímtényezős felbontása nincs. A többi pozitív egész szám esetében a lehetséges legkisebb tényező a 2.

1. Készítsük el egy pozitív egész szám prímtényezős felbontását!

Az algoritmusunk alapgondolata az, hogy a számot elsőként megpróbáljuk 2- vel osztani. Ha osztót találunk, akkor addig írjuk ki a tényezők közé és osztjuk vele a számot, amíg lehet. Ha már nem osztó, akkor az osztót 1-gyel megnöveljük, és ez lesz az új osztó. Az eljárást addig végezzük, amíg 1 nem lesz a szám. Az eljárásban az :n jelentse a vizsgálandó számot, az :o az első osztót

```
eljárás prim.tényező :n :o :lista
  ha :n=1 [eredmény :lista]
  hak (maradék :n :o) = 0
      [eredmény prim.tényező (egészhányados :n :o) :o
    (utolsónak :o :lista)]
      [eredmény prim.tényező :n :o+1 :lista]
vége
```

Az eljárás meghívása a **prim.tényező :n 2** [] formában történhet.

```
? mutatprim.tényező 28 2 []
[2 2 7]
? mutatprim.tényező 12 []
[]
```

Vegyük észre, hogy feleslegesen próbálkozunk a 2 és 3 után a 4 értékkel, mivel az nem prím, és mint a 2 többszöröse, már biztos nem is szerepelhet tényezőként. Míg az 5-tel biztosan kell vizsgálatot végeznünk, a 6-tal nem, mivel a 2 és a 3 többszöröse. Vegyük észre, hogy az eljárás hatékonyabbá tehető, ha a 2 és 3 után már csak a 6k±l alakú osztókkal kísérletezünk!

2. Alakítsuk át úgy az eljárást, hogy a prímek ismeretében csak a prímszámokkal végzi el az osztáspróbát! A prímszámokat egy listában kapja meg az eljárás!

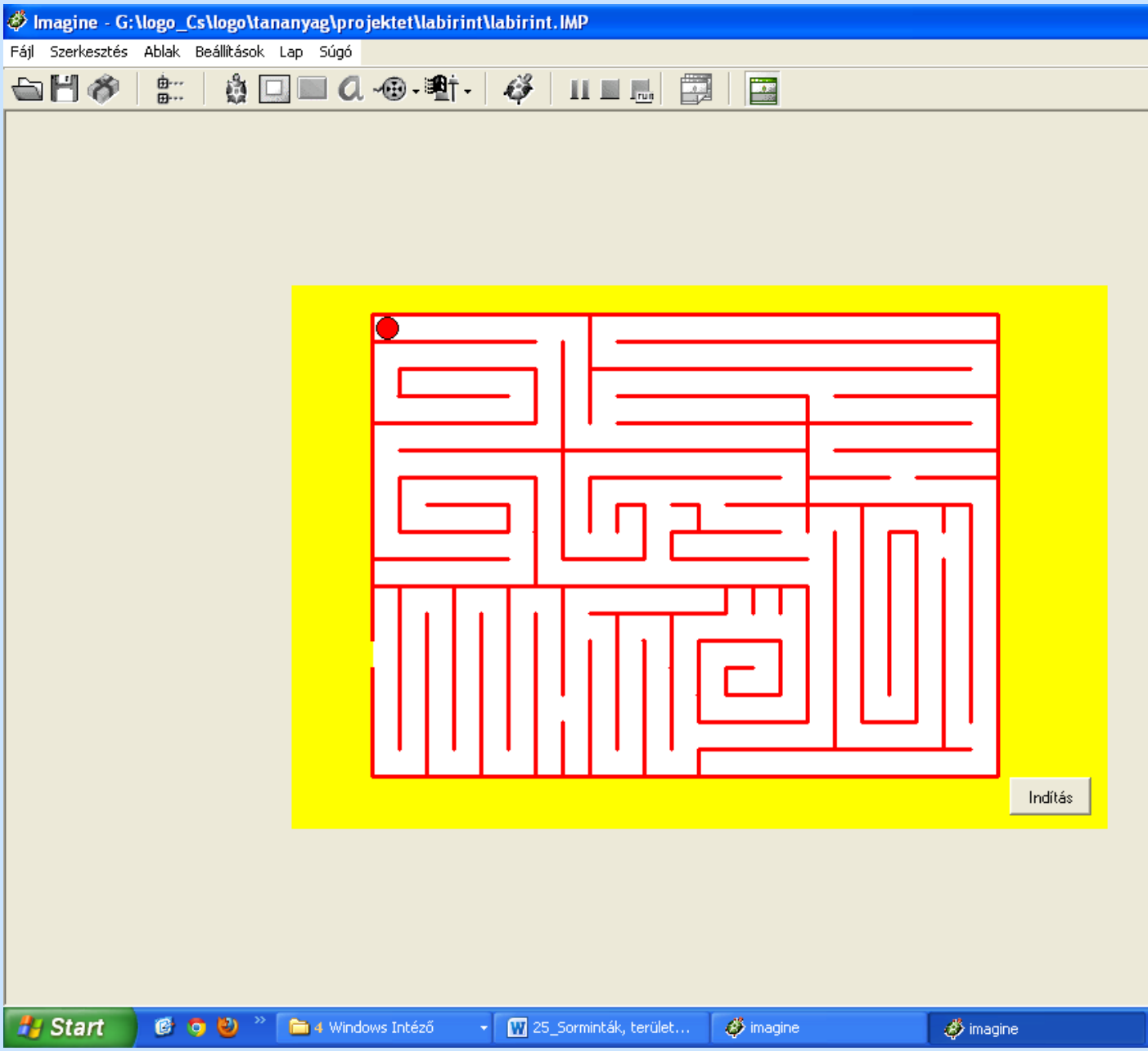
Játékok készítése I.

Az Imagine egyik legnagyobb erőssége, hogy nagyon egyszerűen lehet benne játékokat fejleszteni. Ehhez szükségünk lesz az eddig tanult parancsok, eljárások, vezérlőszerkezetek használatára.

Menjünk végig a labirintuson!

Készítsünk labirintust! A kész labirintusban haladjunk végig, miközben mérjük az eltelt időt!

A játékot két lapon valósítjuk meg. Az egyik lapon - ezt *tervezőnek* nevezzük - a labirintust lehet készíteni, a másik lapon - a *játék* nevűn -, az elkészített labirintusban lehet mozogni.



Tervező lap

Az ***indító*** eljárásunk nem a főablakhoz, hanem a laphoz tartozik. Az összes műveletet itt végezzük most el. Hogy futás közben ne zavarjon az Imagine környezet, az egyes kezelő elemeket parancs segítségével bezárjuk, majd meg is nyitjuk a munka végeztével.

```
eljárás Indító
  tervező'méret! [600 400]
  törölképernyőrejtikonsor rajzlapablak
  törölobjektum mindenteknőc
  új "teknőc
  [név paca poz [-100 -100] irány 0
  tolltl tsz 12 tv 3
  látható igaz alak (szó betöltőút "paca) képkockamód igaz]
  paca'mozgat
  mutatikonsorosztottablak
vége
```

Az eljárásban azt figyeljük meg, hogy a *paca* nevű teknőc szinte minden lehetséges tulajdonságát a létrehozásakor, eljárásban adjuk meg, nem párbeszédablakban. Érdemes megfigyelni a teknőc osztályba kerülő új objektum tulajdonságait, hogy milyen párok alkotják!

A létrehozott teknőcöt mozgatjuk a képernyőn. A mozgatáshoz a leütött billentyű kódját figyeljük. A szóközzel szabályozzuk a toll helyzetét.

```
eljárás mozgat
  elágazásolvaskód
  [-72 [irány! 0 e 20]
  -80 [irány! 180 e 20]
  -75 [irány! 270 e 20]
  -77 [irány! 90 e 20]
  27 [stopmind]
  32 [hak toll = "tollatfel [képkocka! 1 tl][képkocka! 2
tf]]]
  mozgat
vége
```

A vezérlés megadásához az ASCII kódokból kialakított kódokat alkalmaztuk. Az ESC kódja a 27, a szóközé a 32, a kurzornyilak kódjai pedig rendre: a fel -72, jobbra -77, le -80, balra -75.

Átalakíthatod a mozgat eljárást úgy, hogy a kódok helyett az Imagine-ben használatos megnevezések szerepeljenek!

A játék lap

A tervező lapon elkészített és elmentett pályánkon azonnal indulhatna a játék, de előtte az elkészített labirintuson végezzünk el egy kis módosítást: a külső részt fessük be sárgára a rajzeszközök segítségével! Az *indító* eljárásunk a *játék* lapon az alapbeállítások elvégzését, és a mozgatás indítását oldja meg. Elsőként a lap jellemzőit adjuk meg, a **méret!** paranccsal a tényleges méretét képpontban, a **kiindulópont!** parancs pedig a bal felső sarokhoz képest adja meg a lap középpontját. A megoldás idejét is szeretnénk mérni, ezért szükségünk lesz egy stopperre, amelyet a tényleges játék megkezdése előtt - **paca'mozgat** - indítunk el.

```
eljárás indító
   játék'méret! [600 400] játék'kiindulópont! [299 199]
   rejtikonsor rajzlapablak
   betöltháttérképbetöltőút "labi2.bmp
   törölobjektum mindenteknőc
   új "teknőc
   [név paca poz [-229 168] irány 0 toll tfképkockamód igaz
   látható igaz alak (szó betöltőút "paca) vonszolható igaz]
   új "teknőc
   [név óra poz [-50 -165] irány 0 toll tl látható hamis]
   óra'betűtípus! [[Tahoma][16 400 0 00 238]]
   nullázstopper
   paca'mozgat
   mutatikonsorosztottablak
vége
```

A létrehozott teknőcök jellemzőit itt is érdemes megfigyelnünk. Az *óra* teknőc az időpont kiírását segíti majd, de magát a teknőcöt nem fogjuk látni a képernyőn. A *paca* nevű teknőcöt - amely a játék lapon él és így nem azonos a tervező lapon megadott teknőccel - kell végig vinnünk a labirintuson. Mozgatása már Imagine környezetben a **gombmenü!** paranccsal szebben megoldható.

```
eljárás mozgat
   paca1 gombmenü!
   [fel [irány! 0]
   le [irány! 180]
   jobbra [irány! 90]
   balra [irány! 27 0]
   esc [mutatikonsorosztottablak stopmind]]
   paca'előre 10
   ha paca1 pontszín = "vörös [hátra 10]
   célba?
   várj 100
   mozgat
vége
```

Az irányítás már ismert, az újdonság a ***célba?*** függvény. Ez logikai értéket ad vissza.

```
eljárás célba?
   ha paca'pontszín = "sárga [kér "óra [címke (mondat
   maradék
   stopper 1000 "|ms alatt értél ki.|)] átdob "felsőszintre]
vége
```

Az eljárás érdekessége, hogy ha a feltétel teljesül, akkor nem térünk vissza a meghívó eljáráshoz ***(mozgat),*** hanem a fő meghívó eljáráshoz az **átdob** alkalmazásával.

2. Tegyük programunkat századmásodperc pontossá!

Az elkészített pálya fontos eleme a start és a cél koordinátája, valamint maga a pálya. Ezeket most színekkel jelöltük, de megadhatjuk koordinátákkal is.

3. Adjuk meg a célba? függvényt úgy, hogy a célterület koordinátáját ismerve határozzuk meg az eredményt!

Ezeket az elemeket külön listában tároljuk el. A szerkezete ***[fstartxstartyjfcélxcély]*** pályanév] legyen.

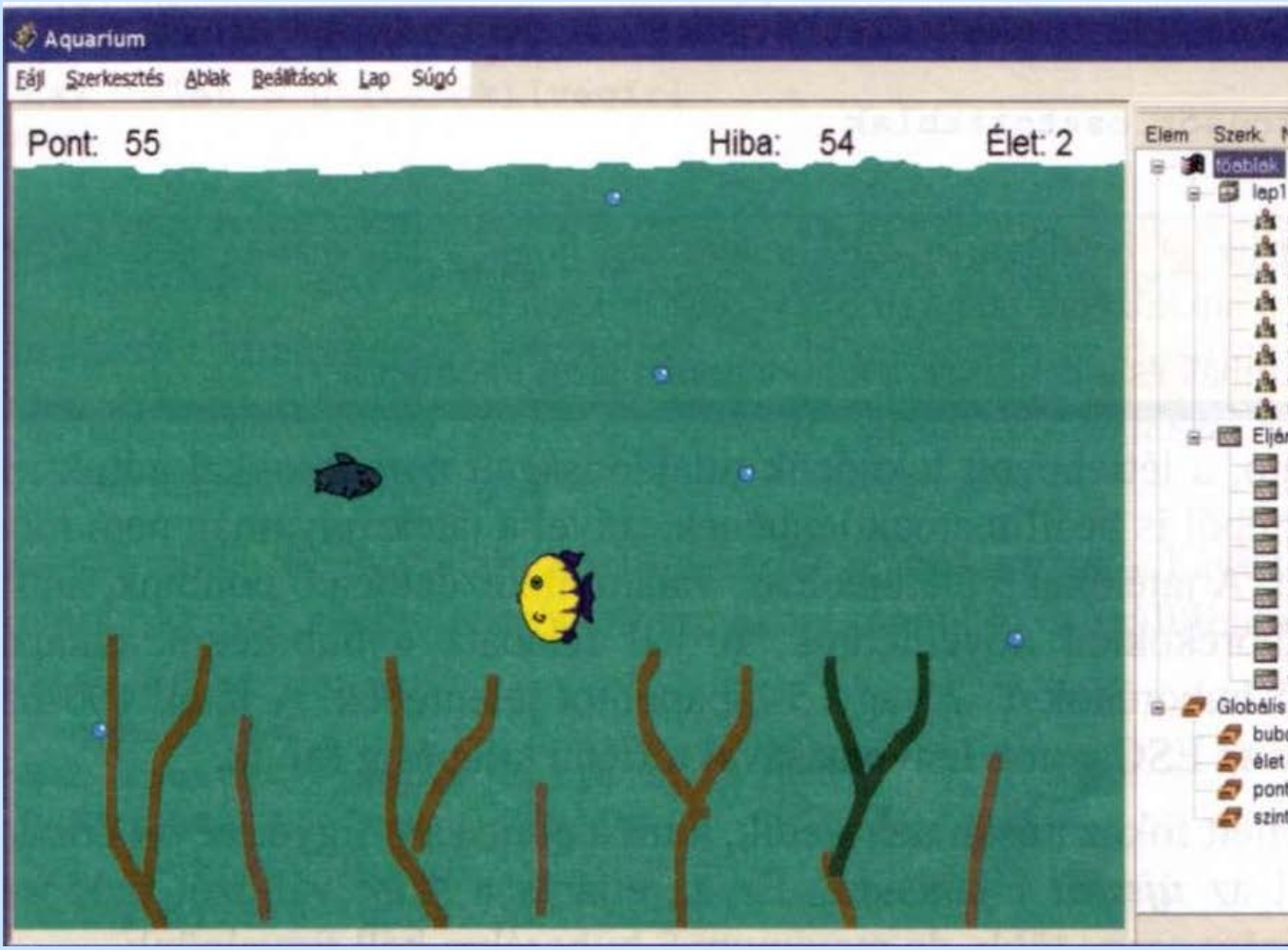
35. Készítsük el a labirintus módosított változatát, ahol a tervező és szerkesztő nézet között válthatunk, valamint a fenti adatszerkezetben mentjük tervező nézetben a pályát, és a játék nézet a fenti adatszerkezetből veszi a pályát!

36. Készítsünk eredménylistát a feladat megoldásához!

Játékok készítése II.

CÁPA AZ AKVÁRIUMBAN

Készítsünk játékot, amelyben egy akváriumban úszó halat kell irányítani úgy, hogy elkerüljük a minket üldöző cápát és összegyűjtsünk minél több vízből felszálló buborékot



A programot indító eljárás a kezdő paraméter átadása miatt az indítnevűt hívja meg, így lesz értéke a kezdetben a szint változónak.

```
eljárás indító
   globvál "szint 1
   indít 50
vége
```

Az érdemi munkát az indítvégzi el.

```
eljárás indít :várakozás
   rejtikonsor rajzlapablak bezárintéző
   betöltháttérképbetöltőút "medence
   ism 5 [új "teknőc [név (szó "t hányadik) poz [10 10]
   irány 270 toll tf látható igaz]]
   ism 5 [kér (szó "t hányadik)
   [alak! betöltőút "buborék xypoz! 50 + vsz 700 5]]
   új "teknőc
   [név cápa poz [400 20] irány 90 toll tf látható igaz]
   figyelj "cápa alak! betöltőút "capa
   új "teknőc
   [név hal poz [400 400] irány 90 toll tf látható igaz]
   kér "hal [ alak! betöltőút "hal
   gombmenü! [balra [irány! 270] jobbra [irány! 90]
   fel [irány! 0] le [irány! 180]
   esc [végír]]]
   új "teknőc [név író irány 0 toll tf látható hamis]
   író'betűtípus! [[Arial][14 400 0 10 238]]
   globvál "pontszám 0 globvál "bubdb 0 globvál "élet 3
   figyelj "író
   xypoz! 10 530 címke "Pont: xypoz! 700 530 címke "Élet:
   xypoz! 750 530 címke :életxypoz! 500 530 címke "Hiba:
   hurok
   mutatikonsorosztottablak
vége
```

Amint látható, a létrehozott teknőcök tulajdonságait parancsokkal adtuk meg, de ezek menüből is beállíthatóak lennének, mivel a játék folyamán nem módosítjuk ezeket. A játékban 3 életünk lesz, valamint kezdetben 0 pontunk, amit az elkapott buborékokkal növelhetünk. A hal feladata a buborékok elkapása. A megszökő buborékok 1, 2 vagy 3 hibapontot jelentenek. A játék 100 hibapontig, vagy az ESC gomb lenyomásával történő kilépésig folyik. A játék emellett fokozatosan

nehezedik, amit a pontszám figyelésével érünk el. Erre szolgál az új szint eljárásunk. Ez az eljárás a szint változót módosítja, aminek hatására egyre több, de maximum 5 buborékra kell figyelnünk.

```
eljárás újszint
   elágazás :pontszám
   [5 [globvál "szint 2]
   15 [globvál "szint 3]
   3 0 [globvál "szint 4]
   50 [globvál "szint 5]]
vége
```

Az eljárás hatását a ***bub.mozog***eljárásban láthatjuk, a buborékok számát adja.

```
eljárásbub.mozog
   ism :szint
   [figyelj (szó "t hányadik)
   irány! 45 - vsz 90
   e 2 + vsz 3
   haypoz> 535
   [növel "bubdb figyelj "író xypoz! 580 530 címke :bubdb]]
vége
```

Érdemes megfigyelni a buborék életszerű mozgását megvalósító programrészt. Itt oldjuk meg az elszökő buborékok figyelését is. Az eljárások meghívását is klasszikus módon végezzük el. A ***hurok*** nevű eljárásunk hívja a mozgató és a vizsgálatokat végző eljárásokat.

```
eljárás hurok
   hal.mozog
   cápa.mozog
   bub.mozog
   találkozó
   várj :várakozás
   új szint
   ha :bubdb> 100 [végír]
```

```
   hak :élet> 0 [hurok][végír]
vége
```

Az önmagát rekurzívan hívó ***hurok*** eljárás kiváltható lenne a már többször használt **minden** vagy örökké parancsokkal is, de most ez nem volt a célunk. A ***várakozás*** paramétert itt használjuk fel, de sehol sem módosítjuk.

Nehezítsük úgy a játékot, hogy a várakozás értékét az újszint eljárásban módosítjuk!

A ***hurok*** által meghívottak közül nézzük a ***hatmozog***eljárást! Ez olyan feladatot old meg, amely teknőcjellemzőként menüből is beállítható lenne.

```
eljáráshal.mozog
   figyelj "hal
   hakxpoz<20 [xpoz! 20] [e 2]
   hakxpoz>780 [xpoz! 780][e 2]
   hakypoz<20 [ypoz! 20][e 2]
   hakypoz>520 [ypoz! 520][e 2]
vége
```

*Helyettesítsük a területmegadást a **hal** jellemzőinek menüből történő beállításával, vagy a* **tartomány** *parancs használatával!*

A cápa mozgása jóval egyszerűbb, mivel annak csak követnie kell a halunkat.

```
eljáráscápa.mozog
   figyelj "cápa
   irány! iránysz̈og kér "hal [poz]
   e 1
vége
```

A találkozások vizsgálatát és az elkapott buborékok helyett új létrehozását végzi a ***találkozó*** eljárás. Az eljárás emellett kiírást is végez.

```
eljárás találkozó
   ism :szint
   [figyelj "hal
   ha takar? (szó "t hányadik)
   [növel "pontszám
```

```
    figyelj "író xypoz! 80 530 (címke :pontszám)
    figyelj (szó "t hányadik) xypoz! 50 + vsz 700 5
    hangsor [S0 1122 T120 L4 02 G 16B]]]
    figyelj "cápa
    ha takar? "hal [xypoz! vsz 700 vsz 400 ( növel "élet -1 )
    figyelj "író xypoz! 750 530 (címke :élet) hanghullám
    "jaj]
vége
```

Amikor a hal elkap egy buborékot, akkor megszólal egy hangsor. Ez megállítja a játék futását. Oldjuk meg, hogy a játék továbbfusson! Már csak a végír eljárásunk maradt ki, amely a játék befejezésekor kap szerepet.

```
eljárásvégír
    figyelj "író
    betűtípus! [[Arial][80 700 10 0 238]]
    xypoz! 150 300
    tsz! 12
    címke "VÉGE!
    stopmind
vége
```

Ez a programrészlet is fejleszthető!

Oldjuk meg, hogy a VÉGE felirat fokozatosan megnőve jelenjen meg a képernyő közepén! Figyeljünk arra, hogy a címke parancs az író teknőc pozíciójában kezdi a kiírást, tehát azt is mozgatnunk kell.

Utolsó feladatunk a meghívási módokra hívja fel a figyelmet.

A **figyelj***és* **kér** *parancsokkal szólítjuk meg szinte mindenhol a teknőcöket. Írjuk át a programot úgy, hogy csak ott használjuk ezeket, ahol ténylegesen nem elkerülhető!*