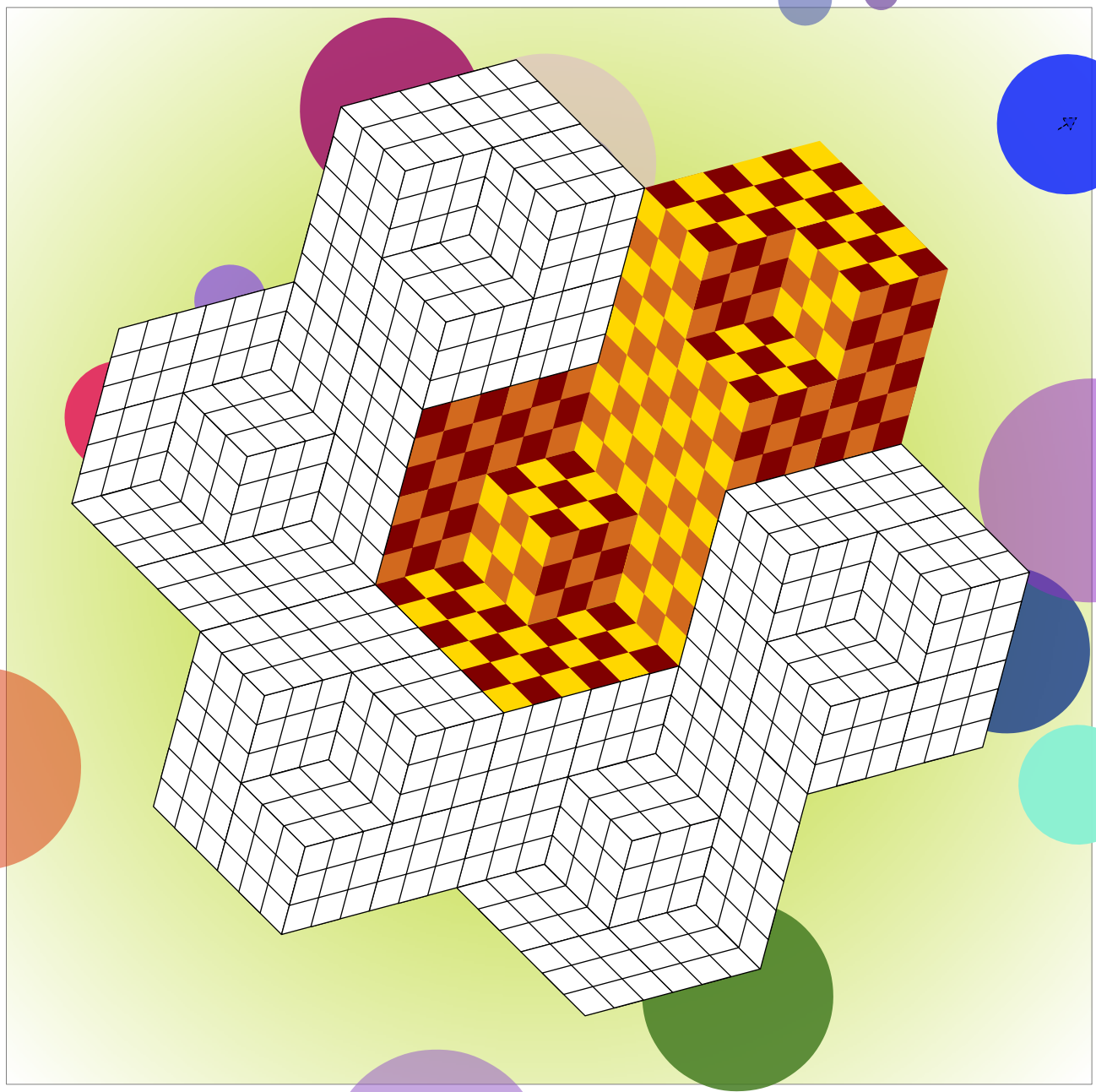
Németh László

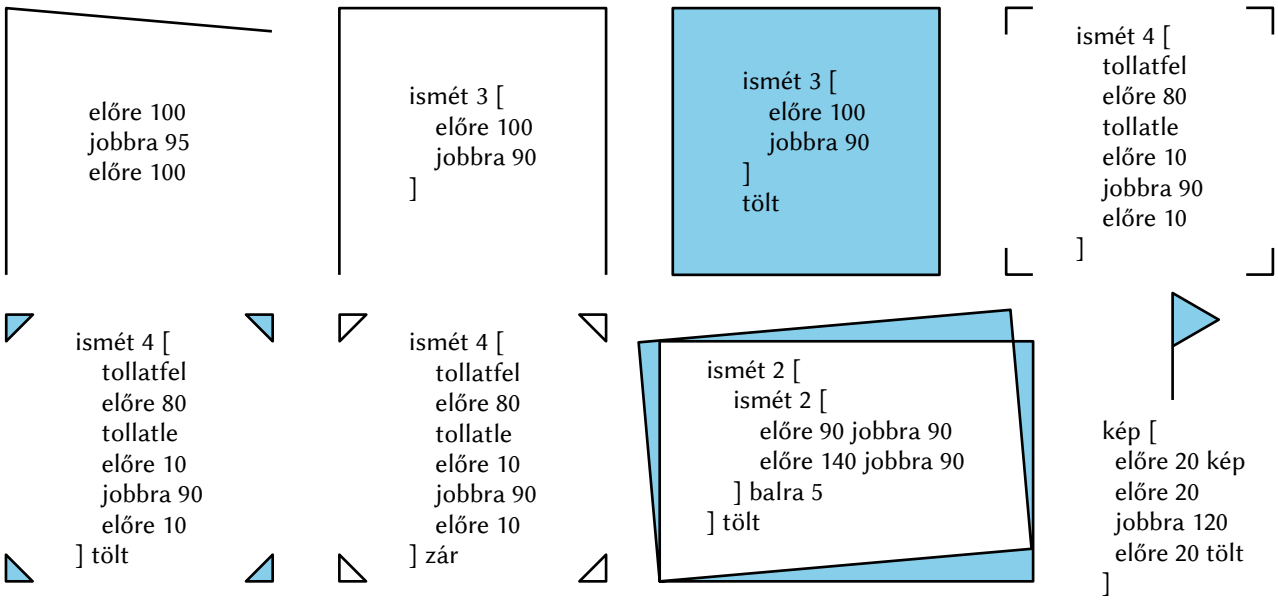
SZÁMÍTÓGÉP-PROGRAMOZÁS MINDENKINEK

LIBRELOGO

TEKNŐCGRAFIKA • ALGORITMUSOK • ADATSZERKEZETEK



FSF.hu Alapítvány, 2012



„fekete”	„világosszürke” „ezüst”	„szürke”	„fehér”	„sötétbarna”
„piros” „vörös”	„lila”	„bíbor” „ciklámen”	„zöld”	„világoszöld”
„olajzöld”	„sárga”	„sötétkék”	„kék”	„kékeszöld”
„ciánkék” „cián”	„rózsaszín”	„világospiros”	„narancssárga” „narancs”	„aranysárga” „arany”
„ibolyakék” „ibolya” „viola”	„égszínkék” „világoskék”	„világosbarna”	„barna”	**„láthatatlan”**

tollszín „piros”

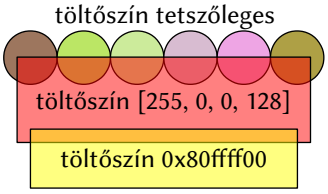
töltőszín „sárga”

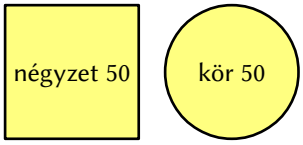
betűszín „kék”

tollszín [0, 0, 255]

tollszín 0xff00ff

töltőszín tetszőleges





tollvastagság 0

tollvastagság 1

tollvastagság 1mm

tollvastagság 0,1in

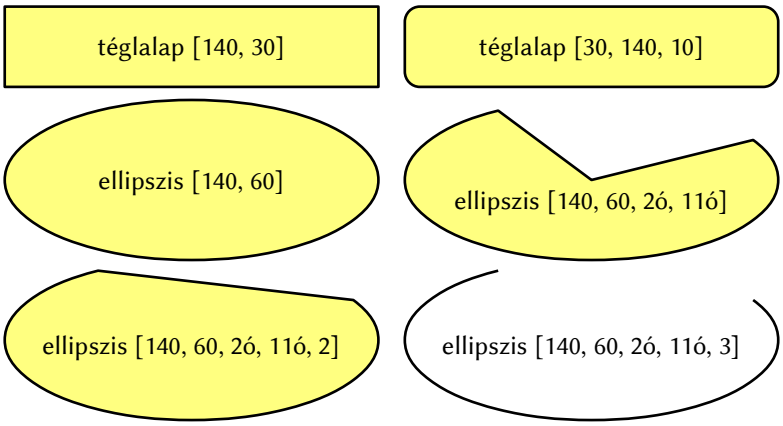
tollstílus „folyamatos”

tollstílus „pontozott”

tollstílus „szaggatott”

tollstílus [3, 1mm, 2, 4mm, 2mm]

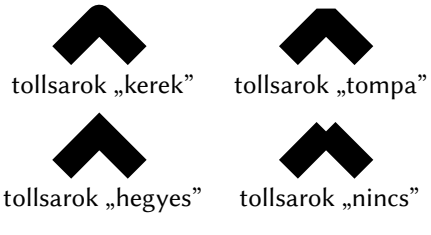
tollstílus [3, 1mm, 2, 4mm, 2mm, 2]

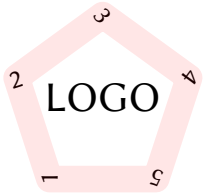


pont

töltőstílus 2

töltőstílus [2, „zöld”, 3pt, 15°]





ismét 5 [
előre 50
címke hányadik
jobbra 360/5
] zár
betűméret 16
szöveg „LOGO”

LibreLogo

betűcsalád „Linux Libertine G”
betűvastagság „félkövér”
betűstílus „kurzív”
fut i „LibreLogo”-ban [
töltőszín [255, véletlen 200, 0]
kör 14 szöveg i
jobbra 90 előre 12 balra 90]
betűvastagság „normál”
betűstílus „álló”

„Az iskolák sokkal inkább nevelő környezetet biztosítanak majd, arra bátorítva a tanulókat, hogy tanulmányozzák és fejlesszék a [GNU] rendszer kódját."*** – Richard Stallman, a szabad szoftver mozgalom alapítója

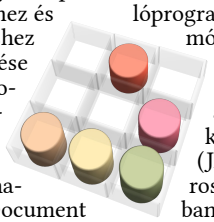
Tartalomjegyzék

Előszó

A LibreLogo programozási környezet és ez a könyv azzal a céllal készült, hogy elősegítse a szabad szoftverek megjelenését, használatát a közoktatásban. Bízom abban, hogy a szabad szoftverek ugyanakkora hatást gyakorolhatnak majd az iskolásokra, mint amelyet az iskolai számítógépek megjelenése okozott valamikor.

Hatodikos voltam 1986-ban, amikor vakáció előtt pár nappal az osztályfőnököm egy, akkor még ritkaságnak számító személyi számítógépet mutatott be az osztálynak az utolsó matematikaórán. Az első találkozás hatására a nyári szünetben – ugyan még számítógép nélkül, egy gyerekeknek szóló tankönyv segítségével – készítettem el első Basic nyelvű programjaimat. Ez az élmény ismétlődött meg jóval később a szabad szoftverekkel kapcsolatban. Őszinte lelkesedéssel csodálkoztam rá arra a mindenki előtt nyitva álló mérhetetlen*** tudásra, amit egy GNU/Linux rendszer és a vele járó, több mint 40 éves Unix kultúra† hordoz. Csatlakoztam a szabad szoftveres közösséghez, és fejlesztéseim ma részei az olyan világszerte ismert programoknak, mint a Mozilla Firefox, Google Chrome és Dokumentumok, Apple Mac OS X, Adobe InDesign, OpenOffice.org vagy LibreOffice.

Mi is a LibreLogo? Egy olyan, a jelenlegi iskolai Logo rendszerekkel rokon programozási környezet és nyelv, amely a honosított utasítások mellé a modern Python programozási nyelvből vesz át elemeket, illetve lehetővé teszi a Python kifejező adatszerkezeteinek használatát is. A LibreLogo ötvözi a Logo teknőcgrafikáját a LibreOffice (vagy az OpenOffice.org) nyomdai minőséget biztosító vektorgrafikus képességeivel, amint ezt a könyv LibreLogóval készült ábraanyaga is bizonyítja.



A LibreLogo a LibreOffice Writerben rajzol, így a képek a szövegszerkesztéshez és kiadványszerkesztéshez egyből rendelkezésre állnak, egérrel módosíthatók, mozgathatók, átméretezhetők, feliratozhatók, egyből kinyomtathatók, ISO OpenDocument (ODF) dokumentumformátumban menthetők, nyomdai szabvány PDF formátumba exportálhatók, vágólapon keresztül új dokumentumba vagy bemutatóba másolhatók, a LibreOffice Draw rajzolóprogramjával egyéb módon módosíthatók (pl. körre feszíthetők, torzíthatók, sőt térbelivé alakíthatók, l. mellékelt kép), raszteres (JPEG, PNG), és vektoros képformátumokban; a DTP standard EPS-ben (Encapsulated PostScript), vagy a Wikipédia által is használt webes standard SVG-ben (Scalable Vector Graphics) elmenthetők.

A LibreLogo izgalmas pedagógiai kísérlet a különböző alap- és emelt szintű informatikai ismeretek egységes keretben történő tárgyalására, és egyben reményeim szerint a LibreOffice-t jól kiegészítő, könnyen elsajátítható eszköz a számítógépes grafikához és kiadványszerkesztéshez.

Németh László

---

*A szabad szoftverekből álló Unix-szerű operációs rendszer, amelynek számos kulcselemét el is készítette a fejlesztést meghirdető Free Software Foundation (FSF). A mai, GNU GPL szabad szoftver licenc alatt kiadott Linux rendszerek FSF által javasolt elnevezése GNU/Linux, tekintettel GNU-s rendszer- és segédprogramjaikra.

**Idézet a GNU kiáltványból (1984), l. http://www.gnu.hu. Az állítást jól példázza a magyar felsőoktatás. A szabad szoftverekkel megismerkedő hallgatók közül sokan ma már nemzetközileg ismert informatikai szakemberek, mint Molnár Ingo (l. Wikipédia), a Linux operációs rendszer egyik vezető fejlesztője, vagy Scheidler Balázs, a világszerte mintegy 850 ezer cégnél üzembe helyezett syslog-ng naplózó szoftver szerzője, és a hálózatbiztonság-technológiai középvállalat, a BalaBit Kft. alapítója és vezetője.

***Becslések vannak a szabad szoftverek értékére vonatkozóan. Csak a Linux operációs rendszer értékét másfél milliárd dollárra becsülték 2008-ban. Egy több ezer programot tartalmazó Linux terjesztés értéke 10 milliárd dollár is lehet, a magyar GDP közel tizede, l. http://www.linuxfoundation.org/sites/main/files/publications/estimatinglinux.html

†Az eredeti szerzők, Ken Thompson és Dennis Ritchie 1998-ban az USA legmagasabb szintű technológiai-innovációs díját is megkapták a Unix és a C programozási nyelv kidolgozásáért.

Bevezetés

A LibreLogo a magyar közoktatásban, sok helyen a felsőoktatásban is használt zárt, licencdíjas, windowsos informatikai oktatóprogramok (Comenius Logo és Imagine Logo) szabad, és szabad operációs rendszereken is futó alternatívája. Mivel egyesíti a Logo és a Python programozási nyelv előnyeit, egyszerűbben oldhatunk meg vele informatikai verseny- és emelt szintű érettségi feladatokat is, mint a zárt Logo rendszerekkel.

2012-től a Python választható programozási nyelv az emelt szintű informatika érettségin. Pythonban a programozási feladatok töredék idő alatt megoldhatók a Pascal, C, C++, Java és Logo nyelvekhez képest. A LibreLogo egyik célja, hogy megkönnyítse a Python nyelv elsajátítását.*

A LibreLogo lehetőségei azonban nem merülnek ki az oktatásban: *interaktív (kézzel is átszerkeszthető) vektorgrafikus ábrákat készíthetünk vele nyomdai minőségben, kiadványszerkesztési céllal.* A LibreLogo jellemzői részletesebben:

Szabad szoftver. Szabadon felhasználható és terjeszthető, forráskódja tanulmányozható és módosítható.

A szabad, más néven nyílt forráskódú szoftverek meghatározó szerepet töltenek be napjainkban, elég, ha csak az olyan ismert, nyílt forráskódra építő szervezetekre gondolunk, mint az Apple, Facebook, Google, Twitter vagy a Wikipédia.** A LibreOffice-on, illetve OpenOffice.org-on kívül olyan népszerű és ismert szoftverek tartoznak ide, mint a Mozilla Firefox vagy a Google Chrome böngészők és a Linux operációs rendszer. Ez utóbbi nemcsak a mai szuperszámítógépek vagy a vállalati kiszolgálók uralkodó operációs rendszere, hanem ügyféloldalon is az élre tört: az okostelefonok piacvezető platformja, a nyílt forráskódú Android valójában egy Linux rendszer, de a riválisiPhone, iPad iOS rendszerének (sőt a Mac OS X-nek) alapja is szabad szoftver, a Darwin névre hallgató operációs rendszer. A Magyarország.hu kormányzati portált több mint 95%-ban szabad szoftverek működtetik, ahogy a hazai önkormányzatok és bíróságok működése is elképzelhetetlen szabad szoftverek nélkül.*** A Gartner 2011-ben publikált felmérésében közel 600 vállalat szerepelt, melyek többségében meghatározó szereppel bírnak a szabad szoftverek. Szabad irodai programcsomagokat, mint az OpenOffice.org és a LibreOffice, a vizsgált vállalatok mintegy negyedénél alkalmaztak. A kutatás egyik legfontosabb megállapítása a szabad szoftverek részesedésének nagymértékű növekedése: a 2006-ban mért kevesebb mint 10%-ról a 2012-re jósolt több mint 30%-ra.† A GNU GPL/LGPL/MPL hármas szabad licence szavatolja a LibreLogo szabad felhasználhatóságát, forráskódjának hozzáférhetőségét és módosíthatóságát minden felhasználó számára.†† A program letölthető a következő címről: http://www.numbertext.org/logo.

Oktatóprogram. A Nemzeti alaptantervben szereplő informatikai tananyag (teknőcgrafika és algoritmusok) oktatására alkalmas programozási környezet.

Teknőcmozgató utasítások, ciklusok, feltételvizsgálat, eljárások, fejlett adatszerkezetek, mindez magyar nyelvű utasításokkal. A LibreLogo egyéb kiemelhető, a közoktatás számára is előnyös tulajdonságai: dokumentumszerkesztő és vektorgrafikus képkezelésének oktatása, művészeti oktatás, a modern és elterjedt Python programozási nyelv adatszerkezeteinek és egyéb tulajdonságainak elsajátítása.

Csereszabatos honosítás. A LibreLogo támogatja a magyar oktatásban elterjedt Comenius Logo és Imagine Logo alapvető utasításait, jelöléseit.

Például a LibreLogo elfogadja a Comenius Logo „tanuld”, és az Imagine Logo „eljárás” (röv. „elj”) utasításait is (az Imagine Logo már nem támogatja a „tanuld” utasítást, tehát az alapvető programszervező utasítások tekintetében sem csereszabatos elődjével, a Comenius Logóval). Egyéb gyakran használt közös parancsok: „előre” („e”), „hátra” („h”), „balra” („b”), „jobbra” („j”), „tollatfel” („tf”), „tollatle” („tl”), „ismétlés” („ism”), „tollszín” („tsz”), „tollvastagság” („tv”), „töltőszín” („tlsz”), „törölképernyő” („törölkép”), „haza”, „elrejt”, „látható”, „eredmény” stb.

Javított honosítás. A LibreLogo helyenként új alternatívák bevezetésével javítja az említett Logo honosítást:

A „tanuld”/„eljárás”, vagyis az eredeti Logo „to” utasítás alternatív honosítása a LibreLogóban az „ez” (l. mellékelt példa).

```
ez teknőc
  címke „🐢”
vége
```

Vektoros rajzolóprogram: a képernyőfüggetlen vektorgrafika (vektoros alakzatok és TrueType, valamint Graphite betűtechnológia), nyomdai mértékegységeket ismerő utasítások segítségével nyomdai minőségű grafikákat készíthetünk†††, szemben az iskolai oktatásban jelenleg használt rossz felbontású raszteres Logo rendszerekkel.

A nyomdai minőségű, a kiadványszerkesztésben egyszerűen felhasználható grafika többek számára nyújtja az alkotás örömét, mint a nagyobb programozási ismereteket igénylő multimédiás és játékprogramozás, amire a jelenlegi iskolai Logo rendszerek a hangsúlyt fektetik.

LibreOffice/OpenOffice.org kiegészítő: az elterjedt LibreOffice (vagy annak eredeti kódbázisa, az OpenOffice.org) irodai programcsomag Writer dokumentumszerkesztőjében készíthetjük és futtathatjuk LibreLogo programjainkat, kihasználva a program magas szintű grafikai képességeit.

Platformfüggetlenség: a program mindenhol fut, ahol a LibreOffice, tehát szabad (FreeBSD, Linux) és zárt (Mac OS X, Windows) operációs rendszereken is.

Idáig központi szoftverlicenc-vásárlás biztosította a Comenius és Imagine Logót, a Microsoft Office irodai programcsomagot és egyéb zárt programokat a magyar iskolák számára. A közeljövőben ez a központi beszerzés megszűnik, hogy a nemzetközi trendeket követve a nyílt alternatívák fokozottan megjelenhessenek az oktatásban is (l. pl. az idézett Gartner jelentést a szabad szoftverek, benne a szabad irodai programcsomagok jelentős vállalati részesedéséről, és ezek folyamatos növekedéséről). A LibreLogo olyan nyílt alternatíva, amely megkönnyíti a nyílt

---

*Vannak tanárok, akik szerint komolytalanná válhat az informatika érettségi, annyira leegyszerűsödik a programozási feladatok megoldása a Python karakterláncokkal, listákkal és szótár adatszerkezettel. Pedig innentől válhat komollyá, hiszen olyan szintű feladatok oldhatók meg kezdő Python programozói tudással, amelyek a professzionális C, illetve Pascal tudással rendelkezőknek is feladják a leckét. A középiskolai informatikaoktatás célja, hogy minél többen legyenek képesek megoldani emelt szintű érettségi feladatokat. Az pedig kifejezetten előny, hogy közben egy elterjedt, komoly programozási nyelvvel ismerkednek meg, amelynek – függetlenül attól, hogy informatikai szakemberek lesznek-e, vagy sem – később is jó hasznát vehetik.

**L. például http://developers.facebook.com/opensource/, http://www.apple.com/opensource/ és http://opensource.apple.com/, http://code.google.com, https://dev.twitter.com/opensource

***Laky Norbert (Fővárosi Bíróság) felmérése szerint a hazai bíróságok többsége szabad irodai programcsomagot használ. Az évi másfél millió bírósági ügyet tekintve ez milliós nagyságrendű dokumentum kezelését jelenti szabad szoftverekkel Magyarországon.

†http://www.gartner.com/it/page.jsp?id=1541414

††http://hu.wikipedia.org/wiki/Szabad_szoftver

†††L. címlap, vagy http://www.numbertext.org/logo/lok_hu.pdf.

irodai programcsomagra és a nyílt operációs rendszerre való átállást az oktatásban is.

Szabványosság: az ábrák, Unicode karakterkódolású feliratok és a Logo programot tartalmazó szöveges dokumentum .odt kiterjesztésű állományba, azaz ISO OpenDocument formátumban menthető, valamint exportálhatók a nyomdai és ISO szabvány PDF-be és (a vektoros képek esetében) a webes szabvány SVG-be.

Interaktív teknőc: a teknőc pozíciója és elforgatása egérrel is beállítható. A teknőc színei és körvonala kijelzi az aktuális tollszínt, töltőszínt, tollvastagságot és a toll felemelését.

Bár a LibreLogo Logo eszköztára tartalmaz teknőcmozgató ikonokat, a teknőc közvetlenül is mozgatható az egérrel: tetszőleges helyre húzható, illetve forgatási szöge (a kijelölésnél automatikusan megjelenő Rajzobjektum tulajdonságai eszköztár Forgatás ikonjára kattintás után) is módosítható. A teknőc az így beállított pozícióban és forgatási szögben halad tovább.

Interaktív grafika: A LibreOffice-ban a LibreLogóval rajzolt élsimított, szabadon nagyítható alakzatokat szabadon elrendezhetjük, átszerkeszthetjük.

Az alakzatot kijelölve módosíthatjuk az alakzat elforgatását, a vonalvastagságot és a -színt. A kitöltésnél akár színárnyalatot is beállíthatunk. Az alakzatra duplán kattintva szöveget adhatunk meg. A LibreOffice beépített súgója magyar nyelven ad a lehetőségekről tájékoztatást.

Fejlett Logo fejlesztőkörnyezet: változtatható lapméret (max. 3 m × 3 m), nagyítás, teknőc-nyomkövetés (képernyő automatikus görgetése programfutás közben). Writer programszerkesztő: szintaktikai ellenőrzés, hibás sorra ugrás, helyesírás-ellenőrzés, több (dokumentumonkénti) rajzlap egyszerre.

Python: a LibreLogo a modern Python programozási nyelvre épül, melynek adatszerkezeteit, könyvtárait itt is elérhetjük. A LibreLogo tömör parancskészlete mögött a Python rugalmassága áll.

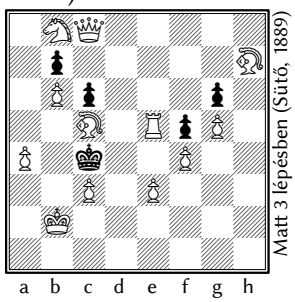
A LibreLogo értelmezője gyakorlatilag egy Python előfordító. A LibreLogo programot az előfordítás után egy Python szál hajtja végre a háttérben. A Python nemcsak a LibreOffice beépített, magas szintű programnyelve, illetve nemcsak az emelt szintű informatikai érettségi legkönnyebben elsajátítható és itt a leghatékonyabb (leggyorsabb fejlesztést lehetővé tevő) programozási nyelve. A kifejezetten oktatási és prototípuskészítési céllal fejlesztett, azóta is folyamatosan fejlődő nyelv és C referenciamegvalósítása, a CPython széles körű népszerűségre tett szert azzal, hogy egyszerűsége ellenére a legkomolyabb célokra használható. Például beépített programnyelve a vezető 3D animációs szoftvereknek, mint az Autodesk Maya, Cinema 4D vagy a szabad Blender; és a Google vállalati adatbázisfelhőjének, az App Engine-nek. További szabad szoftveres példák a LibreOffice-on és OpenOffice.org-on kívül, amelyek programozhatók Pythonban: Scribus kiadványszerkesztő, Fontforge betűtervező, GIMP rajzolóprogram, vagy eleve Pythonban készültek, mint a GNU Mailman levelezőlista-kezelő, Plone tartalomkezelő, Django webes keretrendszer, Bazaar verziókezelő rendszer és az eredeti BitTorrent kliens, de a Microsoft is a Pythonnal (annak egy másik szabad implementációjával, az IronPythonnal) népszerűsíti saját .NET platformját.

Graphite betűtechnológia és betűkészletek: Bár nevükben a Linux szerepel, a LibreOffice Linux Libertine G és Linux Biolium G betűkészleteivel operációs rendszertől függetlenül tudunk különleges betűváltozatokat és betűhatásokat (például VALÓDI KISKAPITÁLISOK, ugráló számok: 1 234 567 890) elérni a LibreLogóban is.

Ilyen különleges betűváltozat a Linux Libertine G valódi tervezett (nem pedig olvashatatlanul lekicsinyített) apróbetűje, amellyel a LibreOffice a professzionális kiadványszerkesztő programok alapváltozatait is túlszárnyalja. (Megfigyelhető e jegyzet nyomtatásra szánt változatában.)

Sőt, a betűkészletek a részben magyar fejlesztésnek köszönhetően kiemelt magyar tipográfiai támogatással bírnak: A felkiáltójel, kérdőjel, kettőspont és pontosvessző elé a magyar tipográfiának megfelelő nagyobb térköz kerül. (Ezt sajnos nemcsak a szövegszerkesztők, még a kiadványszerkesztő programok többsége sem támogatja!) A dokumentumszerkesztők körében amúgy is egyedülálló alapértelmezett ligatúrákat (betűhelyettesítő nyomdai jelek) további magyar jelváltozatokkal egészíti ki: ilyen a kurzív „gy” (l. a *„magyar”* szóban), „gf” (*„megfog”*, *„ágfa”*), „gj” (*„vágja”*, *„megjön”*, *„legjobb”*), valamint az ismertebb, de a betűkészletekben általában nem szereplő „fj” és „ffj” (*„ifjú”*, *„blöffje”*, illetve álló változatban is: „ifjú”, „blöffje”).

A LibreOffice grafikai képességeinek bővítése: Az egyszerű programozási felület, amit a LibreLogo nyújt, jelentős mértékben bővíti a LibreOffice grafikai képességeit, de ezen felül is akad olyan grafikai lehetőség, ami a LibreOffice-ból korábban hiányzott. Ilyen a `pont` utasítással megrajzolható pont alakzat és a pontozott vonalstílus: a LibreOffice pontozott vonalai valójában kis négyzetekből állnak, nem pedig pontokból, mint a LibreLogóé (l. a mellékelt összehasonlítást).



Matt 3 lépésben (Sütő, 1889)

Speciális kiadványszerkesztő: a LibreLogo az előbbiekben felsorolt tulajdonságaival; a képernyőfüggetlen, nyomdai minőségű vektorgrafikával; az olyan nyomdai szabványok támogatásával, mint a PDF; különleges nyelvi elemeivel, mint a közvetlen mértékegység megadás, vagy a hierarchikus csoportosító `kép` utasítás; valamint dokumentumszerkesztőbe ágyazottságával és interaktivitásával speciális kiadványszerkesztési feladatokat is elláthat.

Jó példa erre a LibreLogo sakktáblarajzoló példaprogramja, ahol a sakkbábokat kézzel vagy a lejegyzés megadásával is egyszerűen felhelyezhetjük a táblára, vagy a (például a címoldalon, illetve a mellékelt, a bábokat TrueType betűkészletből, a sakktábla keretét és számozását Logo utasításokkal kirajzoló sakktáblán is megfigyelhető) betűgrafikai képességek.

Egyéb tulajdonságok: a program tömör felépítésű (mindössze 1300 sor Python/PyUNO-ban), könnyen honosítható (összesen 130 szó, illetve egyszerű programüzenet lefordítását igényli).

Várható fejlesztési irány: a jelenleg támogatott angol és magyar mellett további honosítások elkészítése a LibreOffice fordítói közösség segítségével; a LibreLogo programozási nyelvéből még nem elérhető LibreOffice grafikai képességek kihasználása (pl. Bézier-görbék, áttetsző szöveg, átmenetek (szín és áttetszőség), 3D grafika) stb.

Telepítés

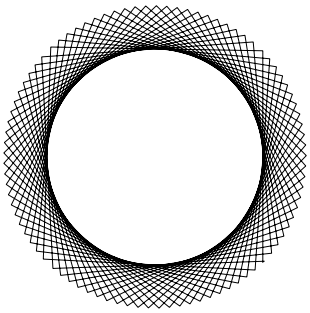
A LibreLogo kiegészítő jelenleg külön telepítendő a LibreOffice-ban az Eszközök » Kiterjesztéskezelő... » Hozzáadás... gomb és a LibreLogo-0.1.oxt állomány kiválasztásával. Telepítés után indítsuk újra a LibreOffice-t. LibreOffice: http://hu.libreoffice.org, LibreLogo: http://www.numbertext.org/logo/

Logo eszköztár

Nyissunk meg egy új, üres szöveges dokumentumot a Writerben. Első telepítés után megjelenik a Logo eszköztár (kikapcsolás: Nézet » Eszköztárak » Logo). Az eszköztár több ikont és egy beviteli mezőt (**Logo parancssor**) tartalmaz. Az ikonok leírása:

- Előre: teknőc 10 pontnyit* előre, beállítástól függően vonalat húzva.
- Hátra: teknőc hátra halad 10 ponttal, beállítástól függően vonalat húzva.
- Balra: teknőc elfordítása 15°-kal balra.
- Jobbra: teknőc elfordítása 15°-kal jobbra.
- Indítás: LibreLogo program (Writer dokumentumban lévő szöveg) futtatása.
- Leállítás: futó program leállítása.
- Haza: teknőc kezdőpozíció és kezdőértékek beállítása.
- Képernyőtörlés: a dokumentum alakzatainak törlése. A teknőc pozíciója és beállításai nem változnak.

Parancssor az eszköztáron



A parancssor kisebb Logo programok beírására, és ismételt végrehajtására ad lehetőséget.

Program futtatása: írjunk be egy parancsot a parancssorba, például **kör 100**, és nyomjuk le az új sor billentyűt (Enter) a parancs végrehajtásához. A beírt parancs nem tűnik el a parancssorból, így az új sor billentyű folyamatos nyomva tartásával ismételten végrehajtható, amivel egyszerű ciklusok kiváltására is alkalmas. A mellékelt (nem méretarányos) ábra is így, az **e 100 b 89** (vagyis **előre 100pt balra 89°**) parancs beírásával, és az új sor billentyű folyamatos nyomva tartásával készült.

Program leállítása: a parancssorból indított, még futó programok az eszköztár Leállítás ikonjával állíthatók le.

Parancssor törlése: hosszabb parancs törléséhez nyomjuk le a Ctrl-A billentyűkombinációt (vagyis a parancssor helyi menüjében lévő Mindent kijelöl parancs gyorsbillentyűjét) a parancssorban, és kezdjük el gépelni az új parancsot.

*Mai nyomdai (DTP vagy PostScript) pont = 2,54 cm / 72, vagyis kb. 0,35 mm.

Programfuttatás

A többsoros LibreLogo programok szerkesztője a LibreOffice Writer dokumentumszerkesztő. Például nyissunk meg egy új Writer dokumentumot, és szövegként írjuk be a következőket (a pontosvesszővel kezdődő sorvégek, vagyis a megjegyzések elhagyhatók):

```
tf                      ; tollat fel
ism 36 [                ; 36-szor ismétel
  tlsz tetsz            ; tetszőleges töltőszín
  kör vszám 8           ; véletlen átmérőjű kör
  e 8 b 10              ; előre 8 pont, balra 10°
]
```

A program indításához kattintsunk a Logo eszköztár Indítás ikonjára. A mellékelt képet a program futtatásával kaptuk.

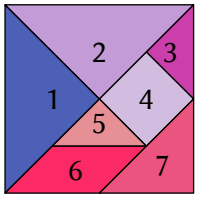
Program leállítása: Hasonlóan a parancssori indításhoz, a Logo eszköztár Leállítás ikonjával szakíthatjuk meg a Logo program futását.

Logo és LibreLogo összevetés

A LibreLogo fejlesztés kiindulásként az általános iskolai tananyagot vette célba, így az abban szereplő Logo teknőcmozgató és egyéb utasítások megegyeznek, illetve alternatívaként használhatók.

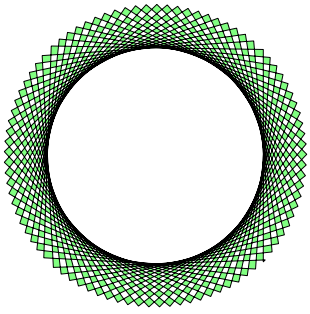
A tananyag miatt került bele olyan utasítás is a LibreLogóba, mint a program lassítására alkalmas „várj”, amire – szemben a gyors raszteres megjelenítést alkalmazó Logo programokkal – a LibreLogóval való ismerkedéshez nincs szükség, mivel az élsimított vektorgrafikus alakzatok rajzolása (új élekkel bővítése) szemmel követhető sebességgel megy végbe.

A fontosabb eltérések, használatbeli különbségek:



A LibreLogo ábrák vektoros alakzatokból állnak. A bal oldali tangram részeit utólag rendeztük teknőc és piton alakba (l. 18. oldal).

Alakzatrajzolás. A LibreLogo teknőcmozgató utasításai vektorgrafikus alakzatokat rajzolnak. Ezek akár rajzolás közben is kijelölhetők, módosíthatók. *A toll felemelése sem jelenti, hogy befejeztük az alakzat rajzolását, mert egy vektorgrafikus alakzat több, nem összefüggő részből is állhat.* Más tollvastagsággal, színnel való rajzolás új alakzat rajzolását eredményezi, egyéb esetben a `kép` utasítással tudunk új alakzat rajzolásába fogni.



Kitöltés. A `tölt` utasítás lezárja az eddig rajzolt alakzatot, és kitölti az aktuális töltőszínnel. Nem szükséges az alakzat belsejébe pozicionálni, mert az alakzat vektorgrafikus leírása alapján történik a kitöltés. Az összetett alakzatok egymást met-

sző, illetve fedő részei összetett minta kialakítását eredményezik, mint ahogy a mellékelt kép (a korábbi ábra kitöltött változata) is mutatja.

Blokk és lista. A LibreLogo megkülönbözteti a ciklustörzset és más utasítássorozatot határoló kapcsos zárójelezést a listákétól: előbbi esetben szóközzel, vagy új sorral kell határolni az utasításoktól a zárójeleket, listáknál pedig szorosan kell záródnia (ez a megoldás visszafelé kompatibilis a Logóval, azaz az egyszerű LibreLogo ciklusok a Logóban is futnak):

```
ism 18 [ b 10 téglalap [10, 200] ]
```

vagy

```
ism 18 [              ; ciklustörzs kezdete
  b 10
  téglalap [10, 200]  ; méret listával
]                     ; ciklustörzs vége
```

de nem

```
ism 18 [b 10 téglalap [10, 200]]
```

Elhagyható és elhagyandó sortörésjel. A LibreLogóban a sortörést jelölő hullámvonalat csak egy utasítás és bemenő értékei külön sorba kerülése esetén kötelező kitenni.

```
előre ~
100                   ; itt kellett sortörésjel
előre 100
előre 100             ; itt nem
```

Egyszerűbb változóhasználat. A Logo változók érték szerinti hivatkozásában szereplő kezdő kettőspont – hasonlóan egy-két Logo megvalósításhoz – elhagyható (csak az egybetűs magyar Logo utasításokra ügyeljünk, mert azok nem lehetnek változónevek).

```
ez régi :i :j :k          ; régi, de használható
  :l = :i + :j + :k
vége

ez új x y z  ; új, de b, e, h, j nem használható
  zs = x + y + z
vége
```

Vesszővel elválasztott listaelemek. A listaelemeket vesszővel kell elválasztani a LibreLogóban:

```
téglalap [100, 200]    ; 100×200 pontos téglalap
```

Kevesebb utasítás. A LibreLogo bár elfogadja a Comenius Logo és Imagine Logo felkiáltójeles értékadó parancsait, nem tesz különbséget ezek és az értéket kiolvasó utasítások között.

```
tv! 10  ; régi szintaxis
tv 10   ; javasolt szintaxis
ki tv   ; megjelenítjük a tollvastagság értékét
a = tv  ; tollvastagság értéke az „a” változóba
```

Alternatív szövegmegadás. A LibreLogo magyar honosítása a karakterláncok tárolására a magyar helyesírásnak megfelelő, a Writer szövegszerkesztő automatikus idézőjelei miatt pedig kézenfekvő idézőjelezést javasolja: **„példa”** (de elfogadja a Logo **"szó**, és a Python **'karakterlánc'** szintaxisát is.

Alternatív utasítás- és adatszerkezetek. A LibreLogo a nehézkes, az Imagine Logo esetében ráadásul túlságosan is változatos Logo programszerkezetek, utasítások helyett kevés számú, de áttekinthető Python programszerkezeteket kínál. Ilyenek az egyszerűen használható listák és szótárak, vagy a listaelemeken, illetve karakterláncon végigfutó `fut -ban/-ben` (a Python `for+in` honosítása) ciklus. Példa az utóbbira:

```
ez felsorol vmi  ; felsoroljuk a bemenet elemeit
  fut i vmi-ben [ ki i ]
vége

felsorol [„alma”, „körte”] ; elemenkénti és
felsorol „karakterlánc”    ; betűnkénti kiíratás
```

Innováció. A LibreLogo újdonságainak egy része a Python nyelv lehetőségeiből fakad, de didaktikai és gyakorlati célból saját megoldásokkal is megkönnyíti a vektoros alakzatokkal való munkát. Ilyen például az órapozíciók használata, amely az alsóbb évfolyamok számára lehetővé teszi a nevezetes szögek megadását a szögekről tanultak előtt is, illetve a már említett `kép` utasítás, amellyel a különálló alakzatokat csoportosíthatjuk a szerkesztés és felhasználás megkönnyítésére (az SVG, EPS formátumú vektorgrafikák is ilyen, egységesen kezelhető alakzatcsoportként tölthetők be a LibreOffice-ba).

A LibreLogo programozási nyelv

„Helló, Világ!”

A következő program a teknőc aktuális helyén kiírja a *Helló, Világ!* üzenetet:

```
címke „Helló, Világ!”
```

A magyar idézőjelek helyett „írógépes” aposztrófok közé is zárhatjuk a szöveget:

```
címke 'Helló, Világ!'
```

A Logo eszköztáron lévő parancssorban a Writer nem cseréli ki automatikusan az „írógépes” dupla idézőjeleket a magyar megfelelőire (a dokumentumban igen). Linuxon az AltGr-O és -P szabványos billentyűkombinációval megkaphatjuk a magyar idézőjeleket. Más, a magyar nyelvet kevésbé támogató operációs rendszeren az Unicode és egyéb karakterkód ismeretét igénylő beviteli módszerek helyett kényelmesebb lehet az alternatív egyszeres írógépes idézőjelek használata a parancssorban.

Utasítássorozat

Az utasításokat egy sorba is írhatjuk:

```
címke „Helló” tf h 12 címke „Világ!”
```

Helló
Világ!!!

A programban a „Helló” kiírása után felemeljük a tollat, majd egy sorral lejjebb lépünk, és kiírjuk a „Világ!”-ot, ahogy a képen látható.

Megjegyzések

A pontosvessző és az azt követő szöveg a sor végéig megjegyzésnek számít.

```
előre 10 ; megjegyzés
```

Sortörésjel

A sor végi hullámvonal (~) jelöli, hogy a következő sor (pontosabban bekezdés a Writerben) még az előző folytatása. A LibreLogóban csak egy-egy nagyon hosszú utasítás több sorban (bekezdésben) való elhelyezésére szolgál, hogy ilyenkor is olvasható maradjon a forráskód.

```
hosszú_nevű_utasítás_aminek_bemenő_adatai ~
  „ez” „és” „ez”            ; három karakterlánc
```

Ahol nem kötelező a pontosvessző, ott érdemes elhagyni, hogy hiba esetén a megfelelő sorra (vagyis utasításra) ugorjon a Writer szövegkurzor. Az értékadó utasításokat sortörésjel nélkül írjuk külön sorba egymás után, l. értékadás.

Teknőcgrafika

Teknőc. A LibreLogo teknőce az eszköztár teknőcmozgató ikonjai, vagy a programindítás hatására jelenik meg a dokumentum első oldalának közepén. A teknőc pozícióját és forgatási szögét a teknőc alakzatának pozíciója és forgatási szöge adja meg, így ezek kézi módosítása a teknőcmozgató utasítások kiadásának felel meg.

A teknőc színe, vonalvastagsága viszont csak jelzi a teknőc állapotát, a kézi módosítás ténylegesen nem változtatja meg a teknőc toll- és töltőszínét, vonalvastagságát.

`Jobbra` és **`balra`** (röviden **`j`** és **`b`**). A két utasítás jobbra, illetve balra forgatja a teknőcöt. Az eljárások bementő értéke a forgatási szög, pl. **60**° vagy egyszerűen **60**), a relatív órapozíció (**2ó** vagy **2h**), vagy a speciális **`tetszőleges`** érték.

```
jobbra 90°
jobbra 90       ; a fokjel elhagyható
j 3ó            ; mint az előzők
b -1h * 3       ; szintén
b tetszőleges   ; véletlen irányba fordul
```

`Előre` és **`hátra`** (röviden **`e`** és **`h`**). A teknőc előre és hátrafelé mozgatása. Az eljárások bemenő értékének alapértelmezett mértékegysége a modern számítógépes tipográfiai pont (2,54 cm, azaz a nemzetközi hüvelyk 72-ed része), pl. az **1** az **1pt** rövidítése, de megadható centiméter (**cm**), milliméter (**mm**), hüvelyk (" vagy **in**) is.

```
előre 10pt
előre 10        ; a pont mértékegysége elhagyható
e 2,54cm/7,2    ; mint az előzők
b 1"/7,2        ; szintén
```

`Tollvastagság` (röviden **`tv`**). Beállítja a toll vonalvastagságát. Használhatók az előző mértékegységek és a tetszőleges érték is.

```
tollvastagság 10    ; 10 pontos vonalvastagság
tv tetszőleges      ; véletlen vastagság (<10pt)
```

`Tollatfel` és **`tollatle`** (röviden **`tf`** és **`tl`**). A toll felemelése után a teknőc nem húz csíkot mozgás közben. A felemelt tollat a teknőc szaggatott körvonala jelzi. Felemelt toll mellett az alakzat rajzolása nem fejeződik be, mivel a komplex vektorgrafikus alakzatok nem összefüggő vonalakból, sőt zárt alakzatokból is állhatnak.

`Pont`. Az utasítás a tollnak megfelelő színű és szélességű pontot helyez el a teknőc pozíciójában (függetlenül a toll felemelt állapotától).

`Tollstílus`. A teknőc által húzott vonal stílusát állítja be a megadott stílusnév alapján. Az alapértelmezett **`„folyamatos”`** vonal lecserélhető **`„pontozott”`** és **`„szaggatott”`** stílusra, illetve listával egyedi pont-vonás kombinációkat tartalmazó mintákat is megadhatunk. L. 2. oldal.

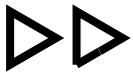
A pont-vonás kombinációkat leíró lista elemei: 1. egymás melletti pontok száma, 2. pontok hossza, 3. egymás melletti vonások száma, 4. vonások hossza, 5. pontok és vonások közötti távolság, 6. nem kötelezően megadandó arányossági tényező: ha értéke 2, akkor nem a megadott méretek, hanem az aktuális vonalvastagság alapján alakul ki a pont-vonások mérete és távolsága.

`Tollsarok`. A vonalak találkozási pontjában, vagyis a csúcsokban alapértelmezés szerint lekerekítést láthatunk, különösen nagyobb vonalvastagságnál. A beállítás módosítható, l. 2. oldal.

`Zár` és **`tölt`**. A LibreLogo teknőc nyitott töröttvonalat rajzol útja során. A `zár` és a `tölt` utasítás kiadásával az utoljára rajzolt nyitott töröttvonalat (vagy töröttvonalakat, l. következő pont) zárja a program, azaz első és utolsó pontjait összeköti, és az így kapott zárt töröttvonalat a `tölt` esetében az aktuális töltőszínnel ki is színezi.

A zár utasításnak abban az esetben is van értelme, ha a rajzolás során visszakerül a teknőc a töröttvonal kezdőpontjába: a vonalvastagságtól és a beállított stílustól függő sarkak csak zárt töröttvonal esetében jelennek meg helyesen a kiindulási csúcsban:

```
tollsarok „hegyes”
tv 3 e 20 j 120 e 20 j 120 e 20 zár
tf j 120+90 e 25 b 90   ; következő:
tl e 20 j 120 e 20 j 120 e 20
```



Az ilyen nem zárt alakzatok a PDF exportálás során „megjavulnak”, azaz azonos kezdő- és végpont esetén automatikusan zártak lesznek. (Emiatt a mellékelt ábra úgy készült, hogy az utolsó megrajzolt oldal valójában két külön szakaszból áll, hogy a PDF-kimenetben is megmaradjon a hiányzó sarok.)

`Töltőstílus`. A paranccsal vonalkázást állíthatunk be a töltőszínen kívül, l. 2. oldal.

Az utasítás bemenő értéke vagy egy 1 és 10 közé eső szám, ami a LibreOffice alapértelmezett vonalkázási stílusait jelöli, vagy egy négy elemet tartalmazó lista, ahol az első elem a vonalkázás típusa (1 = sáv, 2 = négyzetrács, 3 = négyzetrács és sáv), a második elem a vonalkázás színe, a harmadik és negyedik a vonalak távolsága és szöge. A „töltőstílus 0” utasítás kapcsolja ki a vonalkázást.

Összetett alakzatok kitöltése. Ha rajzolás közben megváltoztatjuk a tollvastagságot vagy a tollszínt, akkor a LibreLogo nemcsak új töröttvonal, hanem új vektorgrafikus alakzat rajzolásába is kezd. A toll ideiglenes felemelése viszont nem eredményez új vektorgrafikus alakzatot, mivel az több nem összefüggő töröttvonalat is tartalmazhat. A `zár` és a `tölt` utasítások az utoljára rajzolt vektorgrafikus alakzat minden töröttvonalára egyszerre hajtják végre a zárást, illetve a kitöltést, ahogy erre a jegyzet 2. oldala több példát is hoz. Az előző példa kis módosítása ugyanezt példázza:

```
tv 1 e 20 j 120 e 20 j 120 e 20
tf j 120+90 e 10 b 90   ; következő:
tl e 20 j 120 e 20 j 120 e 20 tölt
```

A példában a „tölt” utasítás lezárja mindkét háromszöget. Látható, hogy az átfedő részek, akár egy önmagát metsző töröttvonal esetében is, olyan kitöltést eredményezhetnek, hogy a kitakart részek kitöltetlenek (átlátszóak) maradnak. Ha ezt nem szeretnénk a példában, akkor az első háromszöget külön tölt utasítással töltsük ki (l. még kép utasítás).

Kép utasítás

Alakzatcsoportok. A kép utasítást elsősorban vektorgrafikus alakzatcsoportok létrehozására használjuk. A következő példában egy vonalat és egy körvonal nélküli, kitöltött kört helyezünk alakzatcsoportba:

```
kép [ tv 1 e 10 tf kör 10 ]
```

Csoport szétbontása. Az alakzatcsoportot a Libre-

Office egy alakzatként kezeli, amíg nem kérjük a felosztását a Rajzobjektumok tulajdonságai eszköztár Csoport szétbontása ikonjával.

Új alakzat kezdése. Ha lezárás és kitöltés nélkül akarunk ugyanolyan vonalvastagsággal és színnek rendelkező új vektorgrafikus alakzatot kezdeni, egy egyszerű `kép` utasítással tehetjük meg:

```
e 10 kép e 10 j 120 e 10 tölt
```

A zászló rúdját bontja két részre a kép utasítás a példában.

Kép utasítás egymásba ágyazása. Összetett ábráknál is lehetőség van az ábra egyes részeinek csoportosítására, ha kép utasításokat ágyazunk egymásba:

```
kép [ kör 15 kép [ kör 5 kör 10 ] ]
```

Ha felbontjuk a példában szereplő (az alapértelmezett félig átlátszó zöld színnek kitöltött) céltáblát a Csoport szétbontása ikon segítségével, egy nagy kört, és egy két kört tartalmazó alakzatcsoportot kapunk (amely utóbbi tovább bontható).

Összetett ábráinkat célszerű csoportba helyezni az egyszerű felhasználás érdekében:

```
kép [ főprogram ]
```

Színek

A színek megadása szöveges, szám és lista paraméterrel, illetve a `tetszőleges` értékkel történhet.

`Tollszín` (röv. **`tsz`**). Beállítja a toll színét. Például:

```
tsz „zöld”
tv 2 kör 10
```

A kör körvonalának színét állítottuk sötétebb zöldre, a kitöltőszínt – az alapértelmezett félig áttetsző zöldet – nem módosítottuk.

`Töltőszín` (röv. **`tlsz`**). Beállítja a kitöltés színét.

```
tlsz „piros” kör 30
tlsz „fehér” kör 20
tlsz „zöld” kör 10
```

Színnevek. 24+1 szín érhető a neve alapján, ilyen pl. a „fekete”, „ezüst”, „szürke” stb. színek (teljes lista a 2. oldalon). A 25. szín a „láthatatlan” szín.

Ezek a színek számérték alapján is elérhetők, ha a szín számát (0-tól 24-ig számozva) egy egyelemű listával adjuk meg:

```
tsz [0] tlsz [24] kör 10 ; csak fekete körvonal
```

Vörös, zöld, kék színkódok. Ezzel a három színösszetevővel leírhatók a számítógép megjelenítette színek. Listával adjuk meg a szín vörös, zöld és kék színösszetevőjét:

```
tollszín [0, 0, 0]          ; fekete
tollszín [255, 0, 0]        ; vörös
tollszín [0, 255, 0]        ; zöld
tollszín [0, 0, 255]        ; kék
tollszín [255, 255, 255]    ; fehér
tollszín [255, 128, 0]      ; narancssárga
```

Egy-egy színösszetevő 8 biten kerül tárolásra, amit itt 0 és 255 közötti számmal adunk meg.

Színek számértékkel. Egy számmal is megadhatjuk a három színösszetevő értékét. A számokat célszerű hexadecimális formában leírni, mert így a hatjegyű szám jegypárainak 0-ff közé eső értéke felel meg a vörös, zöld, kék tartományoknak:

```
tollszín 0x000000           ; fekete
tollszín 0xff0000           ; vörös
tollszín 0x00ff00           ; zöld
tollszín 0x0000ff           ; kék
tollszín 0xffffff           ; fehér
tollszín 0xff8000           ; narancssárga
```

Áttetszőség. Az áttetszőség negyedik (közkeletű nevén alfa) tartományként megadható a színbeállítás során. Lista esetén negyedik elemként, szám esetén a legmagasabb helyi érték felel az áttetszőségnek:

```
tollszín [255, 255, 255, 128]; 50%-ban áttetsző
tollszín 0x80ffffff          ; fehér (mindkettő)
```

Négyzet, téglalap, kör, ellipszis

Külön utasításokkal téglalapot és ellipszis vektorgrafikus alakzatokat, illetve ezek speciális eseteként négyzetet és kört is rajzolhatunk. Alapesetben az aktuális tollszínnel és kitöltőszínnel, a toll felemelése esetén pedig körvonal nélkül rajzolja a LibreLogo az alakzatot a teknőcpozícióval a középpontjában.

`Négyzet`. Megadott oldalhosszúságú négyzet rajzolása.

```
négyzet 10
```

`Téglalap`. Adott szélesség és magasságú, igény szerint lekerekített sarkú téglalap rajzolása.

```
téglalap [20, 10]
téglalap [20, 10, 5]
```

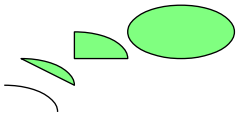
A szélességet, magasságot, és ha szeretnénk, a lekerekítés sugarát listával adjuk meg.

`Kör`. Megadott átmérőjű kör rajzolása.

```
kör 10
```

`Ellipszis`. Adott szélesség és magasságú ellipszis rajzolása. Két szög megadásával cikkelyt rajzolhatunk, illetve egy újabb számérték megadásával szeletet (2) vagy ívet (3).

```
ellipszis [40, 30]
ellipszis [40, 30, 0, 90]
ellipszis [40, 30, 0, 90, 2]
ellipszis [40, 30, 0, 90, 3]
```



Szöveg megjelenítése

`Címke`. Az utasítással a teknőc helyén írathatunk ki szöveget (l. korábbi *„Helló, Világ!”* példa), akár ugyanarra a helyre többször is:

```
betűméret 80 betűszín „piros”
címke „❤”
betűméret 60 betűszín „fehér”
címke „❤”
betűméret 10 betűszín „fekete”
címke „Évi”
```

`Szöveg`. Az utasítás az előzőleg rajzolt vektorgrafikus alakzat (töröttvonal, téglalap, ellipszis) középre igazított feliratát állítja be, vagy módosítja.

```
j 30 e 30 j 120 e 30
szöveg „írás”
```

Lezárt vagy kitöltött alakzatoknál az így beleírt szöveg vízszintes lesz. A felirat együtt mozog az alakzattal, kettős kattintás után módosítható. Így utólag is megadhatunk feliratot az alakzatoknak.

`Betűszín`. Beállítja a `címke` és `szöveg` utasításhoz a kiírt szöveg színét.

Megjegyzés: Az utasításban beállított áttetszőséget a LibreLogo még nem támogatja.

`Betűméret`, **`betűcsalád`**, **`betűvastagság`**, **`betűstílus`**. Betűtulajdonságok megadása a `címke` és `szöveg` utasításokhoz. A betűméretet pontban adjuk

meg. A betűvastagságot vagy olyan arányszámmal, ahol a 100 jelöli a normál betűvastagságot, vagy a **„kövér”** és **„normál”** szöveges értékekkel. A betűcsalád bemenő értéke a betűkészlet neve. A betűstílus pedig lehet az alapértelmezett **„álló”** vagy **„kurzív”** (**„dőlt”**-ként is).

Graphite betűtechnológia. A `betűcsalád` utasításban Graphite betűtulajdonságokat is beállíthatunk, például kiskapitálist, valódi méretezett betűket, vagy egyszerűen vihetünk be matematikai jeleket a Linux Libertine G TeX-módjában:

```
betűcsalád „Linux Biolinum G:smcp=1” ; kiskap.
betűcsalád „Linux Biolinum G:sups=1” ; apró betű
betűcsalád „Linux Libertine G:texm=1”
címke „\\sum_k^n_=_1\\alpha_i”                $\sum_{k=1}^{n}\alpha_i$
```

Bővebb leírás: http://www.numbertext.org/linux, magyar nyelvű jegyzet: http://www.numbertext.org/libreoffice.

Egyéb teknőcgrafikai utasítások

`Törölképernyő` (röv. **`törölkép`**). Az utasítás törli a dokumentum alakzatait (megfelel a Logo eszköztár Képernyőtörlés ikonjának).

`Haza`. A teknőc kezdőpozícióba (a dokumentum első oldalának középpontjába) állítása (megfelel a Logo eszköztár Haza ikonjának).

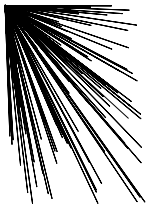
`Elrejt` és **`látható`**. A teknőc elrejtése és láthatóvá tétele.

Rejtett teknőc mellett a rajzolási műveletek gyorsabbak.

`Irány`. A teknőc az adott irányba fordul. Az irányt szöggel, órapozícióval és képernyő-koordinátával is megadhatjuk:

```
irány 0          ; teknőc északnak fordul
irány 3ó         ; teknőc keletnek fordul
irány [0, 0]     ; az oldal bal felső sarka fele
```

`Hely`. A teknőc az adott képernyő-koordináta irányába fordul és odalép, a toll állapotától függően vonalat húzva. `Tetszőleges` bemenő érték esetén tetszőleges helyre lép:



```
hely [0, 0] hely tetsz
```

Vagyis vonalhúzás az oldal bal felső sarkába, majd az oldal egy tetszőleges pontjába. A fenti parancsot többször megismételve kaptuk a mellékelt (itt kicsinyített) ábrát.

Pozíció lekérdezése. A `hely` visszaadja az x és y koordinátát tartalmazó listát is:

```
ki hely                          ; aktuális pozíció
ki hely[0]                       ; x koordináta
ki hely[1]                       ; y koordináta
hely [hely[0] + 10, hely[1] + 10]  ; irány DK-re
```

`Oldalméret`. Az oldal méretét tartalmazó lista.

```
ki oldalméret        ; A4-nél [612pt, 792pt]
```

A lista segítségével vonalat húzhatunk az oldal többi sarkába is:

```
hely oldalméret              ; jobb alsó sarok
hely [oldalméret[0], 0]      ; jobb fölső sarok
hely [0, oldalméret[1]]      ; bal alsó sarok
```

Ismétlődő utasítások (ciklusok)

Gyakori programozási feladat valamely tevékenység, utasítás vagy utasítássorozat ismételt végrehajtása, amelyet ciklusokkal valósítunk meg.

`Ismétlés` (röv. **`ism`**, vagy a LibreLogóban **`ismét`**). Ez a ciklus megadott számban ismétli a kapcsos zárójelek között megadott utasításokat (programblokkot).

```
ism 3 [ e 10 j 120 ] ; szabályos háromszög
```

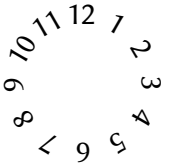
Ügyeljünk a szóközökre, vagy sortörésre a kapcsos zárójelek, és a közbezárt utasítások között, amit a LibreLogo, szemben a Logóval, megkövetel!

`Végtelenszer` (röv. **`vszer`**). A ciklusnak nincs megállási feltétele, így addig fut, amíg nem kerül végrehajtásra egy `kilép`, `stop` vagy `eredmény` utasítás, vagy le nem állítjuk a program futását.

```
végtelenszer [ tf hely tetsz kör 10 ]
```

A LibreLogóban a „végtelenszer” az „ismétlés” alternatív neve.

`Hányadik`. A ciklus sorszámát tartalmazó változó:



```
tf ism 12 [
  j 1ó e 30
  címke hányadik h 30
]
```

A teknőc az óra számlapjának közepéről indul. Mindig 1 órát az óramutató járásának irányába fordulva és a középponttól 30 pontnyira eltávolodva kiírjuk a „hányadik” változóval a ciklus 1-től 12-ig futó ciklusváltozóját.

Ciklusok ciklusokon belül. Az előző példaprogram javításában a fő (külső) ciklus kirakja a számjegyeket, egy cikluson belüli (vagyis belső) ciklus pedig az ötödórás (12 perces) beosztásokat:

```
betűcsalád „Linux Biolinum G:pnum=1”
tf j 1ó
ism 12 [
  e 20 b 1ó * hányadik
  címke hányadik
  j 1ó * hányadik
  h 20 tv 2
  ism 5 [
    e 20 + 6 tl e 3
    tf h 20 + 6 + 3
    j 1ó/5 tv 0,5
  ]
]
```

A program első sorában bekapcsoljuk a Graphite betűkészlet arányos számelhelyezését is, hogy a keskeny számjegyek, mint a 11-ben, közelebb kerüljenek egymáshoz (alapesetben a gyakori többsoros táblázatos elhelyezés miatt a számjegyek ugyanakkora helyet foglalnak el, lásd az első óraszámlapon, de ez sima szövegben kerülendő). Az egész óráknál látható vastagabb beosztást úgy kapjuk, hogy a belső ciklus előtt nagyobb vonalvastagságot állítunk be, amit csak az első beosztás megrajzolása után írunk felül.

`Hányadik` az egymásba ágyazott ciklusoknál. A szorzótábla a legismertebb példa az olyan egymásba ágyazott ciklusokra, ahol a külső (szorzandó) és a belső (szorzó) ciklusváltozót is felhasználjuk. Mivel a belső ciklusok felülírják a külső ciklus következő indulásáig a `hányadik` változó értékét, egymásba ágyazott ciklusoknál mentsük el az értékét egy új változóba:

```
ism 10 [ ; „kirajzoljuk” a szorzótáblát (l. kép)
  szorzandó = hányadik
  ism 10 [ tv szorzandó * hányadik b 3,6 e 5 ]
  ; a hányadik értéke itt mindig 10
]
```

A (felére kicsinyített) ábrán 10 tölcsérszerű alakzatot láthatunk,

amelyek az 1-es, 2-es stb. szorzótábla szorzatait ábrázolják a megfelelő szélességű csíkokkal.

Ciklusmese. Az `ismétlés` (`ism`) ciklus feltételében szereplő `hányadik` a külső ciklusra vonatkozik, ezt mutatja be a következő mese:

Hol volt, hol nem volt, volt egyszer egy Susga nevű kislány. A tanára – azt remélve, hogy jó darabig leköti a gyerekeket –, feladta az osztálynak, hogy adják össze a számokat 1-től 100-ig. De Susga hamarosan jelentkezett, hogy kész van. A tanár meglepődve kérdezte, hogyan jött rá ilyen gyorsan a jó eredményre? Susga elárulta, hogy rajzolgatni kezdett unalmában, mert kedve nem volt a sok összeadáshoz. Lerajzolt egy pöttyöt, majd mellé, a következő oszlopban kettőt, majd még egy új oszlopban hármat, egészen tízig. A LibreLogóval így kaphatjuk meg Susga rajzát:

```
tf tv 3 ism 10 [
  ism hányadik [ pont e 5 ]
  h hányadik * 5
  j 90 e 5 b 90
]
```

Vagyis 10-szer ismételjük a belső ciklust, ami az éppen végrehajtott külső ciklus sorszámának megfelelő számú pöttyöt rajzolja ki egymás fölé (majd visszalépteti a teknőcöt alulra, és a következő oszlopba állítja). Susga rájött, hogy ez az ábra majdnem pontosan olyan, mint egy fél négyzet. Egy csupa pöttyből álló négyzet átlójának két oldalán ugyanannyi pötty van, az átlóban pedig a négyzet oldalhosszának megfelelő pötty található. Ez alapján, ha vesszük a négyzetben lévő pöttyök számának felét (n*n/2), akkor, mivel ez az átló felét már tartalmazza, már csak az átlóban szereplő pöttyök másik felét, n/2-t kell hozzáadni, hogy megkapjuk a számok összegét 1-től 10-ig. Ez 10*10/2 + 10/2 = 100/2 + 5 = 55. Száz pötty oldalhosszúságú négyzetnél ugyanúgy kell eljárni: 100*100/2 + 100/2 = 10 000/2 + 50 = 5050. Míg tanára beírta az ötöst, Susga fejben kiszámolta még az első ezer szám összegét is.

`Amíg`. A megadott logikai kifejezés teljesüléséig hajtja végre a ciklustörzsben szereplő utasításokat.

```
amíg igaz [ e 1 b 1 ]          ; végtelen ciklus
amíg hányadik <= 5 [ ]         ; mint ism 5 [ ]
```

A ciklus feltételében szereplő „hányadik” itt a ciklusra vonatkozik.

`Fut -ban/-ben`. A ciklusváltozó végiglépked a megadott lista vagy karakterlánc elemein:

```
fut i [1, 2, „három”]-ban [ ki i ]
fut i „szöveg”-ben [ ki i ]  ; ki karakterenként
```

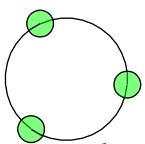
A `hányadik` változó segítségével itt is lekérdezhetjük az aktuális ciklus sorszámát (a feltételben is).

`Újra` és **`kilép`**. A két utasítással a ciklusok végrehajtása szakítható meg. Az `újra` hatására a ciklus elejére ugrik a programvégrehajtás, így a soron következő ciklussal folytatódhat a program. A `kilép` esetén viszont egyből a ciklus után, azt elhagyva fut tovább.

Feltételvizsgálat

`Ha`. Az utasítást követő feltétel (logikai kifejezés) teljesülése esetén a program végrehajtja a kifejezést követő programblokkot.

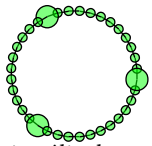
```
ism 36 [
  e 4 b 10
  ha hányadik = 12 vagy ~
    hányadik = 24 [ kör 10 ]
] kör 10
```



A program 36-szöget rajzol, miközben a 12. és a 24., majd az utolsó csúcsnál (itt már a cikluson kívül) kört is rajzol.

A blokk után még egy blokk is szerepelhet, amely a feltétel nem teljesülése esetén hajtódik végre.

```
ism 36 [
  e 5 b 10
  ha hányadik % 12 = 0 [ kör 10 ] ~
    [ kör 4 ]
]
```



Az előző programot egyszerűsítve a feltétel azt vizsgálja, hogy a ciklusváltozó maradék nélkül osztható-e 12-vel. Ha igen, 10 pont átmérőjű kört rajzol, ha pedig nem, akkor 4 pont átmérőjűt.

Logikai kifejezések. A logikai kifejezések értéke igaz, vagy hamis lehet. Az egyenlőség műveleti jele a sima **`=`** vagy a dupla **`==`**.* A nem egyenlőséget a **`<>`** vagy a **`!=`** jelekkel. A kisebb vagy egyenlő, és a nagyobb vagy egyenlő műveleti jelek a **`<=`** és a **`>=`**. A **`vagy`** és az **`és`** műveletek mellett a **`nem`** művelet is használható:

```
ha nem (a > 0 és a <= 10) [ ki „kívül van” ]
```

A példában ha „a” értéke kisebb, vagy egyenlő, mint 0, vagy nagyobb mint 10, kiírásra kerül a „kívül van” üzenet.

Az alapértelmezett **`igaz`**, vagy **`hamis`** logikai értéket kifejezésekben is használhatjuk:

```
a = igaz
ha a [ ki „nagyon igaz” ] [ ki „nem igaz” ]
```

A logikai kifejezést váró utasítások a 0 (és üres karakterlánc, lista, halmaz vagy szótár) értéket hamisnak, a nem 0 értéket pedig igaznak veszik. A logikai kifejezések kiíratásánál vagy karakterlánccá alakításánál az „igaz” vagy „hamis” szót kapjuk:

```
ki 5 == 5              ; „igaz” jelenik meg
b = lánc 2 * 2 == 5    ; b = „hamis”
```

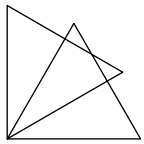
Saját utasítások (eljárások)

A LibreLogo utasításkészletét magunk is bővíthetjük, például az alapértelmezett `négyzet` és `kör` utasítás mellé készíthetünk egy saját, `háromszög` névre keresztelt utasítást is a háromszögek rajzolására. Az ilyen új utasításokat eljárásoknak nevezzük, és az **`ez`** (vagy **`tanuld`**, **`eljárás`**, röv. **`elj`**) paranccsal kezdve és (külön sorban) a **`vége`** paranccsal befejezve hozzuk létre:

```
ez eljárásnév
utasítások
vége
```

Az eljárásban szereplő utasítások csak akkor kerülnek végrehajtásra, amikor az eljárást meghívjuk, azaz nevével hivatkozunk rá, mint a következő példa utolsó sorában kétszer is:

```
ez háromszög
  ism 3 [ e 50 j 360/3 ]
vége

háromszög j 30 háromszög
```

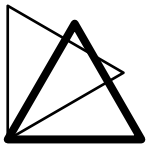


Az eljárás végrehajtása után a hívási pont után folytatódik a program, így a példában kirajzoltunk egy szabályos háromszöget, majd 30 fokkal jobbra elfordulva ismételtük meg a háromszögrajzolást.

Bemenő értékek. Az eljárásnév után felsorolt változók tartalmazzák az eljárás bemenő értékeit. Az eljárás hívásánál, az eljárásnév után adjuk meg az aktuális bemenő értékeket.

* Pythonban csak a dupla egyenlőségjel használható műveleti jelként. A LibreLogo viszont a „ha”, „amíg”, „eredmény” utasításokban szereplő logikai kifejezésekben megengedi a sima egyenlőségjel használatát is.

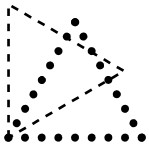
```
ez háromszög méret
  tollvastagság méret
  ism 3 [ e 50 j 360/3 ] zár
vége

háromszög 1 j 30 háromszög 3
```



A példában a „méret” bemeneti változó segítségével állítjuk be a háromszög vonalvastagságát. A háromszöget a zár utasítással le is zárjuk, hogy minden sarka lekerekítve legyen.

Több bemenő érték. Több bemenő érték esetén az eljárás hívásánál egyszerűen felsoroljuk (vessző nélkül) a bemenő értékeket (vagy az azokat visszaadó kifejezéseket).

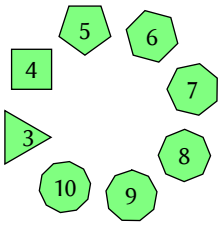
```
ez háromszög méret stílus
  tollvastagság méret
  tollstílus stílus
  ism 3 [ e 50 j 360/3 ] zár
vége

háromszög 1 „szaggatott” j 30
háromszög 3 „pontozott”
```



A példában a háromszög vonalstílusát is beállítja az eljárás egy második bemenő érték alapján.

További példa. A következő példában a „háromszög” eljárás általánosítása, a „sokszög” szerepel. Segítségével kirajzoljuk a háromszögtől a tízszögig a szabályos síkidomokat.

```
ez sokszög szög él
  ism szög [ e él b 360/szög ]
  tölt szöveg szög
vége

ism 8 [
  sokszög 2 + hányadik ~
    60 / (2 + hányadik)
  tf e 18 j 360/8 tl
]
```



A példában a sokszög eljárás két bemenő értéket kap: a „szög” nevű bemeneti változó a szögek (csúcsok) számát, az „él” pedig egy él hosszát fogja tartalmazni. Az eljárás után lévő főprogram egy olyan ciklus, amely 3 és 60/3, majd 4 és 60/4 stb. egészen 10 és 60/10 bemenő értékekkel hívja meg a sokszög eljárást (mivel azt szeretnénk, hogy a síkidomok közel azonos méretűek legyenek, ezért nem az élhosszuk, hanem a kerületük lesz egységnyi, jelen esetben 60 pont). Az egyszerű sokszögeket körbejárva teljes fordulatot, vagyis 360 fokot teszünk meg. Szabályos sokszögeknél minden csúcsnál ugyanannyit, azaz 360/csúcsszám foknyit fordulunk. Az eljárásunk bemenő paramétereiből, a csúcsok számát tartalmazó „szög”-ből így határoztuk meg a fordulás szögét. Megjegyzés: az ábrán egy picit elforgattuk a síkidomokat, hogy jobban elférjenek: a 360/szögnyi elfordulást két részletben, az él rajzolása előtt, és után hajtottuk végre:

```
  ism szög [ b 180/szög e él b 180/szög ]
```

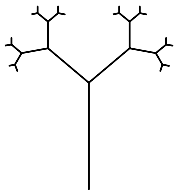
Újrahívás. Az eljárás meghívhatja saját magát is, amivel ciklusokat, bonyolultabb műveleteket valósíthatunk meg. A következő példa a háromszögrajzoló eljárás újrahívással megvalósított változata:

```
ez háromszög
  e 50 j 360/3
  háromszög
vége

háromszög
```

A program nem áll le a három oldal megrajzolásánál, hanem végtelenségig ismétli azt. A gyakorlatban a LibreLogo/Python alapbeállítás csak ezer újrahívást engedélyez. A mélységi, vagy a gyakran előbb jelentkező memóriakorlátot elérve a program befejezi működését. (Az újrahívást tartalmazó eljárások általában tartalmaznak valamilyen erősebb korlátot, így az ilyen leállásra ritkán kerül sor.)

`Stop`. Az utasítással kiléphetünk az eljárásból, vagyis a program az eljáráshívás után folytatódik.

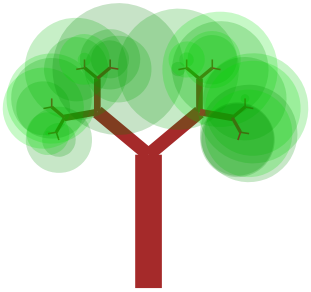
```
ez fa n
  ha n < 2 [ stop ]
  e n b 50 fa n/2
  j 100 fa n/2
  b 50 h n
vége

fa 40
```



A farajzoló eljárás n hosszúságú ágat rajzol, majd elágazást rak az ág végére a következő módon: kétszer hívja meg saját magát, hogy bal és jobb oldali, feleakkora ágat rajzoljon. Ezt minden ág rajzolásánál ismétli, amíg a rajzolandó ág kisebb nem lesz, mint 2 pont. Ekkor már nem rajzol elágazást a program, hanem visszatér az előző hívás helyére. Az elágazás megrajzolása után az eljárás visszalépteti a teknőcöt az ág elejére, hogy folytatni tudja az ágak rajzolását a megfelelő helyről. A teknőc mozgása szemmel követhető módon mutatja a program működését.

További példa. Az előző eljárás kis bővítésével javíthatunk a fa megjelenítésén:

```
ez fa n
  tv n/5
  ha n < 2 [
    tlsz [0, 127 + ~
      vszám 128, 0, 200]
    kör vszám 50 stop
  ]
  tl e n tf b 50 fa n/2
  j 100 fa n/2
  b 50 h n
vége

kép [ tsz „barna” fa 50 ]
```



A barna színű ágak vastagsága az ág hosszának ötöde, és a legkisebb ágak végére a v(életlen)szám függvénnyel véletlen átmérőjű, áttetsző véletlen zöld köröket teszünk. A főprogramot a kép elembe zártuk, hogy a körök és vonalak egy alakzatcsoportba kerüljenek.

Saját függvények

A függvények kimeneti értéket visszadó eljárások.

Az előző példa v(életlen)szám függvénye a bemeneti értékénél kisebb véletlen törtszámmal tér vissza. Ezért a „kör vszám 50” lehet „kör 0” és „kör 49,999” is. Saját függvényeket az „eredmény utasítással” hozhatunk létre.

`Eredmény`. Kilépés az eljárásból az `eredmény` utasítás után megadott kifejezés értékével, vagyis a saját függvény visszatérési értékével:

```
ez neme név
  ha név = „Juliska” [ eredmény „nő” ]
  ha név = „János” [ eredmény „férfi” ]
  eredmény „nem tudom”
vége

ki neme „Juliska”
ki neme be „Kérem a keresztnevét:”
```

A „neme” függvény „Juliska” bemeneti érték esetén „nő”-vel, „János” esetén „férfi”-val tér vissza, különben pedig „nem tudom”-mal. Az első hívás eredménye „nő”, a második hívásnál a gyári „be” függvény párbeszédablakába beírt név lesz a „neme” bemenete.

Zárójelezés. Több bemenő értéket váró, esetleg összetett kifejezésekben szereplő függvényhívásoknál a bemenő értékeket zárójelbe zárjuk, vesszővel elválasztva:

```
ez átlag x y
  eredmény (x + y) / 2
vége

; k = (átlag 5 10)                    ; nem jó
k = átlag(5, 10)                      ; jó
k = átlag (5, 10)                     ; jó
```

Egy bemenő értéknél a zárójelezés egyszerű kifejezésekben elhagyható, de hiba esetén itt is ki kell tennünk a zárójeleket.

Változók

Elhagyható kettőspont, Unicode betűk és aláhúzásjel. A változónevek az Unicode karakterkészlet betűiből, aláhúzásjelből („_”), számokból (kivéve a név első karaktereként) és egy elhagyható kezdő kettőspontból állhatnak.

A névkezdő kettőspont (eredetileg az érték szerinti hivatkozást jele a Logóban) használatának előnye a magyarban, hogy a „j” és „b” parancsrövidítés nem ütközik a „:j” és a „:b” változókkal, de érdemes inkább a szabad k, l, m ... z betűket választani rövid változónévként.

Kisbetű nem egyenlő nagybetűvel. A változónevek (szemben a parancsokkal, vagy az alapértelmezett változókkal) már akkor is különböznek, ha csak a betűk méretében van különbség.

```
K = 1
k = 2
ki K + k                              ; az eredmény 3
```

π. A **π** (alternatív neve **pi**) speciális változó, előre meghatározott értéke 3,14 159 265 358 979. Példa a kiíratására:

```
ki π                    ; írhatunk „ki pi”-t is.
```

Értékadás

A LibreLogóban nemcsak eljáráshívással, hanem egyenlőségjellel is értéket adhatunk a változóknak:

```
k = 1                                 ; egész szám
l = 1,0                               ; tizedestört
m = „szöveges változó”                ; karakterlánc, itt
n = be „adatbevitel:”                 ; párbeszédablakkal
o = [1, 2, „három”]                   ; lista
p = {0: „nulla”, 1: „egy”}            ; szótár
tíz = 0b1010                          ; bináris szám
ezer = 0x3e8                          ; hexadecimális sz.
kilenc = 0o11                         ; oktális szám
```

Az értékadó utasításokat külön sorba helyezzük el. Blokk elején hagyhatunk egy értékadást ugyanazon sorban, illetve nem értékadó utasítás követheti az értékadást:

```
ha a = 1 [ a = 0 tölt ]
```

Globális változók

Alapesetben a változók csak az adott eljáráson vagy a főprogramon belül érvényesek. Ha azt szeretnénk, hogy minden eljáráson belül elérhetők legyenek, adjuk ki a **globálisváltozó** (röviden **globvál** vagy a LibreLogóban **globális**) parancsot:

```
globális k, l, m
```

Értékadás eljáráson belül. Ha a globális változóknak értéket is szeretnénk adni az eljáráson belül, nemcsak kiolvasni azt, akkor az eljárásban is meg kell adnunk a `globális` parancsot a változó nevével.

Számok

A tetszőleges méretű egész számok mellett törtszámokkal (lebegőpontos számokkal) végezhetünk műveleteket. A négy alapművelet, az összeadás, kivonás, osztás, szorzás mellett gyakrabban használjuk a dupla perjellel, illetve százalékjellel megadott maradékos osztást, és a hatványozást:

```
ki 5 // 2   ; 5/2 maradékos osztással, vagyis 2
ki 5 % 2    ; az eredmény 5/2 maradéka, vagyis 1
ki 5**3     ; 5 * 5 * 5 = 125
```

Megjegyzés. A LibreLogóban az osztás tört számot eredményez, míg Pythonban ez csak a 3-as változattól vált alapértelmezetté:

```
x = 5/2           ; 2,5 (Python 2-ben viszont 2)
```

Egész. A függvény egész számmá alakítja a bemenő tört számot (annak egész részét véve) vagy karakterláncot:

```
ki egész 4,9999           ; eredmény: 4
k = egész „5000”
ki k + 1                  ; eredmény: 5001
ki 1 + egész be „Szám?”   ; ki a bemenet + 1
```

Tört. A függvény tört számmá alakítja a bemenő egész számot vagy karakterláncot:

```
ki 1 + tört be „Szám?”   ; ki a bemenet + 1,0
```

Véletlenszám (röv. **vszám** vagy **véletlen**). A függvény a megadott számnál kisebb, de nullánál nagyobb, vagy egyenlő véletlen törtszámot ad vissza.

```
ki vszám 100              ; 0–99,9̇ közötti tört
ki 1 + egész vszám 100    ; 1–100 közötti egész
```

A függvény (alternatív néven „kiválaszt”) lista megadása esetén véletlen listaelemet ad vissza.

Egyéb műveletek. A **gyök**, **kerekítés** (röv. **kerek**), **abszolútérték** (röviden **absz** vagy **abs**), **sin** és **cos** a nevének megfelelő műveletet végzi el.

Bitműveletek. Az **&**, **|**, **^** műveleti jelek a bitenkénti *és*, *vagy*, *kizáró vagy* műveletnek felelnek meg.

Min és **max**. A függvények a megadott lista minimum, illetve maximum értékét adják vissza.

```
ki min (1, 2, 0)            ; ki: 0
ki max [5, 7, 8, 9]         ; ki: 9
```

Karakterláncok

A karakterlánc tetszőleges hosszúságú szöveg tárolására alkalmas változótípus. A LibreLogóban több lehetőség van a karakterláncok megadására. A javasolt forma a magyar helyesírást követi:

```
k = „”          ; üres karakterlánc
l = „példa”     ; lánc öt karakterből
```

A beírást a Writer automatikus idézőjelcseréje egyszerűvé teszi minden operációs rendszeren.

Ezenkívül alkalmazhatjuk a Python egyszeres „írógépes” idézőjeles megadását is:

```
k = ''          ; üres karakterlánc
l = 'példa'     ; lánc öt karakterből
```

Csak betűk (nem írásjelek stb.) esetén a klasszikus Logo megadásra is lehetőség van, opcionálisan annak lezárásával:

```
k = "           ; üres karakterlánc, vagy ""
l = "példa      ; lehet "példa" is.
```

Összefűzés. Az összeadásjellel fűzhetünk össze karakterláncokat:

```
k = „macska” + „jancsi”     ; k = „macskajancsi”
```

Többszörözés. A karakterláncok többszörözése,

vagyis ismétlődő összefűzése látványos példája a Python különleges lehetőségeinek:

```
ki „példa” * 100 ; példapéldapélda... (100-szor)
```

Karakterek. *N* karakterből álló karakterlánc karakterei 0-tól *n*–1-ig számozva érhetők el a következő módon:

```
szöveg = „példa”
ki szöveg[0]            ; első karakter
ki szöveg[4]            ; itt utolsó karakter
```

Láncvégi karakterek. Mínusz számozással a karakterlánc végéről érhetünk el karaktereket:

```
szöveg = „példa”
ki szöveg[-1]           ; utolsó karakter
ki szöveg[-5]           ; itt első karakter
```

Részláncok. Kettőspont segítségével a karakterlánc részletét kaphatjuk vissza:

```
s = „példa”
ki s[:3]          ; a[3] előtti: „pél”
ki s[3:]          ; a[3]-tól: „da”
ki s[1:4]         ; a[1]-től a[4] előttig: „éld”
ki s[:-1]         ; a[-1] előttig: „péld”
```

`Darab` (röv. **`db`**). A lánc hosszát megadó függvény.

```
ki db „példa”         ; Kimenet: 5 (db karakter)
```

Módosítás. Karakterláncok helyben nem módosíthatók, szemben a listákkal, de a változó felülírásával hasonló hatást érhetünk el:

```
A = „VAKÁCIÓ”                                   Ó
amíg A [                                        IÓ
  címke A                                       CIÓ
  A = A[1:]   ; levágjuk az első betűt          ÁCIÓ
  tf e 12     ; egy sorral feljebb              KÁCIÓ
]                                               AKÁCIÓ
                                                VAKÁCIÓ
```

Eleme-e. A ciklusoknál megismert `-ban/-ben` segítségével megvizsgálhatjuk, hogy egy adott karakter vagy karakterlánc része-e a másiknak:

```
ha „x” „mátrix”-ban [ ki „van benne x betű” ]
```

`Lánc`. A `lánc` függvénnyel karakterlánccá alakítható a más típusú változók értéke:

```
a = lánc 5,5                        ; szöveg: „5,5”
```

Egyéb Python függvények. A Python egyéb nem honosított függvényei is elérhetők a LibreLogóban, például:

```
A = „szöVege ”
A.upper()      ; nagybetűsítés: „SZÖVEGE ”
A.lower()      ; kisbetűsítés: „szövege ”
A.capitalize() ; nagy kezdőbetűsítés: „SzöVege”
A.find(„e”)    ; balról első előfordulás: 4
A.rfind(„e”)   ; jobbról első előfordulás: 6
A.replace(„sz”, „f”) ; csere: „föVeg”
A.split(„V”)   ; darabolás: [„szö”, „eg”]
A.strip()      ; szóközök levágása: „szöVeg”
```

Javasolt megismerkedni a szabályos kifejezésekkel is (l. később), amelyek nagy mértékben leegyszerűsítik és felgyorsítják a szövegkezelést.

Sortörés karakter. A **`\n`** a sortörés karaktert jelöli a karakterláncokban. Ennek megfelelően több sorban jelenik meg a kiírt szöveg a LibreLogóban is.

```
címke „Libre-\nLogo”              ; Libre- és Logo
```

Formázott karakterláncok. A **`%d`** helyére számot, a **`%s`** karaktersorozat helyére szöveges változó értékét illeszthetjük be, ha a karakterlánc után százalékjellel elválasztva megadjuk a változót is:

```
ki „Száma: %d.” % 5           ; „Száma: 5.”
ki „Neve: %s.” % „Pista”      ; „Neve: Pista.”
ki „Száma: %s.” % lánc 5,5    ; „Száma: 5,5.”
```

A `d`, illetve `s` előtt számot is megadhatunk, ami azt jelöli, hogy hány karakterpozíciót foglaljon el az érték, függetlenül a hosszától:

```
ki „Száma: %5d.” % 5           ; „Száma:    5.”
```

Ha egyszerre több beillesztendő értékünk van, akkor zárójelben, vesszővel elválasztva soroljuk fel a változókat (l. fix listák):

```
ki „%d: %s.” % (5, „Pista”)   ; „5: Pista.”
```

Kulcs-érték párokat tartalmazó szótárakból is kiolvashatjuk az adatokat (l. szótárak), ha a **`%(kulcs)s`** formában hivatkozunk az értékre a karakterláncban:

```
Adat = {„neve”: „Pista”}
ki „Neve: %(neve)s.” % Adat  ;    „Neve: Pista.”
```

Szabályos kifejezések

A modern programozási nyelvek szinte kivétel nélkül támogatják ezt a mintaillesztési módszert, amellyel a karakterláncokon végzett műveletek jelentősen egyszerűsíthetők.

A LibreOffice Keresés és csere funkciójában is megtalálni reguláris kifejezések néven, l. LibreOffice súgó. (A szabályos kifejezések hasonló helyen jelentek meg eredetileg az informatikában: elsőként Ken Thompson QED szövegszerkesztőjében, később a standard unixos „ed” szövegszerkesztő és egyéb segédprogramok népszerűsítették.)

`.*` (pont csillag). Az egyik leggyakrabban alkalmazott minta a szabályos kifejezésekben a tetszőleges karaktersorozatot jelölő `.*` (pont csillag).

Máshol is találkozhatunk ilyen mintaillesztéssel. A Google kereséseknél (itt igaz csak szóközökkel határolva, szavakra működik) és a parancssori állományneveknél a sima csillag, az SQL adatbázis-kezelő nyelv LIKE parancsánál a % jel jelöli a tetszőleges karaktersorozatot. A legtöbb lehetőséget azonban kétségkívül a szabályos kifejezések, annak is különbözőképpen bővített változatai nyújtják.

A szabályos kifejezéseket három honosított függvény támogatja a LibreLogóban:

`Talál`. A függvény egy szabályos kifejezést (a keresési mintát) és egy karakterláncot vár. Eredménye pedig egy olyan lista, ami a karakterlánc keresési mintára illeszkedő részleteit tartalmazza. Ha nincs ilyen, akkor az eredmény egy üres lista.

```
szavak = talál(„\w+”, „A kutya ugat, nem?”)
```

A fenti függvényhívás eredménye az „A”, „kutya”, „ugat” és „nem” karakterláncokat, vagyis a megadott szöveg szavait tartalmazó lista, mivel a keresési mintaként megadott szabályos kifejezés jelentése „egy, vagy több betűt tartalmazó karaktersorozat”.

`Keres`. A függvény a `talál` függvényhez hasonló bemeneti értékeket vár. Az eredmény az első illeszkedő szövegrészlet adatait tartalmazza, vagy ha nincs találat, akkor az ennek hiányát jelölő „semmi” érték. A találati érték igaz értéknek, a „semmi” hamis értéknek számít a logikai kifejezésekben, így a feltételvizsgálatokban is:

```
szó = be „Kérek egy tetszőleges szót”
ha keres(„sz”, szó) [ ]  ; = ha „sz” szó-ban [ ]

ha keres(„^p.p$”, szó) [ ki „találat” ] [ ki ~
  „nincs találat” ]
```

A szabályos kifejezés jelentése az utolsó sorban: Ha a megadott szó p-vel kezdődik, a második karaktere tetszőleges, a harmadik, egyben a karakterlánc végén található karaktere pedig szintén a „p” betű (például „pap”, „pép”, „púp”), kiírja hogy „találat”, különben pedig hogy „nincs találat”.

`Cserél`. A függvény három bemenő értéke a *mit* (szabályos kifejezés), *mire* (hivatkozhatunk benne az illeszkedő karakterláncra, sőt annak részeire is) és *miben.*

```
írás = cserél(„[A-Z][^.?!]*!”, „”, írás)
```

A példa kitörli a felszólító mondatokat a karakterláncból, vagyis minden nagybetűvel kezdődő, pontot, kérdőjelet, felkiáltójelet nem tartalmazó, de felkiáltójelre végződő tetszőleges hosszú karakterláncot.

A szabályos kifejezés jelölései. A LibreLogo a Python szabályos kifejezéseket ismeri fel. A következő lista a legalapvetőbb jelöléseket sorolja fel:

```
.       ; tetszőleges karakter
*       ; előző karakter 0, vagy többször
.*      ; tetszőleges karakter 0, vagy többször
a*      ; „a” karakter 0, vagy többször
a+      ; „a” karakter 1, vagy többször
[a-d]   ; karaktertartomány („a”, „b”, „c”, „d”)
[A-Z]   ; tetszőleges nagybetű (nem ékezetes)
[A-ZÁÉ] ; latin nagybetűk, plusz „Á” és „É”
[-b]    ; „-” karakter vagy „b”
\.      ; pont karakter
[.]     ; pont karakter
[.*+]   ; pont, csillag vagy plusz karakter
[^.*+]  ; tetszőleges, kivéve „.”, „*”, „+” jel
\w      ; tetszőleges betű, szám vagy aláhúzásjel
(Ft|\$) ; „Ft” vagy „$”
^vmi    ; „vmi” a karakterlánc elején
vmi$    ; „vmi” a karakterlánc végén
\bez\b  ; „ez”, mindkét oldalán szóhatárral
```

A `cserél` függvény második bemenő értékében a szabályos kifejezés zárójeles csoportjaira hivatkozhatunk a `\\1`, `\\2` stb. számozással:

```
írás = cserél(„(\w+) (\w+)”, „\\2 \\1”, írás)
```

A példában felcseréljük a szomszédos szavakat.

Listák

A LibreLogóban Python listákat használhatunk.

Azzal a formai megkötéssel, hogy a listákat határoló kapcsos zárójelek tapadnak (vagyis nem lehet szóköz a belső oldalukon).

```
L = []              ; üres lista
M = [6]             ; egyelemű lista
M = [6,6]           ; egyelemű lista (6,6-tal)
```

A listák létrehozásánál az elemeket vesszővel választjuk el egymástól.

```
N = [6, 6]          ; kételemű lista
```

Elemek elérése. *n*-elemű lista elemeit 0-tól *n*–1-ig számozva érhetjük el a *lista[elem száma]* alakban:

```
ki N[0]             ; első elem kiíratása
N[1] = „lánc”       ; N = [6, „lánc”]
```

Ahogy a fenti példán látható, a listaelemek tetszőleges adattípusúak, akár listák is lehetnek.

Mínusz indexekkel a lista végéről érhetjük el az elemeket:

```
ki N[-1]            ; az utolsó elem („lánc”)
```

Véletlen elemet ad vissza a **`kiválaszt`** függvény:

```
ki kiválaszt [„alma”, „körte”, „barack”]
```

Összefűzés. Listákat az összeadásjellel fűzhetünk össze:

```
Z = N + N           ; Z = [6, „lánc”, 6, „lánc”]
```

`Darab` (röv. **`db`**). A függvény visszaadja a lista elemszámát:

```
ki db Z             ; A kimenet 4
```

`Rendez`. A függvény új, rendezett listával tér vissza a megadott lista alapján.

```
Z = rendez Z        ; Z = [6, 6, „lánc”, „lánc”]
```

`Sor`. A függvény egy bemenő érték esetén olyan listát ad vissza, amely 0-tól a bemenő érték mínusz 1-ig tartalmaz elemeket:

```
t = sor 5 ; t = [0, 1, 2, 3, 4]
```

Az „ismétlés” ciklus alternatívájaként a „sor” függvényt kombinálhatjuk a „fut -ban/-ben” ciklussal. Külső ciklusok ciklusváltozóit ezen a módon könnyebben elérhetjük.

```
fut k sor 10-ben [ ki k ]
```

Két bemenő értékkel a kezdőérték is megadható:

```
t = sor 1 5 ; t = [1, 2, 3, 4]
```

Hárommal pedig a lépésköz:

```
t = sor 1 10 2; t = [1, 3, 5, 7, 9]
```

Eleme-e. A „`-ban`”/„`-ben`” segítségével itt is megvizsgálhatjuk, hogy egy adott érték eleme-e a listának:

```
Z = [1, 2, 5]
ha 5 Z-ben [ ki „benne” ]
```

Üres-e. Üres listával, a `darab` függvénnyel, vagy közvetlenül a változóval is megvizsgálhatjuk, hogy üres-e egy lista:

```
ha L = [] [ ki „üres” ]
ha db L = 0 [ ki „üres így is” ]
ha nem L [ ki „és üres így is” ]
```

Részlisták. A LibreLogo (Python) listakezelésének egyik kényelmes lehetősége, hogy a kettőspont segítségével részlistákra hivatkozhatunk (részletesen l. a karakterláncoknál).

```
ki Z[2:5]       ; részlista 2–4. indexű elemekkel
Z = Z[:-1]      ; utolsó elem törlése
```

Több elem cseréje. Az értékadás bal oldalán is szerepelhet részlista, amivel egy lépésben módosíthatjuk a lista több elemét is:

```
Z = [1, 3, 3]
Z[1:2] = [2]        ; eredmény: Z = [1, 2, 3]
Z[1:2] = [5, 5]     ; eredmény: Z = [1, 5, 5, 4]
```

Beszúrás. Ha üres részlistára hivatkozunk, akkor az adott hely elé szúrhatunk be elemeket:

```
Z = [2, 2]
Z[1:1] = [10]       ; eredmény: Z = [2, 10, 2]
```

Lista megfordítása. A következő eljárás visszaad egy a bemeneti lista elemeit fordítva tartozó listát.

```
eljárás fordít k
        ha db k = 1 [ eredmény k ]
        eredmény fordít(k[1:]) + k[0:1]
vége

ki fordít [1, 2, 3, 4]      ; ki: [4, 3, 2, 1]
```

Az eljárás önhivatkozással oldja meg a feladatot: minden függvényhívásnál a lista első elemét az eredmény végére tesszük, elejére pedig a maradék fordítottját, amit a „fordít” eljárás ismételt meghívásával kapunk meg. Az első elemet tartalmazó részlistát a `0:1` indexszel választjuk ki, a második stb. elemet tartalmazó részlistát pedig az `1:` indexszel. Amint a „fordít” be-

menete egy elemet tartalmaz, azt adjuk vissza. Ez az önhivatkozó eljárás mélységi korlátja.

Az eljárás a bemeneti karakterláncot is megfordítja, ami jelzi, hogy a karakterláncok speciális listaként kezelhetők:

```
ki fordít „kultúra”          ; ki: „arútluk”
```

Megjegyzés: a Python gazdag függvényválasztékot biztosít a legtöbb feladathoz, így a listák megfordításához is. A „reverse” függvény helyben megfordítja a lista elemeit:

```
L = sor 5                    ; L = [0, 1, 2, 3, 4]
L.reverse()                  ; L = [4, 3, 2, 1, 0]
```

Többszörözés. A lista elemeit ismételve nagyobb listákat hozhatunk létre:

```
listanullaegyekkel = [0, 1] * 2   ; [0, 1, 0, 1]
listaszáznullaelemmel = [0] * 100 ; [0, 0, …, 0]
```

***N*-dimenziós tömbök**. A lista egydimenziós tömb adatszerkezetnek felel meg. Mivel azonban elemei listák is lehetnek, *n*-dimenziós tömböket is megvalósíthatunk velük:

```
ez tömb n m
  T = []       ; ide kerül n darab m elemű lista
  ism n [ T = T + [[0] * m] ]
  eredmény T
vége

T = tömb(3, 4)

ki T[0]       ; az első listaelem: [0, 0, 0, 0]

T[0][0] = 1                    ; [[1, 0, 0, 0],
T[2][3] = 2                    ;  [0, 0, 0, 0],
ki T                           ;  [0, 0, 0, 2]]
```

A példa kétdimenziós, vagyis sorokba és oszlopokba rendezett számokat tartalmazó tömb (mátrix) kezelését mutatja be. A tömb eljárás olyan n elemet tartalmazó listát ad vissza, ahol minden listaelem egy m darab nullát tartalmazó lista. Két index megadásával elérhetők és módosíthatók a listában lévő listák elemei.

A szótáraknál egy másik megoldást is megvizsgálunk az *n*-dimenziós tömbök létrehozására.

Halmazok

A halmazokban – szemben a listákkal – az elemek nem ismétlődhetnek, illetve nem számít az elemek sorrendje. Halmazokat listákból hozhatunk létre a **halmaz** utasítással.

```
A = halmaz [2, 3, 2, 1]      ; halmaz: {1, 2, 3}
```

Amint a példa jelzi, a halmazzá alakítással eltűnnek az ismétlődő elemek.

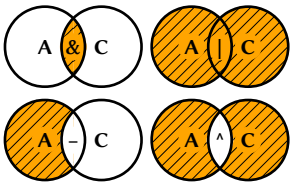
Megjegyzés: A Python 2.7-től már a halmazok szokásos matematikai jelölésmódjával, a kapcsos zárójelek közötti felsorolással is létrehozhatunk halmazokat. Amint a LibreOffice beépített Python 2.6-ja frissül egy újabb változatra, a LibreLogóban az előző példát így is írhatjuk majd:

```
A = {1, 2, 3}
```

Műveletek. A halmazokon alkalmazhatjuk a listaműveletek egy részét, ilyen az *elemszám* (db) és az *eleme-e* (-ban/-ben) művelet. Az *üres-e* műveletnél az elemszám mellett az üres halmazzal tesztelhetünk:

```
ha A = halmaz [] [ ki „üres halmaz” ]
ha db A = 0 [ ki „üres halmaz így is” ]
ha nem A [ ki „és üres halmaz így is” ]
```

Halmazműveletek. A **&**, **|**, **–**, **^** műveleti jelek a metszet, unió, különbség, szimmetrikus különbség halmazműveletek elvégzését teszi lehetővé.




```
; kivesszük a 3-at A-ból
A = A – halmaz [3]
; betesszük az 5-öt A-ba
A = A | halmaz [5]
```

Felsorolás. A halmazok elemeit nem érhetjük el index megadásával, de a listákhoz hasonlóan a fut -ban/-ben ciklussal végiglépkedhetünk rajtuk.

Rendezés. A halmazok elemeit sorba rendezhetjük a korábban látott rendez függvénnyel, amely a halmaz elemeit tartalmazó rendezett listával tér vissza.

Lista. Listára alakításra használhatjuk a lista parancsot is:

```
betűlista = lista halmaz „valamilyen szöveg”
```

A példa a halmaz függvény másik képességét is bemutatja: karakterlánc bemenet esetén a karakterlánc karaktereinek halmazát adja vissza.

Példa. A következő példaprogram Eratoszthenész prímszám-szitáját valósítja meg halmazok segítségével:

```
ez szita n
  számok = halmaz sor(1, n + 1)
  ism gyök n [
    szám = hányadik + 1
    ism n / szám [
      számok = számok – ~
        halmaz([szám * (hányadik + 1)]) ]
    ]
  eredmény számok
vége

ki szita 10          ; eredmény: [1, 2, 3, 5, 7]
```

A prímszám-szita algoritmusa (menete) a következő: vegyük a természetes számok halmazát 1-től n-ig. Vegyük ki 2, 3, ... n többszöröseit, így a végén csak a prímek maradnak a halmazban. A számokat felesleges n gyökénél tovább vizsgálni, mert az utána következő számok n-nél kisebb többszöröseit gyök n-nél kisebb szorzóval kaphatjuk meg, vagyis ezeket a számokat már egy korábbi lépésben kivettük.

A következő változat mintegy 50-szeres sebességgel hajtódik végre az előzőhöz képest:

```
ez szita n
  számok = halmaz sor(1, n + 1)
  ism gyök n [
    számok = számok – halmaz sor ~
      (2 * (hányadik + 1), n + 1, hányadik + 1)
  ]
  eredmény számok
vége
```

A programban a „sor” utasítással egy lépésben állítjuk elő a 2, 3 stb. számszorosait tartalmazó halmazokat, és vonjuk ki ezt a teljes halmazból, ami nagyságrendileg gyorsabb művelet, pl. fél perc alatt megkaphatjuk vele az egymilliónál kisebb prímeket egy átlagos gépen.

Fix listák

Nem módosítható lista adatszerkezet a fix lista.* Előnye, hogy gyorsabb a sima listánál, és hogy kulcsként szerepelhet szótárakban. Kapcsos zárójel nélkül, egyszerűen vesszővel választjuk el elemeit:

```
k = 1, 2, 3          ; kiíratásnál (1, 2, 3)-ként
l = 0,               ; kiíratásnál (0,)-ként
```

Ahol szükséges, mint az üres fix listánál vagy a ko-

* Python „tuple” adatszerkezet.

rábban látott formázott karakterláncoknál, zárójelet használunk:

```
üresfixlista = ()
```

`Fix`. A listákat fix listává alakító függvény:

```
F = fix [1, 2, 3]  ; (1, 2, 3)
L = lista F        ; vissza listává: [1, 2, 3]
T = fix tömb(3, 4) ; gyorsabb tömb (l. listák)
```

Értékadás fix listákkal. Egyszerre több változónak is értéket adhatunk, ha az értékadás jobb oldalán fix lista szerepel:

```
x, y = 5, 6     ; mint külön sorban x = 5, y = 6
```

Ez a megoldás az átmeneti változók használatát (például változók értékének felcserélésénél) feleslegessé teszi. A következő függvény ezzel egyszerűsíti a Fibonacci-sorozat elemeinek kiszámítását:

```
ez Fibonacci n
  f0, f = 0, 1
  ism n [ f, f0 = f + f0, f ] eredmény f0
vége

ki Fibonacci 10000  ; „336” és még 2087 számjegy
```

A Fibonacci-sorozat első eleme a 0, második az 1, a következő elemek pedig az előző két elem összegeként állnak elő: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 stb. A ciklus előtt f0=0, f=1, a ciklusban pedig az új f a régi f és az f0 (a két előző tag) összege lesz, míg f0 új értéke f régi értéke lesz.

Szótárak

A szótárakkal kulcs-érték párok formájában tárolhatjuk adatainkat, itt például a személynév a kulcs, és az életkor az érték:

```
ember = {„Kati”: 25, „Feri”: 33}
ki ember[„Feri”]        ; 33 kiírása
ember[„Zoé”] = 25       ; új kulcs-érték pár
ember = {}              ; törlés üres szótárral
```

Listához hasonló műveleteket használhatunk. A következő példában megfordítjuk az előző szótárt, azaz az életkorhoz rendeljük az adott életkorú személyek listáját:

```
kor = {}
fut k ember-ben [
  l = ember[k]
  ha l kor-ban [ kor[l] = kor[l] + [k] ] [
    kor[l] = [k]
  ]
]
ki kor     ; {25: [„Kati”, „Zoé”], 33: [„Feri”]}
```

A ciklusban szereplő feltételben megvizsgáljuk, hogy létezik-e az adott életkorhoz már lista a szótárban. Ha nem, akkor létrehozzuk, ha igen, akkor bővítjük. A programban szereplő feltételvizsgálat helyettesíthető a szótár típus „get” függvényével, amely a megadott kulcs hiánya esetén a második bemenő értékét adja vissza, a következő példában az üres halmazt:

```
kor = {}
fut k ember-ben [
  l = ember[k]
  kor[l] = kor.get(l, []) + [k]
]
```

***N*-dimenziós tömbök szótárral**. Vesszővel elválasztott indexekkel *n*-dimenziós tömbként használhatjuk a szótár adatszerkezetet:

```
a = {}
fut x sor 10-ben [     ; x 0-tól 9-ig megy
  fut y sor 10-ben [   ; y is a belső ciklusban
    a[x, y] = 0        ; a[0, 0]-tól a[9, 9]-ig
  ]
]
a[0, 0] = 5            ; bal felső sarokba 5
ki a[0, 0]             ; bal felső sarok kiírása
```

A példában 10*10-es kétdimenziós tömböt töltöttünk fel nullákkal, majd módosítottuk és kiolvastuk az egyik elemet.

A „vesszővel elválasztott indexek” valójában egy fix listának felelnek meg. Így nem is szükséges a tömb minden elemét megadni, hogy kezelni tudjuk (amivel tárhelyet és időt spórolhatunk meg, különösen óriási méretű tömböknél):

```
a={}
a[0, 0] = 1     ; megfelel a { (0, 0): 1 }-nek
a[2, 3] = 2
ki a[2, 3]              ; csak létező elemnél
ha (5, 5) a-ban [ ki a[5, 5] ]   ; ellenőrzéssel
ki a.get((5, 5), 0)     ; ha nincs ilyen elem, 0
```

A Python további gazdag adattípusválasztékot biztosít az adatfeldolgozáshoz. A következő (librelogósított) példa betölt egy szöveget a megadott útvonalon található egyszerű szöveges állományból, és kiírja a tíz leggyakoribb szavát a Counter („számláló”) osztály segítségével:

```
from collections import Counter
könyv = open(„konyv.txt”).read()
szavak = talál(„\w+”, könyv)
ki Counter(szavak).most_common(10)
```

Kérdések és válaszok

Hogyan illesszük be a LibreLogóval rajzolt képeinket Writer dokumentumainkba?

Ha több alakzatból állnak, foglaljuk a képet csoportba (az alakzatok kézi kijelölése és a Rajzobjektumok eszköztár Csoportosítás ikonja helyett célszerű a `kép` utasítást használni), majd vágólappal illesszük a dokumentumba. A horgonyt állítsuk az oldalról bekezdésre, ha szükséges. Ha a kép köré szöveget is futtatunk, szükség esetén helyezzük keretbe (a keretben a kép körbefuttatási opciója legyen a „háttérben”, hogy a keret pontosan körbefogja, a kézikönyvben szereplő ábrák többnyire így kerültek beillesztésre).

Hogyan exportáljuk más, pl. DTP programokba a LibreLogóval készített képeket?

Másoljuk a képet a LibreOffice Draw-ba, ahonnan EPS (beágyazott PostScript) vagy SVG formátumba is exportálni tudjuk. Ha a kép szöveget is tartalmaz, a vektoros betűkészletekkel kapcsolatos problémák elkerülhetők, ha a teljes ábrát, benne a szöveggel görbévé alakítjuk exportálás előtt.

A LibreLogo programok forráskódja és a rajzolt alakzatok ugyanazon a területen osztoznak, hogyan férhetünk el kényelmesebben az oldalon?

Kérhetünk fekvő oldalbeállítást és nagyobb oldalméretet is a Formátum » Oldal… párbeszédablakban, így a kezdőpozícióba helyezett teknőcnek több hely áll a rendelkezésére.

Szúrjunk be a dokumentum első sorába egy oldaltörést (Beszúrás… » Töréspont), ezzel üresen hagyjuk az első oldalt a rajzolás számára.

A teknőc rajzolás során elérte az oldal bal szélét, és a rajzolt alakzat „szétesett”. Mit lehet tenni?

Helyezzük a programot egy `kép` utasításba (ilyenkor a rajzolt alakzatot őrzi meg a program, nem pedig a

teknős és az általa rajzolt vonal szinkronját, amire a LibreOffice Writer képmegjelenítése miatt van szükség). A Draw, illetve Impress programban az így kapott alakzatcsoportot az oldal bal oldalán is kilógathatjuk. Bonyolultabb esetben oldalanként is összefűzhetjük PDF-kiadványainkat, vagy beállíthatunk kifutót/vágást is.

Van valamilyen billentyűkombináció a Logo programok gyors indítására, vagy a teknőc mozgatására?

Nincs, de létrehozhatunk az Eszközök » Testreszabás párbeszédablak segítségével, ahol lehetőség van a Logo eszköztár által indított makrók (eljárások) billentyűkombinációhoz való hozzárendelésére (l. Billentyűzet lap » LibreOffice-makrók kategória).

Miért nem használható a Draw-ban, illetve az Impressben a Logo eszköztár?

A Draw és az Impress működése némileg eltér a Writerétől, ezért átmenetileg nem használható a Logo eszköztár ezekben a programokban. A Writerben rajzol ábrák viszont a vágólap segítségével áthelyezhetők ezekbe az alkalmazásokba is.

Hogyan készült a tangram teknőc és piton?

Ha a teknőc és piton alakzatcsoport hátterét világosabb színűre állítjuk, láthatóvá válnak az alakzatok (l. kép). A következő LibreLogo forráskód állította elő a kiindulási tangramot:

```
ez tölt2 cím
  töltőszín tetszőleges tölt
  szöveg cím ; számozás (a 7.-nél korrigált)
vége

ez tangram n
  m = gyök n * n / 2
  e n j 90+45 e m tölt2 1
  h m b 45 e n tölt2 2
  j 90 e n/2 j 90+45 e m/2 tölt2 3
  h m/2 b 90 e m/2 j 90 e m/2 tölt2 4
  b 90 e m/2 b 180-45 e n/2 tölt2 5
  h n/2 j 90+45 e m/2 b 90+45 e n/2 tölt2 6
  e n/2 b 90 e n/2 tölt2 „\n    7”
vége

tangram 4cm
```

Hogyan írhatjuk át LibreLogo programjainkat Pythonra?

Ha a programunk nem tartalmaz teknőcgrafikát, akkor egyszerűen átírhatjuk Pythonra. A Python 2 része a LibreOffice-nak. Programjainkat parancssorban indíthatjuk el a **`python`** értelmezővel, például Linuxon a következő útvonalon:

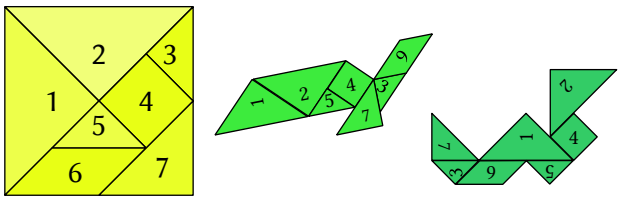
```
/opt/libreoffice/program/python sajat_program.py
```

Ha nem adunk meg futtatandó programot, megjelenik a Python parancssor jele (>>>), és közvetlenül adhatunk meg Python utasításokat: A Python értelmező számológépként használható, például a

```
2**132049-1
```

kifejezéssel $2^{132049}-1$ értékét, egy 39 751 jegyű számot ír ki a program pár oldalon keresztül.

Az akkori idők csúcsgépével, a Cray szuperszámítógéppel megtalált szám 1983 és 1985 között az ismert legnagyobb prímszám volt. A 2 hatványa mínusz 1 prímek, vagyis Mersenne-prímek jelenleg ismert legnagyobbikánál a hatvány 43 112 609. Ez a közel 13 millió jegyű szám egyben a jelenleg ismert legnagyobb prímszám is.



A parancssor további segítséget nyújt a Python nyelv elsajátításában, például a **`help`** eljárással angol nyelvű dokumentációt kaphatunk:

```
help(str)                ; karakterlánc súgó
help([])                 ; lista súgó
import re                ; regex mintaillesztő
help(re)                 ; regex súgó
```

A következő táblázat rövid összefoglalását adja, hogy a LibreLogo utasítások mely Python utasítások felelnek meg, illetve bizonyos szerkezetek hogyan feleltethetők meg egymásnak. A Python programnyelv sajátos tulajdonsága, hogy a programblokkokat nem zárójel, hanem behúzás jelöli. *Az egymás után azonos, vagy nagyobb behúzással kezdődő sorok tartoznak egy programblokkba.*

```
A = sor 5 10             A = range(5, 10)
C = rendez A             C = sorted(A)

; ciklus példa           # ékezet csak Python 3-tól
fut x sor 5-ben [ ]      for x in range(5):
                           pass       # üres blokk

ism 5 [ ki hányadik ]    for hanyadik in range(5):
                           print(hanyadik)
ez átlag x y             def atlag(x, y):
  eredmény (x+y)/2         return (x+y)/2.0
vége

ki átlag 5 6             print(atlag(5, 6))

amíg i < 3 [             while i < 3:
  ha i = 1 [               if i == 1:
    ki „egy”                 print('egy')
    újra                     continue
  ] [ ki „nem egy” ]       else:
  ki i                       print'nem egy')
]                            print(i)

kilép                    break
stop                     return
és/vagy/nem              and/or/not
5 A-ban                  5 in A
cos(pi)                  from math import cos, pi
                         cos(pi)
x = talál(„^a”, sz)      import re
                         x = re.find('^a', sz)
```

Példák

Elforgatott négyzetek

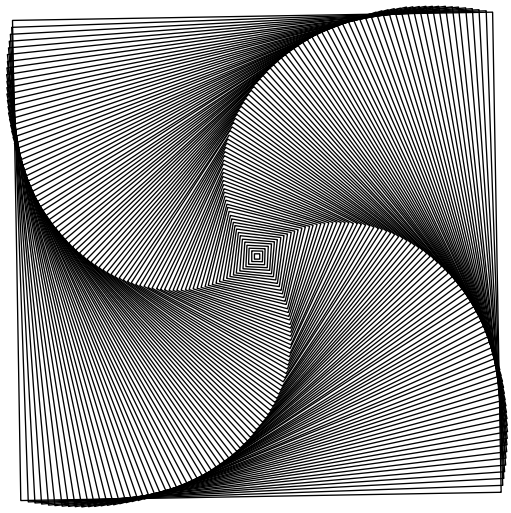
```
tlsz „láthatatlan”
ism 90 [ négyzet hányadik * 2 j 1 ]
```

Teknőcök

```
; betűkészlet Unicode 6.0 karakterekkel
tf betűcsalád „Symbola”
ism 8 [
  betűméret hányadik * 10 címke „🐢”
  h hányadik * 5
]
```

Színes ábrák

```
ez mackó
  kör 100 b 45 e 70 kör 50 h 70
  j 90 e 70 kör 50 h 70 j 45 h 20
  ism 2 [
    tlsz „fehér” kör 25
    tlsz „fekete” kör 10 e 40
  ]
  h 60 j 90 e 25 kör 30 h 25 b 180
vége

tf fut k [„arany”, „narancs”, „világospiros”, ~
    „ibolya”, „világoskék”]-ben [
  tlsz k kép [ mackó ] e 100 b 360/5
]
```



Betűgrafika

```
ez körbe méret felirat
  betűméret méret tollvastagság méret * 0,1
; négyzet nagysága, amibe a szöveg beleírható:
  n = egész kerekítés db(felirat)**0,5 + 0,49
; a maradék helyre szíveket teszünk:
  felirat = felirat + „❤” * (n*n – db felirat)
  k = 0
  l = 0
  fut m cserél(„ ”, „❤”, felirat)-ban [
    töltőszín [127 + vszám 128, ~
      64 + vszám 128, 64 + vszám 128]
    k = k + 1
    tollatle
    ha k = n [
      b 90-45 kör méret * 1,6  j 90-45
      betűszín „fehér” szöveg m
    ] [
      b 90 kör méret * 1,6
      j 90 betűszín „fehér” szöveg m
    ]
    ha k = n [
      k = 0
      l = l + 1
      j 90
      ha l = 1 [ n = n-1 ]
      ha l = 3 [ n = n-1 ]
      ha l > 3 [ l = 0 ]
    ] tollatfel e méret*1,6
  ]
vége

betűcsalád „Linux Biolinum G”
betűvastagság „kövér” j 90
k = be „Kérem a szöveget a kiíráshoz:”
kép [ körbe 40 k ]
```