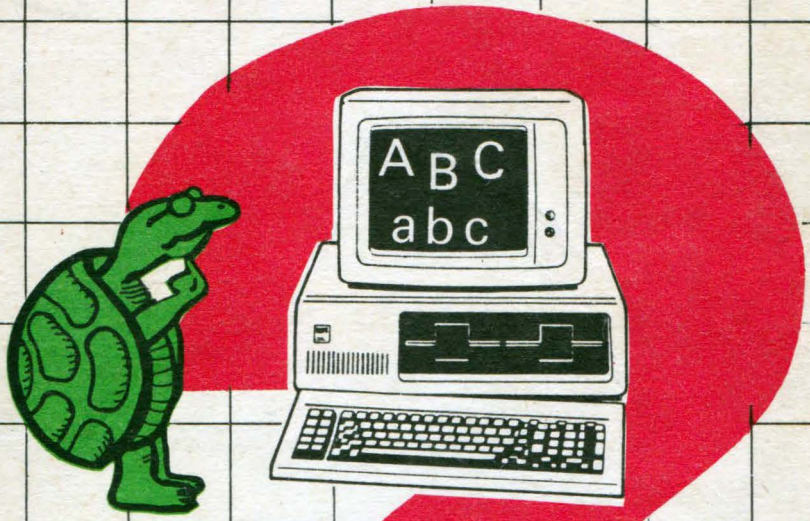
ION DIAMANDI

Cine știe LOGO



Biblioteca de Informatică

Editura Agni

Ion Diamandi

Cine ştie LOGO?

(Introducere în limbajul de programare LOGO)

Editura Agni

Bucureşti 1994

Tehnoredactare : *Cristina Mănoiu*
Coperta şi desene : *Adina Dumitriu*

ISBN 973-95626-9-8

Editura AGNI,
CP:30-107, BUCUREŞTI
tel: 615.55.59 fax: 312.93.33

CUPRINS

CUVÂNT ÎNAINTE

Pe la sfârșitul deceniului '70 și începutul celui următor, odată cu apariția calculatoarelor personale, tehnologia electronică astfel dezvoltată a început să fie utilizată și în sistemul de educație (școli, alte instituții de educație, familie). Uneltele (software-ul și metodologiile) erau împrumutate (neexistând altele) din domeniile în care utilizarea tehnicii de calcul era deja statuată (inginerie, economie etc). De exemplu, pentru inițierea în utilizarea calculatoarelor de către copii erau folosite mijloacele curente din vremea respectivă, FORTRAN și BASIC, cu întreg cortegiul de consecințe negative, atât în planul formării unor capacități cognitive superioare, cât și în planul psihologic.

În această conjunctură a apărut necesitatea dezvoltării unor instrumente specifice care să creeze un mediu propice pentru copii, atât pentru dezvoltarea unor aptitudini privind lucrul cu calculatoarele, cât și pentru organizarea unor activități creative.

Această dezvoltare s-a realizat în primul rând de către specialiștii din pedagogie și informatică (inteligență artificială) de la Massachusetts Institute for Technology care au pus la bază cercetările lui Piaget în psihologia modernă a dezvoltării intelectuale a copiilor. Seymour Papert a fost cel care a condus grupul de cercetători de la MIT pentru definirea noului limbaj pe care l-a denumit **LOGO** pentru a sublinia atât faptul că se referă la o dezvoltare logică a intelectului copiilor, cât și la posibilitatea lucrului cu cuvinte (LOGOS = CUVÂNT).

LOGO este un *limbaj de programare* dedicat învățării atât a introducerii în utilizarea calculatoarelor (având incorporate principiile cele mai moderne ale programării calculatoarelor provenite din inteligența artificială), cât și punerii la dispoziția educatorilor a unui *mediu de dezvoltare și organizare de activități de instruire* în care este implicat și calculatorul ca mijloc didactic. Rolul calculatorului în utilizarea acestui limbaj este de a oferi o unealtă, un mijloc didactic modern, care să vină în sprijinul elevilor.

Mai putem remarca faptul că popularitatea sistemului **LOGO** a crescut permanent, în momentul de față fiind implementat practic pe toate calculatoarele personale. În multe țări (Canada, SUA, Olanda, Bulgaria, Argentina, Costa Rica) învățământul primar și gimnazial nu mai poate fi conceput în afara acestui sistem. Publicațiile privitoare la sistemul **LOGO** au atins cote nebănuite existând, de exemplu, zeci de reviste permanente cu tematici **LOGO**.

Recentele dezvoltări ale sistemului **LOGO** au avut ca direcţii extensiile pentru geometrie plană şi în spaţiu (cercetări dezvoltate cu precădere în Bulgaria şi Portugalia), extensii pentru editări-prelucrări de texte cu desene, eventual animate - cunoscute sub numele de **LOGO WRITER** (aplicaţii dezvoltate în învăţământ în special în SUA şi Canada), extensii şi aplicaţii pentru interfeţe pentru roboţi didactici şi materiale de tip **LEGO-LOGO** (Bulgaria, Anglia, Franţa).

Lucrarea de faţă are drept prim scop punerea la dispoziţia elevilor şi profesorilor din cursul gimnazial a unui manual pentru organizarea de activităţi cu sistemul **LOGO**, fiind în concordanţă cu intenţia Ministerului Învăţământului de a introduce disciplina informatică la nivel gimnazial. Astfel, cea mai mare parte a lucrării este organizată în vederea utilizării la clasă, adică pe lecţii, fiecare din acestea având o parte teoretică şi una practică. Pentru partea practică se propun spre rezolvare aproape 80 de probleme.

Manualul se poate utiliza la oricare din clasele V - VII de gimnaziu, numărul de lecţii fiind calculat astfel încât să acopere perioada unui an şcolar cu două ore de informatică săptămânal (una de teorie şi una de practică) aşa cum prevede programa de învăţământ gimnazial propusă pe următorii ani. În principiu, manualul nu solicită elevilor alte cunoştinţe decât cele pe care le-au însuşit la nivelul anului respectiv. O mică excepţie o reprezintă, în cadrul doar a 3 probleme propuse, cunoştinţele despre teorema lui Pitagora şi a unor funcţii trigonometrice. Evident, cadrele didactice care vor utiliza manualul la nivelul clasei a V-a şi a VI-a vor putea sări peste aceste puţine probleme care fac apel la cunoştinţe suplimentare.

În ţara noastră experimentarea sistemului **LOGO** la nivelul învăţământului gimnazial s-a făcut foarte timid, unul din obstacolele întâlnite fiind legat de răspândirea şi popularitatea foarte mare a limbajului BASIC. De fapt, trebuie să recunoaştem că între cele două stiluri amintite există o concurenţă. Materialul de faţă, însă, prezintă o particularitate (originală) şi anume tratează probleme similare cu cele prezentate în limbajele BASIC, modul de abordare şi realizare fiind deseori asemănător, prin aceasta căutându-se să nu se creeze o antagonie între aceste sisteme.

Deci, hai să învăţăm **LOGO** !

AUTORUL

TASTATURA, ECRANUL ŞI BROASCA ŢESTOASĂ

Lecţia 1

Instalarea programului LOGO

Ne propunem să realizăm activităţi şi să rezolvăm probleme prin intermediul limbajului **LOGO** .

Spre deosebire de BASIC, pentru a folosi comenzile limbajului **LOGO** va trebui, în prealabil, să încărcăm în memoria calculatorului programul care va asigura înţelegerea de către calculator a cuvintelor **LOGO**. Încărcarea programului **LOGO** se face cu comanda:

`LOAD " "` sau `LOAD "logo"`	dacă folosim caseta magnetică
şi	
`LOAD * " d " ; 1 ; "logo"`	dacă folosim discheta

După încărcare pe ecran apare "semnul întrebării" (?). El se mai numeşte *prompterul* **LOGO**, acest cuvânt însemnând în limba engleză "gata", adică, în cazul nostru, "*sunt gata pentru comanda următoare*". Acum, calculatorul aşteaptă să introducem comenzi pe care le va executa.

Tastatura

Comenzile pe care dorim să le execute calculatorul le introducem de la tastatură acţionând, deci, butoanele (tastele) aflate pe consola din faţa noastră. Pe fiecare tastă este gravată o literă sau o cifră; *apăsarea tastei face să apară pe ecran litera sau cifra respectivă.* "Spaţiul liber" (tasta SPACE) sau mai pe scurt "spaţiul" este considerat şi el un semn (caracter).

Caracterele speciale (+ - / * ? = ; și altele) apar, fiecare, în colțul din dreapta sus al unei taste obișnuite. Pentru a face ca un astfel de semn să apară pe ecran este necesar să apăsăm tasta respectivă, dar ținând apăsată, în același timp și tasta SS (ca în BASIC).

Majusculele (literele mari) apar pe ecran dacă, în timp ce se tastează litera respectivă, se ține *apăsată și tasta CS.*

Observați că anumite reguli sunt comune utilizării calculatoarelor. De exemplu: tasta SS (SYMBOL SHIFT) reprezintă trecerea la simbol, prin acționarea ei obținându-se semnele dispuse pe partea superioară a tastelor (ele sunt, de fapt, simboluri); cu tasta CS (CAPS SHIFT) se obțin literele mari, "caps" fiind prescurtarea de la "capitals", care înseamnă în engleză, "majuscule".

Puteți exersa tastând cu un spațiu între ele grupuri de câteva litere (mici sau mari), cifre și caractere speciale.

Ecranul

Fig. 1

Priviți cu atenție ecranul (fig.1); pe el se vede, la începutul rândului scris de noi, "semnul întrebării" (prompterul **LOGO**) iar, la sfârșitul rândului, un pătrat negru clipitor, numit *cursor*. La început cursorul s-a aflat lângă prompt, mutându-se apoi la dreapta, pe măsură ce introduceam caracterele, făcând astfel loc caracterelor tastate.

Cursorul arată, deci, *poziția pe ecran unde va apare (se va afișa) caracterul ce urmează a fi tastat.* Deoarece cursorul se mută automat spre dreapta după orice tastare, putem spune că întotdeauna *caracterul tastat se introduce înaintea (la stânga) cursorului.*

În scopul corectării textelor afișate pe ecran avem nevoie de câteva manevre care, după cum veți vedea, sunt asemănătoare cu cele folosite în BASIC.

Ștergerea unui caracter aflat la stânga cursorului (adică înainte de acesta) se realizează ținând apăsată tasta CS împreună cu tasta 0 (zero), pe aceasta din urmă găsind scris și cuvântul **DELETE** care înseamnă "*șterge*". Acum puteți, de exemplu, să ștergeți de pe ecran ultimul grup de caractere tastat anterior.

Evident, pentru a efectua o modificare (ștergerea sau introducerea de caractere) în interiorul unui text oarecare va trebui să "ducem" mai întâi cursorul la dreapta poziției respective și abia apoi să realizăm ștergerea sau introducerea caracterelor dorite.

Deplasarea cursorului în interiorul unui text, pe un rând al ecranului, se face prin:

CS+5	(acționarea în același timp a tastelor CS și 5) pentru cursor la stânga (←)
CS+8	pentru cursor la dreapta (→)

Se observă că deplasarea cursorului la stânga și la dreapta se face ca în BASIC. Aceste manevre deplasează cursorul prin text, dar niciodată în afara acestuia.

Puteți să vă antrenați ștergând de pe ecran primul grup de caractere și apoi să corectați celelalte grupuri astfel încât să conțină câte cinci caractere de același tip. Dacă "textul" astfel obținut depășește un rând al ecranului, acest lucru va fi semnalat de calculator prin afișarea semnului de "continuare" ("!") la sfârșitul rândului care continuă.

Într-un text care se continuă pe mai multe rânduri, cursorul poate fi deplasat și "pe verticală", de la un rând la altul, prin manevrele:

CS+6	pentru cursor în jos (↓)
CS+7	pentru cursor în sus (↑)

Probabil că ați remarcat, în colțul din dreapta jos al ecranului, o literă stingheră (" l "); această literă reprezintă *modul de lucru al tastaturii*. Tastatura poate fi într-unul din următoarele moduri (regimuri):

l -	pentru scrierea normală, cu litere mici (l reprezentând prima literă a cuvântului "letters" care înseamnă în engleză "litere")
C -	pentru scrierea cu majuscule (C reprezintă prima literă a cuvântului "capitals" - litere mari)
E -	pentru scrierea "extinsă", folosită în **LOGO** doar pentru unele efecte speciale

Trecerea tastaturii dintr-un regim în altul se poate face cu următoarele manevre:

mod "l" ↔ mod "C"	**CS+2**
mod "l" sau "C" ↔ mod "E"	**CS+SS**

Deci, pentru trecerea din regim "l" în regim "C" și invers, se vor acționa împreună tastele CS și 2, iar pentru trecerea în regim "E" sau ieșirea din regim "E" se vor acționa împreună tastele CS și SS.

În **LOGO**, spre deosebire de BASIC, toate comenzile se scriu, tastându-se literă cu literă , folosindu-se litere mari sau mici.

Pentru a vă antrena puteți tasta cuvinte sau propoziții. Puteți folosi litere mari sau mici, la alegere. Dacă greșiți, folosiți DELETE. Când terminați, nu uitați să apăsați tasta CR.

Observați rezultatele afișate pe ecran.

Lecția 2

Comenzi LOGO

Comenzile **LOGO** se introduc prin tastarea unor cuvinte (după cum am văzut, literă cu literă) care au semnificație pentru calculator. De aceea se mai numesc și *cuvinte cheie.*

Comenzile în **LOGO** sunt reprezentate de cuvinte în limba engleză sau prescurtările acestora (de obicei două litere: prima și ultima literă a cuvântului). Veți vedea că *aceste cuvinte sunt astfel alese încât să "descrie" cât mai bine un lucru sau o acțiune pe care o va realiza calculatorul.* De asemenea se pot introduce și comenzi în limba română, existând bineînțeles la unele dintre ele și forma lor prescurtată (de obicei primele două litere ale cuvântului). Toate aceste forme (în engleză și română sau prescurtările lor) au același efect. La început, pentru deprindere, vom folosi formele în engleză și română, iar apoi, formele prescurtate, acestea fiind mai comode.

Comenzile **LOGO** *simbolizează o acțiune sau un lucru. Ele pot fi scrise mai multe pe același rând de ecran, dar cu condiția de a fi separate între ele cel puțin printr-un spațiu* (SPACE). Se formează în acest fel o *linie de comenzi multiple.* O astfel de succesiune de comenzi se poate continua chiar pe mai multe rânduri ale ecranului. *O comandă nu se execută decât dacă la terminarea ei* (sau la terminarea înșiruirii mai multor comenzi între care se află) *se apasă tasta CR.* Ca și în BASIC, *acționarea acestei taste comunică faptul că am terminat ceva adresat calculatorului și acum este rândul său să execute ceea ce i s-a transmis.* Deci, acționarea tastei CR semnifică terminarea liniei **LOGO** dar, în același timp, produce și lansarea în execuție a acesteia; comenzile liniei sunt executate rând pe rând, pînă la sfârșit, iar după executarea ultimei, pe ecran apare din nou prompterul "**?**".

Și acum, în sfârșit, să învățăm prima comandă **LOGO**. Aceasta este **PRINT** (sau prescurtat **PR** sau echivalentul ei în limba română, **SCRIE**); folosirea ei are ca efect *afișarea unui rezultat.* Introduceți, de exemplu:

```
? PRINT 3+2
```

Se observă că după **PRINT** s-a introdus un spațiu (SPACE) pentru a separa acest cuvânt cheie, care reprezintă o comandă, de restul liniei. Convenim, de asemenea, să scriem toate comenzile cu litere mari pentru a atenționa asupra faptului că reprezintă niște cuvinte speciale (cheie). Când lucrați efectiv, puteți însă să introduceți aceste cuvinte cu *litere mari sau cu litere mici, sistemul înțelegând, deopotrivă, ambele forme.*

Semnul de întrebare reprezintă, după cum știm, promptul **LOGO**. Nu uitați ca la sfârșitul liniei să acționați tasta CR. Calculatorul va afișa:

```
5
?
```

Deci, calculatorul a afișat rezultatul operației de adunare a lui 3 cu 2 (adică 5) și apoi a afișat, din nou, promptul **LOGO** pentru a indica faptul că și-a terminat "treaba" și acum așteaptă introducerea altor comenzi.

Ca și în BASIC putem introduce afișarea rezultatului aplicând diferite operații aritmetice cu ajutorul următoarelor semne:

+	pentru adunare
–	pentru scădere
*	pentru înmulțire
/	pentru împărțire

Ordinea efectuării operațiilor este cea știută, și anume, *se execută mai întâi operațiile de înmulțire și împărțire iar apoi cele de adunare și scădere.* De asemenea, se pot folosi pentru efectuarea de calcule aritmetice *parantezele*, dar numai cele *rotunde*. Pentru numere zecimale se folosește *punctul zecimal* (și nu virgula). De exemplu 2,3 se reprezintă 2.3.
Tot cu comanda **PRINT** (**SCRIE**) putem afișa pe ecran și cuvinte, cu condiția ca ele să fie precedate de semnul ghilimele. De exemplu:

```
?SCRIE  "elev
elev
?
```

Spre deosebire de BASIC observați faptul că nu este necesară închiderea ghilimelelor la sfârșitul cuvântului.

Dacă dorim să scriem propoziții (de fapt mai multe cuvinte separate prin spații) va trebui să încadrăm întreaga secvență de cuvinte între paranteze drepte. Exemplu:

```
?SCRIE      [Sunt elev in clasa a 7-a]
Sunt elev in clasa a 7-a
?
```

Activități practice

1. Realizați prin **LOGO** calculul următoarei expresii aritmetice:

$E = 3 + \{5,2 - [4 \cdot 7 - (30 + 20) : (3+2) \cdot 2,1 + (4 + 10,3) \cdot (4 - 0,3)] \cdot 3,5 + 4,3 \cdot 2 - (1,57 \cdot 3) + 12,4\}$

2. Introduceți comenzile **LOGO** pentru ca să se afișeze pe ecran următoarele rezultate:

```
10
TOTAL
T O T A L
2+2=4
2+1=2
Ultimul rezultat este gresit
TOTAL 6
```

Dacă vreunul din cuvintele din linia executată nu este recunoscut de calculator ca fiind o comandă **LOGO** sau dacă aceasta a fost scrisă greșit (atenție, deci, la ortografie), execuția se întrerupe și apare mesajul:

```
NU STIU CUM SA ...
```

urmat de cuvântul respectiv.

Încercați de exemplu cu:

```
?SCRIE    "DESENE TRASEAZA CERC
DESENE
NU STIU CUM SA TRASEAZA
```

Același rezultat îl obțineți și dacă introduceți:

```
?SCRIE   "T O T A L
T
NU STIU CUM SA O
```

După comanda **SCRIE** sistemul a întâlnit litera **T** pe care a considerat-o un cuvânt, deoarece după ea urmează un spațiu. Conform comenzii **SCRIE**, a afișat acest cuvânt, încheind, deci, acțiunea pentru comanda dată. În continuare, întâlnește un nou cuvânt, O (de fapt tot o literă), pe care îl interpretează ca o comandă. Însă nu recunoaște această comandă și, în consecință, afișează mesajul.

Deci sistemul ne atenționează în aceste situații prin *mesaje de eroare,* propoziții care dau informații referitoare la greșelile făcute, în scopul corectării lor. După apariția unui mesaj de eroare, apăsarea oricărei taste va provoca reîntoarcerea la modul comandă ("?") urmând ca eroarea să fie îndepărtată prin tastarea unei comenzi (sau șir de comenzi) diferite.

Lecția 3

Convenția broaștei țestoase

Până acum am învățat o comandă cu care afișam pe ecran rezultatele sau scriam propoziții. Cu alte cuvinte, realizam pe ecran *texte.* Dar putem desena în **LOGO?**

Așa cum desenăm, mișcând vârful creionului pe o coală de hârtie, tot astfel, cu ajutorul unui indicator (schemă) numit convențional "*broască țestoasă*" putem realiza orice formă grafică pe ecran. Pentru aceasta, trebuie să ne imaginăm mai întâi figura pe care dorim să o desenăm și să o descompunem într-un set de elemente cât mai simple, cărora le vom asocia ulterior comenzile corespunzătoare, pentru ca "broasca țestoasă" să deseneze pe ecran imaginea dorită.

De fapt, toate comenzile în **LOGO** le adresăm acestui simpatic personaj care a fost ales drept colaboratorul nostru datorită calităților pe care le întruchipează: meticulozitate, perseverență, executând pe rând toate operațiile necesare atingerii scopului propus.

Pentru simplificare, vom folosi de aici înainte pentru acest personaj un singur cuvânt și anume *broasca*. Practic *toate comenzile pe care le introducem reprezintă ordine pe care broasca le va executa.*

Broasca se poate deplasa conform indicațiilor noastre, iar "urma" lăsată pe ecran va constitui realizarea noastră grafică.

Să învățăm și câteva comenzi grafice.

Broasca țestoasă devine *vizibilă* dacă tastăm comanda **SHOWTURTLE** sau prescurtat **ST**, adică "arată broasca!". De asemenea putem folosi și comanda în limba română, care este **BROASCA**.

La terminarea comenzii de mai sus, în centrul ecranului apare "broasca țestoasă", sub forma unui mic triunghi, el seamănă, mai degrabă, cu un vârf de săgeată, vârful cel mai ascuțit al triunghiului indicând încotro este orientată (îndreptată) broasca.

Inițial, broasca se află în *centrul ecranului*, aici fiind "*casa*" ei și este *orientată spre nord* (fig. 2). Putem aduce broasca acasă (adică în centrul ecranului și cu orientarea spre nord), oriunde s-ar afla ea, cu comanda **HOME** sau, în limba română, **ACASA**.

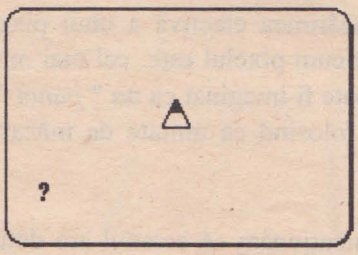


Fig. 2 Poziția și orientarea inițială a broaștei

Convenția folosită pentru orientare este similară cu cea obișnuită de la busolă sau roza vânturilor, fiind considerată orientare nord partea de sus a ecranului (fig. 3).

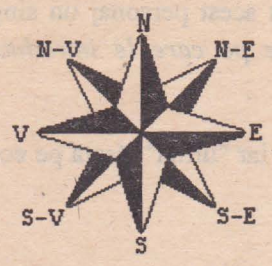


Fig. 3 Roza Vânturilor

Broasca poate fi făcută oricând invizibilă cu comanda **HIDETURTLE** sau prescurtat **HT** , adică "ascunde broasca". În română putem tasta comanda similară **FARABROASCA** sau prescurtat **FB**. Să notăm că **BROASCA** și **FARABROASCA** nu modifică nici locul broaștei pe ecran și nici orientarea ei (direcția vârfului).

Mișcările broaștei țestoase

Partea utilă a ecranului are *22 de rânduri*, fiecare rând având *32 de poziții pentru caractere*; alte două rânduri de ecran, situate sub cele 22 amintite, sunt utilizate pentru afișarea comenzilor curente, dacă partea "principală" este ocupată pentru operații grafice.

Spațiul rezervat pentru un *caracter* este un mic *pătrat format din 8 x 8 = 64 elemente de imagine* sau *pixeli* ; ecranul are, deci, pe orizontală 32 x 8 = 256 pixeli, iar pe verticală, 22 x 8 = 176 pixeli. Mărimea efectivă a unui pixel depinde, evident, de mărimea ecranului nostru; oricum pixelul este cel mai mic element care poate apare pe ecran - de aceea el poate fi imaginat ca un " *punct* ". Astfel, ne vom referi la distanțele de pe ecran folosind ca unitate de măsură punctul, subînțelegând, de fapt, pixel.

Înainte de a trece la realizarea unor desene, să menționăm că ecranul are două regimuri de funcționare:

- regimul " textual ", în care ecranul este utilizat pentru texte
- regimul " grafic ", în care ecranul este folosit pentru desene

Inițial, ecranul este în regim textual, utilizând toate cele 22 de rânduri pentru texte; *la prima comandă grafică* (de exemplu **BROASCA** sau **ACASA**) *ecranul trece automat în regimul grafic.*

Fiind în acest regim, comenzile curente (pe care le introducem) precum și afișarea rezultatelor care nu implică grafică ci numai text, vor fi afișate pe cele două rânduri "suplimentare".

Revenirea din regimul grafic în cel textual se face cu comanda **TEXTSCREEN** (prescurtat **TS**), adică, "ecran textual".

Pentru mișcările broaștei pe ecran se pot folosi două feluri de comenzi:

- *comenzi pentru deplasarea* broaștei (înainte și înapoi)
- *comenzi pentru rotirea* (schimbarea direcției) broaștei (la stânga și la dreapta)

Pentru ca broasca să se deplaseze, ca urmare a unui ordin al nostru, ea trebuie să "știe" câți pași trebuie să facă; comanda respectivă trebuie să conțină, deci, pe lângă cuvântul cheie corespunzător și indicația cantitativă privind mărimea deplasării. O astfel de indicație valorică se mai numește *subiectul comenzii, parametru, parametru de intrare* sau chiar *intrare* și este separată de comanda propriu-zisă printr-un spațiu.

FORWARD 50 adică "*mergi înainte 50 de pași!*", va avea ca efect deplasarea broaștei 50 de pași în direcția spre care a fost orientată (vezi fig. 4). Același efect îl va avea și forma prescurtată a comenzii, **FD 50** , precum și comenzile în limba română **INAINTE 50** sau **IN 50**

Fig. 4 Comanda **INAINTE**

Pentru început, vom comanda deplasări suficient de scurte pentru ca broasca să nu depășească marginile ecranului. Dacă, totuși, se va întâmpla o astfel de depășire, vom constata că broasca ieșită din ecran, "reapare" din partea opusă a acestuia.

Putem remarca faptul că, *dacă parametrul de intrare al comenzii înainte este negativ* (exemplu **INAINTE -20**), atunci are loc o *retragere a broaștei* cu numărul respectiv de pași.

Există, însă, și o comandă specială pentru retragerea broaștei:
BACK 20 sau **BK 20**, adică "*mergi înapoi 20 de pași!*" (vezi fig. 5). Același efect îl va avea, bineînțeles, și comanda în limba română **INAPOI 20** sau **IP 20**

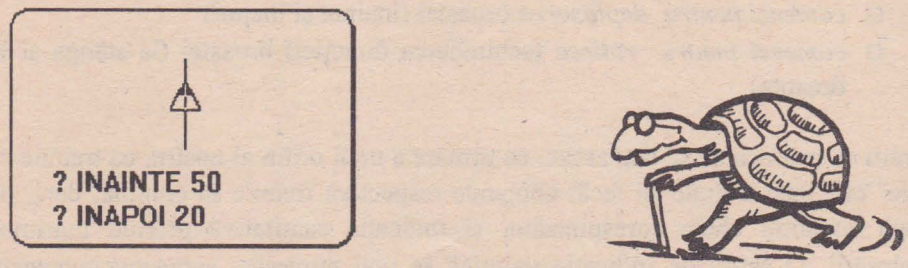


Fig. 5 Comanda **INAPOI**

La fel ca la comanda **INAINTE**, dacă parametrul de intrare al comenzii **INAPOI** este negativ, atunci are loc o *înaintare a broaștei* cu numărul de pași respectiv.

> Rezultă, deci, că schimbarea semnului parametrului de intrare al comenzilor **INAINTE** și **INAPOI** inversează efectul acestora

Comenzile **INAINTE** și **INAPOI** deplasează broasca înainte sau înapoi, dar numai de-a lungul direcției pe care broasca era orientată dinainte.

Schimbarea direcției de deplasare se poate realiza prin *rotirea broaștei* (la stânga sau la dreapta) cu un unghi corespunzător; mărimea unghiului se exprimă în grade (și eventual în fracțiuni zecimale). Unghiul citat se măsoară de la verticală spre partea superioară și în sensul orar (vezi fig. 6).

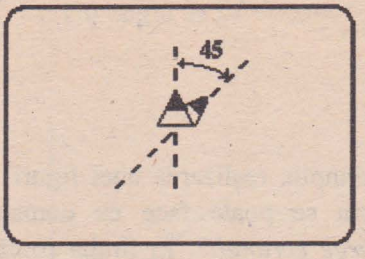


Fig. 6 Măsurarea unghiului

Comanda **LEFT 45** sau **LT 45** va însemna pentru broască "la stânga 45 de grade" (vezi fig. 7). Similar în limba română avem **STINGA 45** sau **SA 45**

Fig. 7 Comanda **STINGA**

RIGHT 90 sau **RT 90** va determina rotirea broaștei spre dreapta cu un unghi de 90 de grade. Similar în limba română avem **DREAPTA 90** sau **DR 90**
Schimbarea semnului parametrului de intrare al comenzilor **STINGA** și **DREAPTA** inversează efectele acestor comenzi: cu alte cuvinte **STINGA -60** este echivalentă cu comanda **DREAPTA 60**

> Deci sensul rotirii (din comanda pe care o dați) se referă la "stânga" sau "dreapta" broaștei. De aceea, la început, un antrenament indicat pentru realizarea unor desene "bune" constă în a vă identifica cu broasca țestoasă și astfel a încerca să realizați mișcările acesteia

Trebuie remarcat și caracterul diferit pe care îl au subiectele comenzilor pe care le-am studiat: cele de la comenzile **INAINTE** și **INAPOI** determină *mărimea* figurii, în timp ce, pentru comenzile **DREAPTA** și **STINGA** determină *forma* figurii. Dacă la oricare din comenzile de deplasare și rotire a broaștei, se omite din greșeală specificarea valorii parametrului de intrare, calculatorul se oprește din execuție, după ce afișează un mesaj: "prea puține intrări **FD**" sau "prea puține intrări **LT**".

Activități practice

Înainte de a începe o activitate practică (de exemplu, realizarea unei figuri) se recomandă "*curățarea*" ecranului. Acest lucru se poate face cu comanda **CLEARSCREEN** sau prescurtat **CS**, adică "*șterge ecranul!*". În limba română vom spune **STERGE**. Această comandă șterge "tot ecranul și readuce broasca în poziție inițială" (în centrul ecranului), redându-i și orientarea inițială.
Indicați secvențele de comenzi **LOGO** pentru desenarea pe ecran a următoarelor figuri (fără liniile punctate, acestea indicând doar direcția broaștei, și fără săgeți și numere, deci, doar liniile drepte care alcătuiesc figurile):

3. Un pătrat cu latura de 50 de pași (puncte) (fig. 8).

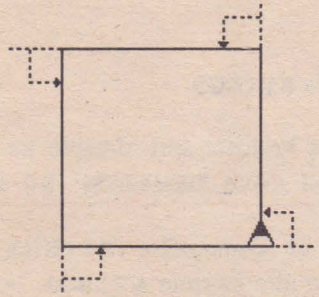


Fig. 8 Pătrat

Notă. Desenarea broaștei indică poziția și direcția acesteia după desenarea figurii.

4. Un dreptunghi cu lungimea de 70 și lățimea de 30 de pași (fig. 9).

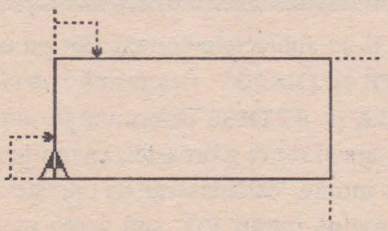


Fig. 9 Dreptunghi

✍ 5. Un unghi de 60 de grade (fig. 10).

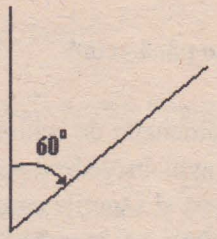


Fig. 10 Unghi

✍ 6. Un triunghi echilateral cu latura de 40 de pași (fig. 11).

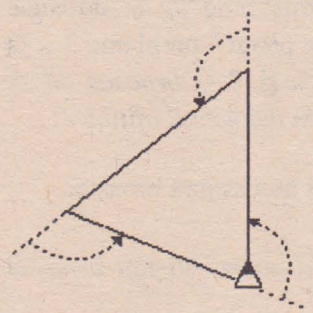


Fig. 11 Triunghi echilateral

✍ 7. Trasați diagonala pătratului din figura 8. Veți obține triunghiul din figura 12. Cum este diagonala pătratului față de laturi. Indicați comenzile **LOGO** pentru trasarea triunghiului din figura 12.

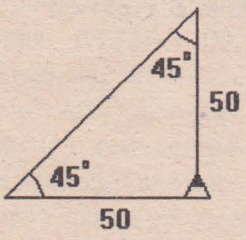


Fig. 12 Triunghi

Lecția 4

Trasee și drumuri radiale

Să observăm mai întâi câteva lucruri din figurile realizate până acum:

❶ Pentru indicarea valorii parametrului de intrare la comenzile de schimbare a direcției de deplasare (**STINGA** și **DREAPTA**) trebuie avut în vedere *mărimea unghiului exterior* figurii geometrice, după cum sugerează și săgețile care arată, de fapt, schimbarea direcției broaștei (rotirea). Nu uitați să încercați să vă "puneți în pielea broaștei" pentru înțelegerea acestui fapt.

❷ Așa cum a fost inițial și indicația din figuri, este bine ca la desenele (construcțiile) geometrice care se *închid* (de exemplu la pătrat - problema 3 -, la dreptunghi - problema 4 -, la triunghi - problemele 6 și 7 -) *broasca să se găsească în final în același loc și în aceeași poziție ca în momentul inițial.*

Impunându-ne această metodă vom avea un control mai bun asupra broaștei.

❸ Pentru trasarea unor *figuri geometrice care se închid, suma rotirilor broaștei este de 360 de grade.*
De exemplu, pentru problema 3, broasca s-a rotit de 4 ori la stânga câte 90 de grade (4 x 90 = 360); pentru problema 4, broasca s-a rotit de 4 ori la stânga câte 90 de grade (4 x 90 = 360); pentru problema 6, broasca s-a rotit de 3 ori la stânga câte 120 de grade (3 x 120 = 360). Acest fapt mai este cunoscut sub numele de *teorema rotirii complete a broaștei.* Pentru a experimenta această teoremă plecați cu broasca "de acasă" și parcurgeți un drum întâmplător ca cel din fig. 13, ajungând în final cu broasca în aceeași poziție ca cea inițială.

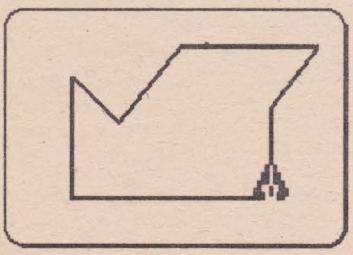


Fig. 13

Faceți *suma algebrică a rotirilor broaștei* ținând cont de faptul că o comandă ca **STINGA 50** este similară cu o comandă **DREAPTA -50**
Ce observați?

Exemplificare:

Comenzile pentru figura 13:

```
INAINTE  30   DREPTA  60   INAINTE  20   STINGA  150
INAINTE  40   STINGA  45   INAINTE  25   DREAPTA  90
INAINTE  15   STINGA 135   INAINTE  35   STINGA  90
INAINTE  52   STINGA  90
```

Să facem *suma algebrică a rotirilor broaștei*:

	DREAPTA	STINGA
	60	150
	90	45
		135
		90
		90
TOTAL	150	510

Suma algebrică:
Total rotiri **STINGA** – Total rotiri **DREAPTA** = 510 – 150 = 360
Se observă că *suma algebrică a rotirilor broaștei este de 360 de grade.*

Să mai învățăm câteva comenzi pentru a putea realiza desene.

Am văzut că pentru ștergerea ecranului și readucerea broaștei acasă se folosește comanda **STERGE**. Dar, uneori, readucerea acasă poate fi necesară și în unele cazuri în care nu dorim să dispară conținutul ecranului și, atunci, vom folosi comanda **ACASA**. Această comandă însă produce și trasarea pe ecran a drumului de revenire, ceea ce este (de cele mai multe ori) neconvenabil. Pentru a înlătura inconvenientul, se poate da în prealabil broaștei comanda de a *nu* trasa drumul pe care îl parcurge. Această comandă este: **PENUP** sau prescurtat **PU** , adică "peniță sus!". În limba română vom spune că broasca este **FARACREION** sau prescurtat **FC**.

Din momentul în care primește această comandă, broasca țestoasă va rămâne cu "penița ridicată" (adică fără creion) până ce va primi comanda contrară: **PENDOWN**, sau prescurtat **PD**, adică, "penița jos!". În română se va da comanda **CREION** sau prescurtat **CR**. Astfel, *revenirea "acasă"*, fără trasarea drumului de întoarcere și fără ștergerea ecranului se va realiza cu ajutorul succesiunii de comenzi:

```
FARACREION ACASA CREION
```

Recapitularea comenzilor învățate

PRINT(SCRIE)-afișează rezultate sau șiruri de caractere pe ecran

```
FORWARD (INAINTE)
BACKWARD (INAPOI)
RIGHT (DREAPTA)
LEFT (STINGA)
SHOWTURTLE (BROASCA)
HIDETURTLE (FARABROASCA)
CLEARSCREEN (STERGE)
HOME (ACASA)
PENUP (FARACREION)
PENDOWN (CREION)
TEXTSCREEN
```

Observați că putem deja încadra comenzile învățate în două categorii: unele care necesită parametru (subiect) de intrare (vom vedea că alte comenzi necesită chiar mai multe intrări), iar din această categorie fac parte: **INAINTE, INAPOI, DREAPTA, STINGA, SCRIE**, în timp ce celelalte nu necesită parametru de intrare, iar din această categorie fac parte: **STERGE, ACASA, FARACREION, CREION, TEXTSCREEN, BROASCA, FARABROASCA**.

Activități practice

Realizați prin intermediul comenzilor **LOGO** următoarele desene geometrice:

8.

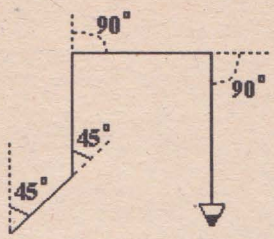


Fig. 14

9.

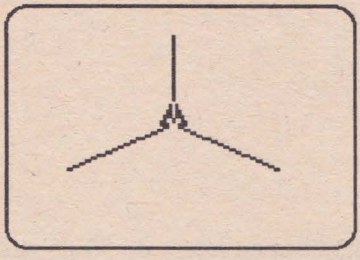


Fig. 15

10.

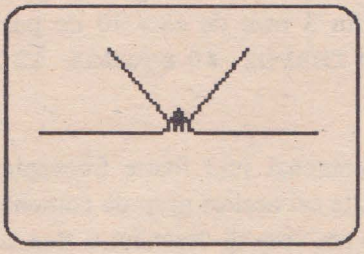
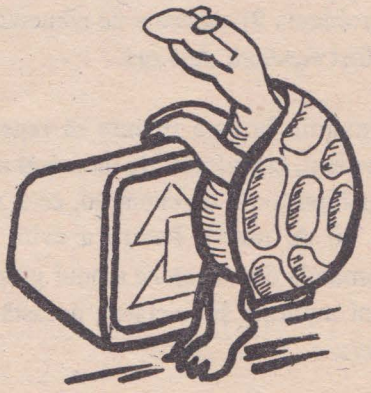


Fig. 16



11. Un pentagon cu laturile egale.

12. Arcul (resortul) din fig. 17.

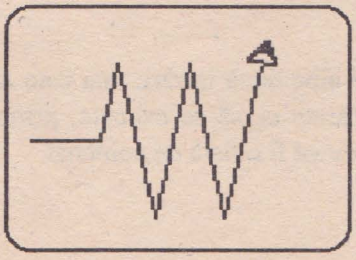


Fig. 17 Resort

13. Arcul (resortul) din fig. 18.

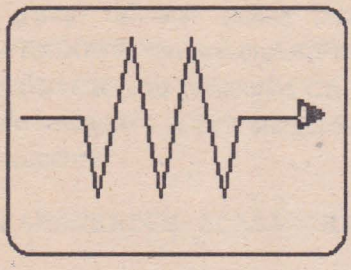


Fig. 18 Resort

Lecția 5

Instrucțiunea de repetare (ciclare)

Să reluăm problema numărul 3 de desenare a unui pătrat cu latura de 50 de pași. Putem observa faptul că lista (șirul) de comenzi `INAINTE 50 STINGA 90` a fost *repetată de patru ori*, deoarece pătratul are 4 laturi și 4 unghiuri egale. Deasemenea pentru a desena o "stea" formată din 3 raze de câte 40 de pași (problema 9) secvența de comenzi `INAINTE 40 INAPOI 40 STINGA 120` a fost *repetată de 3 ori.*

Situațiile în care trebuie să repetăm o listă de comenzi sunt foarte frecvente. Evident, este destul de incomod să înșirăm de multe ori același grup de comenzi (închipuiți-vă, de exemplu, ce s-ar întâmpla dacă am vrea să desenăm o stea cu ... 30 de raze!). Pentru a evita astfel de situații, limbajul **LOGO** conține o comandă de *repetare* a unui grup (secvență) de comenzi. Deoarece repetarea se mai numește și "ciclare", această instrucțiune se mai numește *instrucțiune de ciclare.*

În mod natural, instrucțiunea de ciclare trebuie să aibe două intrări: una care să arate *de câte ori* să se execute, iar alta care să arate *ce* să se execute; prima intrare va fi, deci, un număr ("N"), iar a doua intrare va fi o listă de comenzi.

Instrucțiunea de ciclare prezintă numai forma completă:


```
REPEAT  N [     ]
```

sau în limba română:

```
REPETA  N [     ]
```

adică "repetă de *n* ori comenzile din lista dată!". Lista de comenzi trebuie cuprinsă obligatoriu între paranteze drepte. Cu ajutorul comenzii de repetare, trasarea pătratului de la problema 3 se va putea face mult mai simplu cu linia **LOGO**:

```
REPETA  4 [INAINTE 50 STINGA 90]
```

Activități practice

✍ 14. Să se rescrie comenzile **LOGO** pentru desenarea unui triunghi cu laturile egale (problema 6), unei stele cu raze (problema 9), a unui pentagon (problema 11), a unui arc cu resort (problemele 12 și 13), a unui hexagon, a unui octogon, decagon, folosind instrucțiunea de repetare. Indicați formula pentru desenarea unei figuri geometrice regulate cu latura de 30 și N laturi.

✍ 15. Dacă numărul de laturi al unei figuri geometrice regulate crește, cu ce va începe să semene figura ? Când va fi mai mare această figură, când crește numărul de laturi (N) sau când latura este mai mare? Experimentați.

✍ 16. Desenați o stea cu 5 raze, fiecare rază având 20 de pași (fig. 19). Indicați o formulă pentru desenarea unei stele cu orice număr de raze.

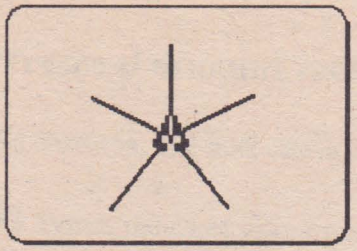


Fig. 19 Stea cu 5 raze

✍ 17. Folosind instrucțiunea de repetare realizați următoarele desene din fig. 20.

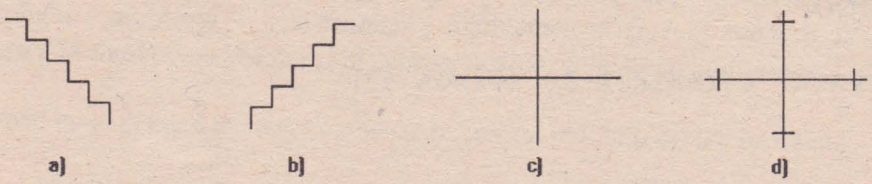


Fig. 20

✍ 18. Folosind instrucțiunea de repetare realizați desenul din fig. 21.

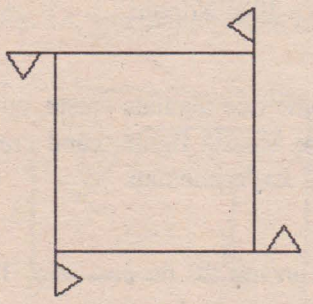


Fig. 21

PROCEDURI

Lecția 6

Proceduri

Să considerăm cazul unor desene simple realizate de noi: pătratul, triunghiul, cercul, steaua, resortul etc. După ce desenează oricare din aceste figuri, calculatorul (sau "broasca țestoasă") "uită" ceea ce tocmai a executat. *Ori de câte ori vrem să mai desenăm figura respectivă, trebuie să scriem din nou toate comenzile necesare.* Există însă și o posibilitate prin care broasca să *"țină minte"* cum se desenează un pătrat, triunghi etc. și să execute desenul respectiv la simpla comandă **PATRAT**, **TRIUNGHI** etc. Limbajul **LOGO** oferă posibilitatea de a completa bagajul de cunoștințe al broaștei țestoase, permițându-ne să o *învățăm înțelesul unor cuvinte (comenzi) noi.*

Să luăm de exemplu, cazul unui pătrat cu latura de 40. Pentru ca, la comanda **PATRAT**, broasca să deseneze imediat pătratul respectiv, trebuie să "știe" ce înseamnă a FACE un **PATRAT**. Altfel spus, ea trebuie să aibă în memorie înșiruirea (secvența) respectivă de comenzi; această înșiruire îi arată cum se *procedează* când primește comanda cu numele respectiv.

> Se numește *procedură* orice secvență de instrucțiuni memorate de calculator sub un *nume*, în vederea executării ei la întâlnirea numelui respectiv

Introducerea procedurii în memorie se mai numește definirea ei; definirea procedurii are loc o singură dată, pe când executarea ei are loc ori de câte ori se cere aceasta, prin indicarea numelui procedurii.
Să definim împreună procedura **PATRAT** amintită în prealabil. În acest scop vom folosi particula **TO**; în limba engleză ea are rolul de a forma infinitivele verbelor (similar cu **a** din limba română: **a** fi, **a** face etc). Linia de titlu a oricărei proceduri începe cu această particulă tocmai pentru a arăta că textul care urmează este o procedură. Traducerea titlului ar fi, deci, "A (face un) PATRAT".

Să observăm faptul că după ce am introdus linia **LOGO**

```
TO PATRAT
```

și am acționat tasta CR, promptul "**?**" cunoscut de noi nu își mai face apariția pe ecran, fiind înlocuit cu alt semn (">") tocmai pentru a arăta faptul că acum comenzile **LOGO** *nu se mai execută imediat ci se memorează, fiind executate ulterior la cererea noastră.* Introducem apoi comenzile necesare desenării pătratului:

```
REPETA  4 [IN 40 SA 90]
```

Comenzile formează, aici, o singură linie **LOGO**, dar, în cazul altor proceduri, vom avea mai multe linii care formează textul procedurii.

Terminarea procedurii trebuie să fie marcată printr-o linie **LOGO** specială, care conține doar cuvântul **END** (sfârșit). Textul de pe ecran este acum:

```
? TO PATRAT
>REPETA  4 [IN 40 SA 90]
> END
?
```

Convenim ca, pentru evidențierea corpului procedurii (linia sau liniile **LOGO** care se găsesc între particulele **TO** și **END**) și deoarece putem adăuga oricâte spații fără ca acest lucru să aibă repercusiuni în funcționarea procedurii, să scriem astfel:

```
? TO PATRAT
>    REPETA 4 [IN 40 SA 90]
> END
?
```

Observăm că după ce am introdus și linia **END** *procedura a fost definită, adică a fost "depusă" în memorie*, revenind, din nou, pe ecran promptul **LOGO** ceea ce arată că se așteaptă introducerea unor alte comenzi sau că se poate începe definirea altor proceduri.

Cum verificăm faptul că procedura noastră este bună?

Simplu! Dând comanda de executare a ei, comandă identică tocmai cu numele procedurii:

```
? PATRAT
```

Ecranul ne arată că procedura este corectă! Observați, deci, că procedurile sunt, de fapt, programe (ele fiind memorate) care se execută prin indicarea numelor lor.

În **LOGO** în loc de exprimarea "execuția unui program" se recomandă exprimarea "apelarea unei proceduri".

Reținem deasemenea că orice procedură trebuie să aibă structura din fig. 22.

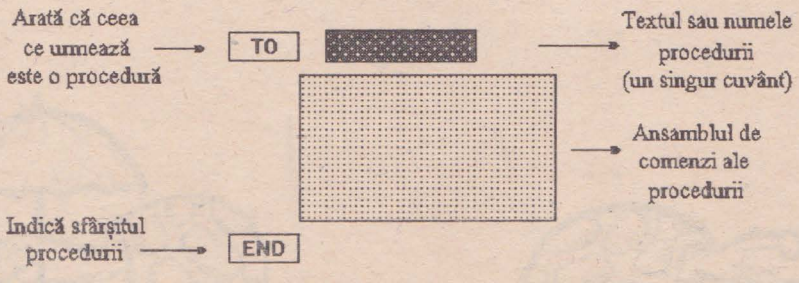


Fig. 22

Activități practice

19. Pe baza desenelor obținute la problemele precedente realizați procedurile: **PATRAT**, **PATRATD** (adică construit pe dreapta), **TRIUNGHI**, **STEA**, **CERC**, **PENTAGON**, **HEXAGON**, **OCTOGON**, **ARC**.

20. Folosind procedura **PATRAT** desenați două pătrate pe ecran.

21. Folosind procedura **PATRAT** desenați un pătrat rotit la stânga cu 45 și altul rotit spre dreapta cu 30 de grade (fig. 23).

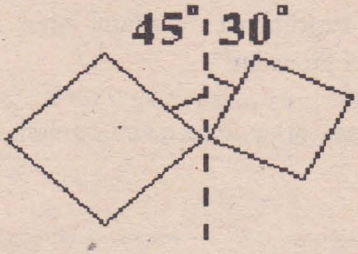


Fig. 23

Lecția 7

Modificarea procedurilor

Când scriem textul unei proceduri putem comite, bineînțeles, diferite erori. În cazul în care terminăm definirea procedurii și introducem **END** atunci la apelarea procedurii ori nu obținem ce am dorit ori "broasca" ne va afișa un mesaj de eroare.

Exemplul 1: să presupunem că la problema 19 am definit procedura astfel:

```
TO TRIUNGHI
   REPETA  3 [IN 40 SA 110]
END
```

Când *vom apela procedura*, aceasta "*va merge*" dar "*va merge prost*" deoarece observăm că nu obținem pe ecran ce am dorit, adică, un triunghi. Spunem că aceasta este o greșeală de "*logică*" deoarece *nu am calculat bine unghiul de rotire la stânga*. În acest caz *va fi necesar să modificăm procedura.*
Exemplul 2:

```
TO TRIUNGHI
   REPEATA  3 [IN 40 SA 120]
END
```

Când vom apela procedura, aceasta "nu va merge" iar "broasca" va afișa mesajul "Nu știu cum să REPEATA". Într-adevăr REPEATA nu este un cuvânt nici în engleză și nici în română, pe care să îl recunoască broasca.

Observați că, deși este o mică greșeală, broasca este nemiloasă. Spunem că aceasta este o greșeală de *sintaxă*. Evident că și în acest caz procedura va trebui modificată.

Modificarea sau modificările de proceduri se realizează prin intermediul *Editorului* **LOGO**. Apelarea Editorului în vederea modificării unei proceduri anterior definite se realizează cu comanda **EDIT** sau prescurtat, **ED**, după care se introduce numele procedurii.

Deoarece procedura există în memoria calculatorului, textul ei va fi tipărit pe ecran în întregime, imediat după comanda noastră de apelare. Să presupunem exemplul 1:

```
? EDIT TRIUNGHI                    → introdus de noi
TO TRIUNGHI
   REPETA  3 [IN 40 SA 110]        } afișat de editor
END
```

Se observă că editorul afișează procedura în mod standard, adică cu un spațiu după cifra 3 chiar dacă noi nu l-am introdus inițial. Vom observa, deasemenea, că editorul pune automat spații înainte și după semnele pentru operații.

Având în față textul procedurii putem face asupra sa orice modificare dorim, în felul următor:

- *coborâm cursorul* de pe prima linie a procedurii (`TO TRIUNGHI`) cu tastele **CS+6** (săgeată în jos) până când ajungem pe linia unde dorim să facem modificări. În mod similar putem să urcăm cursorul pe linii superioare cu tastele **CS+7** (săgeată în sus)

- fiind pe linia pe care se află o eroare și dorim să o îndepărtăm putem să *deplasăm cursorul la dreapta*, cu tastele **CS+8** (săgeată la dreapta) sau cu tastele **CS+5** (săgeată la stânga)

- după ce ne-am deplasat convenabil cu cursorul putem oricând să *intercalăm* (*inserăm*) litere, cifre, semne etc. sau să *ștergem* ceva și apoi să înlocuim fără a mai face greșeli. Ștergerea se realizează cu **CS+0** (**DELETE**) cu care se îndepărtează caracterul.

În cazul nostru vom deplasa cursorul o dată în jos (deci o dată CS+6) iar apoi de 21 de ori la dreapta (deci de 21 ori CS+8), poziționându-ne pe cifra 0. Vom șterge apoi cifra 1 dinaintea lui 0 cu CS+0 și, vom înscrie în locul lui 1 cifra 2. Realizând o rotire de 120 de grade la fiecare unghi, procedura este practic modificată, și urmează să ieșim din Editorul **LOGO**.

Considerând acum că procedura este bună, *manevra de ieşire prin care Editorul depune în memoria calculatorului procedura modificată* are două etape:

- se trece tastatura în regim extins (E) - cu tastele CS şi SS (cursorul E îl vom putea observa în colţul din dreapta jos al ecranului)
- se acţionează tasta C

Efectuând ieşirea din Editor, observăm că ecranul se "curăţă" şi apoi apare mesajul:

`TRIUNGHI` defined

Pentru cazul b) după chemarea Editorului ne deplasăm o dată în jos (CS+6) şi apoi de 5 ori dreapta (CS+8) urmând să ştergem caracterul din stânga cursorului (A). După această manevră, comanda fiind scrisă corect (`REPETA`), nu mai avem altceva de făcut decât să ieşim din Editor prin metoda cunoscută şi să folosim procedura.

Reamintim că fiecare linie **LOGO** scrisă se termină cu apăsarea tastei CR; deşi la apăsarea acestei taste nu apare nimic pe ecran, în memoria calculatorului se introduce în locul respectiv un simbol de "sfârşit de linie". Acest simbol trebuie tastat ca orice alt caracter; el poate fi şters sau introdus după voia noastră.

Acest lucru a fost menţionat, deoarece, de multe ori, când se fac modificări în proceduri, se comit greşeli legate de tasta CR; se apasă această tastă (în loc de SPACE) când cursorul se află la mijlocul unui rând sau se apasă CR când cursorul se află chiar la începutul unui rând. În primul caz, rândul existent se "rupe" în două, iar în al doilea caz apare un rând "gol".

Repararea acestor rânduri este simplă: trebuie şters "sfârşitul de linie" (invizibil) care a fost introdus unde nu trebuia. Pentru aceasta se duce cursorul pe primul caracter al liniei următoare şi se face manevra de ştergere de caracter (CS+0). În faţa cursorului aflându-se tocmai "sfîrşitul de linie" care dădea bătăi de cap, acesta este înlăturat şi situaţia revine la normal.

Trebuie remarcat şi faptul că *se pot realiza noi proceduri chiar din Editorul* **LOGO**, acest mod oferind şi posibilitatea modificării imediate a unor linii din procedură atunci când este cazul. Şi în acest mod, până când procedura este definitiv "pusă la punct", va trebui să se parcurgă ciclul execuţie, verificare, eventuale modificări, iar întreg acest parcurs se numeşte, de obicei, *testarea procedurii*.

Activități practice

22.

a) Modificați procedurile **PATRAT** și **PATRATD** astfel încât ele să deseneze pătrate cu latura de 30. Testați noile proceduri.

b) Definiți procedura **DREPT** pentru desenarea unui dreptunghi de 30 x 50. Testați funcționarea ei.

c) Modificați procedura **STEA** astfel încât să obțineți o nouă procedură **STEA** pentru desenarea unei stele cu 20 de raze a câte 15 pași fiecare. Testați procedura.

d) Realizați procedura **ARC** pentru desenarea resortului din fig. 18. Testați procedura.

Lecția 8

Supraproceduri și subproceduri

Numele unei proceduri definite de noi devine, după cum am văzut, o *comandă* nouă. Această comandă poate fi utilizată acum exact în același mod în care utilizăm comenzile **LOGO** "obișnuite", pe care broasca le știe "de la început". De altfel, comenzile **LOGO** se numesc tot "proceduri" însă, deoarece sunt cunoscute de broască "din primul moment", mai sunt numite *proceduri primitive* sau, mai pe scurt, *primitive*. Exemple de primitive am întâlnit destule până acum: **INAINTE**, **DREAPTA**, **STINGA**, **REPETA**, etc.

Revenind la *procedurile definite de utilizatori*, deci cele noi, vom remarca faptul că, putând fi utilizate ca toate comenzile **LOGO**, *ele pot figura și în textele unor alte proceduri definite de utilizatori*. De exemplu, așa cum procedura **PATRAT** conține comenzile **IN**, **REPETA** etc., tot așa și alte proceduri definite de noi vor putea conține, dacă este necesar, comanda **PATRAT** .

Spunem că o astfel de procedură *apelează* (cheamă) procedura **PATRAT**. Ea este o *supraprocedură* pentru **PATRAT** iar **PATRAT** este o *subprocedură* pentru ea. Evident, orice procedură poate fi o subprocedură pentru unele proceduri și o supraprocedură pentru altele.

Activități practice

23. Definiți o procedură care să deseneze un steag cu pînza pătrată, având partea bățului aflată sub pînză de două ori mai mare ca latura pătratului (fig. 24).
Atenție! Acum pătratul are latura de 30 deoarece procedurile **PATRAT** și **PATRATD** au fost modificate cu Editorul.

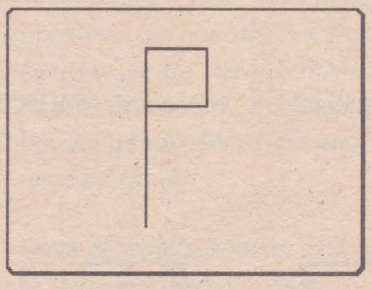


Fig. 24

24. Definiți și testați o procedură numită POM care să deseneze un pom cu tulpina de 50, având drept "coroană" steaua desenată de procedura STEA (fig. 25).

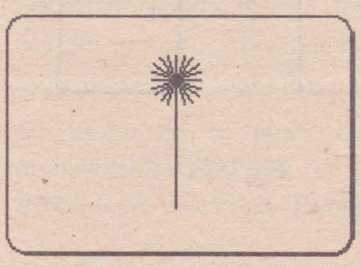


Fig. 25

✍ 25. Definiți și testați o procedură MORISCA care să deseneze opt steaguri ce pornesc din același punct, fiecare "băț", formând cu următorul un unghi de 45 de grade (fig. 26). Analog, o procedură POMI pentru șase pomi ce pornesc din același punct și ale căror tulpini formează unghiuri de 60 de grade (fig. 27).

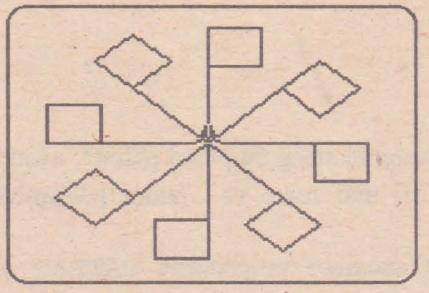


Fig. 26

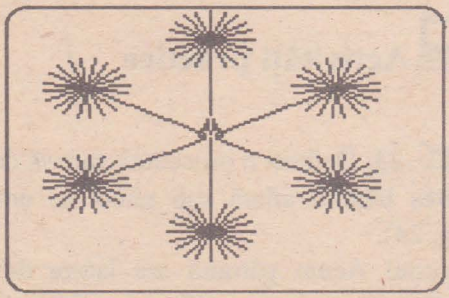


Fig. 27

✍ 26. Folosind o procedură PATRAT realizați o procedură CRUCE (fig. 28a) și o procedură MOARA (fig. 28b). Comparați cele două proceduri obținute. Ce constatați?

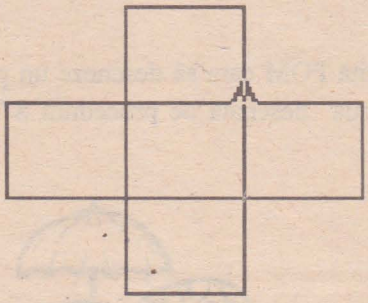


Fig. 28a

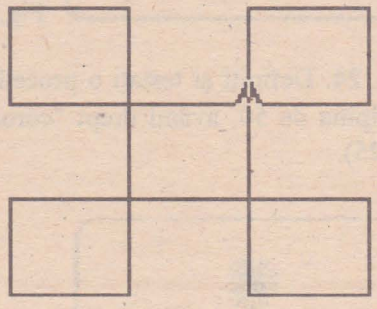


Fig. 28b

Lecția 9

Recomandări privind lucrul cu procedurile

Exercițiile precedente ne-au convins de faptul că definirea și utilizarea procedurilor este deosebit de fructuoasă, permițându-ne realizarea de desene complicate cu un efort minim din partea noastră. Este necesar, însă, să ținem seama de câteva principii care ne vor simplifica lucrul cu procedurile și ne vor ajuta să "stăpânim" cu ușurință proceduri din ce în ce mai complicate.

În primul rând vom porni de la ideea că o procedură rezolvă o problemă (la început de "grafică", apoi de geometrie și de alte tipuri). Prima cerință este deci *formularea clară a problemei respective* de către noi.

Procedura pe care o vom defini "instruiește" broasca țestoasă *cum să rezolve problema formulată.* Va trebui, deci, *să ne clarificăm mai întâi noi modul în care se poate rezolva problema, să găsim calea de rezolvare a problemei. Pentru aceasta, de foarte multe ori este bine să executăm o schiță cu creionul pe hârtie, gândindu-ne la toate acțiunile pe care va trebui să le execute broasca, rând pe rând.*

Dacă desenul respectiv este prea complicat, va trebui să-l analizăm cu atenție și să descoperim cum poate fi "*descompus*" în părți separate sau *să descoperim părți care se repetă* în mai multe locuri ale desenului. *Este preferabil să definim proceduri mai simple pentru aceste părți ale desenului, folosindu-le apoi ca subproceduri în procedura finală.*

De mare importanță este descompunerea "repetițiilor" dintr-un desen, deoarece aceasta permite utilizarea instrucțiunilor de ciclare **REPETA**, simplificându-se mult scrierea procedurilor.

De multe ori se poate întâmpla să nu mai ținem minte numele tuturor procedurilor definite de noi. *Afișarea titlurilor procedurilor* se face la comanda: **POTS** (care înseamnă "Print Out Titles", adică "tipărește titlurile!").

Dacă dorim să se afișeze nu numai titlurile (numele) procedurilor ci și *conținutul* lor atunci se poate da comanda **POPS** ("Print Out Procedures", adică "tipărește procedurile!").

Evident, înainte de a da aceste comenzi, este bine să vedem ecranul în *regim textual*, cu comanda **TEXTSCREEN** (**TS**).

O procedură devenită inutilă poate fi "*ștearsă*" din memorie cu comanda **ERASE** sau prescurtat **ER** (șterge procedura cu numele dat). În limba română vom folosi **UITA**. Exemplu:

```
ERASE "POM
```

va șterge din memoria calculatorului procedura POM. Evident această comandă trebuie să fie folosită cu mare atenție numai când este strict necesar.

Mai există și o comandă cu ajutorul căreia se poate *șterge tot* ce s-a creat și s-a depus în memoria calculatorului, deci și toate procedurile care au fost definite. Această comandă este **ERALL** ("Erase All", adică "uită totul!").

Atenție! folosirea acestei comenzi va avea ca efect și ștergerea din memorie a tuturor comenzilor în limba română.

Activități practice

27. Definiți și testați o procedură numită TRI pentru desenarea unui triunghi echilateral cu o latură orizontală (fig. 29).

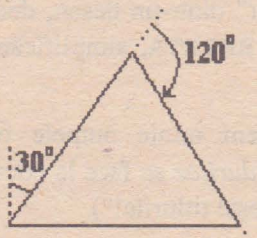


Fig. 29

✍ 28.Definiți și testați o procedură TRAP pentru desenarea unui trapez ca acela din fig. 30.

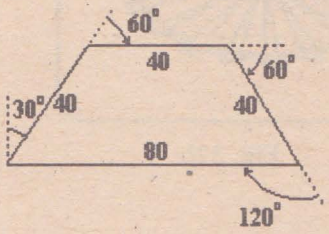


Fig. 30

✍ 29. Definiți și testați o procedură TRIDREP pentru desenarea unui triunghi ca acela din fig. 31.

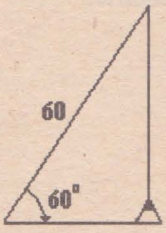


Fig. 31

✍ 30. Definiți și testați o procedură care să deseneze 20 de dreptunghiuri cu un vârf comun, rotite, fiecare față de cel precedent cu același unghi. Folosiți ca subprocedură o procedură DREPT care realizează un dreptunghi (fig. 32a). Aceeași problemă pentru triunghi (fig. 32b).

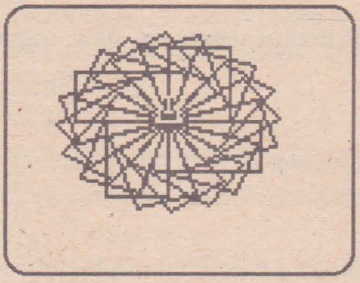


Fig. 32a

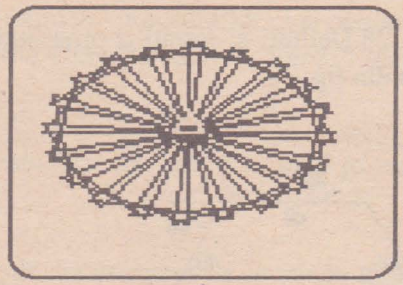


Fig. 32b

✍ 31. Construiți două proceduri prin care să se deseneze două steguleţe. Primul numit STEAGS va avea pînza orientată spre stînga iar al doilea STEAGD va avea pînza orientată spre dreapta. Cu aceste două proceduri încercați să realizați desenul din figura 33 prin care cele două steaguri sunt simetrice față de o axă.
Citiți apoi cu atenție cele două șiruri de comenzi care realizează cele două steguleţe. Ce observați? Încercați să enunțați o regulă pentru simetrie.

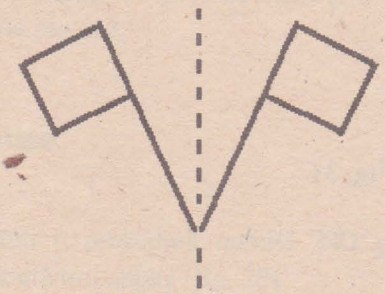


Fig. 33

32. Afișați numele tuturor procedurilor create. Ștergeți procedura cu numele DREPT. Dați comanda de afișare a desenului cu mai multe dreptunghiuri (MULTDREPT). Ce s-a întâmplat? Redefiniți procedura DREPT. Desenați mai multe dreptunghiuri (MULTDREPT).

VARIABILE

Lecția 10

Modificarea dimensiunilor

Am văzut cum se definește o procedură **PATRAT** cu latura de o anumită mărime (40 de pași, de exemplu). Ce facem dacă trebuie să desenăm un pătrat cu latura diferită de 40, cât prevede procedura noastră, **PATRAT**?

Evident, *vom modifica procedura existentă*, punând în textul ei, în loc de **IN 40**, deplasarea **IN 30**, **IN 50**, sau cât dorim. Procedura astfel modificată o putem "salva" sub alt nume, să zicem **PATRAT2**. Dacă dorim mai multe pătrate, de diferite dimensiuni, ar trebui, în acest fel, să îngrămădim în memorie mai multe proceduri, cu diferite nume.

Acest mod de lucru nu este convenabil nici pentru calculator nici pentru noi, deoarece ocupă mult din memoria calculatorului și ne obligă să ținem minte ce face fiecare din procedurile respective.

O altă cale este aceea de a nu schimba numele procedurii, rămânând ca ea să deseneze pătratul cu latura introdusă de noi la ultima utilizare. Evident, nici acest procedeu de schimbare a dimensiunilor figurilor desenate nu este mulțumitor; el ne obligă să apelăm de fiecare dată editorul și, prin aceasta să ștergem de pe ecran figurile desenate anterior. Nu vom putea desena în acest mod niciodată două pătrate cu laturi diferite, care să fie în același timp pe ecran.

Numele procedurii fiind o comandă, *ar trebui ca această comandă să fie astfel definită de noi încât să aibă nevoie de "intrări" adică de valori care să-i dirijeze acțiunea.*

Așa cum dăm comanda **INAINTE 10** sau **INAINTE 70** am vrea să putem da comanda **PATRAT 10** sau **PATRAT 70** și să se realizeze pătratul cu latura respectivă. Sau, în cazul desenării de dreptunghiuri, am vrea să putem da comenzi de genul **DREPT 70 20**, respectiv **DREPT 40 10**.

Dacă dăm, însă, acum comanda **PATRAT 10**, calculatorul execută procedura așa cum e "scrisă" în memorie, adică desenează un pătrat cu latura de 40; după aceea, întâlnind pe aceeași linie **LOGO** valoarea 10, nu știe ce să mai facă și ne dă mesajul:

"Nu ai spus ce să fac cu 10"

(Dați comanda pentru a verifica acest lucru).

Într-adevăr, *calculatorul ar fi trebuit ca, începând să execute procedura, să știe că după numele procedurii, comanda de execuție va mai conține un număr care trebuie "memorat"* iar, apoi, ar fi trebuit ca procedura, în loc să-i ceară (pentru fiecare latură) "mergi înainte cu 40", să-i comande, pentru fiecare latură, "mergi înainte cu cât ai memorat".

Iată, deci, că problema constă în a *modifica procedurile* în așa fel încât ele *să poată utiliza memoria calculatorului pentru a depune aici anumite valori*, pe care apoi să le poată regăsi și folosi.

Aducându-ne aminte că înseși procedurile sunt depuse în zone ale memoriei recunoscute prin numele procedurilor respective, ne dăm seama că și pentru memorarea diferitelor valori poate fi folosit același procedeu.

O "căsuță" (sau "locație") de memorie care a primit un nume dat de un programator se numește variabilă. Această denumire se datorează faptului că locația respectivă poate conține, rând pe rând, diferite valori care se utilizează în același scop. Practic într-o locație de memorie se poate "*pune*" ceva: *un număr sau un cuvânt.* Ca și în cazul procedurilor, programatorul nu trebuie să știe unde se află în memorie variabilele folosite de el; aceasta este treaba calculatorului!

Observăm însă că și variabilele și procedurile au "nume" după care sunt recunoscute; cum să știe calculatorul când ne referim la o procedură și când ne referim la o variabilă?

Simplu! Limbajul **LOGO** ne obligă ca, *atunci când ne referim la conținutul unei variabile să punem semnul* **:** *(două puncte) înaintea numelui variabilei respective.*

Să revenim la procedura **PATRAT** și să afișăm cu editorul textul ei:

```
TO PATRAT
   REPETA 4 [IN 40 SA 90]
END
```

Procedura trebuie modificată astfel încât să știe că, în momentul apelării (punerii în execuție), va primi o valoare pe care trebuie să o ia "în primire" și s-o depună într-o variabilă cu numele dat de noi. În cazul nostru, vom numi cu **LAT** (de la "latura") variabila respectivă. Pentru ca "rezervarea" căsuței (locației) corespunzătoare să se facă de la început, limbajul **LOGO** cere ca variabila care va primi valori prin comanda de apelare să fie introdusă chiar în titlul procedurii după numele acesteia. *Este bine ca numele dat variabilei (locației) să sugereze conținutul ei.*

În cazul nostru vom completa linia de titlu astfel:

```
TO PATRAT :LAT
```

După ce, prin această modificare, am cerut să se rezerve o locație pentru variabila **LAT**, urmează să arătăm cum va folosi procedura conținutul acestei variabile. Evident, procedura trebuie să comande broaștei "înaintează cu **:LAT**" în loc de "înaintează cu 30", pentru fiecare latură a pătratului. Deci întreaga procedură pentru desenarea unui pătrat cu latura variabilă arată astfel:

```
TO PATRAT :LAT
   REPETA 4 [IN :LAT SA 90]
END
```

Acum putem să desenăm pe rând pătrate cu laturi de 10, 20, 70 etc. prin comenzile **PATRAT 10** , **PATRAT 20** , **PATRAT 7** etc.

Activități practice

33. Modificați procedurile **TRIUNGHI** , **STEA** , **CERC** , **PENTAGON**, **HEXAGON** astfel încât să se realizeze figuri geometrice cu laturi variabile. Realizați apoi pe ecran câteva figuri geometrice dând valorile laturilor. Ce se întâmplă dacă introduceți și valori negative?

✍ 34. Modificaţi procedura pentru desenarea dreptunghiului (**DREPT**) astfel încât să se poată realiza desenarea oricărui dreptunghi.

✍ 35. Scrieţi şi testaţi o procedură numită **MUTA**, care să mute broasca pe orizontală la dreapta cu X paşi şi pe verticală în sus cu Y paşi, dar fără a trasa drumul parcurs.

Lecţia 11

Variabile locale şi variabile globale

Evident, de acum înainte nu vom mai scrie proceduri pentru dimensiuni "fixe" ale figurilor, ci ne vom gândi de la început să stabilim care sunt mărimile ce determină dimensiunile figurilor dorite. Vom stabili, apoi, numele variabilelor respective şi vom scrie procedurile folosind aceste variabile. Procedura, astfel şcrisă, va desena figura dorită la orice "scară", oricât de mare dorim.

Variabilele care apar ca parametri de intrare într-o procedură sunt create de procedură în momentul apelării şi folosite în timpul executării procedurii respective. *La terminarea procedurii, aceste variabile sunt "distruse", deci nu mai putem utiliza conţinutul lor.* Din acest motiv, varibilele care sunt declarate ca proceduri de intrare în proceduri se mai numesc şi *variabile locale.*

Deoarece variabilele locale sunt create de fiecare procedură, se poate utiliza acelaşi nume de variabilă locală în diferite proceduri fără ca prin aceasta să se creeze confuzii. De exemplu, putem folosi numele LAT atât pentru latura pătratului, cât şi pentru latura triunghiului, a hexagonului, ca parametru de intrare declarat în titlul acestor proceduri realizate în cadrul problemei 33. Fiecare din aceste proceduri va rezolva şi va utiliza variabila sa locală cu numele respectiv.

Totuși sunt situații când am vrea ca *o aceeași variabilă să poată fi utilizată de mai multe proceduri independente sau chiar în unele comenzi* **LOGO** *din afara procedurilor.* O astfel de variabilă ar fi , deci, *globală.*

Varibilele globale pot fi create în proceduri sau în afara lor cu ajutorul instrucțiunii **MAKE** ("fă-l pe ...").

De exemplu:

```
MAKE "A 10
```

(echivalent cu LET A=10 în BASIC)

sau

```
MAKE "L 30
```

sau

```
MAKE "LAT :L + 20
```

sau

```
MAKE "NUME "CRISTIAN
```

Instrucțiunea **MAKE** creează variabila cu numele respectiv și depune la locația corespunzătoare din memorie ceea ce urmează după numele variabilei.

Memoria calculatorului este formată din mai multe locații (celule sau sertare) de memorie. Acestea pot purta un *nume* (*etichetă a sertarului*) și în ele se pot pune diferite *obiecte.* De aceea în limba română instrucțiunea echivalentă este **PUNE** (vezi fig. 34).

Ce se poate pune într-o variabilă?
Desigur *numere* (exemplul 1 și 2 de mai sus). Deasemenea se poate pune *conținutul unei variabile create anterior* (exemplul 3) sau mai multe numere și variabile legate între ele prin semne de operații aritmetice (*expresii*). În sfârșit, se poate pune un *cuvânt* sau o *propoziție.*

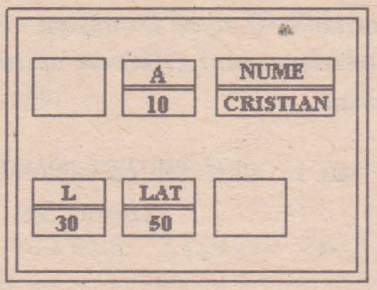


Fig. 34

Se observă că în **LOGO** *toate variabilele globale sunt la fel, nefăcându-se distincția între variabile numerice* (în care pot fi "puse" numere) și *variabile de tip șir de caractere* (în care pot fi "puse" cuvinte, șiruri de caractere).

Bineînțeles putem oricând să ne "*uităm*" în sertărașele memoriei afișând conținutul lor cu comanda **PRINT**. De exemplu, considerând că memoria arată ca în figura 34, dacă dăm comanda:

```
PRINT :LAT
```

se va afișa 50 deoarece în locația numită LAT se găsește numărul 50.

Activități practice

36. Puneți într-o variabilă A valoarea 3 iar în altă variabilă B valoarea 4. Scrieți o procedură SCHIMBA astfel încât conținutul lui A să treacă în B iar al lui B în A.

37. Realizați o procedură astfel încât calculatorul să "numere" până la un număr specificat. Modificați procedura astfel încât calculatorul să numere din 2 în 2. Putem spune că acum această procedură afișează primele N numere pare. Modificați procedura astfel încât calculatorul să afișeze primele N numere impare.

38. Realizați o procedură prin care calculatorul să "numere" cu un pas variabil. Putem spune că această procedură afișează multipli numărului "pas" până la un anumit număr, deci putem să o numim MULTIPLI.

✍ 39. Realizați o procedură SUMA pentru calcularea sumei primelor N numere naturale.

Indicație. Folosiți o variabilă pentru sumă (de exemplu S) pe care o inițializați cu 0 deoarece, la început, în ea nu este nimic. Apoi "puneți" în ea de fiecare dată următorul număr.

Pentru a "pune" în variabila S numărul următor este nevoie de o variabila C (contor) în care se va păstra tot timpul numărul curent care se obține adăugându-se o unitate la vechiul număr.

Care este suma primelor 10 numere naturale? Dar a primelor 100? Care este regula de formare a numărului care exprimă suma?

Verificați formula:

S = N (N+1) / 2

✍ 40. Modificați procedura SUMA pentru calcularea sumei primelor N numere pare. Care este suma primelor 10 numere pare? Dar a primelor 100? Care este regula de formare a numărului care exprimă suma respectivă? Comparați sumele numerelor pare astfel obținute cu sumele numerelor naturale corespunzătoare. Ce puteți spune? Puteți explica rezultatul?

Realizați același lucru și pentru suma numerelor impare. Ce puteți spune comparând sumele numerelor impare cu sumele numerelor naturale corespunzătoare?

✍ 41. Să se realizeze o procedură PRODUS cu care să se efectueze produsul primelor N numere naturale.

Lecția 12

Funcții (operații)

Până acum am numit "comenzi" toate instrucțiunile **LOGO.** Este momentul a se face o distincție între două tipuri de instrucțiuni:

- se numesc *operații* sau *funcții instrucțiunile a căror executare conduce la obținerea unei "valori"*
- se numesc *comenzi* instrucțiunile care nu sunt operații (funcții)

Primele operații pe care le-am întâlnit au fost chiar operațiile aritmetice (+ – * /). În urma executării fiecăreia din acestea se obține o valoare numerică. Celelalte funcții (operații) **LOGO** nu se notează cu simboluri ci cu nume, ca orice primitivă. Chiar și pentru adunare, înmulțire și împărțire există în afară de semnele care le exprimă și nume de primitive cu care se pot executa operațiile respective. Aceste nume sunt **SUM**, **PRODUCT** (sau **PROD**) și **DIV**. Ele reprezintă primitive care necesită doi parametri.

De ce au fost necesare și aceste primitive? Deoarece în **LOGO** parantezele (rotunde) au și altă destinație decât calculul expresiilor algebrice.
De exemplu, pentru a calcula expresia (3 + 2) x 5, fără a apela la paranteze, vom introduce:

```
SCRIE PROD  2 + 3   5
sau
SCRIE PROD  SUM  2  3   5
```

Să observăm că rezultatul unei operații **LOGO** trebuie să fie utilizat întotdeauna ca intrare pentru o comandă sau pentru o altă operație; în caz contrar, calculatorul va afișa mesajul: "nu ai spus ce să fac cu ..." urmat de rezultatul cu care nu știe ce să facă.

În exemplul dat rezultatul obținut (25) a fost utilizat ca intrare pentru **SCRIE** și, în consecință, s-a afișat numărul 25.
În afară de operațiile amintite în limbajul **LOGO** se pot utiliza și alte funcții aritmetice ca:

```
INT A
```

furnizează partea întreagă a numărului **A** înlăturând zecimalele acestuia. O definiție mai corectă a funcției **INTREG** este: *furnizează cel mai mare număr întreg mai mic decât numărul* **A** (vezi problema 44).

```
REMAINDER A B
```

furnizează restul împărțirii numărului **A** la numărul **B**. În limba română se poate folosi **REST**.

```
RANDOM N
```

furnizează un număr aleator (întâmplător) cuprins între **0** și **N-1** inclusiv.

```
SQRT A
```

furnizează radicalul din numărul **A**.

Activități practice

42. Afișați restul împărțirii lui 1989 la 17.

✍ 43. Vedeți dacă numărul 1999 este divizibil cu 53.

✍ 44. Afișați partea întreagă a rezultatului împărțirii lui 256 la 45; afișați partea întreagă a rezultatului împărțirii lui 175 la 45; afișați partea întreagă a rezultatului obținut prin efectuarea calculului 3,24 • 4,23 : 5,707; afișați partea întreagă a rezultatului împărțirii lui -9 la 2. Explicați ultimul rezultat.

✍ 45. Realizați o procedură-funcție, care, aplicată unui număr să producă drept rezultat scrierea numărului cu nu mai mult de două zecimale.

✍ 46. Realizați procedura SUMAPAT pentru calculul sumei pătratelor primelor N numere naturale. Care este suma pătratelor primelor 10 numere naturale? Dar a patratelor primelor 100 de numere? Același lucru pentru calculul sumei inverselor primelor N numere.

✍ 47. Utilizați procedurile de calcul a sumelor:

- primelor N numere naturale SUMA (problema 39)
- primelor N numere pare SUMAPAR (problema 40)
- primelor N numere impare SUMAIMP (problema 40)
- a pătretelor primelor N numere naturale SUMAPAT (problema 46)
- a cuburilor primelor N numere naturale SUMACUB (de realizat)
- a inverselor primelor N numere naturale SUMAINV (de realizat).

Completați tabelul de mai jos încercând să sintetizați regulile de obținere a sumelor respective precum și formulele de calcul.

N	SUMA	SUMAPAR	SUMAIMP	SUMAPAT	SUMACUB	SUMAINV
10	55	110	100	385	3025	2,928968
100						
1000						
10000						
.....						
formule						

✍ 48. Dați comanda de afișare a cinci numere aleatoare cuprinse între 0 și 100 inclusiv.

✍ 49. Realizați o procedură funcție care să afișeze un număr aleator cuprins între o margine inferioară A și una superioară B (numere întregi).

✍ 50. Scrieți procedurile necesare pentru "umplerea" ecranului cu pătrate având aceeași latură (L) și un spațiu de 5 pixeli între două pătrate alăturate.

Indicație: procedura principală va trebui să calculeze numărul de rânduri (NR) și numărul de pătrate de pe fiecare rând (NP).

ACȚIUNI CONDIȚIONATE ȘI RECURSIVITATEA

Lecția 13

Instrucțiunea IF

Deseori în realizarea procedurilor este necesar ca unele acțiuni să se execute numai în anumite condiții. Instrucțiunea care comandă *executarea condiționată* a unor acțiuni se numește **IF** (în limba română **DACA**). Această instrucțiune are următoarea formă generală:

IF condiție [lista1 de instrucțiuni] [lista2 de instrucțiuni]

La întâlnirea acestei instrucțiuni calculatorul verifică dacă la momentul respectiv *condiția* este îndeplinită; dacă *da*, el execută *lista1 de instrucțiuni*, iar în caz contrar (adică în cazul în care condiția este *falsă*) execută *lista2 de instrucțiuni*.

Evident, el nu execută niciodată ambele liste, ci numai una din ele.

Excluderea unor cazuri extreme

Uneori în realizarea procedurilor apare necesitatea de a se exclude cazurile în care, de exemplu, un desen devine necorespunzător, diform. De exemplu, dacă dorim să desenăm pătrate a căror latură crește cu un anumit pas atunci la un moment dat vom obține un pătrat care nu mai încape în ecran și, de fapt, nu s-ar mai desena un pătrat. Cum am putea evita acest lucru?

Ar trebui ca procedura respectivă să conțină o instrucțiune care să realizeze următorul lucru:

Dacă latura este mai mare decât 80 *oprește-te*!

Evident, va trebui să folosim o instrucțiune **IF (DACA)**.

Limbajul **LOGO** are și comanda de "*oprire*" a execuției unei proceduri înainte de epuizarea instrucțiunilor procedurii respective, iar această instrucțiune este **STOP** . Linia de instrucțiune prin care se va opri realizarea unor pătrate cu latura prea mare va fi:

```
IF  :LAT > 80 [STOP]
```

În consecință, *dacă* valoarea respectivă *este mai mică (sau egală) decât 80*, procedura *nu se oprește* trecând la liniile următoare prin care se desenează pătratul. *Dacă* însă valoarea primită pentru lungimea laturii pătratului este *mai mare decât 80,* procedura *se oprește*, adică nu mai desenează nimic și pe ecran apare promptul "?".

Se observă că instructiunea **IF** cu **STOP** reprezintă o formă "incompletă" a lui **IF**, acum această instrucțiune putând fi reprezentată astfel:

IF condiție [lista de instrucțiuni]

Această formă cere executarea listei de instrucțiuni dacă *este indeplinită condiția dată*; în caz contrar lista nu se execută și se trece la linia **LOGO** care urmează.

Dacă o procedură se termină prin obținerea unei valori, acea procedură se aseamănă cu operațiile (funcțiile) **LOGO**. Pentru ca ea să se comporte întocmai ca aceste funcții, este necesar ca terminarea ei să accentueze rolul de funcție pe care îl îndeplinește.

O procedură se transformă în "funcție" dacă se termină cu *instrucțiunea de "ieșire"*. Aceasta este:

OUTPUT expresie

sau prescurtat **OP** expresie, ceea ce se poate traduce prin "furnizează rezultatul expresiei".

Exemplu:

```
IF :LAT > 80  [OUTPUT "GATA STOP]
```

Activități practice

51. Folosind procedurile **PATRAT** și **TRIUNGHI** realizați o procedură **FIGURA** cu care se va realiza o figură cu laturi egale. Procedura va avea doi parametri. Primul parametru va reprezenta latura figurii, al doilea parametru va reprezenta tipul figurii. Dacă al doilea parametru este 1 atunci va desena un pătrat, dacă nu, un triunghi.

52. Realizați o procedură care, primind două intrări, să o depună pe cea mai mare în variabila globală **MAX**, iar pe cea mai mică în variabila globală **MIN**.

53. Realizați o procedură care să furnizeze valoarea absolută (modulul) unui număr dat.

54. Realizați o procedură cu doi parametri de intrare care să furnizeze la ieșire media aritmetică a celor două valori date ca intrări.

55. Realizați o procedură care să simuleze aruncarea unei monede.
Indicație: se va genera un număr aleator cuprins între 0 și 100. Dacă numărul va fi mai mic decit 50 se va afișa "stema", dacă va fi mai mare decât 50 se va afișa "banul".

56. Să se realizeze o procedură **DIVIZOR** cu doi parametri care reprezintă două numere. Se va testa dacă primul număr este un divizor al celui de-al doilea și, dacă este, se va afișa cuvântul "divizor".

57. Realizați o procedură pentru desenarea unui dreptunghi așezat orizontal, indiferent de ordinea intrărilor.

Lecția 14

Apelarea recursivă

Am văzut că procedurile se pot apela de către alte proceduri. De exemplu, procedura **FIGURA** de la problema 51 apelează procedurile **PATRAT** și **TRIUNGHI** realizate anterior. O altă procedură **CASA** cu care am putea desena o casă ar putea apela procedurile **PATRAT, DREPT, TRIUNGHI, CERC,** în construcția casei putând apare mai multe elemente de tip pătrat, dreptunghi, triunghi, etc., de dimensiuni diferite (bineînțeles vom folosi procedurile respective cu parametri de intrare).

Să presupunem că modificăm procedura **PATRAT** care să deseneze un pătrat cu latura variabilă astfel:

```
TO PATRAT :L
   REPETA 4[IN :L SA 90]
   PATRAT :L
END
```

Se observă că după linia principală a procedurii s-a mai intercalat o linie (**PATRAT :L**) în care se cere realizarea unui pătrat cu latura variabilă deși această procedură nu s-a definit încă sau, mai bine zis, deși această procedură este în curs de definire.

Să ne imaginăm ce se întâmplă când invocăm procedura care desenează un pătrat cu latura variabilă, de exemplu: **PATRAT 40.**

- În primul rând se va desena un pătrat cu latura de 40 conform liniei **REPETA 4[IN 40 SA 90]**
- Apoi va fi întâlnit cuvântul **PATRAT :L** și din nou va fi apelată procedura **PATRAT**, prin care se va desena același PATRAT, și așa mai departe

Într-adevăr dacă tastăm **PATRAT 40** vom observa cum se desenează la nesfârșit peste același desen, pătrate cu latura de 40. Putem opri (stopa) desenarea în continuare a pătratului acționând tastele SPACE și CAPS SHIFT (**BREAK**). Observăm că broasca rămâne într-un punct de pe conturul pătratului afișându-se mesajul "**STOP**" care semnifică oprirea unei proceduri în timpul acțiunii ei.

Procedura **PATRAT**, așa cum a fost ea definită este o procedură *recursivă* deoarece conținând în corpul său chiar numele procedurii, realizează *apelarea procedurii de către ea însăși*, adică *recursivitatea*.

În BASIC un program care realizează o apelare recursivă este, de exemplu, 10 GO SUB 10 (executați programul și observați ce se întâmplă!)

Totuși până în acest moment nu este prea clar la ce ne poate folosi apelarea recursivă. Aceasta devine interesantă în momentul în care se apelează aceeași procedură *modificându-se, însă, parametrul său.* Să modificăm definirea procedurii care desenează un pătrat cu latura variabilă astfel:

```
TO   PATRAT  :L
     REPETA  4[IN  :L  SA  90]
     PATRAT  :L + 2
END
```

Se observă că s-a invocat procedura **PATRAT** dar, de fiecare dată, se va desena un pătrat a cărui latură va fi mai mare cu 2 față de latura pătratului precedent. Într-adevăr, dacă invocăm **PATRAT 10**, se va atribui variabilei :L valoarea 10, se va desena un pătrat cu latura 10, iar apoi, ajungându-se la linia **PATRAT :L+2** se va atribui variabilei :L o valoare egală cu cât era înainte valoarea ei (adică 10) la care se va mai adăuga 2 (similar în BASIC cu LET L = L+2); se va invoca, deci, procedura **PATRAT 12**, desenându-se un pătrat cu latura 12. În mod asemănător se vor desena în continuare pătrate cu latura 14, 16, 18 etc., procedura continuând la nesfârșit chiar când mărimea laturii începe să depășească marginile ecranului. Se observă în acest caz că broasca se "*întoarce prin partea cealaltă*" a ecranului, modul acesta de lucru al ei numindu-se "*wrapping*" deoarece seamănă cu cel prin care se coase, vârful acului "întorcându-se prin partea cealaltă". Bineînțeles că și în acest caz putem opri desenarea de pătrate cu laturi din ce în ce mai mari prin **BREAK** (SPACE +CS).

Deoarece nu se oprește decât la intervenția noastră, procedura, deși recursivă, nu ne mulțumește pe deplin. Pentru a nu se produce o repetare fără sfârșit a acțiunilor procedurii, *apelarea recursivă trebuie să fie întotdeauna condiționată.*

În cazul procedurii noastre condiția se poate referi la mărimea laturii, și anume, când aceasta depășește o anumită valoare (de exemplu, 50) procedura se va opri.

Procedura va arăta astfel:

```
TO PATRAT  :L
   IF  :L > 50  [STOP]
   REPETA  4[IN  :L SA 90]
   PATRAT  :L + 2
END
```

Să recapitulăm deci acțiunile întreprinse de procedură:

- dacă latura devine mai mare decât 50 procedura se oprește
- se desenează un pătrat cu latura L
- se execută din nou toate instrucțiunile, începând cu primul punct, dar cu latura modificată, și anume mărită cu 2

S-a văzut că modificarea laturii se face de fiecare dată în așa fel încât broasca să pornească din același punct dar să deseneze un pătrat mai mare. Mărirea, de fiecare dată, a laturii face ca procesul reluării să aibe, în mod sigur, un sfârșit. Într-adevăr, deși, *nu se știe dinainte câte repetiții vor avea loc*, tot mărindu-se, "odată și odată" latura va trebui să devină mai mare decât 50. În acest moment, condiția din **IF** nu mai este îndeplinită și, deci, procedura nu se mai reia prin autoapelare. Procedura noastră, deși repetă de mai multe ori executarea acelorași instrucțiuni, așa cum am mai arătat, *nu "știe" de la început de câte ori trebuie să repete execuția*. Iată de ce desenul nostru nu a putut fi realizat prin utilizarea unei instrucțiuni **REPETA**.

Alt lucru care trebuie subliniat este acela că, pentru a fi mai eficientă, adică pentru a consuma mai puțină memorie, *apelarea recursivă trebuie pusă "la coadă"* (adică înainte de cuvântul **END** care termină orice procedură) așa cum s-a făcut și în procedura noastră.

Activități practice

58. Realizați o procedură recursivă cu care să se deseneze un model grafic decorativ pe bază de pătrate (procedura **PATRAT**) ca în figura 35 (adică cu pătrate de dimensiuni diferite, unul într-altul).

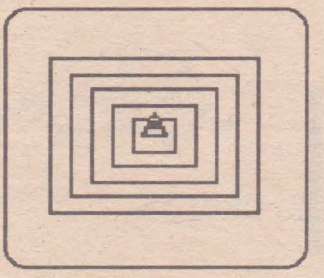


Fig. 35

✍ 59. Priviți figura 36. Ea reprezintă o spirală pe bază de hexagon care se dezvoltă spre exterior și este rezultatul apelării procedurii `SPI`:

```
TO SPI   :L
IF :L    > 60 [STOP]
   IN    :L   SA  60
   SPI   :L + 2
END
```

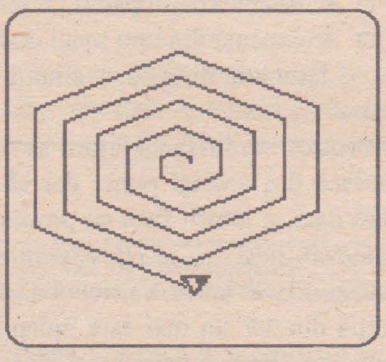


Fig. 36

Se observă că, dacă latura depășește 60, atunci procedura se oprește. Forma spiralei (hexagonică) este dată de unghiul de rotire (60 de grade). Deasemenea se observă că procedura este recursivă, la următorul apel latura crescând cu 2.

Modificați procedura SPI astfel încât să realizați:
- o spirală pe bază de pătrat (fig. 37)
- o spirală pe bază de triunghi (fig. 38)
- o stea (fig. 39)
- o stea cu unghiurile mai ascuțite (fig. 40)

În urma modificării unghiului de rotire va ajunge vreodată procedura pentru spirală să realizeze o simplă dreaptă?
Încercați să vă imaginați după fiecare modificare a unghiului de rotire cum vor arăta rezultatele pe ecran.
Faceți o analiză asupra rezultatelor obținute cu procedura SPI în funcție de mărimea unghiului de rotire.

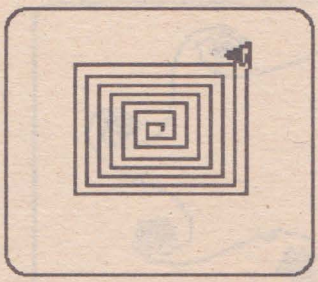


Fig. 37

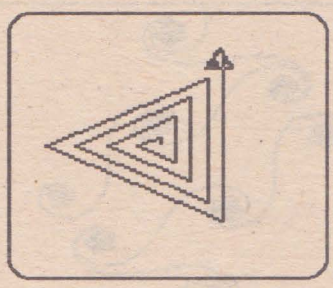


Fig. 38

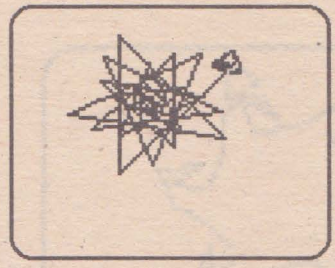


Fig. 39

Fig. 40

60. Introduceți următoarea procedură:

```
TO POLI :L   :U   :C
   IN   :L   SA   :U
   POLI :L   :U + :C   :C
END
```

Utilizând procedura **POLI** puteți realiza următoarele modele interesante:

- POLI 12 100 8 (fig. 41)
- POLI 10 50 7 (fig. 42)
- POLI 10 50 15 (fig. 43)
- POLI 10 50 20 (fig. 44)
- POLI 10 20 60 (fig. 45)

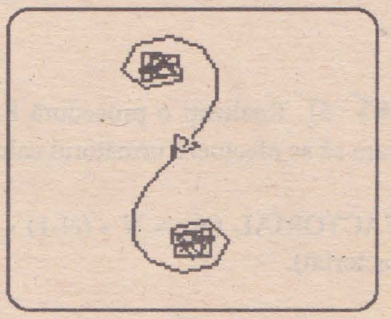


Fig. 41

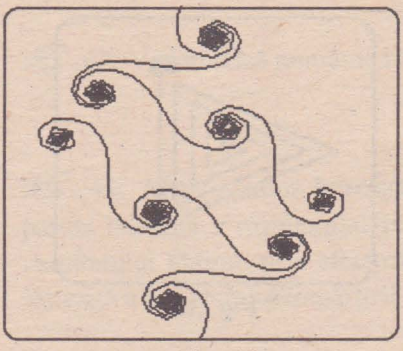


Fig. 42

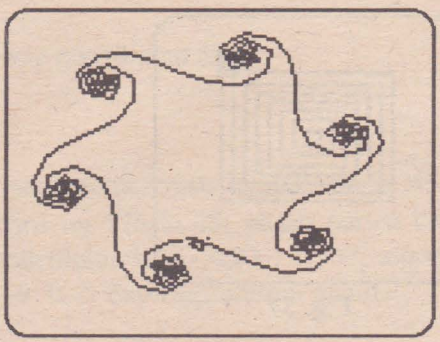


Fig. 43

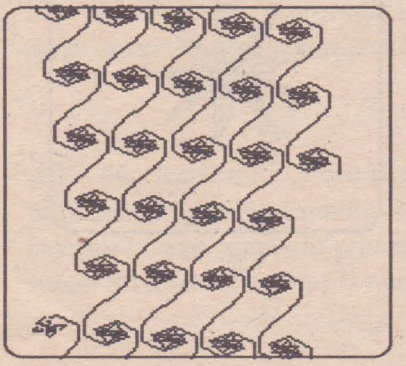


Fig. 44

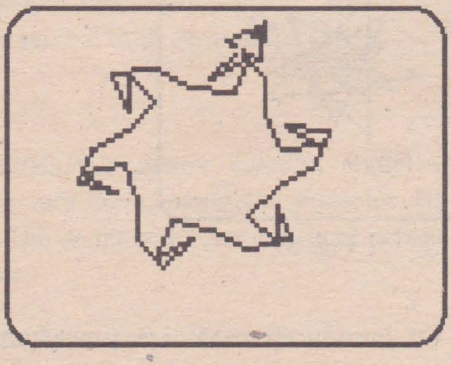


Fig. 45

Caracterizați procedura.
Ce puteți spune despre procedură și utilizarea ei?
Experimentați în continuare procedura pentru mai multe valori ale parametrilor.
Puteți să vă imaginați dinainte cum vor arăta rezultatele pe ecran?

✍ 61. Realizați o procedură FACTORIAL cu un parametru de intrare N cu care să se efectueze următorul calcul:

FACTORIAL (N) = N • (N-1) • (N-2) • ... 3 • 2 • 1 (se mai numește funcția factorial).

UTILIZAREA COORDONATELOR

Lecția 15

Coordonate

Să presupunem că vrem să realizăm o procedură **MUTA** care să deplaseze broasca cu X pixeli pe orizontală și Y pixeli pe verticală, *fără a se trasa desenul parcurs*. Dacă broasca pornește de "acasă", adică din poziția și orientarea inițială, orizontala ei coincide cu orizontala ecranului, iar verticala ei coincide cu verticala ecranului.

Ca să ducem broasca din centru într-un punct oarecare al ecranului folosind procedura **MUTA**, avem nevoie de cele două numere **X** și **Y**, care arată cu cât trebuie mutată broasca din centru "pe orizontală" și respectiv "pe verticală". Procedura ar fi:

```
TO MUTA  :X  :Y
   FC  DR  90  IN  :X
   SA  90  IN :Y
   CREION
END
```

Ca să mutăm acum broasca vom introduce, de exemplu:

```
MUTA 20 50
```

Cu alte cuvinte putem spune că poziția unui punct de pe ecran poate fi *caracterizată* (sau "fixată", sau "definită") printr-un cuplu de două numere. Aceste două numere se numesc *coordonate*; primul se mai numește *abscisă* și se notează, de regulă, cu "X", iar al doilea se mai numește *ordonată*, și se notează, de obicei, cu "Y".

Atunci când indicăm coordonatele unui punct, vom da mai întâi abscisa și apoi ordonata. De exemplu, dacă spunem că un punct A are coordonatele 20 și 50, se subînțelege că abscisa este 20 și ordonata 50. În scris, pentru indicarea poziției unui punct se folosește notația de tipul A(20,50).

Să realizăm acum o procedură **REPER** care să traseze o dreaptă orizontală și una verticală prin poziția inițială a broaștei ca în figura 46, broasca având la sfârșit aceeași poziție și orientare ca la început. Liniile vor fi trasate de la o margine la alta a ecranului care are dimensiunile din figura 47.

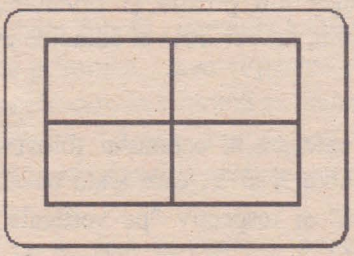


Fig. 46

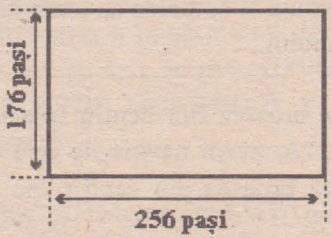


Fig. 47

Reperul desenat de noi pe ecran este format din două drepte perpendiculare. Pe ecran apare doar o parte a acestora, dar ne putem inchipui cu ușurință că aceste drepte sunt nelimitate, fiecare în ambele sensuri.

Un astfel de reper se mai numește *reper ortogonal* (de la "orto", cuvânt grecesc care înseamnă "drept") sau *reper cartezian* (de la "Cartesius", forma latinizată a numelui lui *Descartes*, matematicianul francez care a inventat acest reper).

Cele două drepte ale reperului nu sunt totuși niște "simple drepte". Pe fiecare din ele am stabilit prin convenție un sens pozitiv (la dreapta și, respectiv, în sus) și un sens negativ (la stânga, respectiv, în jos). Am stabilit și o unitate de măsură a lungimilor (a distanțelor și a deplasărilor) care în cazul nostru este "pixelul".

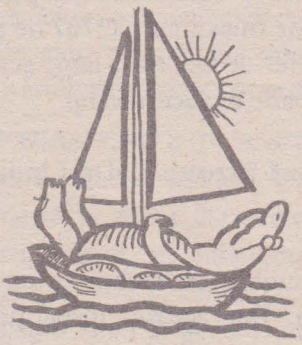
O dreaptă pe care s-a stabilit o origine, un sens pozitiv de parcurgere și o unitate de măsură se mai numește *axă*.

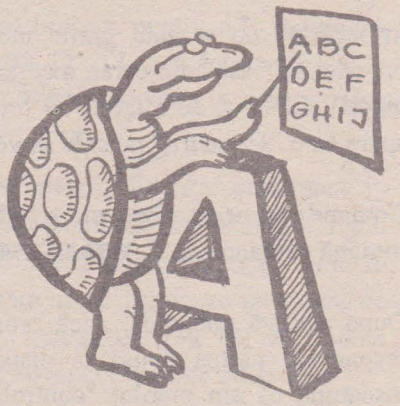
Reperul cartezian este format, deci, din două axe ortogonale.

În plus, am mai stabilit și o ordine a celor două axe: cea orizontală este "prima", iar cea verticală este "a doua".

În concluzie: numim *reper cartezian* o pereche ordonată de axe ortogonale a căror origine comună este chiar punctul de intersecție. Față de un astfel de reper, poziția unui punct oarecare este fixată cu ajutorul a două numere numite "*coordonate*"; cele două coordonate ne arată drumul care trebuie parcurs din origine până în punctul dorit: prima parte a drumului (X-ul) se parcurge pe *axa absciselor*, iar a doua parte (Y-ul) se parcurge ortogonal față de ea. De aceea și coordonatele astfel definite se numesc "ortogonale" sau "carteziene".

Ecranul are pe orizontală (lungime) 256 de pixeli iar pe verticală (lățime) 176 de pixeli.





Lecția 16

Funcții de poziție

Am văzut că dimensiunea orizontală a ecranului este de 256 de pixeli, iar dimensiunea verticală de 176 pixeli. Stabilindu-se ca originea reperului să fie în centrul ecranului, rezultă că pixelii din marginea dreaptă au abscisa de 127, iar cei din marginea stîngă -128.
În mod analog, pixelii de pe latura superioară a ecranului au ordonata +87, iar cei de pe latura inferioară -88 (fig. 48).

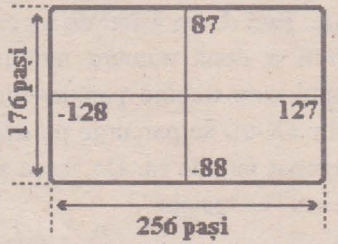


Fig. 48

Limbajul **LOGO** prezintă această organizare a ecranului grafic cu reperul pe care l-am descris. Mai mult decât atât, el conține numeroase primitive care admit ca intrări coordonate ale punctelor sau furnizează informații despre acestea.

În legătură cu ecranul grafic menționăm că există o comandă care permite broaștei să "iasă" din ecran mergând în orice direcție (maximum 32767 de pași-pixeli). Câmpul de mișcare al broaștei devine astfel deosebit de larg; ecranul apare ca o "*fereastră*" pe care se vede doar o mică parte din acest câmp.

Comanda cu acest efect este **WINDOW** care înseamnă "fereastră". Deci în limba română comanda va fi: **FEREASTRA**.

După această comandă, dacă broasca depășește limitele ecranului, ea nu mai reapare în partea opusă ci dispare de pe ecran. Dar această dispariție nu înseamnă că am pierdut "controlul" broaștei. Ea se va afla exact unde i-am comandat și o putem deplasa în continuare.

Dacă nu se dă această comandă, broasca este în situația **WRAP** (sau **IESEREVINE**), ieșirea ei din ecran având ca rezultat revenirea pe partea cealaltă a ecranului. Putem însă să și barăm posibilitatea ieșirii broaștei din ecran cu comanda **FENCE (GARD)**.

Putem să cunoaștem exact coordonatele broaștei cu cele trei primitive care permit acest lucru și care sunt trei funcții (operații) **LOGO**:

XCOR - furnizează abscisa actuală a broaștei

YCOR - furnizează ordonata actuală a broaștei

POSITION (**POS** sau **POZITIE** în limba română) - furnizează "poziția", sub forma unei liste de două numere

Bineînțeles că aceste primitive fiind funcții **LOGO** trebuie să fie folosite ca intrări pentru alte comenzi sau operații. De exemplu:

```
PRINT  XCOR
```

Am văzut că broasca nu este un simplu "punct". Ea se poate *roti* într-o parte sau alta, având, deci, diferite "orientări". Se numește *unghi de orientare* al broaștei (sau, mai scurt, *orientarea* ei) unghiul format de direcția broaștei cu paralela dusă prin ea la axa ordonatelor (axa verticală). Unghiul de orientare se măsoară de la "nord" spre "est", luând valori între 0 și 359 de grade. Funcția **LOGO** **HEADING** (în limba română **DIRECTIE**, adică "orientare") ne furnizează valoarea actuală a orientării broaștei.

Dacă vrem să aflăm ce orientare trebuie să aibe broasca pentru a fi îndreptată spre un anumit punct apelăm funcția **TOWARDS** [X Y], adică broasca să se îndrepte spre punctul de coordonate X și Y. Dintre comenzile care produc modificări în orientarea sau poziția broaștei amintim pe:

- **SETHEADING n** (prescurtat **SETH n**), adică "pune", "fixează" orientarea. În limba română spunem **FIXDIR n.** Această comandă nu schimbă locul broaștei pe ecran, dar o rotește astfel încât să capete orientarea egală cu **n**. În locul intrării "**n**" putem pune și o funcție care să ne furnizeze un unghi de orientare. De exemplu comanda: **FIXDIR TOWARDS [50 70]** va orienta broasca astfel încât să fie orientată spre punctul de coordonate 50 și 70
- **SETPOS** (în limba română **FIXPOZ**), fixează poziția broaștei într-un anumit punct pe ecran

- **SETX** (în limba română **FIXX**), fixează abscisa punctului în care este poziționată broasca
- **SETY** (în limba română **FIXY**), fixează ordonata punctului în care este poziționată broasca

Activități practice

62. Pentru a vă familiariza cu poziția punctelor pe ecran mutați broasca în punctele următoare cu ajutorul procedurii **MUTA** aducând-o apoi de fiecare dată acasă cu ajutorul aceleiași proceduri:
A(20,0) B(125,0) C(-125,0) D(0,-20) E(0,-80) F(80,50) G(-80,50) H(-80,-50) I(80,-50)
Deoarece în procedura **MUTA** noi am considerat că broasca pornește din origine cu orientarea spre NORD, atunci când nu vom mai fi în această situație procedura va da rezultate greșite. Modificați procedura **MUTA** astfel încât, indiferent de poziția și de orientarea broaștei, prin intermediul ei, broasca să se mute X pixeli pe orizontală și Y pixeli pe verticală iar în final orientarea broaștei să fie spre Nord.

63. Ce ordonată are un punct de pe axa absciselor? Ce abscisă are un punct de pe axa ordonatelor? Ce coordonate are centrul ecranului?

64. Dați comanda **FEREASTRA**
Rotiți broasca spre dreapta cu 30 de grade și deplasați-o cu 50 de pași înainte. Cereți afișarea abscisei și ordonatei broaștei. Cereți afișarea poziției actuale a broaștei. Deplasați broasca înainte cu încă 200 de pixeli. Afișați poziția ei.

65. Aduceți broasca ACASA cu procedura **MUTA**. Deplasați-o pe un drum "frânt", format din trei segmente; după fiecare deplasare afișați orientarea actuală a broaștei. Dați comanda pentru a afla orientarea broaștei în situația în care ar fi orientata spre casă (origine). Afișarea unghiului de orientare modifică orientarea însăși a broaștei?
Orientați broasca spre origine! Încercați să "măsurați" cel mai scurt drum de revenire a broaștei acasă.

ALTE OBIECTE LOGO

Lecția 17

Liste, cuvinte, caractere

În limbajul **LOGO** mai multe cuvinte se pot combina și forma astfel o *listă*. Cel mai simplu mod de a realiza o listă este de a închide cuvintele între paranteze pătrate.
O *listă* **LOGO** constă, deci, dintr-o serie de elemente sau *membri* (aceștia se mai numesc *obiecte* **LOGO**) care sunt cuprinse între paranteze drepte. Elementele unei liste sunt separate prin spații. Lista vidă (fără nici un element) este []. *Parantezele drepte aplicate listei nu fac parte din lista, ele avînd drept scop delimitarea listei.*

Exemple de liste:

[ANDREI ADRIAN DOINA 3]

[VACANTA DE VARA]

Am văzut că elementele unei liste se mai numesc *obiecte* **LOGO**. Acestea sunt cuvinte sau liste utilizate ca intrări sau ieșiri ale procedurilor. De exemplu, obiectul **LOGO** "ANDREI" sau obiectul **LOGO** "3", fiecare în parte poate fi folosit ca *subiect al procedurii* SCRIE:

```
SCRIE  "ANDREI
SCRIE  "3   sau   SCRIE  3
```

Deasemenea și obiectul **LOGO** [VACANTA DE VARA] care este o listă poate fi folosit ca subiect al procedurii **SCRIE**:

SCRIE [VACANTA DE VARA]

Cum vom realiza introducerea într-un text a unei linii goale?
Deoarece **SCRIE** este o comandă care necesită neapărat un subiect nu vom putea introduce pur și simplu comanda goală **SCRIE** (ca în BASIC) ci va trebui să folosim lista vidă:

SCRIE []

Un *cuvânt* în **LOGO** reprezintă o serie de caractere alfabetice sau numerice. *Pentru a fi deosebite de numele de proceduri, înaintea cuvintelor se pun ghilimele.* Un cuvânt nu poate cuprinde *caracterul "spațiu"*, deoarece acesta indică terminarea cuvântului Există și *cuvântul vid,* scris doar cu ghilimele, fără nimic după ele. După cum știm cuvintele pot fi utilizate și ca nume de variabile, iar în acest caz nu se folosesc ghilimelele.

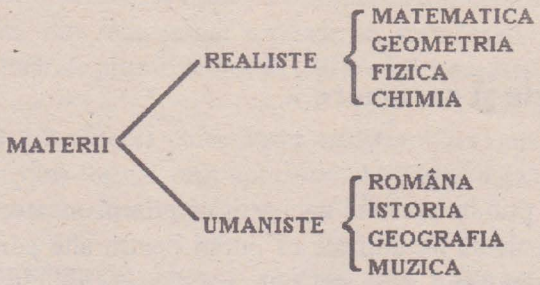
Numerele sunt, de asemenea, cuvinte dar ele se pot scrie și fără ghilimele, așa cum s-a arătat în exemplul dat cu afișarea lui 3.

Elementele unui cuvânt sunt caractere, iar elementele unei liste sunt cuvinte sau liste.

În exemplele de liste date, acestea conțineau numai cuvinte. De aceea se mai numesc *liste plate.* Iată și un exemplu de listă care conține ca elemente nu numai cuvinte ci și liste:

```
[OAMENI [BARBATI FEMEI] ANIMALE [VERTEBRATE NEVERTEBRATE]]
```

Această listă cuprinde două cuvinte (OAMENI și ANIMALE) precum și două liste. Prima conține două cuvinte: BARBATI și FEMEI iar a doua tot două cuvinte: VERTEBRATE și NEVERTEBRATE. Se observă faptul foarte impotant că listele pot fi folosite pentru a grupa (organiza) datele. De exemplu, să considerăm următorul tablou al materiilor scolare:



Această structură formată din părți care, la rândul lor, au alte părți, se poate reprezenta mult mai simplu sub formă de listă **LOGO** astfel:

[MATERII [[REALISTE [ARITMETICA GEOMETRIA FIZICA CHIMIA]]

[UMANISTE [ROMANA ISTORIA GEOGRAFIA MUZICA]]]]

Alt lucru important referitor la liste este acela că *fiecare listă* **LOGO** *este memorată într-o variabilă.* Deci, *atât cuvintele cât și literele pot fi "depuse" în variabile.*

Pentru a ne convinge vom da următorul exemplu:

```
PUNE  "P  POZITIE
```

Deci s-a pus în variabila **P** conținutul lui `POZITIE`. Dar `POZITIE` am văzut că are drept conținut două numere (de fapt, o listă de două numere), primul reprezentând abscisa iar celălalt ordonata. Într-adevăr cu

```
SCRIE  :P
```

vom observa că se va afișa lista coordonatelor broaștei.

Lecția 18

Operații cu liste, cuvinte și caractere

Lucrul cu listele se realizează prin intermediul unei serii de primitive care, fiind funcții (operații) **LOGO**, trebuie să fie utilizate ca intrări pentru alte comenzi sau operații. Descriem în continuare o parte din cele mai folosite primitive cu care se realizează operații cu liste și cuvinte împreună cu rezultatul aplicării fiecărei din aceste funcții.

- **COUNT** obiect - furnizează *numărul de elemente al obiectului dat* ca subiect, așa cum sugerează și cuvântul în limba engleză, adică "*numără câte elemente !*". În limba română primitiva echivalentă este **NREL** (adică numărul relativ de elemente). Bineînțeles, dacă obiectul va fi o listă, atunci **COUNT** va număra câte elemente (cuvinte și liste) conține lista respectivă iar dacă obiectul va fi un cuvânt atunci **COUNT** va număra câte caractere conține

- **ITEM** n lista - furnizează *al n-lea element (articol) din listă*

- **LAST** obiect - furnizează *ultimul element al obiectului*. În limba română primitiva echivalentă este **ULTIMUL**. Bineînțeles că dacă obiectul este o listă, **LAST** va furniza ultimul element (cuvânt sau listă) al listei respective, iar dacă obiectul este un cuvânt, **LAST** va furniza ultimul caracter

- **FIRST** obiect - furnizează *primul element al obiectului*. În limba română primitiva echivalentă este **PRIMUL**

- **BUTLAST** obiect - furnizează obiectul *fără ultimul element*. Există și forma prescurtată **BL**, iar forma echivalentă în limba română este **FARAULTIMUL** (prescurtat **FU**)

- **BUTFIRST** obiect - furnizează obiectul *fără primul element*. Există și forma prescurtată **BF** , iar forma echivalentă în limba română este **FARAPRIMUL** (prescurtat **FP**)

Uneori apare necesitatea ca din mai multe caractere să formăm un nou cuvânt sau ca din mai multe cuvinte să formăm o propoziție (de fapt o listă). Următoarele primitive (tot funcții) rezolvă această cerință:

- **WORD** obiect1 obiect2 sau **WORD** (obiect1 obiect2...obiectn) - furnizează un cuvânt format prin aplicarea obiectelor date. Acestea trebuie să fie numai caractere sau cuvinte .

- **SENTENCE** obiect1 obiect2 sau **SENTENCE** obiect1 obiect2...obiectn furnizează o listă având ca elemente obiectele date. Există și forma prescurtată **SE** precum și forma echivalentă în limba română **FRAZA**.

Primitiva **FRAZA** are o mare importanță deoarece multe instrucțiuni **LOGO** cer o listă ca parametru de intrare; ori, deseori, elementele din care dorim să formăm lista respectivă sunt stocate în diferite variabile sau urmează să rezulte dintr-un calcul.

Astfel, de exemplu, deși sunt delimitate de paranteze drepte, `[:A :B]` și `[5 7 + RANDOM 20]` nu sunt liste, deoarece elementele lor nu sunt cuvinte sau liste. Putem, însă forma liste corecte astfel:

```
SE :A :B
SE 5 7 + RANDOM 20
```

Un exemplu mai elocvent: vrem să orientăm broasca țestoasă cu capul spre un punct ale cărui coordonate sunt depuse în variabilele :X și :Y. Deci scriem:

```
SETHEADING TOWARDS :X :Y
```

Calculatorul nu execută comanda și dă mesajul de eroare :

"TOWARDS nu merge cu :X ca input".

Soluția corectă este:

```
SETHEADING TOWARDS SE :X :Y
```

O altă aplicație a primitivei **FRAZA** este legată de funcția prin care se furnizează o valoare: **OUTPUT**. Se poate observa că multe din procedurile exemplificate realizează dorințele noastre dar rezultatele lor nu se pot folosi în continuare.

De exemplu, procedura care realizează schimbarea valorilor între două variabile (procedura **SCHIMBA** - vezi problema 36). Cu această procedură se vor afișa valorile schimbate între ele ale variabilelor, dar nu le mai putem utiliza în continuare. Pentru a realiza acest deziderat toate aceste tipuri de proceduri vor trebui realizate cu **OUTPUT** deoarece ele ne vor furniza ceva.
În speță procedura **SCHIMBA** va fi modificată astfel:

```
TO SCHIMBA :F :F2
   PUNE "F3 :F1
   PUNE "F1 :F2
   PUNE "F2 :F3
   OUTPUT FRAZA :F1 :F2
END
```

Vom folosi procedura astfel: **SCRIE SCHIMBA 1 2** și se vor afișa numerele în ordine inversă adică **2 1**.
Se observă că această procedură furnizează, spre deosebire de altele, mai multe valori (două). De aceea a fost necesară definirea ei în corelație cu primitiva **FRAZA**.

Următoarele două primitive permit să adăugăm un nou element la o listă constituită anterior:

- **LPUT** obiect listă - furnizează o nouă listă formată din cea dată punându-i ca ultim element obiectul dat. **LPUT** reprezintă, de fapt, prescurtarea cuvintelor LAST PUT care în traducere înseamnă "pus ultimul". De aceea primitiva echivalentă în limba română este **PUNEULTIMUL**.

- **FPUT** obiect listă - furnizează o nouă listă, formată din cea dată, punându-i ca prim element obiectul dat. **FPUT** reprezintă, de fapt, prescurtarea cuvintelor FIRST PUT care în traducere înseamnă "pus primul". De aceea primitiva echivalentă în limba română este **PUNEPRIMUL**.

În sfârșit, următoarele două primitive permit introducerea de la tastatură a unor obiecte **LOGO**, în timpul execuției unei proceduri:

- **READCHAR** (adică "citește caracterul") - comandă calculatorul să aștepte apăsarea unei taste și apoi furnizează caracterul introdus. Acesta trebuie să fie "luat în primire" de procedură, ca orice rezultat al unei operații.

De exemplu:
MAKE "A READCHAR
introduce caracterul citit în A.

Există și forma prrescurtată **RC**, iar forma echivalentă în limba română este **CITCAR.**

- **READLIST** (adică "citește lista") - furnizează lista dată de utilizator la tastatură.

De exemplu:

```
MAKE "L READLIST
```

depune în variabila **L** lista citită. Forma prescurtată este **RL** iar cea în limba română este **CITLIST**.

Aplicații practice

✍ 66. Depuneți în varibilele A, B, C, și D următoarele conținuturi și dați de fiecare dată comandă de afișare a conținutului variabilei respective:

32.26

[35 7]

"MERE

[VACANTA DE VARA LA MARE]

Afișați numărul de elemente al fiecărui obiect creat. Afișați primul element al acestor obiecte.

Depuneți într-o variabilă conținutul variabilei D, mai puțin ultimul element. Afișați al treilea element al noului obiect.

✍ 67. Încercați să vă imaginați ce se va afișa în urma introducerii următoarei linii **LOGO**, apoi verificați-vă cu calculatorul:
SCRIE WORD FU FU FU FU FP FP PRIMUL FP PRIMUL FP [[GOGOSARUL ESTE O LEGUMA] [MULT FOLOSITA DE GOSPODINE]] FU FU FU FU FU FU FU ULTIMUL PRIMUL FP [[GOGOSARUL ESTE O LEGUMA][MULT FOLOSITA DE GOSPODINE]]

Explicați rezultatul obținut.

✍ 68. Realizați o procedură ALDOILEA prin intermediul căreia să se poată extrage al doilea membru al unei liste.

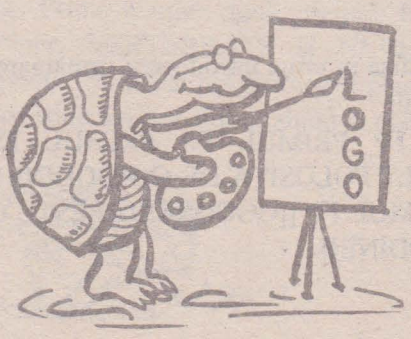
✍ 69. Citiți o listă de la tastatură. Citiți trei caractere de la tastatură. Formați din aceste caractere un cuvânt. Adăugați acest cuvânt la începutul listei. Adăugați-l și la sfârșit.

✍ 70. Să se realizeze o procedură care aplicată unui cuvânt să îl scrie mai întâi complet, apoi fără prima literă, apoi fără primele două litere și așa succesiv până la ultima literă.

✍ 71. Să se realizeze o procedură cu care să se afișeze pătratul numărului introdus de la tastatură.

✍ 72. Să se realizeze un program **LOGO** cu care să se poată face desene pe ecran utilizându-se pentru comenzile de desenat (la dreapta 90 de grade, la stânga 90 de grade, înainte 10 pași și înapoi 10 pași) numai câte o singură tastă. Puteți apoi să îmbunătățiți programul adăugând, de exemplu, posibilități de deplasare a broaștei fără a trasa linii, de ștergere a unor linii și de trasat linii pe alte direcții (diagonale) etc.

✍ 70. Să se introducă de la tastatură o valoare numerică (*simulare* `INPUT`) și apoi să se afișeze rezultatul adunării acestei valori cu numărul 7.



PROBLEME ŞCOLARE

Lecţia 19

Proceduri pentru rezolvări de probleme

Multe din procedurile realizate pot fi folosite pentru rezolvarea diferitelor probleme de şcoală. Am putea să le clasificăm în două grupe, şi anume, *proceduri care pot ajuta la realizarea unor construcţii geometrice* şi, a doua grupă, *proceduri care pot realiza diferite calcule aritmetice*, verificări ale unor egalităţi etc.

Din prima categorie fac parte cele ce trasează diverse figuri geometrice regulate (`TRIUNGHI`, `PATRAT`, `DREPTUNGHI`, `PENTAGON`, `HEXAGON`, `OCTOGON`, `DECAGON`, `CERC`) precum şi `MUTA` care ne poate acorda ajutor în realizarea unor construcţii şi desene geometrice. În a doua categorie putem include procedurile: de calcul a multiplilor numerelor (`MULTIPLI`), de calcul a sumei primelor N numere naturale (`SUMA`), de calcul a sumei pătratelor, a sumei numerelor pare, impare şi a inverselor primelor N numere naturale (`SUMAPAT`, `SUMAPAR`, `SUMAIMPAR`, `SUMAINV`), de calcul al produsului primelor N numere naturale (`PRODUSUL`), de calcul al maximului şi minimului (`MAXMIN`), de calcul al valorii absolute a unui număr (`VALABS`), de calcul al unui număr cu două cifre zecimale (`ZECIMAL`), de calcul al mediei (`MEDIA`), de calcul al factorialului unui număr (`FACTORIAL`), de schimbare a valorilor între două variabile (`SCHIMBA`), de căutare a divizorilor (`DIVIZOR`) etc.

În multe proiecte şi rezolvări de probleme cât şi, mai ales, în realizarea unor experimente, de un foarte mare ajutor ar fi existenţa unor *proceduri prin intermediul cărora să putem măsura diferite mărimi (segmente, unghiuri)*, calculatorul devenind astfel un instrument evoluat. Ne propunem realizarea unor proceduri de acest tip precum şi îmbogăţirea numărului de proceduri care realizează construcţii geometrice cu altele noi.

În vederea măsurării segmentelor de dreaptă, se dovedește utilă existența unor proceduri care să ne permită *manevrarea "manuală" a broaștei țestoase*, adică mișcarea acesteia prin simpla apăsare a unor taste. Astfel de proceduri trebuie să se autoapeleze, pentru ca acțiunea lor să dureze cât dorește utilizatorul. În consecință, ele trebuie să prevadă și posibilitatea opririi autoapelării, prin acționarea unei taste pe care o selectăm de la început, în mod convențional. Să alegem pentru aceasta tasta "S" (de la "STOP").
Structura unei astfel de proceduri va fi, deci:

```
TO MAM
   PUNE  "T CITCAR
   (comenzile de mișcare)
   DACA  :T = "S [STOP] [MAM]
END
```

Am notat cu T variabila în care se depune caracterul furnizat de funcția **CITCAR**, care "preia" caracterul tastat de noi.

Ne propunem acum să scriem o astfel de procedură, *care să conducă broasca înainte, pe direcția ei, la apăsarea unei taste numerice, cu atâția pași, cât indică tasta numerică apăsată*. Deci, dacă tasta apăsată este numerică, broasca va înainta cu pașii respectivi.

În **LOGO** există o funcție cu valori logice, care ia valoarea "adevărat" sau "fals", după cum argumentul ei este numeric sau nu. Această funcție este **NUMBERP**. Folosind funcția aceasta, procedura noastră va arăta astfel:

```
TO MAM
   PUNE "T CITCAR
   IF NUMBERP  :T [IN  :T]
   IF   :T = "S [STOP] [MAM]
END
```

În utilizarea procedurii este posibil ca urmărind înaintarea broaștei, să nu ne oprim la timp. Ar fi de dorit (în acest caz) să ne putem corecta retrăgând broasca. Pentru aceasta, vom introduce, după primul **IF**, instrucțiunea:

```
IF  :T = "G  [GUMA IP 5 CREION]
```

care, prevede ca la apăsarea tastei **G**, broasca să se retragă 5 pași, ștergând linia trasată.

Pentru ștergere se folosește primitiva **GUMA** sau prescurtat **GU**, cuvântul corespunzător în limba engleză fiind **PENERASE** (prescurtat **PE**). După ștergerea liniei este necesar să se revină la starea anterioară a creionului care lasă urme prin înaintarea broaștei, dându-se comanda **CREION**. Bineînțeles dacă dorim ca măsurarea să fie mai exactă, vom înainta și vom șterge linia cu pași mai mici. În acest scop vom modifica subiectul comenzii **INAPOI** punând o valoare mai mică (1, de exemplu).

După cum am arătat, este util ca, deplasând broasca, să *"măsurăm" drumul parcurs*. Pentru aceasta, va trebui ca, *într-o variabilă globală* (fie ea **LD**-lungimea drumului), *să adunăm (sau să scădem) fiecare deplasare.*
Evident **LD** se va inițializa cu 0 înainte de fiecare apelare a procedurii **MAM**. În procedură, după fiecare comandă de deplasare (**INAINTE** sau **INAPOI**), vom introduce modificarea valorii lui **LD**, corespunzătoare deplasării respective. În final, procedura va arăta astfel:

```
TO MAM
   PUNE  "T  CITCAR
   IF NUMBERP  :T  [IN :T PUNE "LD + :T
   IF :T="G [GUMA IP 1 CREION PUNE "LD :LD - 1]
   IF :T="S [STOP] [MAM]
END
```

Apelarea procedurii se face astfel:

```
PUNE "LD 0 MAM
```

urmat, eventual, de:

```
SCRIE  :LD
```

sau

```
PUNE  "var  :LD
```

dacă se dorește "memorarea" lungimii drumului.

Activități practice

74. Utilizați procedura de manevrare manuală a broaștei (**MAM**) pentru a realiza următorul experiment: *trasați un cerc, apoi măsurați diametrul său cu ajutorul procedurii* **MAM** *și afișați rezultatul câtului dintre lungimea cercului și valoarea diametrului măsurat.* Comentați rezultatele obținute.

75. Realizați și testați o procedura **ROTM** care să rotească broasca la dreapta sau la stânga cu câte 5 grade, ori de câte ori se apasă tasta D (de la dreapta) sau S (de la stânga) și încercați să măsurați cu această procedură mărimea diferitelor unghiuri.

76. Realizați și testați o procedură **UNG** cu care să se determine mărimea unui unghi. Procedura **UNG** va avea două intrări: P1 și P2 care reprezintă două puncte distincte și diferite de punctul P în care se află broasca. Practic, procedura P va trebui să calculeze măsura unghiului (P1)P(P2), adică a unghiului U format din dreptele PP1 și PP2 (vezi fig. 49).

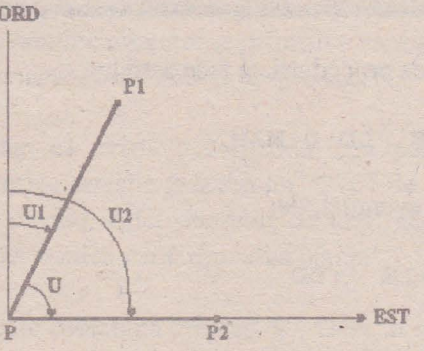


Fig. 49

Lecția 20

Construcții de triunghiuri

Triunghiurile desenate până acum aveau o caracteristică cu totul particulară: erau echilaterale. Ne propunem să desenăm și alte tipuri de triunghiuri.
Știm că pentru a desena un triunghi nu este necesar să cunoaștem toate elementele acestuia (toate laturile și toate unghiurile). Să considerăm, de exemplu, cazul în care cunoaștem doar un unghi și cele două laturi ale acestuia și să notăm cele trei intrări cu L1, U și L2.

Ideea construcției e simplă:

- trasăm latura L1 (de la "coadă" spre vârful unghiului U)
- rotim broasca astfel ca noua ei direcție să facă un unghi U cu cea precedentă. Evident, rotația va fi de 180 - :U deoarece știm că broasca se "rotește" pe unghiul exterior
- trasăm a doua latură (L2)
- rotim broasca spre punctul inițial
- unim punctul final cu cel inițial

Pentru a realiza aceste acțiuni, trebuie să "memorăm" punctul inițial cu:

```
PUNE "P1 POZITIE
```

iar orientarea broaștei spre punctul inițial se face cu:

```
FIXDIR TOWARDS :P1
```

Unirea punctului final cu cel inițial s-ar fi putut face, simplu, cu:

```
FIXPOZ :P1
```

care ne-ar fi "închis" triunghiul. Totuși astfel nu am fi putut afla lungimea celei de a treia laturi. De aceea preferăm să închidem "manual" triunghiul cu:

```
PUNE    "LD 0 MAM PUNE "L3 :LD
SCRIE   :L3
```

aceasta furnizându-ne și mărimea celei de a treia laturi.

Procedura completă **LUL** va fi:

```
TO LUL  :L1 :U :L2
   PUNE "P1 POZITIE
   IN :L1
   SA 180 - :U
   IN :L2
   FIXDIR TOWARDS :P1
   PUNE "LD 0 MAM PUNE "L3 :LD
   SCRIE :L3
END
```

De exemplu, dorim să trasăm un triunghi cu laturile 30 și 22 iar unghiul dintre ele 52 de grade. Vom introduce:

```
LUL 30 52 22
```

și obținem triunghiul precum și mărimea celei de a treia laturi, și anume 23.

Construind astfel triunghiul pe baza a trei elemente am putut constata practic că între cele șase elemente ale triunghiului există unele relații (legături). Într-adevăr, fiind date cele trei elemente cu care am construit triunghiul, celelalte trei elemente *au rezultat automat* : ele nu puteau fi altele decât cele obținute de noi.

Să-ncercăm acum să punem în evidență relația care leagă între ele laturile unui triunghi dreptunghic sau, cu alte cuvinte, *relația lui Pitagora*: *în orice triunghi dreptunghic, pătratul ipotenuzei este egal cu suma pătratelor catetelor*.

Să considerăm un triunghi dreptunghic cu catetele cunoscute. În acest caz triunghiul va putea fi construit cu procedura **LUL**, luând pentru unghiul U valoarea de 90 de grade. După construirea unui triunghi dreptunghic, vom cunoaște și cea de a treia latură (ipotenuza) care este determinată de subprocedura **MAM**. Numele ipotenuzei este :L3.

Vom completa procedura astfel încât, înainte de terminare, ea să tipărească, sub forma unei liste, două valori: *pătratul ipotenuzei* și *suma pătratelor catetelor*. Punem, deci, înainte de **END**, următoarea linie **LOGO**:

```
SCRIE FRAZA  :L3 * :L3  :L1 * :L1 + :L2 * :L2
```

Să luăm câteva exemple:

Pentru

```
LUL 30 90 40
```

obținem valorile: 50 2500 2500

```
LUL 20 90 30
```

obținem valorile: 36 1296 1300

În ultimul exemplu am obținut două valori foarte apropiate (1296 și 1300) deoarece cu procedura **MAM** deplasările nu sunt sub forma fracțiunilor de unități.

Ne convingem de existența acestei relații. Știm că acest adevăr poate fi demonstrat din alte afirmații matematice mai simple; de aceea se mai numește și teoremă.

Activități practice

77. Completați procedura **LUL** astfel încât ea să poată "calcula" și cele trei unghiuri ale triunghiului desenat. Să se folosească procedura **LUL** pentru a rezolva complet triunghiul, adică să se afișeze valorile tuturor laturilor și unghiurilor.

78. Se dau două puncte în plan (vezi fig. 50). Să se realizeze o procedură care să furnizeze *distanța* dintre cele două puncte. Testați procedura comparând rezultatele obținute în cazul unor cupluri de puncte pentru care rezultatul este cunoscut dinainte.

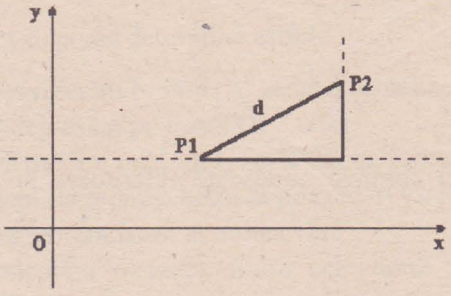


Fig. 50

Lecția 21

Poligoane regulate și cercuri

Știm că pentru a realiza o figură geometrică regulată (poligon) cu N laturi de lungime L putem folosi procedura:

```
TO POLI  :N  :L
   REPETA  :N [IN  :L  SA 360 / :N]
END
```

Pentru a pune în evidență unghiul cu care se rotește broasca (care este, de fapt, unghiul exterior poligonului) vom prefera să scriem procedura **POLI** astfel:

```
TO POLI  :N  :L
   PUNE  "U  360 / :N
   REPETA  :N [IN  :L   SA  :U]
END
```

Ca să ajungem la această formulă am plecat de la ideea că *broasca închide conturul*, deci, *suma rotirilor este de 360 de grade*. Dar broasca se rotește pe unghiul exterior. *De aici rezultă un adevăr pe care îl știam, dar acum îl descoperim pe cale experimentală, și anume, că suma unghiurilor exterioare poligonului regulat este de 360 de grade, dacă poligonul este parcurs într-un singur sens.*

Am notat cu U mărimea unghiului exterior. Unghiul "interior", adică, de fapt, unghiul poligonului are mărimea 180 – U. Prin urmare, suma unghiurilor interioare este:

$N \cdot (180 - U) = N \cdot 180 - N \cdot U$

Dar

$N \cdot U = 360$

Acum putem deduce și din **LOGO** formula cunoscută referitoare la *suma unghiurilor unui poligon* :

$$S = 180 \cdot (N - 2)$$

În legătură cu poziția poligonului desenat cu procedura **POLI** observăm că el are o parte "mai mare" în partea de sus a ecranului față de partea de jos, sau, cu alte cuvinte, este așezat asimetric.
Pentru ca *poligonul să "iasă" simetric* față de orizontala broaștei, vom introduce în procedură, înainte de **REPETA**, rotirea broaștei cu jumătate din unghi, bineînțeles urmând ca în final să readucem broasca în poziția inițială. Noua procedură **POLIS** va realiza poligoane așezate simetric:

```
TO POLIS  :N  :L
   PUNE  "U  360 / :N
   SA  :U/2
   REPETA  :N [IN  :L   SA  :U]
   DR  :U/2
END
```

Prin datele pe care le furnizăm procedurii, noi determinăm *"mărimea"* poligonului, dar *nu știm dinainte cât de mult se va "întinde" el pe ecran.* Știm că poligonul regulat este inscriptibil și, deci, *mărimea lui este caracterizată (determinată) de raza cercului circumscris.*

De fapt, dacă vrem să desenăm un poligon regulat pe hârtie, vom trasa mai întâi un cerc, îl împărțim (cu raportorul) în N părți egale și unim apoi, la rând, punctele respective de pe cerc.

Dar noi am determinat chiar din start mărimea poligonului în momentul când am trasat cercul.

Rezultă că, *dacă vrem să desenăm un poligon regulat cu N laturi, care să "încapă" într-un cerc de rază dată, R, va trebui să determinăm în mod adecvat lungimea laturii L.*

Latura se poate determina astfel:

- considerăm broasca în centrul cercului
- o deplasăm pe cerc (**IN :R**)
- memorăm unghiul U (calculat cu formula 360/N)
- deplasăm iarăși broasca pe cerc în vârful următor al poligonului
- memorăm acest al doilea vârf
- calculăm distanța dintre cele două puncte memorate apelând procedura **DIST**

Procedura **POLIR** de desenare a unui poligon regulat cu N laturi, inscriptibil într-un cerc de raza R (broasca pleacă și revine în centrul cercului) este:

```
TO POLIR  :N  :R
   FARACREION  IN  :R   PUNE  "P1 POZITIE
   IP  :R  SA  360 / :N
   IN  :R  PUNE "P2 POZITIE
   IP  :R  DR  360 / :N
   PUNE    "L  DIST  :P1  :P2
   DR 90   IN  :R   SA 90   CREION
   POLIS   :N  :L
   FARACREION  SA 90  IN :R  DR 90   CREION
END
```

Știm că, dacă numărul laturilor este mare, poligonul obținut seamănă cu un cerc. Prin urmare, pentru desenarea pe ecran a unui cerc de raza **R**, vom da comanda:

```
POLIR  60  :R
```

Totuși, dacă cercul desenat are raza mică, din cauza aproximațiilor de calcul, cerculeţul desenat este prea "colțuros", iar durata desenării este mare.

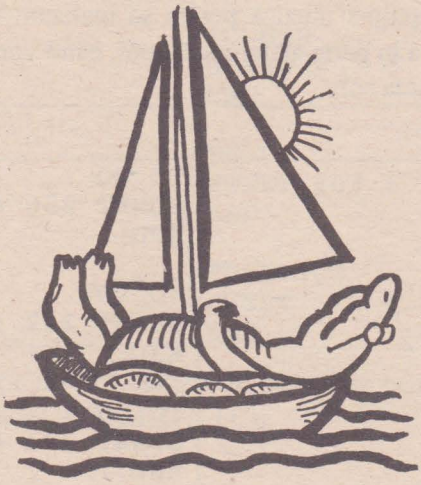
Pentru a remedia aceste neajunsuri, trebuie să reducem volumul de calcule și, deasemenea, numărul laturilor. În acest scop putem modifica modul de calcul al laturii poligonului care "materializează" cercul, mărimea cercului depinzând, în acest caz, de un singur parametru, și anume, raza. Vom folosi formula de calcul a lungimii cercului:

$$L = 2 \Pi r$$

În acest caz latura poligonului (L) se va afla împărțind lungimea cercului la numărul de laturi.

Pentru numărul de laturi vom proceda astfel: dacă lungimea cercului este mai mare de 60 de pași, vom lua ca număr de laturi tot 60 (pentru ca o latura să fie, în orice caz, mai mare de un pas). Dacă lungimea cercului este mai mică decât 60, vom lua ca număr de laturi partea întreagă a lungimii cercului. Practic, căutăm să evităm, de fiecare dată, situația în care latura poligonului ar fi mai mică decât un pas. Iată procedura **CERC** conformă cu planul descris și cu care se pot desena la fel de bine cercuri mari și cercuri mici:

```
TO CERC  :R
   PUNE "LC 2 * 3.14 * :R
   IF :LC < 60 [PUNE  "N INT  :LC] [PUNE  "N  60]
   PUNE  "L  :LC / :N
   PUNE  "U  360 / :N
   FARACREION  IN  :R  SA  90 + :U / 2  CREION
   REPETA  :N [IN  :L  SA  :U]
   DR  90 +  :U / 2  FARACREION  IP  :R  CREION
END
```



ACTIVITĂȚI PENTRU VACANȚĂ

Puțină muzică

Pentru a realiza *sunete* și *note muzicale* putem folosi unul din cuvintele **SOUND** sau **SUNET** urmat de două valori puse într-o listă (deci între paranteze pătrate). Primul va reprezenta *durata sunetului* în secunde, iar al doilea *înălțimea* notei muzicale (este similar cu frecvența sunetului). DO-ul central este reprezentat prin valoarea 0, apoi fiecare unitate va reprezenta câte un semiton. Deci do diez va fi 1, re va fi 2, re diez va fi 3 și așa mai departe pentru sunetele cu înălțimi mai mari. Sunetele cu înălțimi mai mici vor fi reprezentate prin numere negative.

Pentru versiunile **LOGO** pentru calculatoare PC în loc de **SOUND** se folosește **TONE** (în limba română tot **SUNET**), urmat de înălțime și durată dar fără paranteze. În acest caz înălțimea notei muzicale va fi chiar frecvența sunetului respectiv exprimată în herți. Pentru DO-ul central frecvența este, de exemplu, 256. Pentru a face puțină muzică putem să realizăm câte o procedură pentru fiecare notă muzicală în parte astfel încât apoi, când vom comanda, de exemplu, sol, să auzim chiar nota sol:

HC	PC

`TO DO`
`SUNET [1 0]`
`END`
	`TO DO`
	`SUNET 256 4`
	`END`

`TO DO#`
`SUNET [1 1]`
`END`
	`TO DO#`
	`SUNET 262 4`
	`END`

`TO RE`
`SUNET [1 2]`
`END`
	`TO RE`
	`SUNET 271 4`
	`END`

HC	PC
TO RE# SUNET [1 3] END	TO RE# SUNET 304 4 END
TO MI SUNET [1 4] END	TO MI SUNET 322 4 END
TO FA SUNET [1 5] END	TO FA SUNET 342 4 END
TO FA# SUNET [1 6] END	TO FA# SUNET 362 4 END
TO SOL SUNET [1 7] END	TO SOL SUNET 384 4 END
TO SOL# SUNET [1 8] END	TO SOL# SUNET 406 4 END
TO LA SUNET [1 9] END	TO LA SUNET 430 4 END
TO SIB SUNET [1 10] END	TO SIB SUNET 456 4 END
TO SI SUNET [1 11] END	TO SI SUNET 483 4 END
TO DOSUS SUNET [1 12] END	TO DOSUS SUNET 512 4 END

În acest fel ne va fi foarte ușor să realizăm o procedură pentru *gamă* sau *solfegiu* (procedurile vor arăta la fel pentru HC și PC):

```
TO GAMA
DO RE MI FA SOL LA SI DOSUS
END
```

```
TO SOLFEGIU
DO MI SOL DOSUS DOSUS SOL MI DO
END
```

Având la dispoziție aceste note muzicale putem realiza lesne melodii simple ale căror note au, însă, aceeași durată. Putem folosi și *recursivitatea*, în acest caz, cântându-ne aceeași melodie până când întrerupem procedura cu BREAK (CS+Space).

Iată procedurile pentru melodiile Fetițo, Frère Jacque, Copilaș și Hora:

```
TO FETITO
LA LA LA SI LA LA RE LA LA LA SI LA
FETITO
END

TO FREREJ
DO RE MI DO
DO RE MI DO
MI FA SOL SOL
MI FA SOL SOL
FREREJ
END

TO COPILAS
DO MI SOL FA MI RE LA SOL SOL DO MI SOL FA MI RE MI
DO DO RE RE MI DO RE MI FA SOL SOL RE RE MI DO RE
DO DO DO
COPILAS
END

TO HORA
MI MI SOL SOL FA MI SOL LA LA SOL FA MI RE SOL MI
MI SOL SOL FA MI SOL LA LA SOL FA MI RE DO
HORA
END
```

Dar în majoritatea melodiilor notele nu au aceeași durată. De asemenea, se mai folosesc și pauze. Pauzele se obțin cu **WAIT** sau **ASTEAPTA** care va fi urmat de o valoare, N. La HC se va face o pauză de N/50 secunde, iar la PC de N/18 secunde. Pentru realizarea unor note de o anumită durată, procedurile se vor modifica astfel:

HC	PC

`TO DO :D`
`SUNET SE :D 0`
`END`
	`TO DO :D`
	`TONE 256 :D`
	`END`

Să exemplificăm cu melodia **LET'S GO** care, deși cântată pe o singură notă, era foarte la modă prin anii '60 pentru ritmul său. Duratele notelor se succed astfel:

pătrime pătrime optime optime doime
optime optime optime pătrime
optime optime pauză
repetiție

```
TO LETSGO
DO 8 DO 8 DO 4 DO 4 DO 12
DO 4 DO 4 DO 4 DO 8
DO 4 DO 4 WAIT 12
LETSGO
END
```

Într-un mod asemănător putem realiza melodii cu note diferite și de durate diferite, incluzând și pauze.

Fulgi de nea

Deși este foarte cald (sau tocmai de aceea) ne gândim ce frumos era iarna când ningea și în acest scop ne propunem să realizăm niște fulgi de nea.

Iată o idee pentru a realiza un fulg de nea:

Trasați un triunghi echilateral, apoi împărțiți fiecare latură în trei segmente egale. Pornind de la aceste segmente, trasați din nou triunghiul echilateral spre exterior. Ștergeți latura comună cu triunghiul mare. Veți obține o stea cu 5 colțuri (fig. 51). Repetați operația și obțineți fulgul de nea dorit.

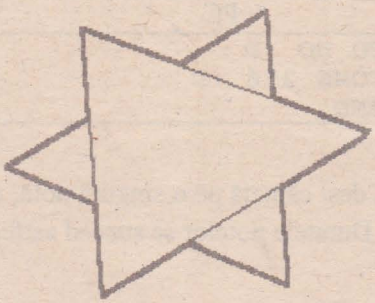


Fig. 51

Să încercăm să facem procedura pentru fulg (pe bază de triunghi). Procedura principală va avea doi parametri: lungimea (L) (care reprezintă mărimea laturii triunghiului de bază determinând, deci, mărimea fulgului) și nivelul, N. Nivelul 0 (cel mai mic) va fi reprezentat de triunghi, nivelul 1, de stea și așa mai departe. Cu cât nivelul va fi mai mare cu atât fulgul va fi mai "pufos". Procedura principală va apela o subprocedură, **LATURA**, care are scopul de a lăsa broasca în poziția corespunzătoare pentru a trasa latura următoare precum și de a calcula laturile mici:

```
TO FULG :L :N
   REPETA 3 [LUNGIME :L :N DREAPTA 120]
END

TO LUNGIME :L :N
IF :N = 0 [INAINTE :L   STOP]
[LUNGIME :L/3 :N-1   STINGA 60   LUNGIME :L/3 :N-1
DREAPTA 120   LUNGIME :L/3   :N-1     STINGA 60
LUNGIME :L/3 :N-1]
END
```

Se obțin următorii fulgi de nea:

FULG 60 0 - (fig. 52)

FULG 60 1 - (fig. 53)

FULG 60 2 - (fig. 54)

FULG 60 3 - (fig. 55)

FULG 60 4 - (fig. 56)

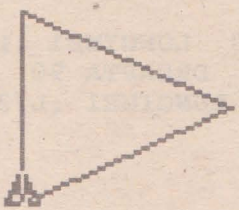


Fig. 52

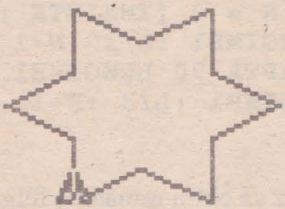


Fig. 53

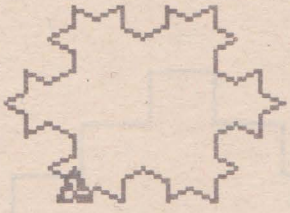


Fig. 54

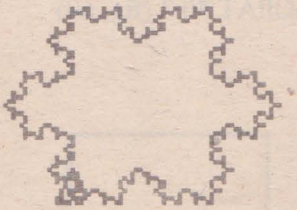


Fig. 55

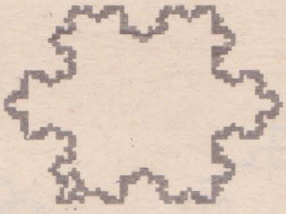


Fig. 56

Se poate obține fulg de nea și pe bază de pătrat, împărțind fiecare latură în trei părți egale și așa mai departe. În acest caz procedura va arăta astfel:

```
TO FULGPAT :L :N
REPETA 4[LUNGIME1 :L :N DREAPTA 90]
END
```

```
TO LUNGIME1 :L :N
IF :N = 0 [INAINTE :L  STOP]
[LUNGIME1 :L/3 :N-1  STINGA 90  LUNGIME1 :L/3 :N-1
DREAPTA 90 LUNGIME1 :L/3 :N-1  DREAPTA 90
LUNGIME1 :L/3 :N-1  STINGA 90 LUNGIME1 :L/3 :N-1]
END
```

În acest caz se obțin următorii fulgi:

FULGPAT 50 0 (fig. 57)
FULGPAT 50 1 (fig. 58)
FULGPAT 50 2 (fig. 59)
FULGPAT 50 3 (fig. 60)

Fig. 57

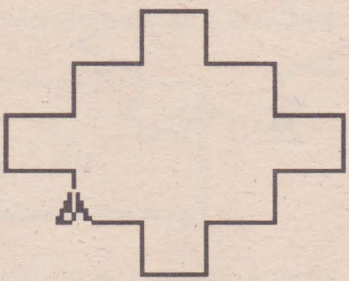


Fig. 58

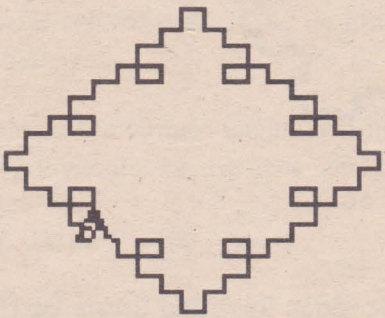


Fig. 59

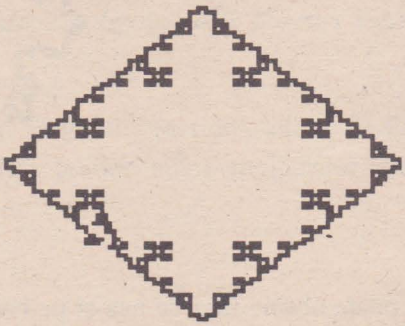


Fig. 60

Forme grafice complexe

Am mai realizat figuri interesante. Am văzut că majoritatea lor sunt forme circulare bazate pe teorema rotirii complete a broaștei. Să pornim de la o formă poligonală, încercând ca, indiferent de poziția inițială a broaștei, să realizăm figura în centrul ecranului iar la sfârșit să lăsăm broasca deasupra figurii, astfel încât să nu ne incomodeze vizualizarea.

```
TO POLI  :R  :U
   PU
   SETPOS LIST  :R * SIN HEADING  :R * COS HEADING
   PD
   POLI1  :R  :U  HEADING
END

TO POLI1  :R  :U  :INC
   RT  :U
   SETPOS  LIST  :R * SIN HEADING  :R * COS HEADING
   IF HEADING = :INC [STOP]
   POLI1  :R  :U  :INC
END
```

Semnificația variabilelor locale este următoarea:

- **R** este raza cercului care circumscrie figura determinând mărimea acesteia
- **U** este unghiul și determină forma figurii. Din acest punct de vedere cea mai importantă instrucțiune (și care determină forma figurii noastre poligonale este **RT :U** din procedura **POLI1**)
- **INC** este direcția pe care o are broasca la început

De notat faptul că am preferat utilizarea cuvântului **LIST** pentru ca procedurile să fie executabile atât pe calculatoare HC cât și PC. Forma **LIST** pune obiectele care urmează într-o listă. (Pentru calculatoarele HC am văzut că se poate folosi în loc de **LIST** cuvântul **SENTENCE**).

Modificând valoarea parametrului **U** (unghi) obținem următoarele forme:

POLI 30 60	un hexagon cu latura 30 (fig. 61)
POLI 30 90	un pătrat cu latura 30 (fig. 62)
POLI 30 144	o stea cu latura 30 (fig. 63)
POLI 30 210	o stea cu mai multe vârfuri (fig. 64)

POLI 30 1000 două poligoane (fig. 65)

POLI 30 1010 un colac (fig. 66)

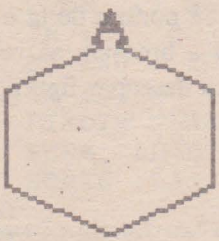


Fig. 61

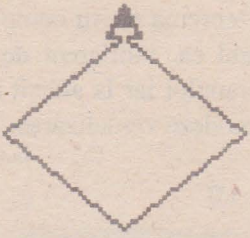


Fig. 62

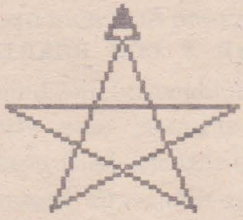


Fig. 63

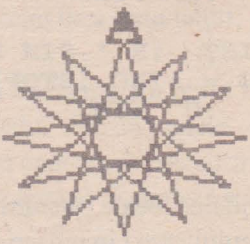


Fig. 64

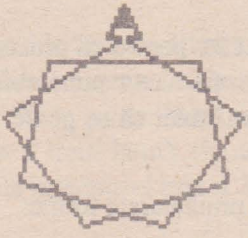


Fig. 65

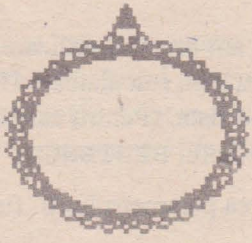


Fig. 66

Umplerea figurilor

Pentru un limbaj de programare care pune un accent deosebit pe grafică, o rutină de umplere a contururilor este deosebit de utilă. În **LOGO** umplerea figurii `POLI` se poate realiza destul de eficient folosind recursivitatea și micșorând latura poligonului cu o unitate până devine nulă. Procedura ar putea fi:

```
TO POLI.PLIN  :R  :U
   IF  :R = 0  [STOP]
   POLI  :R  :U
   POLI.PLIN  :R - 1  :U
END
```

Astfel, `POLI.PLIN  40  90` umple un pătrat cu latura 40 (fig. 67) iar `POLI.PLIN  40  144` umple conturul unei stele cu latura 40 (fig. 68).

Fig. 67

Fig. 68

Procedura `POLI.PLIN` umple întreaga figură. Dar uneori este necesară o procedură de umplere (incompletă) de diferite grosimi. În acest scop se poate modifica procedura astfel încât să prezinte trei parametri și anume, o margine superioară (`MARE`), o margine inferioară (`MIC`) și, bineînțeles, unghiul (`U`) care determină forma figurii. În acest caz umplerea se va realiza doar între marginea superioară (`MARE`) și marginea inferioară (`MIC`). Noua procedură de umplere va arăta astfel:

```
TO POLI.PLIN  :MARE  :MIC  :U
   POLI  :MARE  :U
   IF NOT  :MARE  >  :MIC  [STOP]
   POLI.PLIN  :MARE - 1  :MIC  :U
END
```

`POLI.PLIN 35 30 90` va umple o porțiune (grosime, strat) de 5 dintr-un pătrat (fig. 69) iar `POLI.PLIN 35 15 120` va umple o porțiune de 20 dintr-un triunghi (fig. 70).

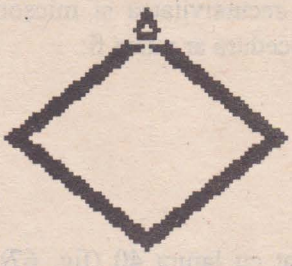


Fig. 69

Fig. 70

Ochean

Folosind procedurile de umplere a figurilor (`POLI.PLIN`) și de ștergere a unor figuri și porțiuni umplute (`STERGE.POLI` și `STERGE.POLI.PLIN`) se pot realiza efecte grafice foarte interesante care, privite în dinamică pe ecranul calculatorului, ne aduc aminte de ocheane - jucării simple prin care se pot vedea superbe combinații de mici figuri geometrice. Ideea este de a se umple un strat dintre două forme poligonale diferite. Iată procedurile:

```
TO OCHEAN :D :R :U1 :U2
   IF :R < 20 [STOP]
   POLI.PLIN :R :R - :D :U1
   POLI.PLIN :R :R - :D :U2
   POLI.PLIN :R :R - 2 * :D :U2
   STERGE.POLI.PLIN :R - :D :R - 3 * :D :U1
END
```

```
TO STERGE.POLI.PLIN :MARE :MIC :U
   IF NOT :MARE > :MIC [STOP]
   STERGE.POLI :MIC :U
   STERGE.POLI.PLIN :MARE :MIC + 1 :U
END
```

```
TO STERGE.POLI  :R  :U
   PU SETPOS LIST  :R * SIN HEADING  :R * COS HEADING
   PE
   RT   :U
   POLI :R  :U
END
```

Semnificația variabilelor este următoarea:

- D este distanța dintre marginile stratului superior și inferior (grosimea)
- R este raza cercului care determină stratul cel mare
- U1 este unghiul care determină forma primului poligon înscris
- U2 este unghiul care determină forma celui de-al doilea poligon înscris

Încercați următoarele și priviți figurile rezultante în devenirea lor:

OCHEAN 10 40 135 45 (fig. 71)

OCHEAN 5 40 90 135 (fig. 72)

OCHEAN 12 80 140 160 (fig. 73)

OCHEAN 5 80 120 60 (fig. 74)

OCHEAN 7 80 40 1000 (fig. 75)

OCHEAN 12 70 120 45 (fig. 76)

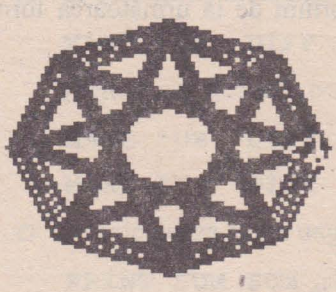


Fig. 71

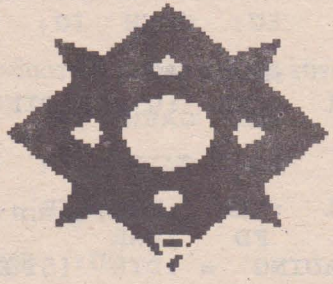


Fig. 72

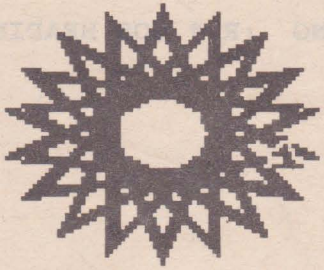


Fig. 73

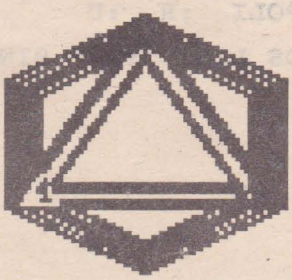


Fig. 74

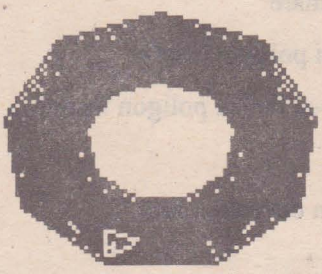


Fig. 75

Fig. 76

Forme circulare complexe

Mergând pe ideea de dinamică să încercăm să realizăm proceduri care dau senzația de rotire a formelor. În acest scop să pornim de la următoarea formă poligonală:

```
TO FORMAP    :MAR   :U
   FORMAP1   :MAR   :U  HEADING
END


TO FORMAP1   :MAR   :U   :DIR
   RT  :U     FD    :MAR
   IF  HEADING  =  :DIR  [STOP]
   FORMAP1   :MAR   :U   :DIR
END
```

```
TO FORMACIRC  :MAR  :U  :D  :R  :NR
   RT  :D / :NR
   PU  SETPOS  LIST  :R * COS :NR * HEADING
   :R * SIN :NR * HEADING  PD
   FORMAP  :MAR  :U
   IF HEADING = 0 [STOP]
   FORMACIRC  :MAR  :U  :D  :R  :NR
END
```

Semnificația variabilelor:

- **MAR** este mărimea laturii poligonului (formei) care se rotește
- **U** este unghiul care determină forma poligonului. Dacă unghiul este 180 se va desena practic o dreaptă, adică o spiță de la roată
- **D** determină în mod invers proporțional densitatea poligonului
- **R** este raza cercului central
- **NR** reprezintă numărul de rotații ale figurii în jurul propriului centru pentru fiecare mișcare de revoluție pe care forma o face pe traiectoria cercului central

Iată și interesante forme circulare care se obțin cu procedura **FORMACIRC**. Puteți să vă imaginați dinainte formele care vor apărea ?

FORMACIRC 35 90 10 50 1 (fig. 77)

FORMACIRC 50 180 10 50 1 (fig. 78)

FORMACIRC 50 180 10 60 1 (fig. 79)

FORMACIRC 30 180 4 60 2 (fig. 80)

FORMACIRC 30 180 5 60 3 (fig. 81)

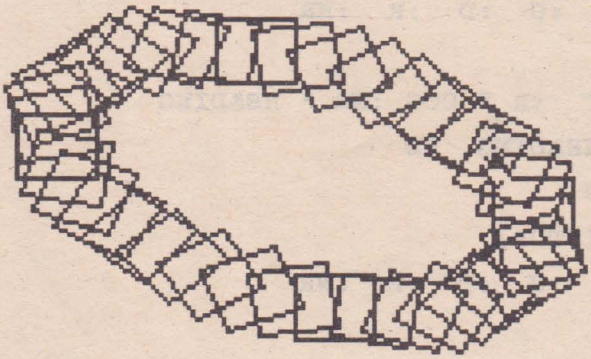


Fig. 77

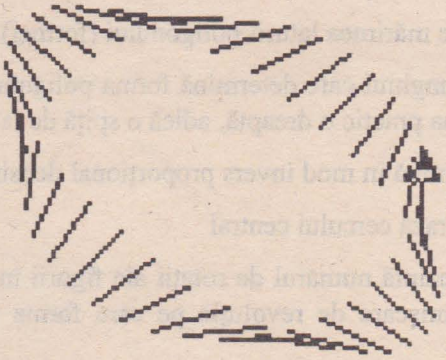


Fig. 78

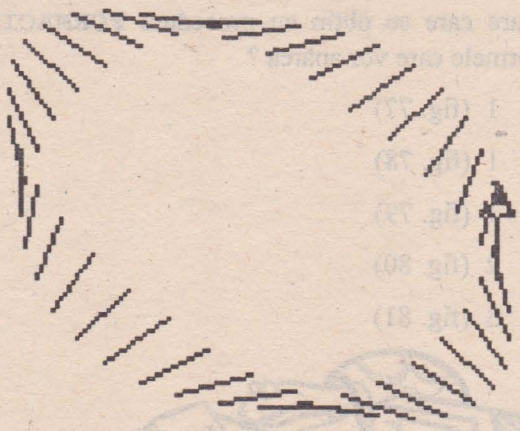


Fig. 79

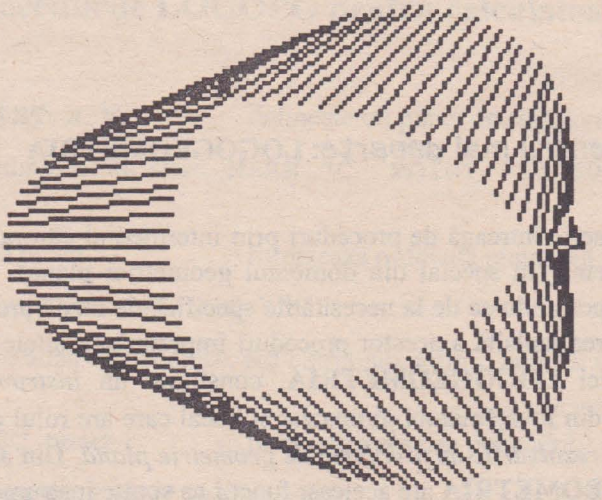


Fig. 80

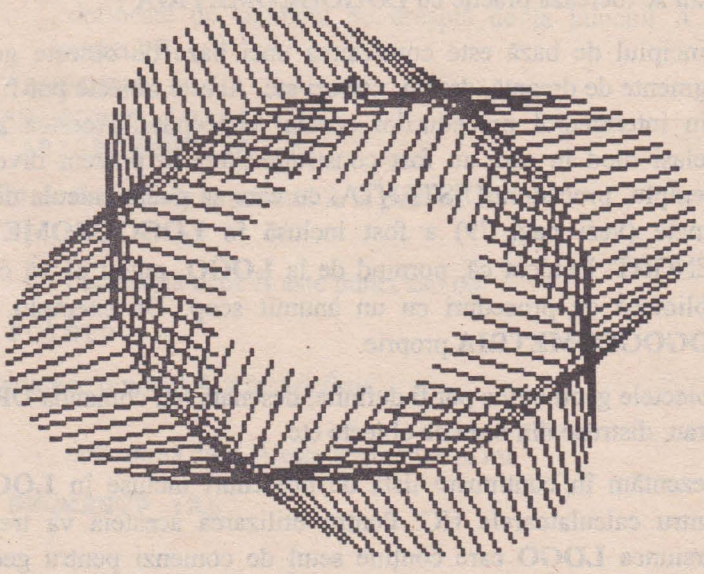


Fig. 81

O deschidere pentru mai departe: LOGOGEOMETRIA

Până acum am realizat o serie întreagă de proceduri prin intermediul cărora am rezolvat o serie de probleme (în special din domeniul geometriei plane). Dar acestea au fost construite pornindu-se de la necesitățile specifice de rezolvare ale anumitor probleme. Punerea laolaltă a acestor proceduri împreună cu altele noi care au același scop, deci **LOGOGEOMETRIA**, constituie un *instrument* pentru elevii și profesorii din învățământul gimnazial și liceal care are rolul de a *facilita experimentarea* și *rezolvările* de *probleme de geometrie plană.* Din acest punct de vedere **LOGOGEOMETRIA** are aceleași funcții ca și alte instrumente folosite de elevi la geometrie (compasul sau rigla, de exemplu), folosind însă, în plus, facilitățile grafice ale sistemului **LOGO**.

Cum se lucrează practic cu **LOGOGEOMETRIA** ?

Principiul de bază este construirea unei baze de obiecte geometrice: puncte, segmente de dreaptă, drepte, vectori etc. Aceste obiecte pot fi apelate prin nume prin intermediul procedurilor special concepute. Acestea au fost incluse în același mod în care au fost construite până în prezent diverse proceduri. De exemplu, procedura DISTANTA, cu care se poate calcula distanța dintre două puncte (vezi pag. 79) a fost inclusă în **LOGOGEOMETRIA** cu numele LENGHT. Evident că, pornind de la **LOGO**, puteți să vă construiți singuri o bibliotecă de proceduri cu un anumit scop. De exemplu, să vă realizați o **LOGOGEOMETRIA** proprie.

Obiectele geometrice pot fi definite, desenate (cu comanda DRAW), șterse de pe ecran, distruse din baza de obiecte etc.

Prezentăm în continuare lista de proceduri incluse în **LOGOGEOMETRIA** pentru calculatoarele HC. Pentru utilizarea acesteia va trebui să se încarce versiunea **LOGO** care conține setul de comenzi pentru geometrie și care se numește **LOGOPG**. De notat că din cauza memoriei mici a calculatorului HC în **LOGOPG** nu se pot folosi și comenzile în limba română.

Procedurile LOGOPG pentru calculatoare HC

POINT X Y definește un punct de coordonate X și Y.

Exemplu de utilizare: **MAKE "C POINT -60 60**

ORD punct furnizează ordonata punctului.

Exemplu: **PRINT ORD :C**

se va afișa 60 deoarece, așa cum l-am definit, punctul C are ordonata 60.

ABSC punct furnizează abscisa punctului.

Exemplu: **PRINT ABSC :C**

se va afișa -60.

SEGMENT A B definește un segment de dreaptă de la punctul A la punctul B.

Exemple:
```
MAKE "A POINT 0 60
MAKE "B POINT 60 60
MAKE "AB SEGMENT :A :B
```

definește un segment de dreaptă de la punctul A la punctul B.

POINTP A determină dacă A este punct sau nu.

Exemplu: **PRINT POINTP :A**

se va afișa **TRUE** deoarece punctul A a fost definit.

SEGMENTP A determină dacă A este segment sau nu.

Exemplu: **PRINT SEGMENTP :AB**

se va afișa **TRUE**.

LENGHT A determină lungimea segmentului A.

Exemplu: **PRINT LENGHT :AB**

se va afișa 60.

LINE A B — definește o dreaptă care trece prin punctele A și B sau o linie care trece prin punctul A având înclinația B față de orizontală.

Exemple: **MAKE "D1 LINE :A :B**

MAKE "D2 LINE :A 60 (vezi fig. 82)

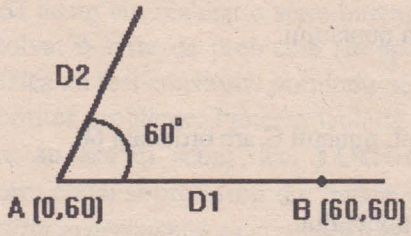


Fig. 82

PRINT LENGHT LINE POINT 0 0 POINT 50 50

se va afișa rezultatul, adică mărimea segmentului dintre punctul (0,0) și punctul (50,50), adică 70,7106878.

HEAD D — furnizează înclinația unei linii față de orizontală.

Exemplu: **PRINT HEAD :D1**

se va afișa 0 deoarece înclinația dreptei D1 față de orizontală este 0 (punctele A și B au aceeași ordonată).

LINEP D — determină dacă D este o dreaptă sau nu.

Exemplu: **PRINT LINEP :D1**

se va afișa **TRUE**.

DIFF X Y — furnizează diferența între X și Y.

Exemplu: **PRINT DIFF ORD :A ABSC :A**

se va afișa 60 (ordonata punctului A este 60 iar abscisa 0).

ISEC A B furnizează punctul de intersecție a două segmente sau drepte A și B.

Exemplu: **PRINT ISEC :D1 :D2**

se va afișa P 0 59.999, adică intersecția dreptelor D1 și D2 este un punct care are abscisa 0 și ordonata 60.

VECTOR A determină un vector dintr-un segment sau o dreaptă A.

Exemplu: **MAKE "V VECTOR :D1**

VECTORP A determină dacă A este vector sau nu.

Exemplu: **PRINT VECTORP :V** se va afișa **TRUE**.

DRAW A desenează obiectul A.

Exemplu:

DRAW :AB se va desena segmentul de dreaptă dintre punctele A(0,60) și B(60,60).

DRAW :C se va desena punctul C(-60,60)

DRAW :D1 DRAW :D2 se vor desena dreptele D1 și D2 (vezi fig. 83).

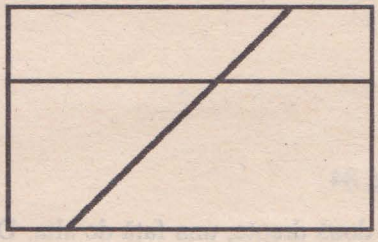


Fig. 83

REDRAW A desenează obiectul A și îl introduce în baza de date.

Exemplu: **REDRAW :AB**

se va desena din nou segmentul de dreaptă AB.

CLRDR șterge ecranul, eliberează spațiul de lucru, desenează obiectele din baza de date.

HIDE șterge baza de date.

3 probleme de geometrie rezolvate cu LOGOPG

1. Care este poziția a trei puncte ? Sunt coliniare ?

Să presupunem că avem trei puncte construite astfel:

```
MAKE "A  POINT   0  60
MAKE "B  POINT  60  60
MAKE "C  POINT  60   0
```

Pentru rezolvarea problemei parcurgem următorii pași:

- se construiesc două drepte care trec prin punctele A și B și, respectiv, B și C:

```
MAKE "D1  LINE  :A  :B
MAKE "D2  LINE  :B  :C
```

pentru realizarea desenului `DRAW :D1 DRAW :D2` (fig. 84)

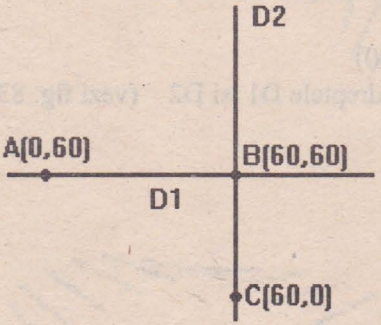


Fig. 84

- se calculează înclinarea pe care o au cele două drepte, una față de alta. Dacă se cercetează dacă cele trei puncte sunt coliniare, se verifică dacă diferența înclinațiilor este 0 sau 180° (două drepte paralele care trec prin același punct sunt identice).

```
PRINT DIFF   HEAD  :D1   HEAD  :D2
```

În cazul nostru se va afișa 90°, de unde rezultă că cele două drepte sunt perpendiculare.

2. Suma unghiurilor unui triunghi este 180°.

Să presupunem că am construit următoarele trei puncte:

```
MAKE "A  POINT   0  0
MAKE "B  POINT  60  0
MAKE "C  POINT   0 60
```

Pentru demonstrație se parcurg următorii pași:

- se construiesc dreptele ce definesc laturile triunghiului (vezi fig. 85)

```
MAKE "D1  LINE  :A  :B
MAKE "D2  LINE  :C  :B
MAKE "D3  LINE  :A  :C
```

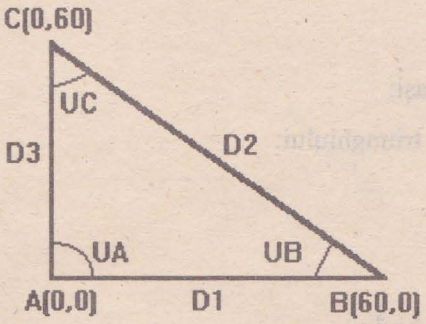


Fig. 85

- se determină înclinația celor trei drepte:

```
MAKE "UA  DIFF  HEAD  :D1  HEAD  :D3
```

(pentru verificare se poate introduce `PR :UA` și se va afișa 90)

```
MAKE "UB  DIFF  HEAD  :D1  HEAD  :D2
MAKE "UC  DIFF  HEAD  :D2  HEAD  :D3
```

- se calculează suma unghiurilor triunghiului:

```
PRINT SUM SUM :UA  :UB  :UC
```

Se va afișa 180.

3. Înălțimile unui triunghi se intersectează în același punct.

Să presupunem că am construit un triunghi care are vârfurile în punctele definite astfel (vezi fig. 86):

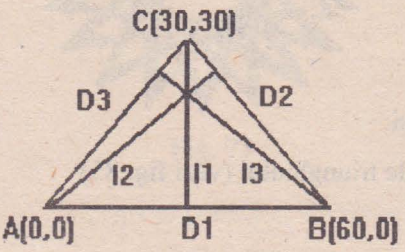


Fig. 86

```
MAKE "A  POINT   0  0
MAKE "B  POINT  60  0
MAKE "C  POINT  30 30
```

Pentru demonstrație se parcurg următorii pași:

- se definesc liniile ce determină laturile triunghiului:

```
MAKE "D1  LINE  :A  :B
MAKE "D2  LINE  :C  :B
MAKE "D3  LINE  :C  :A
```

- se determină înclinațiile laturilor triunghiului:

```
MAKE "U1  HEAD  :D1
MAKE "U2  HEAD  :D2
MAKE "U3  HEAD  :D3
```

- se determină înclinațiile înălțimilor:

```
MAKE "UI1  SUM  90  :U1
MAKE "UI2  SUM  90  :U2
MAKE "UI3  SUM  90  :U3
```

- se determină înălțimile ca drepte ce trec printr-un vârf al triunghiului având o anumită înclinație:

```
MAKE "I1  LINE  :C  :UI1
MAKE "I2  LINE  :A  :UI2
MAKE "I3  LINE  :B  :UI3
```

Pentru vizualizare:

```
DRAW :D1 DRAW :D2 DRAW :D3 DRAW :I1 DRAW :I2 DRAW :I3
```

- se determină punctele de intersecție ale înălțimilor, două câte două:

```
PRINT ISEC  :I1  :I2
PRINT ISEC  :I2  :I3
PRINT ISEC  :I3  :I1
```

Coordonatele acestor puncte coincid (se afișează de fiecare dată 30, 30).

LOGOGEOMETRIA pentru calculatoarele PC este un sistem mult mai complex, fiind constituit din mai multe module care sporesc facilitățile.

După ce a fost construit, un obiect poate fi desenat, legat funcțional de un altul sau transformat prin simpla invocare a numelui său și a procedurii corespondente.

Iată câteva din modulele mai importante:

- **GEOMBAZA** este un modul prin care se pot introduce puncte, segmente și drepte. De asemenea, se pot construi mediatoare de segmente, perpendiculare și punctul de intersecție a două drepte. Prezența acestui modul este necesară în permanență.
- **GEOMANG** introduce semidrepte, unghiuri, bisectoare.
- **GEOMTRI** este un modul referitor la triunghiuri care pot fi construite cu 3 puncte și un nume.
- **GEOMPOLI** este o extensie a modulului GEOMTRI pentru cazul general al poligoanelor.
- **GEOMTRAN** semnifică transformări ca: simetrie față de axă, translații, rotații, omotetii, inversări, simetrii de reflexie (raza reflectată).

Modulul **GEOMBAZA** este lansat automat de **LOGO** când se inițializează **LOGOGEOMETRIA**. Apoi, când se dorește un anumit modul, acesta va fi apelat prin numele său.

Studiul LOGOGEOMETRIA pentru PC și rezolvarea problemelor de geometrie cu acest instrument face însă obiectul unei alte lucrări.

RĂSPUNSURI ȘI INDICAȚII LA PROBLEMELE ȘI EXERCIȚIILE PROPUSE ÎN CADRUL ACTIVITĂȚILOR PRACTICE

1.
```
?SCRIE  3+(5.2-(4*7-(30+20)/(3+2)*2.1+(4+10.3)*
*(4-0.3))*3.5+4.3*2-(1.57*3)+12.4)
```

Rezultatul afișat va fi: **- 185.195**

2.
```
?SCRIE 10
 SCRIE "TOTAL
 SCRIE [T O T A L]
 SCRIE [2+2=4]
 SCRIE [2+1=2]
 SCRIE [Ultimul rezultat este gresit]
 SCRIE [TOTAL 6]
```

3.
```
INAINTE 50 STINGA 90 INAINTE 50 STINGA 90 INAINTE 50
STINGA 90 INAINTE 50 STINGA 90
```

Observați că pătratul este construit pe stânga.

4.
```
INAINTE 30 DREAPTA 90 INAINTE 70 DREAPTA 90 INAINTE 30
DREAPTA 90 INAINTE 70 DREAPTA 90
```

Observați că dreptunghiul este construit pe dreapta.

5.
```
INAINTE 50 INAPOI 50 DREAPTA 60 INAINTE 50
```

6. *Unghiurile unui triunghi echilateral sunt egale având fiecare* 60 *de grade* (180 : 3 = 60). *Broasca se rotește spre stânga cu* 180 - 60 = 120 *de grade.*

```
INAINTE 40 STINGA 120 INAINTE 40 STINGA 120 INAINTE 40
STINGA 120
```

7. *Ipotenuza va avea* 70,7 *pași. Deci, diagonala pătratului este mai mare decât latura. Figura se poate desena și prin încercări.*

```
INAINTE  50  STINGA  135  INAINTE  70,7  STINGA  135
INAINTE  50  STINGA 90
```

8. `DREAPTA 45 INAINTE 60 STINGA 45 INAINTE 30 DREAPTA 90 INAINTE 40 DREAPTA 90 INAINTE 60`

9. `INAINTE 40 INAPOI 40 STINGA 120 INAINTE 40 INAPOI 40 STINGA 120 INAINTE 40 INAPOI 40 STINGA 120`

10. *Se formează* 3 *unghiuri egale cu* 180 : 3 = 60 *de grade.*

```
SA 90 IN 50 IP 50 DR 60 IN 50 IP 50 DR 60 IN 50 IP 50
DR 60 IN 50 IP 50 SA 90 IN 50 IP 50 SA 90
```

11. *Suma rotirilor broaștei trebuie să fie* 360 *de grade, iar broasca se va roti de* 5 *ori, deci, la fiecare rotire* 360 : 5 = 72 *de grade.*

```
IN 50 SA 72 IN 50 SA 72 IN 50 SA 72 IN 50 SA 72 IN 50
SA 72
```

12. *Desenăm mai întâi partea stângă a resortului* (*vezi* fig. 17):

```
DR 90 IN 40 SA 80 IN 30
```

Repetăm apoi de mai multe ori următoarea secvență - care realizează cîte o spirală:

```
DR 160 IN 60 SA 160 IN 60
```

13. `DR 90 IN 20 DR 80 IN 30 SA 160 IN 60 DR 160 IN 60`
și așa mai departe. Sfârșitul arcului va fi:
`SA 160 IN 30 DR 80 IN 20.`

14.

Triunghi	`REPETA 3[IN 40 SA 120]`
stea	`REPETA 3[IN 40 IP 40 SA 120]`
pentagon	`REPETA 5[IN 50 SA 72]`
al doilea arc	`DR 90 IN 20 DR 80 IN 30 REPETA 3[SA 160 IN 60 DR 160 IN 60] SA 160 IN 30 DR 80 IN 20`
hexagon	`REPETA 6[IN 50 SA 60]`
octogon	`REPETA 8[IN 50 SA 45]`
decagon	`REPETA 10[IN 40 SA 36]`

După desenarea fiecărei figuri geometrice regulate broasca ajunge cu orientarea sa în poziția în care era înainte de desenarea figurii, realizând, deci, o rotire completă (360 de grade). Unghiul de rotire se va afla de fiecare dată împărțind 360 la numărul de laturi (N) pe care îl are figura:

```
REPETA  N[IN 30 SA 360/N]
```

15. *Cerc. De exemplu:*

```
REPETA 36[IN 8 SA 10]
```

Când latura este mai mare.

16.
```
REPETA  5[IN 20 IP 20 SA 360/5]
REPETA  N[IN 20 IP 20 SA 360/N]
```

17. a)
```
REPETA  6[IN 5 SA 90 IN 5 DR 90]
```

b)
```
REPETA  6[IN 5 DR 90 IN 5 SA 90]
```

prima scară e pe stânga iar a doua pe dreapta.

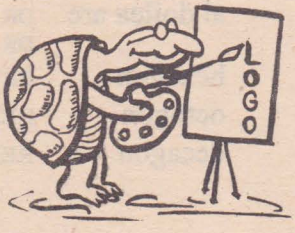
c) *crucea este o stea cu 4 raze:*

```
REPETA  4[IN 30 IP 30 SA 90]
```

d)
```
REPETA  4[IN 30 IP 5 SA 90 IN 5 IP 10 IN 5 DR 90 IP
25 SA 90]
```

18.
```
REPETA 4[IN 50 IN 15 REPETA 3 [IN 15 SA 120] IP 15
SA 90]
```

Se observă că, deși se repetă ceva, (sunt 4 laturi, 4 unghiuri și 4 triunghiuri), pentru realizarea elementului triunghi s-a folosit din nou repetiția. Deci, se pot scrie mai multe cicluri (repetiții) unul într-altul, conform figurii 87.



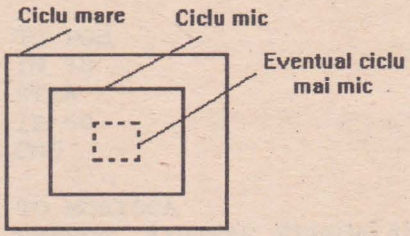


Fig. 87

Se observă că ciclurile nu se intersectează sau, cu alte cuvinte, dacă se deschide un ciclu mai mic într-unul mai mare, acesta din urmă va aştepta să se încheie ciclul cel mai mic şi abia apoi se va încheia şi el.

Se observă deasemenea că se pot folosi practic oricâte cicluri dorim. În acest caz primul se va încheia cel mai mic, adică cel deschis ultimul, iar apoi în ordine se vor încheia şi celelalte.

Aceste cicluri se mai numesc îmbricate şi, după cum puteţi observa din figură, ele seamănă cu foile care formează o ceapă.

Puteţi comenta asemănarea dar şi diferenţa dintre comanda REPETA *din* **LOGO** *şi instrucţiunea* FOR - NEXT *similară cu care se realizează ciclurile în* BASIC. *Observaţi că se folosesc aceleaşi reguli pentru ciclurile îmbricate.*

19.
```
TO PATRAT
REPETA 4[IN 40 SA 90]
END

TO PATRATD
REPETA  4[IN 40 DR 90]
END

TO TRIUNGHI
REPETA  3[IN 40 SA 120]
END

TO STEA
REPETA  3[IN 40 IP 40 SA 120]
END
```

```
TO CERC
REPETA  36[IN 8 SA 10]
END

TO PENTAGON
REPETA  5[IN 50 SA 72]
END

TO HEXAGON
REPETA  6[IN 50 SA 60]
END

TO OCTOGON
REPETA  8[IN 50 SA 45]
END

TO ARC
DR 90 IN 20  DR 80 IN 30
REPETA  3[SA 160 IN 60 DR 160 IN 60]
SA 160 IN 30 DR 80 IN 20
END
```

20.
```
STERGE PATRAT FC DR 90 IN 50 SA 90
CREION PATRAT FC ACASA CREION
```

21.
```
SA 45 PATRAT DR 45+30+90 PATRAT SA 30+90
```

La dreapta s-a mai adunat și 90 *deoarece ambele pătrate sunt pe stânga. Dacă al doilea pătrat se desenează pe dreapta, atunci se poate introduce:*

```
SA 45 PATRAT DR 75 PATRATD SA 30
```

22. *Se apelează editorul* **LOGO** *și se modifică procedurile. Pentru procedura* ARC *se obține:*

```
TO ARC
DR 90 IN 20 DR 85 IN 25
REPETA  3[SA 170 IN 50 DR 170 IN 50]
SA 170 IN 25 DR 85 IN 20
END
```

23.
```
TO STEAG
IN 60
PATRATD
IP 60
END
```

24.
```
TO POM
IN 50
STEA
IP 50
END
```

25.
```
TO MORISCA
REPETA  8[STEAG STINGA 45]
END

TO POMI
REPETA  6[POM STINGA 60]
END
```

26.
```
TO CRUCE                    TO MOARA
PATRAT                      PATRATD
STINGA 90 INAINTE 30        STINGA 90 INAINTE 30
END                         END
```

Se constată că la prima procedură se folosește pătratul pe stânga iar la a doua pătratul pe dreapta, în rest fiind identice.

27.
```
TO TRI
DREAPTA 30
REPETA  3[INAINTE 40 DREAPTA 120]
STINGA 30
END
```

28.
```
TO TRAP
DREAPTA 30 INAINTE 40 DREAPTA 60 INAINTE 40
DREAPTA 60 INAINTE 40 DREAPTA 120 INAINTE 80
DREAPTA 120 STINGA  30
END
```

29. *Cateta alăturată unghiului de* 60 *de grade va fi jumătate din ipotenuză deci* 30, *iar pentru cealaltă catetă aplicăm teorema lui Pitagora:*

$C2^2 = 60^2 - 30^2 = 3600-900 = 2700$

$C2=\sqrt{2700}$

Pentru calcularea radicalului se folosește funcția SQRT*(square root).*

Deci:

```
TO TRIDREP
INAINTE SQRT 2700 STINGA 150 INAINTE 60
STINGA 120 INAINTE 30 STINGA 90
END
```

30.
```
TO DREPT
REPETA  2[IN 20 SA 90 IN 30 SA 90]
END

TO MULTDREPT
REPETA  20[DREPT SA 360/20]
END

TO MULTRI
REPETA  20[TRIUNGHI SA 18]
END
```

31.
```
TO STEAGS                    TO STEAGD
SA 30                        DR 30
IN 30                        IN 30
REPETA 2[IN 20 SA 90         REPETA 2[IN 20 DR 90
IN 30 SA 90]                 IN 30 DR 90]
IP 30                        IP 30
DR 30                        SA 30
END                          END
```

Teorema simetriei în **LOGO**: *imaginea simetrică (pereche) a unei figuri se obține prin menținerea lungimilor și inversarea sensurilor de rotire a unghiurilor.*

32.
```
POTS
UITA  "DREPT
```

Nu se poate realiza `MULTDREPT` *deoarece broasca "nu mai știe" cum se desenează dreptunghiul. Acesta trebuie redefinit.*

33.
```
TO TRIUNGHI :LAT
REPETA  3[IN  :LAT SA 120]
END
```

```
TO STEA  :RAZA
REPETA  3[IN  :RAZA IP  :RAZA SA 120]
END
```

Bineînțeles se pot introduce și valori negative pentru mărimea laturii deoarece variabilelor li se pot atribui numere negative. De exemplu:
`TRIUNGHI - 30`
În acest caz, dacă figura geometrică era concepută pe stânga (SA 120) *ea se va desena pe dreapta, inversându-se și sensul de parcurgere. Pentru comparare încercați:*

```
TRIUNGHI 30 TRIUNGHI -30
```

34. *Pentru realizarea dreptunghiului cu laturile variabile avem nevoie de două variabile: una pentru lungime* (LU) *și alta pentru lățime* (LA)

```
TO DREPT  :LA  :LU
REPETA  2[IN  :LA SA 90 IN  :LU SA 90]
END
```

`DREPT 20 50` *va desena un dreptunghi cu lățimea de 20 și lungimea de 50.*

35.
```
TO MUTA  :X  :Y
FARACREION  DR 90 IN :X  SA 90 IN :Y
CREION
END
```

Și în acest caz, dacă se introduc valori negative, de exemplu

```
MUTA  -20  -30
```

broasca se va muta invers, adică la stânga și în jos.

36.
```
PUNE  "A 3
PUNE  "B 4
TO SCHIMBA
PUNE  "C  :A
PUNE  "A  :B
PUNE  "B  :C
SCRIE  :A
SCRIE  :B
END
```

Se folosește deci regula celor 3 pahare.
Se poate și generaliza o procedura SCHIMBA *cu doi parametrii care schimbă conținutul a două variabile:*

```
TO SCHIMBA  :V1  :V2
PUNE  "V3  :V1
PUNE  "V1  :V2
PUNE  "V2  :V3
SCRIE :V1 SCRIE :V2
END


SCHIMBA  3  4
```

va inversa conținutul celor 2 *variabile* V1 *și* V2, *afișându-se valorile în ordine inversă, adică* 4 *și apoi* 3. *Din păcate nu vom putea folosi valorile acestor variabile în afara procedurii create astfel, variabilele fiind locale.*

36.
```
TO NUMARA  :N
TEXTSCREEN
PUNE  "NUM  0
REPETA  :N [PUNE "NUM  :NUM + 1  SCRIE  :NUM]
END
```

Se observă că este necesară inițializarea unei variabile NUM *cu* 0 *pentru a porni de la acest punct. Apoi se va repeta de* N *ori adăugarea a câte unei unități la vechiul număr și afișarea noului număr obținut.*

Pentru a afișa numerele pare vom modifica pasul de numărare astfel:

```
REPETA  :N[PUNE  "NUM  :NUM + 2  SCRIE  :NUM]
```

Pentru a afișa numerele impare vom porni de la 1, *bineînțeles pasul de numărare rămânând* 2. *Procedura va fi:*

```
TO IMPARE  :N
TS PUNE  "NUM 1 SCRIE  :NUM
REPETA  :N[PUNE  "NUM  :NUM + 2  SCRIE  :NUM]
END
```

38.
```
TO MULTIPLI  :PAS  :N
TS PUNE "NUM :PAS  SCRIE  :PAS
REPETA  :N - 1[PUNE  "NUM :NUM + :PAS SCRIE  :NUM]
END
```

Procedura MULTIPLI *afișează primii* N *multipli ai lui* PAS.
De exemplu: MULTIPLI 17 10
Se afișează primii 10 *multipli ai lui* 17.

39.
```
TO SUMA  :N
PUNE  "S  0
PUNE  "C  0
REPETA  :N [PUNE  "C  :C + 1  PUNE  "S  :S + :C]
SCRIE  :S
END
```

Cu SUMA 10 *obținem* 50
Cu SUMA 100 *obținem* 5050
Cu SUMA 1000 *obținem* 500500

Numărul care reprezintă suma se formează adăugându-se câte un 0 *după fiecare cifră. Formula se verifică.*
De exemplu, pentru N=10, *aplicând formula obținem* 10•11/2=55;
pentru N=100 *obținem* 100•101:2=5050. *Deci, pentru orice* N *obținem același rezultat.*

40. *Se modifică în procedură linia în care contorul* C *va crește cu pasul* 2:
```
REPETA  :N [PUNE  "C  :C + 2  PUNE  "S  :S + :C]
```

După modificare, cu:
SUMA 10 *obținem* 110
SUMA 100 *obținem* 10100

Numărul care reprezintă suma numerelor pare se formează intercalându-se un 0 *între primele două cifre* 1 *și adăugându-se la sfârșit un* 0 *odată cu creșterea cu un ordin de mărime a numărului.*
Se observă că suma primelor N *numere pare este dublă față de suma primelor* N *numere naturale.*

Acest lucru este de așteptat deoarece în componența sumei primelor N *numere pare intră numerele care sunt dublu comparativ cu numerele naturale corespunzătoare care compun suma numerelor naturale. Acest fapt devine evident comparând, de exemplu, suma primelor* 10 *numre naturale cu suma primelor numere pare:*

$1+2+3+4+5+6+7+8+9+10=55$

$2+4+6+8+10+12+14+16+18+20=110$

Se observă că numerele din rândul al doilea fiind duble și suma va fi dublă. Știm că suma primelor N *numere naturale se calculează după formula:* S=N(N+1)/2

Rezultă că, pentru primele numere pare, suma fiind de două ori mai mare, formula va fi: S=N(N+1)

Pentru suma primelor N *numere impare procedura va fi:*

```
TO SUMAIMP  :N
PUNE "S  0
PUNE "C  1
REPEAT  :N[PUNE "S  :S + :C   PUNE  "C  :C + 2]
SCRIE  :SUMAIMP
END
```

Cum va fi această sumă față de suma numerelor pare ?
Să studiem sumele respective pentru primele 10 elemente:

$2+4+6+8+10+12+14+16+18+20=110$

$1+3+5+7+9+11+13+15+17+19=100$

Se observă că numerele care formează suma primelor N *numere impare sunt mai mici cu o unitate față de numerele corespunzătoare care compun cealaltă sumă. Întreaga sumă a numerelor impare va fi cu* N *mai mică decât suma numerelor pare. Folosind formulele obținem:*

$SUMAIMP = SUMAPARE - N = N(N+1) - N = N^2 + N - N = N^2$

$SUMAIMP = N^2$

De altfel acest lucru rezultă și uitându-ne la rezultatul obținut pentru suma primelor 10 *numere impare :* $100 = 10^2$.

41. *De data aceasta variabila care va conține produsul va trebui inițializată cu* 1 *(deoarece dacă se inițializează cu* 0 *orice produs va fi tot* 0*).*

```
TO PRODUS  :N
PUNE   "P   1
PUNE   "C   0
REPETA :N [PUNE  "C  :C  +  1    PUNE   "P   :C  *  :P]
SCRIE  :P
END
```

42. `SCRIE REST 1989 17`

43. `SCRIE REST 1999 53`

Se afișează numărul `38`*. Restul nefiind* `0` *înseamnă că* `1999` *nu este divizibil cu* `53`*.*

44.
```
SCRIE INT  DIV   256  45
SCRIE INT  DIV   175  45
SCRIE INT  PROD  3.24  DIV 4.23  5.707
SCRIE INT  DIV   -9   2
```

Obținem rezultatul -5*.*
Ultimul rezultat (-5) *se explică cu următoarea regulă: întregul reprezintă cel mai mare număr întreg mai mic decât numărul care reprezintă parametrul de intrare. Într-adevăr* -9 : 2 = - 4,5 *iar* -5 *este primul întreg mai mic decât* - 4,5*.*

45.
```
TO ZECIMAL2  :N
SCRIE  DIV  INT  PROD  :N  100  100
END
```

Se observă că se realizează mai întâi produsul N • 100*, apoi se aplică la rezultatul obținut funcția* `INTREG`*, iar apoi rezultatul obținut se împarte la* 100*. Practic se respectă formula:*

```
INT (N*100) / 100
```

Aplicând procedura cu `ZECIMAL2 15.3422` *obținem rezultatul* `15.34`

46.
```
TO SUMAPAT  :N
PUNE  "S   0
PUNE  "C   1
REPETA     :N[PUNE  "D  PROD  :C  :C   PUNE  "S  :S + :D
PUNE  "C   :C + 1]
SCRIE :S
END
```

Sau în loc de :

```
PUNE "D PROD :C :C
```

se poate pune:

```
PUNE "D :C * :C
```

Pentru suma inverselor numerelor se modifică linia astfel:

```
REPETA :N[PUNE "D DIV 1 :C PUNE  "S :S + :D  PUNE "C
:C+1]
```

47. *La suma numerelor naturale se adaugă pentru fiecare ordin de mărime câte un zero după cifra* 5. *Deci* 5050, 500500 *etc.*

La suma numerelor pare se adaugă pentru fiecare ordin de mărime câte un zero după cifra 1. *Numărul obținut este dublu față de suma primelor* N *numere naturale.*

La suma numerelor impare se adaugă pentru fiecare ordin de mărime câte doi de zero. Numărul obținut reprezintă pătratul lui N.

La suma pătratelor numerelor se adaugă cifra 3 *după primul* 3 *și după* 8 *precum și un* 0 *la sfârșit.*

Și așa mai departe.

N	SUMA	SUMAPAR	SUMAIMP	SUMAPAT	SUMACUB	SUMAINV
10	55	110	100	385	3025	2,928968
100	5050	10100	10000	338350	25502500	5,...
1000	500500	1001000	1000000	333833500		
10000						
.....						
	$\frac{N(N+1)}{2}$	N(N+1)	N^2	$\frac{N(N+1)(2N+1)}{4}$		

48.
```
REPETA 5[SCRIE RANDOM 101]
```

49.
```
TO ALEAINT  :A  :B
SCRIE  :A + RANDOM  SUM  :B  SUM 1  PROD -1  :A
END
```

Se observă că se aplică formula de calcul a unui număr aleator cuprins între A *și* B:

```
A + INT(RND * ((B-A) + 1))
```

50. *Procedura principală pentru umplerea ecranului cu pătrate* (UMPLE) *va avea ca parametru mărimea laturii pătratului,* L *și va conține:*

- *o procedură pentru aducerea broaștei de acasă în partea stângă sus a ecranului de unde va începe să deseneze pătrate de la stânga la dreapta pe primul rând* (AS)
- *calculul numărului de pătrate de pe un rând* NP *(de câte ori va intra latura plus cei 5 pixeli dintre pătrate în 256 care reprezintă lungimea ecranului grafic)*
- *calculul numărului de rînduri* NR *(de cîte ori va intra latura plus cei 5 pixeli dintre pătrate în 176 care reprezintă înălțimea ecranului grafic)*
- *un ciclu prin care se va repeta desenarea fiecărui rând și aducerea broaștei pe rândul următor. Acest ciclu va conține, la rândul său, un alt ciclu prin care se va repeta desenarea pătratului pe fiecare rând și saltul broaștei pentru trecerea la desenarea următorului pătrat*

Pentru desenarea pătratului se va folosi o procedură de desenare a unui pătrat pe dreapta cu latura variabilă (PATRATD).

Iată o versiune de program:

```
TO UMPLE :L
   AS
   PUNE "NP INT DIV 256 SUM :L 5
   PUNE "NR INT DIV 176 SUM :L 5
   REPETA :NR[REPETA :NP[PATRATD :L DR 90 FC IN :L + 5
   SA 90 CR] FC SA 90 IN :NP * (:L + 5) DR 90  IP 5 +
   :L  CR]
END
```

```
TO AS
   STERGE
   FC SA 90 IN INT DIV 256 2 DR 90
   IN SUM - :L INT DIV 175 2 CR
END

TO PATRATD :L
REPETA 4 [IN :L DR 90]
END
```

Acum, din comanda `UMPLE 30` *ecranul se va umple cu 35 de pătrate (5 rânduri a câte 7 pătrate) cu o distanță de 5 pixeli între ele.*

51.
```
TO FIGURA  :LAT  :TIP
DACA  :TIP = 1 [PATRAT  :LAT] [TRIUNGHI  :LAT]
END
```

52.
```
TO MAXIM  :A  :B
DACA :A >= :B [PUNE "MAX :A PUNE "MIN :B][PUNE  "MAX :B
PUNE "MIN :A]
OP :MAX
END
```

Exemplu de utilizare:

```
SCRIE MAX   5  -8
```

Se afișează 5.

53.
```
TO VALABS  :A
DACA :A > 0 [OP :A]
DACA :A = 0 [OP :A]
DACA :A < 0 [OP -1 * :A]
END
```

Exemplu de utilizare:

```
SCRIE VALABS -5
```

Se va afișa 5.

Sau

```
TO VALABS  :A
DACA :A <0 [OP -1 * :A] [OP :A]
END
```

Pentru a afla valoarea absolută a unui număr, procedura se va folosi cu `SCRIE`

De exemplu:

```
SCRIE VALABS -7
```

sau

```
SCRIE VALABS  -7 * 5
```

54.
```
TO MEDIA :A  :B
OP (:A  + :B) / 2
END
```

Utilizare:

```
SCRIE MEDIA  8  10
```

Se va afișa 9.

55.
```
TO MONEDA
PUNE  "A RANDOM 100
IF :A > 50 [SCRIE  "BANUL] [SCRIE "STEMA]
END
```

Exemplu de utilizare:

```
REPETA 100 [MONEDA]
```

Este echivalent cu aruncarea monedei de 100 *de ori.*

56.
```
TO DIVIZOR  :A  :B
PUNE  :C  REST  :B  :A
DACA  :C  =  0  [SCRIE "DIVIZOR]
END
```

57.
```
TO DREPTORIZ  :A  :B
DACA :A > :B [IN :B SA 90 IN :A SA 90 IN :B SA 90
IN :A SA 90] [IN :A SA 90 IN :B SA 90 IN :A SA 90
IN :B  SA  90]
END
```

58.
```
TO MODEPAT  :L  :P
PATRAT  :L
FC SA 90 IN :P DR 90 IN :P CREION
PUNE "L :L - 2 * :P
IF :L > 0 [MODEPAT  :L  :P]
END
```

Primul parametru (L) *indică mărimea laturii, iar al doilea* (P) *distanța dintre pătrate. Exemplu de utilizare:*

```
MODEPAT 60 5
```

59. *Modificări pentru:*

- *pătrat:* `SA 90`
- *triunghi:* `SA 120`
- *stea:* `SA 140`
- *stea ascuțită:* `SA 160`

Pentru `SA 180` *sau* `SA 360` *se obține o dreaptă.*

Experimetarea procedurii se face mai lesne prin modificarea ei considerând unghiul de rotire drept al doilea parametru al procedurii:

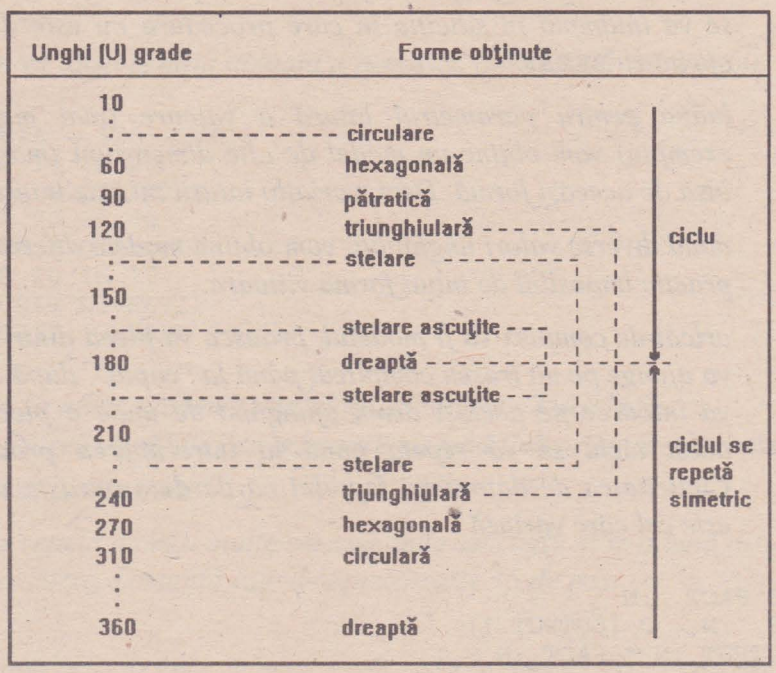
```
TO  SPI :L   :U
IF  :L > 60  [STOP]
IN  :L   SA  :U
SPI :L + 2   :U
END
```

Figurile solicitate se vor obține, în acest caz, direct prin:

```
SPI 3  90
SPI 3 120
SPI 3 140
SPI 3 160
SPI 3 180
```

Discuție asupra relației dintre unghiul de rotire și formele rezultate. Știm că unghiul de rotire reprezintă unghiul exterior figurii geometrice. Unghiul (interior) figurii este, de fapt, 180° - U.
Să pornim de la un unghi mic, să zicem, 10 grade. În acest caz, unghiul figurii este de 180° - 10° = 170°, *deci un unghi foarte mare. Spirala va avea forma unei figuri geometrice cu 36 de laturi, fiind practic un cerc. Apoi, crescând unghiul* (60°, 90°, 120°), *unghiul figurii se va ascuți (va scade), rezultând figuri geometrice cu un număr mai mic de laturi: hexagon, pătrat, triunghi.*

Mărind în continuare unghiul (deci scăzând pe cel al figurii geometrice) se obțin niște figuri care nu mai sunt închise (o figură închisă poate avea cel puțin 3 laturi). În acest caz se obțin forme stelare și, cu cât unghiul va crește, unghiul figurii va scade și formele stelare vor fi mai ascuțite. Pentru un unghi de 180 de grade se va obține o dreaptă, după care, mărind unghiul, ciclul se va repeta simetric, adică se vor obține forme stelare, apoi figuri geometrice, iar la 360 de grade, din nou o linie dreaptă.
Sintetizând rezultatele obținem următorul tabel:



Dându-se un unghi mai mare (de exemplu, 630 de grade) putem calcula cam ce formă vom obține?

Da. În cazul nostru, 630 este un număr multiplu de 90 (90 • 7). *Deoarece 7 este un număr impar, vom obține o figură pătratică.*

Dacă unghiul ar fi fost 720 (*adică* 90 • 8), *atunci* 8 *fiind un număr par, am obține o dreaptă.*

60. *Procedura are 3 parametri: latura, unghiul de rotire și un al treilea parametru care reprezintă creșterea.*
Procedura nu folosește o regulă de oprire, fiind însă recursivă. Prin apelul recursiv se va obține o nouă formă (model) cu un unghi nou (la vechiul unghi se adaugă creșterea).
Utilizarea procedurii ridică următoarele probleme:

- *procedura nefolosind regula de oprire, unghiul va crește continuu, ceea ce va duce la oprirea ei forțată în momentul în care mărimea unghiului va fi un număr prea mare. Acest fapt se va întâmpla în situația în care procedura nu este dinainte oprită cu* **BREAK**
- *luând pentru parametrul latură o valoare (mai mare, de exemplu) vom obține un model de alte dimensiuni (mai mare) însă de aceeași formă. Deci, variația laturii nu este interesantă*
- *dând diverse valori unghiului, vom obține modele diferite, fiind practic imposibil de intuit forma viitoare*
- *oricât de complex va fi modelul, broasca va pleca dintr-un loc, va ajunge pe un traseu complicat până la "capăt", după care se va întoarce pe același drum ajungând de unde a plecat, iar acest ciclu se va repeta până la intreruperea procedurii. Ciclicitatea de datorează faptului că de data aceasta unghiul este cel care variază*

61.
```
TO FACT  :N
IF  :N = 0 [OUTPUT 1]
OUTPUT :N * FACT :N - 1
END
```
Exemplu de utilizare:
```
?SCRIE  FACT 5
 120
```

62.
```
MUTA    20  0
MUTA   -20  0
MUTA   125  0
MUTA  -125  0
MUTA  -125  0
MUTA   125  0
```
și așa mai departe

```
TO MUTA  :X  :Y
FIXDIR 90
FC IN :X  SA 90  IN :Y
END
```

63. 0; 0; 0,0

64.
```
FEREASTRA DR 30 IN 50 SCRIE XCOR
SCRIE YCOR
SCRIE POZITIE
IN 200
SCRIE POZITIE
```

Se va afișa poziția actuală a broaștei, adică, 125 *și* 216,50.

65.
```
MUTA - 125 -216.5
DR 30 IN 20
SCRIE DIRECTIE
DR 60 IN 30
SCRIE DIRECTIE
DR 20 IN 10
SCRIE DIRECTIE
SCRIE TOWARDS [0 0]
```

Se va afișa mărimea unghiului, și anume, 254,27656.
Afișarea unghiului de orientare nu modifică orientarea broaștei.

```
FIXDIR TOWARDS [0 0]
```

Se repetă de mai multe ori comanda INAINTE *și se adună distanțele parcurse. Distanța este de aproximativ* 45 *de pași.*

66.
```
PUNE "A 32.26  SCRIE  :A  SCRIE NREL  :A  SCRIE PRIMUL
:A
```

Se va afișa numărul din variabila A, *adică* 32,26 *apoi numărul de elemente* (5) *și, în sfârșit, primul element al numărului, care este cifra* 3.

Se observă că se numără toate caracterele inclusiv punctul zecimal.

```
PUNE  "B [3 5 7] SCRIE  :B  SCRIE NREL :B  SCRIE PRIMUL
:B
PUNE  :C  "MERE  SCRIE  :C SCRIE  :NREL :C SCRIE PRIMUL
:C
PUNE  "D [VACANTA DE VARA LA MARE] SCRIE :D SCRIE NREL
:D      SCRIE PRIMUL :D
```

În ultimul caz numărul de elemente este 5, *elementele din listă fiind cuvinte. Primul element va fi cuvântul* "VACANTA".

```
PUNE  "E  FU  :D
SCRIE PRIMUL  FP  FP  :E
```

Al treilea element se obține ca primul element din listă după ce au fost eliminate primele două. Se obține "VARA".

67. LOGO

Evaluarea în **LOGO** *se face astfel:* WORD *așteaptă ca intrare două obiecte, deci ce urmează va fi primul obiect. Dar* FU *este o primitivă deci sistemul va merge mai departe până la ultima primitivă, care este* FP. *Acum această primitivă se poate aplica primului obiect care constă dintr-o listă formată din două liste ca obiecte. Apoi evaluarea se face invers, adică se aplică* FP *deci se înlătură primul element din listă, rămânând o listă formată dintr-un singur obiect care este o listă. Se aplică în continuare procedura* PRIMUL. *Va rezultă primul element care este lista* [MULT FOLOSITA DE GOSPODINE]. *Se aplică* FP *și se elimină cuvântul* MULT *apoi se ia primul element din ce a rămas și va rezulta cuvântul* FOLOSITA. *La acest cuvânt se aplică de două ori* FP, *eliminându-se primele două caractere și de patru ori* FU, *rămânând cuvântul* LO. *Pentru al doilea cuvânt se va proceda analog rezultând cuvântul* GO.

Cele două cuvinte se vor "alipi" prin primitiva WORD *formând un singur cuvânt,* **LOGO**

68.
```
TO ALDOILEA  :LUCRU
OP PRIMUL FARAPRIMUL  :LUCRU
END
```

Exemplu de utilizare:

vrem să extragem al doilea element din lista [1 2 3]

```
SCRIE ALDOILEA  [1 2 3]
2
```

Dacă vom aplica procedura unui cuvânt, se va afișa al doilea caracter (literă) din cuvânt. Exemplu:

```
SCRIE ALDOILEA "MARE
```

Se afișează litera A.

69.
```
PUNE  "L  CITLIST
PUNE  "C  CITCAR PUNE  "D CITCAR  PUNE "E CITCAR
PUNE  "F  WORD WORD  :C  :D  :E
PUNE  "L  FRAZA  :F  :L
PUNE  "L  FRAZA  :L  :F
SCRIE :L
```

70. *Se folosește recursivitatea.*

```
TO MISCARE   :CUVINT
SCRIE    :CUVINT
MISCARE  FP  :CUVINT
END
```

De exemplu:

```
MISCARE "VACANTA
```

se va obține următoarea piramidă:

```
VACANTA
ACANTA
CANTA
ANTA
NTA
TA
A
```

71.
```
TO NRPAT
PUNE  "N  CITCAR
SCRIE  :N * :N
NRPAT
END
```

Se observă că, în acest fel, se afișează pătratele doar pentru cifre (CITCAR *citește un singur caracter - cifră*).

72.
```
TO DESEN
PUNE  "Z  CITCAR
IF :Z = 5  [SA 90]
IF :Z = 6  [IP 10]
IF :Z = 7  [IN 10]
IF :Z = 8  [DR 90]
IF :Z = "S [STOP]
DESEN
END
```

73.
```
PUNE  "X  CITLIST
PRINT  SUM PRIMUL  :X  7
```

CITLIST *va genera o listă de caractere formată dintr-un singur element (care la rândul său este format din caractere cifre). De exemplu* [1234] *este o listă formată din elementul* 1234. *De aceea pentru a se face o operație aritmetică a fost nevoie de funcția* PRIMUL *pentru a extrage acest element din listă.*

74. *Cercurile se vor trasa, după o procedură în care se folosește binecunoscuta formulă de trasare a unei figuri geometrice regulate cu mai multe laturi. Procedura pentru cerc va avea doi parametrii, unul pentru numărul laturilor și altul pentru mărimea laturii:*

```
TO CERC  :N  :L
REPETA  :N [IN  :L SA 360 / :N]
END
```

După fiecare cerc se face SA 90 *și* PUNE "LD 0 *și se măsoară diametrul (prin trasarea sa) prin intermediul procedurii* MAM, *apoi, afișându-se mărimea diametrului* (LD) *și rezultatul împărțirii lungimii cercului, care se calculează prin înmulțirea numărului de laturi* (N) *cu mărimea laturii* (L), *la mărimea obținută a diametrului* (LD) - *vezi fig.* 88.

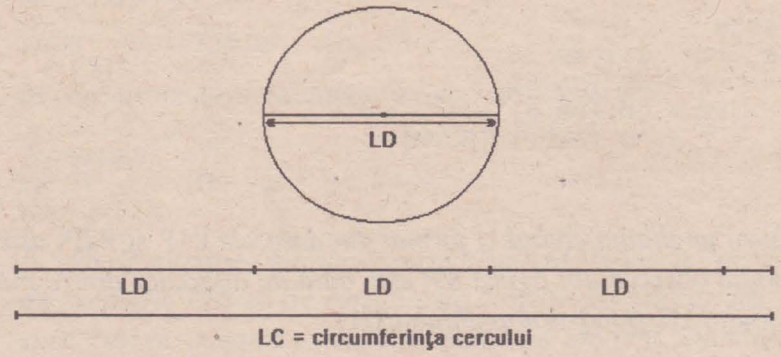


Fig. 88

```
CERC 20 7
SA 90  PUNE  "LD  0
MAM
S
SCRIE  :LD
SCRIE  20 * 7 / :LD
```

Iată unele rezultate obținute prin experimentări.

Pentru un cerc cu 20 *de laturi a câte* 7 *pixeli :*

```
CERC  20  7: l=140, d=43, l/d=3,25
CERC  20  5: l=100, d=31, l/d=3,2258065
CERC  20  8: l=160, d=51, l/d=3,1372549
CERC  40  5: l=200, d=63, l/d=3,17
CERC 100  3: l=300, d=94, l/d=3,1914894
```

Se constată că obținem valori apropiate ale câtului. De fapt, am experimentat faptul că raportul dintre lungimea și diametrul oricărul cerc este o constantă egală cu 3,141941.. *care se mai numește și* Π . *Din experiment mai rezultă că mărimea cercului va crește dacă crește numărul laturilor sau dacă crește mărimea laturii și, deasemenea, cele mai bune aproximații ale lui se obțin pentru cercuri ceva mai mari și cu mai multe laturi pentru ca precizia măsurării diametrului să fie mai mare.*

75.
```
TO ROTM
PUNE   "T   CITCAR
IF     :T = "S [SA 5]
IF     :T = "D [DR 5]
IF     :T = "G [STOP] [ROTM]
END
```

76. *Măsura unghiului căutat* U *format din dreptele* P1P *și* P2P, *așa cum se poate observa din figura* 89, *este dată de diferența dintre măsura unghiului* U2 *și măsura unghiului* U1.

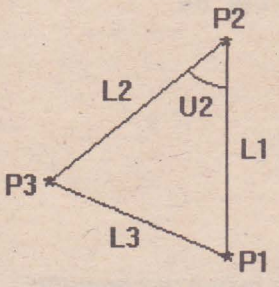


Fig. 89

Știm că funcția `TOWARDS :P` *ne furnizează unghiul de orientare ("direcția") pe care ar trebui să-l aibe broasca spre a fi îndreptată către punctul ale cărui coordonate sunt memorate în variabila* P.
Mai știm că unghiul de orientare se măsoară de la Nord spre Est (vezi fig 2). *Făcând legătura cu funcția* `TOWARDS`, *ne dăm seama că măsura unghiului căutat este dată tocmai de diferența dintre măsurile unghiurilor de orientare către* P2 *și către* P1, *problema rezolvîndu-se prin:*

```
PUNE  "U (TOWARDS  :P2) - TOWARDS  :P1
```

Deoarece se poate întâmpla ca punctele P1 *și* P2 *să fie așezate "invers" (în acest caz se obține aceeași valoare dar cu semnul minus), după instrucțiunea de mai sus, va trebui să punem o comandă de înlocuire a lui* U *cu valoarea sa absolută.*

De exemplu:

```
PUNE  "U  VALABS  :U
```

în care procedura VALABS *este (vezi problema* 53):

```
TO VALABS  :U
IF :U < 0 [OP -1 * :U] [OP :U]
END
```

În sfârșit, se poate întâmpla ca punctele să fie foarte "prost" plasate; de exemplu, PP1 *să fie spre Est de direcția* PN *(nord), iar* PP2 *să fie spre Vest de această direcție. În acest caz unghiul obținut prin calcul va fi mai mare de 180 de grade; noi considerăm, însă, că unghiul celor două direcții este celălalt, care nu depășește 180 de grade.*

Cele două instrucțiuni de mai sus trebuie, deci, să fie urmate de:

```
IF  :U > 180 [PUNE  "U  360 - :U]
```

Procedura UNG *va fi deci:*

```
TO UNG :P1  :P2
PUNE  "U (TOWARDS  :P2) - TOWARDS  :P1
PUNE  "U VALABS  :U
IF  :U > 180 [PUNE  "U  360 - :U]
OP :U
END
```

De exemplu, dorim să afișăm măsura unghiului determinat de punctele de coordonate P1[10 0] *și* P2 [0 30] *și origine. Evident că acest unghi este de* 90 *de grade. Într-adevăr cu* SCRIE UNG [10 0] [0 30] *se va afișa* 90. *Cu* SCRIE UNG [-27 -50][-40 0] *vom obține* 61, 63095.

Dacă mutăm broasca, să zicem cu IN 10 *atunci cu comanda* SCRIE UNG [10 0] [0 30] *vom obține un unghi de* 135 *de grade, acesta creându-se corespunzător odată cu deplasarea broaștei în alt punct.*

77. *Trebuie să se memoreze toate vârfurile triunghiului și să se poziționeze broasca în vârfurile triunghiului (cu* FIXPOZ*). Fiind într-un vârf, prin apelarea corespunzătoare a procedurii* UNG *va conduce la aflarea mărimii unghiului corespunzător vârfului respectiv (vezi fig.* 89).

```
TO LUL :L1 :L2 :L3
PUNE :P1 POZITIE
IN   :L1
PUNE "P2 POZITIE
SA   180 - :U2
IN   :L2
PUNE "P3 POZITIE
FIXDIR TOWARDS :P1
PUNE :LD 0 MAM PUNE "L3 :LD
SCRIE [LATURILE:] ASTEAPTA 200
SCRIE FRAZA FARA :L1 :L2 :L3

TO LUL :L1 :U2 :L2
PUNE  "P1  POZITIE
IN  :L1
PUNE  "P2  POZITIE
SA  180 - :U2
IN  :L2
PUNE  "P3  POZITIE
FIXDIR TOWARDS  :P1     PUNE  "LD  0 MAM PUNE  "L3  :LD
SCRIE [LATURILE SINT:] ASTEAPTA 200
SCRIE FRAZA :L1 :L2 :L3 ASTEAPTA 300
SCRIE [UNGHIURILE SINT:] ASTEAPTA 200
SCRIE :U2 ASTEAPTA 200
FIXPOZ :P3  UNG  :P2 :P1 ASTEAPTA 200
FIXPOZ :P1  UNG  :P2 :P3
END
```

ASTEAPTA (*sau* WAIT *în engleză*) *provoacă oprirea activității calculatorului o perioadă de timp. De exemplu* ASTEAPTA 100 *calculatorul se oprește circa 2 secunde. Aceasta are scopul de a putea să citim valorile obținute deoarece în regimul grafic doar ultimele două linii din partea de jos a ecranului sînt folosite pentru text.*

78. *Din figură se observă că ducându-se paralele la axele Ox și Oy se obține un triunghi dreptunghic. Distanța* d *va fi în acest caz chiar ipotenuza triunghiului. Această distanță o putem afla prin teorema lui Pitagora cunoscând mărimile catetelor: cateta orizontală este tocmai diferența absciselor, iar cateta verticală este diferența ordonatelor.*

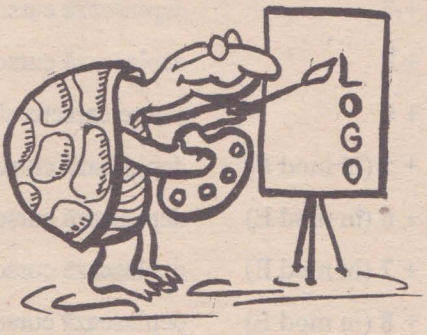
Procedura va fi:

```
TO DIST  :P1  :P2
PUNE  "DX (PRIMUL  :P2) - PRIMUL  :P1
PUNE  "DY (ULTIMUL  :P2) - ULTIMUL  :P1
OP SQRT (:DX * :DX + :DY * DY)
END
```

Exemplu de utilizare: `SCRIE DIST [10 10] [40 50]`
obținem rezultatul (distanța) 50 *deoarece este ipotenuza unui triunghi dreptunghic cu catetele de* 30 *și* 40.

```
SCRIE DIST [-20 30] [40 -50]
```

obținem distanța 100.



➢ *Notă:*
Formele prescurtate sunt trecute în paranteze. Parametrii de intrare sunt trecuți imediat după procedura descrisă și în paranteze atunci când sunt opționali. Dacă procedura prezintă și forma echivalentă în limba română, aceasta se va trece imediat sub ea, având în paranteză (dacă este cazul) și forma prescurtată. În cadrul fiecărui capitol, comenzile și operațiile sunt descrise în ordine alfabetică.

Editorul LOGO

Pentru modificarea procedurilor definite sau chiar pentru definirea noilor proceduri, se folosește editorul **LOGO**.

Apelare: `EDIT (ED)` numele procedurii

Utilizare editor:

CS + 5	deplasează cursorul cu un caracter la stânga.
CS + 6	deplasează cursorul cu o linie mai jos.
CS + 7	deplasează cursorul cu o linie mai sus.
CS + 8	deplasează cursorul cu un caracter la dreapta.
CS + 0	șterge caracterul dinaintea cursorului.
CS + 5 (în mod E)	deplasează cursorul la începutul liniei.
CS + 6 (în mod E)	deplasează cursorul la sfârșitul ecranului.
CS + 7 (în mod E)	deplasează cursorul la începutul ecranului.
CS + 8 (în mod E)	deplasează cursorul la sfârșitul liniei .

B (în mod E)	deplasează cursorul la începutul textului.
E (în mod E)	deplasează cursorul la sfârșitul textului.
N (în mod E)	deplasează cursorul la pagina următoare.
P (în mod E)	deplasează cursorul la pagina precedentă.
Y (în mod E)	șterge de pe ecran linia pe care se află cursorul.
R (în mod E)	reinserează linia care a fost ștearsă cu comanda Y (în mod E).
C (în mod E)	părăsirea editorului cu încorporarea modificărilor efectuate și intrarea în **LOGO.**
CS + SPACE	părăsirea editorului cu ignorarea modificărilor efectuate în **LOGO.**

unde CS=CAPS SHIFT

COMENZI ȘI OPERAȚII

0. Broasca țestoasă

➢ *Notă:*

Broasca țestoasă este numele dat micului triunghi care apare pe ecran când se utilizează ecranul grafic. Mișcându-se dintr-un punct în altul, broasca va lăsa urme, putându-se trasa în acest mod linii pe ecran. Ecranul grafic, numit și câmpul broaștei țestoase, apare oricând se introduce o comandă referitoare la mișcarea broaștei. În mod normal, ecranul grafic are 256 de pași (puncte) pe orizontală și 175 de pași (puncte) pe verticală, iar originea este în centrul ecranului.

BACK n (**BK** n) **INAPOI** n (**IP** n)	Deplasarea înapoi cu n pași (pixeli).
BACKGROUND (BG)	Furnizează un număr (operație) care reprezintă culoarea fondului. Numărul furnizat poate fi: 0 - dacă fondul are culoarea neagră; 1 - albastră; 2 - roșie; 3 - violetă; 4 - verde; 5 - bleu; 6 - galbenă; 7 - albă.
CLEAN **CLEARSCREEN (CS)** **STERGE**	Șterge ecranul fără aducerea broaștei acasă.
DOT [X Y] **PUNCT** [X Y]	Desenează un punct la poziția specificată prin coordonatele X și Y. Broasca nu se mișcă de la locul ei și nu se trasează nici o linie.
FENCE **GARD**	Limitează mișcările broaștei în cadrul ecranului grafic.
FORWARD n (**FD** n) **INAINTE** n (**IN** n)	Deplasare înainte cu n pași.
HEADING **DIRECTIE**	Furnizează un număr (operație) între 0 și 359 care reprezintă orientarea actuală a broaștei în ecran. 0 - direcție N; 90 - Est; 180 - Sud; 270 - Vest.
HIDETURTLE (HT) **FARABROASCA (FB)**	Ascunde broasca (face broasca invizibilă).
HOME **ACASA**	Readucerea broaștei acasă.

LEFT n (**LT** n) **STINGA** n (**SA** n)	Rotire stânga cu n grade fără ca broasca să-și schimbe poziția.
PENCOLOUR (**PC**) **CULOARE**	Furnizează un număr (operație) care reprezintă culoarea cu care va desena broasca. La început culoarea va fi 0 (neagră).
PENDOWN (**PD**) **CREION** (**CR**)	Penița jos : când broasca se va mișca, ea va trasa o linie.
PENERASE (**PE**) **GUMA** (**GU**)	Ștergerea liniei : broasca va șterge orice linie peste care va trece. Comenzile PENDOWN, PENUP sau PENREVERSE vor anula efectul comenzii PENERASE.
PENREVERSE (**PX**) **CI**	Penița inversă : broasca în mișcare va trasa linii, dar va șterge liniile peste care va trece (desenează cu culoarea inversă).
PENUP (**PU**) **FARACREION** (**FC**)	Penița sus : broasca în mișcare nu va trasa linii.
POSITION (**POS**) **POZITIE**	Furnizează o listă de două numere (operație) care reprezintă poziția actuală a broaștei.
RIGHT n (**RT** n) **DREAPTA** n (**DR** n)	Rotire spre dreapta cu N grade fără ca broasca să-și schimbe poziția.
SCRUNCH **RELXY**	Furnizează o listă de două numere (operație), care reprezintă proporția dintre mărimea unui pas pe verticală și a unuia pe orizontală.
SETBG n	Fixează culoarea fondului (0 - negru, 1 - albastru etc.).
SERBORDER n (**SETBR** n) **FIXCHENAR** n	Fixează culoarea chenarului (borderului) (0 - negru, 1 - albastru etc.).

SETHEADING n **(SETH** n**)** **FIXDIR** n	Rotește broasca, astfel încât orientarea ei să fie n. La început, orientarea broaștei este 0.
SETPC n **FIXCULOARE** n	Fixează culoarea cu care se va desena (0 - negru, 1 - albastru etc.).
SETPOS [X Y] **FIXPOZ** [X Y]	Deplasează broasca în punctul de coordonate X și Y fără a-i schimba orientarea (efectuează o translație până în punctul respectiv).
SETSCRUNCH [X Y] **SETSCR** [X Y]	Fixează proporția dintre mărimea pasului pe verticală și mărimea pasului pe orizontală. În mod normal (la început) această proporție este 100.
SETX n **FIXX** n	Deplasează broasca într-un punct a cărui coordonată pe orizontală (X) este n.
SETY n **FIXY** n	Deplasează broasca într-un punct a cărui coordonată pe verticală (Y) este n.
SHOWNP	Operație care furnizează cuvântul TRUE dacă broasca este vizibilă (este în modul SHOWTURTLE) sau FALSE dacă nu este.
SHOWTURTLE (ST) **BROASCA**	Arată broasca (face broasca vizibilă).
TOWARDS [X Y] **CAP** [X Y]	Furnizează orientarea pe care trebuie să o capete broasca pentru a fi îndreptată spre punctul de coordonate X și Y (operație); nu modifică orientarea broaștei.

WINDOW **FEREASTRA**	Permite broaștei să se deplaseze în afara ecranului grafic. Când broasca este în modul WINDOW, va continua să execute comenzile de deplasare chiar fără a fi văzută.
WRAP **IESEREVINE**	Permite broaștei ca atunci când iese din ecran să apară din partea opusă a ecranului.
XCOR	Furnizează coordonata pe orizontală (abscisa) actuală a broaștei (operație).
YCOR	Furnizează coordonata pe verticală (ordonata) actuală a broaștei (operație).

❷ Cuvinte și liste

ASCII caracter	Operație care furnizează codul ASCII al unui caracter.
BUTFIRST obiect **BT** obiect **FARAPRIMUL** obiect **FP** obiect	Operație care furnizează obiectul fără primul element al obiectului.
BUTLAST obiect **BL** obiect **FARAULTIMUL** obiect **FU** obiect	Operație care furnizează obiectul fără ultimul element al obiectului.

CHAR n **CARACTER** n	Operație care furnizează caracterul al cărui cod ASCII este n. Codul n trebuie să fie un întreg cuprins între 32 și 165.
COUNT obiect **NREL** obiect	Operație care furnizează numărul de elemente ale obiectului (cuvânt sau listă).
EMPTYP obiect **GOL** obiect	Operație care furnizează valoarea logică TRUE dacă obiectul **LOGO** este gol și valoarea logică FALSE în caz contrar.
EQUALP obiect1 obiect2 **EGAL** obiect1 obiect2	Operație care furnizează valoarea logică TRUE dacă obiect1 și obiect2 sunt două numere egale, două cuvinte sau liste identice.
FIRST obiect **PRIMUL** obiect	Operație care furnizează primul element al obiectului. Acesta va fi un caracter dacă obiectul este un cuvânt ori un cuvânt sau o listă dacă obiectul este o listă.
FPUT obiect listă **PUNEPRIMUL** obiect listă	Operație care furnizează o nouă listă, formată prin punerea (alipirea) obiectului la începutul listei (FirstPUT).
ITEM n obiect	Operație care furnizează al n-lea element al unui obiect.
LAST obiect **ULTIMUL** obiect	Operație care furnizează ultimul element al obiectului. Acesta va fi un caracter dacă obiectul este un cuvânt sau o listă dacă obiectul este o listă.

LIST obiect1, obiect 2 (**LIST** obiect1, obiect2, ... , obiectn) **LISTA** obiect1, obiect 2 (**LISTA** obiect1, obiect 2,...,obiectn)	Operație care furnizează o listă ale cărei elemente sunt obiect1, obiect2 etc.
LISTP obiect	Operație care furnizează valoarea logică TRUE dacă obiectul este o listă și FALSE în caz contrar. O listă vidă este considerată un cuvânt gol.
LPUT obiect listă **PUNEULTIMUL** obiect listă	Operație care furnizează o nouă listă, formată prin punerea (alipirea) obiectului la sfârșitul listei (LastPUT).
MEMBERP	Operație care furnizează valoarea logică TRUE dacă obiectul este un element al listei și FALSE în caz contrar.
NUMBERP obiect	Operație care furnizează valoarea logică TRUE dacă obiectul este un număr și FALSE în caz contrar.
SENTENCE obiect1, obiect2 (**SE** obiect1, obiect2, ... , obiectn) **FRAZA** obiect1, obiect2 (**FRAZA** obiect1,obiect2,...,obiectn)	Operație cu care se formează o listă compusă din obiectele date.
WORD cuvânt1, cuvânt2 (**WORD** cuvânt1,cuvânt2,..., cuvântn)	Operație cu care se formează un cuvânt din obiectele date.
WORDP obiect	Operație care furnizează valoarea logică TRUE dacă obiectul este un cuvânt și FALSE în caz contrar.

❸ Variabile

MAKE nume obiect **PUNE** nume obiect	Atribuie valoarea "obiect" variabilei "nume" (pune în locația de memorie numită "nume" valoarea "obiect").
NAMEP obiect	Operație care furnizează valoarea logică TRUE dacă obiectul are o valoare și FALSE în caz contrar.
THING nume	Operație care furnizează conținutul unui nume.

❹ Operații aritmetice

➤ *Notă:*
Adunarea, scăderea, înmulțirea și împărțirea pot fi utilizate în forma infix, în care operatorii (+ - * /) *se pun între parametrii de intrare.*

Adunarea, înmulțirea și împărțirea pot fi utilizate și în forma prefix, dar în acest caz, SUM, PRODUCT, *respectiv* DIV, *pot fi urmate de doi parametri de intrare. În* **LOGO** *se pot efectua operații aritmetice atât cu numere întregi cât și cu numere zecimale.*

În operațiile aritmetice (cu semne) se recomandă ca semnul să fie încadrat de spații, altfel **LOGO** *poate interpreta că unul dintre subiecte este un număr negativ, de exemplu în cazul scăderii.*

ARCCOS n	Furnizează valoarea în grade a arccosinusului de n.
ARCCOT n	Furnizează valoarea în grade a arccotangentei de n.

`ARCSIN n`	Furnizează valoarea în grade a arcsinusului de n.
`ARCTAN n`	Furnizează valoarea în grade a arctangentei de n.
`COSINE n (COS n)`	Furnizează valoarea în grade a cosinusului de n.
`COTANGENT n (COT n)`	Furnizează valoarea în grade a cotangentei de n.
`DIV a b`	Furnizează câtul obținut prin împărțirea lui a prin b.
`INT n`	Furnizează partea INTreagă a lui n.

`PRODUCT a b`
`(PRODUCT a b ... n)`
`PROD a b`
`(PROD a b ... n)`
	Furnizează produsul subiectelor (intrărilor). Este echivalentă cu operația de înmulțire simbolizată prin semnul *.

`RANDOM n`	Furnizează un număr aleator întreg, cuprins între 0 și (n-1).
`REMAINDER [a b]` `REST [a b]`	Furnizează restul împărțirii lui a prin b.
`ROUND n`	Furnizează n rotunjit la cel mai apropiat întreg.
`SINE n`	Furnizează valoarea în grade a sinusului de n.
`SQRT n`	Furnizează rădăcina pătrată din n; n trebuie să fie pozitiv.

`SUM a b`	Furnizează suma subiectelor (intrărilor). Este echivalentă cu operația de adunare, simbolizată prin semnul +.
`TANGENT n`	Furnizează valoarea în grade a tangentei de n.

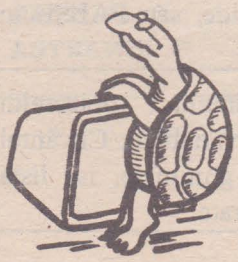
❺ Definiri și editări de proceduri

`EDIT` procedură `ED` procedură	Intrarea în modul de lucru de editare a procedurii.
`EDNS` nume `EDNS` nume lista	Permite editarea numelor și a valorilor lor. Fără subiect se vor lista numai variabilele numite, cu valorile lor.
`END`	Indică sfârșitul definirii unei proceduri.
`TO` nume	Permite începerea definirii unei proceduri.

❻ Condiții de control

`BYE`	Ieșire din **LOGO** cu trecerea controlului în interpretorul BASIC. Se poate intra din nou în **LOGO** prin comanda RUN.

IF condiție [lista1 de comenzi] [lista2 de comenzi] **DACA** condiție [lista1 de comenzi] [lista2 de comenzi]	Dacă este îndeplinită condiția, se execută lista 1 de instrucțiuni, dacă nu, se execută lista 2.
IF condiție [listă de comenzi]	Lista este executată doar dacă este îndeplinită condiția; în caz contrar lista nu se execută și se trece la linia **LOGO** care urmează.
OUTPUT obiect **OP** obiect	Ieșire dintr-o procedură cu furnizarea unui rezultat, care se atribuie ca valoare numelui procedurii. Notă: o procedură se transformă în "funcție" dacă se termină cu instrucțiunea de "ieșire" OUTPUT.
REPEAT n [listă de comenzi] **REPETA** n [listă de comenzi]	Repetă de n ori comenzile din listă. Notă: n trebuie să fie pozitiv.
RUN [listă de comenzi]	Se execută lista ca o linie **LOGO**.
STOP	Oprește execuția procedurii curente și redă controlul procedurii care a apelat-o.
TOPLEVEL	Oprește execuția procedurii curente și redă controlul lui **LOGO** (apare promptul "?"). Este echivalentă cu acționarea tastelor CS și SPACE.



❼ Operații logice

➢ *Notă:*
Parametrii de intrare pentru primitivele care reprezintă operații logice pot fi TRUE *sau* FALSE *sau alte condiții. Termenul de predicat (pred) este folosit pentru a descrie o funcție ai cărei parametri de intrare pot fi* TRUE *sau* FALSE *sau condiții și care furnizează ca rezultat* TRUE *sau* FALSE, *cum este cazul operațiilor logice.*

AND pred1 pred2 **(AND** pred1 pred2 .. predn) **SI** pred1 pred2 **(SI** pred1 pred2 .. predn)	Furnizează TRUE dacă toți parametrii de intrare sunt TRUE și FALSE în caz contrar.
NOT pred **NU** pred	Furnizează TRUE dacă predicatul (parametrul de intrare) este FALSE și FALSE dacă predicatul este TRUE.
OR pred1 pred2 **(OR** pred1 pred2 .. predn)	Furnizează TRUE dacă cel puțin unul din predicate este TRUE și FALSE dacă nici unul din predicate nu este TRUE.

❽ Extragere rezultate

KEYP	Furnizează TRUE dacă este acționată o anumită tastă ori o combinație de anumite taste valide, sau FALSE în caz contrar.
PRINT obiect**(PR** obiect) **SCRIE** obiect	Afișează conținutul obiectului (cuvânt, listă sau variabilă). Cuvântul se va afișă fără ghilimele, iar lista fără paranteze pătrate.

READCHAR (RC) **CITCAR**	Comandă calculatorului să aștepte apăsarea unei taste și apoi furnizează caracterul introdus, care va fi preluat de o procedură, ca orice rezultat al unei operații.
READLIST (RL) **CITLIST**	Furnizează lista dată de utilizator la tastatură. Întreaga linie, introdusă înainte de acționarea tastei CR, este considerat o listă. În urma acționării tastelor, caracterele corespunzătoare sunt afișate pe ecran.
SHOW obiect	Afișează cuvântul, lista sau numerele date ca parametri de intrare. Listele sunt afișate cu paranteze pătrate.
SOUND [durată înălțime] **SUNET** [durată înălțime]	Produce un sunet de o anumită durată și de o anumită înălțime. Durata sunetului se dă în secunde (de la 0 la 255), iar înălțimea în semitonuri. Do-ul central este reprezentat prin valoarea 0, apoi sunetele cu înălțime mai mare prin numere întregi pozitive iar sunetele cu înălțime mai mică prin numere întregi negative. Al doilea parametru trebuie să fie cuprins între -62 și 75.
TYPE obiect (**TYPE** obiect1, obiect2 ...obiectn)	Afișează obiectele. Spre deosebire de PRINT și SHOW, nu provoacă linefeed după afișare.
WAIT n **ASTEAPTA** n	Așteaptă n/50 secunde.

❾ Ecranul

➢ *Notă:*
După încărcarea lui **LOGO**, *ecranul se află în modul text, putându-se afișa* 22 *linii de text. În modul grafic (în care se intră automat după orice comandă grafică), cele* 22 *de linii pot fi folosite pentru grafică, iar alte două linii din partea de jos a ecranului, pentru mesaje.*
Pe fiecare linie se pot afișa 32 *de caractere. Pe prima coloană în modul comandă și în modul de definire a unei proceduri este un prompt (care indică așteptarea introducerii unei comenzi), iar ultima coloană este rezervată pentru semnul exclamării, care indică o linie* **LOGO** *neterminată, mai lungă de* 32 *de caractere.*

`BRIGHT n`	Stabilește dacă afișarea va fi cu luminozitate (n = 1) sau nu (n = 0)
`CLEARTEXT (CT)` `STERGETEXT`	Șterge tot textul de pe ecran. În modul grafic, această comandă va șterge textul de pe ultimele două linii ale ecranului.
`COPYSCREEN`	Realizează o copie a ecranului la imprimantă (hard copy).
`CURSOR`	Furnizează o listă formată din două numere; primul va reprezenta coloana, al doilea linia pe care se găsește cursorul.
`FLASH`	Afișarea se va face pe un fond "clipitor" până la întâlnirea comenzii NORMAL.
`INVERSE`	Afișarea se va face inversându-se culoarea pentru text și desene cu cea a fondului până la întâlnirea comenzii NORMAL.

`NORMAL`	Afișarea se va face fără "clipire" și fără inversarea culorilor.
`OVER n`	Afișarea se va face peste orice desen sau text (supratipărire) în cazul lui OVER 1. Cu OVER 0 se revine în modul obișnuit.

`SETCURSOR [a b]`
`SETCUR [a b]`
`FIXCURSOR [a b]`
	Mutarea cursorului în poziția indicată de cei doi parametri din listă. Primul parametru va reprezenta coloana, iar al doilea, linia. Coloanele sunt numerotate de la 0 la 30, iar liniile de la 0 la 21.

`SETTC [a b]`	Specifică culoarea fondului și a cernelii la afișarea unui text. Cei doi parametri pot lua valorile pentru culori (0 - 7).
`TEXTCOLOUR (TC)`	Furnizează o listă de doi parametri, din care primul va specifica culoarea fondului și al doilea culoarea cernelii la afișarea unui text.
`TEXTSCREEN (TS)`	Trecerea din modul grafic al ecranului în modul text (întreg ecranul pentru text).

⑩ Spațiul de lucru

➤ *Notă:*
O parte din memoria inițial disponibilă fiind ocupată de interpretorul **LOGO**, *partea de memorie care rămâne la dispoziția utilizatorului se numește spațiu de lucru.*

ERALL	Șterge tot ce a fost creat (proceduri și variabile) în spațiul de lucru (ERases ALL).

➤ *Notă:*
Folosirea acestei comenzi va avea ca efect și ștergerea tuturor procedurilor în limba română, acestea fiind și ele memorate în spațiul de lucru. Comanda ERALL *nu afecrează conținutul curent al editorului. Pentru a șterge și editorul se folosește* ERALL *și* EDIT.

ERASE nume(**ER** nume) **UITA** nume	Șterge procedura sau procedurile (din listă) din spațiul de lucru.
ERN nume	Șterge variabila (variabilele) numite din spațiul de lucru (ERases Named).
ERNS	Șterge numele și valorile atribuite tuturor variabilelor din spațiul de lucru (ERases NameS).
ERPS	Șterge toate procedurile din spațiul de lucru (ERases all the ProcedureS).
PO nume	Afișează definiția procedurii sau procedurilor din listă (Prints Out).
POALL	Afișează numele și definițiile tuturor procedurilor, precum și valorile tuturor variabilelor din spațiul de lucru (Print Out ALL).
PONS	Afișează numele și valorile tuturor variabilelor din spațiul de lucru (Prints Out the NameS).
POPS	Afișează definițiile tuturor procedurilor curente din spațiul de lucru (Prints Out ProcedureS).

POTS	Afișează numele tuturor procedurilor definite (Prints Out the TitleS).

11 Salvări și încărcări de fișiere

➢ *Notă:*

Toate procedurile definite în timpul unei sesiuni de lucru sunt memorate de către **LOGO** *în spațiul de lucru. Toate realizările (proceduri, desene etc.) pot fi salvate oricând pe caseta magnetică și refolosite, la nevoie, prin încărcarea lor în memoria calculatorului de pe același suport magnetic.*

Realizările pot fi aranjate în fișiere pe trei tipuri:

- *fișierul de proceduri* **LOGO** *(tipul* LOG*) este un fișier care conține proceduri* **LOGO**
- *fișierul editor (tipul* TXT*) este reprezentat prin conținutul curent al editorului* **LOGO**
- *fișierul ecran (tipul* SCR*) este reprezentat prin imaginea ecran curentă*

LOAD "nume fișier tip fișier	Încarcă fișierul nume fișier de pe casetă în spațiul de lucru. Dacă extensia care precizează tipul fișierului lipsește, atunci se consideră implicit că tipul este LOG. În timpul încărcării sunt afișate pe ecran numele procedurilor din fișier, iar după încărcare, promptul **LOGO** (?) va reapărea pe ecran.
LOAD " nume fișier	Încarcă fișierul editor salvat cu SAVED "nume fișier și îl face conținutul curent al editorului.
LOADSCR " nume fișier	Încarcă fișierul ecran salvat cu SAVESCR "nume fișier și afișează conținutul său (imaginea) pe ecran.

SAVE " nume fișier nume	Salvează procedurile din lista de proceduri sub forma unui fișier cu numele nume fișier. Numele dat fișierului poate avea până la 7 caractere.
SAVEALL " nume fișier	Salvează tot ce există în spațiul de lucru (proceduri și variabile) sub numele nume fișier.
SAVED " nume fișier	Salvează tot ce există în Editorul **LOGO** sub numele nume fișier.
SAVESCR " nume fișier	Salvează imaginea ecran sub numele nume fișier.

12 Funcții și primitive avansate

➤ *Notă:*

Unele primitive permit ca procedurile să fie definite și mânuite din interiorul altor proceduri.

Unele primitive afectează însuși sistemul **LOGO**. *Acestea se pot utiliza în vederea citirii conținutului memoriei. Se recomandă însă salvarea în prealabil a spațiului de memorie, acesta putându-se altfel distruge (în mod accidental). În general, numele acestor primitive începe cu un punct.*

NODES	Furnizează numărul de noduri libere sau, cu alte cuvinte, spațiul de memorie disponibil în spațiul de lucru pentru proceduri, variabile și rulări de proceduri. Un nod ocupă 5 octeți. Prin utilizarea lui NODES, imediat după RECYCLE, se furnizează numărul de noduri care sunt încă libere.
RECYCLE	Eliberează maximum posibil de noduri prin procedeul "colectarea gunoiului".
COPYDEF "nume nou nume	Copiază definiția unui nume de procedură existentă sub un alt nume (nou). Procedura existentă nu este ștearsă.
.CONTENTS	Furnizează o listă care include toate procedurile, variabile etc. din sistemul **LOGO**.
.PRIMITIVES	Listează toate primitivele **LOGO**.

SUMAR DE CUVINTE LOGO PENTRU CALCULATOARELE COMPATIBILE IBM PC

➤ *Notă:*

Acest sumar prezintă limbajul **LOGO** *pentru calculatoare compatibile* IBM PC *care funcționează cu grafică* CGA.

Încărcarea programului se face obișnuit din sistemul de operare DOS *cu comanda* ***logo.***

Se recomandă realizarea procedurilor în cadrul editorului **LOGO**. *Acesta se apelează cu comanda* EDIT *și va memora toate procedurile realizate cu el.*

Ieșirea din Editor ca și întreruperea unei proceduri se face cu ESC.

În cadrul unui capitol cuvintele **LOGO** *sunt trecute în ordine alfabetică.*

❶ Broasca țestoasă

BACK (BK) n **INAPOI (IP) n**	Deplasare înapoi cu n pași (pixeli).
BACKGROUND (BG) **FOND?**	Operație care furnizează un număr ce reprezintă culoarea fondului .
CLEAN **CURATA**	Șterge ecranul fără a aduce broasca acasă.
CLEARSCREEN(CS) **STERGE**	Șterge ecranul și aduce brosca acasă.
DOT [X Y] **PUNCT [X Y]**	Desenează un punct cu culoarea curentă a creionului. Broasca nu se mișcă de la locul ei și nu trasează nici o linie.
FENCE **GARD**	Limitează mișcările broaștei în cadrul ecranului grafic.
FILL **UMPLE**	Umple o zonă închisă cu culoarea curentă a creionului.
FORWARD (FD) n **INAINTE (IN) n**	Deplasare înainte cu n pasi.
HEADING **DIRECTIE?**	Furnizează un număr între 0 și 359 reprezentând orientarea actuală a broaștei în ecran. 0 = nord, 90 = est, 180 = sus, 270 = vest.
HIDETURTLE (HT) **FARABROASCA (FB)**	Ascunde broasca (face broasca invizibilă).

HOME **ACASA**	Readuce broasca acasă.
LEFT (LT) n **STINGA (SA)** n	Rotire stângă în sensul acelor de ceasornic, cu n grade fără ca broasca să-și schimbe poziția.
LOADPIC fișier **INCARCADESEN** fișier	Încarcă imaginea ecran dintr-un fișier.
PALETTE (PAL) **PALETA?**	Furnizează numărul paletei de culori.
PENCOLOUR (PC) **CULOARE?**	Furnizează un număr ce reprezintă culoarea creionului; implicit e neagră.
PEN **CREION?**	Furnizează o listă cu starea creionului (PD, PU, PE, PX).
PENDOWN (PD) **CREIONJOS (CJOS)**	Pune creionul jos; când broasca se va mișca, ea va trasa o linie.
PENERASE (PE) **GUMA (GU)**	Șterge linii; broasca va șterge orice linie peste care va trece. Comenzile PENDOWN, PENUP sau PENREVERSE vor anula efectul comenzii PENERASE.
PENREVERSE (PX) **CREIONINVERS (CINV)**	Pune creionul invers; broasca în mișcare va trasa linii dar va șterge liniile peste care va trece (desenează cu culoarea inversă).
PENUP (PU) **FARACREION (FC)**	Pune creionul sus; broasca în mișcare nu va trasa linii.
POS **POZITIE?**	Furnizează poziția actuală a broaștei (în coordonate).
RIGHT (RT) n **DREAPTA (DR)** n	Rotire dreapta cu n grade fără ca broasca să-și schimbe poziția.
SAVEPIC fișier **SALVAREDESEN** fișier	Salvează imaginea ecran în fișierul specificat.
RELXY	Proporția dintre mărimea unui pas pe verticală și mărimea unui pas pe orizontală.
SETBG n **FOND** n	Fixează culoarea fondului.
SETHEADING (SETH) n **DIRECTIE** n	Rotește broasca astfel încât orientarea ei să fie n. La început orientarea ei este 0.
SETPAL n **PALETA** n	Fixează culoarea paletei.
SETPC n **CULOARE** n	Fixează culoarea creionului.

SETPEN lista **CREION** lista	Fixează modul, culoarea și paleta creionului.
SETPOS [X Y] **POZITIE** [X Y]	Deplasează broasca în punctul de coordonate specificat fără a-i schimba orientarea.
SCARA n	Specifică raportul dintre mărimea pasului pe verticală și orizontală.
SETSHAPE n **FORMA** n	Schimbă forma broaștei.
SETX n **FIXX** n	Deplasează broasca într-un punct de abscisă n.
SETY n **FIXY** n	Deplasează broasca într-un punct de ordonată n.
SHAPE **FORMA?**	Furnizează forma actuală a broaștei.
SHOWNP **BROASCA?**	Furnizează TRUE (ADEVARAT) dacă broasca este vizibilă.
SHOWTURTLE (ST) **CUBROASCA**	Arată broasca.
SNAP n **COPIEZB** n	Copiază imaginea de sub broască și atribuie n formei respective.
STAMP **IMPRIMB**	Imprimă imaginea broaștei pe ecran.
TOWARDS [X Y] **SPRE** [X Y]	Furnizează orientarea pe care trebuie să o capete broasca pentru a fi îndreptată spre punctul de coordonate X și Y.
WINDOW **FEREASTRA**	Permite broaștei să se deplaseze în afara ecranului grafic. Când broasca este în modul WINDOW ea va continua să execute comenzile de deplasare chiar fără a fi văzută.
XCOR	Furnizează abscisa punctului în care se află broasca.
YCOR	Furnizează ordonata punctului în care se află broasca.

❷ Cuvinte și liste

ASCII caracter	Furnizează codul ASCII al unui caracter.
BUTFIRST (BF) obiect **FARAPRIMUL (FP)** obiect	Furnizează obiectul fără primul element al obiectului.
BUTLAST (BL) obiect **FARAULTIMUL (FU)** obiect	Furnizează obiectul fără ultimul element al obiectului.
CHAR n **CARACTER** n	Furnizează caracterul al carui cod ASCII este n.
COUNT obiect **NREL** obiect	Furnizează numărul de elemente al obiectului (cuvânt sau listă).
EMPTYP obiect **GOL** obiect	Furnizează valoarea logică TRUE (ADEVARAT) dacă obiectul este gol și FALSE (FALS) dacă nu.
EQUALP obiect1 obiect2 **EGAL** obiect1 obiect2	Furnizează valoarea logică TRUE (ADEVARAT) dacă cele două obiecte sunt identice și FALSE (FALS) dacă nu.
FIRST obiect **PRIMUL** obiect	Furnizează primul element al obiectului. Acesta va fi un caracter dacă obiectul este un cuvânt sau o listă dacă obiectul este o listă.
FPUT obiect lista **PUNEPRIMUL** obiect lista	Furnizează o nouă listă formată prin punerea obiectului la începutul listei.
ITEM n obiect **ELEMENT** n obiect	Furnizează al n-lea element al obiectului.
LAST obiect **ULTIMUL** obiect	Furnizează ultimul element al obiectului.
LIST obiect **LISTA** obiect	Furnizează o listă ale carei elemente sunt elementele obiectului.
LISTP obiect **LISTA?** obiect	Furnizează valoarea logică TRUE (ADEVARAT) dacă este o listă și FALSE (FALS) în caz contrar.
LPUT obiect lista **PUNEULTIMUL** obiect lista	Furnizează o nouă listă formată prin punerea (alipirea) obiectului la sfârșitul listei (LastPUT).
MEMBERP obiect1 obiect2 **MEMBRU?** obiect1 obiect2	Furnizează o valoare logica TRUE dacă primul obiect este un element al celui de-al doilea obiect și FALSE în caz contrar.

NUMBERP obiect **NUMAR?**	Furnizează valoarea logică TRUE dacă obiectul este un număr și FALSE în caz contrar.
SENTENCE (SE) obiect1 obiect2 **FRAZA** obiect1 obiect2	Furnizează o listă compusă din obiectele date ca parametrii de intrare.
WORD cuvânt1 cuvânt2 **CUVINT** cuvânt1 cuvânt2	Furnizează un cuvânt format din obiectele date ca parametri de intrare.
WORDP obiect **CUVINT?** obiect	Furnizează valoarea logică TRUE dacă obiectul este un cuvânt și FALSE în caz contrar.

❸ Text și ecran

CAPS **LITEREMARI?**	Furnizează TRUE dacă tasta CAPS LOCK este activată.
CLEARTEXT (CT) **STERGETEXT**	Șterge porțiunea de text de pe ecran.
CURSOR **CURSOR?**	Furnizeaza poziția cursorului de pe ecran.
FULLSCREEN (FS) **TOTECRANUL**	Afectează întregul ecran în mod grafic; are același efect ca și tasta F4.
MIXEDSCREEN (MS) **ECRANCOMBINAT**	Permite afișare de text și grafică pe ecran; are același efect ca și F2.
SCREEN **ECRAN?**	Furnizează 1 sau 2 în funcție de care ecran este în lucru.
SETCAPS predicat **LITEREMARI** predicat	Dacă predicatul are valoarea de adevărat se activează tasta de CAPS LOCK, altfel se dezactivează.
SETCURSOR poziție **POZCURSOR** poziție	Pune cursorul pe ecran in poziția specificată.
SETTC lista de culori **CULOARETEXT** lista de culori	Fixează culorile textului: (culoare litere, culoare fond).
SETTEXT n **MARIMETEXT** n	Fixează numărul de linii de text de pe ecranul grafic.
SETWIDTH n (a) **LATIMEECRAN** n (a)	Fixează mărimea ecranului la n coloane; al doilea parametru mută textul cu a coloane la stânga sau la dreapta.

TEXTCOLOR (TC) **CULOARETEXT?**	Furnizează o listă de numere pentru culorile curente ale textului.
TEXTSCREEN (TS) **MODTEXT**	Determină ca întreg ecranul să fie în mod text; are același efect ca și tasta F1.
WIDTH **LATIMEECRAN?**	Furnizează lățimea curentă a ecranului.

❹ Control execuție comenzi

CATCH nume lista **PRINDE** nume lista	Execută lista; se reîntoarce atunci când se execută comanda THROW nume (TRANSFERA nume).
CO (obiect) **REZUMA** (obiect)	Rezumă (termină) o procedură după o pauză; furnizează obiect la comanda PAUSE (PAUZA).
GO cuvânt **DUTELA** cuvânt	Transferă controlul la LABEL (ETICHETA) "cuvânt".
IF predicat lista1 (lista2) **DACA** predicat lista1 (lista2)	Dacă "predicat" are valoarea de adevărat se execută "lista1"; altfel se execută "lista2".
IFFALSE (IFF) lista **DACAFALS** lista	Execută "lista" dacă cel mai recent TEST a avut valoarea de adevărat.
LABEL cuvânt **ETICHETA** cuvânt	Creează o linie etichetată pentru a fi adresată de GO (DUTELA).
OUTPUT (OP) obiect **IESIRE** obiect	Reîntoarce controlul la procedura apelantă cu "obiect" la ieșire.
PAUSE **PAUZA**	Suspendă execuția procedurii.
REPEAT n lista **REPETA** n lista	Execută "lista" de n ori.
RUN lista **EXECUTA** lista	Execută "lista"; are ieșirea pe care o furnizează "lista".
STOP	Oprește procedura și reîntoarce controlul la procedura apelantă.
TEST predicat	Determină dacă "predicat" este adevărat sau fals.
THROW nume **TRANSFERA** nume	Transferă controlul la comanda CATCH (PRINDE) corespunzătoare.

❺ Variabile

EDNS (pachet (lista))	Apelează editorul **LOGO** (EDit NameS); pachetul (lista) conține variabile.
LOCAL nume	Nume este variabila locală.
MAKE nume obiect **PUNE** nume obiect	Atribuie valoarea lui "obiect" variabilei "nume".
NAME nume obiect **NUME** nume obiect	Face ca "nume" să aibă ca valoare "obiect".
NAMEP nume **NUME?** nume	Furnizează valoarea logică TRUE dacă "nume" are o valoare și FALSE în caz contrar.
THING nume **VALOARE** nume	Furnizeaza conținutul (valoarea) lui "nume".

❻ Operații aritmetice

ARCTAN y (x)	Furnizează arctangenta lui y în grade.
COS a	Furnizează cosinus de a în grade.
DIFFERENCE a b **DIFERENTA** a b	Furnizeaza a - b.
EFORM n a **FORMAEXP** n a	Furnizează n în notația științifică cu a digiți.
EXP a	Furnizează e la puterea a.
FORM n a (b) **FORMANUMAR** n a (b)	Furnizează numărul n cu a digiți la partea întreagă și b digiți la partea zecimală.
INT n	Furnizează partea întreagă a lui n.
LN a	Furnizează logaritm natural de a.
PI	Furnizează valoarea lui pi (3.1415...).
POWER n a **PUTERE** n a	Furnizează n la puterea a.
PRECISION **PRECIZIE?**	Furnizează numărul de digiți semnificativi la care este rotunjit un număr când este utilizat de **LOGO.**

PRODUCT a b **PRODUS** a b	Furnizează produsul dintre a și b.
QUOTIENT a b **CIT** a b	Furnizează câtul dintre a și b.
RANDOM n **ALEATOR** n	Furnizează un număr aleator nenegativ mai mic decât n.
REMAINDER a b **REST** a b	Furnizează restul împărțirii lui a la b.
RERANDOM **REALEATOR**	Face RANDOM reproductibil.
ROUND n **ROTUNJIRE** n	Furnizează n rotunjit la cel mai apropiat întreg.
SETPRECISION n **PRECIZIE** n	Fixează precizia curentă a numerelor la n.
SIN a	Furnizează sinus de a în grade.
SQRT n **RADACINAPATRATA** n	Furnizează rădăcina pătrată a lui n.
SUM a b **SUMA** a b	Furnizează suma dintre a și b.

❼ Administrarea spațiului de lucru

BURY pachet **INGHEATA** pachet	Ascunde procedurile și numele variabilelor conținute în pachet.
CONTINUT?	Cuvinte din spațiul de lucru.
ERALL pachet (lista) **STERGETOT** pachet (lista)	Șterge tot ce a fost creat (proceduri și variabile).
ERASE (ER) nume (lista) **STERGEPROC** nume (lista)	Șterge procedura sau procedurile din spațiul de lucru.
ERN nume (lista) **STERGEVAR** nume (lista)	Șterge variabila (variabilele) specificate.
ERNS pachet (lista) **STERGEVARP** pachet (lista)	Șterge numele și valorile atribuite tuturor variabilelor conținute în pachet (lista).
ERPS pachet (lista) **STERGEPROCP** pachet (lista)	Șterge toate procedurile conținute în pachet (lista).

LOAD specificator de fișier **INCARCA** specificator de fișier	Încarcă fișierul în spațiul de lucru (pachet).
NODES **NODURI?**	Furnizează numărul de noduri libere.
PACKAGE pachet nume (lista) **IMAPCHETARE** pachet nume (lista)	Pune procedurile specificate în pachet.
PKGALL pachet **IMPACHTOT** pachet	Pune în pachet tot ce nu este deja împachetat.
PO nume **AFPROC** nume	Afișează definiția procedurii.
POALL **AFTOT**	Afișează numele și definițiile tuturor procedurilor.
PONS **AFVAR**	Afișează numele și valorile tuturor variabilelor.
POPS **AFPROCP**	Afișează definițiile tuturor procedurilor curente.
POTS **AFTP**	Afișează liniile de titlu ale tuturor procedurilor definite.
RECYCLE **STERGEGUNOI**	Realizează curățirea spațiului de lucru.
REPARSE	Reanalizează procedurile.
SAVE dispozitiv/specificator de fișier (pachet (lista)) **SALVARE** dispozitiv/specificator de fișier (pachet (lista))	Salvare pe disc.
UNBURY pachet	Dezgheață procedurile din pachet.

8 Operații logice

AND predicat1 predicat2 **SI** predicat1 predicat2	Furnizează valoarea logică TRUE dacă toate intrările sunt adevarate.
NOT predicat **NU** predicat	Furnizează valoarea logică TRUE dacă predicatul este FALSE și valoarea logică FALSE dacă predicatul este TRUE.
OR predicat1 predicat2 **SAU** predicat1 predicat2	Furnizează valoarea TRUE dacă oricare din intrări este adevărată.

⑨ Comunicarea cu dispozitivele periferice

ALLOPEN **DISPDESCHISE?**	Furnizează lista dispozitivelor/fișierelor care sunt curent deschise.
BUTTONP n	Furnizează TRUE dacă este apăsat butonul de pe manetă sau joystick.
CLOSE dispozitiv/specificator de fișier **INCHIDE** dispozitiv/specificator de fișier	Închide un dispozitiv/fișier curent deschis.
CLOSEALL **INCHIDETOT**	Închide toate dispozitivele și fișierele curent deschise.
DIR ((unitate:)specificator de fișier)	Afișează numele fișierelor.
DISK **DISC?**	Furnizează unitatea de floppy disc curentă.
.DOS	Redă controlul sistemului de operare.
DRIBBLE dispozitiv/specificator de fișier **COPTEXT** dispozitiv/specificator de fișier	Începe procesul de trimitere a unei copii a textului de pe ecran la dispozitivul/fișierul specificat.
EDITFILE specificator fișier de intrare (specificator fișier de ieșire) **EDITFISIER** specificator fișier de intrare (specificator fișier de ieșire)	Editor LOGO având un fișier de intrare și unul de ieșire.
ERASEFILE specificator de fișier **STERGE** specificator de fișier	Șterge fișierul de pe dischetă.
FILELEN specificator de fișier **LUNGIMEFISIER** specificator de fișier	Furnizează lungimea fișierului în număr de octeți.
FILEP specificator de fișier **FISIER?** specificator de fișier	Furnizează TRUE dacă fișierul există.
KEYP **TASTA?**	Furnizează TRUE dacă a fost apăsată o tastă dar încă nu a fost citită.
NODRIBBLE **NUCOPTEXT**	Inhibă procesul de copiere și închide fișierele de copiere.
OPEN dispozitiv / specificator de fișier **DESCHID** dispozitiv / specificator de fișier	Deschide ce se specifică în argument.
PADDLE n	Furnizează la ieșire rotația manetei n sau poziția joystick-ului n.
POFILE specificator de fișier **AFFISIER** specificator de fișier	Afișează conținutul fișierului.

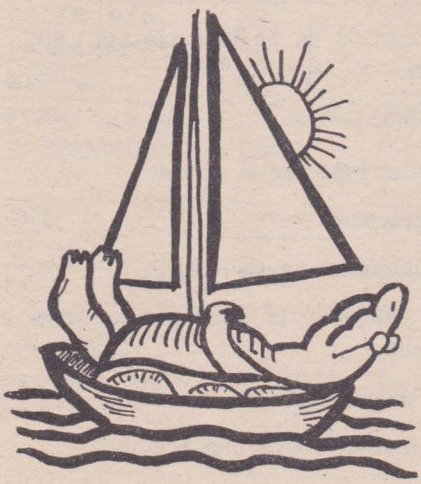
PRINT (PR) obiect **AFIS (AF)** obiect	Afișează obiectul urmat de retur de car și avans linie dar nu afișează și parantezele pentru liste.
READCHAR (RC) **CITCARACTER (CITC)**	Furnizează caracterul citit de fișierul curent sau dispozitivul curent (implicit tastatura); așteaptă, dacă este necesar.
READCHARS (RCS) n **CITCARACTERE (CITCE)** n	Furnizează n caractere citite de fișierul curent sau dispozitivul curent (implicit tastatura); așteaptă dacă este necesar.
READEOFP **SFCITFIS**	Furnizează TRUE dacă poziția fișierului deschis curent la citire este la sfârșitul fișierului.
READER **DISPCITIRE?**	Furnizează numele dispozitivului/ specificatorului de fișier deschis la citire.
READLIST (RL) **CITLINIE**	Furnizează linia citită de fișierul curent sau dispozitivul deschis la citire.
READPOS **POZCITIRE?**	Furnizează poziția la citire a fișierului de citire curent.
READWORD (RW) **CITCUVINT**	Furnizează primul cuvânt citit de fișierul sau dispozitivul curent (implicit tastatura); așteaptă dacă este necesar.
SETCOM n viteză paritate biți date biți stop **COMSER** n viteză paritate biți date biți stop	Setează linia de comunicație serială.
SETDISK unitate: **DISC** unitate:	Setează numărul unității de disc curente.
SETREAD dispozitiv/specificator de fișier **DISPCITIRE** dispozitiv/specificator de fișier	Setează dispozitivul/specificatorul de fișier de la care se citește cu comenzile de citire.
SETREADPOS n **POZCITIRE** n	Fixează poziția pentru citire în fișierul curent.
SETWRITE dispozitiv/specificator de fișier **DISPSCRIERE** dispozitiv/specificator de fișier	Fixează destinația intrărilor din comenzile de afișare.
SETWRITEPOS n **POZSCRIERE** n	Fixează poziția de scriere în fișierul curent.

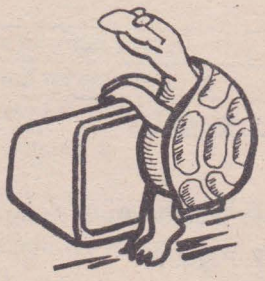
SHOW obiect **ARATA** obiect	Afișeaza obiectul, însoțit de paranteze, retur de car și avans de linie.
TONE frecvența durata **SUNET** frecvența durata	Produce un sunet cu frecvența și durata specificate.
TYPE obiect	Afișează fără paranteze.
WAIT n **ASTEPT** n	Determină o pauza de aproximativ n*1/18 secunde.
WRITEOFP **TERMINATSCRIERE?**	Furnizează TRUE dacă scrierea a ajuns la sfârșitul fișierului.
WRITEPOS **POZSCRIERE?**	Furnizează poziția de scriere în fișierul de scriere curent.
WRITER **DISPSCRIERE?**	Furnizează numele dispozitivului/ fișierului de scriere curent.

⑩ Taste speciale

backspace		Șterge caracterul din dreapta cursorului.
cursor stânga	←	Mută cursorul cu o poziție la stânga.
cursor dreapta	→	Mută cursorul cu o poziție la dreapta.
cursor sus	↑	Mută cursorul cu o poziție în sus.
cursor jos	↓	Mută cursorul cu o poziție în jos.
enter		Retur de car.
CTRL ←		Șterge toate caracterele de pe linia curentă din dreapta cursorului.
CTRL →		Inserează ultima linie ștearsă.
CTRL-Break		1. În afara editorului întrerupe și oprește execuția unei proceduri. 2. În interiorul editorului oprește editarea.
CTRL-NumLock		Întrerupe orice execuție; la acționarea oricărei taste rezumă execuția.
CTRL-PageDown		Mută cursorul la sfârșitul bufferului de editare.
CTRL-PageUp		Mută cursorul la începutul bufferului de editare.

Del	Şterge caracterul de pe poziţia pe care se află cursorul.
End	Mută cursorul la sfârşitul paginii curente a editorului.
Esc	Părăseşte editorul, citind bufferul ca şi cum ar fi fost tastat.
F1	În afara editorului corespunde la TEXTSCREEN.
F2	În afara editorului corespunde la MIXEDSCREEN
F3	Insereaza o copie a ultimei linii tastate.
F4	În afara editorului corespunde la FULLSCREEN.
F5	În afara editorului corespunde la PAUSE.
Home	Mută cursorul la începutul paginii curente din editor.
Ins	Inserează o nouă linie în poziţia curentă a cursorului în editor.
PageDown	Defilează ecranul la următoarea pagină de editor.
PageUp	Defilează ecranul la pagina anterioară de editor.
Shift PrtScr	Afişează ecranul la imprimantă; va tipari în mod grafic dacă se ataşează o imprimantă grafică.





BIBLIOGRAFIE

- Abelson H. di Sessa A. — **Turtle Geometry: The Computers as a medium for exploring mathematics**, MIT Press, Cambridge 1981
- Călinescu C. Diamandi I. — **Pagina de informatică**, "Start spre viitor" nr. 4 - 12 1986, nr. 1 - 8 1987
- Călinescu C. Diamandi I. Mironov A. — **În universul calculatoarelor**, revista "Luminița" nr. 4 - 12 1986, nr. 1 - 12 1987, nr. 3 - 8 1988
- Diamandi I. — **Curs de inițiere în LOGO**, "Știință și tehnică" nr. 1 - 6 1986
- Diamandi I. — **LOGO - univers de cunoaștere a ideilor avansate**, simpozionul "Aplicații ale inteligenței artificiale", Academia Română, București 1987
- Diamandi I. Călinescu C. — **Dialog cu viitorul** (pag. 58 - 81), Editura Științifică și Enciclopedică, București 1988
- Diamandi I. Călinescu C. — **De ce LOGO ?**, Almanahul părinților, 1987
- Diamandi I. Nicolescu R. Pascu A. — **Între stilul BASIC și stilul LOGO**, Tribuna Școlii nr. 314/13 august 1988
- Diamandi I. Constantinescu C. — **Jocuri pe calculator - LOGO**, RECOOP, București 1989
- Diamandi I. Odăgescu I. — **Din spectacolul informaticii - calculatorul personal** (pag. 100 - 128), Editura Militară, București 1991
- Diamandi I. — **Introducere în LOGO**, Gazeta de informatică, nr. 3 - 9, Editura Libris

- Diamandi I. Vass Gh. — **Învățământul general și învățarea științifică prin LOGO**, Simpozion LOGO de la Fribourg, oct.1990
- Diamandi I. Vass Gh. — **Utilizarea limbajului LOGO în sistemul de educație și învățământ în România**, Revista de Pedagogie nr. 6, 1990
- Diamandi I. Vass Gh. — **LOGO, o nouă metodă de a învăța cu ajutorul calculatorului**, Editura Pacific, București 1991
- Filimonov R. Sendov B. — **Drawing LOGO closer to the curriculum**, Educational Computer Sistems Laboratory, Sofia 1988
- Harvey B. — **Computer Science LOGO Style**, MIT Press, 1985
- Papert S. — **Mindstorms: Children, Computers and Powerful Ideas**, BASIC Book, New York 1980
- Reggini H.C. — **Alas para la mente, LOGO un lenguage de computadoros y un estilo de pensar**, Ediciones Galapago, Buenos Aires 1982
- Sparer E. — **Sinclair LOGO**, Sinclair Research Ltd, England 1984
- Vass Gh. — **Statutul epistemologic al astronomiei și unele implicații pedagogice**, Revista de Pedagogie nr. 3, București 1990
- Vass Gh. — **Sistemul de instruire SSIRRIUS-C**, Revista de Pedagogie nr. 4, București 1990
- Veloso E. — **LOGO Geometria 3.0**, Projeto Minerva, Universidado de Lisboa, Lisabona 1988

Editura Agni

În seria **Biblioteca de informatică** (pentru elevi) au apărut:

- **Cum să realizăm jocuri pe calculator** de Ion Diamandi (1992)
- **Hello, BASIC** de Luminița State
  (Ediția I - 1992; ediția a II-a - 1993)
- **Calculatorul, coleg de bancă** de Ion Diamandi (1993)
- **Cum se scrie un algoritm? Simplu** de Adrian Atanasiu (1993)
- **Cine ești tu, BASIC?** de Marian Gheorghe (1994)
- **Cine știe LOGO?** de Ion Diamandi (1994)

În aceași serie vor apărea:

- **Minunata lume a HC-ului** de Vlad Atanasiu
- **Algoritmi fundamentali în C++** de Răzvan Andonie și
  Ilie Gârbacea
- **Probleme pentru Olimpiadele de informatică** de Victor Mitrana
- **Antrenamente LISP** de Mihaela Maliță și Viorica Sofronie
- Etc...

Cărţile noastre se pot procura şi prin sistemul **Cartea prin poştă** - cu plata la primire (ramburs) - expediind o scrisoare simplă, după modelul de mai jos:

```
Subsemnatul.....
Str.....nr....bl....sc....ap....
Localitatea.....Judeţul.....cod.....
Doresc următoarele lucrări, cu plata ramburs:
Titlul.....nr. exemplare....
Titlul.....nr. exemplare....
etc.
```

Pentru difuzarea cărţilor noastre în şcoli, cluburi ale copiilor, cercuri de informatică etc., Editura AGNI oferă reduceri de preţuri. Astfel, pentru comenzi între 5 şi 20 de exemplare, reducerea va fi de 10%, pentru comenzi de peste 20 exemplare, reducerea va fi de 15%.
Cheltuielile de expediţie vor fi suportate de Editura AGNI.
Cărţile se livrează cu plata ramburs, în urma unei comenzi scrise.

Adresa noastră poştală este:
Editura AGNI, CP: 30-107 BUCUREŞTI
Tel: 615.55.59 Fax: 312.93.33

Cine știe LOGO?

Despre carte:

O inițiere pentru toate vîrstele în limbajul de programare LOGO prezentat pe calculatoarele compatibile HC - SPECTRUM şi IBM-PC.

Despre autor:

Ion Diamandi este redactorul şef al revistei PC WORLD, autor a numeroase cărţi de informatică, printre care:

„Dialog cu viitorul",

„Partenerul meu de joc, calculatorul",

„Cum să realizăm jocuri pe calculator",

„Calculatorul coleg de bancă".

ISBN 973-95626-9-8

Preţ: Lei 1450