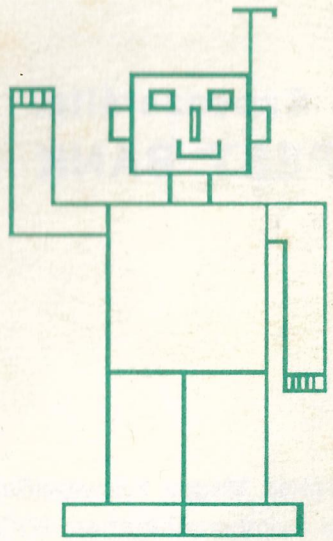
PRAKTICOMP
SZÁMÍTÁSTECHNIKAI KFT.
Gyula, Kossuth u. 22. Pf.: 108
Telefon/Fax: (66) 62-456, 61-058
Telex: 83-732
Minden, ami IBM PC-vel kapcsolatos,
nálunk megtalálható:
Hardware, software
NOVELL hálózatok
MODEMEK
Telexcomputerek
Kellékek
Szervíz
Kiadványszerkesztő rendszer
Angol nyelvoktatás

ABC STÚDIÓ BT.

Teknőc grafika

LOGO 2.9



Békés Megye Képviselőtestülete
Pedagógiai Intézete

1992

Írta: Schneider Ferenc

Szerkesztő: Povázsay Mihály

A jegyzet a KFA pályázat támogatásával készült

Szponzorálta:
BUDAPEST BANK Rt.

Készült a Békés Megye Képviselőtestülete Pedagógiai Intézete sokszorosítójában 1000 példányban

Felelős kiadó: dr. Kincses László

Forgalmazza és terjeszti az APC-Stúdió BT.
5700 Gyula, Kerék u. 9.

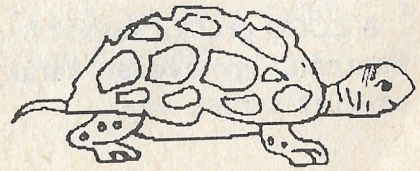
LOGO programozási nyelv (Teknőc grafika)

A LOGO programozási nyelv elsősorban grafikus nyelv. Rajzokat, ábrákat készíthetünk segítségével. Szintaktikája (formai szabálya) azonban jelentősen eltér a mikroszámítógépeken használt programnyelvek többségétől.

A LOGO nyelv 1970-ben indult el világkörüli útjára. Hogy még ma is létezik, ennek oka az, hogy sokan használják. A számítástechnika világában néhány év alatt eltűnnek programnyelvek, ha nem felelnek meg egy adott problémakör megoldásának. A LOGO hasonmásaként emlegethetnénk a LEGO-t. Ezt a játékot biztos ismerős, valószínűleg sok kellemes emlék fűződik hozzá. Kicsi elemekből építkezve rakhatunk össze bonyolult, összetett tárgyakat, figurákat. A LOGO nyelv is hasonló, de itt kicsiny programocskákra kell gondolni a kis építőelemek helyett. Ezeket a kis programokat nekünk kell elkészíteni. A kis programok felhasználásával ezután már nagyobb feladatokat is megoldhatunk. Mire a jegyzet végére érünk, a LOGO-t éppen úgy megszeretjük, mint ahogy a LEGO játékot.

A LOGO-t úgy is emlegetik; "teknőc" grafika. Honnan ez az elnevezés? A LOGO nyelvben nem vagyunk egyedül. Van szolgálatkész társunk is, egy kis teknős személyében.

Sajnos teknőcünknek csak az orrocskáját látjuk!



Teknőcünk mindig arra megy, amerre "mondjuk".

Meg is taníthatjuk arra, mit csináljon, merre "menjen", ha szeretnénk ezt, vagy azt az ábrát kirajzoltatni vele. útja során vagy húz vonalat maga után (ezt látjuk is), vagy nem hagy maga mögött nyomot.

A LOGO programozási nyelvnek is többféle változata, úgynevezett nyelvjárása van.

Ez a jegyzet, a LOGO 2.9-es változatát, és használatát mutatja be, sok-sok példán keresztül. Ennek a nyelvjárásnak néhány sajátossága is van. Mindjárt az elején tudnod kell, ez is magyar nyelvű utasításokkal, illetve parancsokkal rendelkezik, akár a W-programnyelv. Ha valamiben elakadunk, vagy hibásan gépelünk be valamit, csak kérni kell a segítséget, és a teknőc segítségdünkre siet.

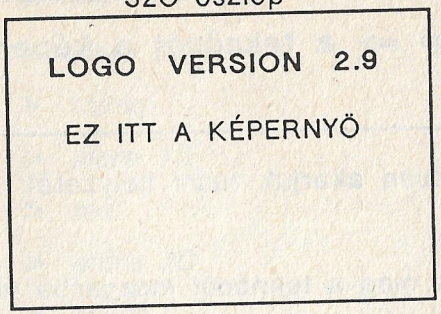
Minden LOGO-ban megírt "program" egy-egy kis építőkocka, akár a LEGO-ban. Ezek összefűzhetők, egy nagyobb feladat megoldásához, akár a LEGO-kockák, ha építkezünk. A "programjaink" kimenthetők kazettára, lemezre. Saját könyvtárunkban egyszerre 50 programot tárolhatunk. Ha nagyon szép ábrát készítettünk, azt ki is nyomtathatjuk. Sajnos nem mindegyik gépen használható ez utóbbi lehetőség.

A LOGO 2.9 azonban egyaránt használható IBM-PC, COMMODORE 128, COMMODORE 64, COMMODORE +4, és TVC gépeken. Nem utolsó az sem, hogy szabadon másolható.

A LOGO grafika sokkal finomabb felbontású, mint az eddig használt, úgynevezett karakteres grafika. Itt 320 oszlopunk és 200 sorunk van.

320 oszlop

200 sor



Az alsó 40 grafikus sor azonban 5 karakteres sort képez. Erre azért van szükség, hogy a begépelt parancsokat, üzeneteket is lássuk a képernyőn. (Később majd megtanuljuk, hogyan használhatjuk az egész képernyőt.)

Ha a LOGO tolmácsot betöltöttük gépünkbe, "bejelentkezik". A képernyő legfelső sorában a tolmács neve és verziószáma olvasható. A legalsó sorban egy ">" jel látható. Ezt a jelet a későbbiekben mindig **promt**-nak fogjuk hívni. Azt a sort mutatja, ahova begépeltük parancsunkat. Mondhatnánk úgy is, hogy teknőcünk parancssorát jelzi. (Akik IBM-PC géppel ismerkednek, azoknak nem idegen ez a kifejezés. Hol találkoztunk már vele?)

Képernyőnk grafikus része azonban üres. Hol a teknőc orrocskája? Teknőcünk nem látható! A képernyőn kívül található. Ez mindig igy lesz, amikor elindítjuk a LOGO 2.9-es tolmácsot. A hasonló eseteket úgynevezett **alapértelmezés**nek nevezzük. Teknőcünk tehát "elbújt". A képernyő 0. sorának 0. oszlopában található.

Csalogassuk elő!

Parancsunk a következő:

kozep => a teknőcöt a képernyő közepére viszi

Ha pontosan akarjuk tudni helyzetét; 159. oszlop és 99. sor.

Próbáljuk meg a teknőcöt mozgásba hozni!

Parancsunk:

elore 20 => a teknőc 20 egységet ment előre

(Helyzete 159. oszlop, 79. sor.)

Próbáld ki most önállóan!

Teknőcünk egy kis tollat "visz" magával. Ha parancsot adunk, akkor a tollát "lerakja" azaz vonalat húz útja során, vagy felveszi, és akkor nem hagy nyomot maga mögött. Mielőtt kipróbálnánk új parancsunkat, hozzuk a teknőcöt középre!

A parancs:

toll => a teknőc lerakja a tollát, és nyomot hagy útja során

tollne => teknőcünk felemeli a tollát, útja során nem hagy nyomot

Próbáljuk ki új tudományunkat! Gépeljük be:

> *toll*
> *elore 20*
> *tollne*
> *elore 10*
> *toll*
> *elore 20*
> *tollne*
> *elore 10*

Ha jól dolgoztunk, szaggatott vonalat rajzolt a képernyőre! Szeretném felhívni a figyelmet, teknőcünk mindig arra megy, amerre az orrocskája mutat!

» **Kérjük középre a teknőcöt!**

új rajzhoz azonban nem kezdhetünk, hisz a képernyőn ott van előző munkánk eredménye. Próbáljuk meg letörölni a képernyőt!

Parancsunk a következő:

szivacs => Törli a teljes grafikus képernyőt!

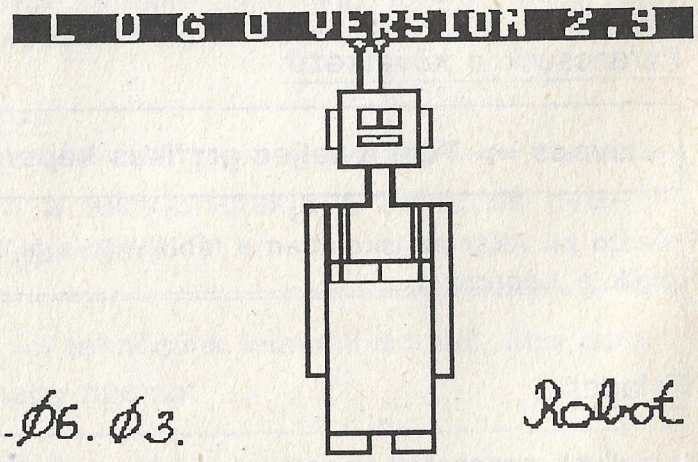
Próbáld ki! Akár az iskolában a táblát, itt is a "szivaccsal" töröljük a képernyőt.

Feladat:

» **Rajzoljunk szaggatott vonalakat a képernyőre úgy, hogy 5-5 egység távolságra legyenek egymástól!**

Hozzunk minden órára milliméterpapírt!

Ezeket a rajzokat 10-11 éves gyermekek készítették! Fél évig tanítgatták teknőcüket, mire elérték azt, hogy butuska teknőcük "megtanulta" ezeket a rajzokat. Kitarással és munkával bárki képes hasonló remekművek alkotására. Csak el kell rá szánni magunkat!



Deák Katalin (4.c) rajza

Milliméterpapír használata, pozicionálás a képernyőn

Milliméterpapírunkon 270 oszlop és 180 sor található. A vastagabb vonalak 10 egységet jelentenek oszloponként és soronként. Számozzuk meg, a vastag vonallal jelzett oszlopokat, sorokat!

Láthatjuk, hogy 270 oszlop és 180 sor van. Jelöljük be most a teknőc orrát a milliméterpapíron!

Nem éppen szerencsés a beosztás! Teknőcünk a milliméterpapír jobb oldalára kényszerült. Ahhoz, hogy középre kerüljön, célszerű, ha az oszlopok számozását 30-tól kezdjük. Javítsuk ki az oszlopok számozását! Rajzoljuk meg most a teknőc orrát! Most már középre kerül! Igy azonban igaz, hogy 30 egységet elvesztettünk, de cserébe középre került a teknőcünk, akár a képernyőn. Bátran tervezhetünk ezentúl!

Teknőcünket azonban csak a képernyő közepére tudjuk elhelyezni. Hogyan helyezhetjük más helyre?

Parancsunk:

helyez *sor oszlop* => **A teknőcöt az általunk megadott sor oszlop pozicióba viszi!**

Próbáljuk ki! Gépeljük be:

> *helyez 100 130*

Teknőcünk a képernyő 100. sorának 130. oszlopába került.

A LOGO nyelvben az egyes adatokat, utasításokat szóközökkel választjuk el egymástól! Ha ezt elmulasztjuk, teknőcünk hibásan értelmezi a parancsot, vagy hibaüzenetet ad.

Feladat:

- Rakjuk le a teknőc tollát!
- Most menjünk előre 100 egységet!
- Helyezzük a teknőcöt a 100. sor 190-es poziciójába!
- Menjünk előre 100 egységet!
- Hozzuk a teknőcöt középre!
- Rajzoljunk 10 egységnyi hosszúságú szaggatott vonalakat a képernyő tetejéig!

Mi lett az eredmény, mire hasonlít a rajzunk? A választ írjuk le!

- Irjuk le a LOGO programot a füzetedbe úgy, hogy a program nevét írjuk le először!

A következő feladat kicsit nehezebb lesz!

- Osszuk fel a képernyőt 8 egyenlő részre!
- A feladat megoldásához használjuk milliméterpapírt.
- A programot írjuk le a füzetedbe! (Segítség: a legalsó sor a 159-es.)
- Ha készen vagyuk próbáljuk ki!

Forgás a teknőccel!

Akármilyen érdekes is a vonalak rajzolása, előbb, vagy utóbb unalmas lesz, hogy teknőcünkkel csak vonalakat rajzolunk a képernyőre. Természetesen teknőcünkkel forogni is lehet. Ehhez azonban fel kell elevenítenünk a matematika órán tanultak közül néhány ide kapcsolódó részt. Már 3. osztályban szó volt róla, hogy a kör 360 fokos.

Hegyes szögnek a 0 foknál nagyobb, de 90 foknál kisebb szögeket nevezzük.

Igy jelöljük: $0° < 90°$

Tompa szögnek a 90 foknál nagyobb, de 180 foknál kisebb szögeket nevezzük!

Igy jelöljük: $90° < 180°$

A derékszög pedig 90° fokos.

» 1. Rajzoljunk le a tankönyvbe egy 3 cm oldalhosszúságú négyzetet!

» 2. Mik a jellemzői ennek az ábrának?

» oldalaik _____

» szögei _____

» 3. Ha sikeresen válaszoltunk a kérdésekre rajzoljunk milliméterpapírra 80 egységnyi oldalhosszúságú négyzetet!

» Beszéljük meg, hogyan lehetne a feladatot megoldani?

» Irjuk le az egyes lépéseket!

» 1_____

» 2_____

» 3_____

» 4_____

» 5_____

» 6_____

» 7_____

» 8_____

Az utasítások, amelyekkel a forgás megvalósítható:

jobbra szög => A teknőc az általunk megadott szögben jobbra fordul.

balra szög => A teknőc az általunk megadott szögben balra fordul.

Ne felejtsük el, az adatokat szóközzel kell elválasztani egymástól!

» 4. Gépeljük be a kész LOGO programot!

» 5. Készítsük el most a négyzet LOGO programját 40 egységnyi oldalhosszúságra úgy, hogy a kezdő pozíció az előzővel megegyezzen!

» 6. Gépeljük be a kész programot! Hasonlítsuk össze a négyzetet és a téglalapot!

» A négyzet oldalainak a száma _____

» A négyzet szögei _____

» A téglalap oldalainak a száma _____

» A téglalap szögei _____

» 7. Önállóan készítsünk milliméterpapírra egy 40X80 egységnyi oldalhosszúságú téglalapot!

» 8. Készítsük el a LOGO programját!

» 9. Próbáljuk ki a programot!

» 10. Próbáljuk ki, az elején a következő módosítással!

> *szivacs*

> *kozep*

> *jobbra 45*

> *ide írjuk saját programunkat*

> .

Az ötlet felhasználásával próbáljunk önállóan hasonló, vagy még érdekesebb rajzokat készíteni.

Az elkészült legszebb munkáidat írd le ide!

Háromszög rajzolása

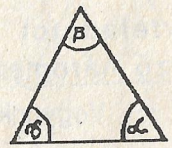
Háromszöget rajzolni viszonylag egyszerű feladat. Nem egyszerű azonban megtanítani a teknőcöt rá! Hogy egyszerűbben boldoguljunk a "teknőccel", elevenítsük fel a matematikaórán szerzett ismereteinket!

Csoportosítsuk először a háromszögeket:

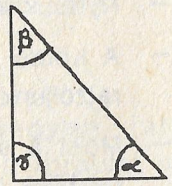
Szögeik szerint	**Oldalaik szerint :**
a, Hegyesszögű	a, Általános
b, Derékszögű	b, Egyenlő szárú
c, Tompaszögű	c, Egyenlő oldalú

Minden háromszög közös jellemzője, hogy belső szögeinek összege 180°! Mi most elsősorban az egyenlő oldalú háromszöggel fogunk közelebbről megismerkedni. Azért kell, hogy teknőcünkkel házakat, rakétát, ... és hasonló izgalmas rajzokat tudjunk készíteni! Az egyenlő oldalú háromszögnek azért ez a neve, mert **oldalai egyenlőek**, akár a négyzetnek. Ha oldalai egyenlőek, akkor **belső szögeinek is egyenlőnek** kell lennie! Ha igaz az állítás miszerint minden háromszög belső szögeinek összege 180°, akkor csak a 180°-ot kell elosztanunk 3 egyenlő részre!

1.



2.



3.

Végezzük el a műveletet, és irdjuk ide az eredményt:

_____

Most vegyük elő a szögmérőt és a vonalzót! Szerkeszünk ki egy 3 cm oldalhosszúságú háromszöget a bal oldalra!

A feladatot körzővel is megoldhatjuk! Rajzoljuk az előző háromszög mellé!

- Vegyünk fel egy 3 cm hosszúságú szakaszt!
- Nyissuk ki a körződ 3 cm -re (Ellenőrizzük vonalzón!)
- A körző hegyét szúrjuk a szakasz kezdőpontjába, és rajzoljunk körülbelül negyed körívet a szakasz felett!
- Most a szakasz végpontjába szúrjuk a körzőt és rajzoljuk körülbelül negyed körívet a szakasz felett!
- A két negyed körívnek egy metszéspontot kell adnia! (Ha négsem, akkor valamit elhibáztunk!
- Kössük össze ceruzával a szakasz kezdő és végpontját a körző által adott metszésponttal!
- Ellenőrizzük vonalzóval a kapott eredményt!

Most próbáljuk ki, mit rajzol a teknőc?

> *kozep*

> *elore 30*

> *jobbra 60* _____

> *elore 30*

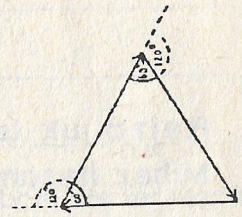
> *jobbra 60* _____

> *elore 30*

> *jobbra 60* _____

Az első két sor hibátlan, de a szögekkel baj van! Mi lehet a hiba?

Nézzük meg figyelmesen az ábrát!



Ha teknőcünkkel csak 60 fokot fordulunk, ez kevés. Ha az egyenesre, szögmérőt helyezünk, akkor 180 fokot mérhetünk. Ahhoz, hogy programunk hibátlanul működjön, 180-60 fokot, azaz 120 fokot kell fordulnunk! úgy képzeljük el, mintha teknőcöd hátán ülnénk, és mi adnánk a parancsot, merre menjen!

Javítsuk ki a programot hibátlanra!

Hogy mindez a magyarázat érthető legyen, játszani fogunk! Kipróbáljuk, merre, hány fokot kell fordulni ahhoz, hogy teknőcünk hibátlanul dolgozzon! A játékban te leszel a teknőc! Vegyünk 3 tárgyat a kezünkbe! Induljunk el előre 2 lépést! Tegyél le egy tárgyat a földre! Fordulj kb. 120 fokot jobbra! Ismételd meg a tevékenységet még kétszer! Szemléld meg a földön levő tárgyakat! Ha összekötnénk őket egy egyenessel, mi lenne az eredmény?

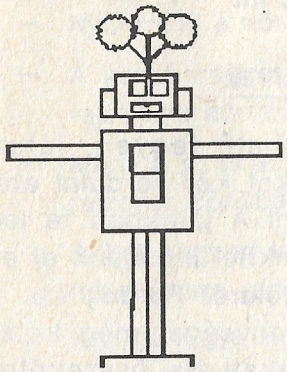
_____

Ha a feladat nehéz volt, próbáljuk meg egy négyzetet "kirajzolni" az előző módszerhez hasonlóan!

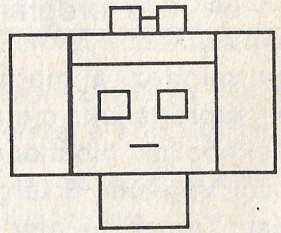
Próbáljuk ki a következő programot!

```
> szivacs
> kozep
> jobbra 30
> elore 50
> jobbra 120
> elore 50
> jobbra 120
> elore 50
> jobbra 120
```

Rajzoljuk le az eredményt a program mellé! Mihez hasonlít?



Huterják László (4.c)



Hevesi Csilla (4.c)

Saját rajzok készítése!

Az elmúlt órán megvizsgáltuk az egyenlő oldalú háromszöget. Az utolsó feladat megoldásaként egy egyenlő oldalú háromszöget rajzoltunk. Ha szabadjára engedjük fantáziánkat, mit lehet készíteni négyzetből, téglalapból és háromszögből? Irjuk le ötleteinket!

_____

_____

_____

_____

_____

1. Válasszuk ki a számunkra legkedvesebbet, és tervezzük meg milliméterpapíron!

2. A füzetbe írjuk le a "programot"!

3. Próbáljuk ki!

Most tervezzünk egy egyszerű házat!

Ha már kész a ház, akkor húzzunk egy hosszú vonalat a házikó alá!

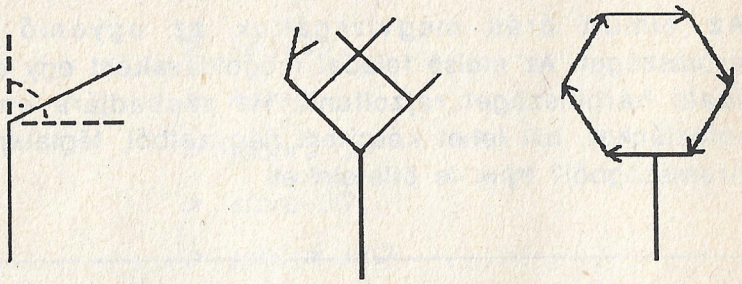
Milyen parancsokra van szükségünk?

_____

_____

4. Irjuk le a módosított "programot"!

Ha sikeresen megoldottuk a feladatot, próbáljunk meg egy kis fát is rajzolni! Mielőtt a feladat megoldásához hozzákezdenénk, tanulmányozzuk a következő rajzokat!



5. A rajz felhasználásával bővítsük ki a milliméterpapíron a házikó rajzát!

6. Irjuk le a füzetbe a fa programját!

7. Próbáljuk ki, és végezzük el a szükséges módosításokat! Munkánk során előfordulhat, hogy egy-egy vonal nem oda kerül, ahova azt szánjuk! Bosszantó, ha ezért letöröljük az eddigi munkát, és újra kezdjük elölről! Megoldásként egy parancs kínálja magát!

torol => A következő utasítások mindegyikét törlésként értelmezi a "teknős"! A törlést, a "toll" parancs oldja fel!

Nézzük a használatát!

> *elore 50*

Parancsunk azonban hibásnak bizonyult, szeretnénk törölni! Gépeljük be:

> *torol*

> *elore -50*

Egy másik megoldás:

Forduljunk meg a teknőccel 180 fokot, menjünk előre 50 egységet, forduljunk vissza, rakjuk le a tollat!

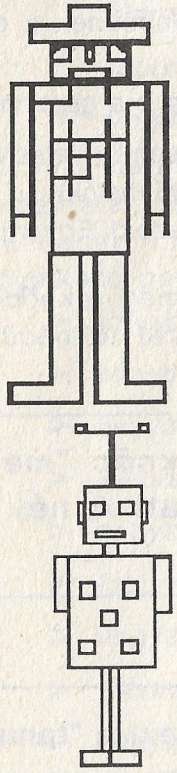
> *jobbra 180*

> *elore 50*

> *balra 180*

> *toll*

Ott vagyunk, ahonnan elindultunk, de hibás parancs következménye már nincs a képernyőn!



robot

Kolarovszki Zoltán (4.c)

Robi

Kovács Ágnes 4.c.

Tanítsuk meg a teknőcöt!

Eddigi munkáinkat teknőcünk szorgalmasan végrehajtotta, de nem "tanult" meg semmit, csak mi gyűjtögettünk tapasztalatokat. Ha sikeresen megoldottunk egy-egy problémát, és végre kész lett a "mű", ha újra szükségünk lenne rá, ismét be kellene gépelnünk. Van azonban egy parancs, mellyel teknőcünket megtaníthatjuk egy-egy tevékenység végrehajtására. Ezek a tevékenységek végső soron programok, de "ELJÁRÁSNAK" nevezzük őket, azért, mert bizonyos értelemben különböznek a programoktól. Mik ezek a különbségek?

- Egyszerre akár 50 eljárás is a gépben lehet!
- Egy eljárásban tetszőlegesen hívhatsz újabb eljárásokat (ezekből újabb eljárások hívhatók)!

Ezek számunkra nem lényegesek! Egyenlőre!

Nézzük, hogy taníthatnánk meg teknőcünket a négyzet rajzolására! A parancs, amivel teknőcünket "tanulásra" bírhatjuk:

eljaras név => A teknőc "megjegyzi" a következő utasításokat! A név megadása kötelező!

vege => A teknőc befejezi a "tanulást"!

Nézzük, hogyan használhatjuk az új tudományunkat!

```
> eljaras negyzet
^ elore 50
^ jobbra 90
^ elore 50
^ jobbra 90
^ elore 50
^ jobbra 90
^ elore 50
^ jobbra 90
^ vege
```

Teknőcünk most már bármikor lerajzol egy 50 egység oldalhosszúságú négyzetet, csak be kell gépelni -"negyzet"- és máris szorgalmasan kirajzolja a megtanult ábrát! Próbáljuk ki a következő parancssorozatot!

```
> szivacs
> kozep
> negyzet
> jobbra 10
> negyzet
> jobbra 10
> negyzet
> jobbra 10
> negyzet
> jobbra 10
> negyzet
```

Érdekes az ábra? Nem tettünk semmi különlegeset, csak elforgattuk a négyzetet 10 fokonként.

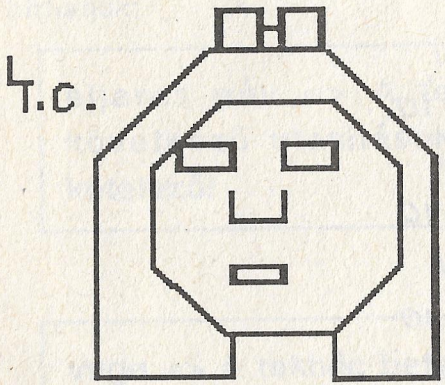
Feladatok:

1/a. Hogyan taníthatjuk meg a teknőcöt egy 50 egységnyi oldalhosszúságú egyenlő szárú háromszög rajzolására?

1/b. Készítsük el az eljárást és próbáljuk ki! Ha hibátlan, írjuk le a füzetbe!

Teknőcünk végrehajtja azt, amit begépelünk, amennyiben az végrehajtható. Ha valamit elhibáztunk, és még nem nyomtuk meg a RETURN billentyűt, egyszerűen javíthatunk! Ha már megnyomtuk, akkor sajnos előlről kell kezdeni a gépelést, úgy, hogy először begépeljük egy új sorba "vege", és kezdjük előlről az egészet! Ezért, ha teknőcöt tanítjuk, különösen figyeljünk arra, mit gépelünk, és mielőtt a RETURN billentyűt megnyomjuk, ellenőrizzük a sort, minden rendben van-e!

2. Készítsünk eljárást, mely a teknőcöt középre helyezi, törli a képernyőt, és lerakja a teknőcünk tollát!



Hatvani Zsuzsanna

Hogyan mentsük, töltsük be eljárásainkat?

Ha egy eljárással kész vagyunk, akkor teknőcünknek elég, ha csak a nevével hivatkozunk rá, például: "negyzet". De sajnos, ha kikapcsoljuk gépünket, oda a teknőc tudománya. Meg kellene őrizni valamilyen háttértárolón az eljárásokat!

Eljárások mentése

ment /d/név => Kimenti kazettára, vagy lemezre az összes eljárást!

Alapértelmezés szerint kazettára menti ki eljárásainkat a HOME (házi) számítógépeken (C128, C64, C+4, bővített C16, TVC. Az IBM-PC változat csak lemezre dolgozik.). Ha azonban lemezegységre szeretnénk kimenteni eljárásainkat, akkor a név elé egy "d" betüt kell írnunk!

Nézzünk rá egy példát:

ment csomag1 => Kimenti KAZETTÁRA "csomag1" névvel az összes eljárást!

ment dcsomag1 => Kimenti LEMEZEGYSÉGRE (diszkre) "csomag1" névvel az összes eljárást! (Az **IBM-PC** gépeken a "d" betű használata felesleges!)

Ha LEMEZRE mentünk, soha ne felejtsük el a név elé beírni a "d" betűt!

Eljárásaink mentésénél célszerű, ha olyan neveket választunk, amellyel egyértelműen megtalálhatjuk programjainkat! Egyezzünk meg, hogy minden mentést "dI"-lel kezdünk. A "d" a lemezegységet az "I" a LOGO-t jelenti, majd mindenki a saját gépszámát gépeli be, két pozíción. Ezt követi a csomagszám. Valahogy így néz ki a mentés, ha a hármas gép mellett:

ment dIO3O1

Igy kimentjük az összes eljárást amit készítettünk. Csak emlékeztető, hogy összesen 5O eljárás lehet egy csomagban. Célszerű, ha az eljáráscsomagok betöltésével kezdjük a munkát! Nézzük hogyan!

Eljárások betöltése

tolt /d/név => Betölti kazettáról vagy lemezről a prg.csomagot!

Alapértelmazésben kazettáról tölti be, az általunk megadott nevű programcsomagot! (IBM-PC esetén csak lemezegység létezik, nem kell a "d" betű!) Ha lemezegységről szeretnéd betölteni, akkor a név elé egy "d" betüt kell írni! A betöltés formai szabályai megegyeznek a mentés előírásaival. Nézzünk egy példát!

tolt dIO3O1 => kazettás egységről tölti az "IO3O1" nevü LOGO eljáráscsomagot.

tolt dIO3O1 => lemezegységről tölti, az "IO3O1" nevü LOGO eljáráscsomagot.

Könyvtárkezelés

Biztosan előfordul majd, hogy elfelejtjük, milyen nevet adtunk utoljára az eljáráscsomagodnak. Célszerű lenne ilyenkor megnézni, mi van a lemezen! Parancsunk:

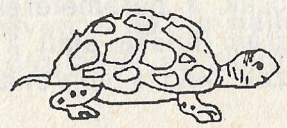
katalogus => Hatására a tolmács kiírja a lemezkatalógust!

Kilépni a katalógus listából, a RETURN lenyomásával lehet!

Lesz rá mód a későbbiekben, hogy társaink munkáit megnézhessük, tanulmányozhassuk! Ilyenkor különösen jó szolgálatot tesz, ha a lemezkatalógust betöltjük.

Az eljáráscsomag névválasztásánál nagyon ügyeljünk a helyes sorrendre és a gépszámra. Ezt úgy hívják "kódolás"! Azért van rá szükség, mert tolmácsunk minden üzenet nélkül "felülírja" (a régi helyére az újat másolja) az azonos nevű programcsomagodat! Ha hibásan gépelünk be valamit, előfordulhat, hogy társaink munkáját tesszük tönkre! Ez az egyszerű kód véd minket a kellemetlen meglepetésektől.
Feladatok:

1. Gépeljük be a képernyőtörlő eljárást, és mentsük ki a lemezegységre!
2. Nézzük meg a katalógust!
3. Töltsük vissza a programcsomagot!



Eljárások paraméterezése

A cím meglehetősen riasztó, de remélem mire a fejezet végére érünk, már jó barátságban leszünk a "paraméterekkel". Eljárásainkkal teknőcünk már sok rajzot képes "megtanulni", de mindig meg kell tanítanunk, ha más oldalhosszúságú négyzetre, háromszögre, stb van szükségünk. Ennél az új eljárásnál minden megegyezik, csupán az *"elore 50 "* utasítás helyére más számot kell írnunk. A paraméterezett eljárások segítségével ezek a problémák egyszerűen megoldhatók.

A paraméterezett eljárások tulajdonképpen változókkal "felszerelt" eljárások. Kanyarodjunk csak vissza a W-nyelvhez! Ha visszaemlékszünk, ott az összes változó a program elején volt felsorolva! A LOGO-nyelvben is hasonló a helyzet, de:

- a paraméterek neve (változónevek) ":"-tal (kettősponttal) kell hogy kezdődjön
- az eljárásnév után kell a változóneveket felsorolni, szóközzel elválasztva.

Igy első hallásra a formai megjelenésen kívül nem tapasztalunk lényeges eltérést a vátozók és a paraméterek között. Mégis van különbség a két fogalom között. **A paraméterek értékét a tevékenység indításakor kell beállítani és az egész tevékenység végrehajtását meghatározzák!** Az eljárás tehát, ha használunk paramétereket, csak a paraméterek beállításával indul. A változók, a program futása alatt kapnak értéket, és nem feltétlen határozzák meg a végrehajtás menetét.

Foglalkozzunk inkább a gyakorlattal, hogy jobban megértsük a különbséget! A négyzet eljárásnál célszerű lenne, ha az oldalt mi adnánk meg, az eljáráshivást követően! Nézzük a megoldást:

```
> eljaras negyzet :oldal
^ elore :oldal
^ jobbra 90
^ elore :oldal
^ jobbra 90
^ elore :oldal
^ jobbra 90
^ elore :oldal
^ jobbra 90
^ vege
```

Hogyan használjuk az új eljárást?

Az eljárás neve után meg kell adni azt is, hány egység legyen az oldal hossza, azaz a paraméternek értéket kell adni!

Gépeljük be: **negyzet 40** => Teknőcünk egy 40 egységnyi oldalhosszúságú négyzetet rajzol.

Gépeljük be: **negyzet 80**

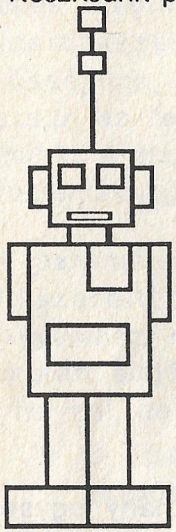
Mi az eredmény? _____

Paraméter alatt tehát azt értjük, amikor az eljárás neve után meghatározzuk milyen értékkel hajtódjon végre, és természetesen erre felkészítettük az eljárást!

Fontos azt tudni, ha azonos nevű eljárást írunk, a régi eljárás helyére az új kerül!

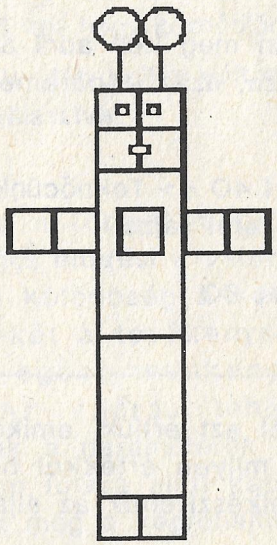
Feladatok:

1. Készítsünk eljárást a háromszögre! Az értékadás paraméteres legyen!
2. Készítsünk paraméteres eljárást a téglalapra!



Robóka

Zahorán Edit (4.c)



Nagy Katalin (4.d)

Eljárások összefűzése, listázása, törlése

Ha eddigi munkáinkat sorra vesszük, akkor igazán nem lehet okunk panaszra. Mértani ábrákból kis fantáziával házat, templomot, vagy akár egy egész falut varázsolhatunk. Eddigi munkáinkban azonban külön hívtuk meg a négyzetet, a háromszöget egy házikó rajzolásához.

Lehetőségünk van azonban arra is, hogy begépeljük "haz", és szolgálatkész teknőcünk azonnal kirajzoljon egy házikót, sőt akár paraméterezve is megtehetjük mindezt, csak a teknőcöt kell megtanítanunk!

Vegyük sorjában! Mi kell ahhoz, hogy egy házikót rajzoljunk? Szükségünk van egy négyzetre és egy háromszögre! Kezdjünk hozzá:

```
> eljaras haz :oldal
^ negyzet :oldal
^ elore :oldal
^ jobbra 30
^ haromszog :oldal
^ vege
```

Teknőcünknek ezentúl elég, ha begépeljük:

```
> haz 50
```

/Más paraméterrel (számmal) is indíthatjuk./

Ezen elindulva, például templom rajzolására is megtaníthatjuk a teknőcöt. Egyúttal szeretném felhívni a figyelmet egy újabb megoldásra a LOGO lehetőségeinél.

/Vigyázzunk a gépelésnél a szóközökre!/

```
> eljaras templom :oldal
^ teglalap :oldal * 3 :oldal
^ elore :oldal * 3
^ jobbra 30
^ haromszog :oldal
^ vege
```

Próbáljuk ki! Elég begépelni:

```
> templom 20
```

Gondoljuk végig a működését!

Eljárások listázása

Eljárásaink száma egyre csak szaporodik. Egyre nehezebben tekinthetjük át, milyen eljárásokat írtunk. Ezen a problémán segít a következő parancs.

program sorszám => Az adott sorszámtól kiírja az eljárásokat.

Például: program 1 => Az első eljárástól kiírja az eljárásokat!

A lista, a teljes képernyőre történik. Egy-egy eljárás után a listázás leáll, hogy lehetőség legyen az eljárás áttekintésére. Továbblépni a szóköz billentyű lenyomásával lehet. Tolmácsunk ekkor visszatér eredeti állapotába.

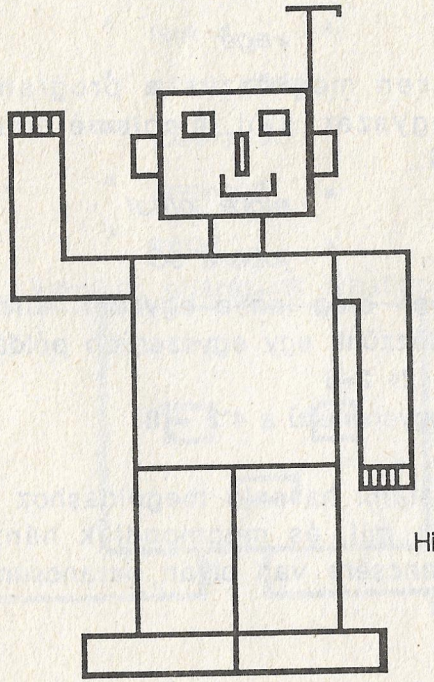
Eljárások törlése

LOGO tolmács lehetőséget kínál arra is, hogy egy-egy eljárást megszüntessünk, kivegyük a csomagból (töröljük)!

Parancsunk:

kivesz név => A megadott nevű eljárást kiveszi az eljáráscsomagból.

A tolmács lehetőségeit, még messze nem használtuk ki, de a LOGO nyelv párhuzama a LEGO játékkal most már talán érezhető. Kicsi építőkockákból nagy feladatok is egyszerűen megoldhatók. Remélem tetszik!



Hidasi László (4.d)

Tevékenységek ismétlése

Nézzük meg, a négyzet rajzolásához szükséges eljárást!

```
> eljaras negyzet :oldal
^ elore :oldal
^ jobbra 90
^ elore :oldal
^ jobbra 90
^ elore :oldal
^ jobbra 90
^ elore :oldal
^ jobbra 90
^ vege
```

Ha figyelmesen megnézzük a programot, láthatjuk, összesen négyszer kell megismételni ugyanazt a tevékenységet!

```
^ elore :oldal
^ jobbra 90
```

Tulajdonképpen elég lenne egyszer leírni és négyszer végrehajtani! Nézzünk egy egyszerűbb példát.

2+ 2+ 2+ 2+ 2=8

ezzel egyenértékű a 4*2 = 8

Elég lenne valami hasonló megoldáshoz folyamodnunk. Egyszer leírjuk mit, és megmondjuk hányszor kell azt ismételni. Szerencsére van olyan parancsunk, amellyel ez megvalósítható!

ism n => Tevékenység ismétlése n szer.

Az ezt követő utasításokat n-szer végrehajtja a tolmács. Az "n" vagy egy szám, vagy paraméternév!

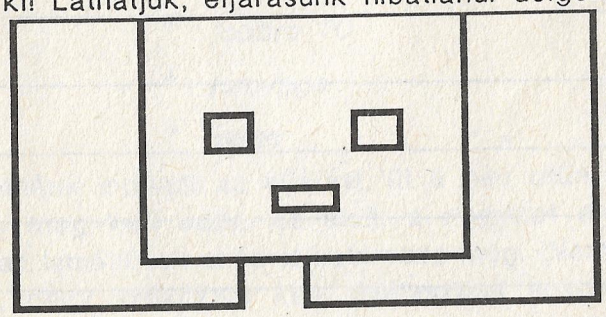
A tolmácsnak a következő utasítással jelezzük az ismétlés végét:

ismvege => Ismétlés végét jelzi a tolmács számára.

Ezek után nézzük, hogyan módusul négyzet eljárásunk az új utasítások felhasználásával!

```
> eljaras neve :oldal
^ ism 4
^ elore :oldal
^ jobbra 90
^ ismvege
^ vege
```

Próbáljuk ki! Láthatjuk, eljárásunk hibátlanul dolgozik!



Palotai Nóra (4.c)

Feladatok:

1. Közösen alakítsuk át a "téglalap" eljárásunkat!

Próbáljuk ki az új eljárást!

2. Alakítsuk át önnállóan a "haromszog" eljárást!

Kalandozás az ismétlések birodalmában

Az ismétlés felhasználásával nem csak eljárásainkat tehetjük sokkal egyszerűbbé. Kis fantáziával valóságos tárháza nyílik meg a lehetőségeknek. Nézzünk egy példát!

```
> eljaras ism1 :ol
^ ism 9
^ negyzet :ol
^ jobbra 10
^ ismvege
^ vege
```

Próbáljuk ki! Az ismétléseket is paraméterezhetjük, ha például az eljárás indításakor szeretnénk beállítani az ismétlések számát. Nézzünk erre is egy példát:

```
> eljaras ism2 :ol :is
^ ism :is
^ negyzet :ol
^ jobbra 10
^ ismvege
^ vege
```

Ha szeretnénk indítani az eljárást, itt a név után már két számot is meg kell adni. Az első, a négyzet oldalát, a második, az ismétlések számát határozza meg. (Nem szabad elfelejteni, hogy szóközzel kell elválasztani egymástól az adatokat)!

Az lenne az igazi, ha nemcsak az ismétléseket, hanem az előfordulás szögét is mi adhatnánk meg! Például:

```
> eljaras ism3 :ol :sz is:
^ ism is:
^ negyzet :ol
^ jobbra :sz
^ ismvege
^ vege
```

Mielőtt indítanánk az eljárást, nem szabad elfelejteni, hogy itt három paramétert kell megadni! Az oldal hosszát, az elfordulás szögét, és az ismétlések számát. Természetesen, ha az eljárásban más sorrendben sorakoztjuk fel a paramétereket, akkor az indításkor is megváltozik a paraméterek sorrendje. Ezt mindenki maga dönti el, azonban a paraméterezés sorrendjét célszerű, ha az eljárás neve után a füzetbe leírnunk, különben könnyen eltévedünk saját eljárásainkban!

Néha egyszerű megoldások is különösen szép alakzatokat hoznak létre. Főleg, ha ezek paraméterezhetőek, mert így szinte univerzálisak. A most bemutatásra kerülő egyszerű eljárás "csodája" nem a programban, hanem a begépelt paraméterekben rejlik!

```
> eljaras vonal :ol :sz :is
^ ism :is
^ elore :ol
^ jobbra :sz
^ ismvege
^ vege
```


Nyugodtan mondhatjuk az eljárásnál, hogy "nyúlfarknyi"! Próbáljuk ki azonban a következő paraméterek begépelésével! A kapott alakzatot írjuk le a paraméter mellé!

vonal 50 90 4 = _____

vonal 50 120 3 = _____

vonal 50 60 6 = _____

vonal 80 175 40 = _____

Most mindenki kedvére gépeljen be új paramétereket! Irjuk le a füzetbe ezeket, és a kapott eredményt is! Válasszuk ki, melyek voltak a legérdekesebbek és írjuk le a tankönyvbe!

_____

_____

_____

_____

_____

_____

_____

Feladat:

A "negyzet" és a "vonal" mintájára készítsünk új eljárást a háromszöggel, "ism4" néven!

Matematikai műveletek a LOGO nyelvben

A LOGO nyelv elsősorban grafikai nyelv, de nem mentes az elemi matematikai műveletektől. Lehetőségeink lényegesen gazdagabbak, szélesebbek lesznek használatukkal.

Értékadás

A W-nyelvben az értékadás úgy kezdődött mindig: "legyen". A LOGO nyelvben is hasonló, de a folytatás alaposan eltér a korábban megszokottól! Nézzünk néhány példát:

legyen :adat 2 => Az utasítás hatására az "adat" nevű változó tartalma 2 lesz.

legyen :oldal :adat => Az "oldal" nevű változó felveszi az ":adat" nevű változó tartalmát!

(Mivel az ":adat" változó tartalma 2 volt, így az ":oldal" változó tartalma is 2 lesz.)

úgy is mondhatnánk, hogy a bal oldalra írt változónév felveszi, a jobb oldalra írt változó tartalmát! Ha pedig egészen szakszerűen akarjuk magunkat kifejezni, akkor úgy mondanánk, **a bal oldali változó a jobb oldali kifejezés értékét veszi fel.** Ez a megfogalmazás már sokkal tágabb, mert egyik változó tartalmát egy másik változó tartalmában helyezzük el. Esetünkben, ez akár matematikai műveleteket is jelenthet!

legyen :adat :adat + 5 => Az utasítás hatására az ":adat" nevű változó jelenlegi tartalmához hozzáad ötöt.

Mivel az ":adat" tartalma 2 volt, az új fiók tartalma 7 lesz. Talán zavaró az egyenlőségjel hiánya, de némi gyakorlat után hozzászokunk a LOGO sajátosságaihoz.

Az egyes műveleti jelek a következők:

\+ = összeadás

\- = kivonás

* = szorzás

/ = osztás

Oldjunk meg néhány gyakorló feladatot!

1. Az utasítássorozat végrehajtása után mennyi lesz az ":aliz" és az ":alladin" változók tartalma?

```
legyen :aliz 5
legyen :alladin :aliz + 21
legyen :aliz :alladin / 2
legyen :alladin :alladin * 3
```

2. Az utasítássorozat végén mennyi lesz ":li" és "la:" tartalma?

```
legyen :li 3
legyen :la 4 * :li
legyen :la 4 + :li
legyen :li :la / 2
legyen :li :li * 3
legyen :la :la / 2
```

Mire használjuk új ismeretünket?

Szeretném egy példán bemutatni, feltételezve, hogy a "vonal" nevű eljárás a gépben van! (Ha nincs akkor töltsük be!)

```
> eljárás spiral :ol :sz :is
^ ism :is
^ vonal :ol :sz 1
^ legyen :ol :ol + 5
^ ismvege
^ vege
```

Mielőtt az eljárást kipróbálnánk, nézzük át, hogyan működik

! Felhívom a figyelmet a "vonal" paraméterezésére!

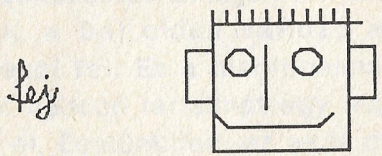
Nézzük meg, milyen paraméterekkel futassuk:

```
> spiral 5 90 40
```

Mi lett az eredmény? _____

Feladatok:

1. Paraméterezük úgy, hogy egy háromszögletű spirált rajzoljunk a képernyőre!
2. Paraméterezzük úgy, hogy egy hatszögletű spirált rajzoljon a teknőc!



Berjési Márton

1991 június 4.

Feltételes ciklusok

Az eddigi ciklusszervező utasításnál ügyelni kellett arra, hogy a ciklus ismétlése ne lépje túl a képernyő méreteit. Nem mintha ez a teknőcöt "zavarba" hozta volna, de látszólag semmi sem történt a képernyőn. Teknőcünk pedig dolgozott, csak éppen olyan területen, amit nem látunk. Ha "spiral" eljárásnál olyan értékadást használtunk volna, amely az oldalból elvesz valamennyit (a kezdőoldal egyébként is egy kis szám), teknőcünket egy olyan feladat megoldása elé állítjuk, amely végzetes következményekkel járhat számára. Keresnünk kell egy olyan utasítást, amely megoldja ezt a problémát. Egy olyan ciklusszervezésre van szükség, amely feltételhez köti a ciklus folytatását! A ciklus tulajdonképpen egy olyan hurok, amely meghatározott számszor, vagy egy feltételtől függően, végrehajt egy utasítást, vagy utasításcsoportot.

Ha a W-nyelvben írt programjainkra visszaemlékszünk, ott is találkozhattunk már azzal a megoldással, amikor feltételhez kötöttük a folytatást.

A ciklus belső utasításait ciklusmagnak nevezzük!
Nézzünk egy példát:

```
> eljaras negyzet :oldal
^ ism 4
^ elore :oldal        ciklusmag
^ jobbra 90           ciklusmag
^ ismvege
^ vege
```

A belső ciklusmagot négyszer ismételjük! A hurok tehát négyszer hajtódik végre, a ciklus meghatározott számban hajtódik végre. De mi a megoldás, ha nem tudjuk előre meghatározni hányszor is hajtódjon végre a ciklusmag?

A LOGO tolmácsunk tartogat olyan utasításcsoportot, amellyel a probléma megoldható.

hurok => Ciklus kezdete

.

ciklusmag

.

amig feltétel => Tevékenység ismétlését eldöntő utasítás

A ciklusmag mindaddig ismétlődik, amíg a feltételben megfogalmazott állítás igaz!

Nézzünk egy példát:

amig :oldal < 100

A tevékenységet mindaddig ismétli, amíg az ":oldal" változó tartalma kisebb, mint 100.

A feltételben megfogalmazott állításokat relációnak nevezzük! Nézzük, milyen relációjeleket használhatunk!

< kisebb

> nagyobb

= egyenlő

A ciklusmag tehát mindaddig ismétlődik, amíg állításunk igaz! Most nézzünk egy példát a gyakorlati felhasználásra! Példaként alakítsuk át korábbi "spiral" programunkat!


```
> eljaras spirál :ol :sz
^ hurok
^ vonal :ol :sz 1
^ legyen :ol :ol + 5
^ amig :ol < 200
^ vege
```

Eljárásunk soha nem lép túl a képernyő 200 soránál nagyobb értéket! Kis módosítással még arra is alkalmassá tehetjük, hogy a növekmény (esetünkben ez mindig 5), amit az oldalhoz hozzáadunk paraméterezzük. Ezáltal sokrétűbben használható.

Nézzük a megoldást! (:nov => növekmény)

```
> eljaras spiral :ol :sz :nov
^ hurok
^ vonal :ol :sz 1
^ legyen :ol :ol + :nov
^ amig :ol < 200
^ vege
```

Figyelmesen kövessük nyomon a működését! Próbáljuk ki különböző paraméterekkel! A paramétereket minden esetben írjuk le a füzetbe! Válaszuk ki, milyen paraméterekkel kaptuk a legizgalmasabb ábrát, és írjuk le a könyvünkbe!

_____

_____

_____

Feladatok:

1. Alakítsuk át a következő eljárást a "spiral"-hoz hasonlóan.

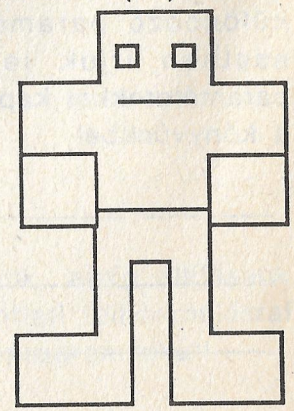
```
> eljaras proba :oldal :szog :is
^ ism :is
^ negyzet :oldal
^ jobbra :szog
^ legyen :oldal :oldal + 2
^ ismvege
^ vege
```

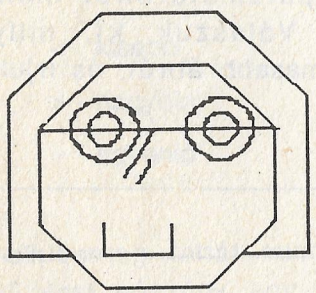
2. Mielőtt elindítanánk az eljárást, határozzuk meg 3 esetre milyen paraméterekkel, milyen ábrát fog rajzolni! Rajzoljuk le a füzetbe!

Ha kész a megoldás ellenőrizzük teknőcünkkel!

Az új ciklusszekvező utasítást nem minden esetben kell alkalmazni! Ha biztosan tudjuk, hogy egy "ciklusmagot" hány alkalommal kell végrehajtani, akkor felesleges a "hurok" utasítás alkalmazása, sokkal célszerűbb az "ism" utasítás használata! Mindig az adott feladattól függ, melyik ciklusszervezést célszerű használni!

Köteles Tamás (4.d)





Tirják Boglárka (4.d)

Hibakaresés, változók tartalmának kiírása

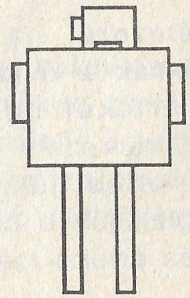
Eljárásaink használata közben gyakran találkozunk olyan esetekkel, amikor az általunk elkészített eljárás bizonyos esetekben hibásan dolgozik, vagy egyáltalán nem azt hajtja végre teknőcünk, amire "megtanítottuk". Ilyen esetekben legtöbbször értetlenül nézzük teknőcünket, vagy kétségbeesetten "kotorászunk" fejünkben, keresve a megoldást, mi lehet a hiba.

Ha a hiba "**durva**", például: => *elre 50*, akkor a hiba helye azonnal meghatározható, tolmácsunk jelzi is, nem tud vele mit kezdeni, sőt felajánlja "segit"ségét.

Valóban, ha begépeljük "segit", akkor az összes utasítást, parancsot kiírja, amit csak ismer. Olyan utasítást azonban, amit eddig nem beszéltünk meg, ne próbáljunk ki, mert következményeit nem ismerjük. Könnyen "VÉGZETES" lehet tolmácsunk számára!

Visszakanyarodva a hibakereséshez, megállapítottuk, a "durva" hibák egyszerűen javíthatók! Ezek általában elírások, hibás utasítások.

A hibák jelentős része azonban nem a programokban, hanem a kis programozók gondolkodásában keresendő. Ezeket a hibákat javítani keserves vállalkozás, mégis meg kell tenni! Az ilyen tipusú hibákat logikai hibáknak nevezzük.

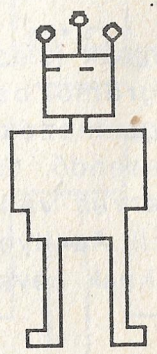


Hogyan javítsuk hibáinkat!

- Meg kell néznünk a programlistát! (E nélkül a javítás elképzelhetetlen!)
- Ha a lista megvan, akkor ki kell próbálni a begépelt paraméterekkel a programot. Természetesen ez már nem a géppel, hanem papíron történik! Ugy is mondhatnánk végrehajtjuk a programot. Ügyeljünk arra, hogy a változók értékét mindig a programnak megfelelően számoljuk ki, és jegyezzük fel! Ha egyszerre több eljárást is tartalmaz a program, akkor valamennyi eljárást be kell helyettesítenünk!
- Az egyes utasításokat lépésről lépésre hajtsuk végre! (Ezt tanácsos a füzetben, vagy milliméterpapíron rögzíteni.)
- A változók, paraméterek tartalmának ismerete is nagyon fontos. Tartalmát a következő paranccsal nézhetjük meg!

valtozok => A parancs hatására a tolmács kiírja az összes használt változót, paramétert és azok jelenlegi tartalmát! (A szóköz billentyű lenyomásával léphetünk a következő változó kiírására.)

Célszerű, ha ezeket leírjuk, mert azoknak a változóknak, paramétereknek az értékét tartalmazza, ahol a hiba bekövetkezett! Ha ezeket az értékeket helyettesítjük be, valószínűleg megtaláljuk a hibát (ha ezt egy változó hibás értéke idézte elő!)



Természetesen nem tehetők felelőssé minden esetben a változók! Ilyenkor nincs más hátra, mint következetesen lépésről lépésre végrehajtani az egyes utasításokat, különös figyelemmel a hiba bekövetkezésekor kapott változóértékekre!

A hibakeresés kemény, fárasztó munka. Nem egyszerű feladat saját gondolkodásunkban megtalálni a hibát. Legjobb megoldás a megelőzés! Ezért minden esetben próbáljuk ki papíron, vajon azt fogalmaztuk meg teknőcünknek, amit szerettünk volna, vagy valami hiba csúszott elképzelésünkbe. Ezt nevezik **tesztelés**nek. Teknőcünkkel csak kipróbált, tesztelt eljárásokat hajtsunk végre! Igy is előfordulhatnak hibák, és nem marad más hátra mint a hibakeresés.

Ha logikailag szerintünk hibátlan a megoldás, ezt teknőcünkkel ki is próbálhatjuk parancs üzemmódban! Amennyiben itt is kiállja a program a próbát, csak akkor készítsük el az eljárást!

A változók értékeit kiirathatjuk a képernyőre az;

ird 'lista' parancs segítségével.

A lista vagy változónév, vagy matematikai kifejezés lehet. Például: ird :oldal vagy ird 10 + 22

Ez utóbbi esetben programterveink esetleges számításánál is felhasználhatjuk! Mivel mi elsősorban rajzokat, grafikus ábrákat készítunk, a tervezett munka a megelőzés egyetlen biztosítéka! Ezért különösen fontos, hogy rajzainkat megtervezzük! Rajzaink már igazi rajzok, tervezésük azonban nélkülözhetetlen!

Színesítsük a programot!

A LOGO 2.9-es változata lehetővé teszi számunkra a keret, a háttér és a vonalak színének megváltoztatását is a Commodore +4-es gépeken. A többi LOGO 2.9-es változat (TVC, IBM, ...) elképzelhető, hogy nem ismeri ezt az utasításcsoportot. Próbáljuk ki, és ha nem látunk változást, akkor nyugodtan lapozzunk át a fejezeten. A keret színét a;

keret A B

paranccsal változtathatjuk meg.

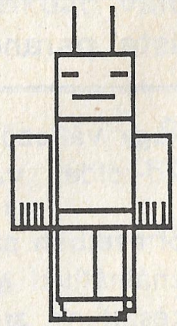
- Az "A" azt a számot jelöli, amilyen színűre a keretet szeretnénk színezni. Ez a szám 1-16 -ig jelöli az egyes színek kódját.
- A "B", a színhez tartozó fényerősséget állítja be! Értéke O-7 -ig. Minél kissebb a szám, annál sötétebb a szín.

A háttér színét, úgy is mondhatjuk a papír színét a;

papir A B

paranccsal állíthatjuk be. Az "A" és "B" ugyanazt jelenti, mint a korábbiakban!

A vonal színét a;



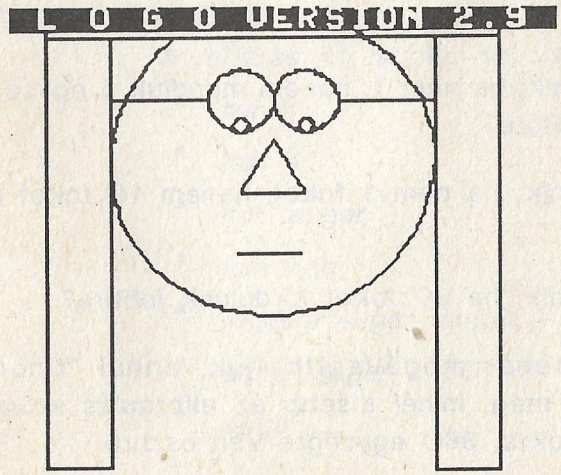
Kovács Károly

tinta A B

paranccsal állíthatjuk a megfelelő értékre.

Használatával nem árt, ha óvatosan bánunk, mert meglepetéseket okozhat! Célszerű a meglepetések elkerülése érdekében, ha rajzainkat egy színnel készítjük el, vagy ha mindenképpen szeretnénk több színt is használni, ezt feltétlen próbáljuk ki az alkalmazás előtt!

Uj parancsaink használatát azonban nem árt, ha alaposan átgondoljuk! A fekete hátteren majdnem minden szín és árnyalat látszik. A képernyő nem vibrál, sokkal kevésbé fárasztó néznünk! Ha élénk színeket választunk "papírnak" és "keretnek", szemünk könnyebben kifárad, könnyezik és előbb-utóbb szemüveg kerül a szemünkre. Ezért a keret és a papír színét nem célszerú változtatni.



Bene Katalin

4.d

Kör rajzolása

Már harmadik osztályban ismert volt, hogy a kör 360 fokos. A kör rajzolására is megtaníthatjuk teknőcünket. Elemezzük a problámát!

Ha egy lépést lépünk előre, majd jobbra fordulunk 1 fokot, akkor 360 lépés után az eredeti, a kindulási helyünkre jutunk vissza. (Aki nem hiszi, próbálja ki az udvaron.) Programunk tehát így nézne ki:

```
> eljaras kor1
^ ism 360
^ elore 1
^ jobbra 1
^ ismvege
^ vege
```

Feladatok:

1. Mi történik, ha nem 1, hanem mondjuk 5 egységet megy teknőcünk előre?

2. Mi történik, ha nem 1 fokot, hanem 10 fokot fordulunk jobbra?

3. Mi történik, ha 90 fokot fordulunk jobbra?

A körről tehát megállapíthatjuk, annál "finomabban" rajzolhatjuk meg, minél kisebb az elfordulás szöge. A kör tehát 360 fokra, 360 egységre van osztva.

A kör másik fontos jellemzője, a mérete. A kört egyértelműen a középpontja (O) és sugara (R) adja meg! A sugár a kör középpontjától a kör széléig (körívig) tart. Az átmérő a kör középpontján áthaladva felosztja a kört 2 egyenlő részre! A következő parancsot nem mindegyik teknőc hajtja végre. Ekkor nincs mit tenni, mint megtanítani a "butuska" teknőcünket.

kor sugár => Hatására a teknőc az aktuális pozíciótól kezdve egy sugár egységű kört rajzol a képernyőre úgy, hogy a teknőc a kör szélén (érintőjén) marad!

Az új utasításra az IBM és a C64-es gépeken meg kell tanítani a teknőcöt. Ennek az elkészítése nem egyszerű feladat! Mire használható az új utasítás? Segítségével rajzolhatunk autókat, bohócot, ... stb. Igazán szabadjára engedhetjük fantáziánkat. A megvalósítás egyedül rajtunk áll! A kör sugara akár paraméterezve is lehet. Ha ezt sikerül megvalósítani, akkor a négyzethez hasonló eljárással érdekes ábrákat hozhatunk létre.

```
> eljaras k1 :sugar :sz :is
^ hurok
^ ism 4
^ kor :sugar
^ ismvege
^ legyen :sugar :sugar + 10
^ amig :sugar < 80
^ vege
```

Számítógépek grafikus alkalmazása a gyakorlatban

Tanulmányaink során ízelítőt kaphattunk számítógépünk grafikus képességeiről, és egy grafikus programozási nyelvről, a LOGO-ról. Nagyobb számítógépek is rendelkeznek ilyen, vagy az általunk ismertnél lényegesen jobb grafikával, grafikus programozási nyelvvel. A gyakorlati életben a grafikus ábrázolásnak igen nagy jelentősége van. Gondoljunk csak a mérnökökre, akik megtervezik eszközeinket. Ök általában egy erre a célra kifejlesztett grafikus programrendszert a CAD-ot használják. Segítségével autókat, repülőgépeket, hidakat, házakat lehet gyorsan és pontosan megrajzolni. Az így elkészült rajzokon különböző változtatásokat is végezhetnek. Kipróbálják, a számítógép segítségével, hogyan viselkedne az autó egy ütközés esetén, vagy hogyan viselkedne a híd különböző terheléseknek kitéve. Ezek az adatok nagyon fontosak, hogy egyre jobb és tökéletesebb berendezéseket hozzon létre az ember, a lehető legkevesebb költséggel. A rajzok még nem kész tárgyak, a rajtuk végzett kisérletek költsége elenyésző ahhoz képest mennyibe kerülne egy kész hídon vagy repülőn kisérletezni. A számítógépen használt úgynevezett CAD rendszerek tehát nagymértékban megkönnyítik a tervezést!

A számítógépes grafika azonban a gyógyászatban is egyre nagyobb teret hódít! Különböző eszközöket kapcsolva hozzá, a számítógép testünk rejtett titkainak megfejtésében van az ember segítségére. A mi általunk használt LOGO-nyelv grafikai lehetőségei szerénynek mondhatók ezekhez a berendezésekhez képest. Mégis ízelítőként

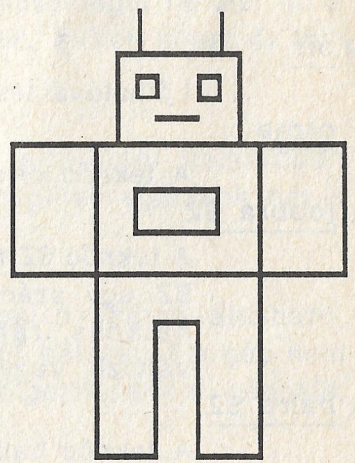
tanulságos kirándulás volt a számítógépes grafika világában.

A LOGO nyelvvel való ismerkedésünknek itt a végére értünk. Lehetőségeit nem használtuk ki, akár évekig kalandozhatnánk a LOGO birodalmában. Arra azonban volt alkalmunk, hogy kitapasztaljuk főbb lehetőségeit. Fontos, hogy tudjuk, a hétköznapi élet számos területén találkozhatunk a számítógépes grafikával. A tervezést egy CAD névre "keresztelt" programrendszer segíti! Vagy gondoljunk a rajzfilmekre, mesefilmekre, vagy a fantasztikus filmekre, amelyeket ma már szintén számítógépek segítségével készítenek.

Mi is újra ellátogatunk majd a grafika világába, egy új programnyelv, a BASIC segítségével. Ezzel a programnyelvvel és lehetőségeivel is megismerkedünk, mert a mikroszámítógépek szinte mindegyike ezen a nyelven programozható.

Feladat: Gyüjtsük össze milyen területen használják a számítógép grafikus lehetőségeit!

Galisz Virág 4.d



segit

Felsorolja a LOGO utasításokat, a teknőc szótárát.

szivacs

Letörli a grafikus képernyőt.

kozep

Középre (160,100) helyezi a teknőcöt.

toll

A teknőc vonalat húz maga után (leteszi a tollát).

tollne

A teknőc nem hagy nyomot maga után (felemeli tollát).

elore BB

A teknőc előre megy BB egységet, ahol BB egy szám, vagy változó, vagy paraméter.

orr

Láthatóvá teszi a teknőc orrát.

orrne

A teknőc orra nem látható.

jobbra SZ

A teknőc SZ fokkal jobbra fordítja az orrát, ahol SZ egy szám 0-360-ig. Ez határozza meg a haladás irányát. SZ a számon kívül lehet változó, vagy paraméter is.

balra SZ

A teknőc balra fordul SZ fokkal. Lásd "jobbra".

helyez XX YY

A teknőcöt XX YY koordinátáju pontra helyezi
0<= XX <= 319
0<= YY <= 199.

eljaras RPOBA [:paraméternév 1 [:p2 ...]]

A teknőcöt "megtanítjuk" a mögötte felsorolt utasításokra. Az utasítás csoportra PROBA névvel hivatkozhatunk. Természetesen ide más nevet is írhatunk. A zárojelbe írt paraméterek használata nem kötelező.

vege

Az eljárásokat ("a tanulást") ezzel a paranccsal zárjuk.

ment név

Kazettára, vagy lemezre menti a megadott névvel az összes eljárást. A kazetta alapértelmezés. Ha a név "d" betűvel kezdődik, akkor lemezre menti az eljárásokat. Az eljárás neve maximum 10 karakter (betű vagy szám) lehet.

tolt név

A megadott nevű eljáráscsomagot betölti. Ha a név "d" betűvel kezdődik, akkor lemezről tölti be.

program sorszám

A megadott sorszámú eljárástól kezdődően kiírja a képernyőre a meglévő eljárások nevét és azok listáját.

lemez 8/9

A lemezegység azonosító száma állítható. Alapértelmezés 9 (TC-NET hálózat). Ha gép nem az említett hálózaton dolgozik, akkor gépeljük be, "lemez 8".

katalogus

Hatására a LOGO tolmács kiírja a lemezen található programokat (ha a név rövidebb 8 karakternél, akkor inverz formában látjuk a LOGO eljáráscsomag nevét).

ujra

Törli az összes LOGO eljárást.

ism N

Eljárásrészek ismétlését hajtja végre N-szer. N egy szám, vagy változó, vagy paraméter. Az utasítás egy ismétlődő tevékenység kezdetét jelzi, amit ciklusnak nevezünk. Az utasítást követő utasításokat ciklusmagnak nevezzük.

ismvege

Zárja a hozzá legközelebb levő nyitott ismétlést (ciklust). Egy ismétlés mindaddig nyitott, amig a ciklusmag N -szer végre nem hajtódott.

fejszog Szög

A teknőc orrának irányát állítja be. "Szög" fokban adott szám, változó, vagy paraméter.

legyen

Értékadó utasítás. A mögötte levő változónak, paraméternek új értéket adhatunk, vagy tartalmát módosíthatjuk. Pl: legyen :ol 5

kivesz név

A megadott nevű eljárást kiveszi az eljárás csomagból. Úgy is mondhatjuk, az eljárás törlésére szolgál.

teljeskep (C+4)

A parancs a teljes grafikus képernyőt mutatja. A parancs mindaddig érvényben van, amíg a "felkep" parancsot be nem gépeljük, ezt azonban "vakon" kell begépelni!

felkep

Alapértelmezés. A LOGO az alsó 5 sort szöveges üzemmódban használja. Itt látható a begépelt parancs.

valtozok

A teknőc által használt összes változót és azok tartalmát kiírja a képernyőre.

hurok

Feltételhez kötött ismétlés, ciklus kezdete.

amig feltétel

Zárja a feltételhez kötött ciklust. Ez a feltétel valamilyen változó értékéhez kötött kifejezés. Feltételek pl.:

:valt < 10 :valt = 10 :valt > 10

A hurok és az amig közötti utasítások addig ismétlődnek, amig az állítás IGAZ! Ezt a példában a :valt paraméter határozza meg.

ird

Művelet eredménye, változó tartalma iratható ki a képernyőre. Csak "felkep" üzemmódban hatásos.

nyomtasd szöveg

A megadott "szöveget" a képernyőre írja. Csak "felkep" üzemmódban hatásos.

torol

A parancs hatására az összes további rajzoló utasítás törlést valósít meg. (Főleg akkor használjuk, ha valamit hibásan gépeltünk be.)

huzz X,Y

A teknőc áthelyezése más koordinátára. X és Y egy-egy szám, vagy változó. Lényeges kölönbség a "helyez"-zel szemben, hogy a teknőc nyomot hagy maga mögött, ha a toll be van kapcsolva. A teknőc orra (a fejszög) nem változik.

urit

Törli az összes változó és paraméter nevét és tartalmát.

sorveg

Soremelést hajt végre. Akkor használjuk, ha a kiíratást nem akarjuk folyamatossá tenni. Csak "felkep" üzemmódban hatásos.

ha

Feltételvizsgálat. A kulcsszót egy feltételnek (állításnak) kell követnie. Feltétel pl.:

ha :fiok < 10 utasítás(ok);

ha :fiok = 10 utasítás(ok);

ha :fiok > 10 utasítás(ok);

A feltétel után egy vagy több utasítás következhet. Az utasításokat szóközzel választjuk el egymástól. A feltételhez kötött utasításcsoport végét a ";" -vel zárjuk.

;

Használata kötelező! Zárja a "ha" feltételhez kötött utasítást, vagy utasításokat.

*** kor sugár**

Az utasítás megadott sugarú kört rajzol a képernyőre. A kör helyzetét a teknőc orrának iránya határozza meg. A sugár vagy szám, vagy változó, vagy paraméter.

*** ellipszis X sugar Y sugar (C+ 4)**

Az "X sugárral" és az "Y sugárral" megadott ellipszist rajzol a képernyőre. Az "X sugár" és az "Y sugár" vagy szám, vagy változó, vagy paraméter. A teknőc orrának iránya határozza meg az ellipszis helyzetét.

*** j.iv X sugár Y sugár fok (C+4)**

A megadott értékekkel a teknőc orrának megfelelő szögben jobb irányban ívet rajzolhatunk. A teknőc pillanatnyi irányához (elfordulási szögéhez) viszonyítva rajzolhatunk a megadott "fok"-kal ívet. Mindhárom kifejezés lehet szám vagy változó vagy paraméter.

*** b.iv X sugár Y sugár fok (C+4)**

Az utasítás a "j.iv"-hez hasonlóan működik, annyi különbséggel, hogy az ívet a teknőc orrához viszonyítva balra rajzolja.

papir szín fényerő

A teknőc a megadott színkódnak (1-16 -ig) megfelelő és a megadott fényerőre (0-7) váltja a hátteret, azaz a papírt. (A színkódok a BASIC PRINT utasításának megfelelők.) Mindkét érték lehet szám, változó, vagy paraméter.

keret szín fényrő (C+4)

A keret színét állíthatjuk be, a "papir" utasításban leírtak szerint.

!

A parancs a grafikus képernyőt nyomtatja ki a nyomtatóra. TC-NET hálózaton NEM használható, csak a tanári gépen! Hálózat nélküli gépen a parancs hatásosan működik, amennyiben van nyomtató csatlakoztatva.

bye

Kilépés a LOGO nyelvnől a szg. visszakerül a BASIC tolmács vagy a DOS felügyelete alá.

gepi cím

Az utasítás saját gépikódú rutinok hívását teszi lehetővé. A rutinokat a LOGO betöltése előtt kell betölteni vagy begépelni. Az utasítás használata csak azoknak ajánlott, akik alaposan ismerik a C+4, C64, vagy TVC arhitektúráját, szabad tárterületeit, és jártasak a gépi kódú programozásban. Azért maradt szótárunk végére az utasítás ismertetése, mert mi ezt nem használjuk.

:veletlen

A LOGO foglalt változója. Egy véletlen egész számot kapunk 1-10 -ig. Ha például beszorozzuk 10-zel akkor 10-100 -ig kapunk véletlen számot.

* Az így jelölt utasításoknál a teknőc az érintőre kerül. A kör, vagy az ellipszis esetén a teknőc iránya 90°-kal jobbra fordul, az ívek esetén a teknőc iránya a rajz irányának megfelelően áll be.

A LOGO 2.9-es egyszerre maximum 50 eljárást tárol. Új eljáráscsomag betöltésekor a tárban levô eljárások törlődnek.

Maximum 50 változónevet tárolhatunk, illetve használhatunk egy eljárásban, illetve eljáráscsomagban. A változónevet csak akkor tárolja a LOGO 2.9-es, ha az eljárást elindítjuk. Változónevek parancs üzemmódban nem használhatók.

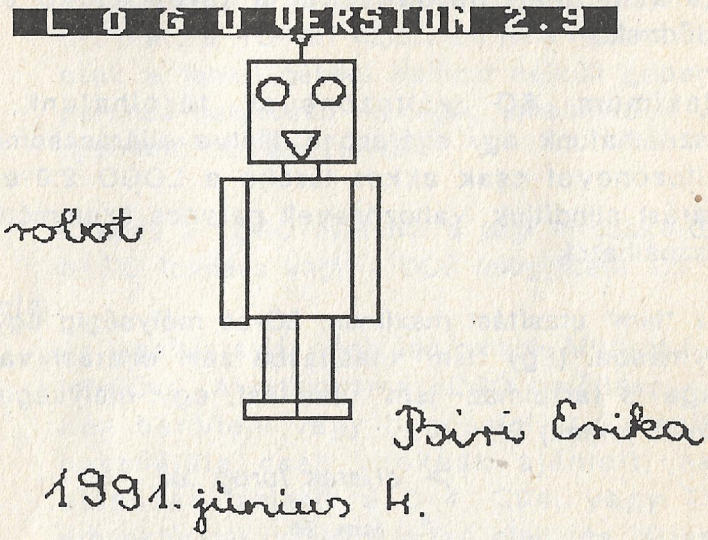
Az "ism" utasítást maximum 20-as mélységig ágyazhatjuk egymásba. (Egy "ism" utasításba zárt eljáráshívás, amely maga is tartalmaz "ism" utasítást, egy mélységet, illetve szintet jelent.) Pl:

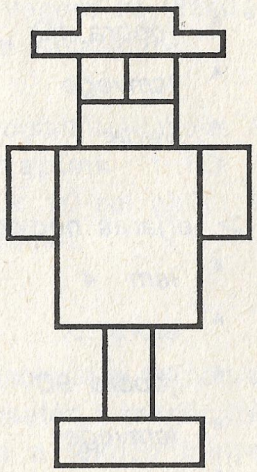
```
> eljaras forog :ol
^ ism 36
^ negyzet :ol
^ jobbra 10
^ ismvege
^ vege

> eljaras negyzet :ol
^ ism 4
^ elore :ol
^ jobbra 90
^ ismvege
^ vege
```

Ne feledjük, a LOGO nyelvben a szóköz billentyű választja el az egyes utasításokat

A változónevek mindíg kettősponttal ":" kezdődnek.





Dobai Gergely 4.c

Határozd meg, mit csinálnak a következô eljárások!

1,

```
> eljaras kezd
^ szivacs
^ kozep
^ toll
^ vege
```

2,

```
> eljaras te
^ elore 50
^ jobbra 90
^ elore 100
^ jobbra 90
^ elore 50
^ jobbra 90
^ elore 100
^ jobbra 90
^ vege
```

3,

```
> eljaras ha
^ elore 50
^ jobbra 120
^ elore 50
^ jobbra 120
^ elore 50
^ jobbra 120
```

4,

```
> eljaras ga
^ te
^ elore 50
^ jobbra 30
^ ha
^ jobbra 180
^ elore 50
^ jobbra 180
^ vege
```

5,

```
> eljaras fo
^ ism 36
^ te
^ jobbra 10
^ ismvege
^ vege
```


6,

```
> eljaras von2 :ol1 :no :sz
^ hurok
^ elore :ol
^ jobbra :sz
^ legyen :ol1 :ol1 + :no
^ amig : ol1 < 100
^ vege
```

_ _ _ _ _ **itt kell levágni a nevezési lapot** _ _ _ _ _

Versenyfelhívás!

A legjobb eljárásom:

Aki a legsikeresebb működő LOGO 2.9-es eljárását (és az ahhoz kapcsolódó részeljárásokat) beküldi minden év május 15-ig, az munkájával versenyen vesz részt! Az elsô helyezett nyereménye 10 db mágneslemez. A borítékba helyezd el az eljárás leírását, és a kitöltött nevezési lappal együtt add fel időben a postára. Junius elején értesülsz az eredményről! Számítunk rád!

Nevem:_____

Életkorom:_____

Címem:_____

Az eljárásokat milyen gépen írtad? _____

Iskolám neve és címe:_____

_____

Tartalomjegyzék

LOGO programozási nyelv (Teknőc grafika)	1
Milliméterpapír használata, pozicionálás	7
Forgás a teknőccel!	9
Háromszög rajzolása	13
Saját rajzok készítése!	17
Tanítsuk meg a teknőcöt!	20
Hogyan mentsük, töltsük be eljárásainkat?	23
Eljárások mentése	23
Eljárások betöltése	24
Könyvtárkezelés	25
Eljárások paraméterezése	26
Eljárások összefűzése, listázása, törlése	29
Eljárások listázása	30
Eljárások törlése	31
Tevékenységek ismétlése	32
Kalandozás az ismétlések birodalmában	35
Matematikai műveletek a LOGO nyelvben	38
Értékadás	38
Feltételes ciklusok	41
Hibakaresés, változók tartalmának kiírása	45
Hogyan javítsuk hibáinkat!	46
Színesítsük a programot!	48
Kör rajzolása	50
Számítógépek grafikus alkalmazása	52
LOGO szótár	54
Hasznos tanácsok	61
Törd a fejed	63
Versenyfelhívás!	65

Eddig megjelent kiadványaink:

- A C+ 4 üzembehelyezése és alapvetô parancsai I.
- A C+ 4 üzembehelyezése és alapvetô parancsai II.

(A jegyzetekben leírtak bármely COMMODORE gépre érvényesek.)

- A Videoton TV-computer üzembehelyezése és alapvetô parancsai I.
- A Videoton TV-computer üzembehelyezése és alapvetô parancsai II.
- Az IBM-PC üzembehelyezése és a DOS alapfokon I. + mágneslemez
- Az IBM-PC üzembehelyezése és a DOS alapfokon II. + mágneslemez

- A WORD programnyelv

(Elsô lépések a programkészítés felé. A programnyelv C64, C128, C+ 4, TVC és IBM gépeken egységes utasításkészlettel és formai megjelenéssel díjtalanul megkapható a kiadványban található megrendelôért. A programnyelv magyar nyelvû, mondat szerkezetû.)

- Ismerkedés a teknôccel

(Látványos grafika. A magyar nyelvû LOGO 2.9-es programnyelv, beépített segítségnyújtással. C64, C128, C+ 4, TVC és IBM PC gépeken egységes utasításkészlettel és formai megjelenéssel. A jegyzethez tartozó mágneslemezen a programnyelv található.)

Tervezett kiadványaink, a korábbi jegyzetekre alapozva:

- HOME computer.
- Basic algoritmusok BASIC nyelven.

Az oktatási módszert átvevô iskolák díjtalanul megkapják a tantervet, tanmenetet, tanári segédletet évfolyamonként.

A jegyzetek az APC-Stúdió BT.-nél megrendelhetôk, elôjegyezhetôk. Cím: 5700, Gyula, Kerék u. 9