

A Comenius Logo országlicencének megvásárlása és magyar nyelvű verziójának elkészülte után egyre több iskola vásárolja meg a 3.0 számozással jelölt programcsomagot. Ők 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 mozgató (alakzatváltoztatási) módokat 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 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 fejlődés-lélektani 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 (Massachusetts 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 technógrafiáját csak néhány évvel később fejlesztették ki. A Logo igazi oktatási sikerét és **nemzetközi népszerűségét** tulajdonképpen a technógráfika 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 technóca a számítógép billentyűzetén keresztül utasítható a számára „érthető” feladatok elvégzésére: tud adott távolsággal előre vagy hátra menni, adott szöggel jobbra vagy balra elfordulni, tollat (ami a hasára van erősítve) felemelni, leereszteni, más színűre cserélni, ezáltal mozgásával érdekes nyomokat hagyni a képernyőn. A technógráfika fokozatosan egy új tudományterület, a **technógeometria** megszületéséhez vezetett.

A diák azáltal, hogy parancsot ad a képernyőtechnócnak, rögtön ellenőrizheti gondolkodásának és cselekedeteinek következményeit. Megfigyeli utasításainak hatását, majd módosíthatja azokat céljának tökéletesebb megvalósítása érdekében. A kipróbálások és módosítások sorozata egybeesik 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-

tőiségei 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 megértése.

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 paramétereirei 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.). Hagományos értelemben vett értékadás nincs. Természetesen definiálni kell az automata kezdeti vagy alapállapotát, s egy vagy több utasítást, amely egyes állapotkomponenseket, vagy az összesét 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 festendő felület, milyen messze van a festőkartól stb.) (A Lógók 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 állapotfüggők 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 által rendelkezésre:

$$- f(x) = g \circ h(x) \quad - \text{függvénykompozíció}$$

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

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

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).

A COMENIUS LOGO 3.0

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, a7-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 a Pozsonyi Comenius Egyetem Informatika Oktatási Tanszékének 1992 óta fejlesztett terméke. Szerzői: Andrej Blaho, Iván 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 változtható, ezzel keltve mozgásérzetet) is működtethetők. Az alakzatok egyes fázisainak megtervezésére kiegészítő képsorszerkesztő szolgál. A kép- és vektorműveletekkel gazdagított sorozat- és rekordkezelés, szöveglablak, sokoldalú színkezelés, zeneszerkesztő, WAV- és AVI-állományok, Windows-programok lejátszása mind újdonság más elterjedt Logo nyelvjáráshoz 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, algoritmusait, 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

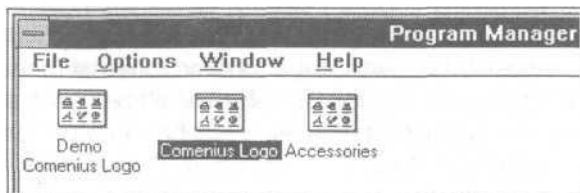
386/SX vagy annál fejlettebb (ajánlott legalább 486-os) konfiguráció, VGA grafikus kártya, 4MB RAM, 6 MB merevlemez-terület. Windows 3.1 vagy annál újabb verzió.

Soundblaster kompatibilis hangkártyával a zenei kiegészítés minden lehetősége élvezhető.

A COMENIUS LOGO ELINDÍTÁSA ÉS A DEMOJÁTÉKOK

A COMENIUS LOGO ELINDÍTÁSA

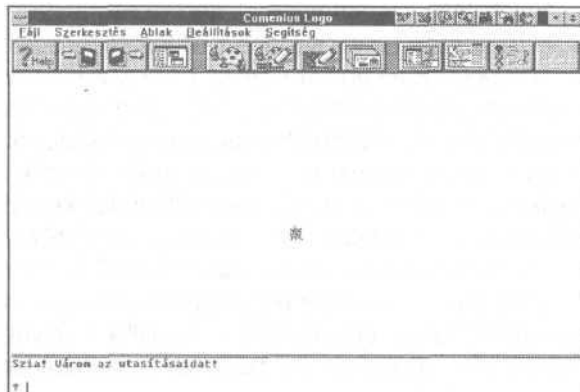
Indítsuk el a Windows programot! Kattintsunk a **Comenius Logo** csoportikonra!



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 Lógóval készült demonstrációs programok. Próbáljuk ki őket!



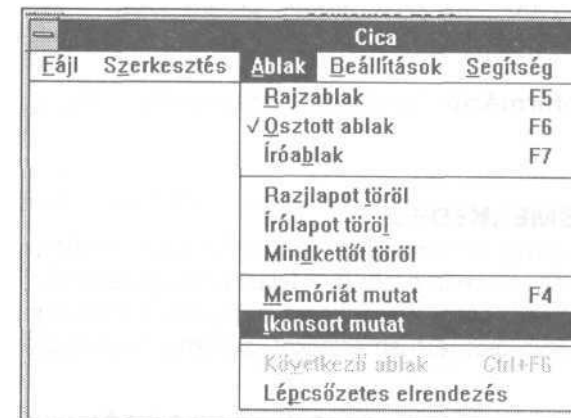
Programot lehet választani úgy, hogy kétszer rákattintunk annak ikonjára, vagy egyszeri kattintás után az OK gombra. A programok a betöltés után azonnal elindulnak és további útmutatásokat adnak. Kattintsunk az ikonsoron a **Rajz** ikonra:



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 játékcsoomag, a Gyerekjátékok ikonra kattintva indítható el.

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

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

Információ: Általános információkat tartalmaz a programcsomagról.

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 jobbra tetszőleges sorrendben végrehajtva, alapvető ismereteket szerezhetünk a magyar Comenius Logo használatáról.

A COMENIUS LÓGÓBÓL 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 tükröhatást eredményező rajzolóprogram.



Helipeti

A Comenius Logo egyik érdekes hatását illusztráló program.



Hexa

„Tedd egyszínűvé!” játék a hexagonokra kattintással.



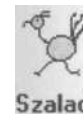
Merlin

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.

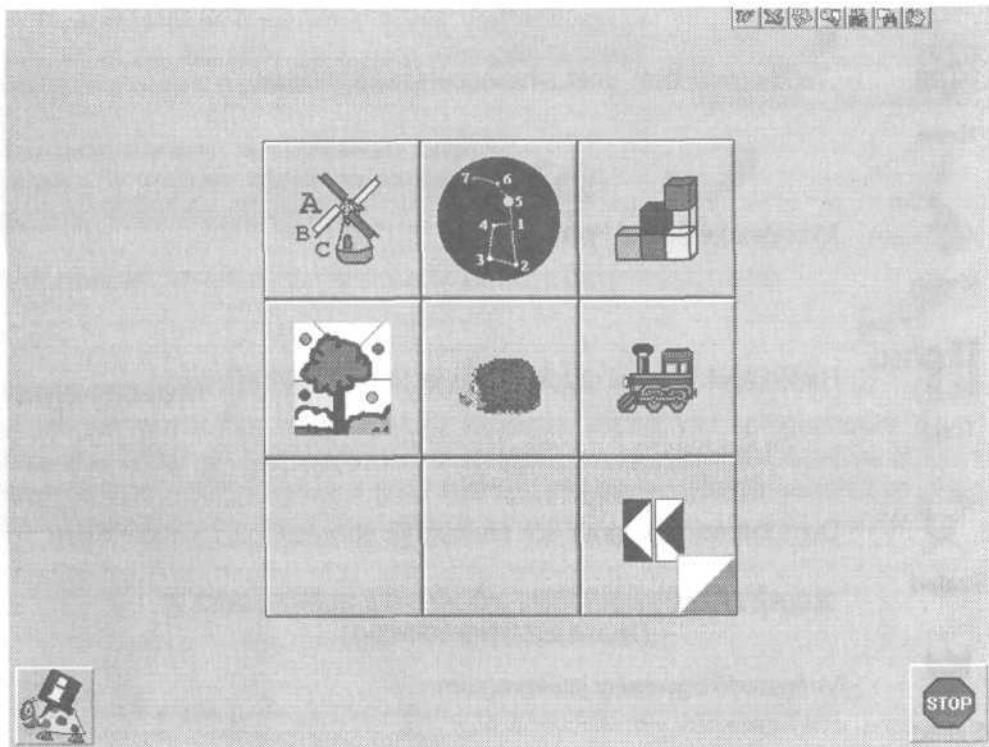


Utca

Egyszerű, de látványos animációt demonstráló program.

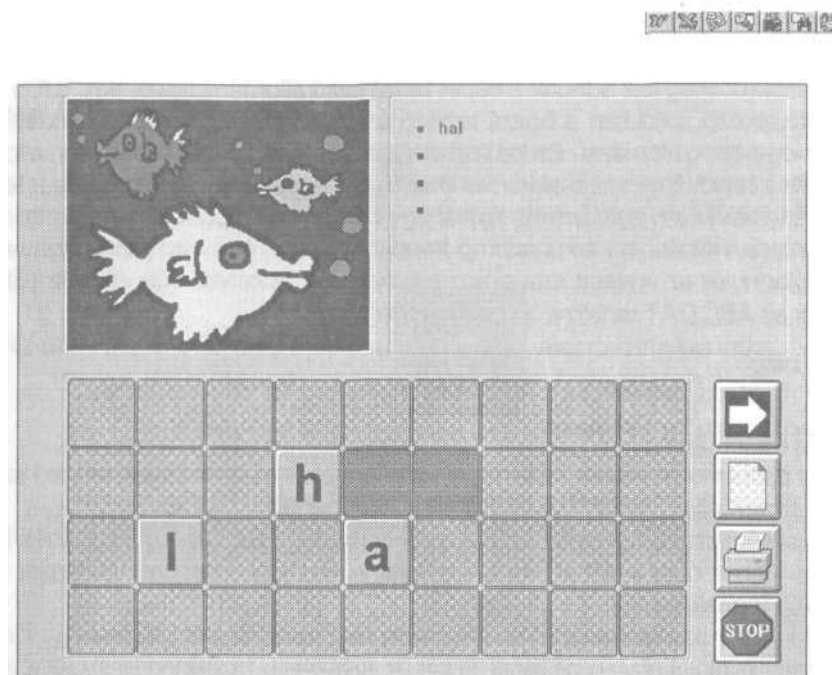
A **GYEREKJÁTÉKOK** a  ikonra való kattintással indul.

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 **SZOKIRAKO JÁTÉK**, a  ikonra kattintva indul.



Adott egy 4x9-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 szinkiemeléssel 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 (J 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 írunk a megfelelő sorba:

Ma_jó_ id_o_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 TUDIMI

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:

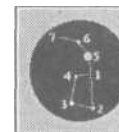
c:\windows\paintbrush\cica.bmp

Ha a megadott állomány mérete a 246x204-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 246x204-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.

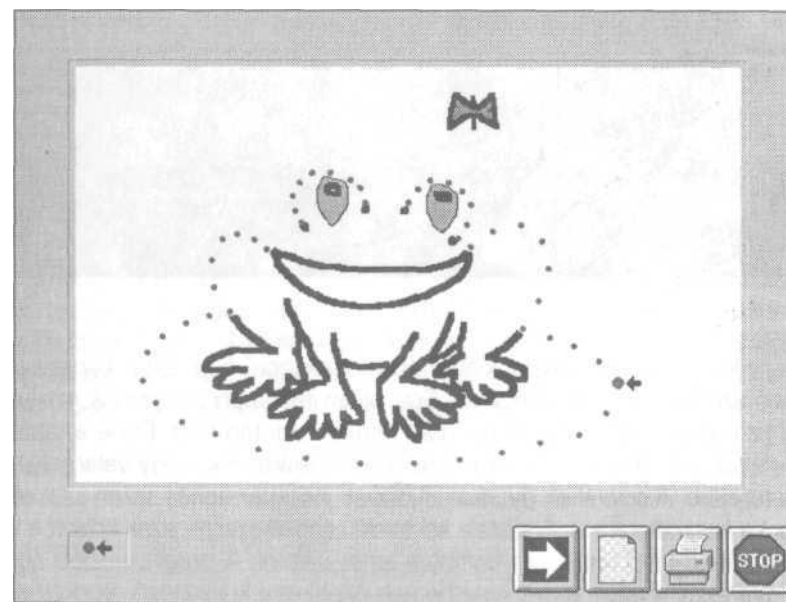
A HANGÁLLOMÁNYOKRÓL

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

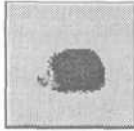


ikonra kattintva indul.

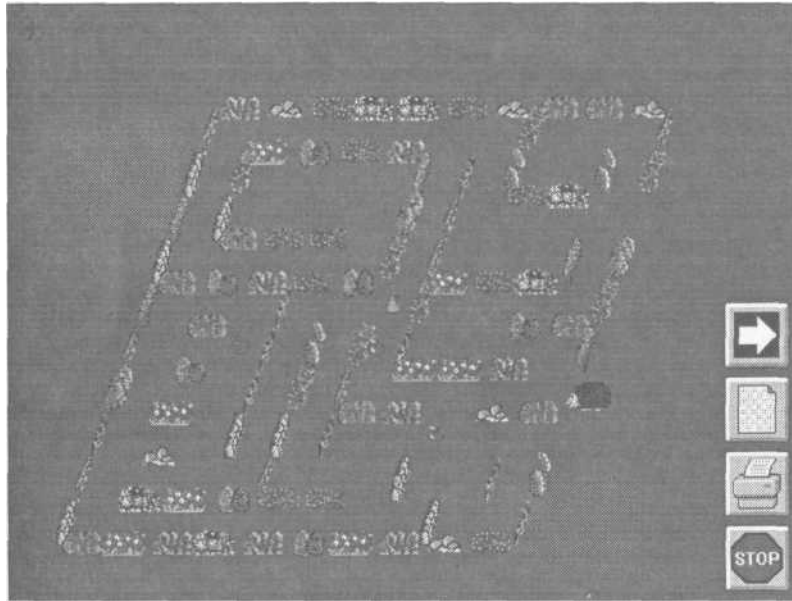


Ez a játék a rejtélyűjságokból jól ismert. Lényege, hogy a gyerekek a betűk, számok és más objektumok sorba rendezését ismerjék fel. Ha a szétszórt és különböző módon (betűkkel, számokkal stb.) megjelölt pontokat a megfelelő sorrendben összekötjük, valamilyen kép alakul ki. A pontok összekötéséhez az 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 GYÜMÖLCSÖSKERT a

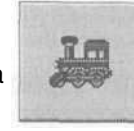


ikonra kattintva indul.

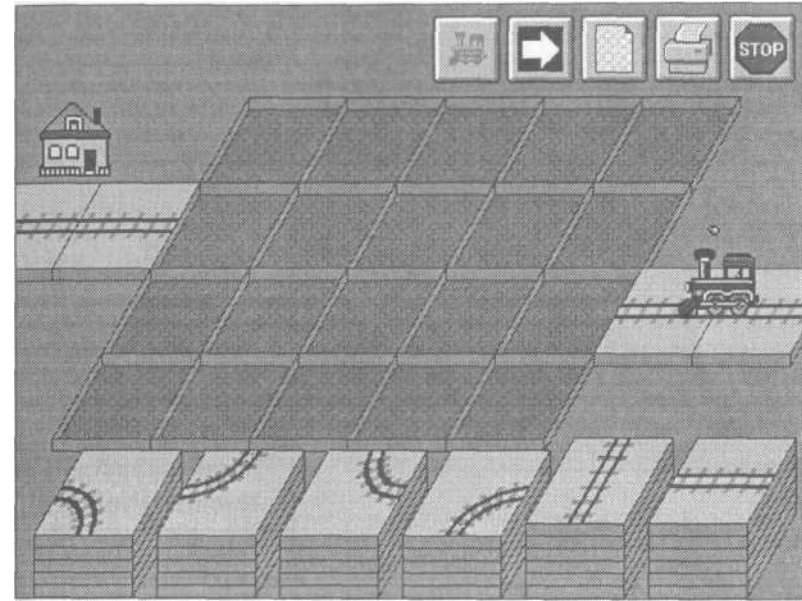


Adott egy kis kertlabirintus gyümölcsökkel. Az a feladat, hogy a süni segítségével összeszedjük a terméseket. Egyszerre csak három gyümölcsöt bír el a süni, ezért minden harmadik felszedett gyümölcs után ki kell őt vezetni a kertből. A sünit úgy irányítjuk, hogy az orra előtt húzzuk az egérmutatót, amit 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 sünik verembe esik. Egy még nehezebb változatban a vermek rejtettek.

A VASÚTÉPÍTÉS a



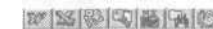
ikonra kattintva indul.




Adott egy terepasztal, és meghatározott számú, különböző alakú sínelem. A sínelemből össze kell rakni a vasúti pályát, hogy a vonat a pályaudvarra mehessen. A sínelemek kiválasztása és elhelyezése a megfelelő sínelemre és terepontra 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.

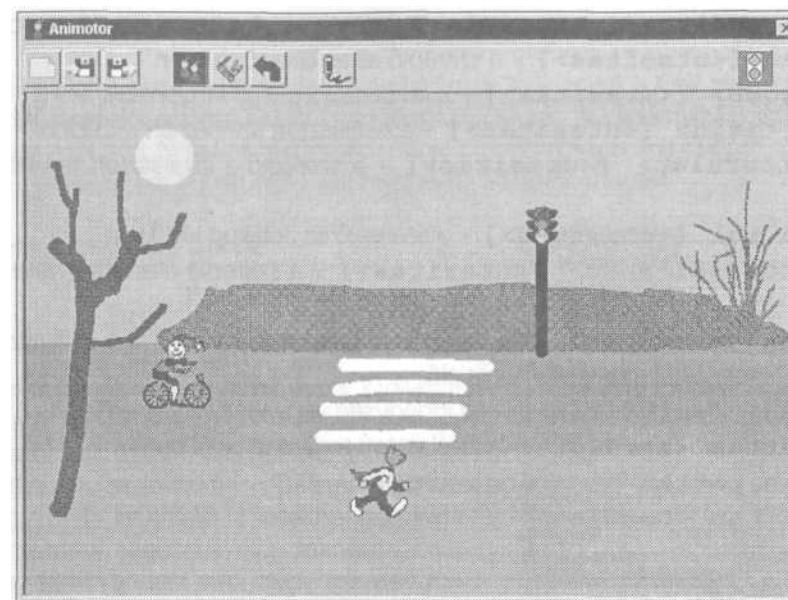
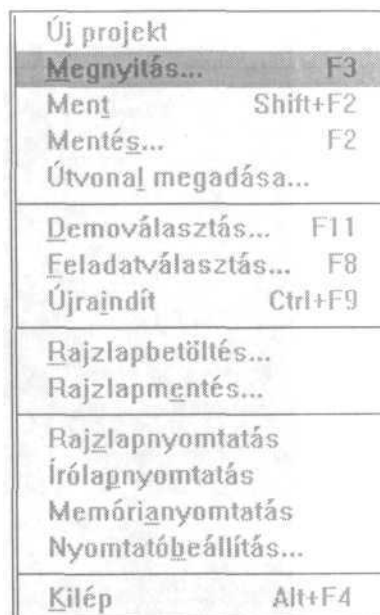
A vasútépítés a leginkább érdekes, gondolkodást igénylő feladatcsoport, némelyik sínelrendezés megoldása komoly fejtörést okoz még a felnőtteknek is. Lehet, hogy némelyik esetben a feladat hosszas gondolkodás után is megoldhatatlannak tűnik. Természetesen mindegyik feladatnak van legalább egy - némelyiknek 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.

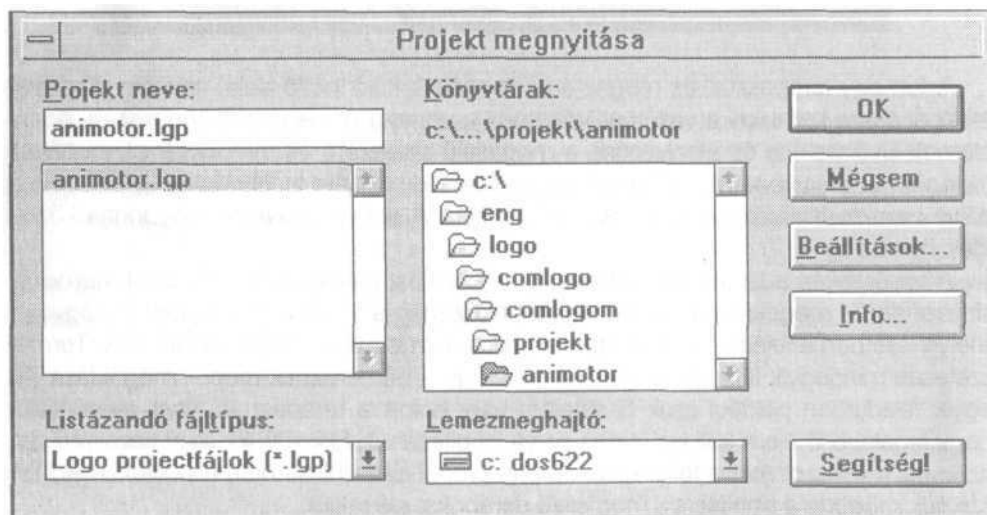
A Megnyitás sorra kattintva kapjuk az alábbi ablakot:



A beszédes ikonsortól útbaigazítást kaphatunk.



rakattintva behívhatunk egy demonstrációs példát: zebra.lgp



Ha az OK gombra kattintunk, betöltődik az Animotor program.

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:

kli^k [<utasítás>] - bal egérgomb lenyomása

ü^tkö^zés <szereplő> [<utasítás>] - a szereplő ütközése másik szereplővel

tá^volsá^g <pont> <szám> [<utasítás>] - a szereplő távolsága a ponttól a számnál nagyobb

ELEMI ALAKZATOK RAJZOLÁSA

távolság <szereplő> <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 <szereplő> <állapot> [**<utasítás>**] - a kiválasztott szereplő állapota a választásnak megfelelő

idő <idő> [**<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 <szereplő> - 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

Indítsuk el a Comenius Lógót! 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 egy pár 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.)



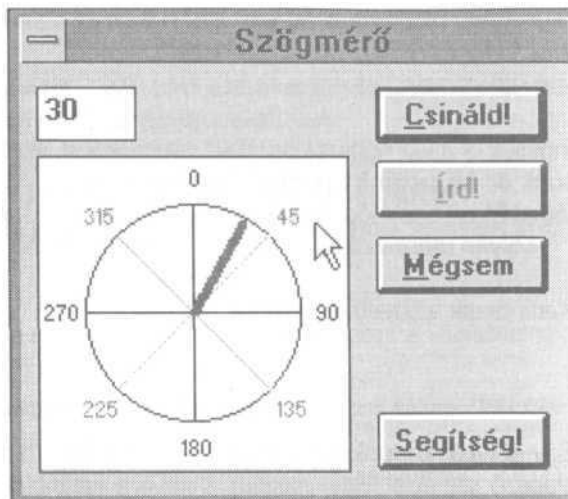
A teknőc az aktuális helyétől az aktuális irányba lép előre a megadott távolságra.



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 ? törölr rajzlap** menüből, vagy törölhetünk a következő utasítás beírásával is:

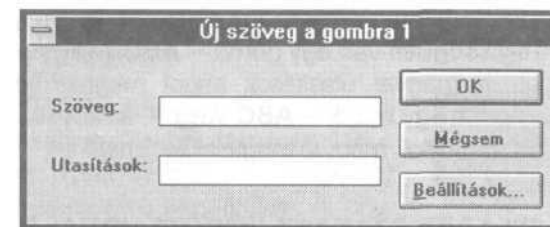
Az elemi alakzatok rajzolásának másik lehetősége egy rajzoló projekt készítése. Az utasítások rárakhatók egy-egy gombra, a gombokra pedig csupán rá kell kattintanunk ahhoz, hogy végrehajtsódjanak. Kattintsunk a következő ikonra:



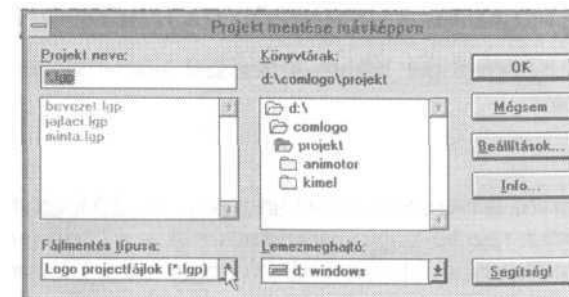
A következő gombsor jelenik meg, melynek gombjaihoz utasításokat rendelhetünk.



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 **.lpp** 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** utasítással kezdődnek és az **end** utasítással fejeződnek be. Ebben az esetben a **Kiválasztott alapszavak** ablakban is angolul jelennek meg az utasítások.



A Segítségben 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

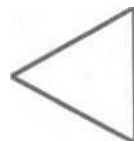
Az utasítások végrehajtásához szükséges adatokat egymástól helyközzel elválasztva közvetlenül is felsorolhatjuk: **? előre 50 jobbra 90 hátra 30 balra 45**

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.

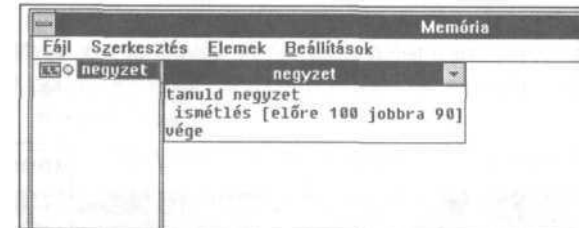


4. FELADAT: NÉGYZET RAJZOLÁSA ELJÁRÁSSAL, AVAGY ÚJ SZÓRA TANÍTJUK A TEKNŐCÖT

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 tanulóshoz a következő ikonra kell kattintani:



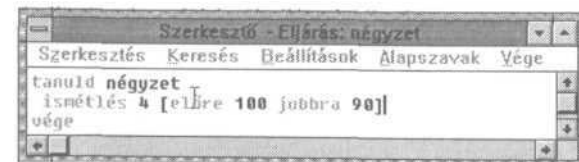
Ekkor felbukkan a tudást tároló **Memória** ablaka. Itt található minden új szó, amit megtanítunk. Az **Elemek** menüpont **Új eljárás** pontját kiválasztva taníthatjuk meg a teknőcöt egy szóra. A szó (azaz a tevékenységsorozat neve, amit hagyományosan eljárásnévnek hívunk) beírása után megjelenik az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



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

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

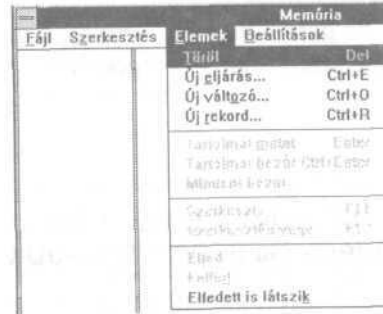
tanuld négyzet
ismétlés 4 [előre 100 jobbra 90]
vége



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

Tanulásként 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 beírásakor 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 szókinccs 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.

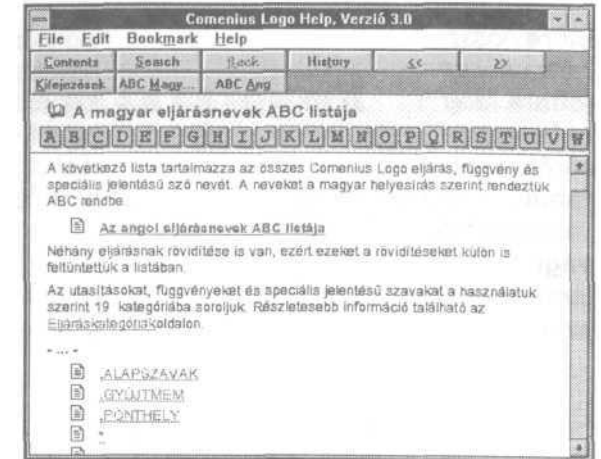
Ezután a most megírt „okosabb” tevékenység pontosan úgy használható, mint a teknőc által eddig ismert szavak:

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

? négyzet 100 négyzet 25

TOVÁBBI UTASÍTÁSOK

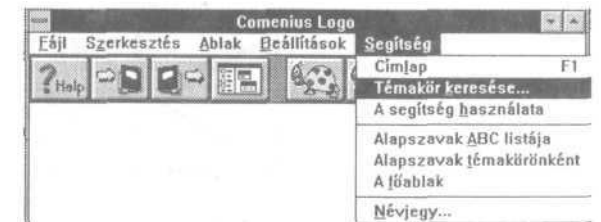
A többi alapszóval a **Segítség** menüpont **Alapszavak ABC listája** pont választásával ismerkedhetünk meg. Egy-egy utasítás használata és magyarázata a szóra való kattintásnál előbukkanó ablakban olvasható el.



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 köszönjük el tőle:



UTASÍTÁS-ÖSSZEFOGLALÓ

előre *hossz*
hátra *hossz*
jobbra *szög*
balra *szög*
törölrajzlap
tanuld *név paraméterek*
utasítások
vége
ismétlés *darab [utasítások]*
szia

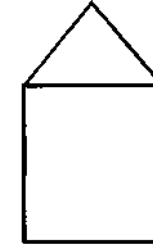
előrelépés *hossz* egységgel
hátralépés *hossz* egységgel
jobbra fordulás *szög* fokkal
balra fordulás *szög* fokkal
rajzlap letörlése
paraméteres eljárás definiálása

ciklus darabszám
a munka befejezése

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 _____

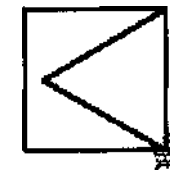
tanuld háromszög :hossz
ismétlés 3 [előre :hossz balra 120]
vége

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

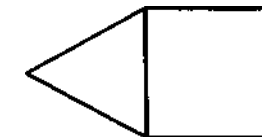
tanuld házikó :hossz
négyzet :hossz háromszög :hossz
vége

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 rajzoláskor 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 nével menti) almenüjében.

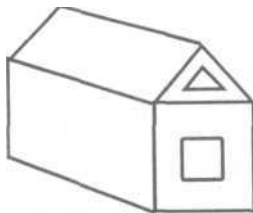
A **Megnyitás** almenüvel tölthetünk be kész programokat.

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!



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

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:

A földszint a ház elejéből, illetve oldalából áll:



Ezt a következőképpen valósíthatjuk meg:

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

Ezt a következő programsorokkal valósíthatjuk meg:

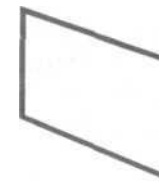
tanuld földszint :hossz
eleje :hossz balra 90 előre :hossz
jobbra 90
oldala :hossz balra 90 hátra :hossz
jobbra 90
vége



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 érzük 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

0'-'

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 r hossz balra :szög]
vége

A Logo utasításai tetszőlegesen írhatók külön sorokba, illetve ugyanabba a sorba, szóközökkel elválasztva. Egy utasítást azonban alapesetben nem lehet megtömi, több sorba írni. Az ismétlések (és más későbbi utasítások) viszont általában elég hosszúak, s ó lenne sorokra törölni 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 mostanitol 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!
- 1 • Alkalmazzunk általánosan kimondható szabályokat!
2. 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 előlről kell kezdenünk a munkát; a kész megoldás nem reprodukálható egyszerűen.
14. Az *összetett tevékenységeket mindig foglaljuk eljárásba!*
ki 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 részeketől, 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! <i>színkód</i>	ezután a megadott kódú színnel rajzol
tollvastagság! <i>szám</i>	ezután az adott vastagságú tollal rajzol

KÖRÖK, KÖRÍVEK, REKURZIÓ, FAK

1. FELADAT: SZABÁLYOS SOKSZÖG RAJZOLÁSA

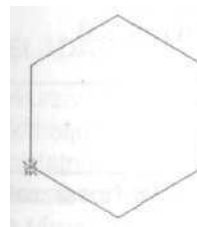
A háromszög, négyzet stb. eljárások alapján készítsünk általános sokszög(poligon)-rajzoló eljárást:

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

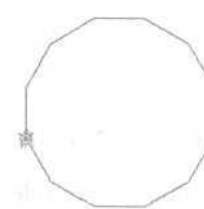
Matematikai tanulság: a sokszög külső szögeinek összege mindig 360 fok. A Lógóban ez nem *bemagolandó* szabály, hanem kísérleti úton kikövetkeztethető tény. Hiszen ha a teknőc bármilyen szabályos sokszög vagy „szabálytalan” alakzat mentén halad, de végül a kiindulási állapotába ér vissza, akkor összességében 360 fokos fordulatot tesz meg. (A közbülső, előjelesen vett szögelfordulások összege 360 fok.) Ezt a **teljes tek-nőctételnek** 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 n értéke.

A poligon eljárást sokféle paraméterrel hívhatjuk meg:

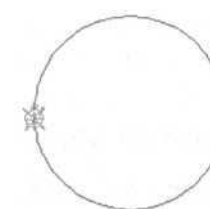
poligon 6 100



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:

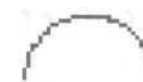
tanuld poligon :oldal :hossz :szög
ismétlés :oldal "
[előre :hossz balra :szög]
vége

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 figyelniük 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=360fok):

**tanuld kör :átmérő
poligon 100 ~
:átmérő7100*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 újraaktiválni az előbbieket 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.

fordulat
27

lépés
19

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

fordulat
27

lépés
19



6. FELADAT: SPIRÁL RAJZOLÁSA

A rekurzív sokszög rajzolását könnyen átalakíthatjuk spirálrajzolássá, csupán arra kell biztatnunk a teknőcöt, hogy mindig egy kicsit nagyobb távolságot, a *hossznak* a *raV*-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 _____**

? reksokszög 40 10 6 2

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

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



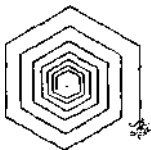
A rekurzív sokszöget másképp is módosíthatjuk:

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

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

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

? újreksokszög 40 10 6 1.05



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

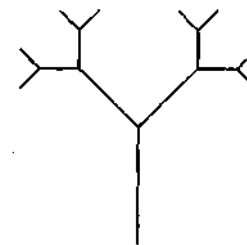
3. Az N körből álló hóembert úgy kell rajzolni, hogy megrajzolunk egy félkört, fölé rajzolunk egy $N-1$ körből álló hóembert, majd befejezzük a félkört: Ez az *általános rekurzió*, az eljárás a közepén hívja meg saját magát.

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.

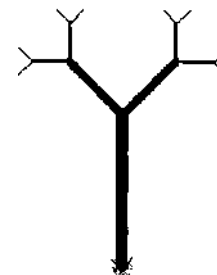
A fa egy törzsből áll, amelynek tetején két ág nő ki, s mindkét ág tulajdonképpen egy-egy alacsonyabb, rövidebb törzsű fa. Ezekről a fákról ugyanezt mondhatjuk el.



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 választás két utasításcsoport között
[akkor ág utasításai]
[különben ág utasításai]

vagy utasítások feltételes végrehajtása
ha feltétel [akkor ág utasításai]

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

SORMINTAK, TERULETMINTAK

1 FELADAT: SORMINTA AZONOS ELEMKBŐL

$f \setminus$ 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:

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.

Ugyanez az eljárás rekurzívan is megfogalmazható:

[~
'ga-

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

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

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.

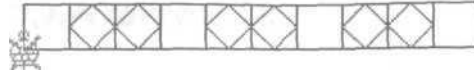


Egy logikai változó paraméterrel jelezzük, hogy éppen melyik alapelemet kell alkalmazni:

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

Megoldható más mintázatképzési szabályok kezelése is:

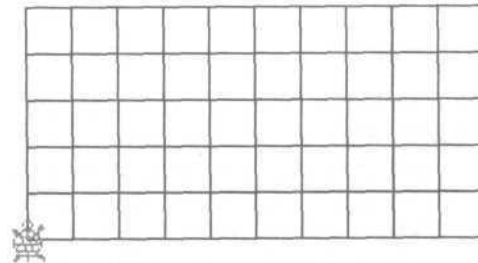
Ebben az esetben a logikai változó helyett egy számváltozót használunk, amelynek most éppen a 3-mal vett osztási maradékát vizsgáljuk. Ha a maradék 1. akkor az első alapelemet kell rajzolni, ha 0 vagy 2, akkor pedig a másodikat.



```
tanuld sorminta :n :x :y :szám
  ha :n>0 [ha :szám mod 3=0 "
    [alapelem 1 :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
```

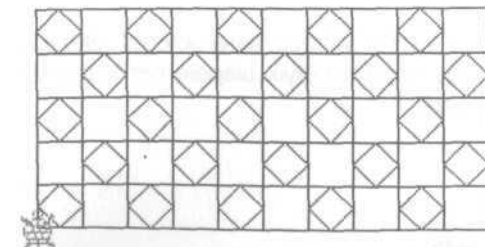
```
tanuld felfelé :y
  tollatfel előre :y tollatle
vége _____
```

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

A területminta is képződhet kétféle elemből:

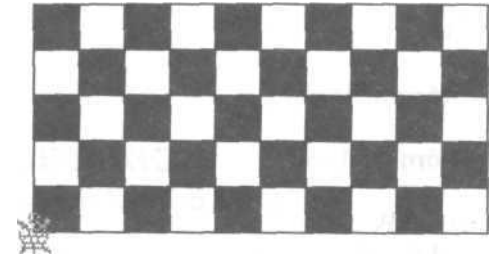
```
tanuld területminta :m :n :x :y :logi
  ha :m>0 "
    [sorminta :n :x :y :logi felfelé :y ~
    területminta :m-1 :n :x :y ~
    nem :logi lefelé :y]
vége _____
```

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



4. FELADAT: FESTETT TERÜLETEK

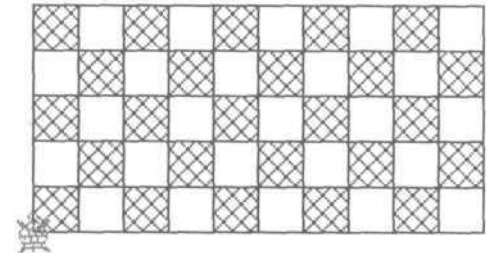
A területminta egyféle alapelem festett és festetlen változatából is állhat:



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.

A KOORDINÁTA-RENDSZER ÉS A FRAKTÁLOK

UTASÍTÁS-ÖSSZEFOGLALÓ

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

A COMENIUS LOGO ÉS A KOORDINÁTA-RENDSZER

Az *euklideszi* 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 logikái stílusát 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 0 pontot (pólus) és egy 0-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 polárkoordinátaival 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 teknőc-tevékenységek 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 (es akár a térben) együttesen jelentik a teknőcgeometria elemeit.

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

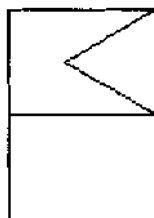
Teknőcjellemző	Beállítása	Lekérdezése
helye (x, y koordináta)	xhely! <i>érték</i> , yhely! <i>érték</i> xyhely! <i>érték érték</i>	xhely, yhely, hely
iránya	irány! <i>szög</i>	irány
alakja	alak! <i>kép</i>	alak
rajzeszköze helyzete	toll!, tollatle, tollatfel, toll- radír, tollváltó	toll
tolla színe	tollszin! <i>színkód</i>	tollszin
tolla vastagsága	tollvastagság! <i>sorszám</i>	tollvastagság
tolla mintázata	tollminta! <i>sorszám</i>	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 irúdhossz :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:

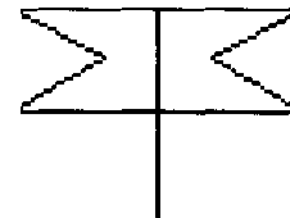
Ha a lépéseket változatlanul hagyjuk, de a fordulatok irányát ellentétesre változtatjuk, akkor a teknőc kiinduló állapotán (pozíció és irány) átmenő egyenesre tükrözzük az alakzatot.

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 pozitív szögelfordulásokat negatívra és viszont).

tanuld tükrözészló :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 _____



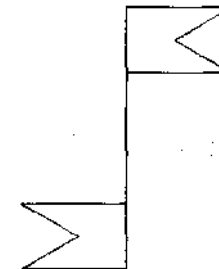
2. TÜKRÖZÉSI TÉTEL

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 forgatszázló :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 kiinduló 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



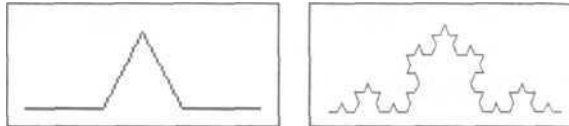
Bármely síkbeli egybevágósági és hasonlósági transzformáció előállítható a fentiek sorozataként (megfelelő számú transzformáció egymásutánjával), így az eredeti alakzatnak síkban tetszőlegesen elhelyezkedő egybevágó vagy hasonló képe előállítható.

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

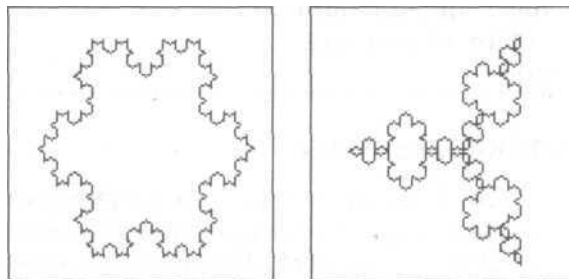
Az egyik nevezetes fraktál az úgynevezett Koch-görbe. Ez egy rekurzív definícióval rajzolt görbe, amelynek adott szintű változatai rajzolhatók meg.



```
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

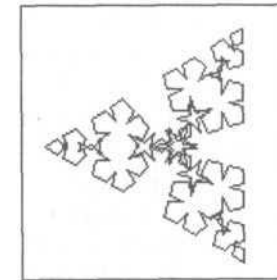
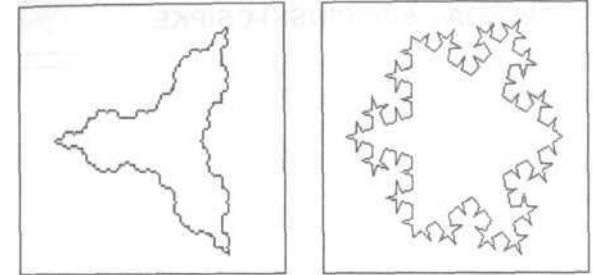
A Koch-görbe nagyon szép hópehely-, illetve kristályszerű ábrákat adhat, ha a kiinduló egyenes szakaszt háromszöggel helyettesítjük. Másféle ábrát kapunk, ha a háromszöghöz képest kifelé, illetve ha befelé rajzoljuk a Koch-görbét.



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

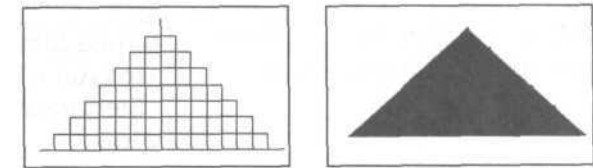
```
tanuld kristály :szint .hossz
ismétlés 3 [Koch :szint :hossz balra 120]
vége_____
```

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.

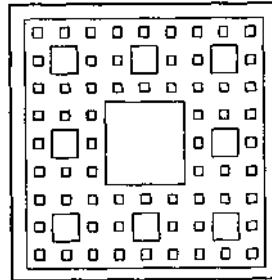
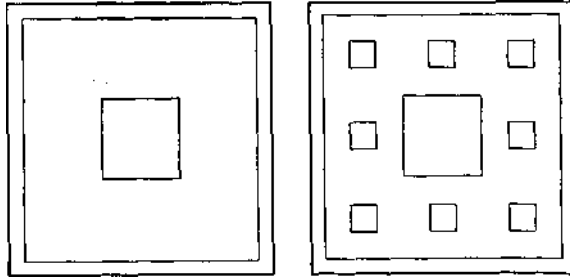


```
tanuld Koch :szint :hossz
ha :szint = 1
  [előre :hossz] ~
  [Koch :szint -1 :hossz / 2 balra 90 "
  Koch :szint -1 :hossz / 2 jobbra 180
  Koch :szint -1 :hossz / 2 balra 90 "
  Koch :szint -1 :hossz / 2]
vége_____
```

AZ ANIMÁCIÓ

4. FELADAT: SIERPINSKI-CSIPKE

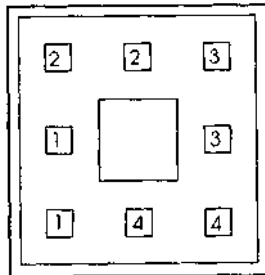
Egy másik érdekes, síkbeli alakzat a Sierpinski-csipke, amely egy négyzetlapból keletkezik úgy, hogy újabb és újabb belső négyzeteket vágunk ki.



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 él-hosszúságú 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 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.

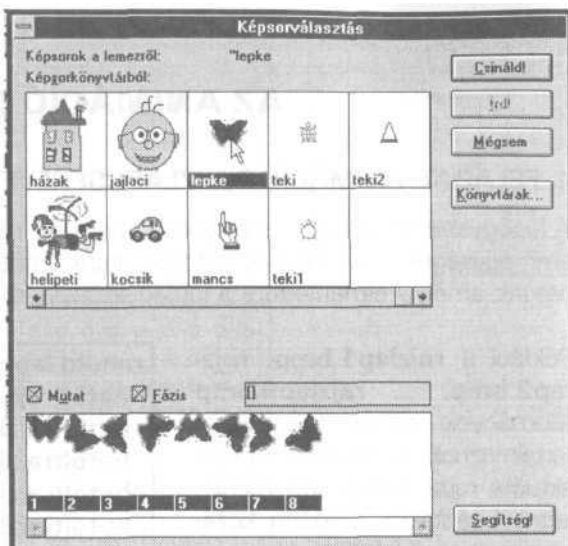
```
tanuld lapozz
betöltrajzlap "rajzlap1.bmp várj 10
betöltrajzlap "rajzlap2.bmp várj 10
betöltrajzlap "rajzlap3.bmp várj 10
betöltrajzlap "rajzlap4.bmp várj 10
betöltrajzlap "rajzlap5.bmp várj 10
betöltrajzlap "rajzlap6.bmp várj 10
betöltrajzlap "rajzlap7.bmp várj 10
betöltrajzlap "rajzlap8.bmp várj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

A **betöltrajzlap** utasítás paramétere egy állomány neve. A szókonstansokat a Lógóban idézőjel vezet 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 mozgására nem nagyon szerencsés a képek betöltése háttértárról.

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.



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

tanuld repül
animációmód! "igaz alak! "lepke
ismétlés 100 ~
[fázis! 2 várj 100 fázis! 3 várj 100 előre 3]
animációmód! "hamis
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. Rákatintva 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.

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!



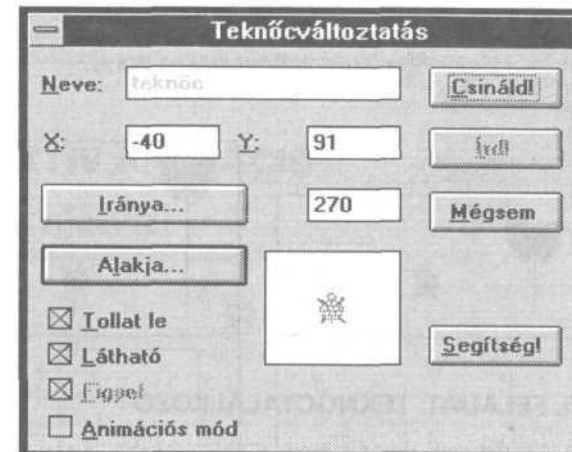
Ehhez úgynevezett véletlenszámokat alkalmazunk: véletlenszám N a Lógóban egy 0 és N-1 közötti egész számot jelent.

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

4. FELADAT: SOK SZEREPLŐ (TEKNŐC) EGYÜTTES MOZGATÁSA

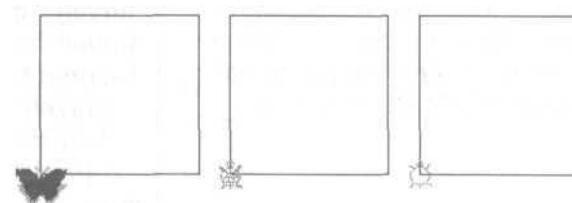
Hozunk létre új teknőcöket úgy, hogy az egér jobb gombjával rákattintunk valahol a képre. Az **Új teknőc** sort kiválasztva megadhatók a teknőc tulajdonságai (név, irány stb.).

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.

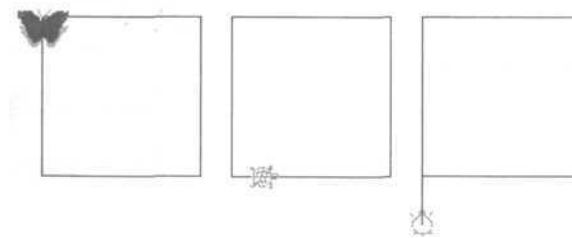


Ha több teknőcünk is van, s mindegyik figyel az utasításainkra, akkor mindegyik ugyanazt a rajzot rajzolja. (A figyelés a teknőc létrehozásakor megjelenő ablakban beállítható.)

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]

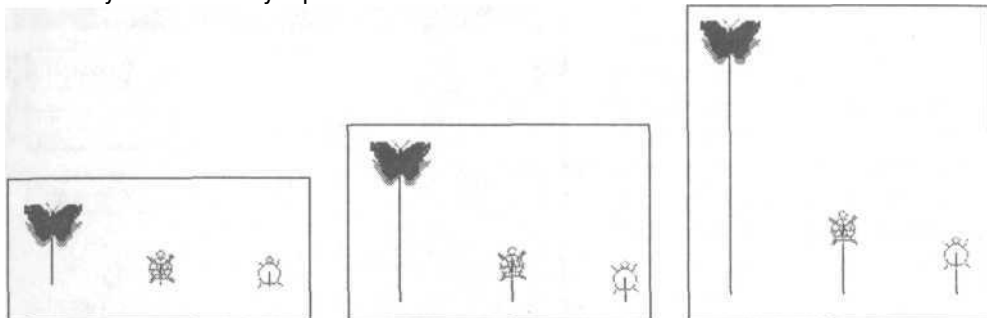
várj 100

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

várj időtartam

alak! "név

fázis! sorszám

véletlenszám n

figyelj teknőc(ök)

kérem teknőc [...]

takar? teknőc(ök)

betölti az aktuális rajzablakba a név nevű képet

időtartam, ezredmásodperc várakozás a teknőc aktuális alakját névre változtatja animációs módban a teknőc képének sorszám, fázisát jeleníti meg

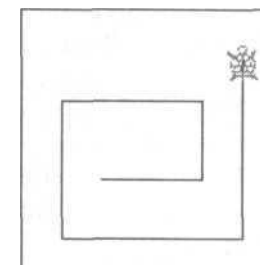
0 és $n-1$ közötti véletlenszámot állít elő az itt megadott teknőc (teknőcök) lesz aktív a zárójelbe tett utasítások az itt megadott teknőcnek szólnak

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 más-más 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

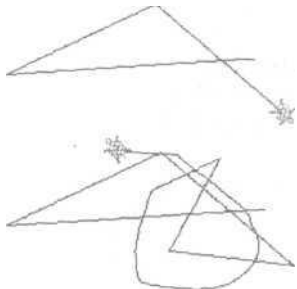
2. FELADAT: KÖVESSÜK AZ EGERET

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 függvény, amely az utolsó művelet végrehajtásakor 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

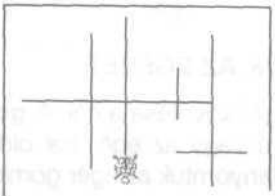
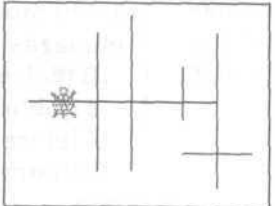
Rajzoljunk egy labirintust, amelyben vonalak jelzik az utakat, s a köztük levő festetlen pontok pedig a falakat. Egy olyan programot írunk, melynek segítségével a teknőcöt kivezethetjük a labirintusból.

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

\ labirintust jelképező útvesztő
:gy tetszőleges kezdő állapot-

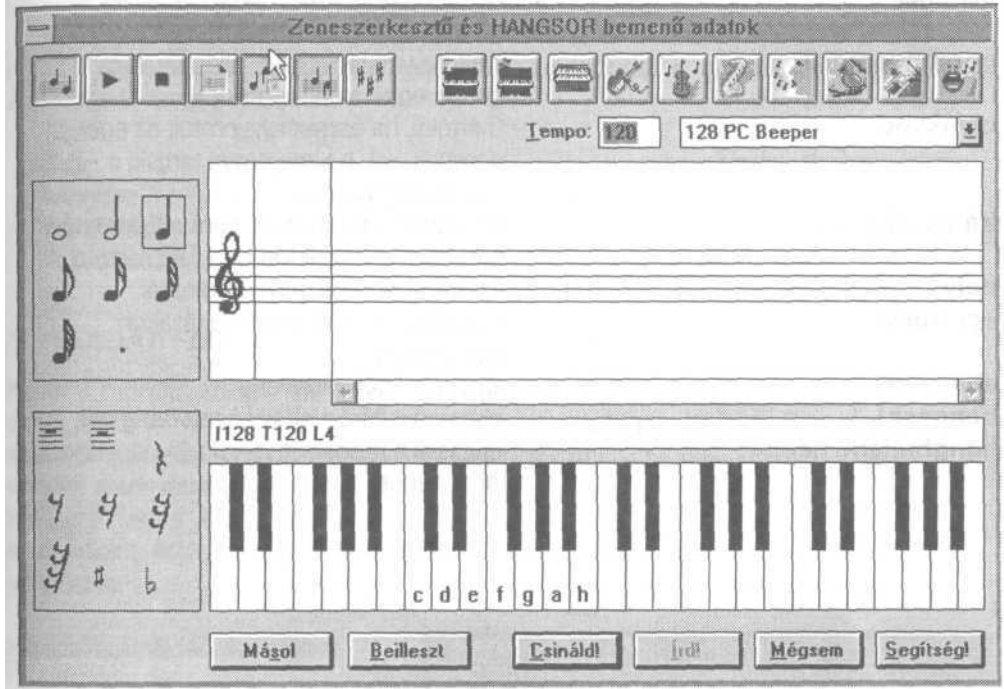


Igyanez célba éréskor:

HANGOK, ZENESZERKESZTÉS

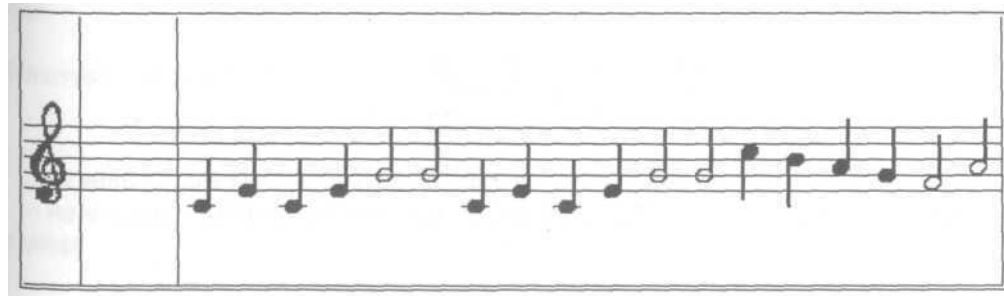
4. FELADAT: HANGSORSZERKESZTÉS A ZENEGÉPPEL

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 kompatibilis 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.



5. FELADAT: HANGHULLÁM LEJÁTSZÁSA

Előre felvett hanghullámok is lejátszhatók **hanghullám "jaj.wav** az alábbi módon:

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 egérgombját, - 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 egérhely	a teknőc aktuális helykoordinátái az utolsó művelet végrehajtásakor
abs szám	abszolút érték függvény
hangsor [...]	lejátssza a zárójelek közé tett hangsort
hanghullám "név"	lejátssza a megadott <i>nevű</i> állományban tárolt zenét

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 eredménykulcsszó 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 előbbieket 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 feladathoz, 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 tegyük, Mászhangzó 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 o ó ö ő u ú ü ű]
vége

...>; .. >i, :~/A . "

cv ...% . o'-. .

11. FELADAT: EGY MONDAT SZAVAI HANGRENDJÉNEK MEGADÁSA

A mondat feldolgozása a szavainak feldolgozását jelenti.

tanuld hangrend :mondat
ha üres? :mondat ~
[eredmény imondat]
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

üres? *sorozat*

eleme? -.*elem* -.*sorozat*
első -.*sorozat*
utolsó -.*sorozat*
elsőnélküli -.*sorozat*

utolsónélküli -.*sorozat*

elsőnek -.*elem* -.*sorozat*

utolsónak -.*elem* -.*sorozat*

a Logo függvények eredménye következik utána

igaz, ha a paramétere üres *sorozat*, hamis, ha nem

igaz, ha az *elem* eleme a *sorozatnak*

a *sorozat* első elemét adja

a *sorozat* utolsó elemét adja

a *sorozat* első elem nélküli részsorozatát adja

a *sorozat* utolsó elem nélküli részsorozatát adja

a *sorozat* első eleme elé szűrja be az új elemet

a *sorozat* utolsó eleme mögé szűrja 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 jogtisztta szoftverhasználat. Itt is igaz a mondás: legjobban 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.

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 alpműveltség 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 szövegekben, egyszerű mozgóképi egységekben. Szöveg- és kép-összeállítási elrendezési technikák.

FÜGGELÉK

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 tömegtájékoztató 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.

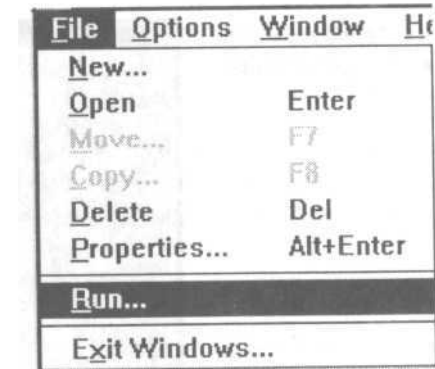
A COMENIUS LOGO PROGRAMCSOMAG INSTALLÁLÁSA

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

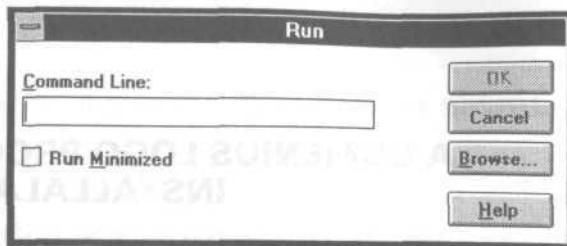
Kattintsunk a Programkezelő **File** menüpontjára!



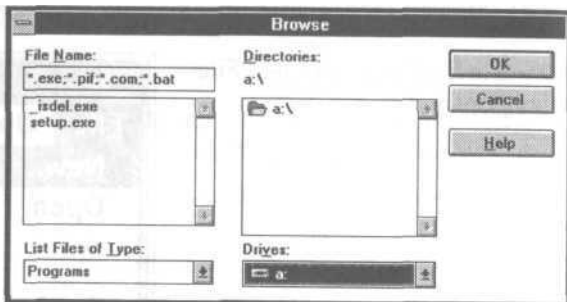
Kattintsunk a **Run** (Futtat) menüpontra!



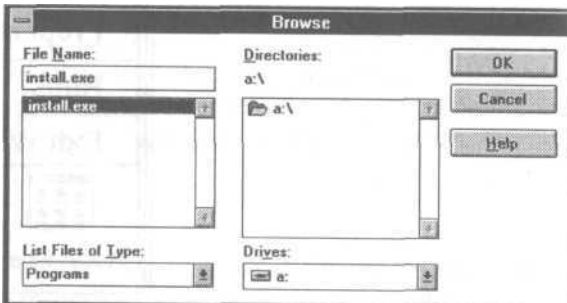
Kattintsunk a **Browse** (Tallóz) gombra!



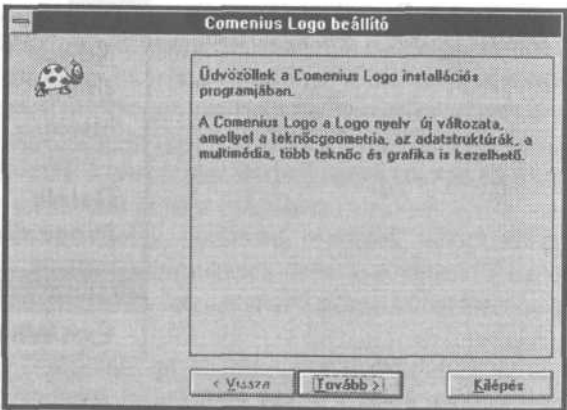
Válasszuk ki annak a meghajtónak a nevét, amelyikbe az installációs lemezt behelyeztük (ez lehet a: vagy b: vagy a winchester megfelelő könyvtára)!



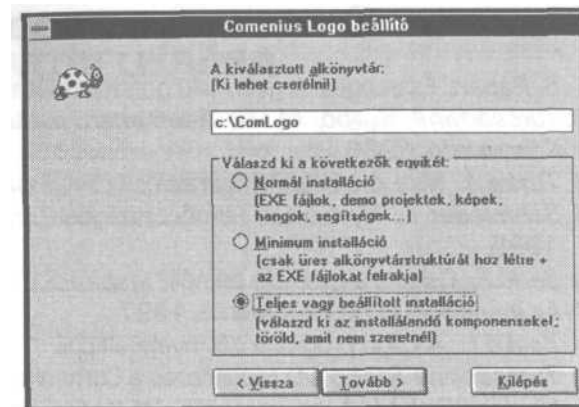
Kattintsunk az **install.exe** program nevére, majd az **OK** gombra! Kattintsunk megint az **OK** gombra!



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
j-HANG		a hangállományok
	-KÉPSOR	az LGW-állományok
I-PROJEKT	- ANIMOTOR	az Animotor program
	I-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 töltsse be a programot.

Példa:

BETÖLTRAJZLAP BETÖLTŐÚT "rajz.bmp

Ha ezt elmulasztjuk, akkor a következő hibaüzenetet kapjuk:

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

JAVASOLT IRODALOM

1. *S. Papért: Észrengés. A gyermeki gondolkodás titkos útjai.. SZÁMALK. 1988.*
 2. *Turcsányiné Szabó M.-D. Senkiében: A Logo programozási nyelv. Műszaki Könyvkiadó, 1986.*
 3. *Tuska Á.: Mire jó a Logo? Inspiráció, 1. évf. 6. szám, 22-23., 1993.*
- A. *Schmieder L: Nyelvújítás Teknőországban? 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 Lógóban.
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
(*paraméter 1 pára méter2...*)

Ekkor az eljárás az alábbi módon használható:

ELJÁRÁS paraméter
és
(*ELJÁRÁS*) ; *nincs paraméter*
(*ELJÁRÁS paraméter 1*) ; *egy paraméter*
(*ELJÁRÁS paraméter 1 paraméter2*) ; *kettő paraméter*
(*ELJÁRÁS paraméter1paraméter2...*) ; *akárhány paraméter*

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

paraméter 1 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
	<i>szám 7 ' szám 2</i> Hld: <i>pont 'szám</i> Hld: <i>szám 'pont</i> Hld: <i>ponti 'pont2</i>	<i>szám</i> <i>pont</i> <i>pont</i> <i>szám</i>	Eredménye a bemenő adatok szorzata.
+	<i>szám 1 + szám 2</i> <i>+szám</i> Hld: <i>pont 7 + pont2</i> Hld: <i>+pont</i>	<i>szám</i> <i>szám</i> <i>pont</i> <i>pont</i>	Eredménye a bemenő adatok összege.
	<i>szám 7 - szám 2</i> <i>-szám</i> <i>H\&.ponti -pont2</i> Hld: <i>-po«?</i>	<i>szám</i> <i>szám</i> <i>pont</i> <i>pont</i>	Eredménye a bemenő adatok különbsége.
/	<i>számi / szám 2</i> <i>H\&. szám /pont</i> <i>H\&. pont /szám</i>	<i>szám</i> <i>pont</i> <i>pont</i>	Eredménye a bemenő adatok hányadosa.
<	<i>szám 1 < szám 2</i> Hld: <i>szó 1 < szó2</i> Hld: <i>pont 1 < pont2</i>	<i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i>	IGAZ, ha a szám 1 kisebb, mint a szám2, egyébként pedig HAMIS.
<=	<i>szám! <=szám2</i> Hld: <i>szó 1 <= szó2</i> Hld: <i>ponti <=pont2</i>	<i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i>	IGAZ, ha a szám 1 kisebb vagy egyenlő, mint a szám2, különben HAMIS.
◇	<i>akármí 1 <> akármí2</i>	<i>logikai érték</i>	IGAZ, ha a két paraméter különbözik egymástól, egyébként HAMIS.
=	<i>szám 1 = szám2</i> <i>szó 1 = szó2</i> <i>lista 1 = lista2</i> <i>akármí7 = akármí2</i>	<i>logikai érték</i>	IGAZ, ha a két operandusa egyenlő, egyébként HAMIS.
>	<i>szám 1 > szám2</i> Hld: <i>szó 7 > szó2</i> Hld: <i>pont 1 > pont2</i>	<i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i>	IGAZ, ha a szám 1 nagyobb, mint a szám2, különben HAMIS.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
>=	<i>számi >=szám2</i> Hld: <i>szó 1 >= szó2</i> Hld: <i>ponti >=pont2</i>	<i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i>	IGAZ, ha a szám 1 nagyobb vagy egyenlő, mint a szám2. egyébként HAMIS.
ALAPSZAVAK	<i>szám. a következők valamelyike:</i>	<i>lista</i>	A Logo alapszavak listája: - angol és magyar alapszavak, - magyar alapszavak, - angol és magyar alapszavak rövidítése, - magyar alapszavak rövidítése.
PRIMITIVES	0- <i>1 - .</i> 2- 3-		
GYŰJTMEM	<i>nincs</i>	<i>nincs</i>	Általános hulladékgyűjtési folyamatot indít el, a felhasználatlan memória-területek újrahasznosításával.
CLEANMEM			
.LÉS			
AND			
LKVAGY			
.XOR			
1NEM			
.NOT			
1VAGY			
.OR			
PARAMÉTEREK			
.MEM			
.PONTHELY	<i>(pont)</i> <i>(11)</i>		Eredménye a <i>pont</i> koordinátája (Windows koordináta-rendszerben). A rajzlap bal felső sarkának abszolút koordinátáját adja.
WINDOWSPOS			

1 A,Á

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ABLAK	<i>nincs</i>		Az aktív teknőcök számára definiálja a felhasználható területet. A <i>pont</i> koordinátája és a rajzlap jobb alsó sarka által meghatározott terület használható. Az adott <i>téglalap</i> a felhasználható terület. Újrdefiniálja a felhasználható területet.
WINDOW	<i>(pont)</i> <i>(téglalap)</i> (ÍJ)		
ABSZ	<i>szám</i> Hid: <i>pont</i>	<i>szám</i> <i>szám</i>	Eredménye a bemenő szám abszolút értéke.
ABS			
ALAK	<i>nincs</i>	<i>képsor</i>	Eredménye az a képsor, ami pillanatnyilag az első teknőc alakja.
GETSHAPE			
ALAK!	(ÍJ) <i>képsor</i> <i>LGWJájl</i> Hid: "Vágólap"		Az aktív teknőcök aktuális alakja az alapalakot veszi fel, egyébként a <i>képsor</i> , illetve az <i>LGWJájl</i> tartalmát használja. A <i>Vágólap</i> aktuális tartalma lesz az aktív teknőcök új alakja.
SETSHAPE, SETTURTLE			
ALAKSZÍN	<i>nincs</i>	<i>szín</i> ÍJ	Eredménye az első aktív teknőc aktuális aktív alakszíne, illetve [], ha nem határoztunk meg aktív alakszínt.
SHAPECOLOR			
ALAKSZÍN!	<i>szín</i> []		Az aktív teknőcök aktív alakszíne <i>szín</i> lesz. Nem lesz aktív alakszín.
SETSHAPECOLOR			
ALAPHELYZET	<i>nincs</i>		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.
DRAW			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ALAPPALETTA	<i>nincs</i>		Abban az esetben használjuk, ha 256 színt alkalmazó módban dolgozunk, és egy betöltendő rajz ettől eltérő palettát használ.
RESETPALETTE, RESETPAL			
ALAPSZÓ?	<i>szó</i>	<i>logikai érték</i>	IGAZ, ha a szó egy Logo alapeljárás neve, különben HAMIS.
PRIMITIVE?			
AMÍG	<i>[kifejezés]</i> <i>utasításlista</i>		Amíg a <i>kifejezés</i> értéke igaz, az <i>utasításlista</i> összes utasítását újra és újra végrehajtja.
WHILE			
ANIMÁCIÓSMÓD!	<i>logikai érték</i>		Meghatározza azt a módot, ahogyan az aktív teknőcök alakja változtatható. Fázis animáció létrehozásához az IGAZ beállítást használjuk.
SETPHASEMODE			
ARCTG	<i>szám</i>	<i>szög</i>	Eredménye az a (fokokban mért) szög, amelynek tangense a paraméterként adott <i>szám</i> .
ARCTAN			
ÁLLAPOT	<i>nincs</i>	<i>[X Yszög szó 1 szó2 képsor]</i> ahol szó 1: TL, TF, TRD vagy TVT, és szó2. LÁTHATATLAN vagy LÁTHATÓ	Az első aktuális teknőc aktuális állapotát adja vissza.
TURTLESTATE			
ÁTDOB	<i>szó</i>		Leállítja az aktuális eljárást és az öt hívókat is mindaddig, amíg nem talál ELKAP szó <i>utasításlista</i> utasítást.
THROW			
ÁTÍR	<i>memória _objektum</i> <i>[memória _objektum</i> Hid: (<i>képsor_változó szám</i>)]		Megnyitja és aktívvá teszi a <i>memória _objektum</i> szerkesztőjét.
EDIT, ED			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ÁTÍRFÁJL	<i>fájl</i>		Kinyitja a Fájlszerkesztő ablakot és aktívá teszi. Ebben a módban a Logo lehetőséget biztosít megfelelő <i>szövegfájlok</i> szerkesztésére, illetve létrehozására.
EDITFILE, EDFILE			
ÁTLÁTSZÓMÓD!	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.
SETIMODE			

B			
BALRA. B	<i>szög</i>		<i>ki</i> összes aktív teknőc balra fordítása szöggel.
LEFT, LT			
BEÁLLÍTÓ	<i>Logo_eljárás</i> <i>Logo_eljárás</i>	<i>[Logo utasítás]</i> <i>[]</i>	A Beállítóablak megjelenítésével lehetővé teszi, hogy a felajánlott lehetőségek közül kiválasszuk az <i>eljáráshoz</i> tartozó megfelelő bemenő paramétereket.
CHOOSE			
BELÜL?	<i>téglalap</i> <i>pont téglalap</i> <i>teknőc téglalap</i>	<i>logikai érték</i>	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.
INSIDE?			
BETOLDKÉP	<i>képsor</i> <i>LGWJájl</i>		Minden aktív teknőc kinyomtatja a <i>képsor</i> -, illetve az <i>LGW_fájlban</i> tárolt képsor első fázisát.
PUT1IMAGE			
BETÖLT	<i>LGO_fájl</i> Hid: "Vágólap		A Logo fájlban tárolt parancsokat betölti és végrehajtja.
LOAD			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
BETÖLTFELADAT	<i>TXTJájl</i> (<i>TXTJájl</i> [<i>szám1</i> <i>szám2</i> <i>szám3</i> <i>szám4</i> <i>szám5</i>])		A TXT fájl tartalma által meghatározott Feladatablakot tölti be. <i>szám1</i> <i>szám2</i> az ablak bal felső sarkának absz. koordinátái. <i>szám3</i> a szövegszlopok száma; <i>szám4</i> a szövegsorok száma. <i>szám5=0</i> esetén az ablak mozgatható és méretezhető. <i>szám5=\</i> esetén az ablak mozgatható, de nem méretezhető. <i>szám5=2</i> esetén az ablak nem mozgatható, de méretezhető. <i>szám5=3</i> esetén az ablak nem mozgatható és nem is méretezhető.
LOADPROBLEM			
BETÖLTKÉPSOR	<i>LGWJájl</i> [] <i>LGWJájl</i> intervallum Hid: <i>LGWJájl</i> [<i>szegm1</i> <i>szegm2</i> ..] Hid: "Vágólap []" Hid: "Vágólap szegmens" Hid: "Vágólap [<i>szegm1</i> <i>szegm2</i> ..]	<i>képsor</i> <i>képsor</i> <i>képsor</i> <i>képsor</i> <i>képsor</i>	Az <i>LGWJájl</i> által meghatározott képsort tölti be. [<i>szegm1</i> <i>szegm2</i>] <i>szegm7</i> -edik fázisától <i>szegm2</i> db fázist tölt be. A megfelelő szegmensek által meghatározott fázisokat összevágva tölti be. A fentiekhez hasonlóan a vágólap tartalmát tölti be, ha az <i>LGW</i> szerkezetű.
LOADIMAGE			
BETÖLTKÖTET	<i>projekt kötet</i> Hid: "Vágólap		Az adott <i>projekt_kötet</i> ex, illetve a <i>Vágólap</i> ox tölti be, ha az projekt formátumú.
LOADLIBRARY, LOADLIB			
BETÖLTÓÚT	<i>szó</i> 0	<i>fájl</i> <i>útvonal</i>	Megadja a szobán lévő fájl kiterjesztését és elérési útvonalát. A legutóbb betöltött fájl útvonalát kapjuk meg.
LOADPATH			
BETÖLTRAJZ	<i>BMPJájl</i> Hid: "Vágólap	<i>képsor</i> <i>képsor</i>	Egyfázisú képpé alakítja a <i>BMPJájl</i> . illetve a <i>Vágólap</i> tartalmát, ha az BMP formátumú volt.
LOADBITMAP			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
BETÖLTRAJZLAP	<i>BMP_fájl</i> Hid: (<i>BMP_fájl pont</i>) <i>Hid:</i> (<i>BMPJájl téglalap</i>) Hid: <i>projekt_rajzlap</i> Hid: (<i>projekt_rajz/ap pont</i>) Hid: (<i>projekt_rajzlap téglalap</i>) Hid: "Vágólap <i>H\&</i> . ("Vágólap pont) Hid: ("Vágólap téglalap)	<i>kép</i> <i>kép</i> <i>kép</i>	Betölti a <i>BMPJájl</i> tartalmát a rajzlapra. A <i>pont</i> a kép bal felső sarkát határozza meg. A rajzlap <i>téglalapnyiterületére</i> tölti a képet.
LOADSCREEN			
BETŰ!	<i>szöveg [szám 1 szám2 szám3 szám4 szám5]</i> ahol <i>szám 7</i> nagyobb vagy egyenlő 0-nál, <i>szám2</i> [0.1000], <i>szám3</i> , <i>szám4</i> és <i>szám5</i> 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; <i>szám 7</i> a méret; <i>szám2</i> a kövérség; <i>szám3</i> =1 esetén dőlt, <i>szám3</i> =0 esetén nincs döntve; ha <i>szám4</i> nem adott, akkor a szöveg mögött nincs háttérszín; <i>szám5</i> =0 esetén a téglalap bal felső sarka, <i>szám5</i> =1 esetén pedig a középtől pozicionál.
SETTEXT. SETTT			
BETŰNÉGYZET	<i>szöveg</i>	[<i>szám 7 szám2 szám3 szám4</i>]	Az első aktuális teknőc által kiírt szöveg befoglaló négyzetét adja (<i>szám 7 szám2</i>)
TEXTSIZE			
BETŰZD	<i>szöveg</i> (<i>szöveg 1 szöveg2...</i>)		Minden aktuális teknőc kiírja a <i>szöveget</i> a BETŰ-ben beállított típussal, mérettel stb.
TTEXT.TT			
BEZÁRFELADAT	<i>TXTJájl</i>		A TXT fájl tartalma által meghatározott Feladatablakot zárja be.
CLOSEPROBLEM			
BEZÁRVIDEO	<i>szó</i>		Az utasítás az adott ablakkal kapcsolatban lévő videofájlt, valamint a szó nevű ablakot zárja be.
CLOSEVIDEO			

[C. CS

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
CIKLUS	<i>szó [N1 N2] utasítás lista</i>		TV HN2, ciklus szó=N 7-től 7N2-ig (N3 lépésközzel).
FOR	<i>szó [N1 N2 N3] utasításlista</i>		
CIKLUSOSAK	<i>szó akármi utasításlista</i>		A ciklus az <i>akármi</i> összes elemére végrehajtja az <i>utasításlistát</i> .
FOREACH			
COS	<i>szög</i>	<i>szám</i>	Eredménye a paraméterként adott (fokokban mért) szög koszinusza.
COS			
CSATOL	<i>LGOJájl</i> (<i>fájl obj2 obj3...j</i>) Hid: <i>LGOJájl obj2 obj3...</i> ahol <i>minden obj2. obj3... vagy memóriajbjektum vagy memória_objektumlista</i>		Az összes felsorolt objektumot egy megadott nevű fájlba menti. Ha ilyen nevű fájl már létezik, akkor a kijelölt objektumokat a fájl végéhez írja.
APPEND			
CSERE	<i>szám lista akármi</i> <i>szám szó 7 szó2</i> <i>szám képsor 1 képsor2</i> <i>intervallum lista 1 lista2</i> <i>intervallum szó 7 szó2</i> <i>intervallum képsor 7 képsor2</i>	<i>lista</i> <i>szó</i> <i>képsor</i> <i>lista</i> <i>szó</i> <i>képsor</i>	A második bemenet (<i>lista</i> , <i>szó 1,...</i>) <i>szám</i> elemét a harmadik bemenetre változtatja. Az <i>intervallum</i> által megadott elemeket kicseréli a <i>lista2</i> -ve\ a <i>lista</i> 7-ben.
REPLACE			
CSÖKKENT	<i>változónév</i> (<i>változónév szám</i>)		A <i>változónév</i> nevű változó értékét 1 -gyei, illetve <i>számmal</i> csökkenti.
DEC			

1 D,DZ, DZS

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

EGÉRALAK!	<i>képsor</i> <i>LGWJájszám</i> ahol <i>szám</i> egész [0. 15] IJ		Az egérkurzor aktuális alakját állítja be képsor alakúra. Különböző adott alakúra. Az eredeti alak visszaállítása.
SETMOUSECURSOR. SETMC			
EGÉRÁLLAPOT	<i>nincs</i> <i>nincs</i>	<i>[szám 1 szám 2 xyj szám 7 szám2 []]</i>	<i>szám 1</i> egész [0. 7] melyik gomb volt lenyomva; <i>szám2</i> egész [0, 3], le volt-e nyomva speciális billentyű (shift, ctrl): xy hely.
MOUSESTATE			
EGÉRHELY	<i>nincs</i>	<i>pont[xy]</i>	Az egér koordinátái.
MOUSE			
EGÉRHELY!	<i>pont</i>	<i>nincs</i>	A meghatározott pontba helyezi az egérkurzort.
.SETMOUSEPOS			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
EGÉSZHÁNYADOS. EH	<i>szám 1 szám2</i> Hld: <i>pont szám</i> Hld: <i>ponti pont2</i>	<i>szám</i> <i>pont</i> _____ <i>pont</i>	A <i>szám 1</i> és <i>szám2</i> egész hányadosa.
DIV			
EGÉSZRÉSZ	<i>szám</i> Hld: <i>pont</i>	<i>szám</i> <i>pont</i>	A paraméterénél nem nagyobb, legnagyobb egész szám.
INT			
EGYENLŐ?	<i>akármí 1 akármí2</i> (<i>akármí!...</i>)	<i>logikai érték</i> <i>logikai érték</i>	IGAZ, ha a két paraméter azonos szó (szövegkonstans vagy szám), azonos képsor vagy lista, egyébként HAMIS.
EQUAL?			
EGYMÁSUTÁN	<i>Logo_utasításlista</i>		Az összes aktív teknőc egymás után végrehajtja az <i>utasításlistát</i> .
EACH			
ELÁGAZÁS	<i>akármí [ág 1 ág2...]</i>		Azt az ágat hajtja végre vagy értékeli ki, amelynek <i>feltétele</i> megegyezik <i>akármí</i> értékével.
CASE	<i>akármí [ág 1 ág2... J</i>	<i>akármí</i>	
ELEM	<i>szám akármí</i> <i>intervallum akármí</i> Hld: [<i>szegm 1 szegm2</i> ...] <i>akármí</i>	<i>akármí</i> <i>akármí</i> <i>akármí</i>	Az <i>akármí</i> szám elemét adja. Az <i>akármí</i> intervallum által meghatározott része.
ITEM			
ELEME	<i>szó 1 szó2</i> <i>akármí lista</i>	<i>sorszám,</i> <i>semmi</i>	A második paraméterében keresi az elsőt, ha megtalálható benne, akkor a sorszámát adja eredményül.
MEMBER			
ELEME?	<i>akármí lista</i> <i>szó 1 szó2</i>	<i>logikai érték</i> <i>logikai érték</i>	Eldönti, hogy a második paraméterében megtalálható-e az első.
MEMBER?			
ELEMNÉLKÜLI	<i>karakter szó</i> <i>akármí lista</i>	<i>szó</i> <i>lista</i>	Az a sorozat, amelyik a második paraméter elemeit tartalmazza, elhagyva belőle az első paraméter összes előfordulását.
BUTMEMBER			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ELEMSZÁM, ESZ	<i>akárm</i>	<i>szám</i>	Az <i>akárm</i> elemeinek száma.
COUNT			
ELEMTŐL	<i>szól szó2</i> <i>akárm lista</i>	<i>szó</i> <i>lista</i>	A második paraméter azon elemeit tartalmazó sorozat, amelyik az első paraméterként megadott érték mögött található.
FROMMEMBER			
ELFED	<i>memória_objektum</i> <i>memória „objektum_lista</i> HId: (<i>paraméter 1</i> <i>paramter2...</i>) ahol minden paraméter vagy memória_objektum vagy memória_objektumjista		Láthatatlanná teszi a paraméterként megadott, felhasználó által definiált <i>memória_objektumokat</i> amelyek ezután fedettek lesznek.
BURY			
ELFEDMIND	<i>nincs</i>		Láthatatlanná teszi a felhasználó által definiált összes <i>memória_objektumot</i> .
BURYALL			
ELFELEJT	<i>akárm</i> (<i>akárm1? akárm2...j</i>)		A kapott paramétereket nem hajtja végre, elfelejti.
IGNORE			
ELKAP	<i>szó Logo_utasítás!lista</i>		Balról jobbra haladva kiértékeli a paraméterként kapott Logo <i>utasításlistát</i> . Amikor ebben az ÁTDOB utasításhoz jut, mely a szíval együtt szerepel, akkor a vezérlés újra az ELKAP eljáráshoz kerül, és az ezt követő kifejezések kiértékelése történik meg.
CATCH			
ELŐRE, E	<i>szám</i>		Az aktuális technőcöket <i>szám</i> lépéssel előre viszi.
FORWARD, FD			
ELSŐ	<i>akárm</i> (nem üres)	<i>akárm</i>	Eredménye a paraméter első eleme.
FIRST			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
ELSŐNEK	<i>akárm lista</i> <i>képsor 1 képsor2</i> <i>szó 1 szó2</i>	<i>lista</i> <i>képsor</i> <i>szó</i>	Eredménye az a sorozat, amely úgy keletkezik, hogy az első paramétert a második paraméter elejére beszúrja.
FPUT			
ELSŐNÉLKÜLI, EN	<i>akárm</i> (nem üres)	<i>akárm</i>	Eredménye az eredeti paraméter első elemének elhagyásával kapott sorozat.
BUTFIRST, BF			
EREDMÉNY, ER	<i>akárm</i>		A pillanatnyilag futó függvény értékét adja meg az <i>akárm</i> ben.
OUTPUT. OP			
ÉRZÉKENYTERÜLET	speciális szó		Egy mező, névvel ellátott téglalap alakú terület a rajlapon, ahol az egeret kezelni lehet. Az ÉRZÉKENYTERÜLET szó definiálja azokat a mezőket, melyek a fenti téglalapok jellemzőit és azok kódjait tartalmazzák.
CODEAREA			
ÉS	<i>logikai érték logikai érték</i> (<i>logikai érték...j</i>)	<i>logikai érték</i> <i>logikai érték</i>	IGAZ, ha minden bemenő adat logikai értéke IGAZ, minden más esetben HAMIS.
AND			

F

FÁJL'	<i>szó</i>	<i>logikai érték</i>	IGAZ, ha a fájl létezik; ha nem adunk alkönyvtárát akkor a munkakönyvtárban keres; kiterjesztés nélkül LGP lesz.
FILE?			
FÁJLVÉGE	<i>nincs</i>	<i>logikai érték</i>	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.
EOF?			
FÁZIS	<i>nincs</i>	<i>szám</i> ([0,255])	Az első aktív technőc fázisát adja (akkor is, ha nem látszik).
PHASE			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
FÁZIS!	<i>szám</i>		Az aktív teknőcök fázisát állítja be mod N-re, ha N fázisból áll a teknőc és az animációmód igaz.
SETPHASE	<i>egész</i> [-32767, 32767]		
FÁZISPONT	<i>nincs</i>	<i>téglalap</i>	Megadja az első aktív teknőc által lefedett területet [x y szélesség magasság] formában. Ha nem látszik, akkor [x y 0 0],
SHAPEPOS			
FED!	<i>logikai érték</i>		A lefedett objektumokat láthatóvá teszi, vagy elrejt.
SETBURY			
FEDETT	<i>nincs</i>	<i>fedett_objektumokjís-tája</i>	Eredménye az összes fedett objektumot tartalmazó lista.
BURIED			
FELFED	<i>memória_objektum</i>		Meggzűnteti az objektumok fedett tulajdonságát.
UNBURY	<i>memória_objektum</i> Jista Hld: (paraméter1 paraméter2...j ahol minden paramé- ter vagy memó- na_objektum vagy memória ^objektum Jista		
FELSŐSZINT	speciális szó		Használata: ÁTDOB "FELSŐSZINT" Kilép a programból.
TOplevel			
FIGYEU	<i>teknőc_iev</i>		Az adott teknőcöket aktívá teszi.
TELL	<i>teknőc_név_lista</i>		
FORDÍT	<i>akármí</i>	<i>akármí</i>	Megfordítja paramétere elemei sorrendjét.
.REVERSE			

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

G, GY

GENERÁL	<i>szám eljárásobjek- tum2 akármí</i>	<i>lista</i>	Egy rekurzív listát generál az <i>akármí</i> kezdőértékkel és az <i>eljárásobjek- tum2</i> eljárással addig, amíg a lista <i>szám</i> elemű lesz vagy a következő elemre <i>eljárásobjektum 7</i> HAMIS lesz.
GENERATE	<i>eljárásobjektum 1</i> <i>eljárásobjektum2</i> <i>akármí</i>	<i>lista</i>	
GOMBNÉV	<i>nincs</i>	<i>lista</i>	Vagy a Gombok ablak gombjaihoz kapcsolódó aktuális Gombsor-defini- ciót tartalmazó lista, vagy a <i>szám</i> gombhoz tartozó aktuális gombdefini- ció.
BUTTONS	(<i>szám</i>)	<i>lista</i>	
GOMBNÉV!	<i>gombsor_definíció_lista</i>	<i>beállított gombok</i>	Gombokhoz kapcsolódó feliratot/grafikát és viselkedést határoz meg.
SETBUTTONS.	(<i>szám gomb- sor_definíció</i>)		
GOMBNYOMÁS?. GNY?	<i>nincs</i>	<i>logikai érték</i>	Eredménye IGAZ, ha lenyomtuk valamelyik billentyűt vagy a bal egér- gombot.
KEY? RC?			
GOMBSOR	<i>nincs</i>	<i>[szám 1 szám2 szám3</i>	A Gombok ablak aktuális pozíciója és mérete, kiegészítve esetleg ennek elnevezésével.
BUTTONSSIZE. BUTTONSIZE	<i>nincs</i>	<i>szám4]</i> <i>[szám 1 szám2 szám3</i> <i>szám4 szöveg]</i>	
GOMBSOR!	<i>[XYwhJ</i>	<i>lista 4 vagy 5 elemű</i>	A Gombok ablak aktuális pozícióját és méretét, esetleg elnevezését állítja be.
SETBUTTONSSIZE, SETBUTTONSIZE	<i>[XYwhfJJ</i> <i>[16</i>	<i>(Gombok ablak akt. poz.)</i>	

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
GYÖK	<i>szám</i> ahol <i>szám</i> >0	<i>szám</i>	Nem negatív <i>szám</i> négyzetgyöke.
SQRT			

H

HA	<i>logikai érték</i>	<i>Logo utasításlista</i>	Ha az első paramétere IGAZ, akkor az első utasításlistát kell végrehajtani, ha HAMIS, akkor a másodikat (ha van).
IF	<i>Logo utasítás_lista</i> <i>logikai érték</i> <i>Logo utasításjista 1</i> <i>Logo utasítás_lista2</i> <i>logikai érték [kif1]</i> <i>[kif2]</i>	<i>akármí</i>	
HAHAMIS, HAH	<i>Logo utasítás_lista</i>	<i>akármí</i> <i>utasításjista</i>	Végrehajtja az utasításlistát, ha az aktuális eljárás utolsó TESZT utasítása HAMIS eredményű volt.
IFFALSE, IFF			
HAIGAZ, HAI	<i>Logo utasítás_lista</i>	<i>akármí</i> <i>utasításjista</i>	Végrehajtja az utasításlistát, ha az aktuális eljárás utolsó TESZT utasítása IGAZ eredményű volt.
IFTRUE, IFT			
HAMIS	speciális szó		HAMIS logikai érték.
FALSÉ			
HANG	<i>szám_lista</i> <i>(szám_lista 1</i> <i>szám_/ista2...j</i>	<i>hang, dallam</i>	A <i>számjista</i> által megadott hangsorozatot játssza le.
SOUND			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
HANGHULLÁM, HANGH	<i>WAV fájl</i> <i>(WAV fájl1 WAV fájl2</i> ... <i>[WAV fájl1 WAV fájl2</i> <i>(WAV fájl1 WAV fájl2</i> ... <i>[]</i> <i>(WAV fájl1 WAV fájl2</i> ... <i>•)</i>	<i>hang, dallam</i>	A <i>WA VJáj/ban</i> megadott dallamot játssza le. (Hangkártya szükséges!)
PLAYWAVE, PLAYW			
HANGSOR	<i>lista</i> <i>(lista 1 lista2...)</i>	<i>zene</i>	A számítógép beépített hangszóróján lejátszza a <i>listában</i> megadott hangjegyeket.
PLAY			
HASZNÁL	<i>eljárásobjektum lista</i>	<i>akármí</i>	Az <i>eljárást</i> meghívja a listában szereplő paraméterekkel.
APPLY			
HATVÁNY	<i>szám</i>	<i>szám</i>	Exponenciális függvény.
EXP			
HAZA	<i>nincs</i>	<i>művelet</i>	Az összes aktív teknőcöt az alapkoordinátáira viszi. Az alapkoordinátákat az ÚJTEKNŐC eljárással definiáljuk.
HOME			
HÁNYADIK	<i>nincs</i>	<i>szám</i>	Eddig hányszor hajtottuk végre az ismétlést. (Csak ISMÉTLÉS, CIKLUSAMÍG, CIKLUSOSAK utasítások belsejében használható.)
REPCOUNT, REPC			
HÁNYADOS	<i>szám 1 szám2</i> Hld: <i>pont szám</i> Hld: <i>szög pont</i>	<i>szám</i> <i>pont</i> <i>pont</i>	A <i>szám 1</i> és <i>szám2</i> hányadosa.
QUOTIENT			
HÁTRA, H	<i>szám</i>		Az összes aktív teknőcöt a paraméterként adott <i>számú</i> lépéssel (azaz képponttal), az irányukkal ellentétes irányban mozgatja.
BACK, BK			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
HELY	<i>nincs</i>	<i>pont</i>	Az első aktív teknőc aktuális [X Y] koordinátái. (A koordináta-rendszer origója a rajzlap középpontjában van.)
POS			
HELY!	<i>pont</i>		Az összes aktív teknőcöt az adott <i>pontba</i> viszi, de aktuális irányukat változatlanul hagyja. TOLLATLE állapotban az eredeti és új pozíciót vonallal köti össze.
SETPOS			
HIBA	<i>nincs</i> mint speciális szó: <i>ELKAP "HIBA lista</i>	<i>lista</i>	Függvényként teljes hibaüzenetet eredményez, ha a számítás alatt hiba lép fel. Ha nem következik be hiba, akkor az eredmény []. Bármilyen, a <i>lista</i> 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.
ERROR			
HIBAÜZENET	<i>lista akármí</i>		Az eljárás alkalmazása esetén leáll a program végrehajtása, és saját hibaüzenetet küldhetünk. A hiba üzenetét a <i>listában</i> (a HIBAÜZENET eljárás első paramétereiként) adhatjuk meg.
MAKEERROR			

I,Í

IDŐ	<i>nincs</i>	<i>[óra perc másodperc százdjvasodperc]</i>	Eredménye egy négy számból álló lista: az aktuális idő.
TIME			
IDŐSZÓ	<i>nincs</i> Hid: <i>(időjormátum)</i> Hid: <i>(időjormátum idő)</i>	<i>szó szó szó</i>	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 paramétereiként adhatunk egy lehetséges <i>időjormátumol</i> (egy listát), ami meghatározza azt a formát, melynek megfelelően módosítva kapjuk eredményül a <i>szót</i> . Két paraméter esetén a szóban az <i>idő</i> az <i>idő_formátumnak</i> megfelelően kapjuk vissza.
TIMEWORD, TIMEW			
IGAZ	— speciális szó —		Az IGAZ logikai érték.
TRUE			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
IRÁNY	<i>nincs</i>	<i>szög</i>	Eredménye az első aktív teknőc aktuális iránya. (Az északi iránytól az óramutató járásával megegyező irányban mért szög.)
HEADING			
IRÁNY!	<i>szög</i>		Elfordítja az aktív teknőcöket úgy, hogy új irányuk megegyezzen az adott szöggel (az aktuális irányuktól függetlenül).
SETHEADING, SETH			
IRÁNYSZÖG	<i>pont teknőc</i>	<i>szög</i>	Eredménye az az irány, melybe az első aktív teknőcnek kellene fordulnia ahhoz, hogy az adott <i>pont</i> felé nézzen. Ha a paraméter egy <i>teknőc</i> , akkor a függvény eredménye az az irány, melybe az első aktív teknőcnek fordulnia kellene, hogy az adott teknőc felé nézzen. (Jegyezzük meg, hogy az IRÁNSZÖG függvény eredménye nem az a szög, amellyel a teknőcnek balra vagy jobbra kellene fordulnia, hogy az adott pontra nézzen.)
TOWARDS			
ISMÉTLÉS, ISM.	<i>szám</i> <i>Logo_utasításjista</i>		<i>számszor egymás után végrehajtja a Logojtisztásjistát.</i> (Ha <i>szám</i> < 1, akkor egyszer sem hajtja végre.)
REPEAT			
ÍRÓABLAK, ÍA	<i>nincs</i>		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..
TEXTSCREEN, TS			
ÍRÓLAPABLAK	<i>nincs</i>	<i>szám</i> ([2,25] egész)	OSZTOTTABLAK módban az íróablakban pillanatnyilag megjelenő sorok száma. Ez a <i>szám</i> mindig egy 2 és 25 közé eső egész.
GETTS			
ÍRÓLAPABLAK!	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.
SETTS			

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

J

JEL	szám (egész[0.255])	karakter	A szám ASCII kódú karakter.
CHAR			
JOBBRA. J	szög		Az összes aktív teknőcöt elforgatja jobbra szöggel.
RIGHT. RT			

K

KERÉKÍT	szám Hid: pont	szám pont	Kerekítés. Mind a két koordinátát kerekíti.
ROUND			
KÉPSOR	képsor 7 képsor2 (képsor 1...) Hid: képsor kép- sorjoperátor Hid: (képsor kép- sor_operátor_kifejezés...j	képsor képsor képsor vagy semmi képsor vagy semmi	Összefűzött képsor.
IMAGE			
KÉPSOR?	akárm	logikai érték	IGAZ, ha a paramétere képsor.
IMAGE?			
KÉPSORMÉRET	képsor	[szám 7 szám2]	Az első fázis szélessége, illetve magassága képpontokban.
IMAGESIZE			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
KÉREM	teknőc Logojtasltásjista teknőcjista L ogo_utasítás_lis ta teknőc [kifejezés] teknőcjista [kifejezés]	akárm akárm	A teknőc, illetve teknőcök, ideiglenesen aktivizálva, párhuzamosan hajtják végre az utasításjistát. A teknőc, illetve teknőcök ideiglenesen aktívá válnak, és a rendszer kiértékeli a kifejezést, amely az első teknőccel kapcsolatos eredményt adja.
ASK			
KIFIGYEL	nincs	teknőc_név teknőc-névjista	Az összes aktív teknőcöt tartalmazó lista.
WHO			
KIHASÍTKÉP	[szám 7 szám2 szám3 szám 4]	képsor	Egyetlen fázisból álló képsor: az első aktív teknőc készíti a rajzlapon öt körülvevő pillanatnyi környezetről.
GETIMAGE			
KÍÍR	akárm (akárm 1 akár mi2... J	szöveg	?KÍÍR "SZÉP SZÉP
PRINT, PR			
KÍÍRBELSŐ	akárm (akárm 1 akárm2... j	szöveg?	Az utolsó kiírt karakter mögött megjelenik a parancsjel. Például: ?KÍÍR "SZÉP SZÉP?
TYPE			
KÍÍRJEL, KI	akárm (akárm 1 akárm2...)		Az esetleges külső zárójelet is kiírja.
SHOW, SH			
KÍÍRÓESZKÖZ	fájl név []		Az aktuális kimenet a fájlnev nevű állomány lesz.
PRINTTO			
KÍÍRSAJÁTSZAVAK	nincs		Az írólapra írja a saját eljárások címsorát.
POTS			
KÍÍRSOR, KS	akárm (akárm 1 akárm2... j	akárm 7 akárm2	Minden egyes paraméterét külön sorban írja ki.
IF-			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
KÍRTARTALOM, KT	<i>memória_objektum</i> <i>memória_objektum_lista</i> Hld: (<i>paraméter 1</i> <i>paraméter2...</i>) ahol <i>paraméter me-</i> <i>mória_objektum</i> vagy <i>memória_objektum_lista</i>		Az írólapra írja a paraméterként adott objektumok definícióit.
PO			
KISEBB?	<i>szám 1 szám2</i> (<i>szám 7...j</i>) Hld: <i>szó 1 szó2</i> Hld: (<i>szó 1...</i>) Hld: <i>ponti pont2</i> Hld: (<i>pont 1...j</i>)	<i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i>	IGAZ, ha <i>szám 1 < szám2</i> . különben HAMIS.
LESS?			
KIVÁLASZT	<i>akármí, de nem üres</i>	<i>szó->karakter</i> <i>lista->listaelem</i> <i>képsor->fázis</i>	Paraméterének egy véletlenszerűen kiválasztott eleme.
PICK			
KÍVÜL?	<i>téglalap</i>	<i>logikai érték</i>	IGAZ, ha az első aktív teknőc az adott <i>téglalapon</i> kívül helyezkedik el.
OUTSIDE?			
KÓD	<i>karakter</i> <i>szó</i>	<i>szám</i> <i>szám [0, 255] közötti</i> <i>egész</i>	A bemenő <i>karakter</i> vagy a bemenő <i>szó</i> első karaktere ASCII kódja.
ASCII			
KÖRKÖRÖSEN	<i>nincs</i> (<i>pont</i>) (<i>téglalap</i>) (IJ)	<i>nincs</i>	A képernyő szélének kezelési módja az aktív teknőcök számára.
WRAP			
KURZORHELY	<i>nincs</i>	<i>[oszlop sor]</i>	A kurzor jelenlegi helyzete az írólapon.
CURSOR			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
KURZORHELY!	<i>[szám 7 szám2]</i> ahol <i>szám 1</i> egész [1, 80], <i>szám2</i> egész [1, 26]		Kiteszi az írólap aktuális parancsjelét a <i>szám 7 szám2</i> pozícióba és mellette megjeleníti a kurzort is.
SETCURSOR			
KÜLÖNBSÉG	<i>szám 7 szám 2</i> Hld: <i>pont 7 pont2</i>	<i>szám</i> <i>pont</i>	Eredménye <i>szám 7</i> és <i>szám2</i> különbsége.
DIFFERENCE			

L,LY

LÁTHATATLAN	<i>nincs</i>		Eltünteti az összes aktív teknőcöt a képernyőről.
HIDETURTLE. HT			
LÁTHATÓ	<i>nincs</i>		Az összes aktív teknőc alakja megjelenik a képernyőn.
SHOWTEKNŐC, ST			
LÁTHATÓ?	<i>nincs</i>	<i>logikai érték</i>	IGAZ, ha az első aktív teknőc aktuálisan látható a képernyőn.
SHOWN?			
LEKÉPEZ	<i>eljárás objektum</i> <i>listái</i> •		Az összes lista első elemét egyetlen listába gyűjti, és az <i>eljárás_objektum</i> ezt a listát alkalmazza. A második elemekkel is ugyanezt csinálja és így tovább.
MAP			
LENYOMAT	<i>nincs</i>		Az összes aktív teknőc kinyomtatja az alakját.
STAMP			
LISTA	<i>akármí 1 akármí2</i> (<i>akármí!...j</i>)	<i>lista</i> <i>lista</i>	A paramétereiből álló sorozatot (listát) adja eredményül.
LIST			

Tejjes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
LISTA?	<i>akármí</i>	<i>logikai érték</i>	IGAZ, ha a paramétere lista.
LIST?			
LN	<i>szám 7 (pozitív)</i>	<i>szám2</i>	$\lfloor n(\text{szám } J) = \text{szám}2$
LOGO	<p><i>kulcs_szó</i> ahol <i>kulcs_szó</i> a következők valamelyike: ERROR, VERSION, HELP, AUTOPATH, RIGHT-CLICKS, STEPPING, DECIMAL, PRINT, SAVE, BEEPvagySOUND Hid: további opciók <i>for key szó:</i> :PICKER, HIGHUGHT, INDENT, HIDEPROBLEMS, CASE, OCTAVES, AXCOLOR-BITS, KEEPPALETTE, PALETTE, OMPRESSBMP.AVE16, MAXIMAGECOLOUR-BITS</p>	<i>szó vagy lista</i>	Eredményül az adott szóhoz tartozó aktuális beállítását kapjuk meg.

Tejjes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
LOGO!	<p><i>[key_szó 7 akármí 1 key_szó2 akármí2... J</i> ahol <i>key_szó</i> a következők valamelyike: ERROR, VERSION, HELP, AUTOPATH, RIGHT-CLICKS, STEPPING, DECIMAL, PRINT, SAVÉ, BEEPvagySOUND Hid: további opciók a <i>kulcs szóhoz:</i> :PICKER, HIGHLIGHT, INDENT, HIDEPROBLEMS, CASE, OCTAVES, AXCOLOR-BITS, KEEPPALETTE, PALETTE, OMPRESS-BMP, AVE16, MAX-IMAGECOLOURBITS</p>	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.
SETLOGO			
LOGOABLAK	<i>nincs</i>	<i>[XYwh szám szöveg]</i> ahol X. Y.w ésh számok	Kiírja a Logo főablak jellemzőit.
LOGOSIZE			
LOGOABLAK!	<p><i>[XYwh]</i> <i>[XYwhszám]</i> <i>[XYwh szám szöveg]</i> <i>ÍJ</i> ahol X Y.w ésh száms</p>		A Logo Fő ablakának pozícióját, méretét és feliratát definiálja.
SETLOGOSIZE			
LOKÁLIS	<p><i>szó</i> <i>szólista</i> <i>(szó 7...)</i></p>		Az adott eljáráshoz tartozó lokális változót hoz létre <i>szó</i> vagy <i>szólista</i> nevekkkel.
LOCAL			
LOKÁLISNÉV			Értéket is ad a lokális változónak, a névadással egyben.
LET			

M			
Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
MARADÉK, MA	<i>szám1 szám2</i> (szám2 nem 0) Hld: <i>pont szám</i> Hld: <i>pont1 pont2</i> ahol <i>szám</i> nem 0 és <i>pont2 = [X Y]</i> nem tartalmaz 0-t	<i>szám</i> (nem negatív egész) <i>pont</i> <i>pont</i>	A <i>szám1</i> és <i>szám2</i> osztási maradéka.
MOD, REMAINDER			
MCI	<i>mci_lista</i>		
MENT	<i>LGO_fájl</i> <i>LGO_fájl paraméter1...</i> (LGO_fájl paraméter1...) Hld: "Vágólap Hld: "Vágólap paraméter1... Hld: ("Vágólap paraméter1...) ahol <i>paraméter1</i> ,... vagy a <i>memória_objektum</i> vagy <i>memória_objektum_lista</i>		Az összes paraméterként szereplő objektumot az adott fájlba menti.
SAVÉ			
MENTKÉPSOR	<i>LGW_fájl képsor</i> <i>LGW_fájl képsor1 képsor2...</i> (LGW_fájl képsor1... j Hld: "Vágólap képsor Hld: "Vágólap képsor1 képsor2... Hld: ("Vágólap képsori...))	<i>fájl</i>	A paraméterként szereplő képsor(oka)t menti az adott fájlba.
SAVEIMAGE		<i>Vágólap</i>	A képsort a Vágólapra helyezi.

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
MENTKÖTET	<i>library</i> <i>libraryparaméter1...</i> (libraryparaméter1...) Hld: "Vágólap Hld: "Vágólap paraméter1... Hld: ("Vágólap paraméter1...) ahol <i>paraméter1</i> ,... vagy <i>memória_objektum</i> vagy <i>memória_objektum_lista</i>	<i>projektkötet</i>	Elmenti a pillanatnyilag létező technőcöket és beállításukat, valamint az éppen nem fedett memóriaobjektumokat. Lehetőség van arra is, hogy konkrétan felsoroljuk az elmentendő memóriaobjektumokat (<i>paraméter1...</i>)
SAVEUBRARY. SAVELIB		<i>Vágólap</i>	Vágólapra menti az adott kötet tartalmát.
MENTRAJZ	<i>BMP_fájl képsor</i> Hld: "Vágólap képsor	<i>fájl</i> <i>Vágólap</i>	Adott képsor első fázisát egyszerű rajzként fájlba menti. A képsor első fázisát a Vágólapra helyezi.
SAVEBITMAP			
MENTRAJZLAP	<i>BMP_fájl</i> (BMP_fájl pont) (BMP_fájl rctszög) Hld: <i>projekt_rajzlap</i> Hld: (projekt_rajzlap pont) Hld: (projekt_rajzlap téglalap) Hld: "Vágólap Hld: ("Vágólap pont) Hld: ("Vágólap téglalap)	<i>fájl</i> <i>fájl</i> <i>Vágólap</i>	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.
SAVESCREEN			
MEZŐ	<i>szó1 szó2</i>	<i>akármí</i>	Megkeresi azon mezőket, amelyeket <i>szó1</i> -re definiáltunk és eredményül a <i>szó2</i> által meghatározott mező értékét adja.
GPROP			
MEZŐ!	<i>szó1 szó2 akármí</i>	<i>akármí</i>	<i>szó1</i> -hez kapcsol egy <i>szó2</i> nevű mezőt, melynek értéke <i>akármí</i> tesz.
PPROP			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
MEZŐK	szó	<i>mezőjista</i>	szóval kapcsolatban definiált mezők listája.
PLIST			
MEZŐK!	szó <i>mezőjista</i>	<i>újmezők</i>	szóhoz definiálja a <i>mezőjista</i> -ban megadott mezőket.
SETPLIST			
MIND	<i>nincs</i>	<i>teknőcjista</i>	A jelenleg létező teknőcök nevének listája.
ALLTURTLES. ALL			
MONDAT, M	<i>akárm1 akárm2</i> (<i>akárm1...j</i>)	<i>lista</i> <i>lista</i>	A paraméterként adott szavakból, illetve listák elemeiből egyetlen listát hoz létre.
SENTENCE, SE			
MUTATGOMBSOR, MG	<i>nincs</i>	<i>ablak</i>	Megjeleníti a Gombok ablakot a képernyőn.
SHOWBUTTONS			
MUTATIKONSOR, Mi	<i>nincs</i>	<i>ikonsor</i>	Megjeleníti az ikonsort a képernyőn.
SHOWSPEEDBAR			

N, NY

NAGYOBB?	<i>szám1 szám2</i> (<i>szám1...</i>)	<i>logikai érték</i>	IGAZ, ha <i>szám1 > szám2</i> , különben HAMIS.
GREATER?	Hld: <i>szó1 szó2</i> Hld: (<i>szó1...</i>) Hld: <i>pont1 pont2</i> Hld: (<i>pont1...</i>)	<i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i> <i>logikai érték</i>	IGAZ, ha <i>szám1</i> után csökkenő sorozat következik. IGAZ, ha szó <i>l > szó2</i> ábécé-sorrendben IGAZ, ha a <i>pont1</i> megfelelő koordinátái <i>> pont2</i> megfelelő koordinátái.
NEM	<i>logikai érték</i>	<i>logikai érték</i>	Logikai tagadás művelet.
NOT			

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

NYOMTATMEMÓRIA, NYM	<i>paraméter 1 paraméter 1 memó- riajobektum 7 me- mória _objektum2 ... paraméter 1 memó- ria _objektumjista ahol paraméter 1 is vagy logikai érték vagy []</i>		naiasa
PRINTMEMORY. PRINTM			A Memória ablak tartalmát nyomtatja. Megadott objektumokat, mindent, rejtett objektumokat { <i>paraméter 7</i> ="HAMIS").
NYOMTATRAJZLAP	<i>logikai érték IJ</i>		Rajzlap aktuális tartalmát nyomtatja.
PRINTGRAPHICS. PRINTG			

O.Ö.Ő

OLVASJEL, OJ	<i>nincs</i>	<i>karakter</i>	Egy karaktert olvas.
READCHAR, RC			
OLVASKÓD. OK	<i>nincs</i>	<i>szám (ASCII kód)</i>	Egy karakter kódját olvassa.
READKEY			
OLVASUSTA. OL	<i>nincs</i>	<i>lista</i>	ENTER lenyomásáig a karaktersorozatot listába fűzi.
READLISTA, RL			
OLVASLISTAPARANCS JEL!. OLPARANCSJEL!	<i>szöveg</i>		A listaolvasás parancsjelét változtatja meg az írólapon.
SETREADLISTAPROMP T. SETRLPROMPT			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
OLVASÓESZKÖZ	<i>fájl []</i>		Az alapértelmezett billentyűzet helyett a <i>fájlból</i> veszi a bemenő adatokat. Ismét a billentyűzetre áll át.
READFROM			
OLVASSZÓ, OSZ	<i>nincs</i>	<i>szó</i>	Egy szót olvas. - *
READWORD, RW			
OLVASÜZENŐ	<i>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 szám lista</i>	<i>lista lista lista lista lista</i>	Párbeszédablakot jelenít meg <i>szöveg 1</i> kérdéssel, <i>szöveg2</i> címsorral, <i>szöveg4</i> kezdőértékkel. A <i>lista3</i> elemei az ablak pozícióját, méretét, keret típusát és színeit határozhatják meg.
READLISTDIALOG, RLD			
OSZTOTTABLAK, OA	<i>nincs</i>		A Fő ablakot rajz- és írólapra osztja.
SPLITSCREEN, SS			
ÖSSZEG	<i>szám 7 szám2 (szám 1...) Hid: pont1 pont2 Hid: (pont 7...)</i>		Kettő vagy több szám összege.
SUM			Kettő vagy több vektor összege.
ÖSSZEKEVER	<i>szó/lista/képsor</i>		A paramétereként kapott sorozat elemeit összekeveri.
SHUFFLE			

P

³ ARANCSJEU	<i>szó ahol szó nem több, mint 1 6 karakter</i>	- -	Az írólap parancsjelét változtatja meg.
SETPROMPT			

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

R

RAJZABLAK, RA	<i>nincs</i>		A grafikus képernyő teljesen kitölti a Fő ablakot.
GRAPHICSCREEN, GS			
RAJZLAPABLAK	<i>nincs</i>	<i>téglalap</i> <i>[szám 1 szám2 egész3 egész4]</i>	Rajzlap aktuális beállításait írja le. <i>szám 1</i> a szélesség. <i>szám2</i> a magasság, <i>egész3</i> az igazítás, <i>egész4</i> a gördítősáv jelenléte.
GETGS			
RAJZLAPABLAK!	<i>[szám 1 szám2 szám3 szám4]</i> ahol <i>szám 1</i> , <i>szám2</i> egész [8, 2048], és <i>szám3</i> , <i>szám4</i> vagy 0 vagy 1		Rajzlapablak beállítása, <i>szám 1</i> a szélesség, <i>szám2</i> a magasság, <i>egész3</i> az igazítás, <i>egész.4</i> a gördítősáv jelenléte.
SETGS			
RAJZLAPSZÍN, RSZ	<i>nincs</i>	<i>szín</i>	A rajzlap háttérének aktuális <i>színe</i> .
BACKGROUND, BG			
RAJZLAPSZÍN!. RSZ!	<i>szín</i>		Törli a rajzlapot és beállítja az új háttérszint.
SETBACKGROUND, SETBG			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
RÁKATTINTOTT	<i>nincs</i>	<i>teknőc név</i> <i>[]</i>	Melyik teknőcöt érintettük meg utoljára az egér bal gombjának kattintásakor.
TOUCHED			
REJGOMBSOR, RG	<i>nincs</i>		Eltünteti a Gombsor ablakot a képernyőről.
HIDEBUTTONS			
REJTIKONSOR, RI	<i>nincs</i>		Eltünteti a Fő ablak ikonsorát a képernyőről.
-IIDESPEEDBAR			
ÉKORDNEVEK	<i>nincs</i>	<i>szavak listája mező definíciókkal</i>	A mezőket tartalmazó szavak listája.
°ROPS			
REKORDOK	<i>nincs</i>		Kírja az összes szót, melyhez eddig mezőket definiáltunk.
°PS			
ÍENDSZERMÉRET	<i>szám szó</i>	<i>szám szám</i>	A képernyő arculatát kezelhetjük.
ÍSTEMMETRICS			

S, SZ

SAJÁTNEVEK	<i>nincs</i>	Logo eljárások nevei	Az összes saját eljárást tartalmazó listát kapjuk.
SROCS			
SAJÁTSZÓ	<i>szó definíció</i> <i>Jista</i>		Egy olyan eljárást definiál, melynek <i>szó</i> a neve, és a <i>definíció Jista</i> a törzse.
dEFINE			
SAJÁTSZÓ?	<i>szó</i>	<i>logikai érték</i>	IGAZ, ha a szó tetszőleges általunk definiált eljárás neve, egyébként HAMIS.
DEFINED?			

Teljesnév, rövidítés	bemenő paraméterek	Eredmény	Hatása
SIN	számi	szám2	szám2=s\ n(szám 1)
SIN			
SOKSZÖG	polygonjista	sokszög	Az összes aktív teknőc egy a bemenő sokszöglistának megfelelő kitöltött sokszöget rajzol.
POLYGON			
SOKSZÖGVONAL	polygonjista		Az összes aktív teknőc egy a bemenő sokszöglistának megfelelő NEM kitöltött sokszöget rajzol.
POLYGONLINE			
SZÁM?	akármí	logikai érték	IGAZ, ha a paramétere számjegyek nem üres sorozata.
NUMBER?			
SZIA	nincs logikai érték		Kilép a Lógóból. 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 Lógót.
BYE			
SZORZAT	szám 7 szám2 (szám 1...) Hid: paraméter 1 paraméter2 Hid: (paraméteri...) 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.
PRODUCT			
SZÓ	szó 1 szo'2 (szó...)	szó szó	Paramétereiként kapott szavakból egyetlen szót állít össze.
WORD			
SZÓ?	akármí	logikai érték	IGAZ, ha a paramétere szó, különben HAMIS.
WORD?			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
SZÓTARTALOM	szó ahol szó név sajáteljárásnév	definícióJista	Eredménye a szó nevű saját eljárásához tartozó definíció Jista.
TEXT			
SZÖVEGDOBOZ	ÍJ szó	lista szó	Az aktív szövegdozoteknőc adata(i).
GETBOX			
SZÖVEGDOBOZ!	[kulcsszó 1 érték 1 kulcsszó2 érték2...] (a szövegdozoz tulajdonságai)		Felveszi a beállított értékeket az aktív szövegdozoteknőc.
SETBOX			
SZÖVEGDOBOZMÓD!	logikai		Az aktív teknőcök szövegdozozzá válnak.
SETBOXMODE			
SZŰRŐ	eljárásobjektum __ lista	lista	Kiválogatás eljárásobjektum alapján.
FILTER			

T, TY			
TAKAR?	teknőc_név teknőc_név_lista	logikai érték logikai érték	Takarja-e a megadott teknőc az aktuálisat.
OVERLAP?			
TAKART	nincs pont	teknőc_név teknőcjista	Azon teknőcök, amelyeket az első aktív teknőc takar. A ponton lévő teknőcök.
OVERLAPPED			
TANULD	szó szó paraméter 1 pa- raméter2... Hid: szó paraméter 1 [szám]		Eljárás létrehozása
TO			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
TARTALOM	szó ahol szó változónév	akármí	A változónévhez kapcsolt értéket adja.
THING			
TARTALOMJEGYZÉK, TART	() szűrő (szűrői szűrő2... j	lista lista lista	Ha egyetlen paramétere nincs, 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...).
DIR			
TEKNŐCOLVASLISTA, TOL	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.
READLISTTURTLE, RLT			
TERÜLET	nincs	téglalap ÍJ	Teknőc által felhasználható terület.
GETWINDOW			
TESZT	logikai érték		A paraméter logikai értékét megjegyzi HAIGAZ, HAHAMIS utasítások által való későbbi felhasználáshoz.
TEST			
TG	szám 1 (szög)	szám2	$lg(\text{szám } 1) = \text{szám}2$
TAN			
TIZEDESJEGY!	szám egész [0, 1 8]		Kijelzendő tizedesjegyek számát állítja be.
SETFORMAT			
FOLL	nincs	szó (TF, TL, TRD vagy TVT valamelyike)	Az első aktív teknőc tollának állapota.
EN			
rOLL!	szó vagy TF, TL, TRD vagy TVT		Az aktív teknőc tolla állapotának beállítása.
5ETPEN			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
OLLATFEL, TF	nincs		=TOLL! TF
ENUP, PU			
OLLATLE, TL	nincs		=TOLL! TL
ENDOWN, PD			
OLLÁLLAPOT	nincs	[szó szín tollvastagság toll minta] ahol szó TF, TL, TRD vagy TVT valamelyike	Aktív teknőc tollának minősége (szín, vastagság, minta).
ENSTATE			
OLLÁLLAPOT!	[szó szín tollvastagság tol lm in taj ahol szó: TF, TL, TRD vagy TVT		Az aktív teknőcök tollának összes jellemzőjét (mód, szín, vonalvastagság, minta) állítja be.
ETPENSTATE			
OLLMINTA, TM	nincs	tolljvinta	Az első aktív teknőc aktuális tollmintájának számkódja.
ATERN			
OLLMINTAITM!	tolljvinta		Az összes aktív teknőc tollmintáját állítja be.
ETPATTERN			
DLLPONT	képsor	pontjista	Eredménye a tollpontok listája, ahol pont 7 a képsor első fázisának tollpontja, pont2 a második fázis tollpontja stb.
OTSPOT			
DLLPONT!	képsor tol/pont képsor [paraméter 1... j ahol minden listaelem [paraméter 7...] 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.
ETHOTSPOT			

TOLLRADÍR. TRD	<i>nincs</i>		Az összes aktív teknőc tollának állapotát megváltoztatja úgy, hogy ha ezek mozognak, akkor letörlik, amin áthaladnak.
PENERASE, PE			
TOLLSZIN, TSZ	<i>nincs</i> -	<i>szín</i>	Az első aktív teknőc tollának aktuális színe.
PENCOLOR. PC			
TOLLSZIN!. TSZ!	<i>szín</i>		Az összes aktív teknőc tollszínét állítja be.
SETPENCOLOR. SETPC			
TOLLVASTAGSAG, TV	<i>nincs</i>	<i>tollvastagság</i>	Az első aktív teknőc pillanatnyi tollvastagságának megfelelő szám.
PENWIDTH. PW			
TOLLVASTAGSAG!. TV!	<i>tollvastagság</i>		Az összes aktív teknőc tollvastagságát állítja be.
SETPENWIDTH. SETPW			
TOLLVALTO. TVT	<i>nincs</i>		Az összes aktív teknőc tollának működését megváltoztatja úgy, hogy azon pontok színét, amelyen áthaladnak, invertálja.
PENREVERSE, PX			
TÖLT	<i>nincs</i>		Az összes aktív teknőc kitölti az öt körülvevő környezetet az aktuális töltőszínnel és töltőmintájával.
FILL			
TÖLTŐMINTA. TLM	<i>nincs</i>	<i>kitöltőminta</i>	Eredménye az a töltőminta, amit előzőleg az első aktív teknőcnek beállítottunk.
FILLPATTERN, FP			
TOLTOMINTA!	<i>kitöltő_minta</i>		Az aktív teknőc aktuális kitöltőmintáját állítja be.
SETFILLPATTERN. SETFP			

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
TÖLTŐMÓD!	0 vagy 1		Az aktív teknőc kitöltési stratégiáját határozza meg.
SETFILLMODE			
TÖLTŐSZÍN, TLSZ	<i>nincs</i>	<i>szín</i>	Az első aktív teknőc töltőszíne.
FILLCOLOR, FC			
TÖLTŐSZÍN!, TLSZ!	<i>szín</i>		Az összes aktív teknőc töltőszínét állítja be.
SETFILLCOLOR, SETFC			
TÖRÖL	<i>paraméter 1 (paraméter 1 paraméter ...)</i> minden <i>paraméter</i> vagy <i>memóriaobjektum</i> vagy <i>memóriaobjektumjista</i>		Törli a felsorolt összes memóriaobjektumot.
ERASE.ER			
TÖRÖLFÁJL	<i>fájl</i>		Törli a /a/A.
ERASEFÁJL, ERFÁJL			
TÖRÖLÍRÓLAP, TI	<i>nincs</i>		Törli az írólap tartalmát, vagyis lefedi az írólap aktuális háttérszínével.
CLEARTEXT, CT			
TÖRÖLMEZŐ	<i>szó 7 szó2</i>		Törli a szó 1 nevű rekord szó2 nevű mezőjét.
REMPROP			
TÖRÖLMUNKA	<i>nincs</i>		Törli az összes memóriaobjektumot és teknőcöt, lefuttatja az COMLOGO.INI fájl - ha ez létezik a Logo alaplönyvtárban. Letörli a rajzlapot is, és alapállapotba hozza a Lógót.
ERALL			

teljesnév, rövidítés	Bemenő paraméterek	Eredmény	Hatása
TÖRÖLRAJZ	<i>nincs</i>		Törli a rajzlap tartalmát, vagyis lefedi az aktuális háttérszínnel.
CLEAN			
TÖRÖLRAJZLAP, TR	<i>nincs</i>		Törli a rajzlap tartalmát, vagyis lefedi az aktuális RAJZLAPSZÍN-nel.
CLEARSCREEN, CS			
TÖRÖLTEKNŐC	<i>teknőc</i> <i>teknőcjista</i>		Törli az adott teknőcöt. illetve teknőcöket.
ERASETURTLE. ERTURTLE			

u, ú, ü, ő

UTOLSÓ	<i>akárm</i> (nem üres)	<i>akárm</i>	A bemenetként kapott sorozat utolsó eleme.
LAST			
UTOLSÓNAK	<i>akárm lista</i> <i>szó 1 szó2</i> <i>képsor 7 képsor2</i>	<i>lista</i> <i>szó</i> <i>képsor</i>	A <i>lista. szó2, képsor2</i> végére illeszti rendre <i>akárm</i> -t, <i>szó 7-t, képsor 7-t</i> .
LPUT			
UTOLSÓNÉLKÜLI. UN	<i>akárm</i> (nem üres)	<i>akárm</i>	Eredménye az eredeti paraméter utolsó elemének elhagyásával kapott sorozat.
BUTLAST. BL			
ÚJRASOROL	<i>teknőcjista</i>		A <i>teknőcjistá</i> megadott sorrendbe rakja a teknőcöket. A sorrend a MIND és KIFIGYEL eljárásokkal vizsgálható.
REORDER			

Teljes név rövidítés	Bemenő paraméterek	Eredmény	Hatása
ÚJTEKNŐC	<i>szófj</i> <i>szó [szám 1 szám 2</i> <i>szám 3]</i> <i>szó [szám 7 szám2</i> <i>szám3..szós..alak]</i> Hld: <i>szó alak</i> Hld: (<i>szó</i>) ahol... szós... <i>akárm</i> - lyen szavak a követke- zők közül: TF, TL, TRD, TVT, TOLLATFEL, TOLLATLE. TOLLRADÍR, TOLLVÁLTÓ, LÁTHATÓ, LÁTHATATLAN, FIGYELJ		Új teknőcöt hoz létre a szobán megadott néven, a többi paraméter az aktuális beállításokra vonatkozik. (Az <i>alak képsor</i> , vagy "LGWJÁJ" vagy <i>változó</i> képsor értékkel.)
MAKETEKNŐC			
ÚT	<i>nincs</i> (<i>Logojlkönyvtár</i>)	<i>útvonal</i> <i>útvonal</i>	Az aktuális munkakönyvtár, a (DEMO, FELADAT, GYEREK, HANG, KÉPSOR, PROJEKT, RAJZLAP) aktuális Logo alkönyvtár helyét adja.
PATH			
ÚT!	<i>útvonal</i> (<i>Logo_alkönyvtár</i> <i>útvonal</i>)		Az útvonalban adott paraméter lesz a munkakönyvtár. A (DEMO, FELADAT, GYEREK, HANG, KÉPSOR, PROJEKT. RAJZLAP) Logo alkönyvtárát állítja be.
SETPATH			
JRES?	<i>akárm</i>	<i>logikai érték</i>	IGAZ, ha paramétere üres szó, üres sorozat vagy üres képsor, egyébként HAMIS.
EMPTY?			
JRESKÉPSOR	<i>nincs</i>	<i>képsor</i>	Egy <i>üresképsor</i> , olyan, amely egyetlen fázist sem tartalmaz.
EMPTYIMAGE			
JZENŐ	<i>szöveg 7</i> <i>szöveg 1 szöveg2</i>	<i>szöveg(P)</i>	Utastásablakot nyit. Ennek címsora <i>szöveg2</i> (ha megadjuk), és az ablak- ban belül nem szerkeszthető üzenet <i>szöveg 7</i> .
MESSAGEBOX			

V

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

X

Teljes név, rövidítés	Bemenő paraméterek	Eredmény	Hatása
XHELY, X	<i>nincs</i>	<i>szám</i>	Az aktív teknőc x koordinátája.
XCOR			
XHELY!. X!	<i>szám</i>		Az aktív teknőc x koordinátáját állítja be.
SETX			
XYHELY!, XY!	<i>szám 1 szám2</i>		Az aktív teknőc x, illetve y koordinátáját állítja be.
SETXY			

Y

YHELY, Y	<i>nincs</i>	<i>szám</i>	Az aktív teknőc y koordinátája.
YCOR			
YHELY!. Y!	<i>szám</i>		Az aktív teknőc y koordinátáját állítja be.
SETY			