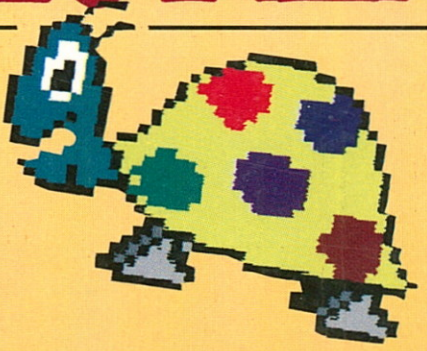
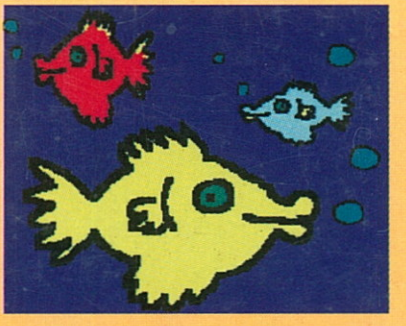
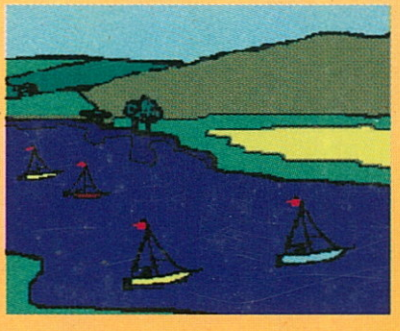
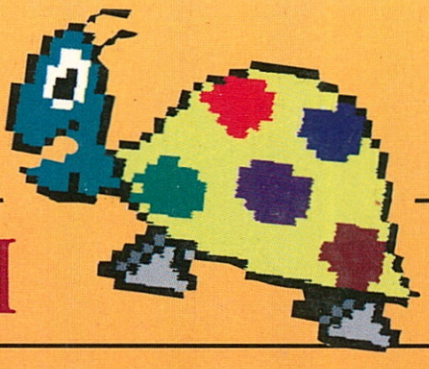
TURCSÁNYINÉ SZABÓ MÁRTA–ZSAKÓ LÁSZLÓ

COMENIUS LOGO GYAKORLATOK









KOSSUTH KIADÓ

TURCSÁNYINÉ SZABÓ MÁRTA – ZSAKÓ LÁSZLÓ

COMENIUS LOGO GYAKORLATOK

KOSSUTH KIADÓ 1997

LEKTORÁLTA
KŐRÖSNÉ MIKIS MÁRTA
SZLÁVI PÉTER

ISBN 963 09 3956 8

FELELŐS KIADÓ KOCSIS ANDRÁS SÁNDOR
A KOSSUTH KIADÓ RT. ELNÖK-VEZÉRIGAZGATÓJA
SZERKESZTETTE HAJNAL GABRIELLA
A FEDÉLTERV KUN GÁBOR MUNKÁJA
MŰSZAKI VEZETŐ KUN GÁBOR
MŰSZAKI SZERKESZTŐ PÁNYI BÉLA
TERJEDELME 10,5 (A/5) ÍV
SZEDTE ÉS TÖRDELTE A KOSSUTH KIADÓ
NYOMTA ÉS KÖTÖTTE AZ ETOPRINT NYOMDAIPARI KFT.
FELELŐS VEZETŐ BALOGH MIHÁLY ÜGYVEZETŐ IGAZGATÓ

TARTALOM

BEVEZETŐ

A Comenius Logo országlicencének megvásárlása és magyar nyelvű verziójának elkészülte után egyre több iskola vásárolja meg a 3.0 számozással jelölt programcsomagot. A forgalomba hozók vállalták, hogy felkészítik a Logo használatára az oktatási intézmények illetékes tanárait, de másokat is, ha a képzésre igényt tartanak és vállalják annak költségeit. Ebből a célból máris kiképezték az „oktatók oktatóit", azaz mintegy negyven Logo szakértőt, akik a tanári felkészítőket vezetni fogják.

Ez a könyv a Logo szakértők tanfolyamának anyagát foglalja össze rendszerezett és közérthető, tanárok és más felnőtt felhasználók számára hasznosítható formában. Megismertet a Logo programozási nyelv alapjaival, magával a Comenius Logo programcsomaggal, annak installálásával és működtetésével. Külön fejezetek szólnak a Comenius Logo egyes programrészeiről, a program különböző funkcióiról. Nagyobb teret szentel a könyv a rajzprogramoknak, a Comenius Logo lehetőségeinek a koordináta-rendszer és a fraktálok vonatkozásában, a Logo-animációknak, különösen az interaktív mozgatási (alakzatváltoztatási) módozatoknak és a hangoknak. Önálló rész foglalkozik a Logo függvényekkel, ezen belül a szövegkezeléssel. Nem nélkülözi a kötet a módszertani tanácsokat sem, s külön értéke a függelékben a Comenius Logo 3.0 verzió definitív leírása.

Minthogy Magyarországon ez az első Comenius Logo-kötet, a kiadvány az alapozó-eligazító kézikönyv funkcióját is betölti. Nyilvánvalóan nem hiánytalanul, reméljük azonban, hogy sikeresen. A könyvet forgatva egyénileg tájékozódni lehet a Logo világában, és természetesen azoknak is melegen ajánljuk kötetünket, akik a Logo tanfolyamok résztvevőiként ismerkednek meg behatóbban a Comenius Logóval.

A Kiadó

A LOGO PROGRAMOZÁSI NYELV

A LOGO NYELV KIALAKULÁSA

A Logo a görög „logosz" szóból származik, amely magyarul értelmet, tudományt jelent. A Logo programozási nyelv és a hozzá kapcsolódó pedagógiai elvek kidolgozása és elterjesztése elsősorban Seymour *Papert* amerikai professzor nevéhez fűződik, aki az ötvenes években Genfben dolgozott, *Piaget* neves pszichológus munkatársaként. Piaget nevét a magyar pedagógusok jól ismerik: e század talán legnevesebb fejlődéslélektani kutatója, a gyermeki megismerés, a kognitív ismeretszerzés vizsgálója. A kisgyermek nem úgy gondolkodik, mint a felnőtt: a körülötte levő világ modelljét tapasztalataiból építi fel. Szinte „szomjazik" a tudásra. A felnőtt feladata ezért olyan interaktív környezet biztosítása, amelyben a gyermek tanulási vágya kibontakozhat, amelyben felfedezéseket tehet, mégpedig saját tempójában, mindenfajta erőltetés és siettetés nélkül.

Piaget elvei és pszichológiai kísérleteinek eredményei nagy hatással voltak a matematikus Papertre, amikor a hatvanas években az amerikai MIT-re (Massachussetts Institutes of Technology, a világ egyik leghíresebb műszaki egyeteme) visszatért. Ebben az időben kezdték meg a számítógép-fejlesztők a mesterségesintelligencia-kutatásokat, és ekkor született meg a LISP programozási nyelv „melléktermékeként" a Logo. Alkalmazásának kísérleti oktatási tapasztalatai már kezdetben igen pozitívak voltak, pedig látványos teknőcgrafikáját csak néhány évvel később fejlesztették ki. A Logo igazi oktatási sikerét és **nemzetközi népszerűségét** tulajdonképpen a teknőcgrafika és a mindenki által hozzáférhető mikroszámítógépek elterjedése hozta meg.

A Logo – bár mindent „tud", amit bármelyik magasszintű programozási nyelv –, mégis lényegesen többet jelent egy számítógépes programnyelvnél. Tulajdonképpen egy olyan **pedagógiai környezetet, „mikrovilágot"** jelent, amelyben a gyermekek maguk tehetnek felfedezéseket, miközben minden kényszer és „magolás" nélkül számos új ismeret birtokába jutnak. A teknőc a számítógép billentyűzetén keresztül utasítható a számára „érthető" feladatok elvégzésére: tud adott távolsággal előre vagy hátra menni, adott szöggel jobbra vagy balra elfordulni, tollát (ami a hasára van erősítve) felemelni, leereszteni, más színűre cserélni, ezáltal mozgásával érdekes nyomokat hagyni a képernyőn. A teknőcgrafika fokozatosan egy új tudományterület, a **teknőcgeometria** megszületéséhez vezetett.

A diák azáltal, hogy parancsot ad a képernyőteknőcnek, rögtön ellenőrizheti gondolkodásának és cselekedeteinek következményeit. Megfigyeli utasításainak hatását, majd módosíthatja azokat céljának tökéletesebb megvalósítása érdekében. A kipróbálások és módosítások sorozata egybecseng a piaget-i értelmi fejlődés sémájával. A **pedagógus és a tanuló kapcsolata** sem hagyományos tanár–diák viszony, hiszen a kreatív gondolatok születésében és realizálásában a felnőtt együtt dolgozik a gyermekkel, de nem irányítóként, hanem munkatársként.

A Logo filozófiája a **modulszerűen felépített** programozás: az egyes elemekből „épül fel" a végső program. A programírás és -építés tehát logikai egységekre bontható, tisztán alkalmazható az alulról felfelé, illetve a felülről lefelé való programtervezési stratégia. A Comenius Logo segítségével, Windows környezetben, az eredeti Logo lehe-

tőségei megsokszorozódnak: a képernyőn megjelenő alkotás varázslat, amelynek megálmodója és létrehozója a gyermek, és ez az, ami oktatási alkalmazását feltétlenül indokolja.

A LOGO PROGRAMOZÁSI NYELV ELMÉLETI ALAPJAI

A programozási nyelveket úgy érthetjük meg igazán, ha a nyelv „filozófiai" alapjait vizsgáljuk meg, azaz azt a végrehajtási mechanizmust (számítási modellt), amit a program mögé elképzelünk. Ebben az értelemben a Logo jelentősen eltér a hagyományos, Neumann-elvű programozási modelltől, két ettől lényegesen különböző koncepcióra építkezik. Az egyes Logo-verziók ezt a két modellt különböző szinten valósítják meg, s amikor ettől eltérnek, annak komoly módszertani problémái vannak. Az alábbi gondolatok célja a tiszta alapok áttekintése, a Logo-szerű programozási filozófia megértetése.

1. AUTOMATAELVŰ NYELVEK

Az automataelvű nyelven írt programok végrehajtójának egy automatát (pl. ipari robotot, festőautomatát stb.) tekintünk. Tipikus példája a Logo programozási nyelv. Ebben a nyelvben legalább egy (de az igazi megvalósításokban sok) automata dolgozik, a teknőc – a rajzolóautomata.

Az automata rendelkezik **állapotokkal**, valamint az **állapotok közötti átmenetet leíró függvényekkel**, műveletekkel. (Ilyen állapothalmaz lehet például a festőautomatánál a festőkar helye, a festékszóró iránya, a festék színe, a festékszóró bekapcsoltsága stb.) (A klasszikus véges automatát a bemenő jelei és állapotai halmazával, valamint az állapotátmenet-függvénnyel adjuk meg. Ehhez jön a kimenő jelek halmaza és kimenettel rendelkező automaták esetén a kimenetet kiszámító függvény.)

Egy összetett állapot **állapotkomponensekből** áll, s az egyes műveletek valamely állapotkomponens értékét változtatják meg.

Az automata fizikai szerkezetéből következően itt nem szükséges a hagyományos értelemben vett **változófogalom**, a változók száma, típusa, elnevezése egyrészt rögzített (**ezek az állapotkomponensek**), másrészt pedig utasítások, eljárások paraméterei lehetnek.

A rögzített nevű állapotkomponensekre általában **állapotmódosító**, illetve **állapotlekérdező** utasítások léteznek (pl. a festőautomata karja mozduljon el az aktuális helyzetéből 30 cm-rel előre, milyen az aktuális festék színe stb.). Hagyományos értelemben vett értékadás nincs. Természetesen definiálni kell az automata kezdeti vagy **alapállapotát**, s egy vagy több utasítást, amely egyes állapotkomponenseket, vagy az összeset alapállapotba hozza.

A paraméterek **érték szerinti** paraméterek, értéket eljárások hívásakor kapnak, s utána csak felhasználjuk őket.

A beolvasást helyettesíti az **eljárások paraméterezése**, valamint az **állapotlekérdezés** (pl. a festőautomata festékszórója előtt van-e festendő felület), az automata tehát bemenő jeleket képes feldolgozni. A bemenet értelmezhető a **környezet állapotának lekérdezéseként** (a festőautomata így lekérdezheti, hogy van-e előtte festenivaló, milyen messze van a festőkartól stb.) (A Logók többségéből ez a lehetőség hiányzik!)

Kiírásra sincs szükség, hiszen az eredmény a program végrehajtása során előálló **állapotváltozás nyoma** (a festőautomatánál a befestett tárgy). A kiírás eredményeként megváltozik a környezet, ami az újabb állapotlekérdezésre (bemenetre) már hatással lehet.

A program az állapotoktól határozottan elkülönül (az első ipari automaták lyukszalagvezérlésűek – NC – voltak, amin adatot nem tároltak).

Az **elágazások, ciklusok paraméter-**, illetve **állapotfüggőek** lehetnek, azaz általában jóval egyszerűbbek a Neumann-elvű nyelvekben megszokottaknál. A paraméterfüggő ciklus csak egy primitív, ciklusváltozó nélküli, adott lépésszámú ciklus lehet, minden más feladatra *rekurzív*, azaz önmagát hívó eljárást célszerű írni.

Az automataelvű nyelvek természetes módon képesek a párhuzamos végrehajtásra. Egy automataelvű rendszerben több automatát is definiálhatunk, s ezek bemenő, illetve kimenő jeleiken keresztül egymással kapcsolatba léphetnek. Ez a lehetőség is kihasználatlan még a Logo-verziók széles körében.

2. FUNKCIONÁLIS NYELVEK

Itt egy pontos matematikai kidolgozottságú nyelvosztályról: a *függvényszerű* nyelvekről lesz szó. Elképzelésünk szerint a **program egy függvény**, s a program végrehajtása a függvény kiértékeléséből áll.

A függvényszerű nyelvek utasításkészlete is eltér a szokásostól. Az alapfogalom itt a **kifejezés**. Az eljárások mindig függvények, s programstrukturálásra a következők állnak rendelkezésre:

$-\ f(x) := g \circ h(x)$ – függvénykompozíció

$$-\ f(x) := \begin{cases} g(x) & ha \quad p(x) \\ h(x) & ha \quad \neg p(x) \end{cases}$$ – alternatív függvény

$$-\ f(x) := \begin{cases} g(x) & ha \quad p(x) \\ h \circ f \circ i(x) & ha \quad \neg p(x) \end{cases}$$ – rekurzív függvény (h, f és i kompozíciója)

Ebben a modellben **nem léteznek változók**, csupán függvényparaméterekről beszélhetünk. Ezek egyszer – a függvényhíváskor – kapnak értéket, s a későbbiekben csak felhasználjuk őket. Ezzel szemben konstansokról beszélhetünk, hiszen a függvényszerű nyelvek konstansai a **konstansfüggvények** (amelyeknek nincs paraméterük).

A kifejezések írásakor néhány funkcionális nyelv kifejezetten ragaszkodik a függvényszerű felíráshoz (pl. A+B helyett +(A, B)), mások megengedik a kétoperandusú operátorok használatát is.

Függvényparaméterként használhatunk újabb függvényeket is. A függvényparaméterek kétféleképpen értékelhetők ki. Az egyik a szokásos **összetett függvénykiértékelési** módszer, melyben először mindig a belső függvény értékét számoljuk ki, s annak eredményére alkalmazzuk a külső függvényt (vö. érték szerinti paraméterátadás). A másik a **késleltetett függvénykiértékelés**, melyben a külső függvény meghívásakor a belső függvényértéket még nem számítjuk ki, hanem csak ott, ahol a külső függvény kiszámításában felhasználjuk az értékét (vö. név szerinti paraméterátadás).

Ha nincs változó, akkor persze **nem beszélhetünk értékadásról** sem. A megőrzendő értékeket függvényparaméterként tárolhatjuk, így amire többször van szükség, azt vagy rekurzívan számítjuk ki, vagy pedig újabb függvényhívás paraméterének adjuk.

A függvényérték éppen ezek miatt a legtöbb más nyelvvel ellentétben nem csupán elemi típusú lehet, hanem tetszőleges összetett típus is.

A beolvasást e nyelvekben a függvényparaméterezés helyettesíti, kiírás helyett pedig a kiszámított függvényérték automatikus kijelzése történik. (Bár beolvasásra és kiírásra elvileg nincs szükség, a funkcionális nyelvek többsége – a felhasználóval való kapcsolattartás minősége miatt – mégis definiál beolvasó és kiíró függvényeket. Ekkor a beolvasó függvény egy paraméter nélküli, *á*-konstansfüggvény, a kiíró függvény pedig egy függvényérték nélküli, de speciális mellékhatással rendelkező függvény lesz.)

A funkcionális programozási nyelvek lényeges eleme azonban, hogy a függvény értékét kizárólag a paraméterei határozzák meg, a mellékhatás fentitől eltérő szerepét ezek a nyelvek szigorúan kerülik.

A COMENIUS LOGO 3.0

A Comenius Logo a Pozsonyi Comenius Egyetem Informatika Oktatási Tanszékének 1992 óta fejlesztett terméke. Szerzői: Andrej Blaho, Ivan Kalas és Péter Tomcsányi. Ez a legújabb Logo-változat a több erőforrással rendelkező számítógép (pl. a multimédia) lehetőségeit megpróbálja teljes mértékben beolvasztani a gyerekek által is könnyen kezelhető számítógépes nyelv világába. A világ több országában (Anglia, Ausztria, Belgium, Brazília, Bulgária, Csehország, Görögország, Hollandia, Lengyelország, Németország, Portugália, Svájc) megvásárolt és 1995 szeptemberétől véglegesített, nemzetközileg elterjedt verziót a legelismertebb Logo-változatként emlegetik világszerte.

Az alapvető Logo utasításkészleten kívül maximum 4000 teknőc mozgatása lehetséges, amelyek animációs módban (a kép több fázisa váltogatható, ezzel keltve mozgásérzetet) is működtethetők. Az alakzatok egyes fázisainak megtervezésére kiegészítő képsorszerkesztő szolgál. A kép- és vektorműveletekkel gazdagított sorozat- és rekordkezelés, szövegablak, sokoldalú színkezelés, zeneszerkesztő, WAV- és AVI-állományok, Windows-programok lejátszása mind újdonság más elterjedt Logo nyelvjárásokhoz képest. Talán csak a teknőc térbeli mozgatásának lehetősége hiányzik belőle.

A Logo régen áhított eszköz a magyar közoktatásban. Az érdeklődés legfőbb oka, hogy a Logo programnyelvi szinten „gyermekközpontú", vagyis a gyermekek nemcsak mint felhasználók találkozhatnak programokkal, hanem könnyen megtanulhatják a Logo programozási nyelvet is – hiszen nekik alkották meg. Emiatt kifejlesztettek már grafikákat, hangokat, animációt és multimédiát kezelő nyelvi elemeket. A Comenius Logo Windows környezetben fut, ami megkönnyíti elterjesztését és felhasználását.

A Comenius Logo kitűnő eszköz a NAT-ban kitűzött számos követelmény elérésére. Számítógépes nyelvként (kicsiktől a nagyokig) és az oktatási mikrovilágok fejlesztésére alkalmas szerzői rendszerként is megállja a helyét. Az informatikai ismeretek elsajátításában, sőt más műveltségi területek tananyagának kísérletező, alkotó elsajátításában is szerepet vállalhat.

A Logo nyelv gyermekek számára létrehozott programozási nyelv, segítségével kisgyerekek is könnyen kifejezhetik gondolataikat, algoritmusaikat, megalkothatják modelljeiket. A Comenius Logót minden ország ellátta anyanyelvi szókészlettel is, mert a gyermekek anyanyelvükön tudják magukat a legtermészetesebben kifejezni. A magyar verzió lehetővé teszi a magyar szókészlet használatát, és természetesen a menük, a hibajelzések és a segítségnyújtásra szolgáló szövegek is magyarul szerepelnek. Mindezek mellett az angol utasításkészlet is használható, az azzal készített programok más nemzeti nyelvű verzióban is működnek.

A COMENIUS LOGO FUTTATÁSÁHOZ SZÜKSÉGES KÖRNYEZET

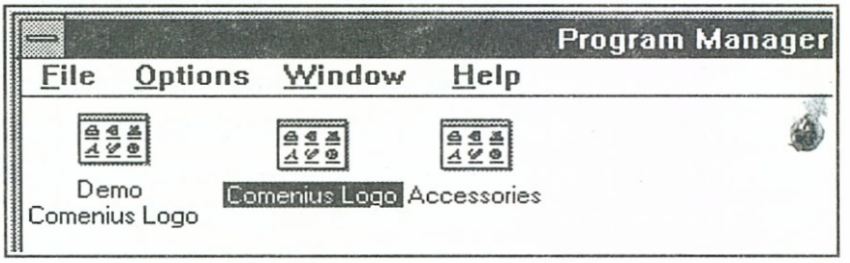
386/SX vagy annál fejlettebb (ajánlott legalább 486-os) konfiguráció, VGA grafikus kártya, 4MB RAM, 6 MB merevlemez-terület. Windows 3.1 vagy annál újabb verzió.

Soundblaster kompatíbilis hangkártyával a zenei kiegészítés minden lehetősége élvezhető.

A COMENIUS LOGO ELINDÍTÁSA ÉS A DEMOJÁTÉKOK

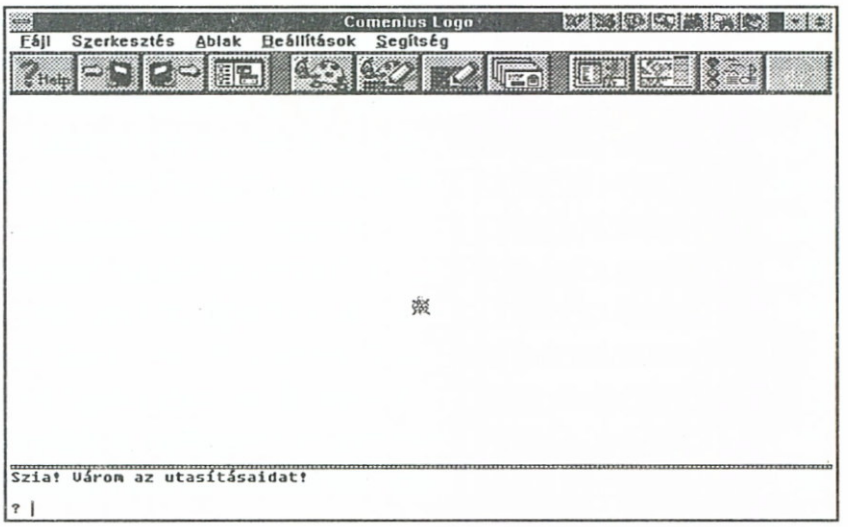
A COMENIUS LOGO ELINDÍTÁSA

Indítsuk el a Windows programot! Kattintsunk a **Comenius Logo** csoportikonra!



Kattintsunk a **Comenius Logo** ikonra!

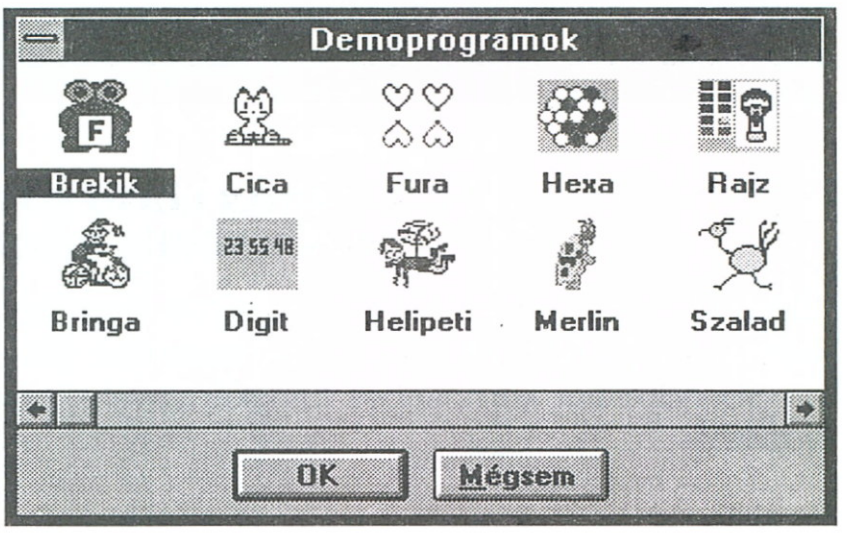
Elindul a **Comenius Logo**, és a következő képernyő lesz látható.



A DEMOJÁTÉKOK

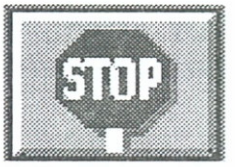
Kattintsunk a következő ikonra:

Ez itt a **Demoprogramok** ablaka. Bármelyik programot elindíthatjuk, ha a képére kétszer rákattintunk. Megjelennek a Comenius Logóval készült demonstrációs programok. Próbáljuk ki őket!

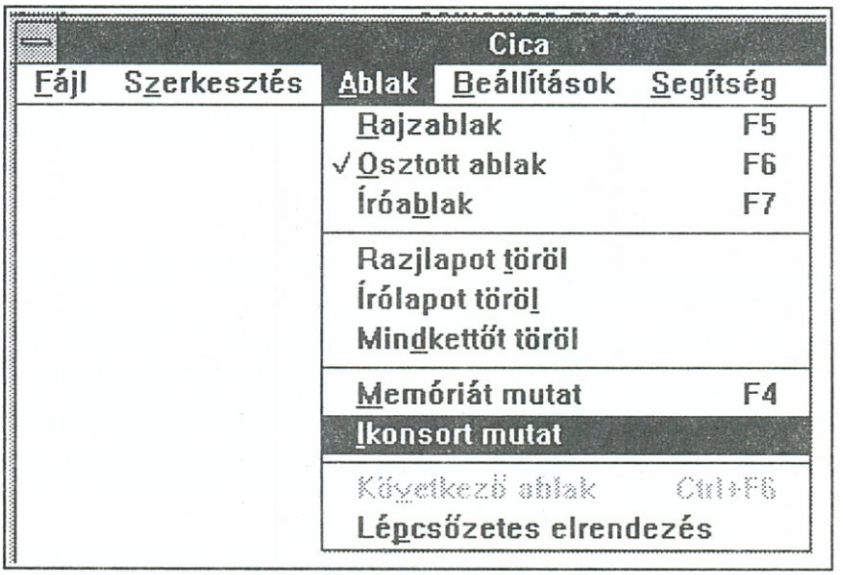


Programot lehet választani úgy, hogy kétszer rákattintunk annak ikonjára, vagy egyszeri kattintás után az OK gombra. A programok a betöltés után azonnal elindulnak és további útmutatásokat adnak. Kattintsunk az ikonsoron a **Rajz** ikonra:

A programokat a **Stop** gombon kívül az **F12** gombbal lehet megállítani.



Ha az ikonsor eltűnt, akkor az **Ablak** menüpontban kattintsunk az **Ikonsort mutat** sorra, és az újból látható lesz.



Ha a teknőc nem látható, akkor nyomjuk meg a szürke + gombot!

A PROGRAMCSOMAG RÉSZEI

Comenius Logo: a Comenius Logo programozási nyelvet értelmező környezet, a **Comenius Logo** ikonra kattintva indítható el.

Képszerkesztő: képek vagy képsorozatok létrehozására és módosítására alkalmas szerkesztő, a **Képszerkesztő** ikonra kattintva indítható el.

Kímélő: a létrehozott Logo programokat képernyőkímélőként futtató segédprogram. A **Kímélő beállítás** ikonra kattintva indítható el az előkészített programok képernyőkímélőként való használata. Természetesen csak olyan Logo projektet érdemes képernyőkímélőként használni, amely nem igényel gombnyomást! Az elkészített Logo projektet, amely ikont is tartalmaz, tegyük a **PROJEKT\KIMELO** alkönyvtárba. Ettől kezdve a projekthez tartozó ikon a Kímélő beállítás megnyitásakor láthatóvá válik és kiválasztható lesz, mint Windows-kímélő.

Gyerekjátékok: kisebbeknek szóló játékcsomag, a **Gyerekjátékok** ikonra kattintva indítható el.


Gyerekjátékok


Gyerekjátékok beállítás

A **Gyerekjátékok-használat** ikonra való kattintással egy szöveges állomány nyílik meg, melyben részletes leírást olvashatunk a játékok használatáról.


Gyerekjátékok-használat

Fordítóprogram: a Comenius Logo régebbi verziójában íródott programokat az új verzió utasításaira fordítja át.


Fordító


Fordító-használat


Alapszavak

Információ: Általános információkat tartalmaz a programcsomagról.


INFO

ISMERKEDÉS:

A programcsomaghoz ízelítőként egy feladatsor készült, példaprogramokkal együtt, melyet az eszközsor 8. – feladatlapokat ábrázoló – ikonjára kattintva érhetünk el. Kilenc egymás utáni lépést követve, a többi feladatot jobbára tetszőleges sorrendben végrehajtva, alapvető ismereteket szerezhetünk a magyar Comenius Logo használatáról.

A COMENIUS LOGÓBÓL INDÍTHATÓ JÁTÉKOK (DEMOPROGRAMOK):


Brekik

A békák növekvő sorszámú elrendezését szorgalmazó játék.


Bringa

Egy kis demonstráció egyszerű, de látványos animációra.


Cica

Egérmozgató ügyességi játék.

Digit

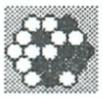
Digitális órát szimuláló példaprogram.


Fura

Érdekes tükörhatást eredményező rajzolóprogram.

Helipeti

A Comenius Logo egyik érdekes hatását illusztráló program.



Hexa

„Tedd egyszínűvé!" játék a hexagonokra kattintással.

Merlin

Mozaikos képkirakó játék.

Rajz

Háttérképek (rajzlap) rajzolására és festésére alkalmas eszköz.

Szalad

Demonstrációs program sok szereplő létrehozására és működtetésére.

Szedd

Virágszedő ügyességi játékprogram.

Tiktak

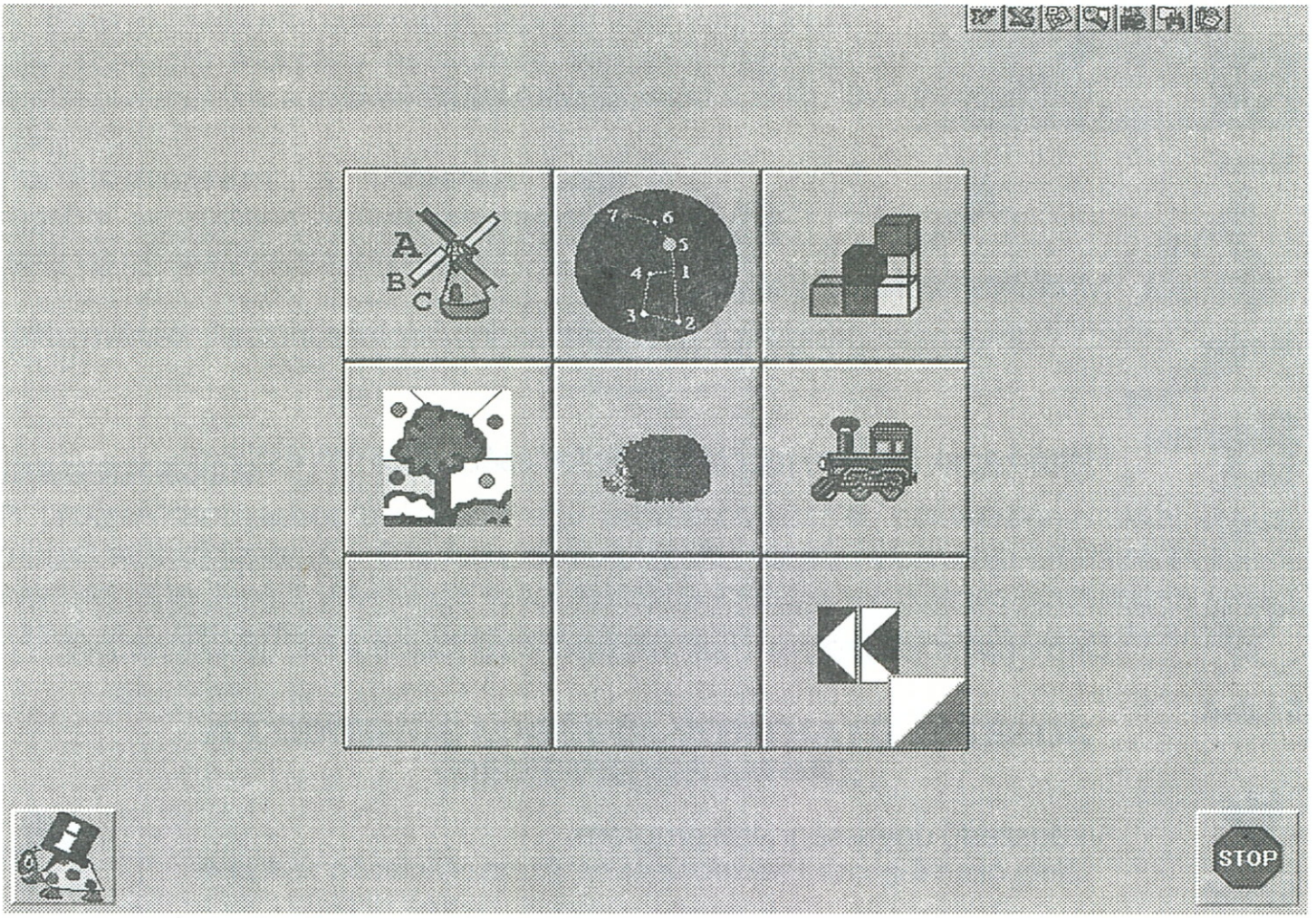
Mutatós órát szimuláló példaprogram.

Utca

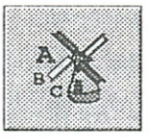
Egyszerű, de látványos animációt demonstráló program.

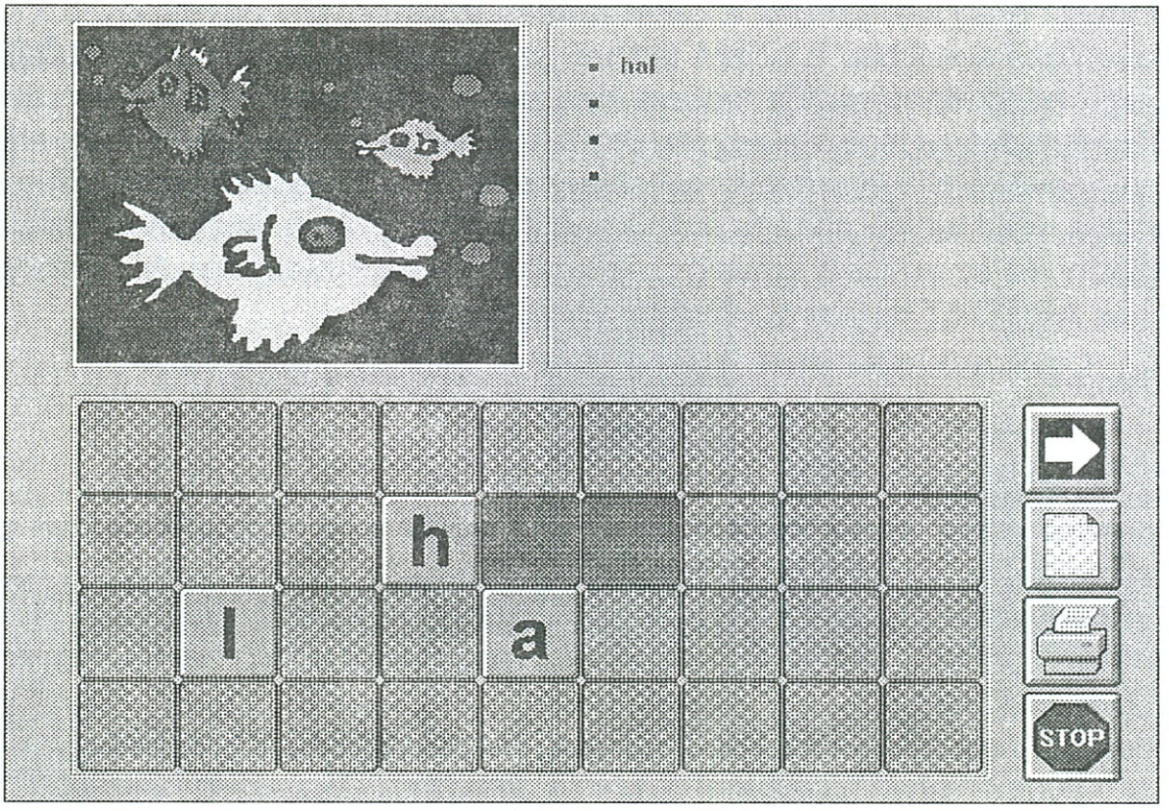
A **GYEREKJÁTÉKOK** a  ikonra való kattintással indul.

Gyerekjátékok



A GYEREKJÁTÉKOK program olvasni, írni tanuló 4–6 éves gyerekek számára készült, Comenius Logo nyelven. Hat kis játékból áll, melyek a gyerekek olvasási készségét és logikai gondolkodását fejlesztik. A feladatok az egyszerűbbektől az összetettebbek felé haladnak, hogy a gyerekek figyelmét egyre jobban lekössék, és a megoldáshoz szükséges lépések természetesen adódjanak számukra. Az olvasni tanuló gyerekek számára főként a szókirakó játék hasznos. A vasútépítő játék egyes feladatai sok felnőttet is elgondolkoztatnak. Különösen kiemelendő egyes játékokban, hogy a feladatokat maguk a gyerekek hozhatják létre és adhatják fel egymásnak.

A **SZÓKIRAKÓ JÁTÉK**, a  ikonra kattintva indul.



Adott egy 4×9-es négyzetháló. A program megad egy – a négyzetháló feletti képhez kapcsolódó – szót vagy mondatot, majd ennek betűit összekeverve elhelyezi ebben a négyzethálóban. A játékos feladata az, hogy az összekevert betűkből kirakja a szót vagy mondatot a kijelölt helyen. A betűket vízszintes vagy függőleges irányban egyenként lehet mozgatni, mégpedig a következő módon: kattintsunk az egér bal gombjával a mozgatni kívánt betűre, majd kattintsunk arra a négyzetre, ahová a betűt tennénk. Ha a betű útját elállja egy másik betű, akkor az csak a másik betűig tolódik el. Miután sikerült a betűket a megfelelő helyre mozgatni, a program ezt jelzi, ha viszont valamely betűket felcseréltük, akkor azokra színkiemeléssel hívja fel a figyelmünket.

A program által használt szavakat és mondatokat a GYEREK alkönyvtárban található ABC.DAT állomány tartalmazza. Ez az állomány egy egyszerű szöveg, amelyet bárki átszerkeszthet (például a Windows Jegyzettömbjében), tehát mi magunk is feladhatunk egymásnak különböző feladványokat.

Mielőtt az állomány átszerkesztéséhez kezdenénk, ajánlatos róla egy másolatot készíteni!

Az ABC.DAT állomány szerkezete a következő:

Az első sor: SHOW IT = YES vagy SHOW IT = NO.

- YES esetén a program a kirakandó szót kiírja a képernyőre.
- NO esetén nem írja ki a program, hogy melyik az a szó vagy mondat, amelyet ki kell rakni.

Ezt a beállítást a program futása közben a CTRL+B billentyűk segítségével módosíthatjuk. Természetesen a CTRL+B használatakor az ABC.DAT állomány nem módosul.

Az első sor után üres sornak kell következnie. Az állomány további része tartalmazza a feladványokat: szavakat és mondatokat, valamint képeket és hangfájlokat. Ennek a résznek a következőképpen kell kinéznie:

Minden képhez tartozhat valamennyi (legalább egy és legfeljebb nyolc) szó vagy mondat. Először meg kell adnunk a képet tartalmazó állomány nevét egy külön sorban, majd a következő sorokban a hozzá tartozó szavakat és mondatokat. Minden szóhoz tartozhat egy hangállomány. Ez kétszer hangzik el: először akkor, amikor a program megjeleníti a feladványt, majd akkor, amikor a játékos helyesen megoldotta a feladatot. Az állomány nevét a hozzá tartozó szóval egy sorban kell megadni, attól pontosvesszővel elválasztva. Például, ha a kutya.bmp képpel a „kutya" és a „kutyusok" szavakat szeretnénk feladni, és az említett szavakhoz a kutya.wav és kutyus.wav állományok tartoznak, akkor az ABC.DAT tartalma a következő legyen:

```
kutya.bmp
kutya; kutya.wav
kutyusok; kutyus.wav
```

Az egyes csoportokat (képet tartalmazó állományokat és a hozzájuk tartozó szavakat) egy üres sorral kell egymástól elválasztani.

Mivel a négyzetháló mérete 4x9-es, egy sorba legfeljebb 9 betű kerülhet. Egy-egy feladvány több szóból is állhat. Ha a szavakat egymástól sorközzel választjuk el, azok külön sorba fognak kerülni.

Előfordulhat, hogy szeretnénk két, értelmileg összetartozó szót (például egy főnév és a jelzője vagy névelője) egy sorba tenni. Ekkor az aláhúzás (_) karaktert használjuk a két szó elválasztására. Például ha a „Ma jó idő van." mondatban a „Ma jó" és az „idő van" szavakat egy-egy sorban szeretnénk látni, akkor a következőt kell írnunk a megfelelő sorba:

```
Ma_jó idõ_van.
```

Ne kerüljön egy sorba 9 betűnél hosszabb rész (beleértve a szóközöket is), és ne legyen 4-nél több sor.

A szavakban a magyar karakterkészlet betűi használhatók, a nagy- és kisbetűket megkülönböztetjük. Használhatók még a pont, vessző, kérdőjel és felkiáltójel karakterek. Ha a szavakban kettősbetűt használunk, akkor azok a játékban automatikusan egy négyzetre kerülnek. Ha két betűt – melyek egyébként kettősbetűt alkotnának – mégis külön négyzetekre szeretnénk rakni, akkor a # karakterrel kell őket elválasztani. Például az „igazság" szót a következőképpen írjuk le:

```
igaz#ság.
```

A KÉPÁLLOMÁNYOKRÓL A KÖVETKEZŐKET KELL TUDNI

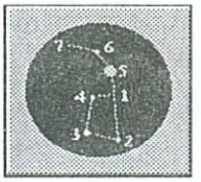
Ha az állomány nevét útvonal nélkül adjuk meg, akkor annak abban a könyvtárban kell lennie, ahol az egész projekt van. Ha útvonallal együtt adjuk meg, akkor a program az adott könyvtárban fogja keresni. Az egyes alkönyvtárakat két \\ jellel kell egymástól elválasztani, a Logo ugyanis csak így tudja értelmezni. Például, ha a cica.bmp állomány a c:\windows\paintbrush könyvtárban van, akkor a megfelelő sorba a következőt írjuk:

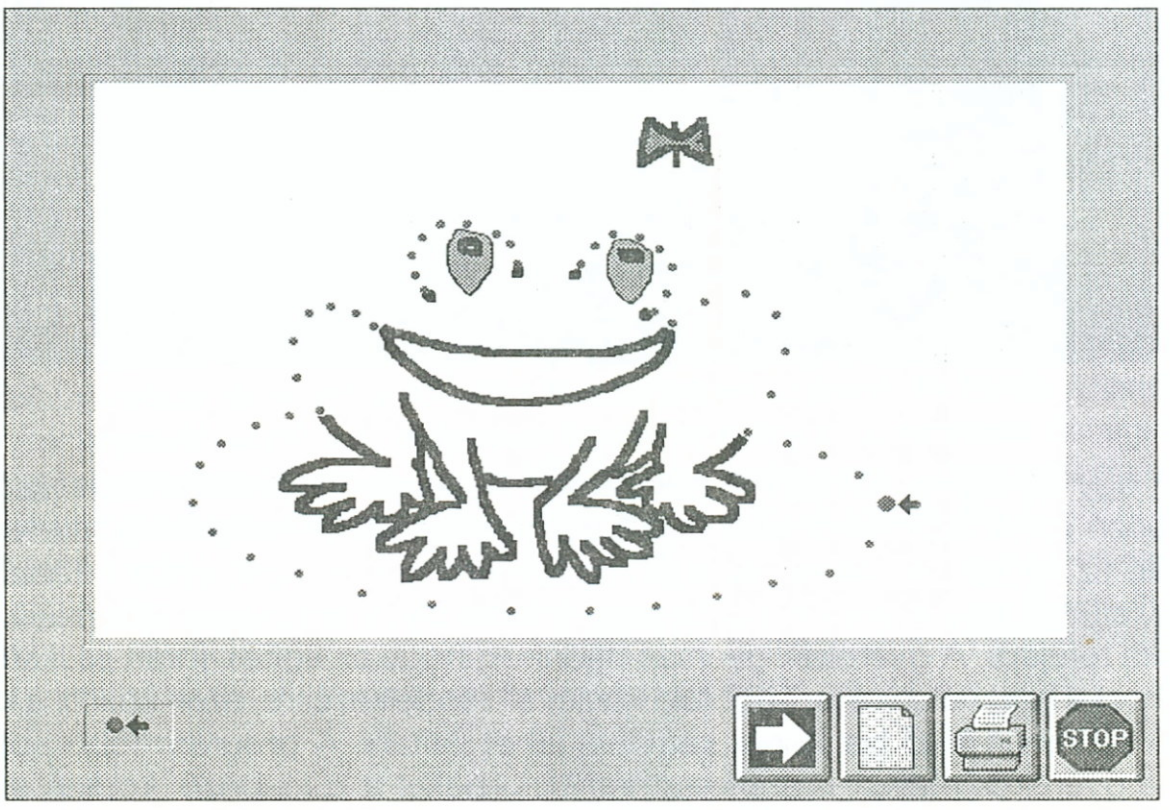
```
c:\\windows\\paintbrush\\cica.bmp
```

Ha a megadott állomány mérete a 246×204-et nem haladja meg, akkor a kép ráfér a kijelölt mezőre, ha kisebb a megadott méretnél, akkor a mező közepén helyezi el a program. Ha a képfájl mérete 246×204-nél nagyobb, akkor a program méretre vágja úgy, hogy a bal felső sarka fog teljes egészében látszani. Használhatunk .LGW állományokat is (ez a Comenius Logo standard képsorainak kiterjesztése). A program mindig csak a képsor első fázisát fogja használni.

A HANGÁLLOMÁNYOKRÓL

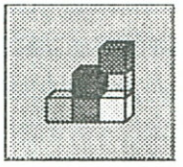
Ha útvonal nélkül adjuk meg, akkor a program ott keresi őket, ahol maga a projekt van. Megadhatjuk a teljes útvonalat is. Ha a játék beállításainál a „Hang bekapcsolva" gombot bekapcsoljuk, akkor a program minden feladványnál kétszer szólal meg. A hangot a program futása közben is kikapcsolhatjuk a CTRL+S billentyűk segítségével. Újabb CTRL+S hatására a hang visszatér.

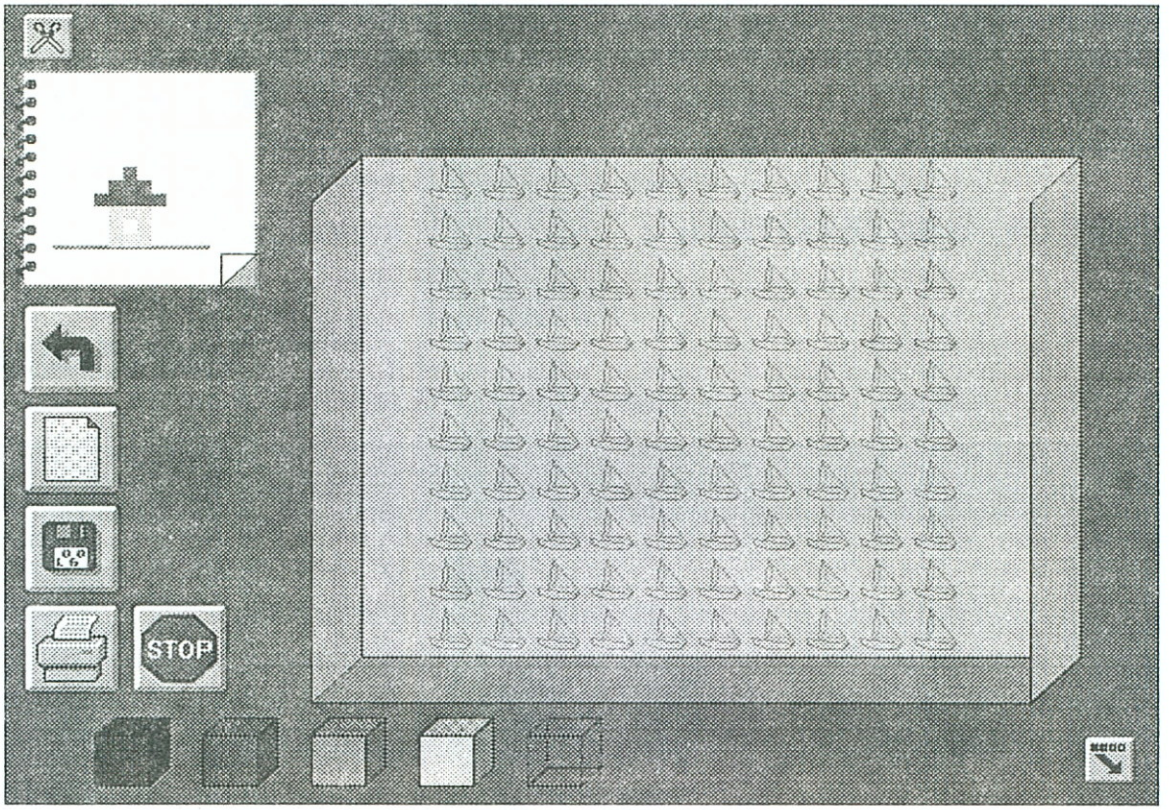
A **PONTSOROK ÖSSZEKÖTÉSE** a  ikonra kattintva indul.



Ez a játék a rejtvényújságokból jól ismert. Lényege, hogy a gyerekek a betűk, számok és más objektumok sorba rendezését ismerjék fel. Ha a szétszórt és különböző módon (betűkkel, számokkal stb.) megjelölt pontokat a megfelelő sorrendben összekötjük, valamilyen kép alakul ki. A pontok összekötéséhez az egérrel rákattintunk a sorba rendezés szerinti első, majd a rendezés szerinti következő pontra. Ekkor a két pont között a program egy szakaszt húz. Miután az összes pontot összekötöttük, egy képet kapunk, amelyet a program kiszínez, sőt egy kis animációt is láthatunk.

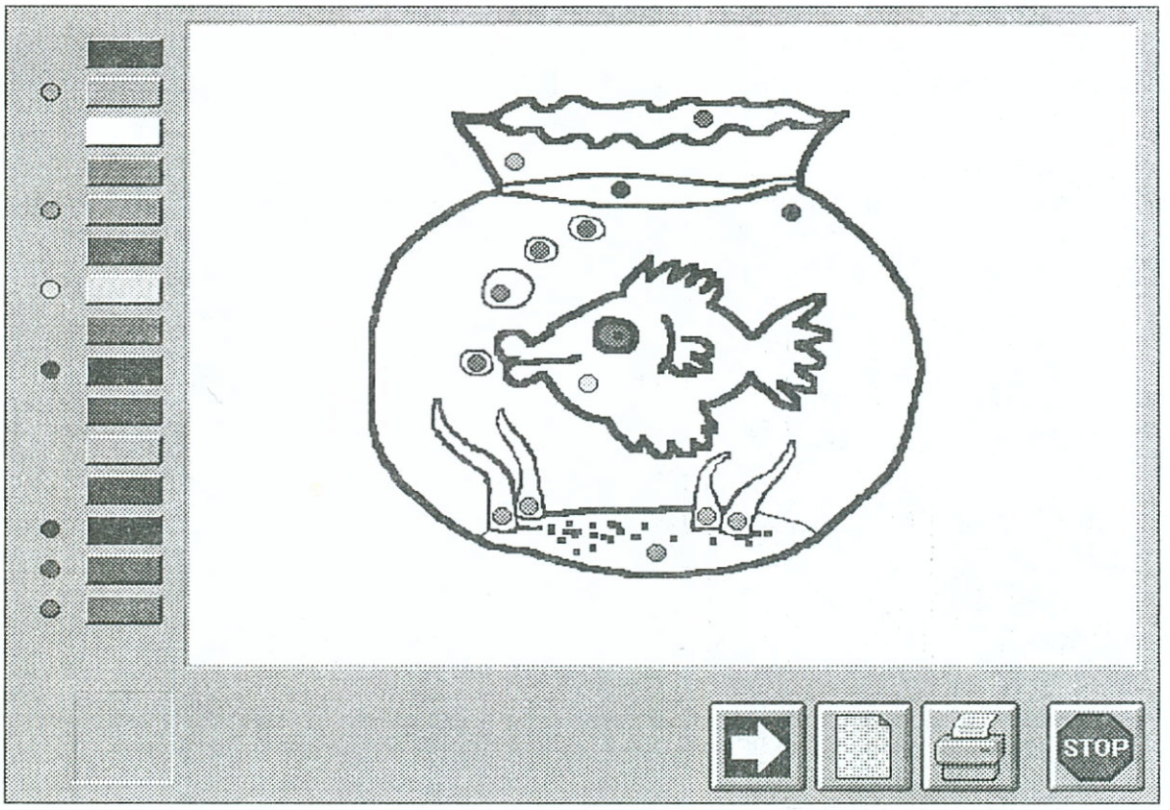
A könnyebb feladatokat egyre nehezebbek követik, s a gyerekek anélkül is kitalálják, mit kell csinálniuk, hogy erre részletes utasításokat kapnának.

A **KOCKAKIRAKÓ JÁTÉK** a  ikonra kattintva indul.

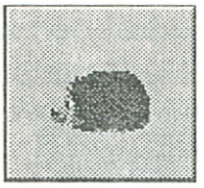


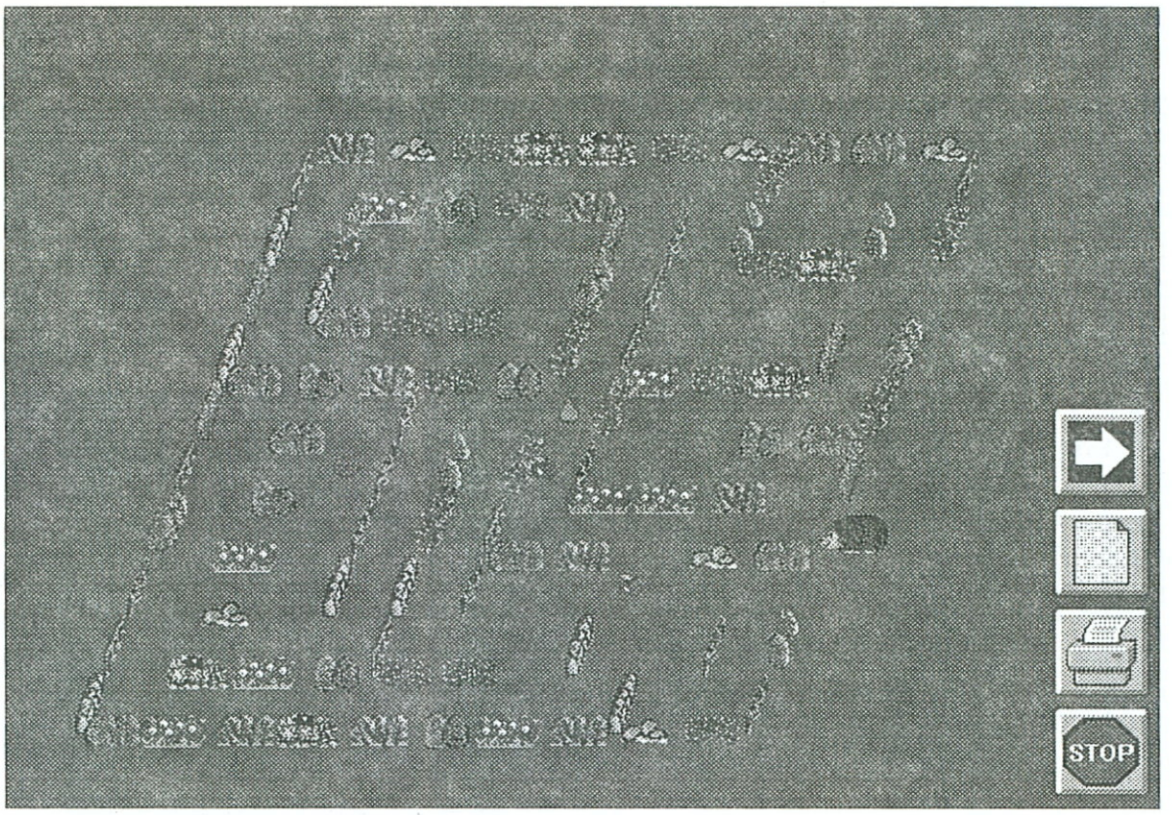
Adott egy kis színes, kockákból kialakított mozaikszerű kép. Ezt a képet kell nekünk is különböző színű kockákból felépíteni. Egy olyan táblában dolgozunk, melyben egymás mellé tíz kockát rakhatunk, és egymás tetejére szintén tízet. Ebbe a táblába kell a kockákat felülről „bedobálni". A kocka vagy a tábla aljára esik, vagy valamelyik, már ott lévő kocka tetejére. A gyerekek gyorsan rájönnek, hogy az építés során szükség van az átlátszó kocka használatára is. Többféle kocka áll rendelkezésre, ezek között a képernyő jobb alsó sarkában lévő kis ikonra kattintva lehet váltani. A program több ábrát tartalmaz, de a gyerekek maguk is szerkeszthetnek képeket a különböző kockák segítségével, és ezeket el is menthetik a lemezt ábrázoló ikonra kattintva. Az elmentett ábrák beépülnek a többi ábra közé. A kis ollóra kattintva az éppen aktuális ábrát törölhetjük.

A **KIFESTŐ JÁTÉK** a  ikonra kattintva indul.

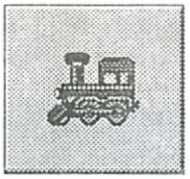


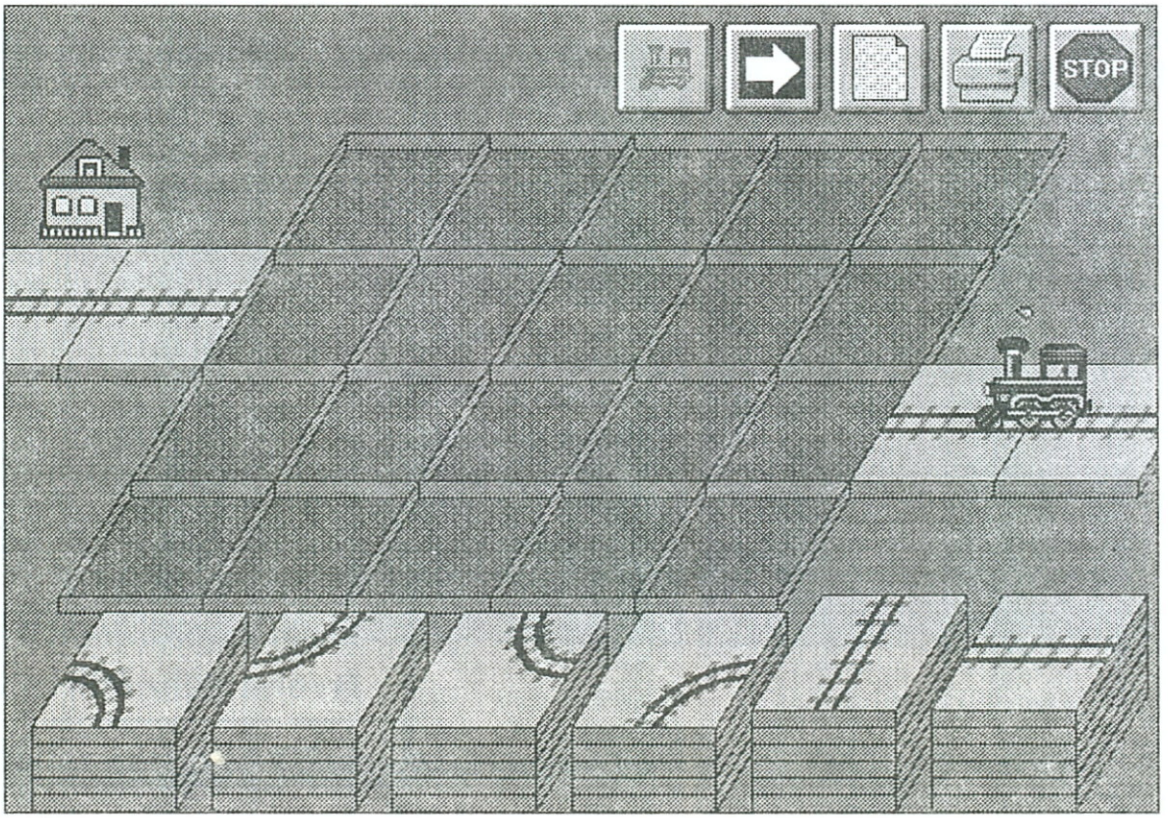
A képernyőn látható képet, a megadott színekkel kell kiszínezni. A színeket kis színes körök, zászlócskák, majd a további feladatokban számok és betűk azonosítják. Az ecsetet a megfelelő színű kis téglalapra kattintva mártjuk a festékbe és egy területet szintén egérkattintással festhetünk be. Ha jó színnel festünk, akkor a képen eltűnik a színt azonosító jel. Miután kifestettük a képet, kis animációt láthatunk.

A **GYÜMÖLCSÖSKERT** a  ikonra kattintva indul.



Adott egy kis kertlabirintus gyümölcsökkel. Az a feladat, hogy a süni segítségével összeszedjük a terméseket. Egyszerre csak három gyümölcsöt bír el a süni, ezért minden harmadik felszedett gyümölcs után ki kell őt vezetni a kertből. A sünit úgy irányítjuk, hogy az orra előtt húzzuk az egérmutatót, amit követ. A gyümölcsök szedése a gyümölcsön való áthaladással történik. A későbbi feladatokban előírt a gyümölcsök szedési sorrendje, a továbbiakban pedig a kertben még vermek is vannak, amelyekre lépve sünink verembe esik. Egy még nehezebb változatban a vermek rejtettek.

A **VASÚTÉPÍTÉS** a  ikonra kattintva indul.



Adott egy terepasztal, és meghatározott számú, különböző alakú sínelem. A sínelemekből össze kell rakni a vasúti pályát, hogy a vonat a pályaudvarra mehessen. A sínelemek kiválasztása és elhelyezése a megfelelő sínelemre és tereppontra kattintással történik. Ha a kis vonatot jelző ikon kiszíneződik, akkor útjára indíthatjuk a vonatot. Ha a pálya nem teljes, akkor a vonat az eddig elkészült pályaszakasz két végpontja között ide-oda mozog.

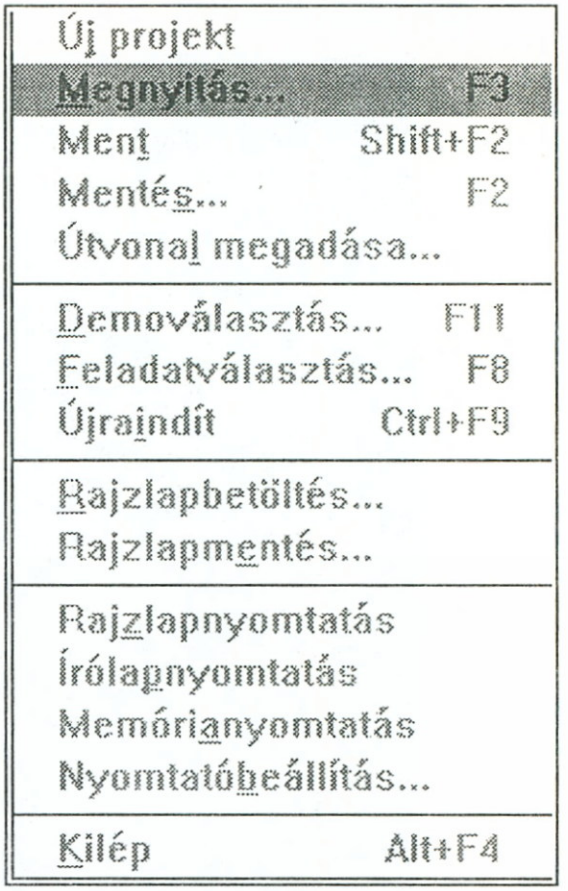
A vasútépítés a leginkább érdekes, gondolkodást igénylő feladatcsoport, némelyik sínelrendezés megoldása komoly fejtörést okoz még a felnőtteknek is. Lehet, hogy némelyik esetben a feladat hosszas gondolkodás után is megoldhatatlannak tűnik. Természetesen mindegyik feladatnak van legalább egy – némelyiknek több – megoldása. Az egyik feladatban például csak 5 sínelem van, holott a terepasztalt látva nyilvánvaló, hogy legalább 6 elem kell egy teljes pálya kiépítéséhez. Ekkor trükk alkalmazására van szükség: a vonatot már a félig felépített vasútvonal elkészülése után elindítjuk, majd felszedjük mögötte a síneket, s a megfelelő darabokat elé rakjuk.

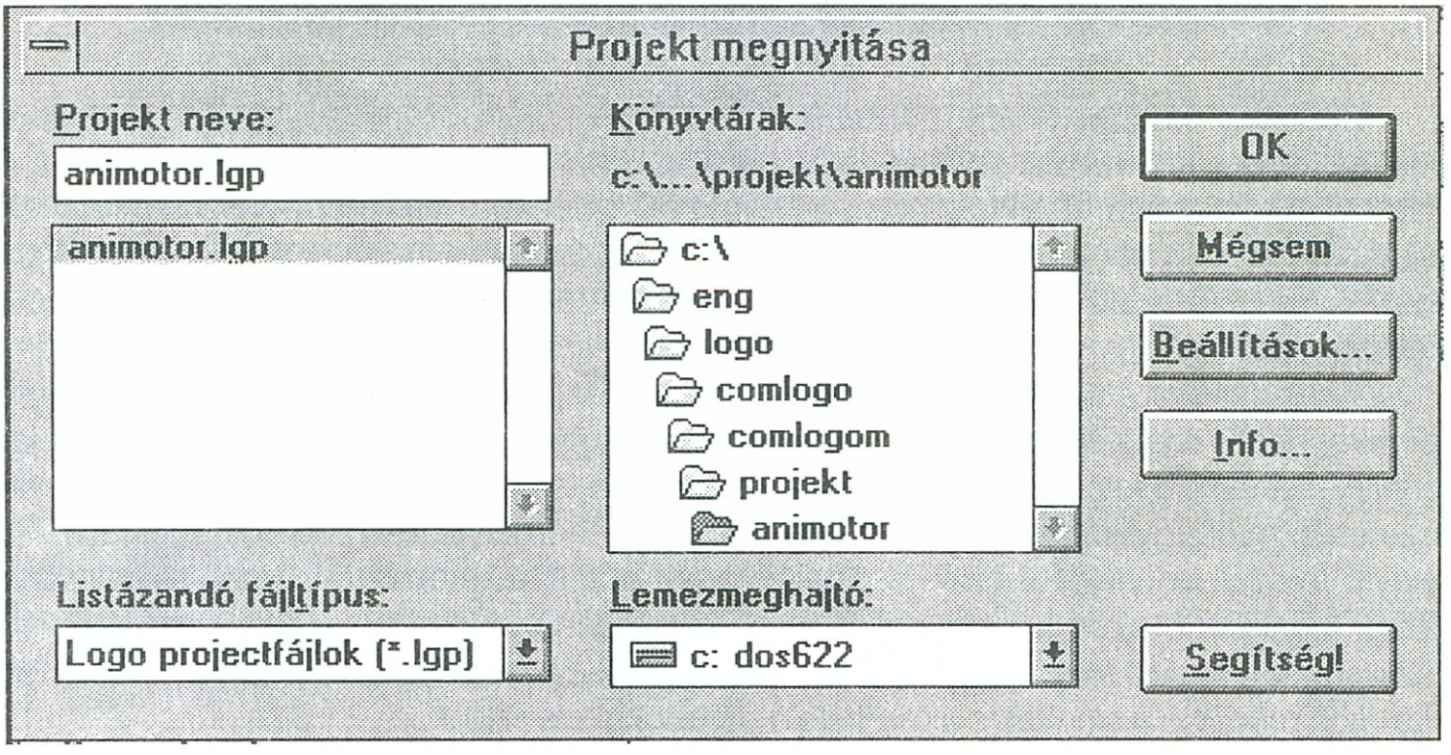
ANIMOTOR

A  ikonra kattintás hatására elindul a Comenius Logo.

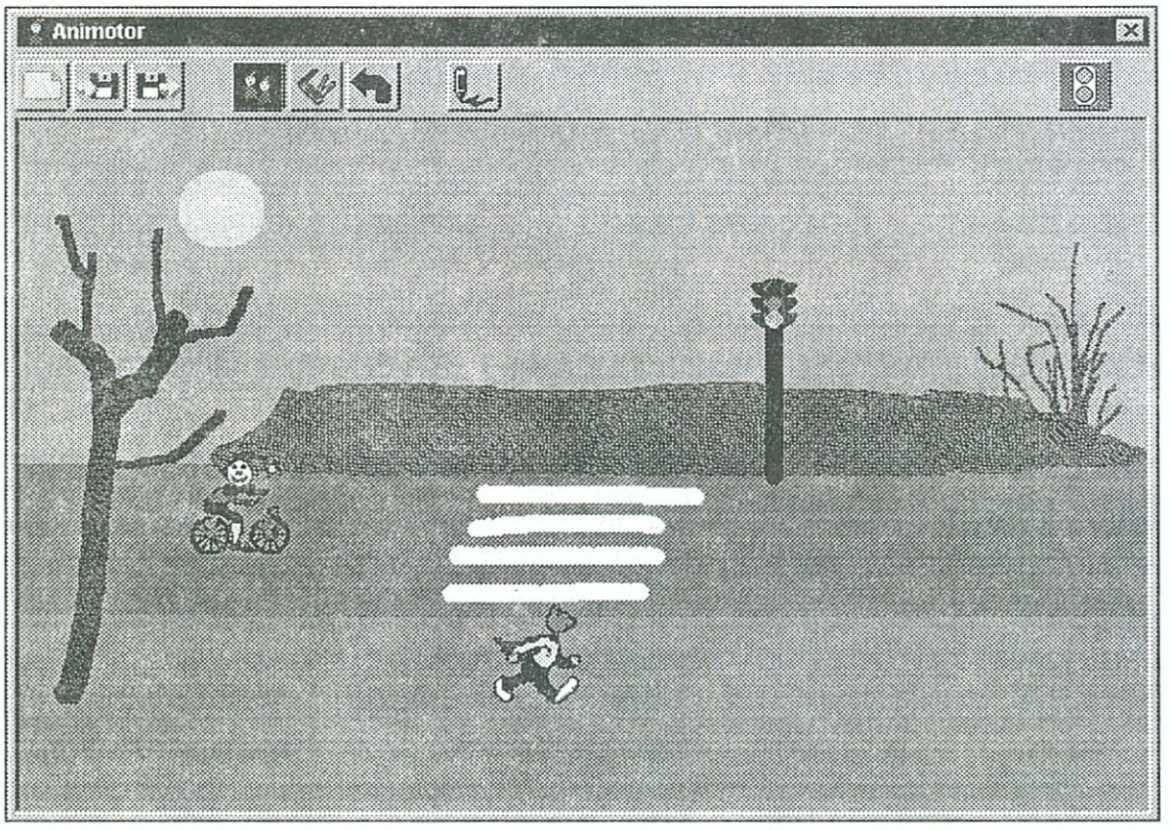
Comenius Logo

A Megnyitás sorra kattintva kapjuk az alábbi ablakot:

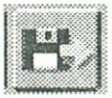




Ha az **OK** gombra kattintunk, betöltődik az **Animotor** program.



A beszédes ikonsortól útbaigazítást kaphatunk.

 rákattintva behívhatunk egy demonstrációs példát: **zebra.lgp**

AZ ANIMOTOR HASZNÁLATA

F1 lenyomására újra előjön a **Segítség** ablaka.
Új szereplő hívható elő, ha az **Új szereplő** ablakból kivonszoljuk a kiválasztottat az egér segítségével.

A szereplőhöz tartozó események és a hozzájuk rendelt utasítások fejlesztése úgy történik, hogy a szereplőre a jobb egérgombbal kattintunk. A fejlesztésnél az **Események** ablakból kell egy sort kijelölni, majd az esetleges paraméterek meghatározásával a hozzárendelt utasítást kell kiválasztani az **Utasítás** ablakból.

AZ ESEMÉNYEK, ILLETVE UTASÍTÁSOK MAGYARÁZATA

ESEMÉNYEK:

`klikk [<utasítás>]` – bal egérgomb lenyomása

`ütközés <szereplõ> [<utasítás>]` – a szereplő ütközése másik szereplővel

`távolság <pont> <szám> [<utasítás>]` – a szereplő távolsága a ponttól a számnál nagyobb

távolság <szereplõ> <szám> [<utasítás>] – a szereplő távolsága másik szereplőtől a számnál nagyobb
belül <terület> [<utasítás>] – szereplő a kiválasztott területen belül van
ablakban [<utasítás>] – a szereplő az ablakon belül van
gomb <gomb> [<utasítás>] – a kiválasztott gomb lenyomása
színen <szín> [<utasítás>] – a kiválasztott színhez ért a tollpont
kívül <terület> [<utasítás>] – a szereplő a kiválasztott területen kívül van
ablakon.túl [<utasítás>] – a szereplő az ablakon kívül van
állapotom <állapot> [<utasítás>] – a szereplő állapota a választásnak megfelelő
állapota <szereplõ> <állapot> [<utasítás>] – a kiválasztott szereplő állapota a választásnak megfelelő
idõ <idõ> [<utasítás>] – az időegység vagy időintervallum letelt
bent.pattan <terület> – a kiválasztott területen belül pattan
ablakban.pattan – az ablakon belül pattan
kint.pattan <terület> – a kiválasztott területen kívül pattan

UTASÍTÁSOK:

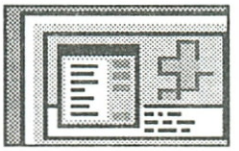
vissza – 180 fokos fordulat
balra – balra 90 fok
jobbra – jobbra 90 fok
irányában <pont> – a szereplő iránya a pont felé **nézzen**
irányában <szereplõ> – a szereplő iránya egy másik szereplő felé nézzen
irány <szám> – a szereplő iránya a megadott fokú legyen
fütyül – fütyülő hangot adjon
hanghullám <wav-állomány> – a kiválasztott hanghullám lejátszása
beszéd <szám> [<szöveg>] – adott időtartamig a szöveg a beszédablakban jelenik meg
állapot <állapot> – állapot beállítása
haza – a létrehozás helyére menjen
<utasítás> <utasítás> – utasítás beszúrása
<semmi> – utasítás törlése

ELEMI ALAKZATOK RAJZOLÁSA

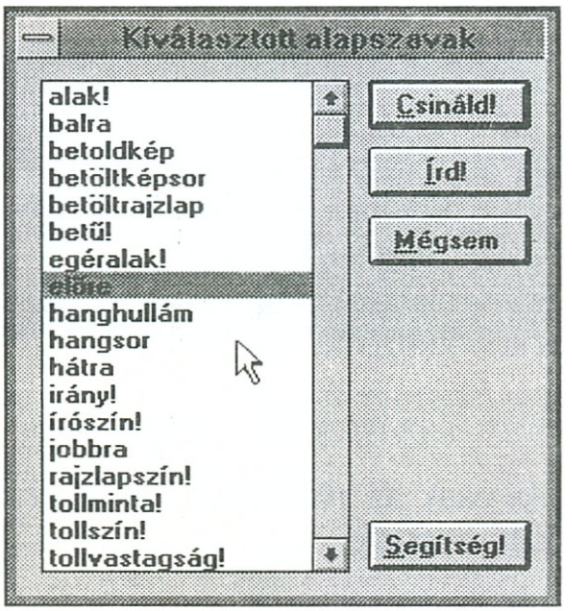
Indítsuk el a Comenius Logót! Az ismerkedést kezdjük új lappal: a **Fájl** menüben kattintsunk az **Új projekt** pontra! Ekkor tiszta lappal indulunk. A képernyő közepén rögtön látható lesz a teknőc. Neki kell megmondanunk, merre haladjon, mit csináljon.

Milyen utasításokat adhatunk? Csak olyat, amit a teknőc megért.

Kattintsunk a következő ikonra.



Egy ablakot kapunk, amelyben egypár kiválasztott alapszó szerepel. Egy utasítást a nevére kattintással választhatunk ki. Ott van például az előre szó. Ha rákattintunk, majd pedig a **Csináld!** gombra, akkor egy párbeszédablak jelenik meg a képernyőn, amelynek segítségével a teknőc üzenetet küld: meg kell mondani neki, hogy mennyit menjen előre. Itt egy számot vár, majd egy kattintást a **Csináld!** gombra. Az így beírt utasítások megjelennek a képernyő alján látható írólapon. A későbbiekben közvetlenül ezt fogjuk használni.



A teknőcnek szóló utasítások, mint például az **előre**, az írólapra is beírhatók.

```
? előre
```

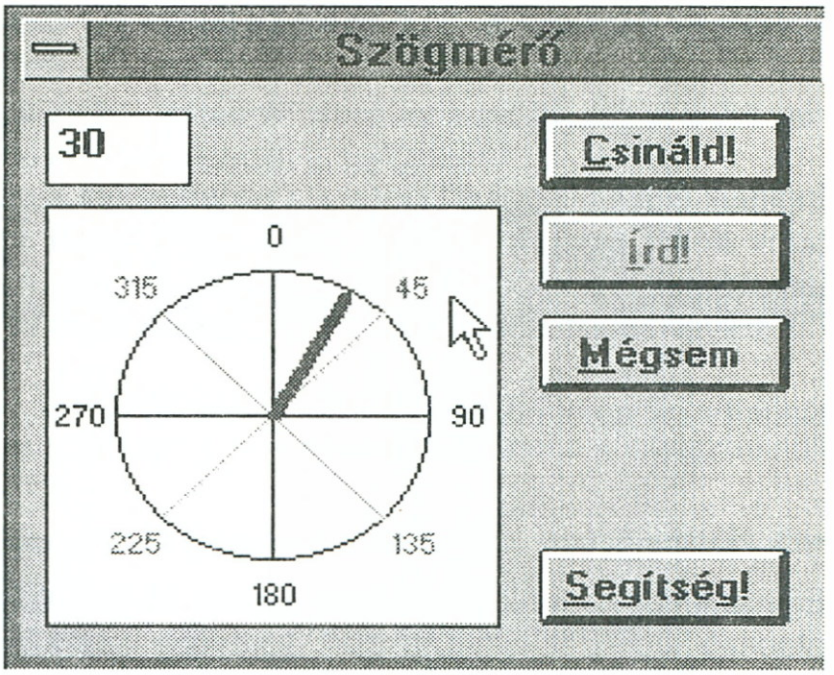
A szó beírása után az **Enter** billentyű is lenyomható, de ekkor a teknőc egy számot vár. Segítségül egy vonalzót helyez a rajzlapra, hogy a vonalzó beosztására rákattintva kiválasztható legyen lépésének mértéke. (Fordulásoknál a vonalzó helyett szögmérő jelenik meg.)

A teknőc az aktuális helyétől az aktuális irányba lép előre a megadott távolságra.

Írjuk most be a **jobbra** utasítást:

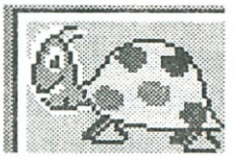
```
? jobbra
?
```

Jobbra forduláskor a szögmérő segédablakban a piros mutató segítségével állítható be a kívánt szög, vagy beírható a megfelelő érték a négyzetbe. Az elfordulás mértékét a teknőc aktuális irányához képest, az óramutató járásával megegyezően, fokban kell megadni.

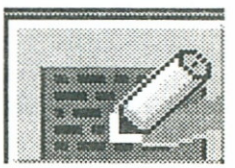


Az előzőekben beírt utasítássorokat elő lehet keresni a felfelé mutató kurzornyíl segítségével, sőt akár módosítani is lehet azokat.

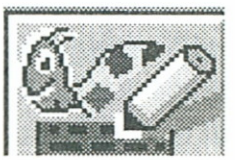
Ha a rajzablakot szeretnénk teljes képernyőként látni, akkor kattintsunk a következő ikonra:



Ha csak az íróablakot szeretnénk látni, akkor a következő ikonra kell kattintanunk:



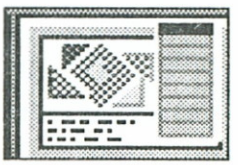
Ha újból osztott ablakot szeretnénk használni, ezt az alábbi ikonra kattintva érhetjük el:



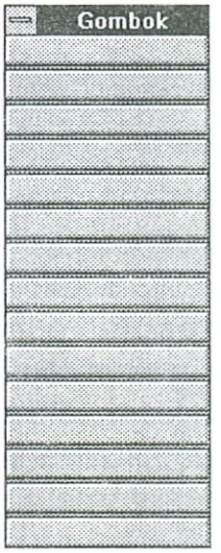
A rajzlap törlése választható az **Ablak** menüből, vagy törölhetünk a következő utasítás beírásával is:

```
? törölrajzlap
```

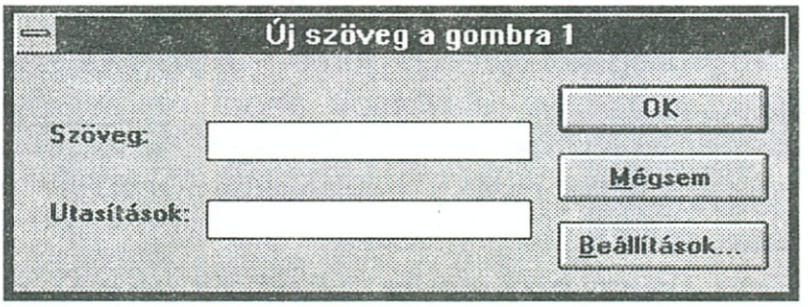
Az elemi alakzatok rajzolásának másik lehetősége egy rajzoló projekt készítése. Az utasítások rárakhatók egy-egy gombra, a gombokra pedig csupán rá kell kattintanunk ahhoz, hogy végrehajtódjanak. Kattintsunk a következő ikonra:



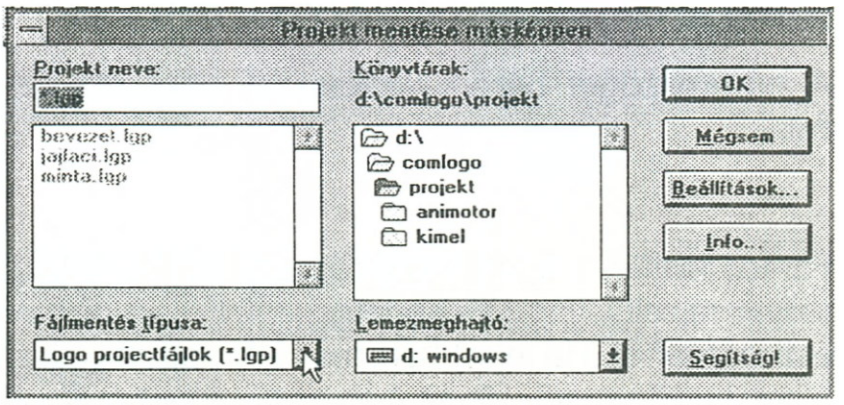
A következő gombsor jelenik meg, melynek gombjaihoz utasításokat rendelhetünk.



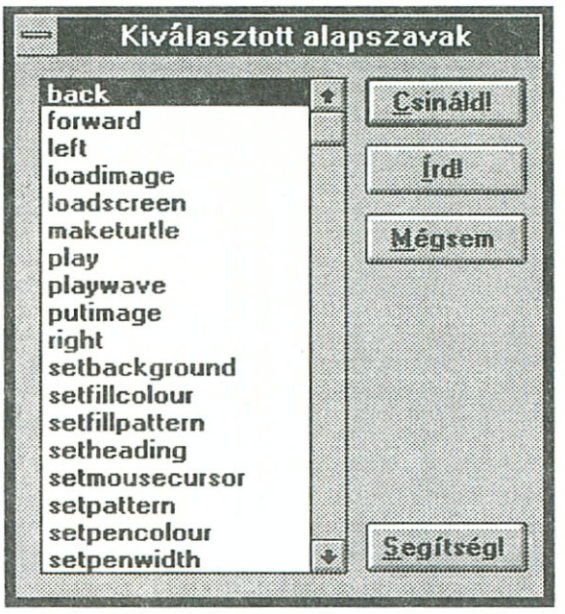
Egy tetszőleges gombra kattintva a jobb oldali egérgombbal, a kinyíló ablakban adhatók meg a gombok jellemzői. A *Szöveg* részbe írható be a gomb neve, az *Utasítás* részbe pedig az az utasítás (vagy utasítások), amit a gombhoz rendelünk, utána az **OK** gombbal kell a hozzárendelést jóváhagyni. Ha nem írunk semmit az *Utasítás* részbe, akkor a *Szöveg* részben szereplő névvel ellátott utasítás hajtódik végre. Ezután a gomb használható az utasítás szövegének beírása helyett.



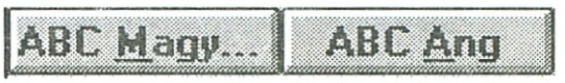
A **Fájl** menüben a **Ment** pont segítségével a hozzárendelés kimenthető. A megjelenő ablakban a *Projekt neve* keretbe kell írni a nevet, ami maximum 8 betű lehet. A kiterjesztés mindenképpen **.lgp** maradjon, mert csak így lehet megkülönböztetni a projekteket az egyéb típusú állományoktól.



Az írólap az a terület, ahol a kérdőjel látszik. Az utasításokat ékezetes betűkkel kell beírni! (A Comenius Logo az ékezeteket a 852-es kódtábla kiosztása szerint fogadja el.) Ha nincs lehetőség az ékezetes betűk beírására, akkor az angol utasításkészletet kell használni. A magyar utasításkészletet csak akkor ismeri a Comenius Logo, ha a **Beállítások** menüpontban a **Magyar alapszavak** sort kipipáljuk. Ha a magyar alapszavak nem érvényesek, akkor a memória ablakában az új szavak definíciói a **to** utasítássorral kezdődnek és az **end** utasítással fejeződnek be. Ebben az esetben a **Kiválasztott alapszavak** ablakban is angolul jelennek meg az utasítások.



A **Segítség**ben van egy gomb – **ABC Magy** –, amely a magyar utasítások angol megfelelőjét mutatja, és egy másik – **ABC Ang** –, amellyel az angol utasítások magyar megfelelője érhető el.



1. FELADAT: SZAKASZ, TÖRÖTT VONAL RAJZOLÁSA

Az utasítások végrehajtásához szükséges adatokat egymástól helyközzel elválasztva közvetlenül is felsorolhatjuk:

```
? előre 50 jobbra 90 hátra 30 balra 45
?
```

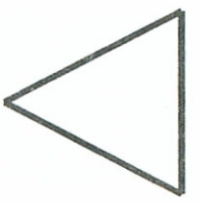
Az előzőekben beírt utasítássorokat elő lehet keresni a felfelé mutató kurzornyíl segítségével, sőt akár módosítani is lehet azokat.

2. FELADAT: HÁROMSZÖG RAJZOLÁSA

A háromszöget három szakasszal lehet megrajzolni:

```
? előre 100 balra 120
? előre 100 balra 120
? előre 100 balra 120
```

Fontos felhívni a figyelmet arra, hogy a 120 fokos fordulatot vagy meg tudjuk rajzolni, vagy kikísérletezhetjük – a Logo a matematika kísérleti eszköze lehet. Ha lehet háromszöget rajzolni, akkor csinálhatunk négyzetet, ötszöget stb., de szabályos tízszöget már valószínűleg senki sem szeretne így leírni.

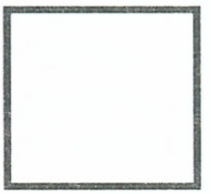


3. FELADAT: NÉGYZET RAJZOLÁSA CIKLUSSAL

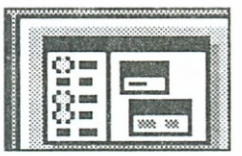
Ismerjük fel, hogy bármelyik sokszög rajzolásáról is van szó, mindig ugyanazt a két lépést kell ismételni: előre lépünk valamennyit, majd elfordulunk valahány fokot. Az ismétlés megvalósítására szolgál az **ismétlés** utasítás:

```
? ismétlés 4 [előre 100 jobbra 90]
?
```

Első paramétere az ismétlésszám, második paramétere pedig – szögletes zárójelben – a megismétlendő tevékenységek.

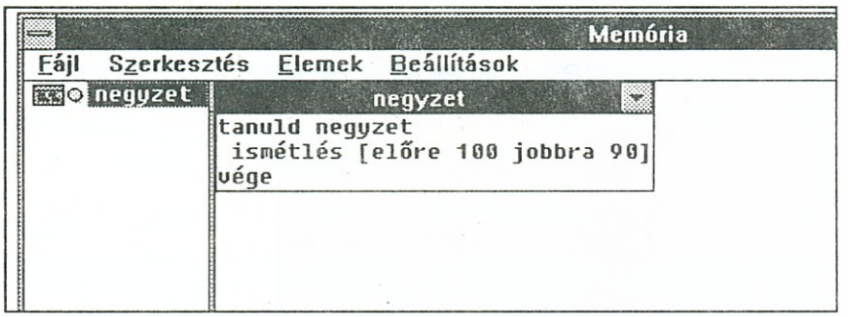


4. FELADAT: NÉGYZET RAJZOLÁSA ELJÁRÁSSAL, AVAGY ÚJ SZÓRA TANÍTJUK A TEKNŐCÖT



Előbb-utóbb fárasztóvá válik az ilyen hosszú parancssorok újbóli beírása. Emiatt feltehetjük a kérdést, nem tudjuk-e a teknőcöt megtanítani a négyzetrajzolásra. A teknőc utasításai egyszerű magyar szavak, így nincs más teendőnk, mint egy új szó jelentésére megtanítani. A tanuláshoz a következő ikonra kell kattintani:

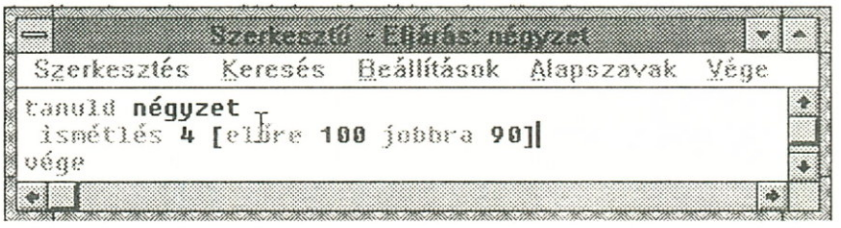
Ekkor felbukkan a tudást tároló **Memória** ablaka. Itt található minden új szó, amit megtanítunk. Az **Elemek** menüpont **Új eljárás** pontját kiválasztva taníthatjuk meg a teknőcöt egy szóra. A szó (azaz a tevékenységsorozat neve, amit hagyományosan eljárásnévnek hívunk) beírása után megjelenik az eljárás saját ablaka, ahova beírhatjuk a szükséges utasításokat (az első és az utolsó sor automatikusan megjelenik).



Minden új szó meghatározása a **tanuld** utasítássorral kezdődik, és a **vége** utasítással fejeződik be.

```
tanuld négyzet
  ismétlés 4 [előre 100 jobbra 90]
vége
```

A szerkesztés befejezésével a **Vége** gombra kattintva, majd a **Memória** ablakot becsukva, az új szó bekerül a szótárba, és használhatóvá válik.

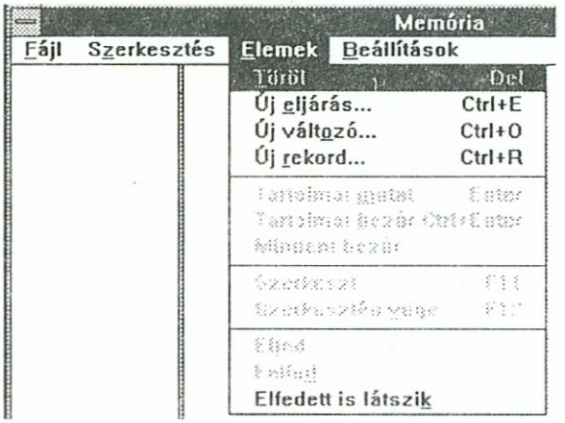


A megtanult új szó használata egészen egyszerűen a neve leírásával történik:

```
? négyzet
?
```

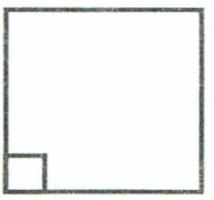
Tanulságként megfogalmazható, hogy a Logóban az eljárás egy természetes igényként felmerülő fogalom, aminek használata pont olyan, mint az alapszavak használata, azaz „megtanítása" után éppen olyan „értelmes" szavává válik a teknőcnek, mint az „eleve" ismert szavai.

A **Memória** ablakban az **Elemek** menüpont használható az eljárások megnézésére, módosítására, mentésre, betöltésre; itt jelölhető ki, hogy beíráskor az alapszavakat kiemelve szeretnénk-e látni stb.



5. FELADAT: NÉGYZET RAJZOLÁSA PARAMÉTERES ELJÁRÁSSAL

Ezután természetes módon merül fel a kérdés: ha az előre szó után írhatunk egy számot, ami az előrelépés mértékét jelenti, akkor a négyzet szó (ami a kibővült szókincs új eleme) után miért ne lehetne a négyzet méretét megadni? A válasz az, hogy természetesen lehet, erre azonban az eljárás megtanításakor fel kell készíteni a teknőcöt



.

A tanult tevékenységeket paraméterező adatokat kettőspont (:) karakterrel kell mindenhol bevezetni. Ez a jel a paraméter nevéhez szorosan hozzátartozik, csak lokálisan, az eljárás belsejében használható.

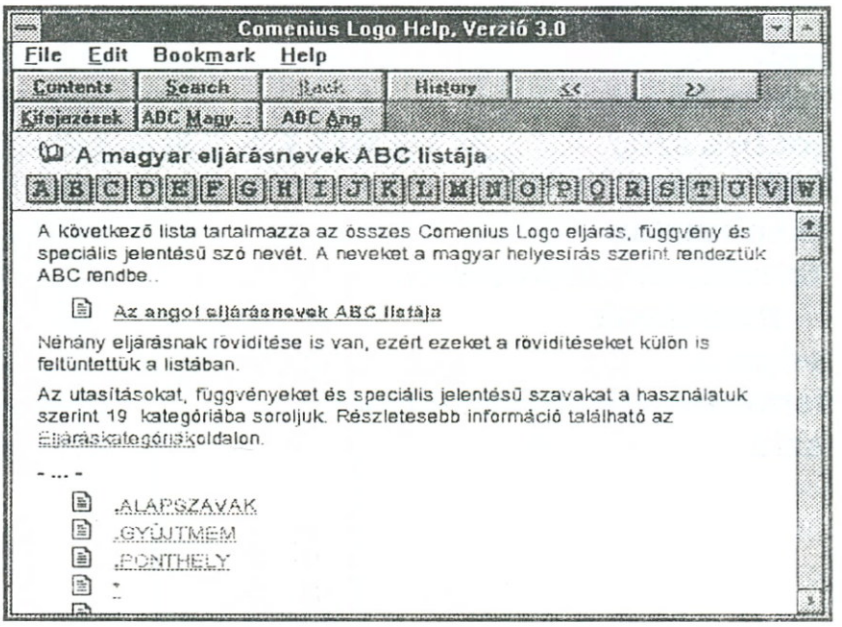
```
tanuld négyzet :hossz
  ismétlés 4 [előre :hossz jobbra 90]
vége
```

Ezután a most megírt „okosabb" tevékenység pontosan úgy használható, mint a teknőc által eddig ismert szavak:

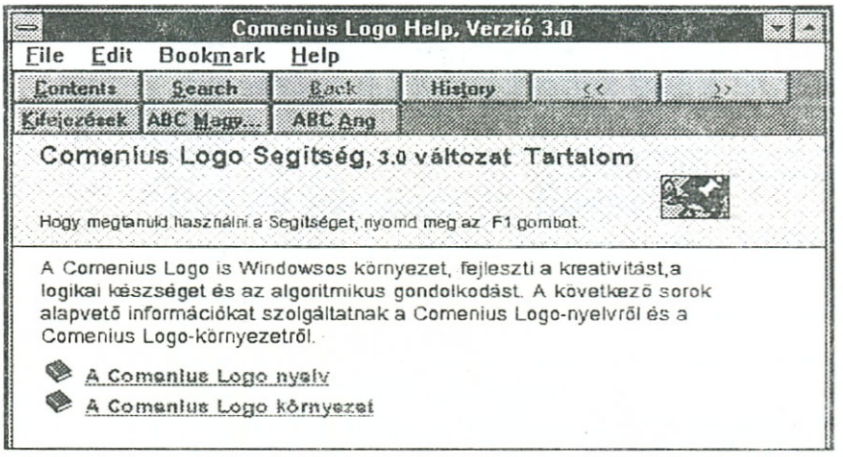
```
? négyzet 100 négyzet 25
?
```

TOVÁBBI UTASÍTÁSOK

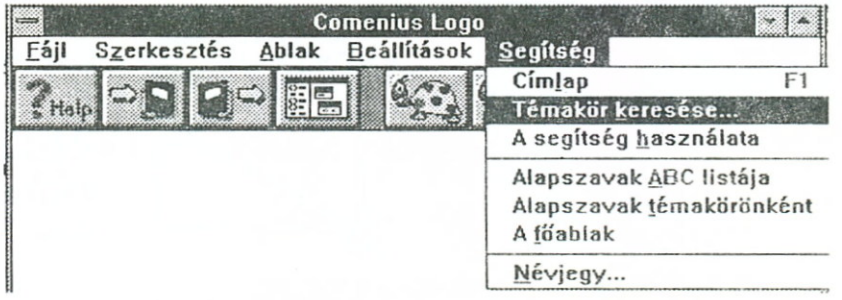
A többi alapszóval a **Segítség** menüpont **Alapszavak ABC listája** pont választásával ismerkedhetünk meg. Egy-egy utasítás használata és magyarázata a szóra való kattintásnál előbukkanó ablakban olvasható el.



A **Segítség** menüpont **Címlap** pontjára kattintva megismerkedhetünk a Comenius Logo nyelvvel és környezettel.



Ha egy konkrét alapszót vagy elnevezést keresünk, akkor a **Segítség** menüpont **Témakör keresése...** pontját választjuk a kifejezés keresésére.



A Logo használatának befejezésekor köszönjünk el tőle:

```
? szia
```

UTASÍTÁS-ÖSSZEFOGLALÓ

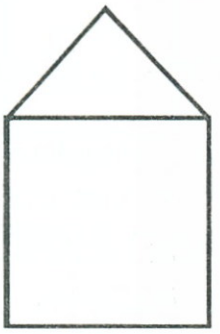
előre *hossz*	előrelépés *hossz* egységgel
hátra *hossz*	hátralépés *hossz* egységgel
jobbra *szög*	jobbra fordulás *szög* fokkal
balra *szög*	balra fordulás *szög* fokkal
törölrajzlap	rajzlap letörlése
tanuld *név paraméterek*	paraméteres eljárás definiálása
utasítások	
vége	
ismétlés *darab* [*utasítások*]	ciklus *darab*számszor
szia	a munka befejezése

LOGO GRAFIKA

ÉPÍTKEZÉS ALAKZATOKBÓL

1. FELADAT: EGYSZERŰ HÁZIKÓ

Az előzőleg megismert háromszög- és négyzetrajzoló eljárások alkalmazásával készítsünk egy egyszerű házikót. Íme a minta:



... és két ismert eljárás:

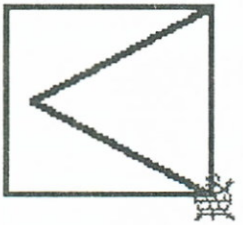
```
tanuld négyzet :hossz
  ismétlés 4 [előre :hossz balra 90]
vége
```

```
tanuld háromszög :hossz
  ismétlés 3 [előre :hossz balra 120]
vége
```

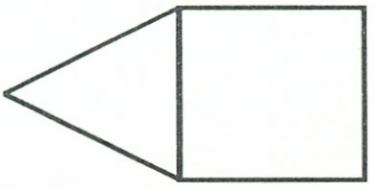
A házikót rajzoló eljárást a négyzet és a háromszög eljárásaiból „építhetjük" fel:

```
tanuld házikó :hossz
  négyzet :hossz háromszög :hossz
vége
```

A kapott eredmény azonban nem felel meg előzetes várakozásainknak:
Itt hívjuk fel a figyelmet arra, hogy a Logóban a hibák is érdekesek, lehet örülni nekik (szemben más programozási nyelvekkel).



Most a hiba oka az, hogy a háromszöget és négyzetet rajzoló eljárásokat rosszul építettük össze.

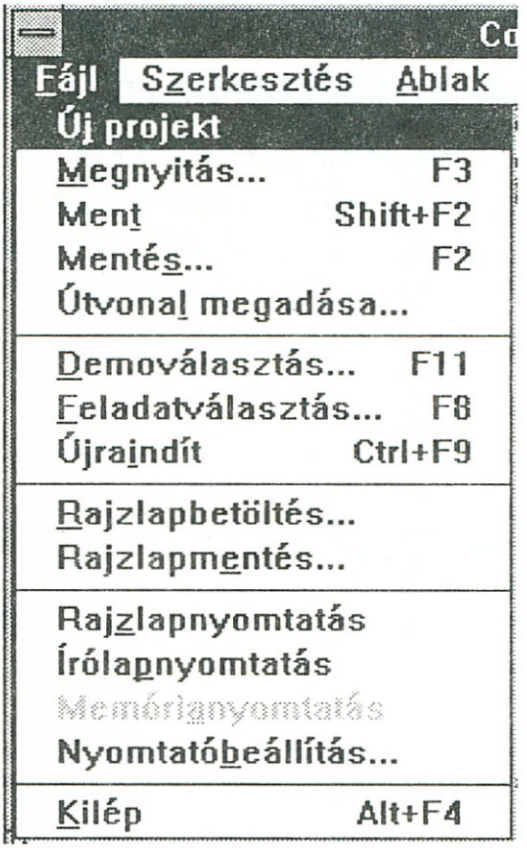


A helyes megoldás:

```
tanuld házikó :hossz
  négyzet :hossz előre :hossz balra 30
  háromszög :hossz jobbra 30
  hátra :hossz
vége
```

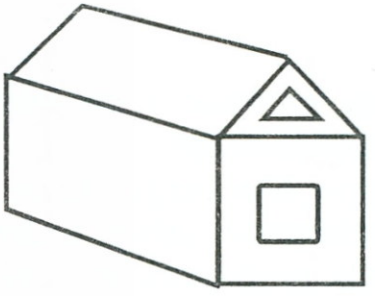
Fontos felhívni a figyelmet az *állapotátlátszóságra*, vagyis arra, hogy akkor egyszerű egy ábra elhelyezése, ha rajzolásakor a teknőc kezdő- és végállapota azonos. Ehhez – általános esetben – azt kell tenni, hogy minden állapotváltozásnak végrehajtjuk az ellentettjét, mégpedig az egyes változások fordított sorrendjében.

Az elkészült programok kimenthetők a **Fájl** menüpont **Ment** (az aktuális néven menti), illetve a **Mentés** (új névvel menti) almenüjében.
A **Megnyitás** almenüvel tölthetünk be kész programokat.
A **Rajzlapbetöltés**, illetve **Rajzlapmentés** menüpontok BMP formátumú állományok betöltésére, valamint kimentésére szolgálnak.



2. FELADAT: BONYOLULTABB HÁZIKÓ – ÉPÍTKEZÉS

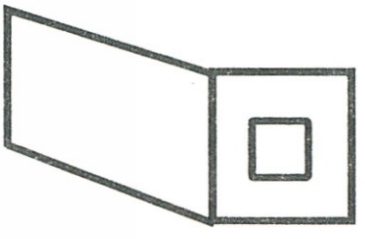
Házunkat bővítsük újabb alapelemekkel! Kiindulópontunk legyen a ház jobb alsó sarka, a kiinduló irány pedig mutasson felfelé!



Ebben az esetben jól meg kell terveznünk a ház felépítésének menetét. A ház földszintből és tetőtérből áll, mint az előbbi esetben. Íme a terv a teljes ház megrajzolásához:

```
tanuld ház :hossz
  földszint :hossz előre :hossz balra 30
  tetőtér :hossz jobbra 30
  hátra :hossz
vége
```

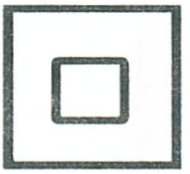
A földszint a ház elejéből, illetve oldalából áll:



Ezt a következőképpen valósíthatjuk meg:

```
tanuld földszint :hossz
  eleje :hossz balra 90 előre :hossz
  jobbra 90
  oldala :hossz balra 90 hátra :hossz
  jobbra 90
vége
```

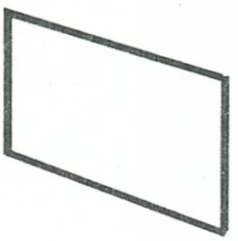
A ház eleje egy négyzetet és egy abban levő kisebb négyzetet tartalmaz. A belső négyzet megrajzolásához újabb utasításokra van szükség: a teknőc tollának felemelésére, illetve leengedésére.



Ezt a következő programsorokkal valósíthatjuk meg:

```
tanuld eleje :hossz
  négyzet :hossz
  tollatfel
  előre :hossz/3 balra 90
  előre :hossz/3 jobbra 90 tollatle
  négyzet :hossz/3
  tollatfel
  balra 90 hátra :hossz/3 jobbra 90
  hátra :hossz/3 tollatle
vége
```

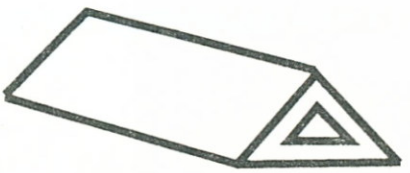
Vegyük észre, hogy az eljárás utolsó hat utasítása pontosan az ellentettje, mégpedig fordított sorrendben, a két sorral korábban szereplő hat utasításnak. Éppen ezzel érjük el, hogy visszajussunk a kezdőállapotba.



A ház oldala egy paralelogramma, melynek hosszabb oldala legyen a rövidebb oldal kétszerese, a kisebbik szöge 75, a nagyobb pedig 105 fokos:

```
tanuld oldala :hossz
  paralelogramma :hossz :hossz*2 75
vége
```

A tetőtér a homlokzatot és a tetőt tartalmazza, mely megvalósítható



... a következő megoldással:

```
tanuld tetőtér :hossz
  homlokzat :hossz
  balra 60 előre :hossz jobbra 120
  tető :hossz
  balra 120 hátra :hossz jobbra 60
vége
```

A homlokzat két háromszög egymás belsejében:

Megvalósítható a következő eljárással:

```
tanuld homlokzat :hossz
  háromszög :hossz
  tollatfel
  előre :hossz/3 balra 60
  előre :hossz/3 jobbra 60
  tollatle
  háromszög :hossz/3
  tollatfel
  balra 60 hátra :hossz/3
  jobbra 60 hátra :hossz/3
  tollatle
vége
```

A tető az oldalhoz hasonlóan egy paralelogramma, de itt a nagyobbik, 105 fokos szögnél kezdjük a rajzolást. A tető legyen piros színű, s dupla vonalvastagságú,

... amelyet megold a következő eljárás:

```
tanuld tető :hossz
  tollszín! 12 tollvastagság! 2
  paralelogramma :hossz :hossz*2 105
  tollszín! 15 tollvastagság! 1
vége
```

Végül utoljára maradt a paralelogramma eljárás:

```
tanuld paralelog :szélesség :hossz :szög
  ismétlés 2 ~
    [előre :szélesség balra 180-:szög~
    előre :hossz balra :szög]
vége
```

A Logo utasításai tetszőlegesen írhatók külön sorokba, illetve ugyanabba a sorba, szóközökkel elválasztva. Egy utasítást azonban alapesetben nem lehet megtörni, több sorba írni. Az ismétlések (és más későbbi utasítások) viszont általában elég hosszúak, s jó lenne sorokra tördelni őket. Erre ad lehetőséget a ~ jel, ami a sor végén jelzi, hogy az utasítás a következő sorban folytatódik.

ÖSSZETETT ÁBRÁK KÉSZÍTÉSÉNEK ALAPELVEI

Hogyan fogjunk hozzá egy elképzelt ábra megfogalmazásához?

ELŐKÉSZÍTÉS

1. Rajzoljuk le elképzelésünket (lehetőleg kockás) papírra!

ELGONDOLKODÁS

2. Készítettünk-e már ilyen ábrát? Esetleg a mostanitól kicsit eltérő formában? Felhasználhatnánk azt valahogy? Talán kisebb módosítással használhatóvá tehetnénk. Felhasználhatnánk esetleg a módszerét?
3. Próbáljuk az ábrát főbb részeire bontani!

TERVKÉSZÍTÉS

4. Határozzuk meg az egyes részek kezdő- és végpontjában a teknőc állapotát!
5. Gyakran hasznos az állapotátlátszó eljárások alkalmazása.
   Az így készített eljárások megkönnyíthetik a részek összerakásának folyamatát. Először csak a részek kiindulási pontjait kell megjelölnünk, míg az egyes részek előállításával nem törődünk. Csak később, a részletek kidolgozásánál kell az állapotátlátszóságra ügyelnünk, hogy biztosak legyünk abban, hogy az ábra egészét nem rontottuk el.
6. A bonyolult részeket először helyettesítsük állapotekvivalens eljárásokkal!
   A helyettesítés nem változtathatja meg az összerakás menetét. Miután az ábra egészének így létrehozott eljárását kipróbáltuk és az helyesnek bizonyult, ráérünk a helyettesített részlettel foglalkozni.
7. Használjuk fel a már meglevő általános eljárásokat!
8. Ha szükségünk van egy új eljárásra, írjuk meg!

MEGVALÓSÍTÁS

10. Fogalmazzuk meg a részek összeillesztésének folyamatát!
11. Alkalmazzunk általánosan kimondható szabályokat!
12. Adjunk értelmes nevet eljárásainknak és azok paramétereinek!

KIPRÓBÁLÁS

13. A kisebb próbálkozásokat parancsszinten végezzük el!
Így rögtön láthatjuk utasításaink hatását. Egy összetett feladatnál azonban már hátrányt jelenthet, ha minden utasítást külön végeztetünk el, mert könnyen belezavarodunk a megoldás menetébe. Hiba esetén elölről kell kezdenünk a munkát; a kész megoldás nem reprodukálható egyszerűen.
14. *Az összetett tevékenységeket mindig foglaljuk eljárásba!*
Az eljárás aktivizálásával próbáljuk ki, mennyire sikerült elérni célunkat. Így újra meg újra elő tudjuk állítani a megalkotott képet, miközben az eljárás tartalmát megfelelően módosítjuk. Egy tevékenységsorozat azonban nem mindig érzékelteti a hozzá tartozó ábra felbontását, lényeges részeit. Ezért próbáljuk úgy megfogalmazni az eljárást, hogy minél jobban eligazodjunk benne. Célszerű azokat a részeket, ahol a teknőc felemelt tollal közlekedik különválasztani azoktól a részektől, amelyek nyomot hagynak a rajzmezőn.
15. *Az eljárásokat beépítésük előtt próbáljuk ki!*
16. *Kövessük a teknőcmozgatást a rajzmezőn látható állapotban!*
17. *Személyesítsük meg a teknőcöt!*
Ha már mindent kipróbáltunk eljárásunk működésének vizsgálatára, de nem jöttünk rá, hogy hol követtünk el hibát, illetve miért nem úgy viselkedik a teknőc, ahogyan szeretnénk, akkor képzeljük magunkat a helyébe. Próbáljuk meg teljesíteni a teknőcnek szánt utasításokat.

UTASÍTÁS-ÖSSZEFOGLALÓ

tollatfel	a tollat felemeli, ezután haladás közben nem rajzol
tollatle	a tollat leengedi, ezután haladás közben rajzol
tollszín! *színkód*	ezután a megadott kódú színnel rajzol
tollvastagság! *szám*	ezután az adott vastagságú tollal rajzol

KÖRÖK, KÖRÍVEK, REKURZIÓ, FÁK

1. FELADAT: SZABÁLYOS SOKSZÖG RAJZOLÁSA

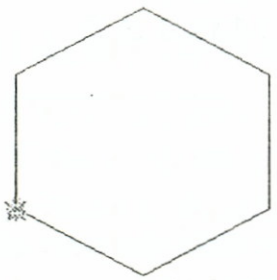
A háromszög, négyzet stb. eljárások alapján készítsünk általános sokszög(poligon)-rajzoló eljárást:

```
tanuld poligon :oldal :hossz
 ismétlés :oldal ~
  [előre :hossz balra 360/:oldal]
vége
```

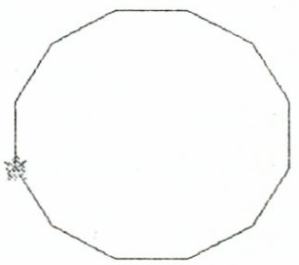
Matematikai tanulság: a sokszög külső szögeinek összege mindig 360 fok. A Logóban ez nem *bemagolandó* szabály, hanem kísérleti úton kikövetkeztethető tény. Hiszen ha a teknőc bármilyen szabályos sokszög vagy „szabálytalan" alakzat mentén halad, de végül a kiindulási állapotába ér vissza, akkor összességében 360 fokos fordulatot tesz meg. (A közbülső, előjelesen vett szögelfordulások összege 360 fok.) Ezt a **teljes teknőctételnek** nevezik. A körrajzolás tehát nem más, mint egy szabályos, nagyon sok oldalú sokszög rajzolása. Matematikai tanulság: a kör átmérője és kerülete viszonyának ismeretében kísérleti úton közelíthető a π értéke.

A poligon eljárást sokféle paraméterrel hívhatjuk meg:

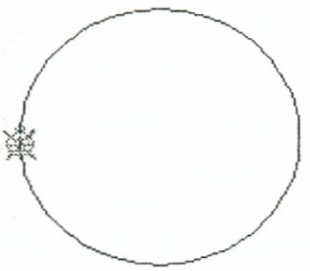
poligon 6 100



poligon 12 50



poligon 100 1

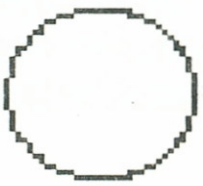


A sokszögrajzoló eljárást módosítjuk úgy, hogy a szöget is paraméterként adjuk meg, s nem számítjuk a szabályos képlet alapján:

```
tanuld poligon :oldal :hossz :szög
 ismétlés :oldal ~
  [előre :hossz balra :szög]
vége
```

Ez az eljárás is meghívható különböző paraméterekkel:

poligon 100 1 3.6



poligon 50 1 3.6

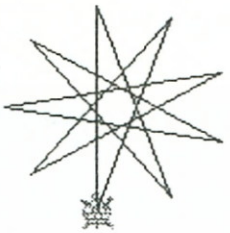
poligon 50 -1 -3.6

Ennek felhasználásával tehát nemcsak köröket, hanem köríveket is rajzolhatunk, s a körívekből különböző ábrákat állíthatunk össze.

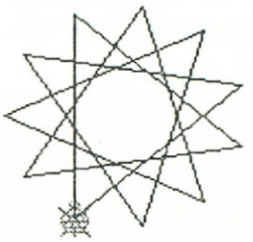
Az új sokszögrajzoló nemcsak sokszöget tud rajzolni, hanem csillagokat is. Csupán arra kell figyelnünk meghívásakor, hogy a fordulatok összege a rajzolás során 360 fok többszöröse legyen.

poligon 5 100 144

poligon 9 160



poligon 11 100 130.91



2. FELADAT: KÖR ÉS KÖRÍV RAJZOLÁSA

Egy adott átmérőjű kör matematikai tudásunk alapján a következő eljárással rajzolható (kerület=átmérő*π, fordulatok összege=360 fok):

```
tanuld kör :átmérő
  poligon 100 ~
  :átmérő/100*3.14159 3.6
vége
```

A fenti körrajzolás nagyon könnyen átalakítható körívrajzolássá, csupán a lépésszámot kell a körív szöge és a 360 fokos teljes kör szöge arányával módosítani.

```
tanuld körív :átmérő :szög
  poligon 100*:szög/360 ~
  :átmérő/100*3.14159 3.6
vége
```

3. FELADAT: SOKSZÖG RAJZOLÁSA REKURZÍVAN

A szabályos N szög rajzolását másképp is felfoghatjuk: először rajzolni kell 1 oldalt, utána el kell fordulni a megfelelő szöggel, majd már csak ugyanezt kell megismételni a maradék N–1 oldalra.

```
tanuld reksokszög :oldal :hossz :szög
  előre :hossz balra :szög
  ha :oldal>1 ~
    [reksokszög :oldal-1 :hossz :szög]
vége
```

Ezzel elkészült az első *rekurzív* eljárás, egy olyan eljárás, amelynek végrehajtásakor meg kell hívnia (azaz végre kell hajtania) saját magát.

4. FELADAT: A GLOBÁLIS (KÍVÜLRŐL LÁTHATÓ) OBJEKTUMOK HASZNÁLATA

Készítsük el a következő **köröz** eljárást:

```
tanuld köröz
  előre :lépés
  jobbra :fordulat
  köröz
vége
```

Az objektumoknak az írólapon adhatunk értéket:

```
? név "lépés 10
? név "fordulat 5
```

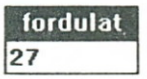
Ezután használható a **köröz** eljárás:

```
? köröz
```

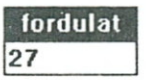
Az eljárás adatait a fent leírt módon a **név** utasítással lehet változtatni, majd újraaktivizálni az előbbiek szerint.

5. FELADAT: SZEMLÉLETES GLOBÁLIS OBJEKTUMOK KÉSZÍTÉSE

A **Memória** ablakból az egér jobb gombját (és a SHIFT gombot) lenyomva kivihető a **lépés** és a **fordulat** objektum a rajzlapra, ahol egy-egy szövegdobozként jelennek meg.



A **köröz** eljárás végrehajtása közben a szövegdobozra kattintva módosítható az egyes objektumok értéke. A változtatás az ablak jobb felső gombjának benyomásával érhető el.



6. FELADAT: SPIRÁL RAJZOLÁSA

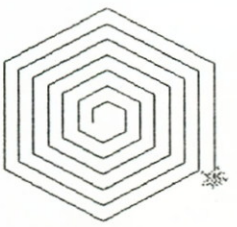
A rekurzív sokszög rajzolását könnyen átalakíthatjuk spirálrajzolássá, csupán arra kell biztatnunk a teknőcöt, hogy mindig egy kicsit nagyobb távolságot, a *hossz*-nak a *táv*-val növelt értékét tegye meg:

```
tanuld reksokszög :oldal :hossz ~
   :szög :táv
  előre :hossz balra :szög
  ha :oldal>1 [reksokszög :oldal-1 ~
   :hossz+:táv :szög :táv]
vége
```

Különböző *táv* paramétereket adva a spirál egyes szakaszai különböző távolságra lesznek az eggyel beljebb elhelyezkedő szakasztól, de a párhuzamos oldalak távolsága mindig állandó.

```
? reksokszög 40 10 6 2
```

Ha az utolsó paraméter pozitív, akkor a spirált kifelé, ha pedig negatív, akkor befelé csavarodva rajzolja a teknőc.



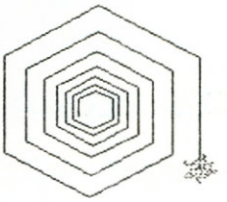
A rekurzív sokszöget másképp is módosíthatjuk:

```
tanuld újreksokszög :oldal :hossz~
    :szög :szor
  előre :hossz balra :szög
  ha :oldal>1 [újreksokszög :oldal-1 ~
    :hossz*:szor :szög :szor]
vége
```

Ebben az esetben a *szor* paraméter mint konstans szorzószám befolyásolja az ábrát.

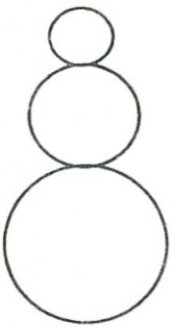
```
? újreksokszög 40 10 6 1.05
```

A spirál egyes ágai egyre jobban távolodnak egymástól. Nem a párhuzamos szakaszok távolsága, hanem két párhuzamos szakaszpár távolságának aránya lesz állandó.



7. FELADAT: HÓEMBER RAJZOLÁSA

A rekurzió egyik egyszerű, nevezetes példája a hóemberrajzolás. Nézzük meg ennek három megoldási ötletét!



1. Az N körből (a valóságban gömbből, gömbszerű hólabdából) álló hóembert úgy kell rajzolni, hogy megrajzolunk egy kört, majd fölé rajzolunk egy N-1 körből álló hóembert. A számítástechnikai szakirodalom ezt hívja *jobb rekurzió*nak, az eljárás a legvégén hívja meg saját magát.

```
tanuld hóember :db :átmérő
 jobbra 90 kör :átmérő balra 90
 ha :db>1 [tollatfel előre :átmérő ~
  tollatle ~
   hóember :db-1 2*:átmérő/3]
vége
```

2. Az N körből álló hóembert úgy kell rajzolni, hogy megrajzolunk egy N-1 körből álló hóembert, majd alá rajzolunk egy kört: A számítástechnikai szakirodalom ezt hívja *bal rekurzió*nak, az eljárás a legelején hívja meg saját magát.

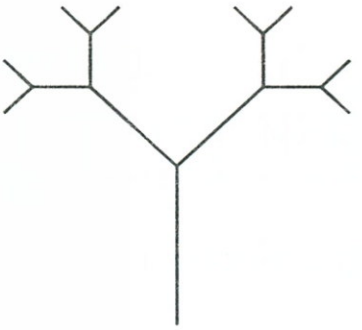
```
tanuld hóember :db :átmérő
 ha :db>1 ~
  [hóember :db-1 2*:átmérő/3 ~
   tollatfel hátra :átmérő tollatle]
 jobbra 90 kör :átmérő balra 90
vége
```

3. Az N körből álló hóembert úgy kell rajzolni, hogy megrajzolunk egy félkört, fölé rajzolunk egy N-1 körből álló hóembert, majd befejezzük a félkört:
Ez az *általános rekurzió*, az eljárás a közepén hívja meg saját magát.

```
tanuld hóember :db :átmérő
 jobbra 90 félkör :átmérő jobbra 90
 ha :db>1 ~
  [hóember :db-1 2*:átmérő/3]
 balra 90 félkör :átmérő balra 90
vége
```

8. FELADAT: FA RAJZOLÁSA REKURZÍVAN

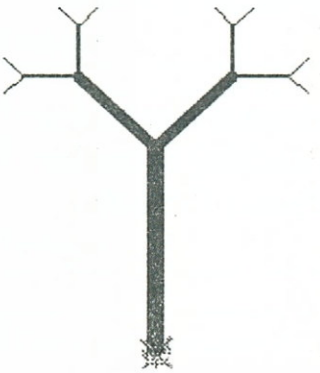
Egy szabályos fa lehet olyan alakú, mint amilyen az ábrán látható.
A fa egy törzsből áll, amelynek tetején két ág nő ki, s mindkét ág tulajdonképpen egy-egy alacsonyabb, rövidebb törzsű fa. Ezekről a fákról ugyanezt mondhatjuk el.



Ha egy ábrában önmagát felfedezzük, akkor mindig rekurzívan rajzolhatjuk meg, méghozzá az eljárás annyiszor hívja meg önmagát, ahányszor az ábrát megtaláljuk önmagában.

```
tanuld fa :db :hossz
 előre :hossz
 ha :db>1 ~
  [balra 45 fa :db-1 :hossz/2 ~
   jobbra 90 ~
   fa :db-1 :hossz/2 balra 45]
   hátra :hossz
vége
```

Olyan fát is rajzolhatunk, amelynek nemcsak rövidülnek az ágai, hanem vékonyodnak is.



```
tanuld fa :db :hossz :vastagság
 tollvastagság! :vastagság
  előre :hossz
 ha :db>1 ~
  [balra 45 fa :db-1 :hossz/2 ~
   :vastagság/2 ~
   jobbra 90 fa :db-1 :hossz/2 ~
   :vastagság/2 ~
   balra 45]
tollvastagság! :vastagság
 hátra :hossz
vége
```

UTASÍTÁS-ÖSSZEFOGLALÓ

ha *feltétel*
[*akkor ág utasításai*]
[*különben ág utasításai*]

választás két utasításcsoport között

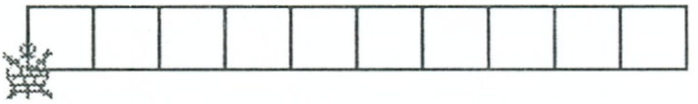
vagy
ha *feltétel* [*akkor ág utasításai*]

utasítások feltételes végrehajtása

A feltétel tetszőleges relációjeleket, valamint logikai műveleteket (és, vagy, nem) tartalmazhat.

SORMINTÁK, TERÜLETMINTÁK

1. FELADAT: SORMINTA AZONOS ELEMEKBŐL



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 ismételni a megfelelő számban.

Ezt a feladatot a következőképpen oldhatjuk meg:

```
tanuld sorminta :n :x :y
 ismétlés :n [alapelem :x :y ~
  elmozdulás :x]
 elmozdulás -:n*:x
vége
```

Az alapelem a legegyszerűbb esetekben négyzet, téglalap, rombusz, egyenlő oldalú háromszög vagy szabályos hatszög. A fenti példában négyzeteket használunk, de a későbbi általánosíthatóság miatt az alapelemet definiáló eljárás téglalapot rajzol.

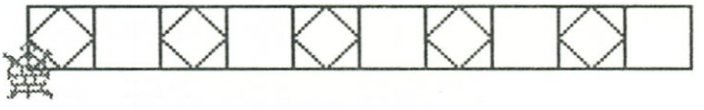
```
tanuld alapelem :x :y
 ismétlés 2 [előre :x jobbra 90 ~
  előre :y jobbra 90]
vége
```

```
tanuld elmozdulás :x
 tollatfel jobbra 90 előre :x
  balra 90 tollatle
vége
```

Ugyanez az eljárás rekurzívan is megfogalmazható:

```
tanuld sorminta :n :x :y
 ha :n>0 [alapelem :x :y ~
  elmozdulás :x sorminta :n-1 :x :y ~
   elmozdulás -:x]
vége
```

2. FELADAT: SORMINTA KÉTFÉLE ALAPELEMBŐL



Az egyik alapelem legyen továbbra is a négyzet, a másik pedig egy, a négyzet belsejébe írt elforgatott négyzet.

Egy logikai változó paraméterrel jelezzük, hogy éppen melyik alapelemet kell alkalmazni:

```
tanuld sorminta :n :x :y :logi
 ha :n>0 [ha :logi [alapelem1 :x :y] ~
  [alapelem2 :x :y] elmozdulás :x ~
   sorminta :n-1 :x :y nem :logi ~
   elmozdulás -:x]
vége
```

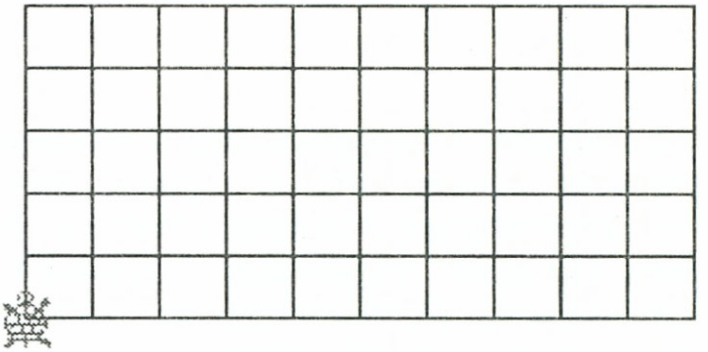
Megoldható más mintázatképzési szabályok kezelése is:

Ebben az esetben a logikai változó helyett egy számváltozót használunk, amelynek most éppen a 3-mal vett osztási maradékát vizsgáljuk. Ha a maradék 1, akkor az első alapelemet kell rajzolni, ha 0 vagy 2, akkor pedig a másodikat.

```
tanuld sorminta :n :x :y :szám
 ha :n>0 [ha :szám mod 3=0 ~
   [alapelem1 :x :y] ~
   [alapelem2 :x :y] ~
   elmozdulás :x ~
   sorminta :n-1 :x :y ~
   :szám+1 elmozdulás -:x]
vége
```

3. FELADAT: TERÜLETKITÖLTÉS

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



```
tanuld területminta :m :n :x :y
 ha :m>0 ~
   [sorminta :n :x :y felfelé :y ~
     területminta :m-1 :n :x :y ~
     lefelé :y]
vége
```

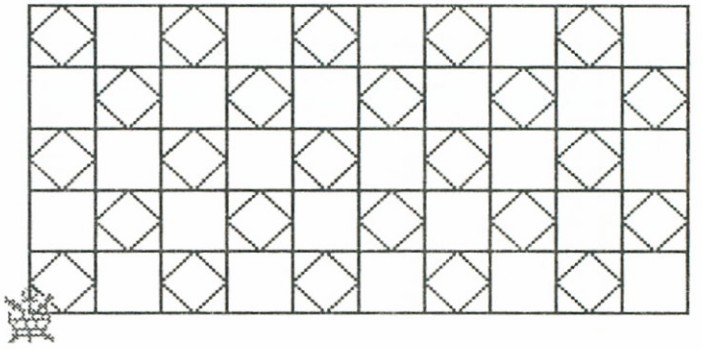
```
tanuld felfelé :y
 tollatfel előre :y tollatle
vége
```

```
tanuld lefelé :y
 tollatfel hátra :y tollatle
vége
```

A területminta is képződhet kétféle elemből:

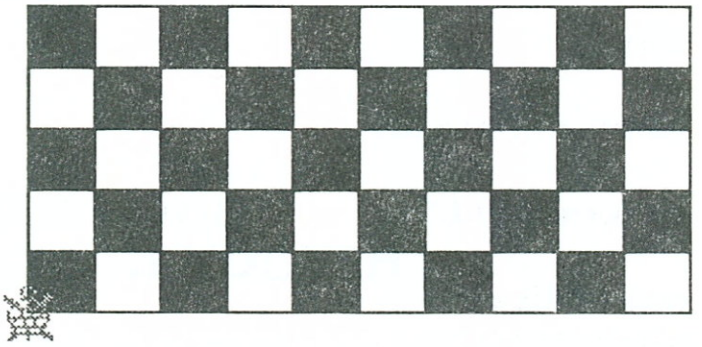
```
tanuld területminta :m :n :x :y :logi
 ha :m>0 ~
   [sorminta :n :x :y :logi felfelé :y ~
     területminta :m-1 :n :x :y ~
     nem :logi lefelé :y]
vége
```

Megoldása hasonló, mint a sorminta esetében. Itt is vezessünk be egy logikai értékű paramétert, amelyet felváltva alkalmazunk.



4. FELADAT: FESTETT TERÜLETEK

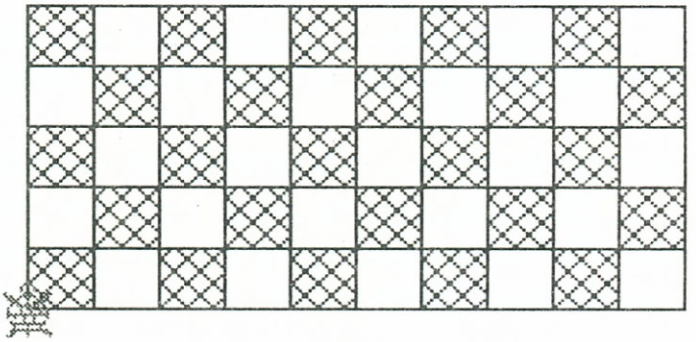
A területminta egyféle alapelem festett és festetlen változatából is állhat:



Ennek megvalósításához csupán azt kell tudni, hogyan lehet egy alakzatot befesteni. Erre a **tölt** utasítás alkalmas, amelyet egy zárt alakzat belsejében kell kiadni. Például egy – legalább 2 oldalhosszúságú – négyzet befestése (amelynek bal alsó sarkában áll a teknőc, s felfelé néz):

```
tanuld négyzetfestő
  jobbra 45 tollatfel előre 2
  tollatle tölt hátra 2 balra 45
vége
```

Festéskor megmondhatjuk a festés mintázatát, valamint a festés közben alkalmazott színt (ennek nem kell azonosnak lennie a teknőc tollszínével). A **töltőszín!** *színkód* és a **töltőminta!** *kód* utasítás a befestés előtt bármikor kiadható.



5. FELADAT: KEZDŐÁLLAPOT BEÁLLÍTÁSA

A Logo rendszer indításakor a teknőc a képernyő közepén áll, s felfelé néz (további állapotkomponensei is valamilyen alapállapotban vannak). Egyes ábrák megrajzolásakor ez nem szerencsés. Minden ábra megrajzolása előtt el kell juttatni a teknőcöt a kívánt kezdőhelyre, s beállítani a kezdő irányba. Ezt megtehetjük az eddigi utasításokkal is, de lehetőségünk van a környezethez viszonyított beállításra is.

```
tanuld kezd
  xhely! -100 yhely! -100
  ; xyhely! -100 -100
  irány! -90
vége
```

A programokba megjegyzést is írhatunk: tetszőleges sorban, a pontosvesszőtől a sor végéig tartó részt a Logo figyelmen kívül hagyja.

UTASÍTÁS-ÖSSZEFOGLALÓ

tölt	befesti az alakzatot, amelynek a teknőc a belsejében van
töltőszín! *színkód*	a befestéskor alkalmazott szín *kódját* állítja be
töltőminta! *kód*	a befestéskor alkalmazott mintázat *kódját* állítja be
maradék *x y*	x és y osztási maradékát adja
xhely! *érték*	a teknőc x koordinátáját az adott *értékre* állítja
yhely! *érték*	a teknőc y koordinátáját az adott *értékre* állítja
xyhely! *érték*	a teknőc x és y koordinátáját az adott *értékre* állítja
irány! *érték*	a teknőc irányát az adott *értékre* állítja, az északi irányhoz képest, az óramutató járása szerint

A KOORDINÁTA-RENDSZER ÉS A FRAKTÁLOK

A COMENIUS LOGO ÉS A KOORDINÁTA-RENDSZER

Az *euklidészi* geometria a sík elemeinek a pontot, az egyenest és a szöget tekinti, így a síkbeli alakzatokat ezek kapcsolatai fejezik ki. A pont ebben a megközelítésben statikus. Euklidész az axiomatikus logikai stílust képviselte.

Descartes a sík elemének a pontot tekintette, mégpedig úgy, hogy helyzetét a sík két irányított, egymásra merőleges egységgel ellátott egyeneséhez (a koordinátatengelyhez) viszonyította. Így a sík bármely pontjának helyzetét a koordináta-rendszer két tengelyétől mért előjeles távolsággal határozta meg. A síkbeli alakzatokat az alakzat pontjainak a koordináta-rendszerhez viszonyított egyenleteivel adta meg.

Ismerünk más koordináta-rendszert is, ilyen például a *polárkoordináta-rendszer*. Ebben egy síkbeli pont helyzetének meghatározásához egy O pontot (pólus) és egy O-ból kiinduló félegyenest (polártengely) kell felvenni. A sík egy pontjának polárkoordinátáját a pont pólustól mért távolsága és a pólustól a ponthoz futó egyenes polártengellyel bezárt szöge határozza meg. A síkbeli alakzatokat pontjainak polárkoordinátáival kifejezett egyenletei adják meg.

A *teknőc* nem más, mint egy állapottal rendelkező pont a síkon. Az állapotot meghatározza a **pozíció:**: a rajzmező síkján elfoglalt hely (Descartes-koordináta) és az **irány:**: északhoz (felfelé) képest az óramutató járásával megegyezően mért elfordulási szög (polárszög). A síkbeli alakzatokat a teknőc állapotváltoztató utasításaival adhatjuk meg. Az alakzat a teknőc által megrajzolt ábra.

A teknőcgeometria legfigyelemreméltóbb tulajdonsága az, hogy létrehozza az alakzatot és nem egyenletben fejezi ki az alakzat tulajdonságait.

Mivel e teknőctevékenységek megegyeznek az emberi mozgásmechanizmus egyes elemeivel, a definíciókat mint tevékenységsorozatokat magunk is könnyűszerrel megvalósíthatjuk, illetve a teknőc megszemélyesítésével megérthetjük. Ezért lehetséges, hogy a teknőcgeometria gyerekeknek is tanítható.

Összefoglalva, az állapottal rendelkező pont és az állapotváltoztató utasítások a síkon (és akár a térben) együttesen jelentik a teknőcgeometria elemeit.

A gyakorlatokban szereplő utasítások közül a következők szolgálnak a teknőcök jellemzőinek (állapotkomponenseinek) a megváltoztatására és lekérdezésére:

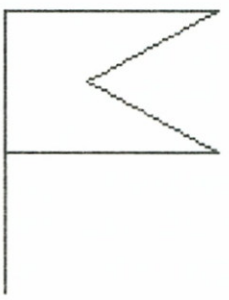
Teknőcjellemző	Beállítása	Lekérdezése
helye (x, y koordináta)	xhely! *érték*, yhely! *érték* xyhely! *érték érték*	xhely, yhely, hely
iránya	irány! *szög*	irány
alakja	alak! *kép*	alak
rajzeszköze helyzete	toll!, tollatle, tollatfel, toll-radír, tollváltó	toll
tolla színe	tollszín! *színkód*	tollszín
tolla vastagsága	tollvastagság! *sorszám*	tollvastagság
tolla mintázata	tollminta! *sorszám*	tollminta
láthatósága	látható, láthatatlan	látható?
figyelése	figyelj	kifigyel
ideiglenes figyelés	kérem	—
létező teknőcök	—	mind

GEOMETRIAI TRANSZFORMÁCIÓK

HASONLÓSÁGI TÉTEL:

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

```
tanuld zászló :rúdhossz :hossz
  előre :rúdhossz jobbra 90 előre :hossz
    jobbra 150
  előre :rúdhossz/2 balra 120
    előre :rúdhossz/2 jobbra 150
  előre :hossz jobbra 90 hátra ~
    :rúdhossz/2
vége
```



1. TÜKRÖZÉSI TÉTEL:

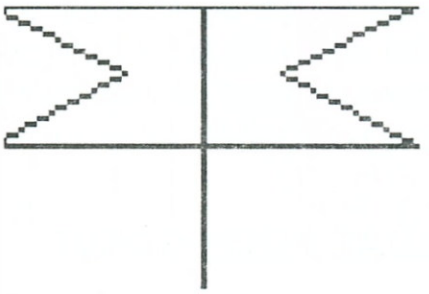
Ha a lépéseket változatlanul hagyjuk, de a fordulatok irányát ellentétesre változtatjuk, akkor a teknőc kiinduló állapotán (pozíció és irány) átmenő egyenesre tükrözzük az alakzatot.

A fordulatok ellentétesre váltásának két módja van:

- minden **balra** utasítás helyére **jobbra** és minden **jobbra** utasítás helyére **balra** írandó;

- minden **balra** és **jobbra** utasítás paraméterét ellentétes előjelűre cseréljük (a pozítív szögelfordulásokat negatívra és viszont).

```
tanuld tükörzászló :rúdhossz :hossz
  előre :rúdhossz balra 90
  előre :hossz balra 150
  előre :rúdhossz/2 jobbra 120
  előre :rúdhossz/2 balra 150
  előre :hossz balra 90 hátra :rúdhossz/2
vége
```

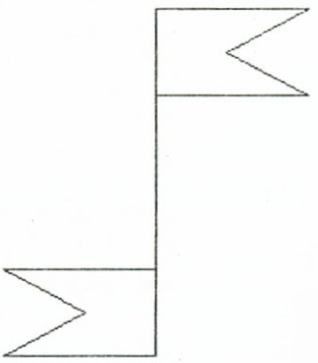


2. TÜKRÖZÉSI TÉTEL

Ha a fordulatokat változatlanul hagyjuk, de a lépések hosszát ellentétesre változtatjuk, akkor a teknőc kiindulópontjára tükrözzük az alakzatot (vagy másképp fogalmazva: a kiindulópont körül elforgatjuk 180 fokkal).

A lépések ellentétesre váltásának két módja van:
- minden előre utasítás helyére hátra és minden hátra utasítás helyére előre írandó;
- minden előre és hátra utasítás paraméterét ellentétes előjelűre cseréljük.

```
tanuld forgatzászló :rúdhossz :hossz
  hátra :rúdhossz jobbra 90
  hátra :hossz jobbra 150
  hátra :rúdhossz/2 balra 120
  hátra :rúdhossz/2 jobbra 150
  hátra :hossz jobbra 90
    előre :rúdhossz/2
vége
```



3. TÜKRÖZÉSI TÉTEL

Ha a lépések hosszát és a fordulatok irányát is ellentétesre változtatjuk, akkor a teknőc induló állapotára (pozíció és irány) merőlegesen átmenő egyenesre tükrözzük az alakzatot.

Az eredeti alakzat eltolt, illetve elforgatott képét az eljárás meghívása előtt végrehajtott előre, hátra, jobbra, balra utasításokkal kaphatjuk.

```
tanuld zászlótükör :rúdhossz :hossz
  előre -:rúdhossz
  jobbra -90 előre - :hossz jobbra -150
  előre -:rúdhossz/2 balra -120
  előre -:rúdhossz/2
  jobbra -150 előre -:hossz
  jobbra -90 hátra -:rúdhossz/2
vége
```

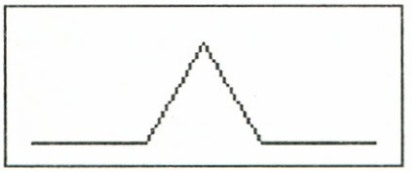
Bármely síkbeli egybevágósági és hasonlósági transzformáció előállítható a fentiek szorzataként (megfelelő számú transzformáció egymásutánjával), így az eredeti alakzatnak a síkban tetszőlegesen elhelyezkedő egybevágó vagy hasonló képe előállítható.

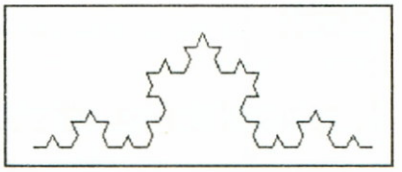
A FRAKTÁLOK

Érdekes, látványos alkalmazási terület a fraktáloké, azaz a törtdimenziós alakzatok köre. A fraktálok jellemzői közé tartozik az önhasonlóság, azaz mindegyikben felfedezhetünk olyan részeket, amelyek hasonlóak az eredeti ábrához. Ilyen alakzat a korábban látott fa is, amely egy törzsből, valamint kettő, az eredetihez hasonló, csak kisebb fából áll.

1. FELADAT: KOCH-GÖRBE

Az egyik nevezetes fraktál az úgynevezett Koch-görbe. Ez egy rekurzív definícióval rajzolt görbe, amelynek adott szintű változatai rajzolhatók meg.

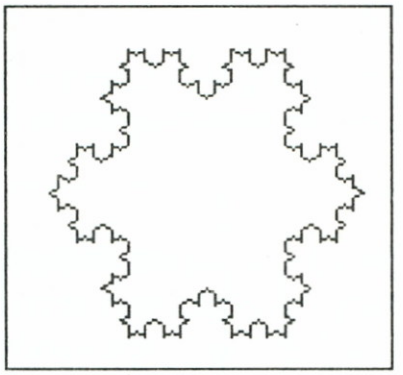


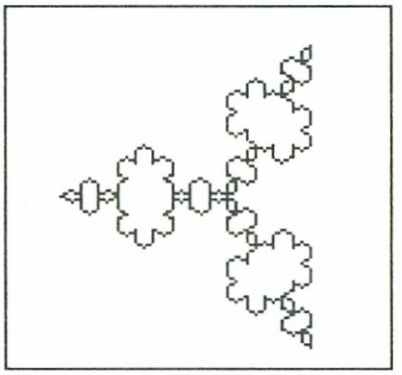


```
tanuld Koch :szint :hossz
 ha :szint = 1 ~
  [előre :hossz] ~
  [Koch :szint - 1 :hossz / 3 balra 60 ~
   Koch :szint - 1 :hossz / 3 jobbra 120 ~
   Koch :szint - 1 :hossz / 3 balra 60 ~
   Koch :szint - 1 :hossz / 3]
vége
```

2. FELADAT: KRISTÁLY ÉS HÓPEHELY

A Koch-görbe nagyon szép hópehely-, illetve kristályszerű ábrákat adhat, ha a kiinduló egyenes szakaszt háromszöggel helyettesítjük. Másféle ábrát kapunk, ha a háromszöghöz képest kifelé, illetve ha befelé rajzoljuk a Koch-görbét.

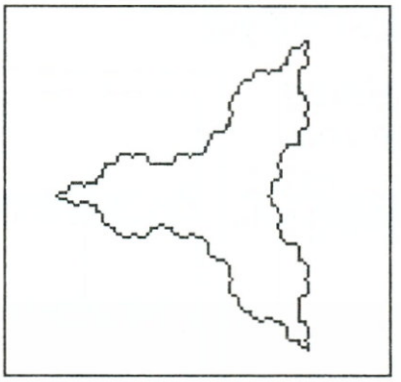


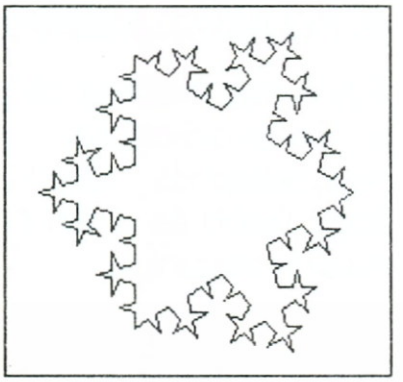


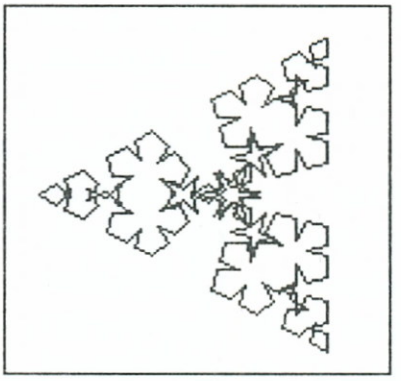
```
tanuld hópehely :szint :hossz
  ismétlés 3 [Koch :szint :hossz jobbra 120]
vége
```

```
tanuld kristály :szint :hossz
  ismétlés 3 [Koch :szint :hossz balra 120]
vége
```

Az eredeti Koch-görbében a balra, illetve jobbra fordulás szögét változtatva különlegesen szép ábrákat kaphatunk.

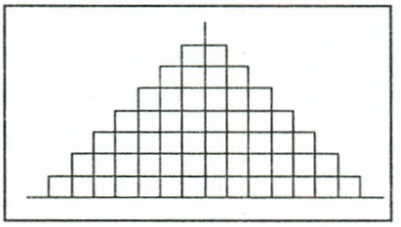


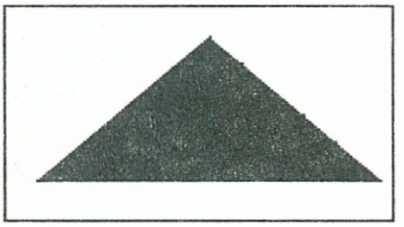




3. FELADAT: TERÜLETFESTÉS

Ha a fordulatok szöge 90, illetve 180 fok, a lépéshossz pedig az eredeti fele, akkor egy háromszög alakú területet lefedő négyzethálót kapunk, s elég magas szintű ábrát rajzolva befesthetjük az egész háromszöget.

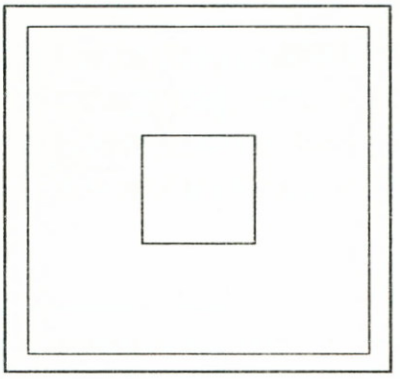


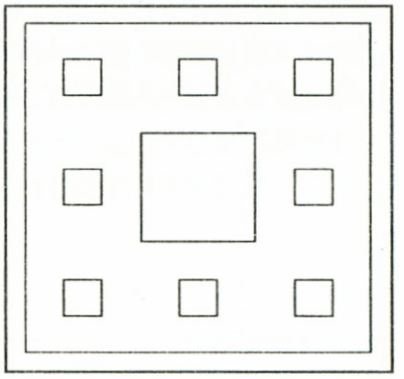


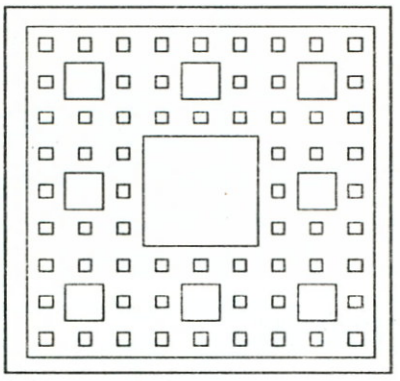
```
tanuld Koch :szint :hossz
 ha :szint = 1 ~
  [előre :hossz] ~
  [Koch :szint - 1 :hossz / 2 balra 90 ~
   Koch :szint - 1 :hossz / 2 jobbra 180 ~
   Koch :szint - 1 :hossz / 2 balra 90 ~
   Koch :szint - 1 :hossz / 2]
vége
```

4. FELADAT: SIERPINSKI-CSIPKE

Egy másik érdekes, síkbeli alakzat a Sierpinski-csipke, amely egy négyzetlapból keletkezik úgy, hogy újabb és újabb belső négyzeteket vágunk ki.



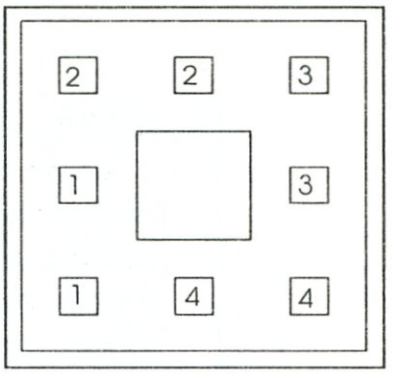




Először rajzolunk egy négyzetet, majd elkezdjük a kivágásokat

```
tanuld Sier :szint :hossz
  tollatle négyzet :hossz tollatfel
  Sierpinski :szint :hossz
vége
```

Minden egyes négyzetből kivágjuk a közepét, majd a megmaradó nyolc kisebb négyzetre alkalmazzuk ugyanezt az eljárást. A nyolc négyzet négy kettes csoportba sorolható az ábra szerint.




```
tanuld Sierpinski :szint :hossz kivágás :hossz / 3
  ha :szint > 0 [ismétlés 4 [ismétlés 2 [Sierpinski :szint - 1 :hossz / 3 ~
    előre :hossz / 3] előre :hossz / 3 jobbra 90]]
vége
```

A kivágáshoz a négyzet közepére kell menni, majd egyharmad élhosszúságú négyzetet kell rajzolni.

```
tanuld kivágás :hossz
  előre :hossz jobbra 90 előre ~
    :hossz balra 90
  tollatle négyzet :hossz tollatfel
  jobbra 90 hátra :hossz balra 90
  hátra :hossz
vége
```

AZ ANIMÁCIÓ

AZ ANIMÁCIÓ ELEMEI

1. FELADAT: ANIMÁCIÓ RAJZLAPCSERÉVEL

A legegyszerűbb animációs projekthez nem kell más, mint sok előre megrajzolt rajzlap, amit egységes időközönként váltogathatunk. Ehhez bármilyen teknőcrajzot felhasználhatunk, amelyet elmentettünk a rajzlap alkönyvtárba valamilyen néven.

Például a **rajzlap1.bmp, rajzlap2.bmp, ... rajzlap9.bmp** állományok különböző rajzokat tartalmaznak. A betöltött rajz az aktuális rajzablakban fog megjelenni, beállítástól függően középen vagy a bal felső sarokban.

```
tanuld lapozz
  betöltrajzlap "rajzlap1.bmp várj 10
  betöltrajzlap "rajzlap2.bmp várj 10
  betöltrajzlap "rajzlap3.bmp várj 10
  betöltrajzlap "rajzlap4.bmp várj 10
  betöltrajzlap "rajzlap5.bmp várj 10
  betöltrajzlap "rajzlap6.bmp várj 10
  betöltrajzlap "rajzlap7.bmp várj 10
  betöltrajzlap "rajzlap8.bmp várj 10
  betöltrajzlap "rajzlap9.bmp várj 10
vége
```

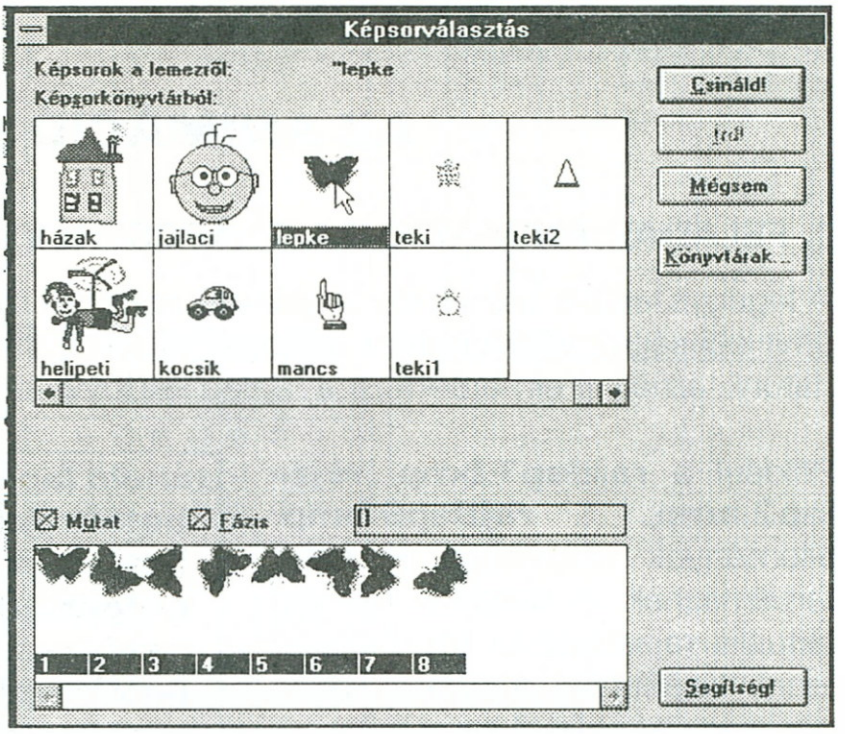
A **betöltrajzlap** utasítás paramétere egy állomány neve. A szókonstansokat a Logóban idézőjel vezeti be, s az első szóköz karakter zárja le.

A rajzok egymás után jelennek meg, megadott ideig, egy diasorozathoz hasonlóan. A várj utasítás hatására a megadott ezredmásodperc (az utána írt szám) idejéig a program várakozik. Az animációt gyorsítani lehet a várj utasításnak megadott idő csökkentésével, vagy az utasítás elhagyásával.

E módszer hátránya, hogy kicsi alakzatok gyors mozgatására nem nagyon szerencsés a képek betöltése háttértárról.

2. FELADAT: ALAKVÁLTOZTATÁS ÉS MOZGATÁS

Az animáció másik fajtájaként változtassuk meg a teknőc alakját, majd mozgassuk. Az alak megváltoztatásának legegyszerűbb módja betölteni a háttértárról egy előre elkészített képet, ami a teknőc alakja lesz. A segítségként megadott utasításgyűjteményből válasszuk ki az *alak!* utasítást! A megjelenő ablakból válasszuk ki a lepkét és kattintsunk a **Csináld!** gombra. Ekkor a teknőc képe helyett egy lepkéé jelenik meg. Az egyes alakokhoz különböző fázisok is tartozhatnak, normál módban az aktuális iránytól függ, hogy a képernyőn melyik fázis látható.



Ezután a teknőc helyett a lepkét mozgassuk a megszokott módon:

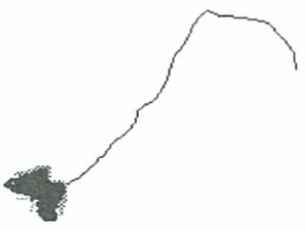
```
tanuld halad
  alak! "lepke
  ismétlés 100 [előre 1 balra 1]
vége
```

A teknőcöt működtethetjük animációs módban is, ekkor nem az aktuális iránytól függ a megjelenő fázis, hanem mi állíthatjuk át tetszés szerint a **fázis!** utasítással. Rákattintva a jobb egérgombbal a teknőc képére, az előbukkanó ablakban ikszeljük be az **Animációs mód** dobozt. Ekkor a teknőc fázisainak cseréjéről magunknak kell gondoskodnunk. Ha a lepke két alakját felváltva rajzoljuk előrehaladás közben, akkor úgy látjuk, mintha a lepke szárnyával csapkodva repülne.

```
tanuld repül
  animációsmód! "igaz alak! "lepke
  ismétlés 100 ~
    [fázis! 2 várj 100 fázis! 3 várj 100 előre 3]
  animációsmód! "hamis
vége
```

3. FELADAT: VÉLETLENSZERŰ REPÜLÉS

Módosítsuk a teknőc alakját lepkére! Olyan eljárást készítünk, amely a végtelenségig röpteti a lepkéket (vagy más alakokat) úgy, hogy véletlenszerű útvonalat járhat be a lepke!



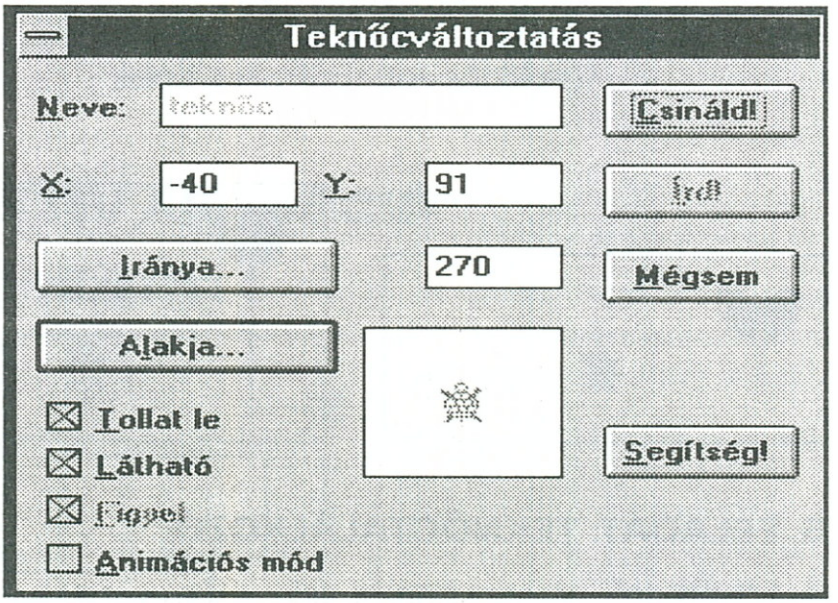
Ehhez úgynevezett véletlenszámokat alkalmazunk: véletlenszám N a Logóban egy 0 és N–1 közötti egész számot jelent.

```
tanuld repülj
  előre (véletlenszám 10)
  jobbra (véletlenszám 60)-30
  várj 10 repülj
vége
```

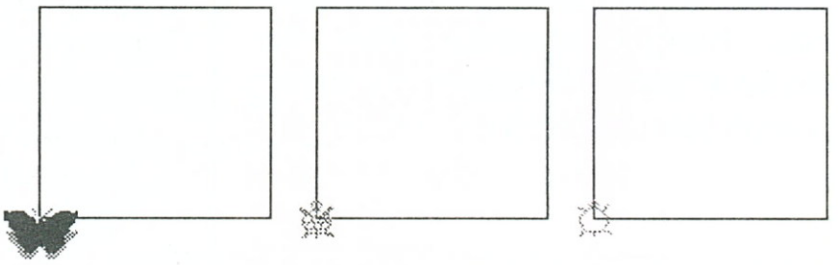
4. FELADAT: SOK SZEREPLŐ (TEKNŐC) EGYÜTTES MOZGATÁSA

Hozzunk létre új teknőcöket úgy, hogy az egér jobb gombjával rákattintunk valahol a képre. Az **Új teknőc** sort kiválasztva megadhatók a teknőc tulajdonságai (név, irány stb.).

Minden teknőcnek van neve és alakja is. Ezeket meg lehet változtatni. A 0 nevű teknőc alakja alapértelmezésben a zöld teknősbéka. Minden teknőc alakja változtatható, ha az egér jobb gombjával rákattintunk, majd a **Teknőcváltoztatás** menüben az **Alakja** gombra kattintunk. Itt a képsorkönyvtárból egy másik alaksorozatot választhatunk, vagy akár a meglévő képsort módosíthatjuk, ha az egér bal gombjával a képre duplán kattintunk.

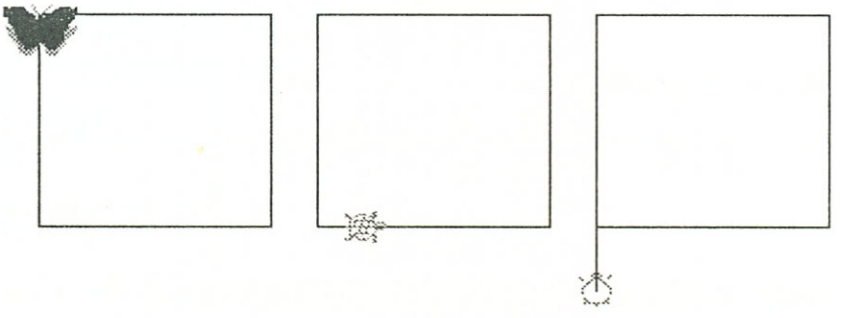


Ha több teknőcünk is van, s mindegyik figyel az utasításainkra, akkor mindegyik ugyanazt a rajzot rajzolja. (A figyelés a teknőc létrehozásakor megjelenő ablakban beállítható.)



Ha nem kell mindegyik teknőcnek dolgoznia, akkor az aktív (ránk figyelő) teknőcöket a figyelj utasítással beállíthatjuk:

```
figyelj "lepke előre 100
figyelj "teknőc jobbra 90 előre 20
figyelj [teknőc üreshasúteknőc] hátra 30
```

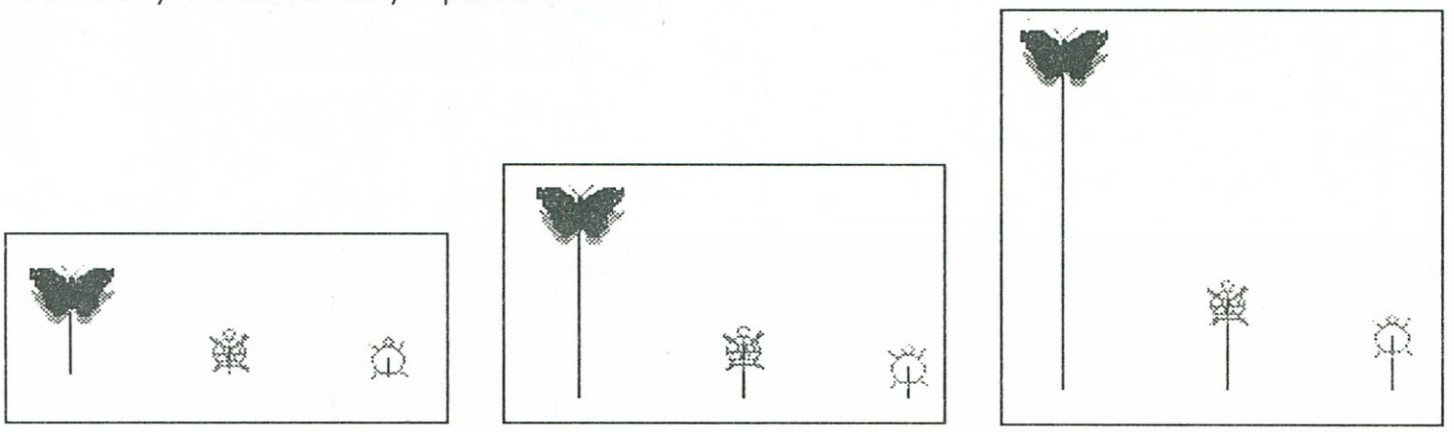


A többféle teknőcnek más-más utasítást is adhatunk anélkül, hogy az aktív, figyelő teknőcök körét megváltoztatnánk. Az alábbi program versenyfutást rendez a három „teknőc" között:

```
tanuld verseny
  kérem "lepke [előre 20]
  kérem "teknőc [előre 5]
  kérem "üreshasúteknőc [előre 3]
  várj 100
  verseny
vége
```

A kérem utasítás csak az utána felsorolt utasítások végrehajtásának idejére teszi figyelőképessé a teknőcöt.

A verseny állása néhány lépés után:

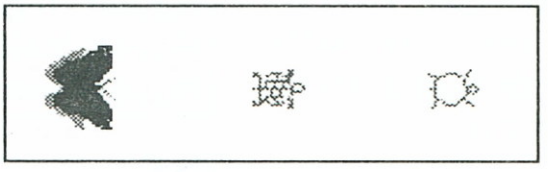


5. FELADAT: TEKNŐCTALÁLKOZÓ

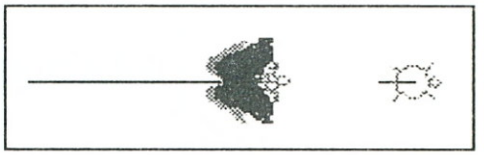
A következő versenyben a teknőcök nem párhuzamosan futnak, hanem egymás után. A verseny akkor ér véget, ha valamelyik teknőc utoléri az előzőt.

```
tanuld verseny
  figyelj "üreshasúteknőc előre 3
  figyelj "teknőc előre 5
  ha nem takar? "üreshasúteknőc ~
    [figyelj "lepke előre 20 ~
     ha nem takar? "teknőc ~
       [várj 100 verseny]]
vége
```

A rajt:



A verseny vége:



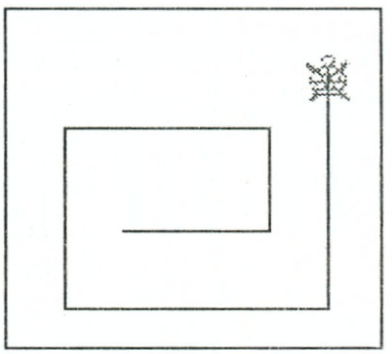
UTASÍTÁS-ÖSSZEFOGLALÓ

betöltrajzlap! *"név*	betölti az aktuális rajzablakba a *név* nevű képet
várj *időtartam*	*időtartam*, ezredmásodperc várakozás
alak! *"név*	a teknőc aktuális alakját *név*re változtatja
fázis! *sorszám*	animációs módban a teknőc képének *sorszám*. fázisát jeleníti meg
véletlenszám *n*	0 és *n* – 1 közötti véletlenszámot állít elő
figyelj *teknőc(ök)*	az itt megadott teknőc (teknőcök) lesz aktív
kérem *teknőc* [...]	a zárójelbe tett utasítások az itt megadott teknőcnek szólnak
takar? *teknőc(ök)*	az első aktív teknőc takarja-e az itt megadott *teknőcöt* (teknőcöket)?

INTERAKTÍV ANIMÁCIÓ

1. FELADAT: RAJZOLÁS BILLENTYŰZETTEL

A billentyűzetet használjuk fel egy rajzolóprogram írására! A J betű lenyomására a teknőc a képernyőn – abszolút koordinátákban – jobbra mozduljon 10 egységet, a B hatására balra, az F hatására felfelé, az L hatására pedig lefelé.



Az **olvasjel** függvény egy karakter begépelésére vár, majd azt adja vissza eredményül.

```
tanuld rajzoló
  mozog olvasjel
  rajzoló
vége
```

A mozgás egy elágazás, paraméterének értékétől függően másmás tevékenységet hajtunk végre.

```
tanuld mozog :merre
  elágazás :merre ~
    [B [balra 90 előre 10 jobbra 90] ~
     J [jobbra 90 előre 10 balra 90] ~
     F [előre 10] ~
     L [hátra 10]]
vége
```

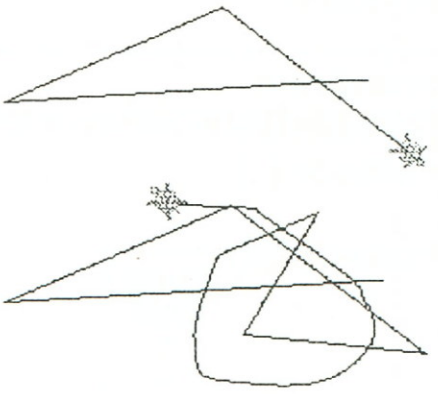
2. FELADAT: KÖVESSÜK AZ EGERET

Az alábbi feladatban az egér követése a cél. A **gombnyomás?** függvény értéke igaz, ha van lenyomva billentyű vagy az egér bal oldali gombja. Az **olvaskód** függvény akkor 0 értékű, ha éppen lenyomtuk az egér gombját, –1, ha lenyomva tartjuk, s –2, ha éppen felengedjük. Az **olvaskód** függvény használata után alkalmazhatjuk az **egér-**

hely függvényt, amely az utolsó művelet végrehajtásakori egérpozíciót adja eredményül.

Az **irányszög** függvény megadja azt az irányt, amerre a paraméterként megadott pont van a teknőc jelenlegi helyzetétől. A **hely - egérhely** egy pontkifejezés, amelynek abszolút értéke a két pont távolságát adja.

```
tanuld követ
  ha gombnyomás? [ha olvaskód=0 []]
  irány! irányszög egérhely
  ha abs hely-egérhely >= 1 [előre 1]
  követ
vége
```



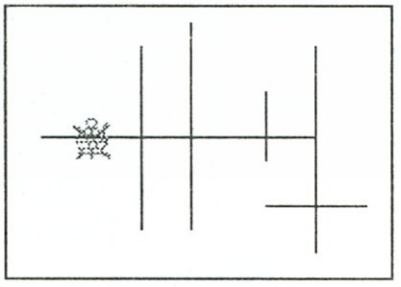
3. FELADAT: ÉRZÉKELŐS TEKNŐC A LABIRINTUSBAN

Rajzoljunk egy labirintust, amelyben vonalak jelzik az utakat, s a köztük levő festetlen pontok pedig a falakat. Egy olyan programot írunk, melynek segítségével a teknőcöt kivezethetjük a labirintusból.

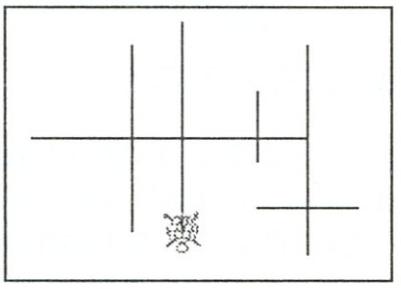
```
tanuld labirintus
  ha hely<>[0 0] ~
    [balra 90 előre 1 ~
     ha festett? [labirintus] ~
       [hátra 1 jobbra 90 előre 1 ~
        ha festett? [labirintus] ~
          [hátra 1 jobbra 90 előre 1 ~
           ha festett? [labirintus] ~
             [hátra 1 jobbra 90 előre 1 ~
              labirintus]]]]
vége
```

```
tanuld festett?
  eredmény pontszín<>rajzlapszín
vége
```

A labirintust jelképező útvesztő egy tetszőleges kezdő állapotban:



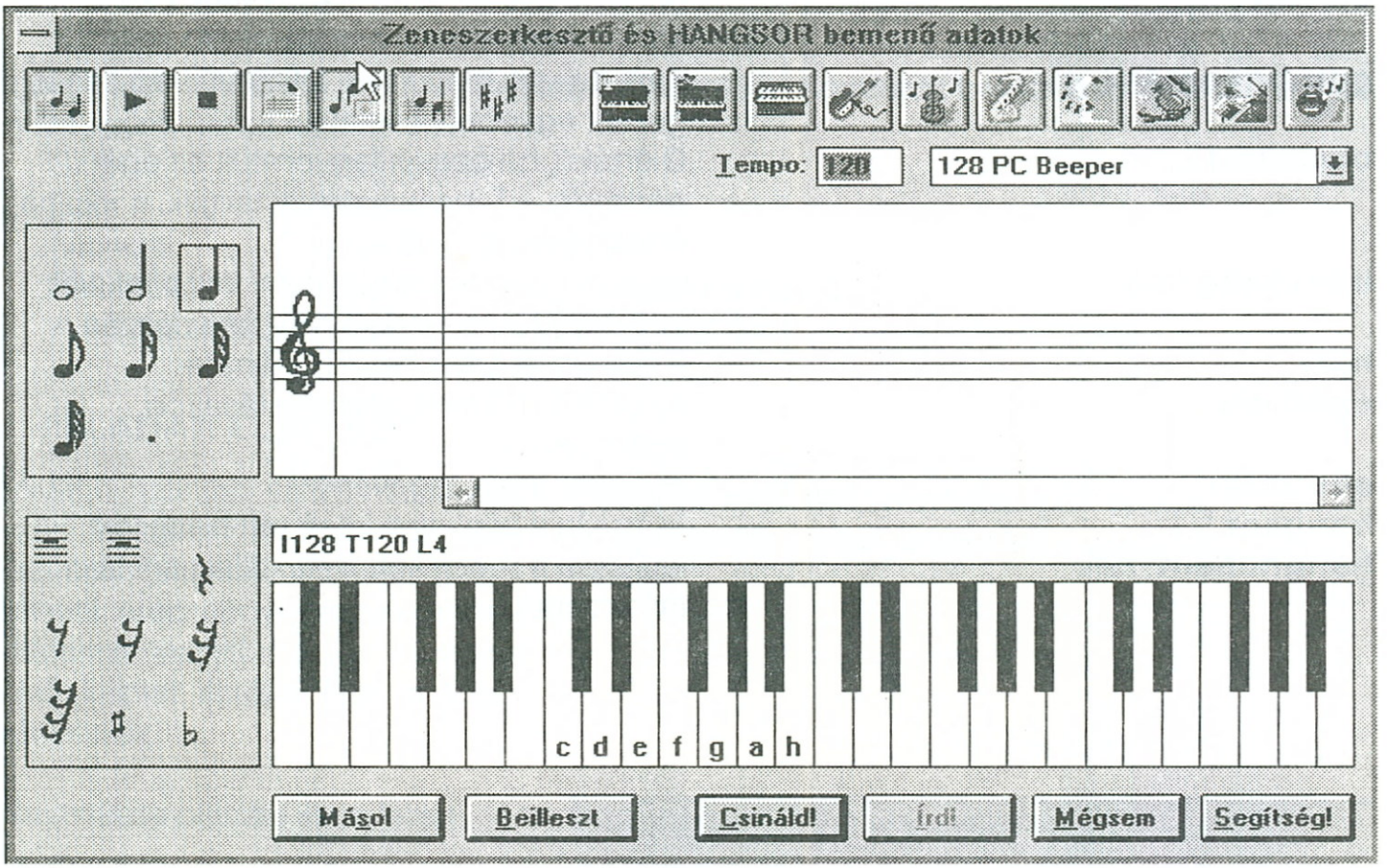
Ugyanez célba éréskor:



HANGOK, ZENESZERKESZTÉS

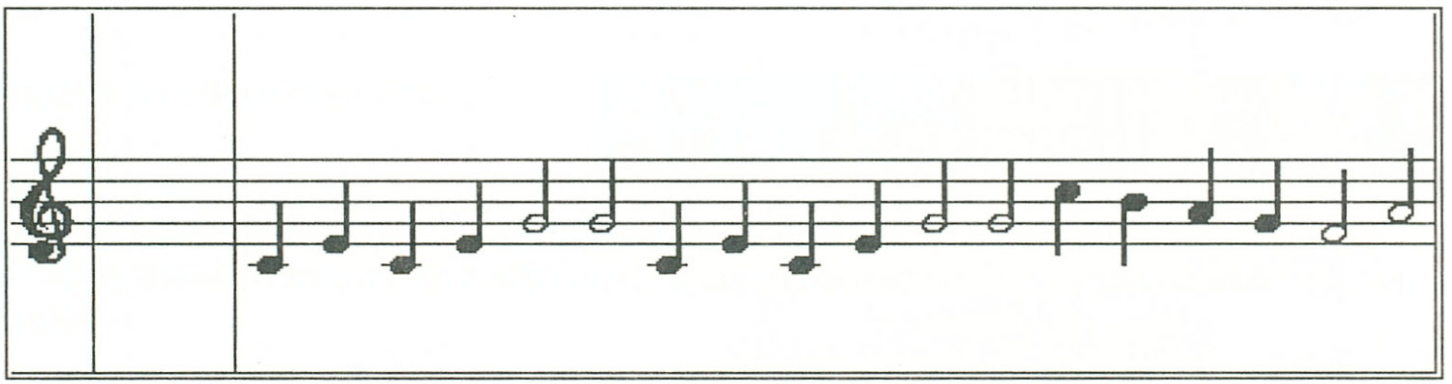
4. FELADAT: HANGSORSZERKESZTÉS A ZENEGÉPPEL

A hangsor utasítás hatására előbukkanó kiegészítő ablakkal zenét lehet szerkeszteni. A kottára vagy a zongorabillentyűzetre kell kattintani, vagy az egérrel húzni a hangjegyeket. A program bőven nyújt kísérletezési lehetőséget.



Vigyázat! A különböző hangszerek hangja csak például Soundblaster kompatíbilis hangkártyával és jó MIDI-beállítások esetén hallható!

A zenemű elkészülte után a **Csináld!** gombra kell kattintani. A zenegép ablaka eltűnik ugyan, de az írólapon a megkomponált zene utasítássora látható, s a zene azonnal hallható.



```
hangsor [I114 T120 L4 O2 C E C E 2G 2G C E C E 2G 2G O3 C O2 H A G
2F 2A]
```

5. FELADAT: HANGHULLÁM LEJÁTSZÁSA

Előre felvett hanghullámok is lejátszhatók az alábbi módon:

```
hanghullám "jaj.wav
```

UTASÍTÁS-ÖSSZEFOGLALÓ

olvasjel	beolvas billentyűzetről egyetlen jelet, s ennek értékét adja eredményül
gombnyomás?	igaz, ha van lenyomva billentyű vagy egérgomb, egyébként hamis
olvaskód	0 értékű, ha éppen lenyomtuk az egér gombját, –1, ha lenyomva tartjuk, s –2, ha éppen felengedjük
irányszög *hely*	az a szög, amennyit fordulni kell a teknőcnek, hogy a megadott hely felé nézzen
hely	a teknőc aktuális helykoordinátái
egérhely	az utolsó művelet végrehajtásakori egérpozíció
abs *szám*	abszolút érték függvény
hangsor [...]	lejátssza a zárójelek közé tett hangsort
hanghullám *"név*	lejátssza a megadott *nevű* állományban tárolt zenét

LOGO FÜGGVÉNYEK

ELEMI SZÖVEGKEZELÉS

A Comenius Logo másik fő része nem rajzolással, hanem szövegmanipulációval, függvényekkel kapcsolatos. Az ehhez tartozó eszközökkel függvényeket definiálhatunk.

A Logo elemi adata a karakter, amelyből számok, illetve szavak épülhetnek fel. A számokból és szavakból mondatokat képezhetünk, a mondatok bekezdéseket alkothatnak. A struktúrák tehát az alábbiak:

Szám:	számjegyek
Szó:	"karakterek
Mondat:	[szavak és számok]
Bekezdés:	[[első mondat] ... [utolsó mondat]]

A következőkben néhány szövegmanipulációs függvényt nézünk meg.

1. FELADAT: ELSŐ SZÓ

Mivel a mondat szavak sorozata, ezért van értelme a mondat első szaváról beszélni. A Logo függvények eredményét az eredmény kulcsszó vezeti be: a mögötte levő képlet értéke lesz a függvényérték.

```
tanuld elsőszó :mondat
  eredmény első :mondat
vége
```

Használata például ilyen lehet:

```
? kiír elsőszó [Comenius Logo]
? Comenius
```

2. FELADAT: ELSŐ BETŰ

Az első betű ugyanúgy első eleme a szónak, ahogyan az első szó az első eleme volt a mondatnak.

```
tanuld elsőbetű :szó
  eredmény első :szó
vége
```

Használata például ilyen lehet:

```
? kiír elsőbetű "Logo
? L
```

Az **első** függvény tehát megadja bármilyen struktúra első elemét.

Az előbbiek mintájára definiálhatjuk az **„utolsószó"** és az **„utolsóbetű"** függvényeket.

3. FELADAT: UTOLSÓ SZÓ

Az **utolsó** függvény (az elsőhöz hasonlóan) egy tetszőleges struktúra utolsó elemét adja eredményül.

```
tanuld utolsószó :mondat
  eredmény utolsó :mondat
vége
```

4. FELADAT: UTOLSÓ BETŰ

```
tanuld utolsóbetű :szó
  eredmény utolsó :szó
vége
```

5. FELADAT: MONDAT CIKLIKUS LÉPTETÉSE SZAVANKÉNT EGGYEL BALRA

A ciklikus balra léptetés alapelve: vegyük le a mondat első szavát, majd az így kapott, egy szóval rövidebb mondat végére (azaz utolsó szavának) tegyük vissza.
Az **elsőnélküli** függvény paramétere elemszámát csökkenti eggyel, mivel elhagyja annak első elemét. Az **utolsónak** függvény ezzel szemben eggyel növeli a sorozatparaméterének elemszámát, a végére illeszt egy elemet.

```
tanuld balralép :mondat
  eredmény utolsónak első :mondat
    elsőnélküli :mondat
vége
```

6. FELADAT: MONDAT CIKLIKUS LÉPTETÉSE SZAVANKÉNT EGGYEL JOBBRA

Megoldása hasonló az előző feladatéhoz, az ott alkalmazott függvények megfelelő párját kell használni.
(elsőnélküli → utolsónélküli, utolsónak → elsőnek)

```
tanuld jobbralép :mondat
  eredmény elsőnek utolsó :mondat
    utolsónélküli :mondat
vége
```

7. FELADAT: FORDÍTSUK MEG EGY MONDAT SZAVAINAK SORRENDJÉT!

Ha a mondat egyetlen szót sem tartalmaz, akkor a megfordítás önmaga. Ha legalább egy szó van benne, akkor vegyük ki az első szót, fordítsuk meg a maradék mondatot, majd tegyük az eredmény végére a kivett első szót.
Az *eredmény* alapszó nemcsak a függvény eredményét határozza meg, hanem egyben be is fejezi a függvény kiértékelését. Ezért a fenti példában az elágazás mögé írt utasítást csak abban az esetben hajtjuk végre, ha az elágazás feltétele *hamis* volt.

```
tanuld mondatfordít :mondat
 ha üres? :mondat [eredmény :mondat] ~
  eredmény utolsónak első :mondat ~
              mondatfordít ~
              elsőnélküli :mondat
vége
```

8. FELADAT: FORDÍTSUK MEG EGY MONDAT SZAVAINAK ÉS BETŰINEK SORRENDJÉT!

Az előző feladat megoldását alkalmazzuk a mondat szavaira, illetve a szavak betűire.

```
tanuld szófordít :szó
 ha üres? :szó [eredmény :szó] ~
  eredmény utolsónak első :szó ~
              szófordít ~
              elsőnélküli :szó
vége
```

```
tanuld mondatfordít :mondat
 ha üres? :mondat [eredmény :mondat] ~
  eredmény utolsónak szófordít ~
              első :mondat ~
              mondatfordít ~
              elsőnélküli :szó
vége
```

9. FELADAT: EGY SZÓ ÖSSZES MAGÁNHANGZÓJÁT CSERÉLJÜK E BETŰRE!

Menjünk végig a szó betűin! Ha magánhangzót találunk, akkor az eredménybe e betűt tegyünk, mássalhangzó esetén pedig magát a betűt.

```
tanuld csere :szó
 ha üres? :szó [eredmény :szó]
 ha eleme? első :szó magánhangzók ~
  [eredmény elsőnek "e csere ~
              elsőnkívüli :szó]
   eredmény elsőnek első :szó csere ~
              elsőnélküli :szó
vége
```

A Logo nyelv konstansai a konstansfüggvények (azaz a paraméter nélküli függvények). Ezt használjuk a magánhangzók megadására.

```
tanuld magánhangzók
  eredmény [a á e é i í o ó ö ő u ú ü ű]
vége
```

10. FELADAT: EGY SZÓ HANGRENDJÉNEK MEGADÁSA SZÓTAGONKÉNT

A szótagok hangrendjéhez a szó magánhangzóit kell megvizsgálni. Az a, á, o, ó, u, ú mély magánhangzók, a többiek pedig magasak.
A magas és a mély magánhangzókat az előző feladat megoldásához hasonlóan egy-egy konstansfüggvénnyel adjuk meg.

```
tanuld hangrendek :szó
  ha üres? :szó [eredmény :szó]
  ha eleme? első :szó mélyek ~
    [eredmény elejére "mély hangrendek ~
                elsőnélküli :szó]
  ha eleme? első :szó magasak ~
    [eredmény elejére "magas hangrendek ~
                elsőnélküli :szó]
    eredmény hangrendek elsőnélküli :szó
vége
```

```
tanuld mélyek
  eredmény [a á o ó u ú]
vége
```

```
tanuld magasak
  eredmény [e é i í ö ő ü ű]
vége
```

11. FELADAT: EGY MONDAT SZAVAI HANGRENDJÉNEK MEGADÁSA

A mondat feldolgozása a szavainak feldolgozását jelenti.

```
tanuld hangrend :mondat
  ha üres? :mondat ~
    [eredmény :mondat]
     eredmény elejére szóhangrend ~
                első :mondat ~
                hangrend elsőnélküli :mondat
vége
```

A szavak lehetnek magas, mély vagy vegyes hangrendűek.

```
tanuld hangrendek :szó
  ha vanmély? :szó ~
    [ha vanmagas? :szó ~
      [eredmény "vegyes] ~
      [eredmény "mély]]
  ha vanmagas? :szó ~
    [eredmény "magas] ~
    [eredmény "hibás]
vége
```

Az alábbiakban eldöntjük, hogy egy szó betűi között van-e magas, illetve mély magánhangzó.

```
tanuld vanmagas? :szó
  ha üres? :szó [eredmény "hamis]
  ha eleme? első :szó magasak ~
    [eredmény "igaz]
  eredmény vanmagas? elsőnélküli :szó
vége
```

```
tanuld vanmély? :szó
  ha üres? :szó [eredmény "hamis]
  ha eleme? első :szó mélyek ~
    [eredmény "igaz]
  eredmény vanmély? elsőnélküli :szó
vége
```

UTASÍTÁS-ÖSSZEFOGLALÓ

eredmény	a Logo függvények eredménye következik utána
üres? *:sorozat*	igaz, ha a paramétere üres *sorozat*, hamis, ha nem
eleme? *:elem :sorozat*	igaz, ha az *elem* eleme a *sorozatnak*
első *:sorozat*	a *sorozat* első elemét adja
utolsó *:sorozat*	a *sorozat* utolsó elemét adja
elsőnélküli *:sorozat*	a *sorozat* első elem nélküli részsorozatát adja
utolsónélküli *:sorozat*	a *sorozat* utolsó elem nélküli részsorozatát adja
elsőnek *:elem :sorozat*	a *sorozat* első eleme elé szúrja be az új elemet
utolsónak *:elem :sorozat*	a *sorozat* utolsó eleme mögé szúrja be az elemet

AZ OKTATÁS MÓDSZERTANI KÉRDÉSEI LOGO KÖRNYEZETBEN

A LOGO ÉS A NAT

A Nemzeti Alaptanterv hatályba lépésével önálló műveltségi területté válik az informatika, így sok tanárban felmerül az a kérdés, hogy mit és mivel tanítson.

MIVEL TANÍTSUNK:

HARDVER – ez elsősorban az oktatási intézmény anyagi lehetőségeitől függ.

SZOFTVER – sokan nem számolnak vele, de igen komoly anyagi vonzata van, ezért nagyon végig kell gondolni, hogy milyen programokra lesz szükség az oktatásban.

Az informatikaoktatás feladata az információ kezelésével kapcsolatos jogi és etikai szabályok elsajátíttatása is, például a jogtiszta szoftverhasználat. Itt is igaz a mondás: legjobb nevelés a példamutatás. Vásároljuk meg, legálisan szerezzük be a tanításhoz használt programokat!

A Comenius Logo hasznosítása elsősorban az informatika különböző tárgyterületein képzelhető el. A következőkben azonban más tárgykörökkel való lehetséges integrálását is felvetjük. Célszerű a különböző tárgyterületeket oktató tanárok konzultálása az együttes tananyag kialakításában.

Kiemelt pontok a számítástechnika általános fejlesztési követelményeinek tárgyterületéből:

2. (A tanuló) tudjon információt különféle formákban kifejezni; legyen képes a különböző formákban megjelenített információt felismerni.
3. Szerezzen tapasztalatokat a hagyományos és az új technológiákon alapuló informatikai eszközök és információhordozók használatában.
4. Legyen képes a gyakorlati életben használt legfontosabb írásos formátumok gépi megvalósítására, legyen igénye a mondanivaló lényegét tükröző esztétikus külalak kialakítására.
5. Legyen képes az adott probléma megoldásához kiválasztani az általa ismert módszerek és eszközök közül a megfelelőket.
9. Ismerkedjen az informatika és a társadalom kölcsönhatásával.

A műveltségi területek oktatásának közös követelményei között szerepel többek között a kommunikációs kultúra, a megismerést, a tanulást, a tudást, az emberi kapcsolatokat, az együttműködést, a társadalmi érintkezést szolgáló információk felfogása, megértése, szelektálása, elemzése, értékelése, felhasználása, közvetítése, alkotása. Összetevői a szimbolikus (verbális, matematikai) jelek útján történő, a képi, valamint a mozgásban, a tevékenységben, a magatartásban megnyilvánuló kommunikáció képességei.

Napjainkban értesüléseink túlnyomó részét nem személyesen hozzánk intézett „üzenetekből" merítjük, hanem mesterséges közvetítő rendszerek útján. A tömeges, passzív információfogyasztás az életvitel és gondolkodás torzulásához vezethet. Ezért az iskoláknak az új audiovizuális környezetet értő, szelektíven használó fiatalokat kell nevelnie.

A számítógépek felhasználásával az audio és vizuális információk megalkotását a végső formától (papír vagy elektronikus/multimédia) függetlenül, a megfelelő részek kihangsúlyozásával kell elérni, amelyhez az illusztrációk, hangok és tipográfia, sőt az animáció is hozzátartoznak.

Az informatikán kívül érdemes tovább vizsgálni más műveltségi területekhez tartozó ismeretek kapcsolatait.

Kiemelt pontok az egyéb általános fejlesztési követelmények tárgyterületéből:

Magyar nyelv

5. A tanulási képesség fejlesztése, az alapműveltség megszerzéséhez szükséges ismeretfeldolgozás kulturális technikáinak ismerete és használata.

Vizuális kultúra

1. A vizuális nyelv alapjai.
3. A vizuális kommunikáció. A vizuális jelenségek, információk megértése és saját gondolatok közérthető megjelenítése. Az információt megbecsülő, azt kritikusan szemlélő magatartás alapjai.

Ének-zene

II. A zenei hallás fejlesztése

c) Zenei élmény szóbeli, vizuális, mozgásos megfogalmazása.

A TANANYAG KOROSZTÁLYOKRA VALÓ BONTÁSA

4. ÉVFOLYAM VÉGÉRE

VIZUÁLIS KULTÚRA

„Különböző témák, történetek megjelenítése. Különböző, meghatározott célú vizuális közlések létrehozása, megfelelő technikai megoldásokkal."

6. ÉVFOLYAM VÉGÉRE

VIZUÁLIS KULTÚRA

„Események előadása képsorozatokban, egyszerű animációval. Változást, fejlődést, folyamatot szemléltető ábrák készítése. Gondolati tartalmak, információk képi sűrítése, forma- és színredukálás. A rögzített közlés, információ. A képi közlések szabályainak, konvencióinak tanulmányozása ábrákon, közismert jelekben, szimbólumokban, betű- és szövegképekben, egyszerű mozgóképi egységekben. Szöveg- és kép-összeállítási, elrendezési technikák."

8. ÉVFOLYAM VÉGÉRE

MAGYAR NYELV
„A magán- és közéleti kommunikáció, az írásos és az élőszóbeli szövegek felfogásának és alkotásának különbsége."

VIZUÁLIS KULTÚRA
„Szöveg és kép összekapcsolása különböző karakterű közlésekben. A tömegkommunikáció legismertebb formái. Kiadványok, műsorok kategorizálása különböző szempontok alapján. A tartalom, a közlési szándék és külső megjelenés összefüggése. A szövegírás technikái."

MOZGÓKÉPKULTÚRA ÉS MÉDIAISMERET
„Gyakorlati ismerkedés a mozgókép elemi kifejezőeszközeivel: kompozíció a térben és az időben. A technikai képírás fejlődése. A tömegtájékoztatás fajtái. A tömegkommunikációs rendszerekben megjelenő üzenet hatása."

10. ÉVFOLYAM VÉGÉRE

Vizuális kultúra
„A vizuális nyelv sajátosságai (összefüggései a kifejezés más formáival – verbális, zenei). A kontextus. Néhány különleges technika és alkotói módszer kipróbálása. Elvont, nem vizuális természetű információk megjelenítése különböző vizuális hatáselemekkel. Mozgás, hang, fény és statikus látványelemek együttes alkalmazása. A tömegkommunikáció képi közléseinek elemzése. A tömegkommunikáció hatásának jellemzői. Ábrák a nyomtatványban. A műszaki rajz technikája."

MOZGÓKÉPKULTÚRA ÉS MÉDIAISMERET
„Hogyan fogalmaz a mozgókép: a jelentésalkotás eszközei. A látvány (és hangzóvilág) térbeli és időbeli megszervezése. A mozgókép alkalmazási területei. ‚Információs országút' – interaktív médiumok."

A számítógép új író-, rajzoló-, animáló eszközt képvisel, melynek alkalmazása nélkülözhetetlen a hatékony információátadás elsajátításában. Azonban a technikai ismereteket feltétlenül a tartalom megalkotásának elsődleges szempontjai szerint kell alkalmazni. A korosztályokra bontott tananyag kialakításában meg kell oldani a társterületek témáinak integrálását, hogy az információátadás: a beszerzés, elemző nyelvi alkotás és esztétikai megjelenítés minden hatékony mozzanatát foglalja magában.

A különböző folyamatok, kísérletek, elméletek, gyakorlati modellek felfoghatóbbá, rögzíthetőbbé válnak, ha azokat aktív és kreatív szimulációs játékokon keresztül ismerhetjük meg. Sok esetben az egyetlen kísérletezési formát jelenti például veszélyes vagy nagyon költséges kísérletek végrehajtásakor.

Egy egyszerű programozási nyelv elsajátításával, az utasítások gyakorlati alkalmazásával jobban meg lehet érteni a számítógépek működését, annak korlátait és bővíthetőségét. Az egyes szakterületen felmerülő elvek, absztrakt elméletek modellezése nagymértékben segíti a tananyag elmélyítését az aktív alkotás művelésén keresztül. A szimulációs programok és modellező eszközök használata minden egyes műveltségi részterületen kivívhatja létjogosultságát, megfelelő eszközök birtokában.

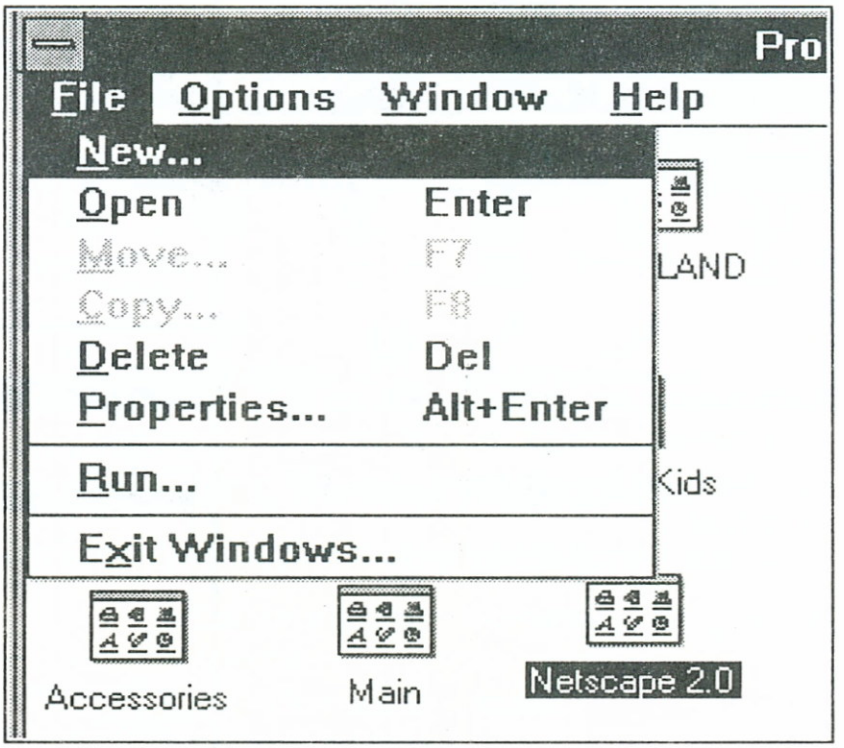
A Comenius Logo nagyban hozzájárulhat a fent említett célok eléréséhez.

FÜGGELÉK

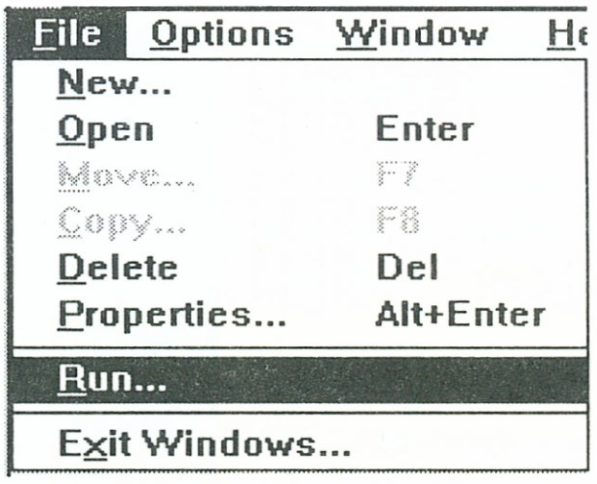
A COMENIUS LOGO PROGRAMCSOMAG INSTALLÁLÁSA

Kapcsoljuk be a számítógépet! Indítsuk el a Windowst! Ha nem a winchesterről installáljuk a Comenius Logót, akkor tegyük be az 1. sorszámú lemezt a lemezmeghajtóba!

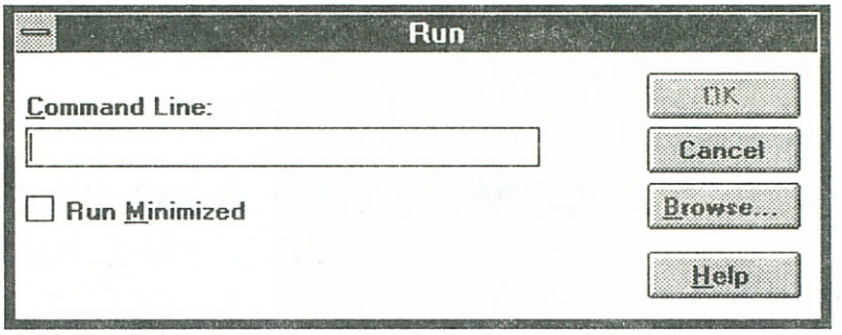
Kattintsunk a Programkezelő **File** menüpontjára!



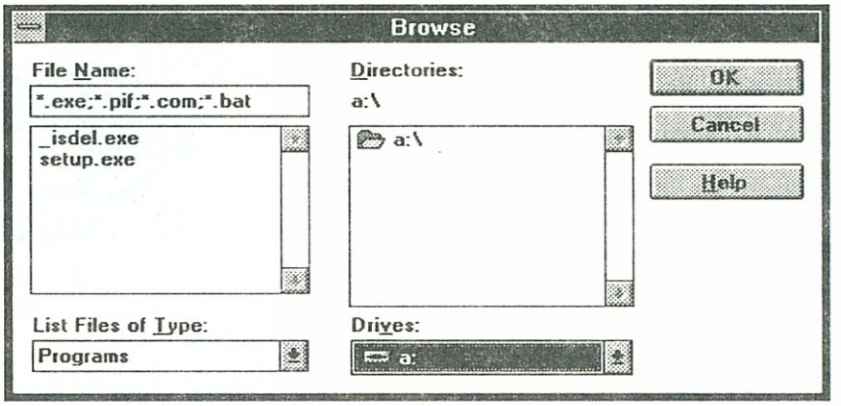
Kattintsunk a **Run** (Futtat) menüpontra!



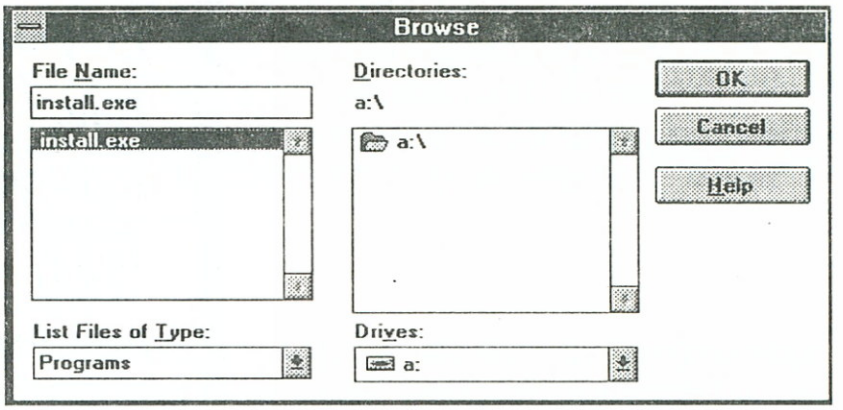
Kattintsunk a **Browse** (Tallóz) gombra!



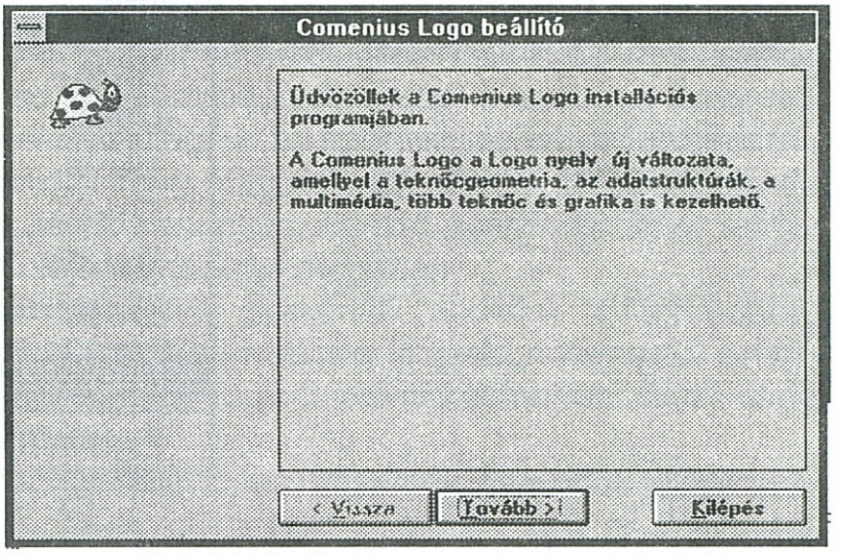
Válasszuk ki annak a meghajtónak a nevét, amelyikbe az installációs lemezt behelyeztük (ez lehet a: vagy b: vagy a winchester megfelelő könyvtára)!



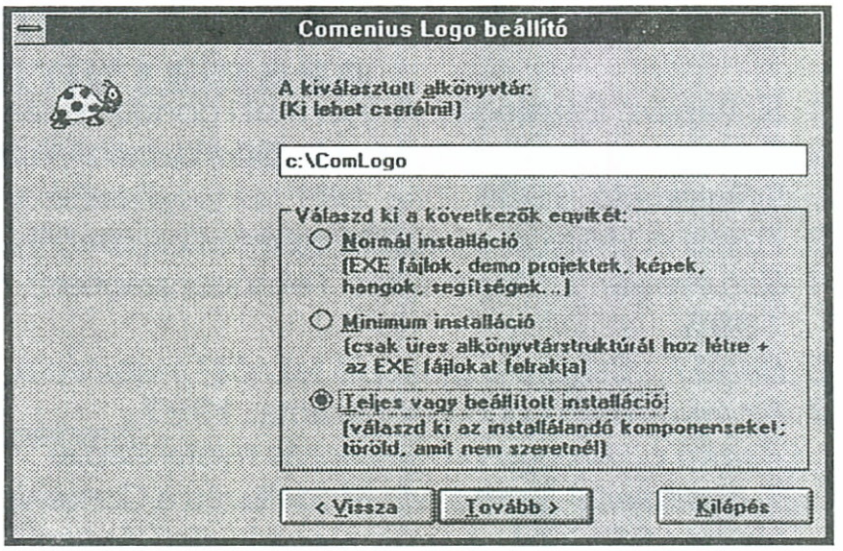
Kattintsunk az **install.exe** program nevére, majd az **OK** gombra! Kattintsunk megint az **OK** gombra!



Ezek után már csak követni kell az installáláskor megjelenő utasításokat.



Figyelem! A programcsomagot részleteiben is lehet installálni. Választhatunk minimális, normál és teljes installáció közül. Az egész programcsomaghoz a teljes installálást kell választani.



Ha a program installálása befejeződött, akkor a következő ablak látható a Programkezelőben:

KÖNYVTÁRSZERKEZET, ÚTVONAL MEGHATÁROZÁSA

Alapvető könyvtárszerkezet az installálásnak megfelelően a következőképpen alakul:

COMLOGO	**—DEMO**	a Demo ablakban megjelenő programok
	|–FELADAT	a feladatok
	|–GYEREK	a gyerekprogram részei
	|–HANG	a hangállományok
	|–KEPSOR	az LGW-állományok
	|–PROJEKT —ANIMOTOR	az Animotor program
	|–KIMEL	a kímélő programok
	|–RAJZLAP	BMP-állományok

Fontos, hogy ezt a szerkezetet így hagyjuk és a mentéseket a megfelelő helyre tegyük. Ha az egy projekthez tartozó összes állományt ugyanabban az alkönyvtárban tároljuk, akkor a különböző állományok betöltésekor jelezni kell, hogy a projekttel megegyező helyről töltse be a programot.

Példa:

BETÖLTRAJZLAP BETÖLTŐÚT "rajz.bmp

Ha ezt elmulasztjuk, akkor a következő hibaüzenetet kapjuk:

Problémáim vannak a(z) "rajz.bmp fájllal!

JAVASOLT IRODALOM

1. *S. Papert:* Észrengés. A gyermeki gondolkodás titkos útjai.. SZÁMALK, 1988.
2. *Turcsányiné Szabó M.–D. Senftleben:* A Logo programozási nyelv. Műszaki Könyvkiadó, 1986.
3. *Tuska Á.:* Mire jó a Logo? *Inspiráció,* 1. évf. 6. szám, 22–23., 1993.
4. *Schmieder L.:* Nyelvújítás Teknőcországban? *Inspiráció,* 3. évf. 3. szám, 23–24., 1995.
5. *Bedő F.:* Csiga-e a gyorsuló teknős? *Inspiráció,* 4. évf. 1. szám, 13–15., 1996.
7. Az *Inspiráció* Logo különszáma, 1997:
   Zsakó L.: A Logo programozás nyelvi alapjai.
   Turcsányiné Szabó M.: Modellezés a Comenius Logo felhasználásával.
   Vajdáné Szili M.: A programozás első lépései Logo nyelven.
   Kiss Zs.: Csillagok rajzolása.
   Bedő F.: Csiga-e a gyorsuló teknős?
   M. Tomcsányiová: Logo óratervek.
   Hoffmann A.: A strukturált programozás örömei Logóban.
8. *Turcsányiné Szabó M.:* Alkalmazói rendszerek a NAT tükrében. *Informatika a felsőoktatásban,* Debrecen, 1996.

COMENIUS LOGO 3.0 VERZIÓ DEFINITÍV LEÍRÁSA

A Comenius Logo eljárásai a paraméterek bonyolultsága szempontjából két csoportba oszthatóak: **alap** és **haladó**. A haladó lehetőségeket **Hld:** jelöli. Ha az eljárásnak egy vagy több paramétere van, akkor azt a következőképpen jelöljük:

paraméter
(paraméter1 paraméter2 ...)

Ekkor az eljárás az alábbi módon használható:

ELJÁRÁS paraméter
és
(ELJÁRÁS) *; nincs paraméter*
(ELJÁRÁS paraméter1) *; egy paraméter*
(ELJÁRÁS paraméter1 paraméter2) *; kettő paraméter*
(ELJÁRÁS paraméter1 paraméter2 ...) *; akárhány paraméter*

Ha az eljárásnak két vagy több paramétere lehet, akkor azt a következőképpen jelöljük:

paraméter1 paraméter2
(paraméter1 ...)

A Logo eljárás **utasítás,** ha nincs eredménye – ilyenkor az **EREDMÉNY** oszlop üres –, és **művelet,** ha mind a **PARAMÉTEREK**, mind pedig az **EREDMÉNY** oszlop nem üres.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
.*	*szám1 * szám2* **Hld:** *pont * szám* **Hld:** *szám * pont* **Hld:** *pont1 * pont2*	*szám* *pont* *pont* *szám*	Eredménye a bemenő adatok szorzata.
+	*szám1 + szám2* *+szám* **Hld:** *pont1 + pont2* **Hld:** *+pont*	*szám* *szám* *pont* *pont*	Eredménye a bemenő adatok összege.
-	*szám1 - szám2* *- szám* **Hld:** *pont1 - pont2* **Hld:** *- pont*	*szám* *szám* *pont* *pont*	Eredménye a bemenő adatok különbsége.
/	*szám1 / szám2* **Hld:** *szám / pont* **Hld:** *pont / szám*	*szám* *pont* *pont*	Eredménye a bemenő adatok hányadosa.
<	*szám1 < szám2* *Hld: szó1 < szó2* *Hld: pont1 < pont2*	*logikai érték* *logikai érték* *logikai érték*	IGAZ, ha a szám1 kisebb, mint a szám2, egyébként pedig HAMIS.
<=	*szám1 <= szám2* **Hld:** *szó1 <= szó2* **Hld:** *pont1 <= pont2*	*logikai érték* *logikai érték* *logikai érték*	IGAZ, ha a szám1 kisebb vagy egyenlő, mint a szám2, különben HAMIS.
<>	*akármi1 <> akármi2*	*logikai érték*	IGAZ, ha a két paraméter különbözik egymástól, egyébként HAMIS.
=	*szám1 = szám2* *szó1 = szó2* *lista1 = lista2* *akármi1 = akármi2*	*logikai érték*	IGAZ, ha a két operandusa egyenlő, egyébként HAMIS.
>	*szám1 > szám2* *Hld: szó1 > szó2* *Hld: pont1 > pont2*	*logikai érték* *logikai érték* *logikai érték*	IGAZ, ha a *szám1* nagyobb, mint a *szám2*, különben HAMIS.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
>=	*szám1 >= szám2* *Hld: szó1 >= szó2* *Hld: pont1 >= pont2*	*logikai érték* *logikai érték* *logikai érték*	IGAZ, ha a *szám1* nagyobb vagy egyenlő, mint a *szám2*, egyébként HAMIS.
.ALAPSZAVAK .PRIMITIVES	*szám, a következők valamelyike:* *0 –* *1 –* *2 –* *3 –*	*lista*	A Logo alapszavak listája: – angol és magyar alapszavak, – magyar alapszavak, – angol és magyar alapszavak rövidítése, – magyar alapszavak rövidítése.
.GYŰJTMEM .CLEANMEM	*nincs*	*nincs*	Általános hulladékgyűjtési folyamatot indít el, a felhasználatlan memória-területek újrahasznosításával.
.LÉS .AND			
.LKVAGY .XOR			
.LNEM .NOT			
.LVAGY .OR			
.PARAMÉTEREK .MEM			
.PONTHELY .WINDOWSPOS	*(pont)* *([])*		Eredménye a *pont* koordinátája (Windows koordináta-rendszerben). A rajzlap bal felső sarkának abszolút koordinátáját adja.

A, Á

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ABLAK WINDOW	*nincs* *(pont)* *(téglalap)* *([ ])*		Az aktív teknőcök számára definiálja a felhasználható területet. A *pont* koordinátája és a rajzlap jobb alsó sarka által meghatározott terület használható. Az adott *téglalap* a felhasználható terület. Újradefiniálja a felhasználható területet.
ABSZ ABS	*szám* **Hld:** *pont*	*szám* *szám*	Eredménye a bemenő szám abszolút értéke.
ALAK GETSHAPE	*nincs*	*képsor*	Eredménye az a képsor, ami pillanatnyilag az első teknőc alakja.
ALAK! SETSHAPE, SETTURTLE	*([])* *képsor* *LGW_fájl* **Hld:** *"Vágólap*		Az aktív teknőcök aktuális alakja az alapalakot veszi fel, egyébként a *képsor*, illetve az *LGW_fájl* tartalmát használja. A *Vágólap* aktuális tartalma lesz az aktív teknőcök új alakja.
ALAKSZÍN SHAPECOLOR	*nincs*	*szín* *[ ]*	Eredménye az első aktív teknőc aktuális aktív alakszíne, illetve [ ], ha nem határoztunk meg aktív alakszínt.
ALAKSZÍN! SETSHAPECOLOR	*szín* *[ ]*		Az aktív teknőcök aktív alakszíne *szín* lesz. Nem lesz aktív alakszín.
ALAPHELYZET DRAW	*nincs*		Előkészíti a rajzlapot az új rajzoláshoz: letörli a képernyőt, törli az összes teknőcöt, és a 0 nevű teknőcöt ismét létrehozza, valamint aktívvá és láthatóvá teszi.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ALAPPALETTA RESETPALETTE, RESETPAL	*nincs*		Abban az esetben használjuk, ha 256 színt alkalmazó módban dolgozunk, és egy betöltendő rajz ettől eltérő palettát használ.
ALAPSZÓ? PRIMITIVE?	*szó*	*logikai érték*	IGAZ, ha a *szó* egy Logo alapeljárás neve, különben HAMIS.
AMÍG WHILE	*[kifejezés] utasításlista*		Amíg a *kifejezés* értéke igaz, az *utasításlista* összes utasítását újra és újra végrehajtja.
ANIMÁCIÓSMÓD! SETPHASEMODE	*logikai érték*		Meghatározza azt a módot, ahogyan az aktív teknőcök alakja változtatható. Fázis animáció létrehozásához az IGAZ beállítást használjuk.
ARCTG ARCTAN	*szám*	*szög*	Eredménye az a (fokokban mért) *szög*, amelynek tangense a paraméterként adott *szám*.
ÁLLAPOT TURTLESTATE	*nincs*	*[X Y szög szó1 szó2 képsor]* ahol *szó1:* TL, TF, TRD vagy TVT, és *szó2.* LÁTHATATLAN vagy LÁTHATÓ	Az első aktuális teknőc aktuális állapotát adja vissza.
ÁTDOB THROW	*szó*		Leállítja az aktuális eljárást és az őt hívókat is mindaddig, amíg nem talál ELKAP szó *utasításlista* utasítást.
ÁTÍR EDIT, ED	*memória_objektum* *[memória_objektum* **Hld:** *(képsor_változó szám)]*		Megnyitja és aktívvá teszi a *memória_objektum* szerkesztőjét.

Teljes név/rövidítés	Bemenő paraméterek	Eredmény	Hatása
ÁTÍRFÁJL EDITFILE, EDFILE	*fájl*		Kinyitja a Fájlszerkesztő ablakot és aktívvá teszi. Ebben a módban a Logo lehetőséget biztosít megfelelő *szövegfájl*ok szerkesztésére, illetve létrehozására.
ÁTLÁTSZÓMÓD! SETIMODE	*0 vagy* *1 vagy* *2*		A háttérszínnel egyező alakpontok átlátszóak lesznek, mentésnél fekete (alapbeállítás). Egy pont sem átlátszó. A külső pontok közül a háttérszínnel egyezők lesznek átlátszóak.

B

BALRA, B LEFT, LT	*szög*		Az összes aktív teknőc balra fordítása *szög*gel.
BEÁLLÍTÓ CHOOSER	*Logo_eljárás* *Logo_eljárás*	*[Logo_utasítás]* *[ ]*	A Beállítóablak megjelenítésével lehetővé teszi, hogy a felajánlott lehetőségek közül kiválasszuk az *eljárás*hoz tartozó megfelelő bemenő paramétereket.
BELÜL? INSIDE?	*téglalap* *pont téglalap* *teknőc téglalap*	*logikai érték*	IGAZ, ha az első aktuális teknőc a területen belül vagy a határán van. IGAZ, ha a pont a területen belül vagy a határán van. IGAZ, ha a teknőc a területen belül vagy a határán van.
BETOLDKÉP PUTIMAGE	*képsor* *LGW_fájl*		Minden aktív teknőc kinyomtatja a *képsor*-, illetve az *LGW_fájl*ban tárolt képsor első fázisát.
BETÖLT LOAD	*LGO_fájl* **Hld:** *"Vágólap*		A Logo fájlban tárolt parancsokat betölti és végrehajtja.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
BETÖLTFELADAT LOADPROBLEM	*TXT_fájl* *(TXT_fájl [szám1 szám2 szám3 szám4 szám5])*		A TXT fájl tartalma által meghatározott Feladatablakot tölti be. *szám1 szám2* az ablak bal felső sarkának absz. koordinátái. *szám3* a szövegoszlopok száma; szám4 a szövegsorok száma. *szám5*=0 esetén az ablak mozgatható és méretezhető. *szám5*=1 esetén az ablak mozgatható, de nem méretezhető. *szám5*=2 esetén az ablak nem mozgatható, de méretezhető. *szám5*=3 esetén az ablak nem mozgatható és nem is méretezhető.
BETÖLTKÉPSOR LOADIMAGE	*LGW_fájl [ ]* *LGW_fájl intervallum* **Hld:** *LGW_fájl [szegm1 szegm2..]* **Hld:** "*Vágólap [ ]* **Hld:** "*Vágólap szegmens* **Hld:** "*Vágólap [szegm1 szegm2 ...]*	*képsor* *képsor* *képsor* *képsor* *képsor* *képsor*	Az *LGW_fájl* által meghatározott képsort tölti be. *[szegm1 szegm2] szegm1*-edik fázisától *szegm2* db fázist tölt be. A megfelelő szegmensek által meghatározott fázisokat összevágva tölti be. A fentiekhez hasonlóan a vágólap tartalmát tölti be, ha az LGW szerkezetű.
BETÖLTKÖTET LOADLIBRARY, LOADLIB	*projekt_kötet* **Hld:** "*Vágólap*		Az adott *projekt_kötet*et, illetve a *Vágólap*ot tölti be, ha az projekt formátumú.
BETÖLTŐÚT LOADPATH	*szó* *( )*	*fájl útvonal*	Megadja a *szó*ban lévő fájl kiterjesztését és elérési útvonalát. A legutóbb betöltött fájl útvonalát kapjuk meg.
BETÖLTRAJZ LOADBITMAP	*BMP_fájl* **Hld:** "*Vágólap*	*képsor* *képsor*	Egyfázisú képpé alakítja a *BMP_fájl*, illetve a *Vágólap* tartalmát, ha az BMP formátumú volt.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
BETÖLTRAJZLAP LOADSCREEN	*BMP_fájl* **Hld:** *(BMP_fájl pont)* *Hld: (BMP_fájl téglalap)* **Hld:** *projekt_rajzlap* **Hld:** *(projekt_rajzlap pont)* **Hld:** *(projekt_rajzlap téglalap)* **Hld:** *"Vágólap* **Hld:** *("Vágólap pont)* **Hld:** *("Vágólap téglalap)*	*kép* *kép* *kép*	Betölti a *BMP_fájl* tartalmát a rajzlapra. A *pont* a kép bal felső sarkát határozza meg. A rajzlap *téglalapnyi* területére tölti a képet.
BETŰ! SETTTEXT, SETTT	*szöveg [szám1 szám2 szám3 szám4 szám5]* ahol *szám1* nagyobb vagy egyenlő 0-nál, *szám2* [0, 1000], *szám3, szám4* és *szám5* is vagy 0, vagy 1		Betűkészlet és -méret beállítása, amelyet az aktuális teknőc a BETŰZD utasítás hatására a rajzlapra íráskor használ. szöveg a betűkészlet neve; *szám1* a méret; *szám2* a kövérség; *szám3*=1 esetén dőlt, *szám3*=0 esetén nincs döntve; ha *szám4* nem adott, akkor a szöveg mögött nincs háttérszín; *szám5*=0 esetén a téglalap bal felső sarka, *szám5*=1 esetén pedig a közepétől pozlcionál.
BETŰNÉGYZET TEXTSIZE	*szöveg*	*[szám1 szám2 szám3 szám4]*	Az első aktuális teknőc által kiírt szöveg befoglaló négyzetét adja *(szám1 szám2)*
BETŰZD TTEXT, TT	*szöveg* *(szöveg1 szöveg2 ...)*		Minden aktuális teknőc kiírja a *szöveg*et a BETŰ-ben beállított típussal, mérettel stb.
BEZÁRFELADAT CLOSEPROBLEM	*TXT_fájl*		A TXT fájl tartalma által meghatározott Feladatablakot zárja be.
BEZÁRVIDEO CLOSEVIDEO	*szó*		Az utasítás az adott ablakkal kapcsolatban lévő videofájlt, valamint a *szó* nevű ablakot zárja be.

C, CS

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
CIKLUS FOR	*szó [N1 N2] utasítás_lista* *szó [N1 N2 N3] utasítás_lista*		*N1<N2*, ciklus *szó=N1*-től *N2*-ig (*N3* lépésközzel).
CIKLUSCSAK FOREACH	*szó akármi utasítás_lista*		A ciklus az *akármi* összes elemére végrehajtja az *utasítás_listá*t.
COS COS	*szög*	*szám*	Eredménye a paraméterként adott (fokokban mért) *szög* koszinusza.
CSATOL APPEND	*LGO_fájl (fájl obj2 obj3 ...)* **Hld:** *LGO_fájl obj2 obj3 ...* ahol *minden obj2, obj3, ... vagy memória_objektum vagy memória_objektum_lista*		Az összes felsorolt objektumot egy megadott nevű fájlba menti. Ha ilyen nevű fájl már létezik, akkor a kijelölt objektumokat a fájl végéhez írja.
CSERE REPLACE	*szám lista akármi* *szám szó1 szó2* *szám képsor1 képsor2* *intervallum lista1 lista2* *intervallum szó1 szó2* *intervallum képsor1 képsor2*	*lista* *szó* *képsor* *lista* *szó* *képsor*	A második bemenet *(lista, szó1, ...) szám* elemét a harmadik bemenetre változtatja. Az *intervallum* által megadott elemeket kicseréli a *lista2*-vel a *lista1*-ben.
CSÖKKENT DEC	*változónév (változónév szám)*		A *változónév* nevű változó értékét 1-gyel, illetve *szám*mal csökkenti.

D, DZ, DZS

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
DÁTUM DATE	*nincs*	*[nap hónap év a_hét_napja]* ahol *a_hét_napja egész szám* [0 6], 0 a vasárnapot jelöli	Eredménye a dátum, megadott formátumban.
DÁTUMSZÓ DATEWORD, DATEW	*nincs* **Hld:** *(dátum_formátum)* **Hld:** *(dátum_formátum dátum)*	*szó* *szó* *szó*	Az aktuális dátumot a Windows Vezérlőpult Nemzetközi beállításának megfelelő formára alakítja.

E, É

EGÉRALAK! SETMOUSECURSOR, SETMC	*képsor* *LGW_fájl szám* ahol *szám egész* [0, 15] *[ ]*		Az egérkurzor aktuális alakját állítja be képsor alakúra. Különböző adott alakúra. Az eredeti alak visszaállítása.
EGÉRÁLLAPOT MOUSESTATE	*nincs* *nincs*	*[szám1 szám2 xy]* *[szám1 szám2 [ ]]*	*szám1* egész [0, 7] melyik gomb volt lenyomva; *szám2* egész [0, 3], le volt-e nyomva speciális billentyű (shift, ctrl); *xy* hely.
EGÉRHELY MOUSE	*nincs*	*pont [x y]*	Az egér koordinátái.
EGÉRHELY! .SETMOUSEPOS	*pont*	*nincs*	A meghatározott pontba helyezi az egérkurzort.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
EGÉSZHÁNYADOS, EH DIV	*szám 1 szám2* *Hld: pont szám* *Hld: pont 1 pont2*	*szám* *pont* *pont*	A *szám 1* és *szám2* egész hányadosa.
EGÉSZRÉSZ INT	*szám* **Hld:** *pont*	*szám* *pont*	A paraméterénél nem nagyobb, legnagyobb egész szám.
EGYENLŐ? EQUAL?	*akármi 1 akármi2* *(akármi 1 ...)*	*logikai érték* *logikai érték*	IGAZ, ha a két paraméter azonos szó (szövegkonstans vagy szám), azonos képsor vagy lista, egyébként HAMIS.
EGYMÁSUTÁN EACH	*Logo_utasításlista*		Az összes aktív teknőc egymás után végrehajtja az *utasításlistá*t.
ELÁGAZÁS CASE	*akármi [ág 1 ág2 ...]* *akármi [ág 1 ág2 ...]*	 *akármi*	Azt az *ág*at hajtja végre vagy értékeli ki, amelynek *feltétele* megegyezik *akármi* értékével.
ELEM ITEM	*szám akármi* *intervallum akármi* **Hld:** *[szegm 1 szegm2 ...] akármi*	*akármi* *akármi* *akármi*	Az *akármi szám* elemét adja. Az *akármi intervallum* által meghatározott része.
ELEME MEMBER	*szó 1 szó2* *akármi lista*	*sorszám,* *semmi*	A második paraméterében keresi az elsőt, ha megtalálható benne, akkor a sorszámát adja eredményül.
ELEME? MEMBER?	*akármi lista* *szó 1 szó2*	*logikai érték* *logikai érték*	Eldönti, hogy a második paraméterében megtalálható-e az első.
ELEMNÉLKÜLI BUTMEMBER	*karakter szó* *akármi lista*	*szó* *lista*	Az a sorozat, amelyik a második paraméter elemeit tartalmazza, elhagyva belőle az első paraméter összes előfordulását.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ELEMSZÁM, ESZ COUNT	*akármi*	*szám*	Az *akármi* elemeinek száma.
ELEMTŐL FROMMEMBER	*szó1 szó2* *akármi lista*	*szó* *lista*	A második paraméter azon elemeit tartalmazó sorozat, amelyik az első paraméterként megadott érték mögött található.
ELFED BURY	*memória_objektum* *memória_objektum_lista* **Hld:** *(paraméter1 paramter2 ...)* ahol *minden paraméter* vagy *memória_objektum* vagy *memória_objektum_lista*		Láthatatlanná teszi a paraméterként megadott, felhasználó által definiált *memória_objektum*okat, amelyek ezután fedettek lesznek.
ELFEDMIND BURYALL	*nincs*		Láthatatlanná teszi a felhasználó által definiált összes *memória_objektum*ot.
ELFELEJT IGNORE	*akármi* *(akármi1 akármi2 ...)*		A kapott paramétereket nem hajtja végre, elfelejti.
ELKAP CATCH	*szó Logo_utasításlista*		Balról jobbra haladva kiértékeli a paraméterként kapott Logo *utasításlistát*. Amikor ebben az ÁTDOB utasításhoz jut, mely a *szó*val együtt szerepel, akkor a vezérlés újra az ELKAP eljáráshoz kerül, és az ezt követő kifejezések kiértékelése történik meg.
ELŐRE, E FORWARD, FD	*szám*		Az aktuális teknőcöket *szám* lépéssel előre viszi.
ELSŐ FIRST	*akármi* (nem üres)	*akármi*	Eredménye a paraméter első eleme.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ELSŐNEK FPUT	*akármi lista* *képsor1 képsor2* *szó1 szó2*	*lista* *képsor* *szó*	Eredménye az a sorozat, amely úgy keletkezik, hogy az első paramétert a második paraméter elejére beszúrja.
ELSŐNÉLKÜLI, EN BUTFIRST, BF	*akármi* (nem üres)	*akármi*	Eredménye az eredeti paraméter első elemének elhagyásával kapott sorozat.
EREDMÉNY, ER OUTPUT, OP	*akármi*		A pillanatnyilag futó függvény értékét adja meg az *akármi*ben.
ÉRZÉKENYTERÜLET CODEAREA	speciális szó		Egy mező, névvel ellátott téglalap alakú terület a rajzlapon, ahol az egeret kezelni lehet. Az ÉRZÉKENYTERÜLET szó definiálja azokat a mezőket, melyek a fenti téglalapok jellemzőit és azok kódjait tartalmazzák.
ÉS AND	*logikai érték logikai érték* *(logikai érték ...)*	*logikai érték* *logikai érték*	IGAZ, ha minden bemenő adat logikai értéke IGAZ, minden más esetben HAMIS.

F

FÁJL? FILE?	*szó*	*logikai érték*	IGAZ, ha a fájl létezik; ha nem adunk alkönyvtárat, akkor a munkakönyvtárban keres; kiterjesztés nélkül LGP lesz.
FÁJLVÉGE EOF?	*nincs*	*logikai érték*	Eredménye HAMIS, ha a megfelelő bemenet a billentyűzetről érkezik. Ugyancsak HAMIS az eredmény, ha a bemenet egy fájlból érkezik, és ezt a fájlt még nem olvastuk végig. A kimenet IGAZ, ha éppen a fájl utolsó sorát/szavát/karakterét olvastuk be.
FÁZIS PHASE	*nincs*	*szám* ([0, 255])	Az első aktív teknőc fázisát adja (akkor is, ha nem látszik).

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
FÁZIS! SETPHASE	*szám* egész [-32767, 32767]		Az aktív teknőcök fázisát állítja be mod N-re, ha N fázisból áll a teknőc és az animációsmód igaz.
FÁZISPONT SHAPEPOS	*nincs*	*téglalap*	Megadja az első aktív teknőc által lefedett területet [x y szélesség magasság] formában. Ha nem látszik, akkor [x y 0 0].
FED! SETBURY	*logikai érték*		A lefedett objektumokat láthatóvá teszi, vagy elrejti.
FEDETT BURIED	*nincs*	*fedett_objektumok_listája*	Eredménye az összes fedett objektumot tartalmazó lista.
FELFED UNBURY	*memória_objektum* *memória_objektum_lista* **Hld:** *(paraméter1 paraméter2 ...)* ahol *minden paraméter* vagy *memória_objektum vagy memória_objektum_lista*		Megszünteti az objektumok fedett tulajdonságát.
FELSŐSZINT TOPLEVEL	speciális szó		Használata: ÁTDOB "FELSŐSZINT Kilép a programból.
FIGYELJ TELL	*teknőc_név* *teknőc_név_lista*		Az adott teknőcöket aktívvá teszi.
FORDÍT .REVERSE	*akármi*	*akármi*	Megfordítja paramétere elemei sorrendjét.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
FUTTAT RUNPROGRAM, RUNPROG	*szó* *szó szám*	*szám* *szám*	Elindítja a paraméterben megadott programot a második paraméterben megadott ablakban, eredmény a program hibakódja.

G, GY

GENERÁL GENERATE	*szám eljárásobjektum2 akármi* *eljárásobjektum 1* *eljárásobjektum2* *akármi*	*lista* *lista*	Egy rekurzív listát generál az *akármi* kezdőértékkel és az *eljárásobjektum2* eljárással addig, amíg a lista *szám* elemű lesz vagy a következő elemre *eljárásobjektum 1* HAMIS lesz.
GOMBNÉV BUTTONS	*nincs* *(szám)*	*lista* *lista*	Vagy a Gombok ablak gombjaihoz kapcsolódó aktuális Gombsor-definíciót tartalmazó lista, vagy a *szám* gombhoz tartozó aktuális gombdefiníció.
GOMBNÉV! SETBUTTONS,	*gombsor_definíció_lista (szám gombsor_definíció)*	*beállított gombok*	Gombokhoz kapcsolódó feliratot/grafikát és viselkedést határoz meg.
GOMBNYOMÁS?, GNY? KEY? RC?	*nincs*	*logikai érték*	Eredménye IGAZ, ha lenyomtuk valamelyik billentyűt vagy a bal egérgombot.
GOMBSOR BUTTONSSIZE, BUTTONSIZE	*nincs* *nincs*	*[szám 1 szám2 szám3 szám4]* *[szám 1 szám2 szám3 szám4 szöveg]*	A Gombok ablak aktuális pozíciója és mérete, kiegészítve esetleg ennek elnevezésével.
GOMBSOR! SETBUTTONSSIZE, SETBUTTONSIZE	*[X Y w h]* *[X Y w h szöveg]* *[X Y w h [ ]]* *[ ].6*	*lista 4 vagy 5 elemű (Gombok ablak akt. poz.)*	A Gombok ablak aktuális pozícióját és méretét, esetleg elnevezését állítja be.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
GYÖK SQRT	*szám* ahol *szám*≥0	*szám*	Nem negatív *szám* négyzetgyöke.

H

HA IF	*logikai érték* *Logo_utasítás_lista* *logikai érték* *Logo_utasítás_lista 1* *Logo_utasítás_lista2* *logikai érték [kif 1]* *[kif2]*	*Logo utasításlista* *akármi*	Ha az első paramétere IGAZ, akkor az első utasításlistát kell végrehajtani, ha HAMIS, akkor a másodikat (ha van).
HAHAMIS, HAH IFFALSE, IFF	*Logo_utasítás_lista*	*akármi* *utasítás_lista*	Végrehajtja az utasításlistát, ha az aktuális eljárás utolsó TESZT utasítása HAMIS eredményű volt.
HAIGAZ, HAI IFTRUE, IFT	*Logo_utasítás_lista*	*akármi* *utasítás_lista*	Végrehajtja az utasításlistát, ha az aktuális eljárás utolsó TESZT utasítása IGAZ eredményű volt.
HAMIS FALSE	speciális szó		HAMIS logikai érték.
HANG SOUND	*szám_lista* *(szám_lista 1* *szám_lista2 ...)*	*hang, dallam*	A *szám_lista* által megadott hangsorozatot játssza le.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
HANGHULLÁM, HANGH PLAYWAVE, PLAYW	*WAV_fájl* *(WAV_fájl1 WAV_fájl2 ...)* *[WAV_fájl1 WAV_fájl2 ...]* *(WAV_fájl1 WAV_fájl2 ... [ ])* *(WAV_fájl1 WAV_fájl2 ... ")*	*hang, dallam*	A *WAV_fájl*ban megadott dallamot játssza le. (Hangkártya szükséges!)
HANGSOR PLAY	*lista* *(lista1 lista2 ...)*	*zene*	A számítógép beépített hangszóróján lejátssza a *listá*ban megadott hangjegyeket.
HASZNÁL APPLY	*eljárásobjektum lista*	*akármi*	Az *eljárás*t meghívja a listában szereplő paraméterekkel.
HATVÁNY EXP	*szám*	*szám*	Exponenciális függvény.
HAZA HOME	*nincs*	*művelet*	Az összes aktív teknőcöt az alapkoordinátáira viszi. Az alapkoordinátákat az ÚJTEKNŐC eljárással definiáljuk.
HÁNYADIK REPCOUNT, REPC	*nincs*	*szám*	Eddig hányszor hajtottuk végre az ismétlést. (Csak ISMÉTLÉS, CIKLUSAMÍG, CIKLUSCSAK utasítások belsejében használható.)
HÁNYADOS QUOTIENT	*szám1 szám2* **Hld:** *pont szám* **Hld:** *szög pont*	*szám* *pont* *pont*	A *szám1* és *szám2* hányadosa.
HÁTRA, H BACK, BK	*szám*		Az összes aktív teknőcöt a paraméterként adott *szám*ú lépéssel (azaz képponttal), az irányukkal ellentétes irányban mozgatja.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
HELY POS	*nincs*	*pont*	Az első aktív teknőc aktuális [X Y] koordinátái. (A koordináta-rendszer origója a rajzlap középpontjában van.)
HELY! SETPOS	*pont*		Az összes aktív teknőcöt az adott *pont*ba viszi, de aktuális irányukat változatlanul hagyja. TOLLATLE állapotban az eredeti és új pozíciót vonallal köti össze.
HIBA ERROR	*nincs* mint speciális szó: *ELKAP "HIBA lista*	*lista*	Függvényként teljes hibaüzenetet eredményez, ha a számítás alatt hiba lép fel. Ha nem következik be hiba, akkor az eredmény [ ]. Bármilyen, a *lista* kiértékelése alatt bekövetkező hibát itt kezel le a rendszer – nem küld hibaüzenet, és az ELKAP utasítást követő eljárásokkal adhatjuk meg a saját reakciónkat a bekövetkezett sajátos hibára.
HIBAÜZENET MAKEERROR	*lista akármi*		Az eljárás alkalmazása esetén leáll a program végrehajtása, és saját hibaüzenetet küldhetünk. A hiba üzenetét a *listá*ban (a HIBAÜZENET eljárás első paramétereként) adhatjuk meg.

I, Í

IDŐ TIME	*nincs*	*[óra perc másodperc század_másodperc]*	Eredménye egy négy számból álló lista: az aktuális idő.
IDŐSZÓ TIMEWORD, TIMEW	*nincs* **Hld:** *(idő_formátum)* **Hld:** *(idő_formátum idő)*	*szó* *szó* *szó*	Megadja az aktuális időt a Windows vezérlőpultja International opciójának idő-formátum beállításának megfelelően. Például 13:48:50, 1:48:50 DU vagy 1:48p D.U. A függvény paramétereként adhatunk egy lehetséges *idő_formátum*ot (egy listát), ami meghatározza azt a formát, melynek megfelelően módosítva kapjuk eredményül a *szó*t. Két paraméter esetén a szóban az *idő*t az *idő_formátum*nak megfelelően kapjuk vissza.
IGAZ TRUE	— speciális szó —		Az IGAZ logikai érték.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
IRÁNY HEADING	*nincs*	*szög*	Eredménye az első aktív teknőc aktuális iránya. (Az északi iránytól az óramutató járásával megegyező irányban mért szög.)
IRÁNY! SETHEADING, SETH	*szög*		Elfordítja az aktív teknőcöket úgy, hogy új irányuk megegyezzen az adott *szög*gel (az aktuális irányuktól függetlenül).
IRÁNYSZÖG TOWARDS	*pont* *teknőc*	*szög*	Eredménye az az irány, melybe az első aktív teknőcnek kellene fordulnia ahhoz, hogy az adott *pont* felé nézzen. Ha a paraméter egy *teknőc*, akkor a függvény eredménye az az irány, melybe az első aktív teknőcnek fordulnia kellene, hogy az adott teknőc felé nézzen. (Jegyezzük meg, hogy az IRÁNYSZÖG függvény eredménye nem az a szög, amellyel a teknőcnek balra vagy jobbra kellene fordulnia, hogy az adott pontra nézzen.)
ISMÉTLÉS, ISM, REPEAT	*szám* *Logo_utasítás_lista*		*szám*szor egymás után végrehajtja a *Logo_utasítás_listá*t. (Ha *szám*<1, akkor egyszer sem hajtja végre.)
ÍRÓABLAK, ÍA TEXTSCREEN, TS	*nincs*		A teljes Fő ablakot szövegesre állítja be. A rajzlap tartalma nem vész el, csak egy részét ideiglenesen lefedi az íróablak, és aktív is marad: a teknőcök továbbra is rajzolhatnak rá, mozoghatnak rajta..
ÍRÓLAPABLAK GETTS	*nincs*	*szám* ([2,25] egész)	OSZTOTTABLAK módban az íróablakban pillanatnyilag megjelenő sorok száma. Ez a *szám* mindig egy 2 és 25 közé eső egész.
ÍRÓLAPABLAK! SETTS	*szám* (egész [2, 25])		OSZTOTTABLAK módban az írólapon megjelenő sorok számát állítja be. Ennek a számnak a megváltoztatásával megváltoztatjuk a rajzlap képernyőn megjelenő részét (vagyis a rajzablak méretét), de sem a rajzlap aktuális méretét (melyet a RAJZLAPABLAK! utasítással állíthatunk be), sem ennek aktuális tartalmát nem módosítja.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ÍRÓSZÍN! SETTC	*szám* (egész [0, 255])		Az írólap hátterének és magának a szövegnek a színét állítja be. Ha a bemenő számot a következő formában írjuk le, 16*X + Y (ahol az X és Y is 0 és 15 közé eső számok), akkor az X szám jelöli az írólap hátterének színét, az Y pedig a szöveg színét. Az írólap alapbeállítása 240, azaz 16 * 15 + 0.

J

JEL CHAR	*szám* (egész [0..255])	*karakter*	A *szám* ASCII kódú *karakter*.
JOBBRA, J RIGHT, RT	*szög*		Az összes aktív teknőcöt elforgatja jobbra *szög*gel.

K

KEREKÍT ROUND	*szám* **Hld:** *pont*	*szám* *pont*	Kerekítés. Mind a két koordinátát kerekíti.
KÉPSOR IMAGE	*képsor1 képsor2 (képsor1 ...)* **Hld:** *képsor képsor_operátor* **Hld:** *(képsor képsor_operátor_kifejezés...)*	*képsor* *képsor* *képsor* vagy semmi *képsor* vagy semmi	Összefűzött képsor.
KÉPSOR? IMAGE?	*akármi*	*logikai érték*	IGAZ, ha a paramétere képsor.
KÉPSORMÉRET IMAGESIZE	*képsor*	*[szám1 szám2]*	Az első fázis szélessége, illetve magassága képpontokban.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
KÉREM ASK	*teknőc* *Logo_utasítás_lista* *teknőc_lista* *Logo_utasítás_lista* *teknőc [kifejezés]* *teknőc_lista [kifejezés]*	*akármi* *akármi*	A *teknőc*, illetve *teknőcök*, ideiglenesen aktivizálva, párhuzamosan hajtják végre az *utasítás_listá*t. A *teknőc*, illetve *teknőcök* ideiglenesen aktívvá válnak, és a rendszer kiértékeli a *kifejezés*t, amely az első teknőccel kapcsolatos eredményt adja.
KIFIGYEL WHO	*nincs*	*teknőc_név* *teknőc-név_lista*	Az összes aktív teknőcöt tartalmazó lista.
KIHASÍTKÉP GETIMAGE	*[szám1 szám2 szám3 szám4]*	*képsor*	Egyetlen fázisból álló képsor: az első aktív teknőc készíti a rajzlapon őt körülvevő pillanatnyi környezetről.
KIÍR PRINT, PR	*akármi* *(akármi1 akármi2 ...)*	*szöveg*	?KIÍR "SZÉP SZÉP
KIÍRBELSŐ TYPE	*akármi* *(akármi1 akármi2 ...)*	*szöveg?*	Az utolsónak kiírt karakter mögött megjelenik a parancsjel. Például: ?KIÍR "SZÉP SZÉP?
KIÍRJEL, KI SHOW, SH	*akármi* *(akármi1 akármi2 ...)*		Az esetleges külső zárójelet is kiírja.
KIÍRÓESZKÖZ PRINTTO	*fájl_név* *[ ]*		Az aktuális kimenet a *fájl_név* nevű állomány lesz.
KIÍRSAJÁTSZAVAK POTS	*nincs*		Az írólapra írja a saját eljárások címsorát.
KIÍRSOR, KS PRINTLINE, PL	*akármi* *(akármi1 akármi2 ...)*	*akármi1* *akármi2*	Minden egyes paraméterét külön sorban írja ki.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
KIÍRTARTALOM, KT PO	*memória_objektum* *memória_objektum_lista* **Hld:** *(paraméter 1 paraméter2 ...)* ahol *paraméter memória_objektum* vagy *memória_objektum_lista*		Az írólapra írja a paraméterként adott objektumok definícióit.
KISEBB? LESS?	*szám 1 szám2* *(szám 1 ...)* **Hld:** *szó 1 szó2* **Hld:** *(szó 1 ...)* **Hld:** *pont 1 pont2* **Hld:** *(pont 1 ...)*	*logikai érték* *logikai érték* *logikai érték* *logikai érték* *logikai érték* *logikai érték*	IGAZ, ha *szám 1<szám2*, különben HAMIS.
KIVÁLASZT PICK	*akármi, de nem üres*	*szó->karakter* *lista->listaelem* *képsor->fázis*	Paraméterének egy véletlenszerűen kiválasztott eleme.
KÍVÜL? OUTSIDE?	*téglalap*	*logikai érték*	IGAZ, ha az első aktív teknőc az adott *téglalap*on kívül helyezkedik el.
KÓD ASCII	*karakter* *szó*	*szám* *szám* [0, 255] közötti egész	A bemenő *karakter* vagy a bemenő *szó* első karaktere ASCII kódja.
KÖRKÖRÖSEN WRAP	*nincs* *(pont)* *(téglalap)* *([ ])*	*nincs*	A képernyő szélének kezelési módja az aktív teknőcök számára.
KURZORHELY CURSOR	*nincs*	*[oszlop sor]*	A kurzor jelenlegi helyzete az írólapon.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
KURZORHELY! SETCURSOR	*[szám 1 szám2]* ahol *szám 1* egész [1, 80], *szám2* egész [1,26]		Kiteszi az írólap aktuális parancsjelét a *szám 1 szám2* pozícióba és mellette megjeleníti a kurzort is.
KÜLÖNBSÉG DIFFERENCE	*szám 1 szám2* **Hld:** *pont 1 pont2*	*szám* *pont*	Eredménye *szám 1* és *szám2* különbsége.

L, LY

LÁTHATATLAN HIDETURTLE, HT	*nincs*		Eltünteti az összes aktív teknőcöt a képernyőről.
LÁTHATÓ SHOWTEKNŐC, ST	*nincs*		Az összes aktív teknőc alakja megjelenik a képernyőn.
LÁTHATÓ? SHOWN?	*nincs*	*logikai érték*	IGAZ, ha az első aktív teknőc aktuálisan látható a képernyőn.
LEKÉPEZ MAP	*eljárás objektum* *lista 1*		Az összes lista első elemét egyetlen listába gyűjti, és az *eljárás_objektum* ezt a listát alkalmazza. A második elemekkel is ugyanezt csinálja és így tovább.
LENYOMAT STAMP	*nincs*		Az összes aktív teknőc kinyomtatja az alakját.
LISTA LIST	*akármi1 akármi2* *(akármi 1 ...)*	*lista* *lista*	A paramétereiből álló sorozatot (listát) adja eredményül.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
LISTA? LIST?	*akármi*	*logikai érték*	IGAZ, ha a paramétere lista.
LN	*szám1* (pozitív)	*szám2*	ln(*szám1*)=*szám2*
LOGO	*kulcs_szó* ahol *kulcs_szó* a következők valamelyike: ERROR, VERSION, HELP, AUTOPATH, RIGHTCLICKS, STEPPING, DECIMAL, PRINT, SAVE, BEEP vagy SOUND **Hld:** további opciók *for key_szó:* :PICKER, HIGHLIGHT, INDENT, HIDEPROBLEMS, CASE, OCTAVES, AXCOLORBITS, KEEPPALETTE, PALETTE, OMPRESSBMP, AVE16, MAXIMAGECOLOURBITS	*szó vagy lista*	Eredményül az adott szóhoz tartozó aktuális beállítását kapjuk meg.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
LOGO! SETLOGO	*[key_szó 1 akármi 1 key_szó2 akármi2 ...]* ahol *key_szó* a következők valamelyike: ERROR, VERSION, HELP, AUTOPATH, RIGHTCLICKS, STEPPING, DECIMAL, PRINT, SAVE, BEEP vagy SOUND **Hld:** további opciók a *kulcs_szó*hoz: :PICKER, HIGHLIGHT, INDENT, HIDEPROBLEMS, CASE, OCTAVES, AXCOLORBITS, KEEPPALETTE, PALETTE, OMPRESSBMP, AVE16, MAXIMAGECOLOURBITS	Magyar megfelelői: HIBA, VERZIÓ, SEGÍTSÉG, ALAPÚT, LÉPÉS, DECIMÁLIS, NYOMTAT, MENT, BEEP, SOUND	Néhány jellemző megváltoztatásával a Logo környezetét a felhasználó számára előnyösebbé alakíthatjuk át. VERZIÓ=1 esetén csak az angol utasításokat használhatom, VERZIÓ=2 esetén a magyar utasításokat is használhatom.
LOGOABLAK LOGOSIZE	*nincs*	*[X Y w h szám szöveg]* ahol *X, Y, w* és *h számok*	Kiírja a Logo főablak jellemzőit.
LOGOABLAK! SETLOGOSIZE	*[X Y w h]* *[X Y w h szám]* *[X Y w h szám szöveg]* *[ ]* ahol *X, Y, w* és *h szám*s		A Logo Fő ablakának pozícióját, méretét és feliratát definiálja.
LOKÁLIS LOCAL	*szó* *szó_lista* *(szó 1 ...)*		Az adott eljáráshoz tartozó lokális változót hoz létre *szó* vagy *szó_lista* nevekkel.
LOKÁLISNÉV LET			Értéket is ad a lokális változónak, a névadással egyben.

M

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
MARADÉK, MA MOD, REMAINDER	*szám 1 szám2* (szám2 nem 0) **Hld**: *pont szám* **Hld:** *pont 1 pont2* ahol *szám* nem 0 és *pont2 = [X Y]* nem tartalmaz 0-t	*szám* (nem negatív egész) *pont* *pont*	A *szám 1* és *szám2* osztási maradéka.
MCI	*mci_lista*		
MENT SAVE	*LGO_fájl* *LGO_fájl paraméter 1 ...* *(LGO_fájl paraméter 1 ...)* **Hld:** *"Vágólap* **Hld:** *"Vágólap paraméter 1 ...* **Hld:** *("Vágólap paraméter 1 ...)* ahol *paraméter 1, ...* vagy a *memória_objektum* vagy *memória_objektum_lista*		Az összes paraméterként szereplő objektumot az adott fájlba menti.
MENTKÉPSOR SAVEIMAGE	*LGW_fájl képsor* *LGW_fájl képsor 1 képsor2 ...* *(LGW_fájl képsor 1 ...)* **Hld:** *"Vágólap képsor* **Hld:** *"Vágólap képsor 1 képsor2 ...* **Hld:** *("Vágólap képsor 1 ...)*	*fájl* *Vágólap*	A paraméterként szereplő képsor(oka)t menti az adott fájlba. A képsort a Vágólapra helyezi.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
MENTKÖTET SAVELIBRARY, SAVELIB	*library* *library paramétert 1 ...* *(library paraméter 1 ...)* **Hld:** *"Vágólap* **Hld:** *"Vágólap paraméter 1 ...* **Hld:** *("Vágólap paraméter 1 ...)* ahol *paraméter 1, ...* vagy *memória_objektum* vagy *memória_objektum_lista*	*projektkötet* *Vágólap*	Elmenti a pillanatnyilag létező teknőcöket és beállításukat, valamint az éppen nem fedett memóriaobjektumokat. Lehetőség van arra is, hogy konkrétan felsoroljuk az elmentendő memóriaobjektumokat *(paraméter 1 ...)* Vágólapra menti az adott kötet tartalmát.
MENTRAJZ SAVEBITMAP	*BMP_fájl képsor* **Hld:** *"Vágólap képsor*	*fájl* *Vágólap*	Adott képsor első fázisát egyszerű rajzként fájlba menti. A képsor első fázisát a Vágólapra helyezi.
MENTRAJZLAP SAVESCREEN	*BMP_fájl* *(BMP_fájl pont)* *(BMP_fájl rctszög)* **Hld:** *projekt_rajzlap* **Hld:** *(projekt_rajzlap pont)* **Hld:** *(projekt_rajzlap téglalap)* **Hld:** *"Vágólap* **Hld:** *("Vágólap pont)* **Hld:** *("Vágólap téglalap)*	*fájl* *fájl* *Vágólap*	Elmenti az aktuális rajzlapot, illetve annak egy téglalapnyi területét. Elmenti a rajzlap vagy egy részének tartalmát legutóbbi képernyőként a project fájlba. Vágólapra menti a rajzlap vagy egy részének tartalmát.
MEZŐ GPROP	*szó 1 szó2*	*akármi*	Megkeresi azon mezőket, amelyeket *szó 1*-re definiáltunk és eredményül a *szó2* által meghatározott mező értékét adja.
MEZŐ! PPROP	*szó 1 szó2 akármi*	*akármi*	*szó 1*-hez kapcsol egy *szó2* nevű mezőt, melynek értéke *akármi* lesz.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
MEZŐK PLIST	*szó*	*mező_lista*	*szó*val kapcsolatban definiált mezők listája.
MEZŐK! SETPLIST	*szó mező_lista*	*új mezők*	*szó*hoz definiálja a *mező_lista*ban megadott mezőket.
MIND ALLTURTLES, ALL	*nincs*	*teknőc_lista*	A jelenleg létező teknőcök nevének listája.
MONDAT, M SENTENCE, SE	*akármi1 akármi2* *(akármi ...)*	*lista* *lista*	A paraméterként adott szavakból, illetve listák elemeiből egyetlen listát hoz létre.
MUTATGOMBSOR, MG SHOWBUTTONS	*nincs*	*ablak*	Megjeleníti a Gombok ablakot a képernyőn.
MUTATIKONSOR, MI SHOWSPEEDBAR	*nincs*	*ikonsor*	Megjeleníti az ikonsort a képernyőn.

N, NY

NAGYOBB? GREATER?	*szám 1 szám2* *(szám 1 ...)* **Hld:** *szó 1 szó2* **Hld:** *(szó 1 ...)* **Hld:** *pont 1 pont2* **Hld:** *(pont 1 ...)*	*logikai érték* *logikai érték* *logikai érték* *logikai érték* *logikai érték* *logikai érték*	IGAZ, ha *szám 1*>*szám2*, különben HAMIS. IGAZ, ha *szám 1* után csökkenő sorozat következik. IGAZ, ha *szó 1*>*szó2* ábécé-sorrendben IGAZ, ha a *pont 1* megfelelő koordinátái > *pont2* megfelelő koordinátái.
NEM NOT	*logikai érték*	*logikai érték*	Logikai tagadás művelet.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
NEVEK NAMES	*nincs*	*memória_elem_lista*	A pillanatnyilag létező változók listája.
NEVEZ NAME	*akármi szó*	*változó*	*szó* nevű változót definiál *akármi* értékkel.
NÉV MAKE	*szó akármi*	*változó*	*szó* nevű változót definiál *akármi* értékkel.
NÉV? NAME?	*szó*	*logikai érték*	IGAZ, ha *szó* globális változó, aktív eljárás lokális változója vagy bemenő adata.
NÖVEL INC	*változó_név* *(változó_név szám)* csak numerikus változó lehet	*numerikus*	1-gyel növel. *szám*mal növel.
NYITVIDEO OPENVIDEO	*szó1 szó2 lista3* *lista1 szó2 lista3*, ahol *szó2* nem üres		AVI fájlokat játszik le.
NYOMOZ DEBUG	*Logo_utasításlista*		A Logót hibafelderítő módra állítja. Lehetőséget biztosít nyomon követni és láthatóvá tenni az utasításlista végrehajtását, illetve annak részeredményeit (lépésenként vagy nagyobb egységekben haladva).
NYOMOZÓSZÜNET PAUSE	*nincs*		Hibafelderítő módban leállítja a program végrehajtását, ami a későbbiekben folytatható.
NYOMTATÍRÓLAP PRINTTEXT, PRINTT	*logikai érték* *[ ]*		Az Írólap tartalmát nyomtatja.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
NYOMTATMEMÓRIA, NYM PRINTMEMORY, PRINTM	*paraméter 1* *paraméter 1 memória_objektum 1 memória_objektum2 ...* *paraméter 1 memória_objektum_lista* ahol *paraméter 1* is vagy *logikai érték* vagy *[ ]*		A Memória ablak tartalmát nyomtatja. Megadott objektumokat, mindent, rejtett objektumokat (*paraméter 1*="HAMIS).
NYOMTATRAJZLAP PRINTGRAPHICS, PRINTG	*logikai érték* *[ ]*		Rajzlap aktuális tartalmát nyomtatja.

O, Ö, Ő

OLVASJEL, OJ READCHAR, RC	*nincs*	*karakter*	Egy karaktert olvas.
OLVASKÓD, OK READKEY	*nincs*	*szám* (ASCII kód)	Egy karakter kódját olvassa.
OLVASLISTA, OL READLISTA, RL	*nincs*	*lista*	ENTER lenyomásáig a karaktersorozatot listába fűzi.
OLVASLISTAPARANCSJEL!, OLPARANCSJEL! SETREADLISTAPROMPT, SETRLPROMPT	*szöveg*		A listaolvasás parancsjelét változtatja meg az írólapon.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
OLVASÓESZKÖZ READFROM	*fájl* *[ ]*		Az alapértelmezett billentyűzet helyett a *fájl*ból veszi a bemenő adatokat. Ismét a billentyűzetre áll át.
OLVASSZÓ, OSZ READWORD, RW	*nincs*	*szó*	Egy szót olvas.
OLVASÜZENŐ READLISTDIALOG, RLD	*nincs* *(szöveg 1)* *(szöveg 1 szöveg2)* *(szöveg 1 szöveg2 lista3 szöveg4)* ahol *szöveg 1, szöveg2* és *szöveg4 szám* és *lista3 0, 2, 4, 5* vagy *8 számlista*	*lista* *lista* *lista* *lista* *lista*	Párbeszédablakot jelenít meg *szöveg1* kérdéssel, *szöveg2* címsorral, *szöveg4* kezdőértékkel. A *lista3* elemei az ablak pozícióját, méretét, kerettípusát és színeit határozhatják meg.
OSZTOTTABLAK, OA SPLITSCREEN, SS	*nincs*		A Fő ablakot rajz- és írólapra osztja.
ÖSSZEG SUM	*szám 1 szám2* *(szám 1 ...)* **Hld:** *pont 1 pont2* **Hld:** *(pont 1 ...)*		Kettő vagy több szám összege. Kettő vagy több vektor összege.
ÖSSZEKEVER SHUFFLE	*szó/lista/képsor*		A paramétereként kapott sorozat elemeit összekeveri.

P

PARANCSJEL! SETPROMPT	*szó* ahol *szó* nem több, mint 16 karakter		Az írólap parancsjelét változtatja meg.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
PONT DOT	*nincs*		Lerakott tollú teknőc kitesz egy pontot az aktuális pozíciójában, tolla aktuális színének és vonalvastagságának megfelelően. Megegyezik az ELÓRE 0 paranccsal.
PONTSZÍN DOTCOLOR	*nincs*	*szín* *[0 15]*	A rajzlapon az első aktív teknőc aktuális pozíciójában levő pont *színe*.

R

RAJZABLAK, RA GRAPHICSCREEN, GS	*nincs*		A grafikus képernyő teljesen kitölti a Fő ablakot.
RAJZLAPABLAK GETGS	*nincs*	*téglalap* *[szám1 szám2 egész3 egész4]*	Rajzlap aktuális beállításait írja le. *szám1* a szélesség, *szám2* a magasság, *egész3* az igazítás, *egész4* a gördítősáv jelenléte.
RAJZLAPABLAK! SETGS	*[szám1 szám2 szám3 szám4]* ahol *szám1, szám2* egész [8, 2048], és *szám3, szám4* vagy 0 vagy 1		Rajzlapablak beállítása. *szám1* a szélesség, *szám2* a magasság, *egész3* az igazítás, *egész4* a gördítősáv jelenléte.
RAJZLAPSZÍN, RSZ BACKGROUND, BG	*nincs*	*szín*	A rajzlap hátterének aktuális *színe*.
RAJZLAPSZÍN!, RSZ! SETBACKGROUND, SETBG	*szín*		Törli a rajzlapot és beállítja az új háttérszínt.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
RÁKATTINTOTT TOUCHED	*nincs*	*teknőc_név* *[ ]*	Melyik teknőcöt érintettük meg utoljára az egér bal gombjának kattintásakor.
REJGOMBSOR, RG HIDEBUTTONS	*nincs*		Eltünteti a Gombsor ablakot a képernyőről.
REJTIKONSOR, RI HIDESPEEDBAR	*nincs*		Eltünteti a Fő ablak ikonsorát a képernyőről.
REKORDNEVEK PROPS	*nincs*	*szavak listája mező definíciókkal*	A mezőket tartalmazó szavak listája.
REKORDOK PPS	*nincs*		Kiírja az összes szót, melyhez eddig mezőket definiáltunk.
RENDSZERMÉRET SYSTEMMETRICS	*szám* *szó*	*szám* *szám*	A képernyő arculatát kezelhetjük.

S, SZ

SAJÁTNEVEK PROCS	*nincs*	Logo eljárások nevei	Az összes saját eljárást tartalmazó listát kapjuk.
SAJÁTSZÓ DEFINE	*szó definíció_lista*		Egy olyan eljárást definiál, melynek *szó* a neve, és a *definíció_lista* a törzse.
SAJÁTSZÓ? DEFINED?	*szó*	*logikai érték*	IGAZ, ha a *szó* tetszőleges általunk definiált eljárás neve, egyébként HAMIS.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
SIN SIN	*szám 1*	*szám2*	*szám2*=sin*(szám 1)*
SOKSZÖG POLYGON	*polygon_lista*	*sokszög*	Az összes aktív teknőc egy a bemenő sokszöglistának megfelelő kitöltött sokszöget rajzol.
SOKSZÖGVONAL POLYGONLINE	*polygon_lista*		Az összes aktív teknőc egy a bemenő sokszöglistának megfelelő NEM kitöltött sokszöget rajzol.
SZÁM? NUMBER?	*akármi*	*logikai érték*	IGAZ, ha a paramétere számjegyek nem üres sorozata.
SZIA BYE	*nincs* *logikai érték*		Kilép a Logóból. Ha az utasításnak nincs bemenő adata, vagy ha a paraméter IGAZ, akkor egy párbeszédablak tűnik fel, amely a munkánk elmentésére hívja fel figyelmünket. Ha HAMIS bemenő adattal hívjuk meg, akkor kérdés nélkül elhagyjuk a Logót.
SZORZAT PRODUCT	*szám 1 szám2* *(szám 1 ...)* **Hld:** *paraméter 1 paraméter2* **Hld:** *(paraméter 1 ...)* ahol minden *paraméter* vagy *szám* vagy *pont*	*szám* *szám* *szám* vagy *pont* *pont,* ha legalább egy bemenet *pont,* különben *szám*	Két vagy több szám szorzata. Pont és pont szorzata a nekik megfelelő vektorok skaláris szorzata.
SZÓ WORD	*szó 1 szó2* *(szó ...)*	*szó* *szó*	Paramétereiként kapott szavakból egyetlen szót állít össze.
SZÓ? WORD?	*akármi*	*logikai érték*	IGAZ, ha a paramétere szó, különben HAMIS.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
SZÓTARTALOM TEXT	*szó* ahol *szó* név sajáteljárásnév	*definíció_lista*	Eredménye a *szó* nevű saját eljáráshoz tartozó *definíció_lista*.
SZÖVEGDOBOZ GETBOX	*[ ]* *szó*	*lista* *szó*	Az aktív szövegdobozteknőc adata(i).
SZÖVEGDOBOZ! SETBOX	*[kulcsszó 1 érték 1 kulcsszó2 érték2 ...]* (a szövegdoboz tulajdonságai)		Felveszi a beállított értékeket az aktív szövegdobozteknőc.
SZÖVEGDOBOZMÓD! SETBOXMODE	*logikai*		Az aktív teknőcök szövegdobozzá válnak.
SZŰRŐ FILTER	*eljárásobjektum_ lista*	*lista*	Kiválogatás eljárásobjektum alapján.

T, TY

TAKAR? OVERLAP?	*teknőc_név* *teknőc_név_lista*	*logikai érték* *logikai érték*	Takarja-e a megadott teknőc az aktuálisat.
TAKART OVERLAPPED	*nincs* *pont*	*teknőc_név* *teknőc_lista*	Azon teknőcök, amelyeket az első aktív teknőc takar. A ponton lévő teknőcök.
TANULD TO	*szó* *szó paraméter1 paraméter2 ...* **Hld:** *szó paraméter1 [szám]*		Eljárás létrehozása.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
TARTALOM THING	*szó* ahol *szó* változónév	*akármi*	A változónévhez kapcsolt értéket adja.
TARTALOMJEGYZÉK, TART DIR	*( )* *szűrő* *(szűrő 1 szűrő2 ...)*	*lista* *lista* *lista*	Ha egyetlen paramétere sincs, akkor az eredménye a munkakönyvtárban található összes fájlt tartalmazó lista egyébként csak azokat a fájlokat listázza ki, amelyek megfelelnek a paraméterek által adott követelménynek *(szűrő 1, szűrő2 ...).*
TEKNŐCOLVASLISTA, TOL READLISTTURTLE, RLT	*nincs* *(paraméter 1)* *(paraméter 1 szám)*	*lista* *lista* *lista*	Az első aktív teknőc pozíciójában szerkesztősor jelenik meg. Meg kell adni valamilyen szöveget (zárása ENTER), és ez adja a TOL függvény eredményét.
TERÜLET GETWINDOW	*nincs*	*téglalap* *[ ]*	Teknőc által felhasználható terület.
TESZT TEST	*logikai érték*		A paraméter logikai értékét megjegyzi HAIGAZ, HAHAMIS utasítások által való későbbi felhasználáshoz.
TG TAN	*szám 1* (szög)	*szám2*	tg(*szám 1*)=*szám2*
TIZEDESJEGY! SETFORMAT	*szám* egész [0, 18]		Kijelzendő tizedesjegyek számát állítja be.
TOLL PEN	*nincs*	*szó* (TF, TL, TRD vagy TVT valamelyike)	Az első aktív teknőc tollának állapota.
TOLL! SETPEN	*szó* vagy TF, TL, TRD vagy TVT		Az aktív teknőc tolla állapotának beállítása.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
TOLLATFEL, TF PENUP, PU	*nincs*		=TOLL! TF
TOLLATLE, TL PENDOWN, PD	*nincs*		=TOLL! TL
TOLLÁLLAPOT PENSTATE	*nincs*	*[szó szín tollvastagság tollminta]* ahol *szó* TF, TL, TRD vagy TVT valamelyike	Aktív teknőc tollának minősége (szín, vastagság, minta).
TOLLÁLLAPOT! SETPENSTATE	*[szó szín tollvastagság tollminta]* ahol *szó:* TF, TL, TRD vagy TVT		Az aktív teknőcök tollának összes jellemzőjét (mód, szín, vonalvastagság, minta) állítja be.
TOLLMINTA, TM PATTERN	*nincs*	*toll_minta*	Az első aktív teknőc aktuális tollmintájának számkódja.
TOLLMINTA!, TM! SETPATTERN	*toll_minta*		Az összes aktív teknőc tollmintáját állítja be.
TOLLPONT HOTSPOT	*képsor*	*pont_lista*	Eredménye a tollpontok listája, ahol *pont 1* a képsor első fázisának tollpontja, *pont2* a második fázis tollpontja stb.
TOLLPONT! SETHOTSPOT	*képsor tollpont* *képsor [paraméter 1 ...]* ahol minden listaelem *[paraméter 1 ...]* vagy *pont* vagy *szám*	*képsor* *képsor* *képsor*	Egy olyan képsor, amelynek fázisszáma és tartalma megegyezik a bemenő képsoréval. Ha a második paraméter tollpont, akkor csak a képsor első fázisának tollpontja változik meg.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
TOLLRADÍR, TRD PENERASE, PE	*nincs*		Az összes aktív teknőc tollának állapotát megváltoztatja úgy, hogy ha ezek mozognak, akkor letörlik, amin áthaladnak.
TOLLSZÍN, TSZ PENCOLOR, PC	*nincs*	*szín*	Az első aktív teknőc tollának aktuális színe.
TOLLSZÍN!, TSZ! SETPENCOLOR, SETPC	*szín*		Az összes aktív teknőc tollszínét állítja be.
TOLLVASTAGSÁG, TV PENWIDTH, PW	*nincs*	*tollvastagság*	Az első aktív teknőc pillanatnyi tollvastagságának megfelelő szám.
TOLLVASTAGSÁG!, TV! SETPENWIDTH,SETPW	*tollvastagság*		Az összes aktív teknőc tollvastagságát állítja be.
TOLLVÁLTÓ, TVT PENREVERSE, PX	*nincs*		Az összes aktív teknőc tollának működését megváltoztatja úgy, hogy azon pontok színét, amelyen áthaladnak, invertálja.
TÖLT FILL	*nincs*		Az összes aktív teknőc kitölti az őt körülvevő környezetet az aktuális töltőszínével és töltőmintájával.
TÖLTŐMINTA. TLM FILLPATTERN, FP	*nincs*	*kitöltő_minta*	Eredménye az a töltőminta, amit előzőleg az első aktív teknőcnek beállítottunk.
TÖLTŐMINTA! SETFILLPATTERN, SETFP	*kitöltő_minta*		Az aktív teknőc aktuális kitöltőmintáját állítja be.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
TÖLTŐMÓD! SETFILLMODE	0 vagy 1		Az aktív teknőc kitöltési stratégiáját határozza meg.
TÖLTŐSZÍN, TLSZ FILLCOLOR, FC	*nincs*	*szín*	Az első aktív teknőc töltőszíne.
TÖLTŐSZÍN!, TLSZ! SETFILLCOLOR, SETFC	*szín*		Az összes aktív teknőc kitöltőszínét állítja be.
TÖRÖL ERASE, ER	*paraméter1 (paraméter1 paraméter2 ...)* minden *paraméter* vagy *memóriaobjektum* vagy *memóriaobjektum_lista*		Törli a felsorolt összes memóriaobjektumot.
TÖRÖLFÁJL ERASEFÁJL, ERFÁJL	*fájl*		Törli a *fájl*t.
TÖRÖLÍRÓLAP, TÍ CLEARTEXT, CT	*nincs*		Törli az írólap tartalmát, vagyis lefedi az írólap aktuális háttérszínével.
TÖRÖLMEZŐ REMPROP	*szó1 szó2*		Törli a *szó1* nevű rekord *szó2* nevű mezőjét.
TÖRÖLMUNKA ERALL	*nincs*		Törli az összes memóriaobjektumot és teknőcöt, lefuttatja az COMLOGO.INI fájlt – ha ez létezik a Logo alapkönyvtárában. Letörli a rajzlapot is, és alapállapotba hozza a Logót.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
TÖRÖLRAJZ CLEAN	*nincs*		Törli a rajzlap tartalmát, vagyis lefedi az aktuális háttérszínnel.
TÖRÖLRAJZLAP, TR CLEARSCREEN, CS	*nincs*		Törli a rajzlap tartalmát, vagyis lefedi az aktuális RAJZLAPSZÍN-nel.
TÖRÖLTEKNŐC ERASETURTLE, ERTURTLE	*teknőc* *teknőc_lista*		Törli az adott teknőcöt, illetve teknőcöket.

U, Ú, Ü, Ű

UTOLSÓ LAST	*akármi* (nem üres)	*akármi*	A bemenetként kapott sorozat utolsó eleme.
UTOLSÓNAK LPUT	*akármi lista* *szó1 szó2* *képsor1 képsor2*	*lista* *szó* *képsor*	A *lista, szó2, képsor2* végére illeszti rendre *akármit, szó1*-t, *képsor1*-t.
UTOLSÓNÉLKÜLI, UN BUTLAST, BL	*akármi* (nem üres)	*akármi*	Eredménye az eredeti paraméter utolsó elemének elhagyásával kapott sorozat.
ÚJRASOROL REORDER	*teknőc_lista*		A *teknőc_listá*ban megadott sorrendbe rakja a teknőcöket. A sorrend a MIND és KIFIGYEL eljárásokkal vizsgálható.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ÚJTEKNŐC MAKETEKNŐC	*szó [ ]* *szó [szám1 szám2 szám3]* *szó [szám1 szám2 szám3..szós..alak]* **Hld:** *szó alak* **Hld:** *(szó)* ahol ... *szós* ... akármilyen szavak a következők közül: TF, TL, TRD, TVT, TOLLATFEL, TOLLATLE, TOLLRADÍR, TOLLVÁLTÓ, LÁTHATÓ, LÁTHATATLAN, FIGYELJ		Új teknőcöt hoz létre a *szó*ban megadott néven, a többi paraméter az aktuális beállításokra vonatkozik. (Az *alak képsor*, vagy *"LGW_fájl* vagy *változó* képsor értékkel.)
ÚT PATH	*nincs* *(Logo_alkönyvtár)*	*útvonal* *útvonal*	Az aktuális munkakönyvtár. a (DEMO, FELADAT, GYEREK, HANG, KEPSOR, PROJEKT, RAJZLAP) aktuális Logo alkönyvtár helyét adja.
ÚT! SETPATH	*útvonal* *(Logo_alkönyvtár útvonal)*		Az útvonalban adott paraméter lesz a munkakönyvtár. A (DEMO, FELADAT, GYEREK, HANG, KEPSOR, PROJEKT, RAJZLAP) Logo alkönyvtárat állítja be.
ÜRES? EMPTY?	*akármi*	*logikai érték*	IGAZ, ha paramétere üres szó, üres sorozat vagy üres képsor, egyébként HAMIS.
ÜRESKÉPSOR EMPTYIMAGE	*nincs*	*képsor*	Egy üres*képsor*, olyan, amely egyetlen fázist sem tartalmaz.
ÜZENŐ MESSAGEBOX	*szöveg1* *szöveg1 szöveg2*	*szöveg(?)*	Utasításablakot nyit. Ennek címsora *szöveg2* (ha megadjuk), és az ablakban belül nem szerkeszthető üzenet *szöveg1*.

V

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
VAGY OR	*logikai érték 1 logikai érték2 (logikai érték ...)*	*logikai érték logikai érték*	Logikai vagy művelet.
VÁGÓLAP CLIPBOARD	*nincs*	*lista*	A *Logo_eljárás* egy fájlkezelő eljárás, mely a fájlok betöltését, illetve elmentését teszi lehetővé. Ekkor a műveletet közvetlenül a "VÁGÓLAP tartalmával végzi el.
VÁRJ WAIT	*szám* pozitív, nem nagyobb mint 65535	*szünetelés*	A *szám*ban megadott ezredmásodpercre megáll a program futása.
VÁRJAMÍG WAITUNTIL	*[logikai érték_kifejezés]*	*várakozás*	Mindaddig várakozik, amíg a paramétere értéke igaz nem lesz.
VÉGE END	speciális szó		Az eljárásdefiníció végét jelzi. A VÉGE szó zár minden egyes saját eljárást.
VÉGREHAJT RUN	*Logo_utasítás_lista kifejezés*	*akármi*	Eljárás: a lista összes utasítását végrehajtja. Függvény: eredménye a *kifejezés* eredménye.
VÉLETLEN RANDOMIZE	*nincs*		Véletlenszám-generátor inicializáló eljárása.
VÉLETLENSZÁM, VSZ RANDOM	*szám*	*szám*	Véletlenszámot ad a [0, *szám*−1] intervallumban.
VISSZATÉR STOP	*nincs*		Leállítja az aktuálisan futó eljárást és a vezérlés egy szinttel feljebb kerül.

X

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
XHELY, X XCOR	*nincs*	*szám*	Az aktív teknőc x koordinátája.
XHELY!, X! SETX	*szám*		Az aktív teknőc x koordinátáját állítja be.
XYHELY!, XY! SETXY	*szám1 szám2*		Az aktív teknőc x, illetve y koordinátáját állítja be.

Y

YHELY, Y YCOR	*nincs*	*szám*	Az aktív teknőc y koordinátája.
YHELY!, Y! SETY	*szám*		Az aktív teknőc y koordinátáját állítja be.

A könyv a Comenius Logo szakértői tanfolyamának anyagát foglalja össze rendszerezett és közérthető, tanárok és más felnőtt felhasználók számára hasznosítható formában. Megismertet a Logo programozási nyelv alapjaival, magával a Comenius Logo programcsomaggal, annak installálásával és működtetésével. Külön fejezetek szólnak a Comenius Logo szoftver egyes programrészeiről, a program különböző funkcióiról. Nagyobb teret szentel a könyv a rajzpogramoknak, a Logo lehetőségeinek a kordináta-rendszer és a fraktálok vonatkozásában, a Logo-animációknak, különösen az interaktív mozgatási (alakzat változtatási) módozatoknak, és a hangoknak. Önálló rész foglalkozik a Logo függvényekkel, ezen belül a szövegkezeléssel. Nem nélkülözi a kötet a módszertani tanácsokat sem, s külön értéke a függelékben a Comenius Logo 3.0 verzió definitív leírása.

ISBN 963-09-3956-8

9 789630 939560