301. ZO ZVÁZARM ATARI KLUB BRATISLAVA

METODICKÁ PRÍRUČKA PROGRAMÁTORA

PROGRAMOVACIE JAZYKY

MS BASIC LOGO LISP

PRE POČÍTAČE ATARI

LEN PRE POTREBY ATARI KLUBOV 1988

Úvod

V nasledujúcej príručke, ktorá je určená všetkým užívateľom osembitových osobných počítačov ATARI, ponúkame tri programovacie jazyky. Je to BASIC od firmy MICROSOFT, LOGO od firmy LCSI a LISP. Príručka vznikla v spolupráci AK Bratislava, AK Hradec Králové a AK Teplice. Pri písaní tejto príručky sme nepoužívali originálnu firemnú literatúru, lebo táto nebola v tom čase k dispozícii. Preto autori ani neručia za úplnosť a je len samozrejmé, že táto príručka je len akýsi návod k využívaniu a na ostatné možnosti už prídete sami. MsBASIC sme sa rozhodli dať do príručky z toho dôvodu, že pri práci s jazykom ATARI BASIC nevystačíme s matematickým vyjadrením - nie sú k dispozícii funkcie a dvojitá presnosť a v grafike neexistuje príkaz FILL na vyplnenie plôch farbou. Hlavná výhoda, podľa nás, je práca so súbormi, ktorú MsBASIC umožňuje.

Programovací jazyk LOGO ponúkame tým, ktorí si už dačo o tomto zaujímavom jazyku prečítali, ale zostali stáť na tom, že nepoznajú ostatné príkazy pre narábanie s korytnačkou. Pre zaujímavosť v tomto jazyku boli napísané aj kompilátory.

No a posledný jazyk LISP je zaujímavý svojou metódou stavby programu a následne aj prácou. Má v sebe plnú rekurziu, ale vysvetlenie tejto metódy programovania by presiahlo rámec tejto príručky. Je zvlášť vhodný pre tých, ktorí sa chcú venovať malým expertným systémom s umelou inteligenciou ovšem už na veľkých počítačoch. Túto časť spracoval Dr. Viktor Bajdun z Moskovského inštitútu aplikovanej kybernetiky počas svojej stáže v Bratislave.

Príručku recenzoval Doc. Ing. Štefan Baláži, CSc. z katedry ASR VŠE v Bratislave a spolu s autormi Vám praje veľa úspechov pri využívaní osobných počítačov ATARI.

Redakcia

Programovací jazyk Microsoft BASIC

Väčšina domácich počítačov má už implementovaný v operačnom systéme niektorý dialekt programovacieho jazyka BASIC. Tento jazyk (Beginners All-purpose Symbolic Instruction Code) bol vyvinutý J. Kemenym a T. Kurtzom v šesťdesiatich rokoch v USA s cieľom poskytnúť veľkému počtu užívateľov prístup k počítaču prostredníctvom jednoduchého jazyka.

Až doteraz nebola v oblasti osobných počítačov pre domáce použitie žiadna koordinácia. Je niekoľko hlavných výrobcov veľkých sérií počítačov, ale tieto sú navzájom nekompatibilné - Apple, Atari, Commodore, Sinclair, atď.

Pre vzájomnú kompatibilitu (prenositeľnosť programov) v osobných počítačoch bol vo svete prijatý štandard jazyka a to od firmy Microsoft, ktorý umožňuje na rôznych typoch počítačov s veľmi malými úpravami spúšťať programy naprogramované v tomto štandarde.

Najdôležitejšie vlastnosti tohto jazyka sú:

1. jednoduchosť a z toho vyplývajúce rýchle osvojenie si jazyka
2. efektívnosť pri tvorbe a ladení programov
3. široké možnosti použitia.

Mikrosoft Basic implementovaný na osembitových počítačoch ATARI (800XL,130XE, atď), keďže je to staršia verzia, je mierne odlišný voči štandardu a niektoré príkazy nemôžu byť vykonávané. Pre úplnosť je uvedená tabuľka príkazov a v ďalšom texte budú vysvetlené príkazy jazyka a funkcie, s ktorými je možné narábať.

Príkazy v jazyku MsBASIC nemožno skracovať a je potrebné

prísne dodržiavať syntax jazyka (medzery, atď.). Pri práci programu počítač prípadné chyby neohlási hneď po odoslaní riadku stlačením <RETURN> , ale až pri vykonávaní programu. Toto je určitá nevýhoda, ktorá je ale vyvážená ostatnými vlastnosťami a možnosťami tohoto jazyka.

Po zavedení prekladača do počítača zostáva voľných 21022 bajtov (pri spolupráci s disketovou jednotkou) a môžete pracovať buď v priamom móde alebo programovom móde. Program sa ohlási hlavičkou ATARI 800 BASIC V1.0 (C) 1981 MICROSOFT a znakom pripravenosti > .

V priamom móde sa príkazy vykonávajú hneď po stlačení klávesy <RETURN> . Niektoré príkazy sa ale v priamom móde nemôžu používať, o čom vydá kompilátor chybové hlásenie ≪ ILLEGAL DIRECT ERROR ≫ , resp. majú vedľajšie účinky.

Oproti ATARI BASICu je podstatná zmena MsBASICu v tom, že programátor má k dispozícii väčší počet vstavaných funkcií, môže sa rozhodnúť definovať svoju funkciu a pre vyjadrenie číselných konštánt a premenných môže používať dvojnásobnú presnosť výpočtu, t.j. maximálny zobraziteľný počet miest je 16.

Tabuľka štandardných príkazov jazyka MsBASIC

AUTO	CONT	DELETE
LIST	LLIST	NEW
RENUM	RUN	TRON
ERASE	INPUT	LINE INPUT
READ	DATA	RESTORE
END	FOR ... NEXT	GOSUB
RETURN	GOTO	ON GOTO

IF ... THEN ... ELSE	ON GOSUB	PRINT
PRINT USING	REM	STOP
ABS	ATN	CINT
COS	EXP	FIX
INPUT$	INT	LEFT$
LEN	LOG	RIGHT$
RND	SGN	SIN
SPACES	SPC	SQR
TAB	TAN	TIME

Najčastejšie chybové hlásenia

SUBSCRIPT ERROR - chyba v indexovaní

SYNTAX ERROR IN N - syntaktická chyba

FOR WITHOUT NEXT ERROR - príkaz FOR nemá odpovedajúci NEXT

TYPE MISMATCH ERROR - miešanie typov premenných

OVERFLOW - preplnenie

#n ERROR - podobne ako v ATARI BASICu

Typy údajov

V programe napísanom programovacím jazykom Mickrosoft BASIC sa môžu v zásade vyskytovať dva základné typy údajov:

a) číselné údaje (čísla) - predstavujú číselnú hodnotu z určitej množiny číselných hodnôt;

b) reťazcové údaje (reťazce) - predstavujú postupnosť znakov vybraných zo štandardnej množiny znakov. Reťazcový údaj je vlastne text, ktorý sa skladá z písmen, číslic, prípadne i z ďalších znakov.

V jazyku MsBASIC rozoznávame tieto objekty : konštanty, pre-

menné, polia a funkcie. Spoločnou vlastnosťou objektov je, že predstavujú hodnoty. Premenné môžu byť jednoduché (ktoré sú samostatné premenné a nemajú väzbu s inými premennými) a indexové (ktoré sú prvkami polí). Pole môže za určitých okolností vystupovať tiež ako jediný objekt.

V závislosti od typu hodnoty, ktorú objekt predstavuje, rozlišujeme:

a) číselné objekty

b) reťazcové objekty.

Rozoznávame tri základné číselné objekty:

1) číselné konštanty - sú to racionálne čísla vyjadrené v priamom (desatinnom) alebo exponenciálnom (semilogaritmickom) tvare typu integer, single alebo double precision, floating point, hexadecimálne (s prefixom &).

2) číselné premenné - predstavujú jednotlivú číselnú hodnotu, ktorá môže byť behom spracovania programu menená vykonávaním príkazov. Môže byť jednoduchá alebo indexová, celé čísla integer, s jednoduchou presnosťou (single precision), dvojitou presnosťou (double precision).

3) číselné funkcie - sú to vopred definované algoritmy, ktoré sa často využívajú. Funkcie môžu bať vstavané alebo definované používateľom.

Podobne rozoznávame tri základné reťazcové objekty:

1) reťazcové konštanty - sú postupnosti znakov ohraničené na začiatku a na konci znakom úvodzovky.

2) reťazcové premené- predstavujú jednotlivú reťazcovú hodnotu, ktorá môže byť behom spracovania menená vykonávaním príkazov.

Môže byť jednoduchá (označovaná znakom $) alebo indexová.

3) reťazcové funkcie - vstavané alebo definované používateľom. Vyhodnotením reťazcovej funkcie vznikne jednoduchá reťazcová hodnota, ktorá sa vo výraze dosadí na miesto, kde bola funkcia volaná.

Práca programu

Zdrojový program napísaný v jazyku MsBASIC sa používateľovi javí ako postupnosť riadkov. Program má i štruktúrnu stavbu, ktorá je daná požadovanou funkciou programu.

Program sa skladá z programových jednotiek. Programová jednotka je v určitom zmysle samostatná časť programu, ktorá sa skladá z hlavného programu, vnútornej procedúry a používateľských funkcií.

Riadky tvoriace text zdrojového programu majú určitú formu. Každý riadok začína číslom riadku, ktoré ho v programe jednoznačne identifikuje. Čísla riadkov vyberá používateľ z určitého intervalu prirodzených čísiel buď sám alebo pomocou funkcie AUTO. Krok pre číslovanie sa volí väčší ako jedna, aby bola možnosť vkladať do textu programu ďalšie riadky. Zoradenie riadkov vzostupne podľa čísiel vykoná sám procesor. Číslo riadku predstavuje i návestie riadku, na ktoré sa možno odvolať.

Za číslom riadku nasleduje text príkazu. Každý príkaz začína názvom príkazu vyjadrujúcim mnemotechnicky činnosť príkazu, za ním nasledujú prípadné operandy príkazu. Za textom príkazu je možno na tom istom riadku napísať poznámku. Riadok programu je ukončený znakom konca riadku, ktorý je definovaný ako <RETURN>. Na jednom riadku môže byť i viac ako jeden príkaz. Príkazy od

seba musia byť oddelené znakom :

Príklady programovania

1) definovanie užívateľskej funkcie

Odpovede operačného systému sú podčiarknuté

```
10 X=10:Y=12
20 DEF SQUARE(X,Y)=SQR(X*X+Y*Y)
30 ?SQUARE
15.6205
```

2) skok podľa hodnoty premennej A

```
10 IF A=2 THEN GOTO 10 ELSE GOTO 5
10 ON A GOTO 10,5
```

3) otvorenie kanálu #1 pre vstup

```
OPEN (#1),"D:INVEN.DAT" INPUT
```

4) úprava tlače

```
10 A$="AHOJ":?SPC(5),A$
     AHOJ
10 KO=13450:PRINT USING"### .##":KO
134.50
```

5) tlač premennej I

```
10 FOR I=1 TO 100 STEP 3:?I:NEXT I
1
4
7
.
.
.
```

Zoznam príkazov

Vysvetlivky: n - číslo

op - operácia

pr - premenná

rg - reťazcová premenná

meno - meno súboru

z - periférne zariadenie

pod - podmienka

prk - príkaz

v - logický výraz

a - adresa

meno fun - meno funkcie definovanej používateľom

par - parameter funkcie

1. Aritmetické príkazy, funkcie a operátory

ABS (n)	vyčísľuje absolútnu hodnotu čísla n
ATN (n)	vypočítava arcustangens uhla n
COS (n)	vypočítava cosínus uhla n
EXP (n)	povyšuje základ prirodzených logaritmov (E=2.71828179) na n-tú
INT (n)	zaokrúhlenie čísla n na celé čísla
LOG (n) LOG (op)	vyčísľuje prirodzený logaritmus čísla n alebo operácie op
RANDOMIZE *1	zaistí náhodné počiatočné nasta-

	venie generátora pseudonáhodných čísiel
RND (n) *1	generuje pseudonahodne číslo z z intervalu 0 až n-1
SGN op	zistenie znamienka výsledku operácie op; výsledok je 1, ak výsledok operácie je väčší ako nula, výsledok je -1, ak výsledok operácie je menší ako nula, výsledok je nula, ak výsledok operácie je nula
SIN (n)	vypočítava sínus uhla n
SQR (n)	vypočítava druhú odmocninu čísla n, ktoré musí byť nezáporné
TAN (n)	vypočítava tangens uhla n (sin (n)/cos(n))
<	operátor menší
>	operátor väčší
=	operátor rovný
<=	operátor menší alebo rovný
>=	operátor väčší alebo rovný
><	operátor väčší alebo menší
-	operátor odčítania
^	operátor umocnenia
*	operátor násobenia
/	operátor delenia
+	operátor sčítania

-	operátor odčítania
|	bez odskúšania
&	vyjadruje číslo v hexadecimálnej číselnej sústave
#	operátor na kanále
%	jednoduchá číselná premenná
E	exponenciálne vyjadrené číslo
D	dekadicky vyjadrené číslo
!	celočíselná hodnota

*1 Pseudonáhodné čísla. Pri modelovaní náhodných procesov je v MsBASICu k dispozícii vstavaná funkcia RND. Táto funkcia pri svojom opakovanom volaní programu poskytne postupnosť racionálnych čísiel v intervale od nula do jedna. Pretože z teoretického hľadiska nie je na počítači možné naprogramovať skutočne náhodné čísla, hovoríme o pseudonáhodných číslach. Funkcia RND generuje pri každom vykonaní programu tú istú postupnosť pseudonáhodných čísiel, pretože algoritmus tvorby postupnosti vychádza vždy z rovnakého začiatočného bodu. Táto konvencia je zvolená preto, aby program používajúci funkciu RND poskytoval pri každom spracovaní rovnaké výsledky, čo je nevyhnutné pri ladení programov.

Pri spracovaní pseudonáhodných čísiel sa používa tiež príkaz RANDOMIZE. Má tú funkciu, že zruší konvenciu vytvárania stále rovnakej postupnosti pseudonáhodných čísiel a pre každé vykonanie programu vytvára odlišnú postupnosť. Každé vykonanie programu zahajuje generovanie postupnosti z iného (náhodného) počítačového bodu, ktorý sa získava z hodín reálneho času.

2. Príkazy vstupu/výstupu a práce so súbormi

AT (n1,n2)	určenie súradníc tlače pri výstupe na obrazovku; n1 je súradnica x, n2 je súradnica y
CLOAD	zavádza z kazety do počítača program, ktorý bol uložený pomocou CSAVE a vytvorený MsBASICom, program v pamäti sa zmaže a automaticky sa vykoná OPEN
CLOSE #n	uzatvára datový kanál n
CSAVE	ukladá program z pamäte RAM na kazetu
GET #n,pr	z údajového kanálu n, otvoreného príkazom OPEN sa vyberie 1 bajt a uloží do premennej pr
INKEY$ pr$=INKEY$	práve stisknutá klávesa na klávesnici sa uloží do reťazcovej premennej pr$; ak nie je žiadna klávesa stisnutá, bude pr$ prázdna
INPUT #n,pr1,.....,prn	čítanie údajov z príslušného kanála (otvoreného OPEN) do premenných pr1,....prn
INPUT "TEXT";pr1,... prn	výpis textu na obrazovku (miesto implicitného) a načítanie údajov do premenných
LINE INPUT "TEXT";r$	načítanie riadku z klávesnice do

LINE INPUT r$	reťazcovej premennej r $, znak ; za príkazom potlačí RETURN, na obrazovke sa zjaví ?
LOAD "z:meno"	z periférneho zariadenia z sa do pamäti počítača zavedie program meno
NOTE#n1,n2,n3	pri práci s diskovým súborom, otvoreným na kanáli #n1, ukladá do sektoru číslo n2 a do bajtu číslo n3 v tomto sektore nové údaje alebo z daného sektoru číta údaje
OPEN#n,"z:meno" x	otvorenie datového kanálu n pre vstup alebo výstup údajov, z je zariadenie (D:), meno je meno súboru, x je INPUT pre vstup, OUTPUT pre výstup, UPDATE pre opravu, APPEND pre pripojenie údajov na koniec súboru
PUT #n,n1	výstup z bajtu n1 na výstupný kanál #n, ktorý musí byť otvorený
PRINT pr ?	výstup informácií na obrazovku; premenná, ktorá sa má vypísať je pr; ak nie je určená, vypíše sa prázdny riadok
SAVE "z:meno"	uloženie programu z pamäti RAM

	na zariadenie z pod menom meno; prevádza sa automaticky OPEN
SPC (n)	použitie s PRINT, vypíše n medzier od miesta nastavenia kurzoru a po nich vypíše obsah premennej určenej v PRINT-e
STATUS #n1 pr	číta stav zariadenia otvoreného na kanáli n1 a ukladá ho do číselnej premennej pr. Zistený stavový kód je dostupný v zozname chybových hlásení
TAB (n)	použitie s PRINT na programové riadenie klávesy TAB; konštantný tabelátor, n je celé číslo väčšie ako nula
USING	použitie s PRINT pre formátovanie výstupu; pre výstup sa určuje tzv maska (tvar); PRINT USING "#.#"; pr; povolené znaky:.,+-$ medzera.
VERIFY "z:meno"	porovnáva program v pamäti s programom na zariadení z pod názvom meno
EOF	ukazovateľ konca súboru, je možné sa priamo naň pýtať (IF EOF:..)
KILL "súbor"	ruší zadané diskové súbory
LIST "z:meno",n1-n2	uloží programové riadky od n1 po n2 na priradené zariadenie z: pod

	menom meno; ak z nie je určené ide výpis na obrazovku, ak nie je určené n1,n2 vypíše sa celý program
LOCK "z.meno"	zablokovanie súboru meno na diskete z:
MERGE "z:meno"	pripojí program meno zo zariadenia z: k programu v pamäti počítača (podľa čísiel riadkov sa program zatriedi)
NAME "D:meno1" AS "meno2"	premenovanie súboru meno1 na diskete na súbor s menom2
UNLOCK "z:meno"	uvoľnenie blokovania súboru, ktorý bol zablokovaný príkazom LOCK

3. Príkazy riadenia programu

AFTER (n)	prerušenie behu programu na časový interval zadaný v n (n=1/50s)
CONT	znovuspustenie programu od riadku, ktorý nasleduje po riadku, na ktorom bol program prerušený príkazom STOP,END
END	zastavenie behu programu, riadenie sa predá klávesnici, uzavrú sa všetky údajové kanály, vypnú sa zvukové generátory a zaradí sa

	grafický mód 0
FOR n=n1 TO n2 STEP n ...NEXT	cyklus, vykonávanie príkazov medzi FOR a zodpovedajúcim NEXT až po dosiahnutie koncovej hodnoty, riadi počet prechodov, kde n je číselná premenná, n1 je počiatočná hodnota, n2 je koncová hodnota, n3 je krok. Ak n3=1, krok netreba zadávať.
GOTO n	skok v programe bez návratu na návestie n, n je číselná konštanta, premenná alebo výraz
GOSUB n	návratový skok do podprogramu, n je číselná konštanta, premenná alebo výraz
IF pod THEN prk	ak je podmienka pod TRUE, prevedie sa príkaz pr. Inak sa prejde na ďalší riadok.
IF pod THEN prk1 ELSE prk2	funguje ako IF THEN, ale keď nie je splnená podmienka pod, vykoná sa príkaz prk2
NEXT n	ukončenie alebo pokračovanie cyklu FOR/NEXT v závislosti na hodnote n; počiatočná a konečná hodnota je nastavená vo FOR
ON pr GOTO n1,n2,...	v závislosti od hodnoty premennej pr sa prevedie skok bez návratu

	na príslušné ni (pr=1 skok na n1, pr=2 skok na n2,...)
ON pr GOSUB n1,n2,...	v závislosti od hodnoty premennej pr sa prevedie skok s návratom do príslušného podprogramu
RETURN	ukončenie podprogramu volaného príkazom GOSUB a návrat na inštrukciu bezprostredne nasledujúcu za príkazom GOSUB, ktorý podprogram vyvolal
RUN "z:meno" RUN n	zavedenie a spustenie programu meno zo zariadenia z
STOP odpoveď je BREAK	prerušenie vykonávania programu; uvedie sa riadok, na ktorom bol program prerušený. Nemaže premenné, neuzatvára súbory, predáva riadenie klávesnici. V programe možno pokračovať príkazom CONT

4. Logické operátory

v1 AND v2	logický súčin; v1 a v2 musia byť typu integer, výsledok je TRUE, len ak oba výrazy v1, v2 sú TRUE
NOT n	logická negácia n; výsledok je TRUE, ak n je FALSE, a naopak
v1 OR v2	logický súčet; výsledok je FALSE,

	ak oba výrazy sú FALSE, výsledok je TRUE. ak aspoň jeden z výrazov v1, v2 je TRUE
v1 XOR v2	logický exklusívny OR: výsledok je TRUE, ak výraz v1 alebo v2 je TRUE, ale nie oba súčasne, inak je výsledok FALSE

5. Príkazy pre grafiku

CLS CLS n	zmaže textovú časť obrazovky; ak je zadané číslo n, toto slúži na nastavenie farby a jasu pozadia obrazovky
COLOR n	priradenie farby k určitému registru pre inštrukcie, ktoré za ním nasledujú (PLOT)
FILL n1 n2 TO n3 n4	vyplní miesto medzi dvoma bodmi určenými súradnicami n1=x1,n2=y1, n3=x2,n4=y2 farbou, ktorá sa zadá príkazom COLOR
GRAPHICS n	voľba grafického módu, n je z intervalu 1 až 12
PLOT n1;n2 PLOT n1,n2 TO m1,m2 TO ... pre kreslenie úsečiek	zobrazenie bodu na obrazovke na súradniciach n1=x, n2=y, bod 0,0 je ľavý horný roh obrazovky
POS (n)	určuje horizontálnu pozíciu kurzora

SCRN$ (n1,n2)	priradenie znaku alebo čísla farby na obrazovke zadanej bunke
SETCOLOR n1,n2,n3	uloženie do farebného registra n1 farby n2, ktorá má jas n3. n1 je z intervalu 0 až 4, n2 je z intervalu 0 až 15, n3 je z intervalu 0 až 14

6. Príkazy riadenia pracovnej pamäti

DEL n1 [-n2]	príkaz ruší programové riadky n1 až n2
FRE (0)	zistenie voľného miesta pamäti RAM v bajtoch
MOVE a1,a2,n	presun údajov n z adresy a1 na adresu a2
NEW	vymazanie programu v pamäti a všetkých premenných
OPTION CHR n	rezervovanie určitej oblasti pamäti RAM pre alternatívnu znakovú sadu. n=0 uvoľnenie rezervovanej pamäti, n=1 pre znakový súbor 128 znakov vytvorí oblasť 1024 bajtov, n=2 pre znakový súbor 64 znakov vytvorí oblasť 512 bajtov
OPTION PLM n	vymedzenie miesta v pamäti pre Player-missile grafiku (PMG).n=0 uvoľnenie pamäti,n=1 vyhradzuje

	2048 bajtov PMG s rozlíšením jedného TV riadku, n=2 vyhradzuje 1024 bajtov pre rozlíšením dvoch TV riadkov
OPTION RESERVE (n)	vymedzenie miesta v pamäti pre program v strojovom kóde, n udáva veľkosť v bajtoch
PEEK (n)	udáva dekadickú hodnotu obsahu pamäťového miesta n, ktoré je zadané v desiatkovej sústave
POKE n1,n2	zápis hodnoty n2 do pamäťového miesta n1. n1 je z intervalu 0 až 65535, n2 je z intervalu 0 až 255
VARPTR (n)	vyjadruje adresy premenných alebo grafických oblastí v pamäti

7. Príkazy definovania premenných a funkcií

CCLEAR	nulovanie všetkých číselných a reťazcových premenných
CLEAR	vymazanie všetkých premenných
CLEAR STACK	nulovanie všetkých vstupov zásobníka
COMMON pr1, ... pr n COMMON ALL	špecifikuje premenné spoločné rôznym programovým jednotkám, premenné sú pr1, ... prn alebo všetky
DATA n1,n2, ... nn	obsahuje vstupné hodnoty pre inštrukciu READ, môže byť kdekoľvek

	v programe, umožňuje zápis údajov do programu
DEFINT pr1,pr2 ..... pr1%,pr2%,....	definovanie celočíselnej premennej údaja typu Integer, ak sa nepoužije DEFINT je nutné uviesť za názvom premennej znak % (prefix).Celočíselná premenná môže nadobúdať hodnoty od -32768 do +32768
DEFSNG pr1,pr2 ..... pr1pr2....	definovanie číselnej premennej v jednoduchej presnosti typu single precision; implicitná definícia je bez sufixu alebo prefixu. V jednoduchej presnosti je možné vyjadriť číslo z intervalu od -1.71041E+38 do -2.938737E-39 a od +2.938737E-39 do +1.71041E+38
DEFDBL pr1,pr2, ... pr1 ,pr2 , ...	definovanie číselnej premennej dvojnásobnej presnosti (double precision). Implicitná definícia je so sufixom medzera alebo prefixom D. Presnosť je až 16 platných miest. Je možné použiť aj sufix S pre deklarácia znamienka
DEFSTR pr1,pr2, ... r$1,r$2, ..	definovanie textovej premennej. Implicitný zápis je so sufixom $.
DEFFN funkcia(pr1,pr2,..) v	definovanie vlastnej funkcie používateľa, v je aritmetický výraz

DIM pr(n1 n2 ...)	definovanie číselnej alebo textovej tabuľky. Maximálny počet elementov tabuľky je závislý od dostupnej veľkosti pamäte. Textové premenné nepotrebujú definovanie príkazom DIM.
LET pr=n	priradenie hodnoty n premennej pr premenná môže byť číselná alebo reťazcová; n može byť číselná alebo reťazcová konštanta, premenná alebo výraz
OPTION BASE n	určuje nastavenie minimálnych možných hodnôt indexov, n je 0 alebo 1. Používa sa pred inštrukciou DIM a v programe môže byť použité len raz
READ pr	číta za sebou hodnoty uvedené v príkaze DATA a ukladá ich do premennej pr. Typ premennej uvedenej v DATA musí zodpovedať typu premennej v READ
RESTORE n	určuje, že nasledujúci príkaz READ bude čítať údaje z prvej položky príkazu DATA na riadku n. Ak n nie je zadané, údaje sa berú z prvého DATA v programe

8. Príkazy pre prácu s reťazcami

ASC (r$)	udáva ascii hodnotu prvého znaku reťazca r$. Hodnota je z intervalu 0 až 255
CHR$ (n)	udáva reťazcovú konštantu zodpovedajúcu ascii kódu hodnoty n
INSTR (n,R$1,r$2)	určuje pozíciu výskytu reťazca r$2 v reťazci r$1, prehľadávanie začína od pozície n. V prípade neúspešného prehľadávania je výsledok 0.
LEFT$ (r$,n)	dosadzuje prvých n znakov reťazca r$ do nového reťazca (A$=LEFT$ (r$,n)
LEN (r$) LEN (pr)	zistenie dlžky premennej pr alebo reťazca r$
MID$ (r$,n1,n2)	priradenie n2 znakov reťazca r$ novej premennej od pozície n1 (A$=MID$(r$,n1,n2))
RIGHT$ (r$,n)	dosadzuje posledných n znakov (sprava) reťazca r$ do nového reťazca
STRING$ (n,r$)	priradenie n-krát reťazca r$ novej reťazcovej premennej
STR$ (n)	prevádza číselnú hodnotu n na reťazcovú premennú

VAL (r$)	prevádza číselnú hodnotu, ktorá je súčasťou reťazcovej hodnoty a a stojí na jej začiatku na číselnú hodnotu

9. Príkazy pre prácu so zvukovými kanálmi

SOUND n1,n2,n3,n4 n5	generovanie zvukových tónov n1=kanál z intervalu 0 až 3 n2=výška tónu z intervalu 0 až 255 n3=zafarbenie, skreslenie z intervalu 0 až 14, 10 je čistý tón n4=hlasitosť z intervalu 0 až 15 n5=dĺžka tónu v 1/50 s

10. Príkazy pre ošetrovanie chybových stavov v programe

ERL	premenná, ktorá určuje číslo riadku, v ktorom sa vyskytla chyba
ERR=n	premenná, ktorá slúži pre vlastné ošetrenie chýb v programe, kde n je kódové číslo chyby n=1 bez chyby
ERROR n	generuje chybu zadanú kódovým číslom n
ON ERROR GOTO n	v prípade chyby v programe vyko-

	ná skok na riadok n, kde sa chyba ošetruje (spracuje)
RESUME n	návrat z podprogramu spracovania chýb do hlavného programu (do podprogramu bol skok príkazmi ON ERROR alebo AFTER). n=NEXT - návrat na nasledujúci riadok po výskyte chyby, n=číslo - návrat na riadok s daným číslom. Ak n nie je zadané, je návrat na ten istý riadok, v ktorom sa chyba vyskytla

11. Ostatné rezervované slová a príkazy .

AUTO n1,n2	automatické číslovanie riadkov, n1 je číslo prvého riadku, od ktorého sa má číslovať, n2 je krok. Ukončiť sa môže pomocou klávesy RETURN. Ak už v pamäti je nejaký program, tak číslovanie sa začína od posledného riadku zväčšeného o n. Implicitná hodnota n1=100,n2=10.
REM n '	zaradenie komentárov na zvolené miesto v programe. Komentáre sa zobrazujú pri výpise programu.

RENUM n1,n2,n3	prečíslovanie riadkov v programe n1 = dolná hranica čísla riadku n2 = horná hranica čísla riadku n3 = krok. Implicitné hodnoty sú 10,10,10.
STACK	vyčísli počet položiek v zásobníku časového prerušenia
TIME	určuje numerickú hodnotu reálneho času
TIME$	obsahom tejto premennej je skutočný čas v tvare hh:mm:ss, po zadaní TIME$ funguje ako hodiny
TRON	zapnutie módu TRACE; výpis čísiel práve prevádzaných riadkov programu. Príkaz slúži ako ladiaci prostriedok pre odladenie programu.
TROFF	vypnutie módu TRACE
USR (a n1,n2,...)	spustenie programu v strojovom kóde začínajúceho na adrese a, do zásobníku sa ukladajú parametre ni, ktoré sú nepovinné.

MsBASIC je jednostranne kompatibilný s ATARI BASICom uloženým príkazom LIST. Je asi jeden a polkrát rýchlejší (vo výpočtoch päťkrát), čo ho predurčuje pre použitie v matematických programoch a pri práci s mnohorozmernými tabuľkami.

Programovací jazyk LOGO

Programovací jazyk LOGO bol vyvinutý v USA na Masseachusett Institut of Technology (MIT) Seymorom Paperom. Cieľom bolo dať k dispozícii začiatočníkom v programovaní komunikačný prostriedok medzi počítačom a človekom. V skutočnosti to nie je len jazyk pre deti, ako by niekto predpokladal podľa mylných informácií, lebo pri jeho zostrojení sa vychádzalo z mnohých iných pohľadov, nielen z vynikajúcej grafiky. Preto ani nie je dosť možné vytvoriť program v jazyku BASIC a preložiť ho do LOGA. Pri zostavovaní programov treba využívať všetky možnosti tohoto zaujímavého jazyka.

Význam slova logós v starovekej gréčtine je to, čo v slovenčine vedomosť a v perskom jazyku znamená logika. Filozofia jazyka LOGO spočíva v jeho modulárnosti stavby programu, nakoľko z jednotlivých elementov programátor "stavia" program. Písanie programov, vlastne jeho stavba, je v logických jednotkách realizovaná buď zospodu nahor (bootton-up) alebo zvrchu nadol (top-down).

Na počítačoch ATARI 800XL a 130XE je implementovaný interpréter LCSI ATARI LOGO a po jeho zavedení zostáva k dispozícii 20 kB pracovnej pamäti. Ako dialekt jazyka LOGO je tento najlepší hneď po LCSI APPLE LOGO používaný na počítačoch Apple, s ktorým je plne kompatibilný.

Ako oddeľovač medzi príkazmi a údajmi sa v programovacom jazyku LOGO používa znak medzera, pri viacerých argumentoch sú tieto v hranatej zátvorke oddelené od seba znakom :.

Práca programu

Štandardne údaje do spracovania vstupujú z klávesnice a vy-

stupujú na obrazovku. Niektoré príkazy môžu mať okrem hlavného výsledku i vedľajší efekt.

Ak vznikne potreba programy kopírovať bez nahrávania relatívne dlhého prekladača, je možné použiť kopírovací program napr. SUPERCOPY 35 kB, ktorý bude pracovať v móde 0 (t.j. dlhé medziblokové medzery) alebo kopírovací program CASDUP.

Typy údajov

V jazyku LOGO existujú dva základné typy údajov:

a) jednoduché premenné

b) súbory.

Oba druhy údajov môžu byť číselné alebo textové.

Techniky programovania

Programovací jazyk LOGO umožňuje používať:

1) štrukturálne programovanie
2) úplnú rekurziu
3) editovanie.

Príklady programovania

Odpovede systému sú podčiarnuté.

1) priradenie jednoduchej premennej A a jej vytlačenie

```
≥ MAKE "A "10
≥ PR :A
```

2) priradenie číselného súboru

```
≥ MAKE "C[1 2 3]
```

3) nastavenie textového kurzora

```
≥SETCURSOR [10 10]
```

4) presun korytnačky na súradnice X=50, Y=50

```
≥SETPOS 50 50
```

5) tlačenie adries

```
?TO ADR :MENO :MESTO :PSC :UL
≥PR "
≥PR :MENO
≥PR :MESTO
≥PR :UL
≥PR :PSC
≥PR "
≥END
ADR DEFINED
?ADR [JAN DOBRY] "62656 [BRNO] [1.MAJA 2]
JAN DOBRY
BRNO
1.MAJA 2
62656
```

Zoznam príkazov

Vysvetlivky: chr - znak

pr - príkaz

lst - zoznam

n - číslo

obj - objekt (niekoľko znakov, zoznamov,.....)

lv - logický výraz

prem - premenná

v - výraz

sub - súbor

pr g - program

prv - prvok

1. Aritmetické príkazy a operátory

Volanie	Význam
COS n	kosínus uhla n v stupňoch
INT n	celočíselná časť čísla n
PRODUCT n1 n2	súčin čísiel n1, n2
RANDOM n	generuje pseudonáhodné číslo z intervalu nula až n-1
REMAINDER n1 n2	dáva zvyšok po delení n1/n2
RERANDOM	dáva opakovane postupnosť pseudonáhodných čísiel generovaných príkazom RANDOM
ROUND n	výsledkom je zaokrúhlené číslo n
SIN n	sínus uhla n v stupňoch
SQRT n	druhá odmocnina čísla n
SUM n1 n2	súčet čísiel n1, n2
+	operátor sčítania
-	operátor odčítania
/	operátor delenia
<	operátor menší
>	operátor väčší
=	operátor rovný

*	operátor násobenia
(	ľavá zátvorka
)	pravá zátvorka

2. Príkazy obrazovky

Tieto príkazy sú bezargumentové.

CT	zmazanie textovej obrazovky
CURSOR	dáva aktuálnu hodnotu súradníc textového kurzora
FS alebo klávesa CTRL + F	vyčistí obrazovku a celá obrazovka je použiteľná pre grafiku
SETCURSOR [n1 n2]	nastavenie textového kurzora na súradnice n1=x, n2=y (súradnice sú 0÷31)
SS klávesa CTRL + S	textové okno, rozdeľuje obrazovku na grafickú a textovú časť
TS klávesa CTRL + T	návrat do textovej obrazovky (listing programov)

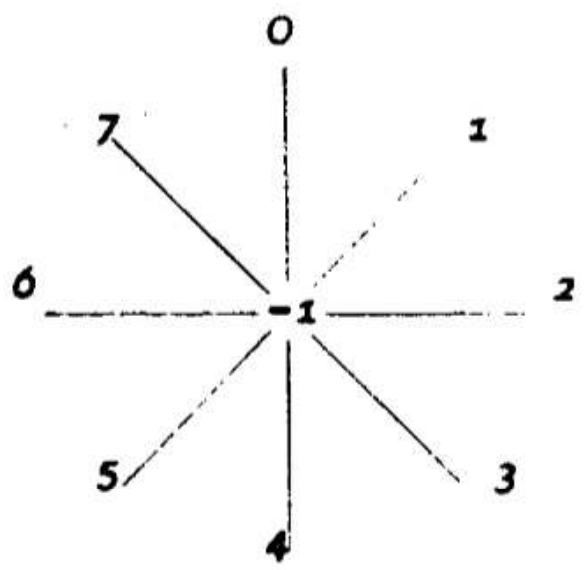
3. Príkazy definovania a editovania

EDIT "lst ED "lst	spustenie editora pre uvedený zoznam
EDNS	editácia premenných
TO meno:prem1:prem2.....:prem n	začiatok programu; zadáva sa meno programu a premenné, ktoré bu-

CTRL +↑	pohyb kurzoru hore
CTRL +↓	pohyb kurzoru dole
CTRL+1	zastavenie výpisu na obrazovku, opätovným stlačením výpis pokračuje
CRTL+A	nastavenie kurzora na začiatok riadku
CTRL+CLEAR	výmaz riadku od pozície kurzora do konca riadku
CRTL+DEL	výmaz znaku pod kurzorom
CTRL+E	nastavenie kurzora na koniec riadku
CTRL+INSERT	vloženie znaku v mieste kurzora
CTRL+U	odroluje text s kurzorom uprostred obrazovky
CTRL+V	v móde editora posúva kurzor na ďalšiu stranu obrazovky
CTRL+W	vracia späť predošlú stranu obrazovky v edit móde
CTRL+X	nastaví kurzor na začiatok editovaného programu
CTRL+Y	tlač posledného napísaného riadku
CTRL+Z	nastaví kurzor na koniec editovaného programu
DELETE	vymaže znak vľavo od kurzora
ESC	ukončí prácu editora
RETURN	odošle napísaný riadok, príkaz

	dú po vyvolaní programu naplnené hodnotami; začiatok príkazovej definície

4. Príkazy vstupu a výstupu

CATALOG "chr:	vylistuje obsah katalógu, chr=C pre kazetu, chr=D pre disketu
ERF "chr:prg	výmaz súboru zo zadaného periférneho zariadenia
JOY (n)	dáva hodnotu joysticku, kde n je z intervalu 0 až 3
JOYB (n) 	dáva TRUE, ak je zapnutý spínač pre niektorý smer alebo TRIGGER
KEYP	dáva hodnotu TRUE, ak je stlačená nejaká klávesa
LOAD "chr:prg	zavedenie programu z periférneho zariadenia, chr=C kazeta, chr=D disketa
PADDLE (n)	dáva hodnotu paddle, kde n je z intervalu 0 až 3
PADDLEB (n)	dáva hodnotu TRUE, ak je zapnutý

	tý spínač pre niektorý smer alebo TRIGGER (platí pre paddle)
PRINT obj PR obj	tlač objektu
READCHAR prem READCHAR "meno RC prem RC "meno	načítava znak zo zvolenej periférie, implicitne z klávesnice
READLIST prem READLIST "meno RL prem RL "meno	načítava súbor znakov zo zvolenej periférie, implicitne jeden riadok
SAVE "chr:prg	zápis programu na periférne zariadenie; chr=C kazeta, chr=D disketa
SETREAD "chr:prg SETREAD [prg1 ... prgn]	číta a upravuje údaje pre príkazy RC a RL z priradenej periférie
SETWRITE "chr:prg SETWRITE [prg1 ... prgn]	zápis údajov na priradené periférne zariadenie
SHOW obj	ako PRINT
TYPE obj	ako PRINT, ale bez pohybu kurzora

5. Príkazy pre slová, zoznamy, súbory

ASCII chr	dáva ascii kód znaku chr
BUTFIRST sub BF sub	vyberá zo zoznamu (súboru) všetky prvky okrem prvého
BUTLAST sub BL sub	vyberá zo zoznamu (súboru) všetky prvky okrem posledného

CHAR n	dáva znak od hodnoty ascii kódu n
COUNT obj	udáva počet článkov objektu
EMPTYP obj	dáva hodnotu TRUE, ak je objekt prázdny
EQUALP obj1,obj2	dáva hodnotu TRUE, ak sa obj1 rovná obj2
FIRST sub	vyberá prvý prvok zo súboru
FPUT sub:prv	pripojí jeden prvok na začiatok súboru
LAST sub	vyberá posledný prvok zo súboru
LIST obj1 obj2	vylistuje obsah označených zoznamov
LISTP obj	dáva hodnotu TRUE, ak je objekt listom
LPUT sub:prv	pripojí jeden prvok na koniec súboru
MEMBERP obj lst	dáva hodnotu TRUE, ak je objekt v liste
NAMEP prem	určuje, či existuje zadaná premenná
NUMBERP prem	určuje, či parameter je jednoduchá číselná premenná
SENTENCE obj1 obj2 SE obj1 obj2	spojí dve premenné do súboru
WORD prem1 prem2	spája dve jednoduché premenné do jednej jednoduchej premennej
WORDP prem	určuje, či parameter je jednodu-

	chá premenná

6. Príkazy riadenia programu

COND n	dáva hodnotu TRUE, ak nastala podmienka označená prípadovým číslom, n je z intervalu 0 až 21
IF v[pr1] [pr2]	ak je výraz v pravdivý uskutoční sa pr1, ak nie je, uskutoční sa pr2, ak je zadaný, inak nastane prechod na nový riadok
OUTPUT obj OP obj	umiestňuje hodnotu do pracovnej pamäti
REPEAT n[pr]	cyklus - opakuje príkaz pr n-krát
RUN lst	spustenie programu (program sa dá spustiť i zadaním jeho mena a parametrov)
STOP	zastavenie behu programu
WAIT n	prerušenie behu programu na n/50 s, napr. n=150=3s
WHEN n	príkaz nie je odskúšaný

7. Logické operátory

AND lv1 lv2	dáva hodnotu TRUE, ako oba lv sú true (pravda)
NOT lv	dáva hodnotu TRUE, ak lv je FALSE (nepravda) a opačne

OR lv1 lv2	dáva hodnotu TRUE, ak jeden výrazov lv1, lv2 je true

8. Systémové príkazy

.CALL n	skok do podprogramu v strojovom kóde na adresu n
.DEPOSIT n	ako POKE v jazyku BASIC
.DOS	odovzdanie riadenia DOS-u
.EXAMINE n	ako PEEK v jazyku BASIC
NODES	dáva počet voľných bajtov v užívateľskej pamäti RAM
.PRIMITIVES	výpis všetkých rezervovaných slov jazyka
RECYCLE	zväčšenie pamäti, reorganizácia pracovnej oblasti

9. Príkazy grafiky s korytnačkou

ASK n[pr]	prevedenie príkazu pr s korytnačkou n
BACK n BK n	posun korytnačky o n krokov vzad
BG	dáva hodnotu farby pozadia
CLEAN	zmazanie grafickej obrazovky, korytnačka zostáva na svojom mieste
CS	zmazanie grafickej obrazovky, korytnačka sa nastaví do východzej polohy (0,0)

COLOR	dáva hodnotu farby korytnačky
DOT [n1 n2]	urobí bod na súradniciach n1 (x), n2 (y), ale korytnačku tam nepresunie
EACH [obj]	všetky korytnačky vykonajú povely z objektu
FORWARD n FD n	presun korytnačky o n krokov vpred
EDSH n	vyvolanie shapeeditora pre vlastný tvar korytnačky, n je z intervalu 0 až 15
GETSH n1 ... n16	dáva súbor 16 čísiel, ktoré vyjadrujú tvar vlastného grafického kurzora, ni je z intervalu 0 až 15, odpoveďou je číslo z intervalu 0 až 255
HEADING	dáva uhol korytnačky
HOME	presun korytnačky do východzej polohy (0,0), zastavenie posunu, nastavenie uhlu 0
HT	zmiznutie korytnačky
LEFT n LT n	otočenie korytnačky vľavo o n stupňov
OVER n1 n2	udáva kód, ktorý symbolizuje kolíziu medzi číslom korytnačky a číslom pera
PC n	udáva farebnú hodnotu pera, n je z intervalu 0 až 2

PE	mazacie pero
PEN	dáva hodnotu, aké pero je nastané
PENDOWN PD	nastaví pero na písanie
PENUP PU	zdvihne pero, korytnačka nebude kresliť
PN	dáva číslo práve použitého pera
POS	dáva hodnotu súradníc korytnačky
PUTSH n1[n21 ..... n216]	vpisuje tvar vlastného grafického kurzora (nie korytnačky), n1 je číslo grafického kurzora z intervalu 1 až 15, n2 vyjadruje tvar, je to 16 čísiel, kde každé číslo z n2i je z intervalu 0 až 255
PX	inverzné pero
RIGHT n RT n	otočenie korytnačky o n stupňov vpravo
.SCRUNCH	dáva hodnotu mierka súradníc
SETBG n	zmena farby pozadia, n je z intervalu 0 až 127
SETC n	zmena farby korytnačky, n je z intervalu 0 až 127
SETH n	nastavenie uhlu korytnačky
SETPC n1 n2	nastavenie farby registrov (ako SETCOLOR v jazyku BASIC), n1 je z intervalu 0 až 2, n2 je farba z intervalu 0 až 127

SETPN n	zmena pera (ako COLOR v jazyku BASIC), n je z intervalu 0 až 2
SETPOS n1 n2	presun korytnačky na zadané súradnice
.SETSCR n1 n2	nastavenie mierítka súradníc (rozpätia obrazovky)
SETSP n	nastavenie rýchlosti posunu, n je z intervalu -199 až 199, ak je n=0, korytnačka stojí
SETSH n	nastavenie iného grafického kurzora ako korytnačka, n=0 pre korytnačku, n je z intervalu 1 až 15 pre iný kurzor, ktorého tvar je možné si nadefinovať (príkazom EDSH). Pomocné kurzory sú vždy korytnačky.
SETX n	pohyb korytnačky horizontálne do pozície n
SETY n	pohyb korytnačky vertikálne do pozície n
SHAPE	dáva hodnotu nastaveného kurzora
SHOWNP	dáva hodnotu TRUE, ak je korytnačka na popísanej polohe
SPEED	dáva hodnotu rýchlosti posunu korytnačky
ST	korytnačka sa objaví
TELL n	práca s pomocnými korytnačkami, n=0 je hlavná korytnačka, n=1 až

	3 sú pomocné: všetky nasledujúce povely sú priradené uvedenej korytnačke
TOUCHING n1 n2	udáva kód kolízie korytnačiek n1. n2
WHO	dáva číslo aktivovanej korytnačky od 0 do 3
WINDOW	okno - korytnačka zmizne hore, ale dole sa neobjaví
WRAP	nekonečná obrazovka - korytnačka hore zmizne a dole sa objaví
XCOR	dáva hodnotu súradnice x korytnačky
YCOR	dáva hodnotu súradnice y korytnačky

10. Príkazy riadenia pracovnej pamäte

ERASE [prg1 .... prgn] ER [prg1 .... prgn]	výmaz zadaných programov
ERALL	výmaz všetkých programov a premenných
ERN prem	výmaz zadanej premennej
ERNS	výmaz všetkých premenných
ERPS	výmaz všetkých programov, príkazov
PO [prg1 ... prg2]	vylistovanie zadaných programov, príkazov
POALL	vylistovanie všetkých programov

	a premenných
POD	príkaz nie je odskúšaný
PODS	príkaz nie je odskúšaný
PONS	vylistovanie názvov a hodnôt všetkých premenných
POPS	vylistovanie všetkých definovaných premenných
POTS	vylistovanie programov zadaných príkazom TO

11. Príkazy definovania premenných

MAKE prem	priradenie premennej, ktorá môže byť jednoduchá číselná, jednoduchá textová, číselný súbor, textový súbor (ako LET v jazyku BASIC)
THING prem : prem	dáva premennú, na ktorú sa vzťahuje názov pred jej ďalším spracovaním

12. Príkazy pre prácu so zvukovými kanálmi

SETENV n1 n2	definovanie zvuku tónu (dozvuku), n1 udáva čistý alebo skreslený tón, n2 udáva rýchlosť dozvuku
TOOT n1 n2 n3 n4	nastavenie zvuku n1 - línia zvuku (0-čistý tón, 1-skreslený tón)

	n2 - frekvencia (číslo väčšie ako 62) n3 - hlasitosť (číslo z intervalu 0 až 15) n4 - dĺžka znenia (číslo z intervalu 0 až 255)

13. Ostatné rezervované slová

BREAK klávesa BREAK	prerušenie činnosti programu
END	ukončenie programu, ukončenie definície začínajúcej príkazom TO
FALSE	špeciálne slovo, ktoré označuje výsledok logickej operácie (nepravda); je možné sa spýtať, či je výsledok FALSE
TRUE	špeciálne slovo, ktoré označuje výsledok logickej operácie (pravda); je možné sa spýtať, či je výsledok TRUE

14. Funkčné klávesy

ATARI 	inverzné zobrazenie
CTRL + →	pohyb kurzoru vpravo
CTRL + ←	pohyb kurzoru vľavo

SHIFT+DEL (BACK SPACE)	vymaže riadok, v ktorom je kurzor
SHIFT+INSERT	vloží prázdny riadok na pozíciu kurzora

Programovací jazyk Lisp.

Programovací jazyk LISP 1.5 bol vyvinutý v roku 1959, má originálnu koncepciu a bol určený pre spracovanie zoznamov.

V roku 1963 bol zdokonalený pod označením LISP 2. a čerpal aj z prvkov jazyka ALGOL 60. Je vhodný pre heuristické programovanie, algebraické operácie, kybernetiku, matematickú logiku, dokumentácie, numerické výpočty a informatiku.

Jazyk LISP je charakterizovaný ako funkcionálny jazyk. Toto označenie vyjadruje skutočnosť, že všetky operácie uskutočňované v programovacom jazyku LISP sa realizujú pomocou funkcií. Termín program bude v ďalšom texte označovaný ako postupnosť foriem. Forma predstavuje ľubovoľnú premennú alebo volanie funkcií. Každé volanie funkcií má tvar

(f-výraz argument ..... argument)

kde f-výraz môže byť meno funkcie (vytvorenej používateľom alebo zabudovanej v jazyku), LAMBDA alebo NLAMBDA výrazom, SUBR alebo NSUBR výrazom. Argument môže byť i formou.

Okrem tvaru zápisu funkcií je pre jazyk LISP charakteristická i štruktúra programu a spôsob jeho vykonávania. Každý program je vždy tvorený postupnosťou príkazov, doplňovaný prípadnými vstupnými údajmi. Spracovanie programu prebieha tak, že sa každý príkaz bezprostredne po jeho načítaní vykoná. Pri programovaní v jazyku LISP neexistujú fázy prekladu a zostavenia, program sa okamžite vykonáva. Popísaný spôsob vykonávania programu určuje v zásade interpretačnú povahu jazyka LISP.

Nekonvenčnosť jazyka LISP pozostáva z toho, že ten istý identifikátor môže byť v programe používaný na označenie objektov rôz-

nych tried. Tak môže napr. identifikátor FN označovať premennú, funkciu alebo návestie. Význam identifikátora sa vždy odvodzuje z kontextu jeho použitia, ale táto viacznačnosť nevedie k nejednoznačnosti.

V závislosti od metódy vyhodnocovania argumentov pri volaní všetkých funkcií v LISPe, či už zabudovaných alebo vytvorených používateľom sa tieto rozdeľujú na dva základné typy:

- funkcie volané významom (typ CBV)
- funkcie volané menom (typ CBN).

Pred volaním funkcie typu CBV sa jej argument vyhodnocuje. Funkcie typu CBN si argument musia vyhodnotiť samy.

Podľa toho, ako funkcie typu CBV odovzdávajú argumenty, môžeme použiť funkcie QUOTE a makrosymbol '. Napr. výraz

(f-výraz 'argument) je ekvivalentný s výrazom

(f-výraz (QUOTE argument)).

Všetky LAMBDA alebo SUBR výrazy sú typu CBV a všetky NLAMBDA alebo NSUBR výrazy sú typu CBN.

Práca programu.

Štandardne vstupujú údaje do spracovania z klávesnice a vystupujú na obrazovku. Používateľ si môže tieto preddefinované hodnoty zmeniť.

Každá funkcie (okrem GO) má výsledok. Niektoré funkcie (napr. PRINT) môžu mať okrem hlavného výsledku ešte vedľajší efekt.

Typy foriem.

V jazyku LISP sa rozlišujú tri typy foriem:

1. premenná -vyjadruje sa nečíselným atomickým symbolom. Premenná je viazaná na hodnotu príslušného argumentu funkcie. Tento argument sa pokladá za výsledok vyhodnotenia premennej. Takto vyhodnoteným premenným sa hovorí dynamické. Ak majú premenné priradenú hodnotu bez ohľadu na aktiváciu funkcií, označujú sa ako statické premenné. Niektoré premenné majú v jazyku LISP štandardne stanovený statický charakter a priradenú implicitnú hodnotu. Najdôležitejšie statické premenné sú:

   - T hodnotou je T
   - NIL hodnotou je NIL.

   Atómy s priradenou konštantnou hodnotou nie je možné používať ako dynamické premenné, lebo ich vyhodnotenie vždy dáva implicitnú hodnotu.

2. zápis funkcie -je najdôležitejší typ formy. Každý zápis funkcie obsahuje určenie použitej funkcie a príslušný počet foriem, ktorých hodnoty budú použité ako argumenty pri aplikácii tejto funkcie. Vyhodnocovanie zápisu funkcie prebieha v dvoch etapách:

   a) vyhodnotenie formy argumentov

   b) na vyhodnotené argumenty sa aplikuje funkcia.

   Výsledná hodnota je celkovou hodnotou zápisu funkcie.

3. konštanta -je forma, ktorá má hodnotu pevnú a nezávislú na argumentoch funkcie. Základným druhom konštánt sú čísla (tzv. číselné atómy). Ak má byť hodnotou konštanty ľubovoľný iný (nečíselný) S-výraz, je nutné vytvoriť z S-výrazu tzv. Q-formu. Vyhodnotenie konštanty možno zhrnúť do týchto pravidiel:

   a) hodnotou čísla je číslo samo

   b) hodnotou Q-formy je S-výraz, ktorý je uvedený za atómom QUOTE v danej Q-forme.

Najčastejším prípadom konštanty v jazyku LISP je Q-forma. Tvorí

ju vždy zoznam dvoch elementov, ktorý je pri vyhodnotení tejto Q-formy jej hodnotou. Q-forma patrí medzi tzv. špeciálne formy jazyka LISP.

Typy údajov.

V jazyku LISP existujú dva základné typy údajov: atómy a dvojice identifikátorov.

Atómom sa nazýva ľubovoľná postupnosť symbolov, ktorá neobsahuje nasledovné symboly: ((ľavá zátvorka),) (pravá zátvorka), medzera, ohraničenia oddeľovačov. Atómom nie je ani bodka ohraničená medzerami. V tomto prípade sa označuje ako oddeľovač. Atómy sa rozdeľujú na symboly a čísla (celé a desatinné).

Dvojica identifikátorov má tvar:

(ľavá časť . pravá časť)

kde ľavá i pravá časť môžu byť atómy alebo znovu dvojice identifikátorov. Hĺbka vloženia môže byť ľubovoľná. Príklad dvojice identifikátorov

((Toto.je).dvojica-identifikátorov)

Častým prípadom dvojice identifikátorov je zoznam. Všeobecne možno definovať zoznam ako usporiadanú n-ticu výrazov oddelených medzerou, uzatvorených do zátvoriek. Zoznam sa teda nazýva dvojica identifikátorov, v ktorom všetky pravé časti sú dvojice identifikátorov alebo atóm NIL. Atóm NIL je zvláštny atóm, ktorý slúži na definovanie prázdneho zoznamu, t.j. NIL = (). V tom prípade sa dvojica identifikátorov zobrazuje zoznamom notácií bez zátvoriek. Prevod S-výrazov z notácie dvojice identifikátorov do zoznamovej notácie možno vykonávať úplne automaticky použitím nasledujúcich pravidiel:

1. Ak je druhou časťou zvolenej dvojice identifikátorov atóm NIL, nahradíme ho prázdnym zoznamom.
2. Ak je druhou časťou zvolenej dvojice identifikátorov atóm rôzny od NIL, potom v tejto dvojici identifikátorov upravíme podľa týchto pravidiel iba prvú časť a druhú časť necháme v pôvodnom tvare.
3. Ak začína druhá časť zvolenej dvojice identifikátorov zátvorkou, potom vykonáme úpravu spočívajúcu v nahradení bodky upravovanej dvojice identifikátorov medzerou a vo vynechaní úvodnej otváracej a zodpovedajúcej zatváracej zátvorky druhej časti dvojice identifikátorov.
4. Proces začneme na dvojici identifikátorov najnižšej úrovne, ktorá je prvá sprava a postupne prechádzame do ľavej časti S-výrazu.

V skratke by sme postup mohli zhrnúť takto: všetky pravé atómy NIL sa zamenia za (), potom sa vyčiarknu všetky bodky, ktoré stoja pred ľavými zátvorkami spolu so zodpovedajúcimi pravými a ľavými zátvorkami. Príklad dvojice identifikátorov:

(Toto.(je.(5.(prvkov.(listu.NIL)))))

v zozname notácií vyzerá táto dvojica identifikátorov takto

(Toto je 5 prvkov listu)

S-výraz sa volá dvojica identifikátorov alebo atóm.

Rekurzia a iterácia.

Na opísanie vyhodnotenia výrazov rôznej hĺbky sa v jazyku LISP používa rekurzia, hoci jazyk má možnosť i iteratívneho programovania. V dalšom texte je uvedený príklad výpočtu N! pomocou rekurzie (príklad a) a iterácie (príklad b).

a)
```
(DEFINEQ FACTREC (LAMBDA (N) (COND
  ((EQ N 0) 1)
  (T (* N (FACTREC (SUB N 1)))) )))
```

b)
```
(DEFINEQ FACTITER (LAMBDA (N) (PROG (M)
  (SETQ M 1)
LOOP (SETQ M (* M N))
  (SETQ N (SUB N 1))
  (COND ((EQ N 0) (RETURN M)))
  (GO LOOP) )))
```

Ako je vidieť z príkladov, rekurzia má kratší zápis a lepšie odráža algoritmus výpočtu.

Príklad programovania.

V príklade PERM sa podľa zadaného zoznamu atómov zostrojí zoznam jeho elementov. Program obsahuje tri vzájomne rekurzívne funkcie PERM, PTIMES, PINSERT a súčasne ukazuje volanie týchto funkcií. Odpovede systému sú podčiarnuté.

Vysvetlenie funkcií:

- *hlavná funkcia PERM určuje zoznam všetkých definícií z elementov zoznamu AL;*
- *funkcia PTIMES dosadzuje element A na všetky možné miesta v podzoznamoch zoznamu PL;*
- *funkcia PINSERT dosadzuje element A na všetky možné miesta v zozname LP a vracia zoznam dosiahnutých vsuniek.*

LISP

```
(DEFINEQ PERM (LAMBDA (AL) (COND
```

```
  ((EQ (CDR AL) NIL) (LIST AL))
  (T (PTIMES (CAR AL) (PERM (CDR AL)))) )))
```

(LAMBDA (AL) ...

...)

LISP

```
(DEFINEQ PTIMES (LAMBDA (A PL) (COND
  ((EQ PL NIL) PL)
  (T (APPEND (PINSERT A (CAR PL))  (PTIMES A (CDR PL)))) )))
```

(LAMBDA (A PL) ...

...)

LISP

```
(DEFINEQ PINSERT (LAMBDA (A LP RP) (COND
  ((EQ LP NIL) (LIST (CONS A RP)))
  (T (CONS (APPEND LP (CONS A RP))
           (PINSERT A (CDR LP) (CONS (CAR LP) RP)) )) )))
```

(LAMBDA (A LP RP) ...

...)

LISP

```
(PINSERT 'A '(B C))
```

((A B C) (B A C) (B C A))

LISP

```
(PTIMES 'A '((B C) (C B)))
```

((A B C) (B A C) (B C A) (A C B) (C A B) (C B A))

LISP

(PERM '(A B C))

((A B C) (B A C) (B C A) (A C B) (C A B) (C B A))

Zoznam funkcií zabudovaných v systéme.

Vysvetlivky:

at - atóm

se - s-výraz

fe - f-výraz

form - forma

nm - meno (nečíselný atóm)

lst - zoznam

n - číslo

dp - dvojica identifikátorov

fn - meno funkcie

p-form - forma, ktorá môže byť používaná ako predikát (ľubovoľná forma okrem GO a RETURN).

1.Riadiace funkcie

Typ	Volanie	Význam
CBN	(APPLYx fe se ... se)	Zámena f-výrazov fe na argumenty se ... se
CBV	(EVAL form)	Vypočítava formu form
CBN	(AND form ... form)	Postupne (zľava doprava) vypočítava formy, pokiaľ nebude dosiahnutý NIL. Ak je výsledok všetkých foriem rôzny od NIL, výsledkom bu-

		de výsledok poslednej vypočítanej formy
CBN	(OR form ... form)	Postupné vypočítavanie formy, kým nebude dosiahnutý výsledok rôzny od NIL. Ak je výsledok všetkých foriem NIL, potom výsledkom OR bude NIL.
CBN	(COND alt ... alt)	Podmienený výraz. Každá alternatíva má tvar: (p-form form ... form) Postupne sa vyhodnocujú p-formy všetkých alternatív, kým sa nedosiahne výsledok rôzny od NIL. Vtedy sa vyhodnocujú ostatné formy zo zodpovedajúcej alternatívy a nasleduje výstup z COND, pričom výsledok je vždy výsledok poslednej vyhodnotenej formy.
CBN	(PROG lv form ... form)	Iterácia. Každej lokálnej premennej zo zoznamu lv sa priraďuje NIL. Potom sa postupne vyhodnocujú formy form: - ak form je atóm, vtedy sa nevyhodnocuje a reprezentuje sa ako návestie v GO; - ak form má tvar (GO lbl), nasleduje prechod na formu, ktorá nasleduje za značkou lbl; - ak form má tvar (RETURN rform),

		vtedy nastane výstup z PROG na návestie rform; - v ostatných prípadoch sa forma form jednoducho vyhodnotí. Tak isto sa spracúvajú GO i RETURN vnútri CONDu, stojaceho v PROGu. Ak chýba operátor RETURN, výstup nastane po vyhodnotení poslednej formy.
CBN	(GO lbl)	Prechod na návestie lbl (iba v PROGu!)
CBN	(RETURN rform)	Výstup z PROG.Možný tiež pri výstupe z režimov ERR a BREAK, pričom rform sa stanovuje výsledkom z formy, z ktorej nastal výstup.
CBN	(PROGN form ... form)	Postupne sa vyhodnocujú formy form. Vracia sa význam poslednej formy.
CBN	(BREAK se)	Prerušenie. Výstup do režimu ERR. Návrat po BREAK - RETURN.
CBN	(BAKTRACE)	Výstup zo zoznamu nedokončených funkcií. Výsledkom je NIL.
CBV	(GETD fn)	Výsledok - zoznam, ktorý je telom funkcie fn.
CBN	((LAMBDA lv form) aform ... aform)	λ - výraz typu CBV. Najprv sa postupne vyhodnocujú argumenty aform ... aform. Dosiahnuté výsledky sa potom spájajú so zodpovedajúcimi premennými zo zoz-

		namu lv. Ako výsledok sa vracia význam formy form.
CBN	((NLAMBDA lv form) se ... se)	λ-výraz typu CBN. Zoznam nevyhodnotených argumentov (se ... se) spojený s prvou lokálnou premennou zo zoznamu lv. Výsledok je význam formy form.
CBN	((MACRO lv form) se ... se)	Makroinštrukcia (definovanie makra)
CBN	((SUBR . ◎) aform ... aform)	Tento výraz predstavuje jadro zabudovanej funkcie (dostaneme ju len cez GETD). Výsledok - označenie funkcie požitej v argumente aform, ktorý sa priebežne vypočítava
CBN	((NSUBR . ◎) se .. se)	To isté ako SUBR, ale argumenty sa nevypočítavajú.

2. Definovanie konštánt, premenných a funkcií

CBV	(SET nm se)	Výsledok je se. Atóm nm bude menom se.
CBN	(SETQ nm form)	To isté ako SET, ale s formou.
CBV	(DEFINE fn fe)	Atóm fn bude menom funkcie fe.
CBN	(DEFINEQ fn fe)	To isté ako DEFINE, rozdiel je len v type.
CBV	(RPLACA dp se)	Nahradenie ľavej časti dvojice

CBV	(RPLACD dp se)	identifikátorov dp za se. Ako RPLACA, ale pravú časť.

3. Funkcie zoznamov

CBV	(CAR dp)	Ľavý element dvojice identifikátorov.
CBV	(CDR dp)	To isté čo CAR, ale pravý element
CBV	(CONS lse rse)	Výsledok je dvojica identifikátorov s ľavým elementom lse a pravým rse
CBV	(LIST se ... se)	Zoznam elementov
CBV	(APPEND lst ... lst)	Konkatenácia zoznamov
CBV	(ASSOC at alst)	Prvý podzoznam zoznamu alst, ktorého prvý element je at
CBV	(LAST lst)	Posledný element zoznamu lst
CBV	(LENGTH lst)	Dĺžka zoznamu lst
CBV	(MEMBER at lst)	Zostávajúce členy zoznamu lst od člena at do konca
CBV	(PACK lst)	Výsledkom je atóm vzniknutý konkatenáciou atómov zoznamu lst
CBV	(UNPACK at)	Zoznam zo symbolov mena atómu at, opak PACK

4. Predikáty

Poznámka: Všetky funkcie v tejto tabuľke majú výsledok NIL, ak je nepravda, T ak je pravda (T = TRUE).

CBV	(ATOM se)	Výsledok je T, ak se = atóm
CBV	(EQ at1 at2)	Výsledok je T, ak atómy at1 a at2 sa rovnajú
CBV	(EQ dp1 dp2)	Výsledok je T, ak dp1 a dp2 zdieľajú rovnaké miesto v pamäti
CBV	(# se)	Výsledok je T,ak se je číslo
CBV	(> n1 n2)	Výsledok je T, ak číslo n1 je väčšie než n2

5. Aritmetické funkcie

CBV	(SUB n1 n2)	Výsledok je rozdiel n1-n2
CBV	(+ n1 n2)	Výsledok je súčet n1+n2
CBV	(* n1 n2)	Výsledok je súčin n1*n2
CBV	(/ n1 n2)	Výsledok je podiel n1/n2
CBV	(INT n)	Výsledok je najbližšie celé číslo
CBV	(EXP n)	Výsledok je umocnenie e na n
CBV	(LOG n)	Výsledok je $\log_e n$

6. Textový vstup/výstup

Poznámka: Všetky funkcie v tejto tabuľke predpokladajú štandardný vstup z klávesnice a štandardný výstup na obrazovku.

CBV	(PRINT se)	Výstup se-výrazu a RETURN, výsledkom je se.
CBV	(PRINT1 se)	Ako PRINT, ale bez RETURN

CBV	(PRIN2 se)	Funkcia nebola odskúšaná
CBN	(PAGE)	Nová strana, výsledok je NIL
CBN	(TERPRI)	Vytlačenie riadku a RETURN, výsledok je NIL
CBN	(READ)	Načítanie s-výrazu
CBN	(READA)	Načítanie atómu
CBN	(READC)	Načítanie symbolu (funkcia nie je odskúšaná)
CBV	(TAB n)	Nastavenie kurzora na n pozíciu bežného riadku

7. Práca s perifériami a súbormi

Vysvetlivky: iocb - číslo súboru

dev -zariadenie (K: P: C: S: E: R: D:fname D2:fname)

type - typ súboru (4 - READ, 8 - WRITE, 12 - R W,

9 - APPEND, 6 - DIRECTORY)

CBV	(OPEN iocb type dev)	V jazyku BASIC: OPEN iocb,type,0, dev. Výsledok je 1, ak bola operácia úspešná
CBV	(CLOSE iocb)	V jazyku BASIC: CLOSE iocb
CBV	(IN #iocb)	Vstup ide zo súboru iocb (musí byť otvorený)
CBV	(PR #iocb)	Výstup ide do súboru iocb (musí byť otvorený)
CBV	(LOAD dev)	Natiahnutie pracovného poľa zo zariadenia dev

CBV	(SAVE lst dev)	Uchovanie objektu s menom lst na periférne zariadenie dev
CBV	(DIR dev)	Výpis obsahu disku dev
CBV	(POINT iocb sctr byte)	V jazyku BASIC: POINT iocb,sctr, byte
CBV	(NOTE iocb sctr byte)	V jazyku BASIC: NOTE iocb,sctr, byte
CBV	(XIO cmd,iocb,m,n,dev)	V jazyku BASIC: XIO cmd,iocb,m,n, dev

8.Grafika

CBN	(GR n)	V jazyku BASIC: GRAPHICS n
CBV	(SETCOL reg hue lumin)	V jazyku BASIC: SETCOLOR reg,hue, lumin
CBV	(COL reg)	V jazyku BASIC: COLOR reg
CBV	(COL ascii)	V jazyku BASIC: COLOR ascii
CBV	(PLOT x y)	V jazyku BASIC: POSITION x,y
CBV	(DRAW x y)	V jazyku BASIC: DRAWTO x,y

9. Požívanie joysticku

CBV	(STICK n)	V jazyku BASIC: STICK(n), kde n= 0 alebo 1
CBV	(STRIG n)	V jazyku BASIC: STRIG(n), kde n= 0 alebo 1

10. Zvukové efekty

CBV	(SOUND n1 n2 n3 n4)	V jazyku BASIC: SOUND n1,n2,n3,n4

11. Špeciálne systémové funkcie

CBN	(MEM)	Výsledkom je veľkosť voľnej pamäti RAM
CBN	(OBLIST)	Výsledkom je zoznam všetkých atómov v pracovnej oblasti
CBN	(NEW)	Nastavenie počiatočného stavu systému. Všetky existujúce objekty sa strácajú
CBN	(RESET)	Ako NEW, ale objekty zostávajú
CBV	(POKE adr val)	V jazyku BASIC: POKE val,adr
CBV	(PEEK adr)	V jazyku BASIC: PEEK(adr)
CBV	(@ lst)	Funkcia nie je odskúšaná.

O b s a h

Autorsky a redakčne spracovali v AK n.p. Doprastav PR, Bratislava

Ing. Ladislav Gál

Ing. Eva Blechová

Bratislava Marec 1988

lgi (2026)