infortech®



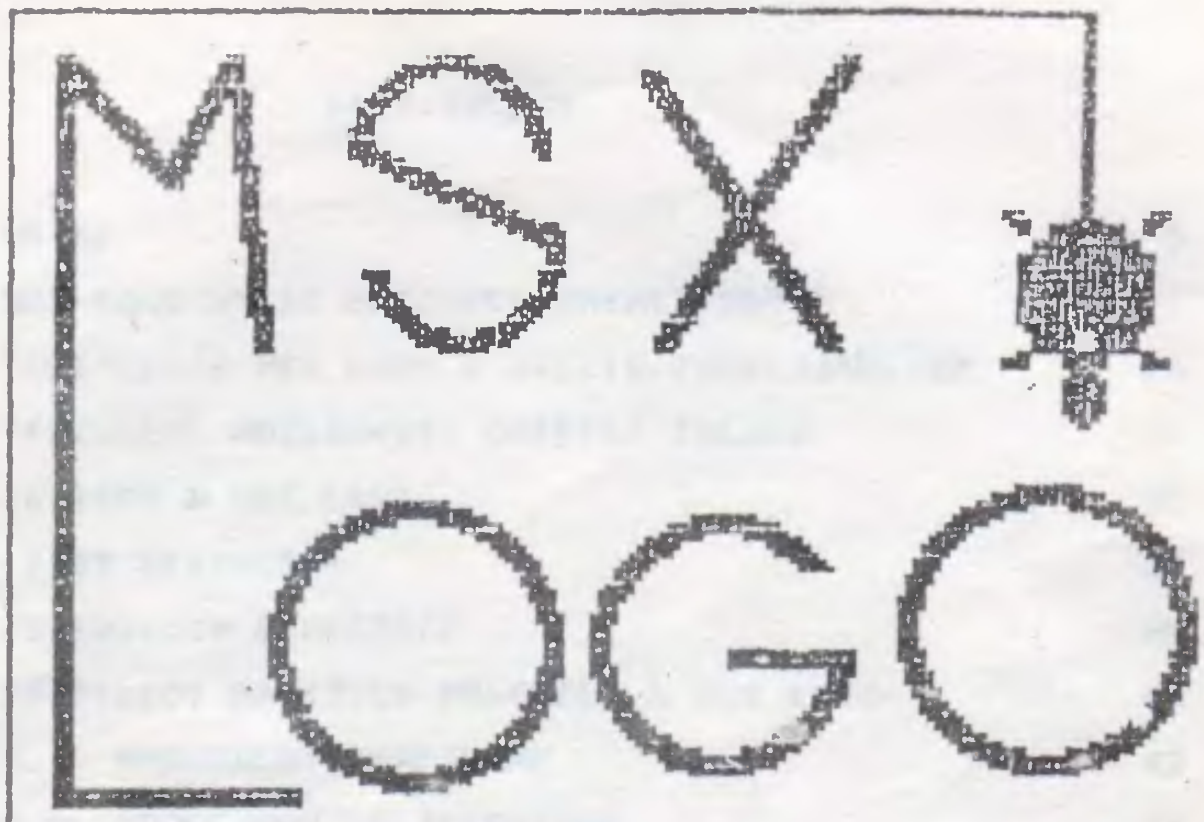
MSX LOGO



Cartridge MSX LOGO do mikrokomputera
SPECTRAVIDEO SVI 738

Infortech

MSX LOGO



PORADNIK UŻYTKOWNIKA

MAGDALENA DŁUGOZIMA

ADAM JESKE

ZESPOŁ SZKOŁ ELEKTRONICZNYCH

W ZIELONEJ GÓRZE

1987

SPIS TREŚCI

1.	WSTĘP	5
2.	MSX LOGO WYKAZ PROCEDUR PRYMITYWNYCH	6
3.	INSTRUKCJE MSX LOGO W UJECIU FUNKCJONALNYM	24
4.	SPECJALNE MOŻLIWOŚCI GRAFIKI ŻOŁWIA	26
5.	EDYTORY W MSX LOGO	33
6.	LISTY WŁASNOŚCI	36
7.	KOMUNIKATY O BŁĘDACH	38
8.	PRZYKŁADY PROSTYCH PROCEDUR W MSX LOGO	42
	8.1 PROCEDURY GRAFICZNE	42
	8.2 PROCEDURY OBLICZENIOWE	47
	8.3 PROCEDURY PRZETWARZAJACE STRUKTURY LISTOWE	49
9.	LITERATURA	51

1. WSTĘP

Niniejsze opracowanie przedstawia w skróconej formie opis języka LOGO dla mikrokomputerów rodziny MSX. Nie należy traktować tego opracowania jako podręcznika do nauki programowania w języku LOGO. Jest ono przeznaczone dla użytkowników mikrokomputerów MSX-SVI, którzy posiadają podstawowe umiejętności programowania w LOGO.

Interpreter języka MSX LOGO dostarczany jest w postaci cartridga, który należy włożyć w specjalne gniazdo rozszerzenia pamięci mikrokomputera. Po kilku sekundach od momentu włączenia zasilania mikrokomputer zgłasza swą gotowość do pracy w języku MSX LOGO.

Wersja MSX LOGO jest podobna do znanych w kraju interpreterów LOGO na mikrokomputery ZX-Spectrum, Amstrad-Schneider, Atari, Commodore czy Apple. Istotną różnicę w stosunku do wymienionych dialektów LOGO, stanowi rozbudowana grafika żółwia. Obok znanych, standardowych komend i operacji graficznych użytkownik MSX LOGO ma do dyspozycji specjalną grupę instrukcji pozwalających na używanie jednoczesne trzydziestu żółwi, z możliwością nadawania im dowolnych kształtów.

UWAGA !!!!

Cartridge MSX LOGO należy umieścić w gnieździe rozszerzenia pamięci przed włączeniem mikrokomputera. Próba wprowadzenia cartridge do tego gniazda przy włączonym napięciu zasilania mikrokomputera może spowodować uszkodzenie mikrokomputera lub cartridga.

---

2. M S X L O G O WYKAZ PROCEDUR PRYMITYWNYCH

Poniżej przedstawiony jest wykaz procedur pierwotnych (prymitywnych) występujących w MSX LOGO wraz z ich opisem w ujęciu alfabetycznym. Procedury te dzielą się na komendy (procedury nie dające na wyjściu wartości) oraz operacje (procedury wyprowadzające, "zwracające" wartość na wyjściu). Argumentami lub wartościami procedur w LOGO są obiekty (liczby, słowa, listy). W opisie procedur znakiem # przed nazwą procedury, oznaczono procedury zachłanne (procedury pozwalające na użycie ich wraz z tymi procedurami w nawiasy okrągłe.

ALL operacja

wyprowadza listę liczb od 0 do 29

 # AND *war1 war2... warn* operacja

wyprowadza TRUE jeżeli wszystkie argumenty są TRUE w przeciwnym wypadku FALSE. Jeżeli AND ma więcej niż dwa argumenty musi być wraz ze swoimi argumentami otoczone nawiasami.

ARCTAN *liczba* operacja

wyprowadza wartość w stopniach

ASCII *znak* operacja

wyprowadza kod ASCII pierwszej litery wprowadzonego słowa

ASK *nrżółwia lista* operacja lub komenda

przypisuje *numerowi żółwia* (albo liście numerów żółwi) wprowadzoną *listę*

nrżółwia - liczba całkowita od 0 do 29

BACK komenda

BK przesuwa żółwia o podaną ilość kroków do tyłu

BG operacja

wyprowadza liczbę określającą kolor tła

BUTFIRST *obiekt*

BF *obiekt* operacja

wyprowadza wszystkie z wyjątkiem pierwszego elementu podanego *obiektu*

BUTLAST *obiekt* operacja

BL *obiekt*

wyprowadza wszystkie z wyjątkiem ostatniego elementu podanego *obiektu*

BUTTONP *nrjoysticka* operacja

wyprowadza wartość logiczną TRUE jeżeli przycisk podanego *joysticka* jest naciśnięty w przeciwnym razie wyprowadza wartość logiczną FALSE

nrjoysticka - liczba całkowita 1 lub 2

CG komenda

czyści grafikę i ustawia żółwia w położeniu początkowym

CHANGE. COLOUR *nrkoloru1 nrkoloru2* komenda

zmienia kolor grafiki i tekstu na ekranie

nrkoloru - liczba całkowita od 0 do 15

CHAR *liczba* operacja

wyprowadza znak, którego kod ASCII został podany

CLEAN komenda

czyści ekran bez zmiany pozycji żółwia

CLOAD *nazwa* komenda

czyta plik o podanej *nazwie* z kasety magnetofonowej do pamięci

nazwa - słowo składające się z 6 znaków

COLOUR operacja

wyprowadza numer koloru bieżącego żółwia

COLOUR. OVER operacja

wyprowadza numer koloru pisaka bieżącego żółwia

COPYDEF *nazwa nowanazwa* komenda

kopiuje definicje istniejącej procedury o podanej *nazwie* pod *nowąnazwę*

COPYSH *nr1 nr2* komenda

kopiuje kształt żółwia o *numerze1* pod *numer2*

nr (numer kształtu żółwia) - liczba całkowita od 0 do 59

COS *liczba* operacja

wartość w stopniach

COUNT *obiekt* operacja

wyprowadza liczbę elementów podanego *obiektu*

CS komenda

kasuje ekran graficzny i przesuwa żółwia do pozycji pierwotnej

CS jest równoważne sekwencji komend **CLEAN SETSP 0 SETPOS [0 0] SETH 0**

CSAVE *nazwa* komenda

zapisuje wszystkie procedury, zmienne i listy własności z parametrem do zbioru (pliku) o podanej *nazwie* na kasecie magnetofonowej

nazwa - ciąg znaków

CT komenda

czyści tekst na ekranie

CURSOR operacja

wyprowadza pozycję kursora jako listę dwuelementową (nrkolumny nrwiersza)

DEFINE *nazwa lista* komenda

nadaje *nazwie* procedury treść *listy* będącej drugim argumentem komendy

DEFINEDP *nazwa* operacja

wyprowadza TRUE jeżeli słowo jest nazwą procedury, FALSE jeżeli nie

DIFF *liczba1 liczba2* operacja

wyprowadza rezultat dzielenia *liczby1* przez *liczbę2*

DIR komenda

wyprowadza listę nazw zbiorów LOGO znajdujących się na dysku

DISTANCE [*x y*] operacja

wyprowadza odległość między pozycją bieżącego żółwia a punktem o współrzędnych *x y*

DOT [*x y*] komenda

stawia punkt na ekranie o współrzędnych *x y*

EACH *listainstrukcji* komenda

powoduje sekwencyjne wykonanie *listy instrukcji* przez każdego żółwia

EDIT *nazwa* komenda

EDIT *listanazw*

ED *nazwa*

ED *listanazw*

włączenie edytora

EDNS komenda

umieszcza wszystkie nazwy zmiennych w edytorze

EMPTYP *obiekt* operacja

wyprowadza TRUE jeśli obiekt jest pusty w przeciwnym razie FALSE

END komenda

mówi LOGO że zostało zakończone definiowanie procedury. END musi być jedynym słowem w ostatniej linii procedury

EQUALP *obiekt1 obiekt2* operacja

wyprowadza TRUE, jeżeli *obiekt1* i *obiekt2* są równymi liczbami, tymi samymi słowami lub identycznymi listami

ERALL komenda

usuwa wszystkie procedury, zmienne i listy własności z pamięci

ERASE *nazwa* komenda

ERASE *listanazw*

ER *nazwa*

ER *listanazw*

usuwa wskazane procedury z pamięci

ERDS komenda

usuwa wszystkie demony

ERF *nazwa* komenda

usuwa wskazany zbiór z dysku

ERN *nazwa* komenda

ERN *listanazw*

usuwa podane *nazwy* zmiennych z pamięci

ERNS komenda

usuwa wszystkie nazwy zmiennych z pamięci

ERPROPS komenda

usuwa wszystkie listy własności z pamięci

ERPS komenda

usuwa wszystkie procedury z pamięci

ES *nrkształtu* komenda

ustawia kształt żółwia o podanym *numerze* w edytorze kształtu

nrkształtu - liczba całkowita od 0 do 59

FALSE operacja

wartość FAŁSZ

FILL komenda

pokrywa obszar na ekranie aktualnie używanym kolorem zmieniając punkt pod żółwiem (oraz wszystkie pionowe i poziome ciągłe linie tego samego koloru) odpowiednio do aktualnego stanu pióra

FIRST *obiekt* operacja

wyprowadza pierwszy element słowa lub listy. FIRST słowa jest znakiem, FIRST listy może być słowem lub listą

FD *liczba* komenda

przesuwa żółwia do przodu, bez zmiany jego kierunku, o podaną liczbę kroków

FPUT *obiekt lista* operacja

wyprowadza nową listę utworzoną przez wstawienie *obiektu* na początek *listy*

FREEZE komenda

zawiesza szybkość biegu wszystkich żółwi, dopóki nie wystąpi komenda THAW

GETSH *nrkształtu* operacja

wyprowadza listę 32 liczb reprezentujących kształt żółwia

nrkształtu - liczba całkowita od 0 do 59

GPROP *nazwa własności* operacja

wyprowadza wartość własności danego elementu z listy własności o podanej *nazwie*

HEADING operacja

wyprowadza liczbę (między 0 a 359) kierunek (kąt) ustawienia bieżącego żółwia

HOME komenda

przesuwa żółwia do środka ekranu (pozycja o współrzędnych [0 0] i kąt 0)

NT komenda

czyni żółwia niewidocznym na ekranie

IF *war listainstrukcjiz listainstrukcjiz* komenda

pierwszym wyrażeniem po IF jest warunek lub operacja logiczna. Jeżeli warunek jest TRUE wykonywana jest *lista instrukcjiz* ,jeżeli warunek jest FALSE wykonywana jest *lista instrukcjiz* (jeżeli istnieje)

INT *liczba* operacja

wyprowadza część całkowitą liczby utworzoną przez obcięcie części ułamkowej

ITEM *liczba lista* operacja

wyprowadza n-ty (określony *liczbą*) element *listy*

JOY *nrjoysticka* operacja

wyprowadza liczbę z przedziału od 0-8 reprezentującą bieżącą pozycję joysticka

nrjoysticka - liczba całkowita 1 lub 2

JOYP *nrjoysticka* operacja

wyprowadza wartość TRUE jeśli badany *joystick* jest odchylony, w przeciwnym razie wyprowadza FALSE

KEYP operacja

wyprowadza wartość logiczną TRUE jeśli znak z klawiatury gotowy jest do odczytania przez system

LAST *obiekt* operacja

wyprowadza ostatni element listy lub ostatni znak słowa

LEFT *liczba* komenda

LT *liczba*

obraca żółwia o podaną liczbę stopni w lewo bez zmiany jego pozycji

 # LIST *obiektz obiekt2 ... obiektn* operacja

wyprowadza listę utworzoną z elementów *obiektz obiekt2* itd.

LISTP *obiekt* operacja

wyprowadza TRUE jeżeli *obiekt* jest listą, w przeciwnym wypadku FALSE

LOAD *nazwa* komenda

czyta plik o podanej *nazwie* z dysku do przestrzeni roboczej

nazwa - słowo składające się z co najwyżej 8 znaków

LOADPIC *nazwa* komenda

czyta z dysku i wyświetla na ekranie obraz zapisany uprzednio jako plik instrukcją SAVEPIC

LPUT *obiekt lista* operacja

wyprowadza nową listę , w której *obiekt* umieszczony jest na końcu *listy*

MAKE *nazwa obiekt* komenda

przypisuje wartość *obiekt* do *nazwy* zmiennej.Można także uważać zmienną za symbol lub wskaźnik odnoszący się do *obiektu*

NAME *obiekt nazwa* komenda

przypisuje *nazwę* podanemu *obiektowi*

NAMEP *obiekt* operacja

wyprowadza TRUE jeżeli *obiekt* ma wartość , FALSE jeżeli nie ma

NOISE *ton amplituda czastrwania* komenda

generuje szum o podanym *tonie*, *amplitudzie* i *czasietrwania*

ton - liczba całkowita od 0 do 31

amplituda - liczba całkowita od 0 do 15

czastrwania - liczba całkowita od 0 do 255 (50 jest równoważne 1 sekundzie)

NOPRINTER komenda

wyłączenie drukarki

NOT *war* operacja

wyprowadza TRUE jeżeli *warunek* jest FALSE i FALSE jeżeli warunek jest TRUE

NUMBERP *obiekt lista* operacja

wyprowadza TRUE jeżeli *obiekt* jest elementem *listy*, w przeciwnym wypadku FALSE

ON TOUCH *nrżółwia nrżółwia listainstrukcji* komenda

ustawia demona. Jeśli nastąpi kolizja między żółwiami o podanych *numerach* wykonywana jest *lista instrukcji*

nrżółwia - liczba całkowita od 0 do 29

M OR *war1 war2* operacja

(OR *war1 war2 ... warn)*

wyprowadza TRUE jeżeli którykolwiek z *warunków* jest TRUE w przeciwnym wypadku FALSE.

OUTPUT *obiekt* operacja

OP *obiekt*

zatrzymuje wykonywanie procedury bez kontynuacji do END, lecz odmiennie niż STOP, OUTPUT przekazuje swój *obiekt* do procedury wywołującej (nadrzędnej0).

PC operacja

wyprowadza kolor pisaka bieżącego żółwia

PE komenda

zmienia kolor pióra żółwia na kolor tła

PENDOWN komenda

PD

opuszcza pisak bieżącego żółwia

PENUP komenda

PU

podnosi pisak bieżącego żółwia tak, że żadna linia nie jest rysowana, kiedy żółw się przesuwa

PLIST *nazwa* operacja

wyprowadza listę własności obiektu związanego z podaną *nazwą*

PO *nazwa* komenda

wypisuje definicje procedury o podanej *nazwie*. Można wypisać kilka definicji przez podanie ich nazw jako listy

POALL komenda

wypisuje tytuły i definicje wszystkich procedur i wartości wszystkich zmiennych z przestrzeni roboczej

PODS komenda

drukuje wszystkie demony ustawione komendami ON.TONCH i WHEN oraz ich listę instrukcji

POFILE *nazwazbioru* komenda

wyprowadza zbiór o podanej *nazwie*

PONS komenda

wypisuje wszystkie nazwy i wartości zmiennych z przestrzeni roboczej

POPS komenda

wypisuje definicje wszystkich procedur aktualnie istniejących w przestrzeni roboczej

POS operacja

wyprowadza pozycje bieżącego żółwia jako listę dwóch liczb (współrzędne położenia x y)

POTS komenda

wypisuje tytułowe nazwy wszystkich procedur z przestrzeni roboczej

PPROP *nazwa własności obiekt* komenda

nadaje określonemu słowu (procedurze) z danej listy jakąś własność

PPS komenda

wyprowadza wszystkie listy własności z przestrzeni roboczej

PRIMITIVEP *słowo* operacja

wyprowadza TRUE, jeżeli *słowo* jest procedurą podstawową LOGO, FALSE jeżeli nie

PRIMITIVES komenda

wypisuje nazwy wszystkich procedur które LOGO "zna" - (procedury pierwotne)

 # PRINT *obiekt* komenda

PR *obiekt*

wyprowadza podany *obiekt* na ekran, usuwa nawiasy ograniczające, uzupełnia ostatni wyprowadzony *obiekt* znakiem CR (powrót katetki))

 # PRODUKT *liczba1 liczba2* operacja

(PRODUKT *liczba1 liczba2 ... liczban)*

wyprowadza iloczyn czynników wejściowych. Jest równoważny "*".

PUTSH *nrkształtu kształt* komenda

nadaje żółwiowi o kształcie określonym *numerem kształtu* formę zdefiniowaną przez drugi parametr *kształt*

nrkształtu - liczba całkowita od 0 do 59

kształt - lista 32 liczb reprezentujących matrycę kształtu

PX komenda

powoduje, że kolor punktów na przebytej przez żółwia drodze zostaje zamieniony na kolor będący dopełnieniem poprzedniego

QUOTIENT *liczba1 liczba2* operacja

wyprowadza część całkowitą z dzielenia *liczby1* przez *liczbę2*

RANDOM *liczba* operacja

jeżeli liczba jest dodatnią liczbą całkowitą wyprowadza liczbę losową z zakresu od 0 do n-1

RC operacja

czeka na naciśnięcie klawisza (lub kombinacji klawiszy). Operacja wyprowadza znak, lecz nie jest on zapisywany na ekranie

RECYCLE komenda

uwalnia tak dużo komórek (węzłów) jak jest to tylko możliwe przez "odśmiecenie" pamięci

REMAINDER *liczba1 liczba2*

wyprowadza resztę z dzielenia *liczby1* przez *liczbę2*

REMPROP *nazwa własności* komenda

usuwa "wartość" *własności* danego elementu z listy

REPEAT *liczba listainstrukcji* komenda

powtarza *listę instrukcji* podaną ilość razy. *Liczba* musi być dodatnia.

RERANDOM komenda

generuje tę samą sekwencję liczb losowych

RG komenda

kasuje grafikę (również demony)

RIGHT *liczba* komenda

RT *liczba*

obraca żółwia opodaną *liczbę* stopni w prawo bez zmiany jego pozycji

RL operacja

linia wprowadzona przed naciśnięciem ENTER jest wyprowadzana jako lista, każdy znak jest wypisywany na ekranie

ROUND *liczba* operacja

wyprowadza zaokrąglenie do najbliższej *liczby* całkowitej

RUN *listainstrukcji* komenda

z listą LOGO na wejściu RUN wykonuje l i s t ę jako l i n i ę LOGO

SAVE *nazwa* komenda

nagrywa program o podanej *nazwie* poprzedzonej apostrofem na dysk. Długość nazwy jest ograniczona do ośmiu znaków

SAVEPIC *nazwa* komenda

LOGO zapisuje na dysku całą aktualną zawartość ekranu pod podaną *nazwą*

SCRUNCH operacja

wyprowadza współczynnik [x y], stosunek wielkości pionowego kroku żółwia do wielkości jego poziomego kroku

W SENTENCE *obiekt1 obiekt2* operacja
SE *obiekt1 obiekt2.... obiektn*

wyprowadza listę złożoną z *obiektów* na wejściu

SETBG *nrkoloru* komenda

ustawia kolor tła na podany kolor

nrkoloru - liczba całkowita od 0 do 15

SETC *nrkoloru* komenda

ustawia kolor bieżącego żółwia na podany *kolor*

SETCURSOR *[x y]* komenda

przesuwa kursor do pozycji wyznaczonej przez listę [x y]. Pierwszym elementem tej listy jest nrkolumny drugim nrlinii

SETH *liczba* komenda

ustawia żółwia na kierunek określony *liczbą* stopni

SETPC *nrkoloru* komenda

ustawia kolor pisaka żółwia na podany *numer koloru*

SETPOS *[x y]* komenda

przesuwa żółwia do podanej pozycji [x y] jeżeli PD - zostawia linię

SETSCR *[x y]* komenda

ustawia współczynnik skali na [x y]

SETSH *nrkształtu* komenda

ustawia kształt bieżącego żółwia

nrkształtu - liczba 0 do 59

SETSP *liczba* komenda

ustawia prędkość bieżącego żółwia

liczba - liczba całkowita od -128 do 128

SETTEXT *nrlinii* komenda

przesuwa kursor do początku linii o podanym *numerze*

SETX *liczba* komenda

przesuwa żółwia do punktu o podanej współrzędnej bez zmiany współrzędnej y. Jeżeli PD rysuje poziomą linię

SETXVEL *liczba* komenda

ustawia składową x szybkości żółwia

SETY *liczba* komenda

przesuwa żółwia do punktu o podanej współrzędnej bez zmian współrzędnej x. Jeżeli PD rysuje pionową linię

SETYVEL *liczba* komenda

ustawia składową y szybkości żółwia

SHADE komenda

zamalowywuje ekran kształtami żółwia

SHAPE operacja

wyprowadza numer kształtu bieżącego żółwia

SHOW *obiekt* komenda

wypisuje słowa, listy lub liczby podawane jako argumenty

SHOWNP operacja

wyprowadza TRUE jeżeli żółw jest widzialny, FALSE jeżeli nie

SNAP komenda

zamienia kształt żółwia z obrazem graficznym

SPACE operacja

wyprowadza liczbę wolnych komórek przestrzeni roboczej. Komórka zajmuje 5 bajtów pamięci (jeden węzeł)

SPEED operacja

wyprowadza szybkość bieżącego żółwia

SQRT *liczba* operacja

wyprowadza pierwiastek kwadratowy z podanej liczby, która musi być dodatnia

ST komenda

przywraca na ekranie symbol żółwia ukryty komendą HT

STAMP komenda

odbija kopię kształtu bieżącego żółwia na ekranie

STOP komenda

zatrzymuje wykonywanie aktualnej procedury i przekazuje sterowanie do procedury która ją wywołała

(SUM *liczba1 liczba2 ... liczban*) operacja

wyprowadza sume składników *liczb*

TELL *nrżółwia* komenda

TELL *listażółwi*

czyni żółwia lub listę żółwi bieżącymi żołwiami

TEXT *słowo* operacja

kiedy *słowo* jest nazwą procedury, TEXT wyprowadza tekst procedury jako listę

THAW komenda

jeśli występuje po FREEZE, kasuje poprzednią szybkość żółwia

VERSION komenda

drukuje numer kodów aktualnie używanych w wersji LOGO

THING *nazwa* operacja

wyprowadza wartość nazwy

TO *nazwa parametr1 parametr2 ...* komenda

pozwala na definiowanie własnych procedur. LOGO nie wykonuje instrukcji pisanych w tym trybie ale je zapamiętuje

TOOT *kanał częstotliwość amplituda czastrwania* komenda

wydaje dźwięk w podanym kanale o określonej *częstotliwości, amplitudzie i czasietrwania*

TOPLEVEL komenda

zatrzymuje wykonywanie procedury i przekazuje sterowanie na najwyższy poziom LOGO

TOWARDS *[x y]* operacja

wyprowadza kierunek (azymut), który żółw musiałby mieć aby być skierowanym na punkt [x y] żółw nie jest przestawiony po użyciu TOWARDS

TRUE operacja

PRAWDA

 # TYPE *obiekt* komenda

(TYPE *obiekt1 obiekt2 ... obiektn***)**

wypisuje podane *obiekty* na ekranie lecz nie przechodzi do następnej linii po wypisaniu obiektów

WAIT *liczba* komenda

przerywa działanie procedury na czas wynikający z argumentu komendy (czas = *liczba* / 50)

WHEN *nrwydarzenia listainstrukcji* komenda

jeśli wystąpi wydarzenie o podanym numerze (0 do 5) wykonana zostanie *lista instrukcji*

WHO operacja

wyprowadza listę bieżących żółwi

WINDOW komenda

umożliwia żółwiowi poruszanie się poza obszarem ekranu

WORD *słowo1 słowo2* operacja

(WORD *słowo1 słowo2 ... słowon)*

wyprowadza słowo złożone ze *słów* na wejściu. WORD nie przyjmuje listy na wejściu

WORDP *obiekt* operacja

wyprowadza TRUE jeżeli *obiekt* jest słowem, FALSE jeżeli nie

WRAP komenda

tworzy pole dla żółwia "sklejone" na krawędziach ekranu

XCOR operacja

wyprowadza współrzędną x bieżącego położenia żółwia

XVEL operacja

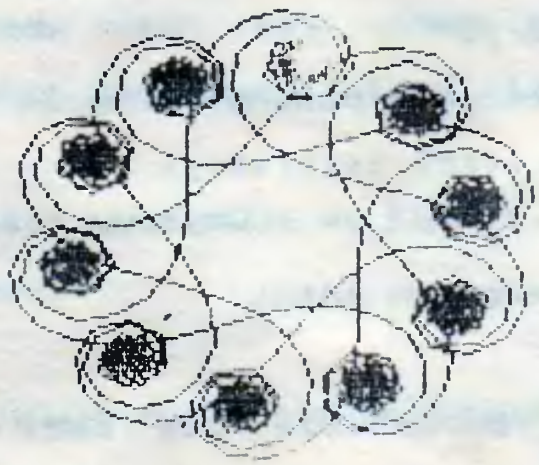
wyprowadza składową x prędkości bieżącego żółwia

YCOR operacja

wyprowadza współrzędną y bieżącego położenia żółwia

YVEL operacja

wyprowadza składową y prędkości bieżącego żółwia



PROCEDURY SPECJALNE, ZORIENTOWANE SYSTEMOWO

.call *adres* komenda

uruchamia program w kodzie maszynowym umieszczonym w pamięci od podanego *adresu*

adres - liczba całkowita od 0 do 65535

.deposit *adres bajt* komenda

wpisuje liczbę *bajt* do komórki pamięci komputera określonej przez *adres*

bajt - liczba o 0 do 255

.examine *adres* operacja

wyświetla zawartość komórki pamięci o podanym *adresie*

.in *port* operacja

wyprowadza liczbę (bajt) z *portu* o podanym numerze

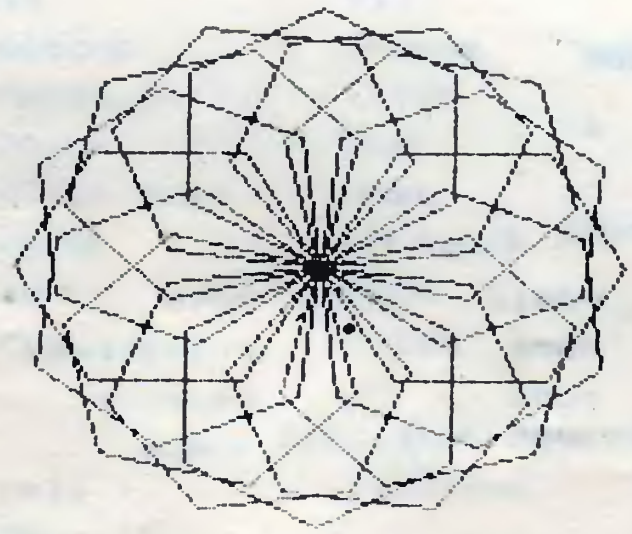
port - liczba od 0 do 255

.out *port bajt* komenda

wysyła liczbę (*bajt*) do *portu* o podanym numerze

port - liczba od 0 do 255

bajt - liczba od 0 do 255



3. INSTRUKCJE MSX LOGO W UJĘCIU FUNKCJONALNYM

GRAFIKA ŻÓŁWIA

back,bk	home	right,rt	shownp
bg	ht	scrunch	st
cg	left,lt	setbg	towards
clean	pc	seth	window
cs	pe	setpc	wrap
distance	pendown,pd	setpos	xcor
dot	penup,pu	setscr	ycor
fordward,fd	pos	setx	
heading	px	sety	

SŁOWA I LISTY

ascii	empty	last	sentence,se
butfirst,bf	eqalp	list	word
butlast,bl	first	listp	wordp
char	fput	lput	
count	item	numberp	

ZMIENNE

make	name	namep	thing

OPERACJE ARYTMETYCZNE

arctan	int	random	round
cos	product	remainder	sin
diff	quotient	rerandom	sqrt
			sum

OPERACJE LOGICZNE

and	not	or

PRZESTRZEŃ ROBOCZA

erall	erns	poall	pots
erase,er	érps	pons	primitives
ern	po	pops	recycle
			space

DEFINIOWANIE I EDYCJA PROCEDUR

copydef	definedp	edns	primitivep
define	edit,ed	end	text
			to

INSTRUKCJE STERUJACE

false	output,op	run	toplevel
if	repeat	stop	true

EKRAN, KLAWIATURA I DRUKARKA

ct	noprinter	rc	settext
cursor	print,pr	rl	show
keyp	printer	setcursor	type
			wait

LISTY WLASCIWOSCI

erprops	gprop	plist	pps
		pprop	remprop

MUZYKA

noise	toot		

ZARZADZANIE ZBIORAMI

cload	dir	load	pofile
csave	erf	loadpic	save
			savepic

GRAFIKA SPRITOW

all	es	rg	speed
ask	fill	setc	stamp
buttonp	freeze	setsh	tell
change.colour	getsh	setsp	thaw
colour	joy	setxvel	when
colour.over	joyp	setyvel	who
copysh	on.touch	shade	xvel
each	pods	shape	yvel
erds	putsh	snap	

SPECJALNE PROCEDURY PIERWOTNE

.call	.examine	.out	version
.deposit	.in		

4. SPECJALNE MOŻLIWOŚCI GRAFIKI ŻÓŁWIA

W trakcie pracy w MSX LOGO programista może używać trzydziestu żółwi tzn. może dysponować wszystkimi żółwiami naraz oraz każdym z osobna. Wszystkie żółwie mogą tworzyć rysunki jednocześnie. Ponadto każdy z nich może posiadać dowolny kształt (zdefiniowany przez użytkownika).

Żółwie ponumerowane są od 0 do 29. W momencie startu LOGO traktuje wszystkie komendy użytkownika tak jakby były przeznaczone dla żółwia o numerze 0. Chcąc adresować komendy do innego żółwia należy wydać polecenie TELL z argumentem określającym numer żółwia np.

```
? TELL 1
```

Natomiast chcąc adresować komendy jednocześnie do kilku żółwi należy ich numery umieścić w nawiasach kwadratowych np.

```
? TELL [ 0 2 4 5 ]
```

Polecenie

```
? TELL ALL
```

uaktywnia wszystkie żółwie

Komenda SETSP wprawia żółwia w ruch nadając mu stałą prędkość. I tak ciąg poleceń

```
? TELL [ 0 1 2 3 ]
? RT 90
? FD 20
? SETSP 50
```

spowoduje, że żółwie o numerach 0, 1, 2 i 3 obrócą się w prawo i przejdą 20 kroków na przód po czym zaczną się poruszać w zdłuż ekranu.

W odróżnieniu od standardowej grafiki żółwia, LOGO poruszając żółwiami jest gotowe wykonywać inne polecenia użytkownika, np. komendy

```
? TELL 1 RT 180
```

zmienią kierunek poruszania się pierwszego żółwia.
Komenda EACH powoduje sekwencyjne (kolejne) wykonanie przez aktywne żółwie (wywołane komendą TELL) listy instrukcji będącej jej argumentem. Np. ciąg poleceń

```
? TELL [ 1 2 3 ]
? EACH [ st setx who * 30 ]
```

spowoduje następujące ustawienie żółwi na ekranie
żółw nr1 przyjmie pozycję określoną parą współrzędnych [30 0]
żółw nr2 przyjmie pozycję określoną parą współrzędnych [60 0]
żółw nr3 przyjmie pozycję określoną parą współrzędnych [90 0]
Operacja WHO wyprowadza listę bieżących żółwi.
Natomiast efektem wywołania procedury

```
to start.
tell all
cs st pu
each [ seth 12 * who ]
setsp 50
end
```

będzie ustawienie wszystkich żółwi pod kątem będącym wielokrotnością liczby 12 i poruszanie ich ze stałą prędkością wzdłuż nadanego kierunku.

Ruch żółwi można w dowolnym momencie przerwać komendą FREEZE, a następnie ponownie kontynuować komendą THAW.

Jak już wspomniano żółwiom można nadawać różne kształty.

Początkowo MSX LOGO przechowuje w pamięci sześćdziesiąt standardowo zdefiniowanych kształtów (wyglądów) żółwi. W trakcie pracy można dowolnemu żółwiowi nadać określony kształt (od 1 do 60) komendą SETSH.

Np. ciąg poleceń

```
? tell 0 st
? setsh 3
? pu fd 60
```

Wywołuje żółwia o numerze 0. Nadaje mu kształt o numerze 3 (pies) i przenosi w przód o 60 kroków bez zostawiania śladu. MSX LOGO pozwala ponadto zmieniać kształt żółwia. Do tego celu służy Edytor Kształtu wywoływany komendą ES. Po wydaniu tej komendy z parametrem określającym numer kształtu na ekranie pojawi się siatka złożona z szesnastu kolumn po szesnaście kwadratów w każdej kolumnie. Wypełnione na czarno kwadraty definiują kształt. Przesuwając migający kursor po kwadratach siatki można ten kształt zmieniać .

Jeśli ustawimy kursor w wybranym kwadracie i przyciśniemy klawisz spacji - to wypełni się on kolorem czarnym, jeśli jeszcze raz przyciśniemy klawisz spacji to kwadrat będzie pusty. Stosując tą technikę można "namalować" na siatce dowolny rysunek, który definiuje nową postać kształtu żółwia.

Od momentu opuszczenia Edytora Kształtu (klawisz ESC) zdefiniowany kształt jest przechowywany w przestrzeni roboczej komputera tak długo, dopóki komputer nie zostanie wyłączony.
Można zdefiniowane kształty przechowywać pod nazwami zmiennych.

Do tego celu służy operacja GETSH, która wyprowadza listę 32 liczb reprezentujących siatkę kształtu żółwia o numerze będącym argumentem tej operacji. I tak polecenie

```
? pr getsh 3
```

wyprowadzi następującą listę 32 liczb reprezentujących aktualny kształt (zdefiniowany standardowo) o numerze 3 [0 0 0 128 128 79 127 31 31 28 58 114 68 68 66 0 0 28 59 63 120 240 240 240 240 240 144 144 72 72 108]

Można teraz zapamiętać ten kształt przypisując dowolnej nazwie listę reprezentującą kształt. Np.

```
make "pies getsh 3
```

Odtąd w pamięci komputera przechowywany jest kształt psa zdefiniowany jako kształt 3 pod nazwą pies.

Komenda SAVE zapisuje na dyskietce opis wszystkich zdefiniowanych procedur także kształt o numerze 3 pod nazwą PIES. Po późniejszym wczytaniu zbioru z dyskietki komendą LOAD odzyskujemy dany kształt.

Obok definiowania kształtu w Edytorze Kształtu, MSX LOGO pozwala określić pożądany kształt komendą PUTSH, która nadaje (podanemu jako pierwszy argument) numerowi kształtu formą zdefiniowaną przez listę 32 liczb (będącą drugim argumentem komendy) reprezentyjących siatkę kształtu..

Poniższy program obok działania wyżej opisanych komend i operacji ilustruje możliwości dynamiczne (animacja) jakie daje MSX LOGO.

```
to ptaki
danepoczatkowe
tell :listaptakow
st pu cs setc 1
rt 60
each [ fd who * 15 ]
rt 30
setsp 20
skrzydla
end

to danepoczatkowe
putsh 11 [ 0 0 0 192 48 12 6 3 1 0 0 0 0 0 0 0 0 0 0
3 12 48 192 128 0 0 0 0 0 0 0 0 ]
putsh 12 [ 0 0 0 0 0 0 0 0 3 12 48 64 128 0 0 0 0  0
0 0 0 0 0 128 192 48 12 2 1 0 0 0 ]
make "dogory 11
make "wdol 12
name [ 2 3 4 5 6 7 8 9 10 11 ] "listaptakow
end

to skrzydla
setsh :dogory
toot 0 440 15 15
wait 20
setsh :wdol
toot 0 880 15 15
wait 20
skrzydla
end
```

Komenda TOOT generuje dźwięk określony odpowiednimi parametrami, natomiast komenda SETC ustawia żądany kolor żółwia. Po wprowadzeniu procedur do pamięci mikrokomputera program uruchamiamy wydając polęcenie

```
? ptaki
```

DEMONY W MSX LOGO

DEMON jest specjalnym środkiem programowym. Pracuje niezależnie od jakichkolwiek bieżących programów. Czeka na pewne zdarzenie lub na kolizję między dwoma żółwiami. Kiedy wystąpi to zdarzenie lub kolizja DEMON przerywa jakąkolwiek bieżącą procedurę i umieszcza swoją listę instrukcji. Kiedy skończą się instrukcje DEMONA program kontynuuje od miejsca , w którym przerwał pracę. Procedury prymitywne, które uaktywniają DEMONY to **ON.TOUCH** i **WHEN.**

DEMONY **ON.TOUCH** reagują na kolizję między jakimikolwiek żółwiami. DEMONY **WHEN** czekają na pewne rodzaje wydarzeń (wypisane w tabeli). DEMON czeka dalej na kolizję lub wydarzenie, dopóki nie jest usunięty. DEMONY są kasowane przez **ERDS**, albo kiedy lista instrukcji na wejściu komend **ON.TOUCH** lub **WHEN** jest listą pustą.

TABELA WYDARZEŃ DLA PROCEDURY **WHEN**

NUMER	WYDARZENIE
0	w przybliżeniu raz na sekundę
1	naciśnięto klawisz
2	wciśnięty rzycisk joysticka 1
3	wciśnięty przycisk joysticka 2
4	joystick 1 poruszony
5	joystick 2 poruszony

Poniższe prodramy ilustrują działanie demonów.

```
to odbijaniepilki
tell [ 0 1 2 3 4 5 ]
st pu
each [ seth 60 * who ]
fd 60
rt 180
tell 6
setsh 0
rt 180 st pu
on.touch 6 0 [ rt 150 ]
on.touch 6 3 [ rt 150 ]
on.touch 6 1 [ rt 150 ]
on.touch 6 2 [ rt 150 ]
on.touch 6 4 [ rt 150 ]
on.touch 6 5 [ rt 150 ]
setsp 20
end


to rysowanie
when 2 [ noise 10 12 60 ]
when 4 [ ruch ]
rysowanie
end

to ruch
setsp 5
if (joy 1) = 5 [seth 180 fd 1]
if (joy 1) = 1 [seth 0 fd 1]
if (joy 1) = 3 [seth 90 fd 1]
if (joy 1) = 7 [seth 270 fd 1]
if (joy 1) = 2 [seth 45 fd 1]
if (joy 1) = 4 [seth 135 fd 1]
if (joy 1) = 6 [seth 225 fd 1]
if (joy 1) = 8 [seth 315 fd 1]
if (joy 1) = 0 [setsp 0]
end
```

5. EDYTORY W MSX LOGO

1.Edytor procedur

Edytor wykorzystywany jest do definiowania i modyfikowania procedur oraz zmiennych, które są następnie wprowadzane do przestrzeni roboczej LOGO w pamięci komputera. Komendy TO, EDIT (ED) oraz EDNS inicjują Edytor.

W trakcie pracy w Edytorze można wykorzystywać następujące klawisze bądź kombinacje klawiszy:

Uwaga!

symbol * określa klawisze lub kombinacje klawiszy wykorzystywane zarówno w Edytorze LOGO jak i poza nim.

*	——>	przesuwa kursor w prawo o jedną spację
*	<——	przesuwa kursor w lewo o jedną spację
*	↓	przesuwa kursor w dół do następnego wiersza
*	↑	ptzesuwa kutsot w gótę do poprzedniego wiersza
*	CTRL - A	przesuwa kursor na początek wiersza
*	CTRL - E	przesuwa kursor na koniec wiersza
*	CTRL - T	przesuwa kursor na szczyt bufora edycji (na początek tekstu)
*	CTRL - B	przesuwa kursor na dół bufora edycji (na koniec tekstu)
*	ENTER	tworzy nowy wiersz przesuwając kursor i resztę wiersza tekstu na początek nowego wiersza (do nowej linii). Wciśnięcie klawisza poza Edytorem nakazuje LOGO realizację tego co zostało wprowadzone z klawiatury.
	INS	Otwiera nowy wiersz w miejscu, gdzie znajduje się kursor, ale nie przesuwa kursora (do nowej linii).

* **BS** Wymazuje znak znajdujący się na lewo od kursora.
* **DEL** Wymazuje znak znajdujący się pod kursorem.
* **CTRL - K** Wymazuje tekst począwszy od miejsca, w którym znajduje się kursor do końca wiersza. Tekst ten zostaje wprowadzony do bufora (zostaje zapamiętany w specjalnym buforze wymazań. Bufor ten może pomieścić 256 znaków.
* **CTRL - Y** Wstawia kopię tekstu zawartego w buforze wymazań w miejsce gdzie znajduje się kursor.

ESC Realizuje (powoduje) wyjście z Edytora i wprowadza wszystkie definiowane i modyfikowane procedury oraz zmienne do przestrzeni roboczej.

* **CTRL - STOP** Realizuje (powoduje) wyjście z Edytora. Wszelkie zmiany dokonane w Edytorze zostają zignorowane.

Wykorzystanie tej kombinacji klawiszy poza Edytorem zatrzymuje (przerywa) wykonanie procedury.

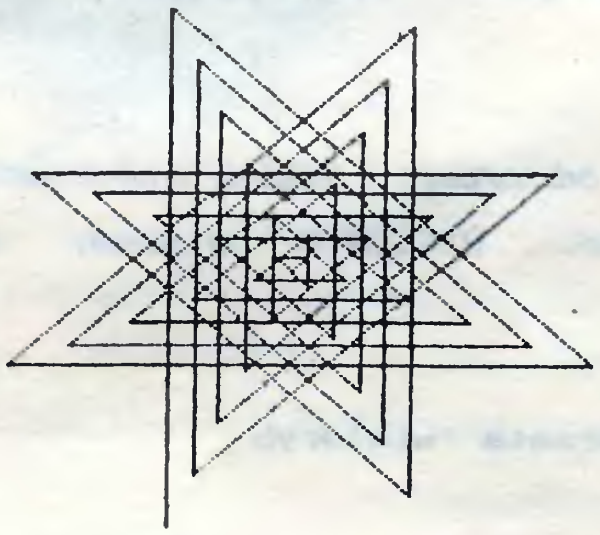
2.Edytor kształtu

Komenda ES inicjuje Edytor kształtu, tzn. wprowadza kształt o podanym numerze do Edytora kształtu, co z kolei umożliwia zmianę tego kształtu i utworzenie nowego.

Edytor kształtu jest dużą kwadratową siatką składającą się z 16 x 16 kwadratów. Wypełnione (czarne) kwadraciki określają kształt. Kursor migający (biały kwadracik) to pole, które aktualnie ma być wypełnione lub czyszczone.

Tworząc kształty w Edytorze kształtu użytkownik ma do dyspozycji następujące klawisze lub ich kombinacje.

——>	przesuwa kursor w prawo o jedno pole (kwadracik)
<---	przesuwa kursor w lewo o jedno pole
↑	przesuwa kursor w górę o jedno pole
↓	przesuwa kursor w dół o jedno pole
SPACJA	wypełnia (zaczernia) puste pola lub czyści pole wypełnione
CTRL - K lub CLS/HOME	wymazuje cały kształt, pozostawiając pustą siatkę
CTRL - Y	wprowadza kształt znajdujący się w Edytorze kształtu po dokonaniu zmian
ESC	wyjście z Edytora kształtu z zapamiętaniem dokonanych zmian (nowego kształtu)
CTRL - STOP	wyjście z Edytora kształtu. Wszelkie zmiany kształtu dokonane w Edytorze zostają zignorowane.



6. LISTY WŁASNOŚCI

Listy własności wzbogacają system programowania o nowe, ciekawe możliwości.

Lista własności jest parą dwóch elementów. Pierwszy określa nazwę własności, drugi jej wartość. W LOGO każdemu słowu, każdej liście można przyporządkować jedną lub wiele list własności.

Przykład:

Skonstruujemy najpierw listę o nazwie ZWIERZETA w skład której będą wchodzić następujące słowa: WILK, WIELORYB, OWCA.

```
MAKE "zwierzeta [ wilk owca wieloryb ]
```

Dla każdego elementu listy określimy pewne własności:

```
PPROP "zwierzeta "wilk [ ssak stadny zyjacy w lasach lubi
zjadac owce ]
PPROP "zwierzeta "owca [ ssak hodowany dla welny lubiany
przez wilka ]
PPROP "zwierzeta "wieloryb [ ssak wodny zywiacy sie
planktonem ]
```

Jeżeli chcemy teraz odczytać, jakie informacje zostały zapisane pod hasłem np. WIELORYB piszemy następującą sekwencję:

```
PR GPROP "zwierzeta "wieloryb
```

Gdy interesują nas wszystkie listy własności zapisane pod hasłem ZWIERZETA:

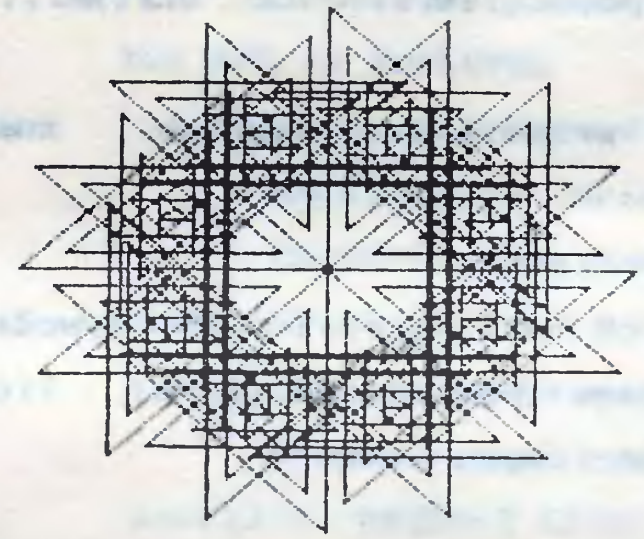
```
PR PLIST "zwierzeta
```

Kiedy pożądana jest modyfikacja wprowadzonych danych używamy instrukcji **REMPROP** - umożliwiającej kasowanie własności w określonym miejscu, np.:

REMPROP "zwierzeta "wilk

skasuje wszystkie informacje o wilku.

Instrukcja **PPS** - ujawnia wszystkie aktualne nazwy własności i ich wartości.



7. KOMUNIKATY O BŁĘDACH

W LOGO istnieje bardzo dobry system komunikowania o błędach. Interpreter LOGO nie podaje, tak jak ma to miejsce w wielu innych językach programowania, numeru błędu, czy jego identyfikatora, lecz w sposób naturalny sygnalizuje wystąpienie błędnej sytuacji pełnym zdaniem. Poniżej przedstawiono spis komunikatów wydawanych przez interpreter LOGO.

1. *procedura* **DEFINED**

   procedura o nazwie *procedura* została zdefiniowana

2. *procedura* **IS ALREDY DEFINED**

   komunikat ten wysrępuje wówczas, gdy LOGO rozpozna w nazwie *procedura* już zdefiniowaną procedurę (za pomocą **TO...** lub **DEFINE...**)

3. **NUMBER TO BIG**

   dopuszczalny przedział liczbowy został przekroczony

4. **NOT ENOUGH INPUTS TO** *procedura*

   Za mało danych dla *procedura*. Wywołanie procedury wymaga podania większej liczby parametrów wejściowych

5. **I DON'T KNOW TO** *nazwa*

   Nie wiem jak zrobić *nazwa*

   Interpreter nie wie co ma zrobić z nieznaną mu procedurą czy listą o nazwie *nazwa*

6. **I DON'T SAY WHAT TO DO WITH ...**

   Nie powiedziałeś co zrobić z ...

   Częsty błąd kiedy programista nie dość dobrze określił co zrobić z wynikiem danej operacji

7. **TO MANY INSIDE PARANTHESES**

Zbyt dużo nawiasów okrągłych wewnątrz danej procedury

8. *procedura* **DOES NOT OUTPUT TO** *obiekt*

procedura nie daje wyniku potrzebnego *obiekt*
Informacja ta sugeruje użycie procedury o nazwie *procedura* jako parametru wejściowego dla *obiektu*, jednak procedura nie zwraca żadnej wartości.

9. **...IS NOT TRUE OR FALSE**

...nie jest ani prawdą ani fałszem
Instrukcja IF... napotkała operację, której wynik nie jest ani prawdą, ani fałszem

10. **STOPPED!**

Przerwanie pracy interpretera poprzez ingerencję zewnętrzną programisty

11. **YOU ARE AT TOPLEVEL**

Jesteś na najwyższym poziomie
Kontynuacja procedury nie jest możliwa w danym miejscu i chwili

12. **CAN ONLY DO THAT IN A PROCEDURE**

Dana operacja może być wykonana w obrębie procedury, np. operacja OUTPUT

13. *procedura* **DOESN'T LIKE** *nazwa* **AS INPUT**

procedura nie chce *nazwa* jako danej
Błąd nie jest sygnalizowany wówczas, gdy programista pomyli typy danych operacji.

14. **TURTLE OUT OF FILE**

Żółw poza polem. Komunikat pojawia się w topologiach FENCE oraz WINDOW.

15. **CAN'T FIND A THING NAMED** *nazwa*

Użyta zmienna o nazwie *nazwa* nie ma wartości.

16. **MISPLACED...**

Umieszczony w niewłaściwym miejscu.

17. *nazwa* **IS A PRIMITIV**

nazwa jest słowem specjalnym. Błąd ten występuje, kiedy programista próbuje nazwać nową procedurę nazwą zarezerwowaną dla instrukcji w LOGO.

18. **NOT ENOUGH ITEMS IN** *lista*

Za mało elementów w liście *lista*, np. przy użyciu operacji ITEM.

19. **CAN'T DIVIDE BY ZERO**

Dzielenie przez zero nie jest możliwe.

20. **OVERFLOW**

Przepełnienie

21. **I'M HAVING TROUBLE WITH THE DISK**

Stacja dysków nie funkcjonuje dobrze

22. **DISK FULL**

Dyskietka jest całkowicie zapełniona.

23. **FILE ALREADY EXISTS**

Próba zapisania danej procedury (zbioru) na dyskietce kończy się niepowodzeniem, ponieważ użyta nazwa została już użyta poprzednio.

24. **FILE IS WRONG TYPE**

Zbiór nie odpowiada standardowi LOGO

25. **FILE NOT FOUND**

Szukanego zbioru nie ma na dysku.

26. **NO MORE FILE BUFFERS**

Brakuje miejsca na otworzenie dodatkowego bufora pamięci na kolejny zbiór wywoływany z dyskietki.

27. **NOT ENOUGH SPACE TO PROCEED**

Za mało pamięci by kontynuować

Przestrzeń robocza LOGO została całkowicie zapełniona, system nie zawiesza się, lecz nie jest możliwe wykonanie żadnej zdefiniowanej procedury (często pomaga "odśmiecenie" pamięci przez użycie komendy RECYCLE lub skasowanie zbytecznych procedur).

28. **DISK IS WRITE PROTECTED**

Dyskietka jest zabezpieczona przed zapisem.

29. **WITHOUT**

Interpreter napotkał zamknięcie nawiasu okrągłego bez jego otwarcia.

30. *nazwa* **HAS NO VALUE**

Zmienna o nazwie *nazwa* nie posiada żadnej wartości lub posiada ją tylko lokalnie, a programista wywołał ją w sposób globalny.

8. PRZYKŁADY PROSTYCH PROCEDUR W MSX LOGO

8.1. Procedury graficzne

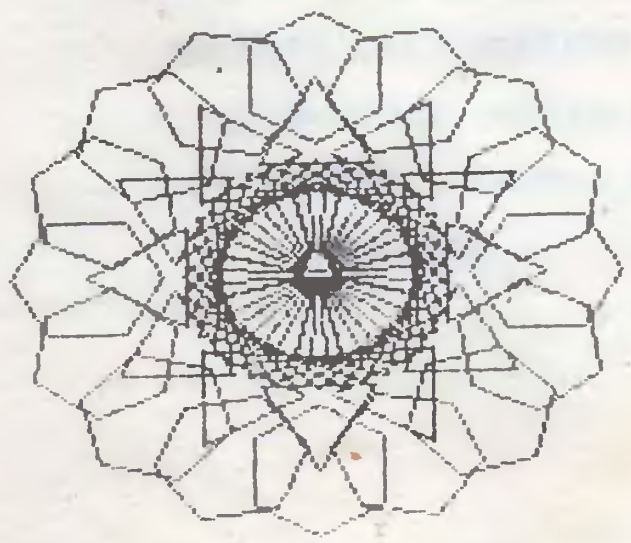
PROCEDURA ROZETA

```
TO ROZETA :promien :ileplatkow :ilekatow
 :bokplatka
REPEAT :ileplatkow [przeskocz :promien p
latek :ilekatow :bokplatka przeskocz - :
promien obrot :ileplatkow]
END

TO PLATEK :ilekatow :bokplatka
RT 90 WIELOKAT :ilekatow :bokplatka LT 9
0
END

TO WIELOKAT :ilebokow :bok
RT 180 / :ilebokow
REPEAT :ilebokow [fd :bok rt 360 / :ileb
okow]
LT 180 / :ilebokow
END

TO PRZESKOCZ :ile
PU FD :ile PD
END

TO OBROT :jaki
RT 360 / :jaki
END
```

W wyniku wywołania:

?ROZETA 0 36 4 25 ROZETA 60 12 3 30 ROZETA 80 16 6 20 ↵

na ekranie powstanie poniższy rysunek.



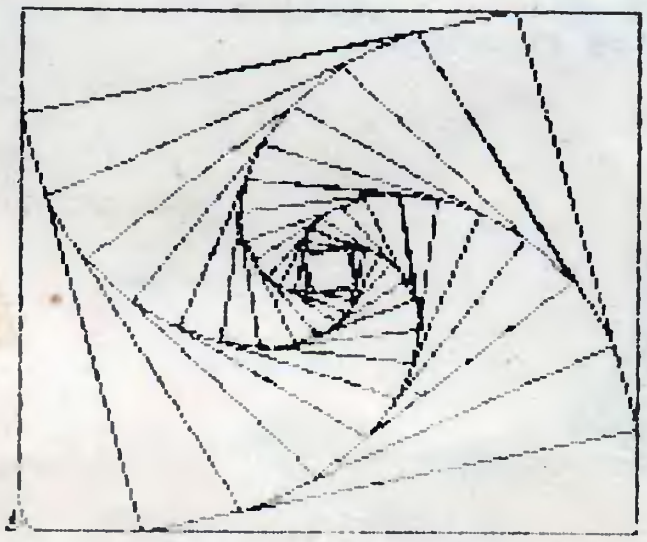
WIRUJACE KWADRATY

```
TO WIRKWADR :bok :wsp
MAKE "poczpoz POS
MAKE "poczkat HEADING
RYSKWADR :bok ARCTAN :wsp / ( 1 - :wsp )
PU SETPOS :poczpoz
SETH :poczkat PD
END

TO RYSKWADR :bok :kat
IF :bok < 10 [stop]
KWADRAT :bok
FD :bok * :wsp
RT :kat
RYSKWADR ( :wsp * :bok ) / SIN :kat :kat
END

TO KWADRAT :bok
REPEAT 4 [fd :bok rt 90]
END
```

W wyniku wywołania :

? **WIRKWADR 160 0,8**

na ekranie powstanie poniższy rysunek.



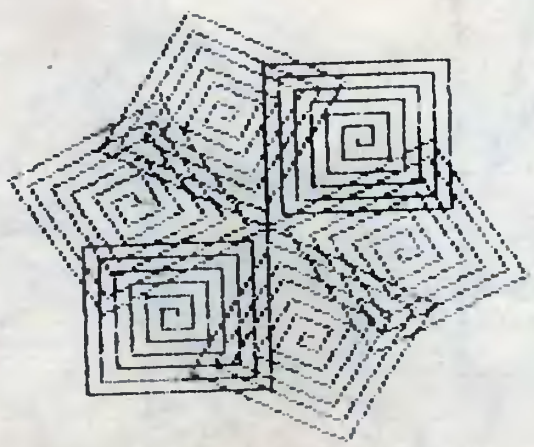
PROCEDURA PEK SPIRAL

```
TO PEKSPIRAL :ilespi :krok :kat :skok
WINDOW CS HT
REPEAT :ilespi [spi :krok :kat :skok rt
360 / :ilespi]
WRAP
END

TO SPI :krok :kat :skok
MAKE "pozpocz POS
MAKE "katpocz HEADING
SPIRALA :krok :kat :skok
PU
SETPOS :pozpocz
SETH :katpocz
PD
END

TO SPIRALA :krok :kat :skok
IF :krok < :skok [stop]
FD :krok
RT :kat
SPIRALA :krok - :skok :kat :skok
END
```

W wyniku wywołania :

? PEKSPIRAL 6 50 90 2

na ekranie powstanie poniższy rysunek.



PROCEDURA ZEGAR

```
TO ZEGAR :godz :min :szybkosc
CS HT PU
REPEAT 12 [fd 50 pd fd 10 pu bk 60 rt 30 ]
]
MAKE "katmin 6 * :min
MAKE "katgodz 30 * :godz + 0.5 * :min
REPEAT 1000 [tyk :szybkosc]
ST
END

TO TYK :szybkosc
PE
SETHEADING :katmin FD 40 BK 40
SETHEADING :katgodz FD 20 BK 20
MAKE "katmin :katmin + 6
MAKE "katgodz :katgodz + 0.5
PD
SETHEADING :katmin FD 40 BK 40
SETHEADING :katgodz FD 20 BK 20
WAIT 3000 / :szybkosc
END
```

Wywołując procedurę ZEGAR z parametrami :godz :min określającymi aktualny czas oraz parametrem :szybkosc dobranym doświadczalnie (ok. 60) na ekranie uzyskamy tarczę zegara z poruszającymi się wskazówkami odmierzającymi czas.



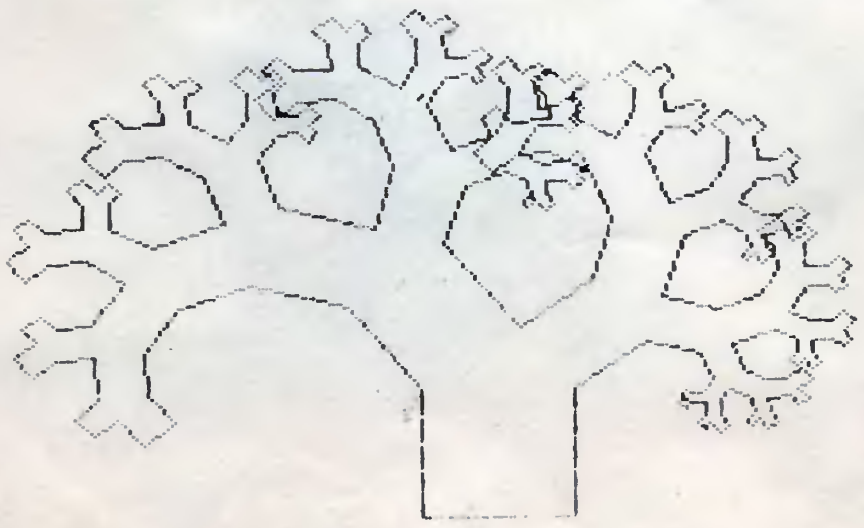
PROCEDURA DRZEWO PITAGORASA (FRAKTAL)

```
TO DRZEWOPITAGORASA :n :bok
MAKE "kt ARCCOS.8
DRZEW01 :n :bok
RT 90 FD :bok BK :bok LT 90
END

TO DRZEW01 :n :bok
IF :n = 0 [pien1 stop]
PIEN FD :bok LT :kt
DRZEW01 :n - 1 :bok * 4 / 5 PU
RT 90 FD :bok * 4 / 5 PD
DRZEW01 :n - 1 :bok * 3 / 5 PU
BK :bok * 4 / 5
LT 90 - :kt BK :bok PD
END

TO PIEN
REPEAT 2 [fd :bok rt 90 pu fd :bok rt 90
 pd]
END

TO PIEN1
REPEAT 3 [fd :bok rt 90]
PU FD :bok RT 90 PD
END

TO ARCCOS.8
OP 36.869898
END
```

W wyniku wywołania procedury:

? DRZEWOPITAGORASA 6 40

na ekranie powstanie poniższy rysunek.



8.2. Procedury obliczeniowe - rekurencyjne

PROCEDURA SILNIA

```
to silnia :n
if :n = 0 [op 1]
if :n = 1 [op 1]
op :n * silnia :n - 1
end
```

W wyniku wywołania:

```
? PR SILNIA 5
```

na ekranie zostanie wyświetlona wartość 5!

120

PROCEDURA POTEGA

```
to potega :baza :wykladnik
if not ( int :wykladnik ) = :wykladnik [pr [wykladnik musi
byc czescia calkowita] toplevel]
if :wykladnik < 0 [make "baza 1 / :baza make
"wykladnik - :wykladnik]
if :wykladnik = 0 [op 1]
if :wykladnik = 1 [op :baza]
op :baza * potega :baza :wykladnik - 1
end
```

procedura ta wyznacza wynik funkcji potęgowania o podstawie :baza i wykładniku :wykladnik.

W wyniku wywołamia:

```
? pr potega 2 5
```

na ekranie zostanie wyświetlona wartość 2^5.

32

PROCEDURA EXP

```
to exp :x :krok
if :krok = 0 [op 1]
op ( potega :x :krok ) / ( silnia :krok ) +
exp :x :krok - 1
end
```

Procedura EXP wyznacza wartość funkcji wykładniczej e^x z zadaną dokładnością (:krok). Jest to procedura rekurencyjna wykorzystująca poprzednie procedury **silnia** i **potega**.

W wyniku wywołania :

? pr exp 4 3

na ekranie zostanie wyświetlona wartość funkcji e^4

23,666666

natomiast w wyniku wywołania

? pr exp 4 5

na ekranie zostanie wyświetlona wartość funkcji e^4

42,866666

a wywołanie

? pr exp 4 10

wyświetli

54,443104

Z otrzymanych wyników widać, że dokładność obliczeń zależy od parametru :krok.

8.3 Procedury przetwarzające struktury listowe

PROCEDURA SREDNIA

```
to srednia :lista
op ( sumuj :lista ) / ( ile :lista )
end

to sumuj :lista
if :lista = [] [op 0]
op ( first :lista ) + sumuj bf :lista
end

to ile :lista
if emptyp :lista [op 0]
op 1 + ile bf :lista
end
```

Proocedura **srednia :lista** wyznacza średnią arytmetyczną liczb znajdujących się w liście. Wykorzystuje ona procedurę **sumuj :lista**, która wyznacza sumę liczb znajdujących się na liście oraz procedurę **ile :lista** wyznaczającą ilość liczb na liście **:lista**.

W wyniku wywołania

```
? pr srednia [ 1 2 3 4 5 6 7 8 9 10 ]
```

na ekranie zostanie wyświetlona średnia arytmetyczna podanych liczb

5,5

PROCEDURA ODWRACANIE LISTY

```
to odwracanielisty :lista
if :lista = [] [op[]]
op put first :lista odwracanielisty bf :lista
end
```

Powyższa procedura odwraca listę tzn. odpowiednio przestawia elementy na liście.

W wyniku wywołania

```
? pr odwracanielisty [1 2 3 4 5]
```

na ekranie zostanie wyświetlona lista odwrócona

```
[5 4 3 2 1]
```

Natomiast w wywołania

```
? pr odwracanielisty [ala ela ola ula]
```

na ekranie zostanie wyświetlona lista odwrócona w postaci:

```
[ula ola ela ala]
```

9. LITERATURA

An Instructin to MSX-LOGO - LCSI Logo Computer Systems Inc.Montreal,Canada H8T1A1-1985r.

Stanisław Waligórski "LOGO na SINCLAIR Spectrum"
cz.1 - dla początkujących
cz.2 - dla zaawansowanych IWZZ W-wa 1987

Ewa Gurbiel,H.Krupnicka, Z.Płoski "Programowanie i LOGO"
Wydawnictwa Uniwersytetu Wrocławskiego
Wrocław 1987r.

Kształty żółwia

setsh 0 - koło
1 - serce
2 - kot
3 - pies
4 - samochód
5 - rakieta
6 - mur
7 - helikopter
8 - lokomotywa
9 - wagon
10-35 kwadrat - prostokąty
36 - żółw
37-59 - żółw pod różnym kątem

Przekazujemy niniejszym Państwu cartridge zawierający interpreter języka MSX LOGO
Wersja MSX LOGO podobna jest do znanych w kraju interpreterów LOGO na mikrokomputery ZX SPECTRUM, AMSTRAD - SCHNEIDER, ATARI, COMODORE.
Istotna różnicę w stosunku do wymienionych dialektów LOGO stanowi rozbudowana grafika żółwia. Obok znanych standardowych komend i operacji graficznych użytkownik MSX LOGO ma do dyspozycji specjalną grupę instrukcji pozwalających na używanie jednoczesne trzydziestu żółwi, z możliwością nadawania im dowolnych kształtów.

cartridge wykonano

w Zakładzie Elekronicznym



na zlecenie: Ministerstwa Edukacji Narodowej
z wykorzystaniem software'u
przekazanego przez Zleceniodawcę

przy współpracy: Zespołu Szkół Elektronicznych
w Zielonej Górze.