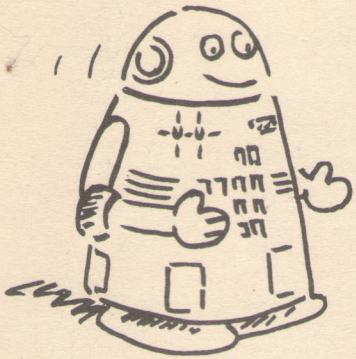

LOGO - učebnice pro počítač PMD85-2 a Consul

Obsah

1. Úvod

MicroLOGO je implementací jazyka LOGO vyvinutého v 70. letech na MIT (Massachustetts Institute of Technology) pro účely výuky. Jazyk nutí studenty programovat strukturovaně a umožňuje programovat metodou "shora dolů". Tím se dosahuje lepších výsledků při výuce, než je tomu při použití populárního jazyka BASIC. Jazyk LOGO navíc poskytuje možnost použití "želví grafiky", což činí výuku příjemnější a zajímavější. Ve výuce je vhodné jazyk LOGO zařadit jako mezistupeň mezi jazyky KAREL a PASCAL. Programy v jazyce MicroLOGO se nepřekládají do strojového kódu, jako je tomu v jazyce PASCAL, ale provádějí se přímo interpretem jazyka. Vaše verze programu MicroLOGO je vhodná pro mikropočítače PMD-85/2 a C 2717 (Zbrojováček).

2. Spuštění programu

Interpret jazyka MicroLOGO nahrajte do mikropočítače následovně:

1) Zapněte mikropočítač.

2) Máte-li v mikropočítači modul BASIC (u CM 2717 vždy), stiskněte [↑] a zároveň [RCL].

3) Na dialogovém řádku je zobrazeno

```
+++ os ready +++
```

4) Napište *MGLD 00* a stiskněte tlačítko [EOL].

5) Přetočte kazetu na začátek strany 1 a spusťte magnetofon.

6) Po úvodním zapísknutí by se na dialogovém řádku mělo objevit

```
00/? LOGO.
```

Je-li tomu tak, MicroLOGO se nahrává, v opačném případě opakujte postup od bodu 2.

7) Asi po 3 minutách se na obrazovce objeví nápis

```
MicroLOGO v1.3 (c) 1989 PEZIK.
```

Ten signalizuje, že MicroLOGO je připraveno k práci a že můžete zastavit magnetofon. Objeví-li se na obrazovce nápis

```
+++ File error +++,
```

zkuste opakovat celý postup s tím, že kazetu již nepřetáčíte. Na pásce je totiž ještě jedna záložní kopie. Nepovede-li se nahrání ani poté, máte pravěpodobně špatně nastavený magnetofon.

Na druhé straně kazety (strana 2) se nacházejí demostrační programy nazvané "Demo1" a "Demo2". Chcete-li je spustit, řiďte se následujícím postupem:

1) Nahrajte do mikropočítače interpret jazyka MicroLOGO (použijte předchozí postup).

2) Napište na klávesnici: *load* " a stiskněte EOL.

3) Přetočte kazetu na začátek demonstračního programu a spusťte magnetofon.

4) Na obrazovce by se měl objevit nápis

```
Loading Demo1
```

nebo

```
Loading Demo2.
```

Není li tomu tak, stiskněte tlačítko STOP a opakujte postup od bodu 2.

5) Po chvíli se na obrazovce objeví

```
?,
```

zastavte magnetofon a napište *Demo1* nebo *Demo2* (podle toho který z programů jste nahráli) a stiskněte tlačítko EOL. Tím spustíte demonstrační program. Vyskytla-li se v průběhu nahrávání chyba, je hlášena nápisem

```
Chyba v souboru.
```

V takovém případě postup opakujte od bodu 2.

3. Seznámení s jazykem LOGO

Interpret jazyka MicroLOGO může pracovat ve dvou základních režimech. Prvním z nich je režim přímého provádění příkazů a druhým je režim programování. Po nahrání se interpret nachází v režimu přímého provádění příkazů. Tento režim interpretu poznáte na první pohled podle toho, že na začátku řádku je zobrazen znak

```
?.
```

Napíšete-li za ? nějaký příkaz, kterému interpret jazyka MicroLOGO rozumí a stisknete-li tlačítko EOL, příkaz se ihned provede. Například:

```
?print 3+4
7
?print "ahoj
ahoj
?
```

Chcete-li použít nějaký příkaz, který MicroLOGO nezná, musíte jej nejprve naprogramovat. K tomu slouží příkaz *To*. Například pokud chcete naprogramovat příkaz *pozdrav*, který napíše *ahoj*, postupujte takto:

```
?To pozdrav
>print "ahoj
>end
?
```

Znak >, který se objeví po stisknutí prvního EOL značí, že se interpret dostal do režimu programování a že napsané příkazy nebude ihned provádět. Programovaný příkaz končí napsáním slovem *end*, po jehož napsání přejde interpret opět do přímého režimu. Kromě toho lze programování ukončit stisknutím tlačítka END. Naprogramovaný příkaz lze ihned použít:

```
?pozdrav
ahoj
?
```

Na jeden řádek je možno napsat i několik příkazů za sebou. Potom se budou provádět postupně v tom pořadí, v jakém jsou na řádku napsány:

```
?print "prvni print "druhy print "treti
prvni
druhy
treti
?
```

Místo některých příkazů můžeme psát i jejich zkratky, které se budou dále uvádět v závorkách. Například místo *print* lze použít *pr*. Místo názvů příkazů je možno psát jejich počáteční znaky ukončené tečkou, tedy místo *print* lze napsat i *pri.*. Při použití tohoto zápisu se najde první příkaz, který zkratce odpovídá, je tedy třeba při použití dávat pozor. Příklad použití zkratek:

```
?print 10 pr 20 pri. 30
10
20
30
?
```

Naprogramované příkazy je možno ihned použít i pro programování dalších příkazů:

```
?To pozdrav2
pozdrav pozdrav
?end
?pozdrav2
ahoj
ahoj
?
```

3. 1. Řádkový editor

Místo, kam se napíše z klávesnice následující znak, je označeno blikajícím obdélníkem, který se nazývá "kursor". Znaky napsané z klávesnice se před znak, na kterém je kursor, buď vkládají, nebo uvedený znak přepisují. Pokud při psaní uděláte chybu, můžete ji před stiskem tlačítka EOL ještě opravit. K tomu slouží následující editační tlačítka (je li označení ↑ X stiskněte nejprve ↑ (shift) a aniž je pustíte, stiskněte i X):

← Smaže znak před kursorem.

← Posune kursor vlevo.

→ Posune kursor vpravo.

⇤ Posune kursor na začátek řádku.

⇥ Posune kursor na konec řádku.

DEL Smaže znak, na kterém je kursor.

INS Přepíná režim vkládání/přepisování.

CLR Smaže celý řádek.

C-D Smaže řádek od kursoru do konce.

RCL Vyvolá předchozí řádek (odeslaný EOL).

↑ CLR Přepínání velkých a malých písmen (shift lock) (U C 2717 Zbrojováček i tlačítko vlevo od ?).

WRK Prefix znaků české abecedy. (Stisknete-li WRK a písmeno, zobrazí se písmeno s háčkem nebo čárkou, odpovídající znak v kódu KOI-8čs, viz příloha B.)

↑ DEL Napíše znak "~".

↑ ← Napíše znak "|".

Jste-li už s obsahem řádku spokojeni, stiskněte EOL. Tím řádek potvrdíte a interpret jej zpracuje (nezáleží přitom na poloze kursoru na řádku).

Některá z často používaných slov jsou pro zrychlení zápisu předdefinována na funkčních klávesách F1 .. F12 (stisknutí funkční klávesy má stejný účinek jako vypsání slova, které je na něm nadefinováno):

přímá klávesa	význam	Shift + klávesa	význam
F1	load "	F1	save "
F2	To	F2	saveall "
F3	ed [	F3	edall
F4	make	F4	ern
F5	po [	F5	type
F6	print	F6	show
F7	stop	F7	output
F8	repeat	F8	while
F9	clearscreen	F9	clean
F10	left	F10	penup
F11	right	F11	pendown
F12	forward	F12	back

Pokud je potřeba napsat příkaz, který je delší než řádek na obrazovce, je možno pokračovat v psaní příkazu i na dalším řádku (aniž stiskneme EOL). Přitom se na konci každého zaplněného řádku zobrazí znak "!", který signalizuje, že příkaz pokračuje na dalším řádku. Editační tlačítka zůstávají beze změny funkce.

3. 2. Želví grafika

Součástí interpretu jazyka MicroLOGO je i "želví" grafika. Želví proto, že pro znázornění kreslení je použita želva, která má na sobě pero. Želvě můžete prostřednictvím interpretu MicroLOGO dávat příkazy, kterými ji přinutíte k pohybu. Například příkazem

```
?forward 50
?
```

donutíte želvu, aby odešla o 50 kroků vpřed. Želva po sobě nechá stopu o délce 50 kroků. Pro pohyb želvy je vyhrazeno "grafické okno" v horní části obrazovky. Okno se vytvoří automaticky použitím libovolného grafického příkazu (například použitím uvedeného příkazu *forward*),

zároveň se zobrazí i želva, pokud již není zobrazena. Grafické okno můžete smazat pomocí příkazu *clearscreen* (místo příkazu *clearscreen* lze použít zkratku *cs*, zkratky budou dále uváděny v závorce již bez komentáře). Při zobrazení grafického okna zbývá v dolní části obrazovky 6 řádků pro psaní příkazů. Grafické okno se ruší příkazem *textscreen* (*ts*).

Základními příkazy pro ovládání želvy jsou:

forward (*fd*) - posuv želvy vpřed o zadaný počet kroků;
left (*lt*) - otočení želvy o zadaný úhel vlevo;
right (*rt*) - otočení želvy o zadaný úhel vpravo.

Vhodnou kombinací příkazů je možno nakreslit libovolný obrazec. Například kombinací

```
fd 40 lt 90 fd 40 lt 90 fd 40 lt 90 fd 40 lt 90
```

nakreslíte na obrazovce čtverec o straně 40 kroků. Další příkazy pro ovládání želvy a grafické příkazy jsou uvedeny v příloze A.

3. 3. Obrazovkový editor

Pokud je třeba opravit nějaký program (příkaz), lze použít obrazovkového editoru. Nejprve editor spustíme příkazem *edit* (*ed*) a zadáním programu, který chceme opravovat. Například máme-li nadefinován příkaz *pozdrav2* z kapitoly 2, můžeme napsat

```
?ed "pozdrav2
```

Tím se vyvolá obrazovkový editor a zobrazí se obsah příkazu *pozdrav2*. Příkaz lze v editoru opravovat obdobně jako na řádku (editační tlačítka pro editaci řádku pracují) s tím, že jsou k dispozici další editační tlačítka:

Posuv kursoru o řádek nahoru.

Posuv kursoru o řádek dolů.

Posuv kursoru na konec textu.

Posuv kursoru na začátek textu.

Funkce [EOL] se v obrazovkovém editoru liší od řádkového editoru. V obrazovkovém editoru totiž neslouží [EOL] pro potvrzení řádku, ale pro oddělení řádků. Chápe se tedy v obrazovkovém editoru jako znak. Z toho plyne, že chcete-li v obrazovkovém editoru přejít na další řádek, nesmíte použít [EOL]. Tím byste totiž řádek rozdělili na dva (použijte [↑] [→]). Pokud potřebujete spojit dva řádky v jeden, stačí nastavit kursor na konec prvního z nich (použijte [→]) a tlačítkem [DEL] zrušit konec prvního z řádků.

Pokud jste s opravami hotovi, stiskněte tlačítko [END]. To znamená, že jste s opravami spokojeni, opravené příkazy se uloží a interpret přejde opět do přímého režimu. Pokud s opravami nejste spokojeni, stiskněte [STOP] [STOP] a příkazy se neuloží (zůstanou tak, jak byly před vyvoláním editoru).

Příkaz edit (ed) lze použít i tak, že jako jeho parametr zadáte ne jeden příkaz, ale seznam příkazů. Například

```
?edit [pozdrav pozdrav2]
```

Pomocí editoru je možno vytvářet i dosud neexistující příkazy. K tomu postačí jejich definici připsat k právě editovaným. Editor je možno vyvolat i jako prázdný pro definici nových příkazů

```
?ed [].
```

Pokud editor vyvoláte příkazem *edit* nebo *ed*, provede se před editací ještě příkaz *textscreen*, což má za následek smazání grafického okna. Editor lze ovšem vyvolat ještě příkazem *edi*. Při použití tohoto příkazu zůstane grafické okno zachováno (pokud ovšem existuje).

3. 4. Hlášení chyb

Pokud interpret MicroLOGO narazí při práci na chybu, hlásí ji zapísknutím a zobrazením hlášení o druhu chyby, chybné hodnoty a v případě, že chyba nastala v programu i označením místa, kde chyba nastala. Pokud chyba nastane na příkazovém řádku, zobrazí se jeho obsah znovu a kursor se nastaví do místa, kde chyba nastala. Příklad:

```
?pozdrav aaa
ahoj
```

```
Neznámy název 'aaa'
?pozdrav aaa
```

A kursor zůstane na začátku slova aaa. Pokud nastala chyba v programu, zobrazí se kursor na začátku příkazového řádku, kterým byl vyvolán program s chybou a obrazovkový editor lze vyvolat pouhým stiskem tlačítka END s tím, že kursor je opět na místě, kde nastala chyba (editor se vyvolá jako příkazem edi).

Stiskem tlačítka STOP lze zobrazit (například po chybě) názvy a obsahy všech proměnných a parametrů použitých v programu.

4. Popis jazyka

Program v jazyce MicroLOGO je tvořen množinou procedur a funkcí. Přitom nezáleží na pořadí jejich definice. Procedury i funkce je možno používat ihned po jejich definování. Z hlediska uživatele lze proceduru v jazyce LOGO chápat i jako příkaz, který lze vyvolat názvem z příkazového řádku. Procedura se od funkce liší jen tím, že neposkytuje výstupní hodnotu, zatímco funkce ano. Procedury i funkce se skládají z posloupnosti příkazů ukončené slovem "end". Procedury i funkce mohou mít libovolný počet parametrů, z nichž každý musí mít svůj název. Seznam parametrů spolu s názvem procedury nebo funkce tvoří záhlaví. To se obvykle pro zvýšení přehlednosti programu píše na jeden řádek. Seznam parametrů může být uveden v závorkách a jednotlivé parametry mohou být odděleny čárkami (rozšíření). Před názvy parametrů (pokud nejsou uvedeny v závorkách) musí být znak ":". (Uvnitř procedury se musí ve standardním jazyce LOGO vyskytovat před parametrem opět znak ": ", v MicroLOGu nemusí - rozšíření.) Kdekoliv v proceduře nebo funkci může být uveden komentář uzavřený do složených závorek. Příklady definice procedur:

```
To ctverec {procedura bez parametru}
fd 40 lt 90 fd 40 lt 90 fd 40 lt 90 fd 40 lt 90
end

To ctvpar :X {parametr standardní LOGO}
fd :X lt 90 fd :X lt 90 fd :X lt 90 fd :X lt 90
end

To ctvpar2(X) {parametr MicroLOGO}
fd X lt 90 fd X lt 90 fd X lt 90 fd X lt 90
end
```

Pro funkce platí to, co bylo řečeno o procedurách. Rozdílem je to, že se v těle funkce musí vyskytnout příkaz *output* (*op*) s parametrem, který je výsledkem funkce. Příklad:

```
To Mocnina(X,Y) {Výsledkem je X^Y}
output X^Y
end
```

```
To plus1(X) {Vrací X zvětšené o 1}
output X+1
end
```

V jazyce MicroLOGO lze volat procedury a funkce s parametry uzavřenými v závorkách (rozšíření). Příklad:

```
To NaDruhou(X) {Vrací druhou mocninu X}
output(Mocnina(X,2))
end
```

4. 1. Cyklus typu repeat

Jazyk MicroLOGO poskytuje několik jazykových konstrukcí pro strukturované programování. Vzhledem k tomu, že v jazyce LOGO neexistuje příkaz typu Goto, jsou tyto konstrukce jediným prostředkem pro vytváření cyklů a větvení programů.

Cyklus typu repeat se používá tehdy, je-li potřeba opakovat nějakou činnost programu víckrát a je předem známo, kolikrát ji bude potřeba opakovat. Syntaxe cyklu je následující:

```
repeat n [posloupnost příkazů]
```

Číslo n udává, kolikrát se bude opakovat posloupnost příkazů, která je uzavřena v hranatých závorkách. Na místě čísla n může být libovolný číselný výraz. Vyhodnocuje se celá část, n musí být menší než 65536 a je-li n nulové nebo záporné, posloupnost se neprovádí ani jednou. Typickým příkladem je kreslení čtverce:

```
To ct(X) {Čtverec s použitím repeat}
 repeat 4 {Čtverec má 4 strany}
 [
  forward X {Strana o délce X}
  left 90 {Úhly při vrcholech pravé}
 ]
end
```

Pomocí konstrukce repeat lze definovat i N-úhelník:

```
To N_uhel(N,X) {Strana X, N stran}
 repeat N
 [
  forward X {Strana o délce X}
  left 360/N {Želva se otáčí o vnější úhel}
 ]
end
```

4. 2. Cyklus typu while

Cyklus typu while se používá tehdy, pokud je nutno nějaký úsek programu opakovat víckrát, není přesně známo, kolikrát se takový úsek bude opakovat, ale je známa podmínka, při které se má daný úsek ještě opakovat. Syntaxe:

```
while podmínka [posloupnost příkazů]
```

Dokud je splněna podmínka, posloupnost příkazů se opakuje. U tohoto typu cyklu je třeba dbát na to, aby se cyklus vůbec někdy dokončil. Příklad použití cyklu while:

```
while not keyp {Dokud není stisknuto tlačítko}
 [print "a ] {Tiskni znak "a"}

while X<3 {Dokud je X<3}
 [make "X X*Y] {Proveď X:=X*Y změň X na X*Y}

while 1=1 [left 90] {Pozor, nekonečný cyklus}
```

4. 3. Příkaz if

Příkaz if slouží k větvení programu na základě nějaké podmínky. Syntaxe je následující:

```
if podmínka
 [příkazy nebo výraz]{[příkazy nebo výraz]}
```

Závorky {} značí, že příslušná část může chybět. První z posloupností příkazů nebo funkce se provede, pokud je podmínka splněna, druhá se provede, pokud podmínka splněna není. V jazyce MicroLOGO lze příkaz if používat i uvnitř výrazu. Příklad:

```
To absol(X) {Vrátí absolutní hodnotu X}
 if X>0 [output X] [output -X]
end

To Absol2(X) {Druhá varianta výraz v if}
output(if X>0 [X] [-X])
end

To Absol3(X) {Třetí varianta neúplný if}
 if X<0 [output -X]
 output X
end
```

V posledním z příkladů se využívá toho, že při provedení příkazu output se funkce opouští. Je-li tedy X záporné, provede se output s parametrem -X a na příkaz output X již interpret nenarazí.

4. 4. Zanořování příkazů

Programové konstrukce (cykly while, repeat a podmíněné větvení) jsou chápány jako jeden příkaz a je možno je neomezeně zanořovat (je možno je zapsat do seznamu příkazů některé z programových konstrukcí). Příklad:

```
To Sachovnice {Tisk šachovnice na obrazovku}
ts {Smaže obrazovku a přepne do text. módu}
repeat 8 {Šachovnice má 8 řádků}
 [
  repeat 4 {a 4 dvojice různých řádků}
  [
   print '@ @ @ @ ' {První typ řádku}
    print ' @ @ @ @' {Druhý typ řádku}
  ] {Konec repeat 4}
 ] {Konec repeat 8}
end
```

4. 5. Rekurze

Jednou z nejčastěji používaných jazykových konstrukcí v jazyce LOGO je rekurze. Rekurze spočívá ve volání programu sebou samým. Toto volání musí být ovšem podmíněné, jinak by se program volal až do naplnění zásobníku. Příklad:

```
To fact (I) {Výpočet faktoriálu I-I!}
 if I<=1 [output 1] {faktoriál 0 a 1 je 1}
 output I*fact(I-1) { I! = I*(I-1)! }
end
```

Rekurze ovšem můžeme použít například i pro kreslení:

```
To spirala(X) {Nakreslí spirálu}
 if X<10 [stop] {Je-li X již příliš malé, konec}
 forward X {Želva popoleze o X kroků}
 left 10 {Otočí se o 10 stupňů doleva}
 spirala X-1 {Zkrátí krok a pokračuje}
end
```

Zanořování při rekurzi je omezeno pouze rozsahem volné paměti (při prázdné paměti se u procedury s jedním parametrem pohybuje maximální úroveň zanoření kolem 500).

5. Proměnné a typ hodnoty

Při definici procedury nebo funkce se často setkáváme s nutností uložit nějaký mezivýsledek pro pozdější potřebu. K tomu se používají proměnné. Proměnnou lze chápat jako pojmenovanou hodnotu. Hodnotu proměnné lze přečíst nebo změnit. K tomu ovšem musíme znát její název. Kromě toho může být potřeba vytvořit novou proměnnou nebo zrušit starou proměnnou, která již není potřeba. (Proměnné v jazyce LOGO se nedeklarují. Vytvoří se při běhu programu.)

K vytvoření proměnné se používá příkaz "make". Příkaz má dva parametry. Prvním parametrem je název proměnné, druhým parametrem je její hodnota. Příklad:

```
make "A 5 {Vytvoří proměnnou A s hodnotou 5}.
```

Příkaz make se také používá při změně hodnoty proměnné. Pokud zadáme jmého proměnné, která již existuje, nevytvoří se nová proměnná, ale změní se obsah původní. Například:

```
make "B 4 {Vytvoří proměnnou B}
make "A 3 {Jen změní obsah již existující
proměnné}
```

Pokud chcete proměnnou použít (přečíst její hodnotu), napište pouze její název na místo, kde by jinak bylo například číslo. (Rozšíření proti standardnímu LOGu, kde se musí název uvodit znakem ":". To je ovšem v jazyce MicroLOGO možné také.) Příklad:

```
make "C A {Vytvoří proměnnou C s hodnotou 3}
make "D B {Vytvoří proměnnou D s hodnotou 4}
make "X :A {Vytvoří proměnnou X s hodnotou 3}
```

Proměnnou je možno zrušit příkazem *ern*. Příklad:

```
ern "A {Zruší proměnnou A}
```

Příkaz *erns* slouží ke zrušení všech proměnných.

5. 1. Typ číslo

V jazyce MicroLOGO jsou definovány 4 datové typy. Jsou to číslo, slovo, seznam a logický datový typ. Číslo můžeme zapisovat v desetinném nebo exponenciálním tvaru. Příklad:

```
123       {Celé číslo}
12.45     {Číslo v desetinném tvaru}
0.12      {Desetinný tvar, nula je povinná!}
1e5       {Exponenciální tvar = 100000}
12.4E5    {Exponenciální tvar = 1240000}
```

Rozsah čísel je asi ±1E-37 až 1E37 (10^{-37} až 10^{37}), přesnost zobrazení je asi 6 platných desetinných míst bez ohledu na to, zda jde o celá nebo desetinná čísla (pro zobrazení je použita 24 bitová mantissa a 8 bitový exponent).

Nad čísly jsou definovány základní aritmetické operace, operace porovnání velikosti, logaritmické, exponenciální, goniometrické a cyklometrické funkce (viz přílohu A). Příklad:

```
3+4       {Číselný výraz, hodnota 7}
2+cos(0)  {Číselný výraz, hodnota 3}
```

5. 2. Logický typ

Logický datový typ může nabývat jen dvou hodnot: TRUE (pravda) nebo FALSE (nepravda). Hodnota logického typu se uvádí například na místě podmínky v konstrukcích if a while. Můžeme ji získat jako výsledek operací srovnání hodnot nebo použitím předdefinovaných funkcí *True* a *False*. Příklad:

```
3>2       {Logický výraz, výsledek TRUE}
True      {Standardní funkce True, výsledek TRUE}
False     {Standardní funkce False, výsl. FALSE}
2<=1      {Logický výraz, výsledek FALSE}
```

Nad logickým typem jsou definovány operace logického součtu (or), součinu (and) a negace (not). Příklad:

```
3>2 and False {Logický výraz, hodnota FALSE}
2<=1 or 3>2   {Logický výraz, hodnota TRUE}
3>2 and not 2<=1  {Logický výraz, hodnota TRUE}
```

5. 3. Typ slovo

Typ slovo (textový typ) se používá pro zpracování textů. Hodnotu typu slovo je možno zapsat dvěma způsoby:

1) Hodnota se uvodí znakem " a končí mezerou, koncem řádku nebo koncem definice.

2) Hodnota se uvodí znakem ' a končí opět znakem ' nebo koncem řádku, případně koncem definice. (Rozšíření.)

Příklad:

```
"Ahoj          {Slovo Ahoj = 'Ahoj'}
"Dobry den      {Slovo Dobry = 'Dobry', den přebývá}
'Jak se mate' {Text 'Jak se mate', nic nepřebývá}
```

Nad typem slovo jsou definovány operace spojení word, čtení prvního resp. posledního znaku slova first a last, čtení textu bez prvního resp. posledního znaku butfisrt (bf), butlast (bl) a další (viz přílohu A). Příklad:

```
word "ab "cd  {Slovní výraz, hodnota 'abcd'}
first "ahoj   {Slovní výraz, hodnota 'a'}
last "ahoj    {Slovní výraz, hodnota 'j'}
butfirst "ahoj {Slovní výraz, hodnota 'hoj'}
butlast "ahoj {Slovní výraz, hodnota 'aho'}
```

5. 4. Typ seznam

Hodnota typu seznam je uspořádanou posloupností hodnot libovolného typu (i typu seznam, definice je rekurzivní). Hodnotu typu seznam můžeme dostat zřetězením hodnot příkazem sentence (se) nebo ji můžeme zapsat přímo do hranatých závorek. V hranatých závorkách se ovšem mohou vyskytovat pouze hodnoty typu číslo nebo slovo (u konstant typu slovo, uzavřených do kulatých závorek, se nemusí psát znak "). Příklad:

```
[1 2 3] {Seznam 3 čísel} [ab cd ef] {Seznam 3
slov}
[ab 3 cd] {Seznam 2 slov a 1 čísla}
[[a b]c] {Seznam 1 seznamu (s prvky a b) a
slova}
sentence "a "b {Seznam 2 slov}
```

Nad typem seznam jsou definovány obdobné operace jako nad typem slovo. Jsou to spojení sentence (se), čtení prvního, resp. posledního prvku seznamu first, resp. last, čtení seznamu bez prvního, resp. bez posledního prvku butfirst (bf), resp. butlast (bl). Dále je definována operace vytvoření seznamu ze dvou hodnot libovolného typu list (rozdíl mezi list a sentence!). Příklad:

```
first [1 2 3] {Výraz se seznamam, hodnota 1}
last [a b c] {Výraz se seznamam, hodnota 'c'
butfirst [a b c]{Výraz se seznamam, hodnota [b
c]}
butlast [1 2 3] {Výraz se seznamam, hodnota [1
2]}
list [1] [2] {Výraz se seznamy, hodnota
[[1][2]]}
sentence [1] [2]{Výraz se seznamy, hodnota [1
2]}
```

5. 5. Typ proměnných a parametrů

Do proměnné v jazyce LOGO se může dosadit hodnota libovolného typu. Typ hodnoty v proměnné se může během výpočtu měnit. Například posloupnost příkazů

```
make "A 10 make "A [1 2 3] make "A False
```

je z hlediska jazyka LOGO zcela korektní. Stejná pravidla platí i pro parametry. Jako parametr funkce nebo procedury se může předat hodnota libovolného typu. Typ parametru se může uvnitř funkce nebo procedury zkontrolovat, je-li to zapotřebí. Pro tuto kontrolu jsou k dispozici standardní funkce (např listp, viz přílohu A). Můžeme tedy napsat:

```
?To projdi(X) {Hodnota X projde funkcí beze
>změn}
>output X
>end
?print projdi 3
3
?print projdi projdi "abc
abc
?print projdi [1 2 3]
1 2 3
?
```

5. 6. Lokální proměnné

Je-li potřeba uvnitř nějaké procedury nebo funkce uchovat mezivýsledek, je možno použít příkaz "make" (viz úvodní odstavec kapitoly 5). Použijeme-li ovšem tento příkaz a procedura skončí, zůstává vytvořená proměnná v paměti. Navíc, pokud procedura na nižší úrovni (tedy ta, která uvažovanou proceduru nebo funkci vyvolala) měla již proměnnou se stejným názvem vytvořenou, její obsah by se přepsal. Proto existuje v jazyce MicroLOGO možnost vytvoření lokální (místní jen pro proceduru, v níž je vytvořena) proměnné. Lokální proměnnou lze vytvořit příkazem "local". Takto vytvořená proměnná se zruší v okamžiku návratu z procedury nebo z funkce, v níž byla vytvořena. Proměnná vytvořená příkazem local má stejné vlastnosti jako parametr. Příklad:

```
To Na2(X) {2. mocnina přirozeného čísla}
local "S {Vytvoření lokálních proměnných}
 local "P
make "P X {Dosazení počáteční hodnoty}
 make "S 0
while P>0 {Vlastní výpočet}
 [
  make "S S+X
  make "P P-1
 ]
 output S
end
```

6. Další vlastnosti jazyka

V této kapitole jsou uvedeny některé další podstatné vlastnosti jazyka MicroLOGO, které se vymykaly obsahu předchozích kapitol.

6. 1. Definice binárních operátorů

V jazyce MicroLOGO je možno definovat operátory binárních operací, které lze potom použít v infixové notaci (jsou to vlastně funkce se dvěma parametry, jejichž název se uvádí mezi parametry). Definované operátory musí mít dva parametry s povinnými názvy _1 a _2. (Záhlaví musí mít tvar název(_1,_2).) Operátory se od funkcí formálně odlišují tím, že za názvem operátorů se při definici uvádí značka operátoru "~ " nebo "-" a číslice udávající prioritu operátoru. Čím vyšší číslice, tím vyšší priorita. Příklad:

```
?To &~6(_1,_2) {Operátor spojení slov}
> output word _1 _2 {Priorita 6}
>end

?print "abc & "def {vytiskni spojení slov}
abcdef
?
```

Pokud se jako znak operátoru uvede ~, znamená to, že za názvem operátoru nemusí následovat oddělovač. Například operátor + je definován jako + ~4, lze tedy psát 3+4. Naproti tomu operátor mod je definován jako mod-6, nelze tedy psát 3 mod4, ale je třeba napsat 3 mod 4. Obdobný význam jako "~" pro operátry má pro procedury znak "|".

Definované binární operátory lze volat i jako funkce dvou parametrů (v prefixové notaci). Můžeme tedy psát:

```
+ 2 3      {Číselný výraz s hodnotou 5}
+(2,3)     {Číselný výraz s hodnotou 5}
& "ab "cd {Slovní výraz s hodnotou 'abcd'}
```

6. 2. Tvoření jmen

Jména v jazyce MicroLOGO (jména procedur, funkcí a proměnných) nesmí obsahovat znaky [ ] ( ) : . Jména nesmějí začínat znaky " ' : a nesmí začínat číslicí, jinak jsou vyhodnocena jako text nebo číslo (takto vytvořené proměnné či definice by bylo možno používat pouze s pomocí funkcí "thing" a "run"). Jako oddělovače lze použít všech znaků kromě písmen, znaků $ _ a číslic, pokud ovšem tyto znaky nejsou součástí jména.

Velká a malá písmena se při volání nerozlišují, ale interpret jazyka MicroLOGO si je pamatuje a při výpisu používá takových písmen, jaká byla použita v definici.

6. 3. Vícenásobné volání

V jazyce LOGO lze používat vícenásobného volání procedur a funkcí. Vícenásobné volání se označuje uzavřením procedury nebo funkce i s parametry do kulatých závorek (parametrů musí být více než vyžaduje procedura nebo funkce). Pokud se v závorce vyskytuje procedura, volá se postupně s parametry uvedenými v závorce až do jejich vyčerpání. Pokud se v závorce vyskytne funkce s více parametry, z nichž první má název _1 (jedná se tedy o všechny operátory), postupuje se takto: První volání funkce se provede s prvními parametry v závorce. Pro další volání se jako první parametr s názvem _1 dosadí výsledek funkce a další parametry se přečtou z dalších pozic v závorce. To se provádí až do vyčerpání parametrů. Výsledkem závorky (pokud se volá funkce) je výsledek posledního volání. Příklad:

```
(forward 10 20 30) {Posune o 10, 20 a 30 kroků}
print (+ 1 2 3 4 5) {vytiskne (((1+2)+3)+4)+5 =
15}
```

6. 4. Příkazy print, show a type

Pro tisk hodnot v jazyce LOGO se používají příkazy print, show a type. Příkaz print má jeden parametr, který se vytiskne. Pokud je tímto parametrem seznam, netiskne se první levá a poslední pravá hranatá závorka značící seznam. Příkaz show je shodný až na to, že se hranaté závorky u seznamu tisknou. Příkazy print a show po vytisknutí hodnoty

vyvolají přechod na nový řádek s jedinou výjimkou. Jsou li totiž vícenásobně volány (viz předchozí odstavec), provede se přechod na nový řádek až po vyčerpání parametrů. Příkaz type řádkování neprovádí. Parametrem všech uvedených příkazů může být hodnota libovolného typu. Příklad:

```
?print [1 2] print [3 4]
1 2 3 4
?show [1 2] show [3 4]
[1 2]
[3 4]
?type [1 2] type [3 4]
[1 2][3 4]
?(print [1 2] [3 4])
1 2 3 4
?(show [1 2] [3 4])
[1 2] [3 4]
?
```

6. 5. Funkce readlist a readquote

Funkce readlist (rl) a readquote (rq) slouží ke čtení údajů z klávesnice. Příkaz readlist čte seznam (jakoby uzavřený v hranatých závorkách), zatímco příkaz readquote čte výraz, který po přečtení z klávesnice vyhodnotí. Příklad:

```
?make "A readlist show A
:1 + 2 * 3
[1 + 2 * 3]
?

make "A readquote show A
:1 + 2 * 3
7
?
```

Pokud je kursor v okamžiku volání funkcí na první pozici na řádku, vytisknou funkce readlist a readquote uvozovací znak :, pokud se kursor nachází na jiné pozici, tento znak se netiskne. Příklad:

```
?print 'Zadej A' make "A readlist (print 'A=' A)
Zadej A
:5
A= 5
?

?type 'Zadej B > ' make "B readlist (print 'B='
B)
Zadej B > 23
B= 23
?
```

6. 6. Příkazy save a load

Příkazy save a load slouží k záznamu programů na vnější paměť a čtení programů z této paměti. V případě PMD-82/2 i C 2717 se jedná o magnetofon. Příkaz save má dva parametry. Prvním parametrem je procedura nebo funkce, případně jejich seznam, který se má nahrát do vnější paměti. Druhým parametrem je název, kterým bude nahrávka označena (název souboru na mgf. pásce). Příklad:

```
save [prvni druha treti] "vsechny
```

Před vykonáním příkazu save musí být již magnetofon spuštěn. Postup ukládání je tedy následující:

1) Napište příkaz save s příslušnými parametry, ale bez stisknutí EOL.

2) Spusťte magnetofon v režimu nahrávání.

3) Stiskněte EOL a vyčkejte, až se na dalším řádku objeví otazník.

4) Zastavte magnetofon, ukládání programu je ukončeno.

6. 7. Trasování

K ladění programů poskytuje interpret jazyka MicroLOGO dva základní prostředky. Je to přehledné hlášení chyb a možnost trasování programů. Hlášení chyb bylo popsáno v odstavci 3.4.

Trasováním programu se rozumí průběžné informování uživatele o tom, která část programu se právě zpracovává, případně i s jakými hodnotami parametrů. Interpret může takovou informaci vydávat, běží-li v režimu trasování. Do režimu trasování se interpret přepne buď po provedení příkazu "trace", nebo po stisku  při běhu programu. Režim trasování se ruší provedením příkazu "notrace" nebo opětovným stisknutím  . V režimu trasování se vypisuje informace o každém vstupu do procedury nebo funkce i s hodnotami parametrů a o každém výstupu s hodnotami parametrů a lokálních proměnných a u funkcí i s návratovou hodnotou. Příklad:

```
->Pro1 X=1 Y=2 {Vstup do procedury 1}
->Fce2 A=5 {Procedura volá funkci}
->Fce2 A=5 L=1 10 {Funkce si vytvořila lokální
proměnnou L a vrací 10}
->Pro1 X=10 Y=11 {Návrat z procedury 1,
parametry se změnily}
```

7. Standardní funkce a procedury

V této příloze jsou popsány všechny předdefinované procedury a funkce jazyka MicroLOGO. U každé procedury a funkce jsou uvedeny deklarované parametry a popis funkce. U procedur a funkcí, pro které existuje zkratka, je tato uvedena pod názvem. Funkce jsou odlišeny od procedur znakem = uvozujícím popis.

`abs(_)` =Vrací absolutní hodnotu parametru (čísla).

`acs(_)` =Vrací arcuskosinus parametru (ve stupních). Rozsah úhlů je 0,180.

`agtr-3(_1,_2)` =Vrací pravdivou logickou hodnotu, je-li první parametr v absolutní hodnotě větší než druhý.

`all` =Vrací seznam všech uživatelem definovaných funkcí a procedur.

`and-2(_1,_2)` =Vrací logický součin parametrů (logických).

`ascii(_)` =Vrací kód prvního znaku textového řetězce. Je-li řetězec prázdný, vrací 0.

`asn(_)` =Vrací arcussinus parametru (ve stupních). Rozsah úhlů je -90,90.

`atn(_)` =Vrací arcustangens parametru (ve stupních). Rozsah úhlů je -90,90.

`atn2(_1,_2)` =Vrací arcustangens podílu parametrů (ve stupních) s respektováním znamének čitatele i jmenovatele. Rozsah úhlů je -180,180).

`back(_)`
`bk` –Posouvá želvu o parametrem zadaný počet kroků zpět.

`beep` –Pískne.

`break` –Ukončí provádění programu.

`butfirst(_)`
`bf` =Vrací buď textový řetězec bez prvního znaku nebo seznam bez prvního prvku.

`butlast(_)`
`bl` =Vrací buď textový řetězec bez posledního znaku nebo seznam bez posledního prvku.

`char(_)` =Vrací znak odpovídající kódu v rozsahu 32-255.

`clean` –Smaže obrazovku.

`clearscreen` `cs`	–Smaže obrazovku, nastaví želvu do základní polohy.
`cload(_)`	–Nahrání programu z magnetofonu. Parametry jako load. Je implementováno jen v síťové verzi BASICu na C 2717.
`cos(_)`	=Vrací kosinus parametru (ve stupních).
`count(_)`	=Vrací buď počet znaků textového řetězce nebo počet prvků seznamu.
`csave(_1,_2)`	–Nahrání programu na magnetofon. Parametry jako load. Je implementováno jen v síťové verzi BASICu na C 2717.
`definedp(_)`	=Vrací true, je-li definována funkce nebo procedura se zadaným názvem, jinak vrací false.
`div-5(_1,_2)`	=Vrací celočíselný podíl parametrů (čísel).
`dot(_)`	–Zobrazí bod na souřadnici zadané parametrem (ve stejném tvaru jako v setpos).
`edit(_)` `ed`	–Edituje funkci, proceduru nebo jejich seznam v textovém režimu.
`edall`	–Edituje všechny funkce a procedury v textovém režimu.
`edi(_)`	–Edituje funkci, proceduru nebo jejich seznam.
`emptyp(_)`	=Vrací pravdivou logickou hodnotu, je-li textový řetězec nebo seznam prázdný.
`end`	–Označuje konec uživatelem definované funkce nebo procedury.
`equalp(_1,_2)`	=Vrací true, jsou-li si parametry rovny.
`erall`	–Smaže procedury, funkce a proměnné.
`erase(_)` `er`	–Maže funkci nebo proceduru, případně jejich seznam.
`ern(_)`	–Maže zadanou proměnnou.
`erns`	–Maže všechny proměnné.
`erps`	–Smaže všechny procedury a funkce.

`exec`	– Provede následující blok. (Musí být v podobě textu, ne jako proměnná nebo výraz.)
`exp(_)`	= Vrací hodnotu e ^ _ (exponenciální funkce).
`false`	= Vrací hodnotu false.
`fence`	– Nastavuje mód "fence" (želva nesmí překročit okraj obrazovky).
`first(_)`	= Vrací buď první znak řetězce nebo první prvek seznamu.
`forward(_)` `fd`	– Posouvá želvu o parametrem zadaný počet kroků vpřed.
`fput(_1,_2)`	= Vrací seznam vytvořený tak, že první parametr vloží na začátek druhého parametru (seznamu).
`frac(_)`	= Vrací desetinnou část parametru. (Rozdíl číselného parametru a nejbližšího nižšího celého čísla.)
`free`	= Vrací počet volných byte paměti.
`heading`	= Vrací úhel nastavení želvy (pravotočivé ve stupních). Nastavení svisle nahoru odpovídá 0.
`hideturtle` `ht`	– Potlačí zobrazení želvy.
`home`	– Nastaví želvu do základní polohy.
`if(_)`	– V závislosti na logickém parametru provede nebo neprovede následující blok (bloky) programu.
`int(_)`	= Vrací nejbližší nižší celé číslo k parametru.
`item(_1,_2)`	= Vrací n-tý znak řetězce nebo n-tý prvek seznamu. První parametr udává číslo n, druhý parametr je řetězec nebo seznam.
`keyp`	= Je-li stisknuta klávesa, vrací pravdivou logickou hodnotu.
`last(_)`	= Vrací poslední znak řetězce nebo poslední prvek seznamu.

`list(_1,_2)`	=Vrací seznam, jehož prvky jsou postupně první a druhý parametr.
`listp(_)`	=Vrací pravdivou logickou hodnotu, má- li proměnná, jejíž název je parametrem, hodnotu typu seznam. Jinak vrací nepravdivou hodnotu.
`load(_)`	–Čte z vnější paměti zadaný soubor. Vnější pamětí je buď magnetofon nebo disk (disk v diskových verzích na C 2717).
`local(_)`	–Vytvoří lokální proměnnou s názvem udaným prvním parametrem (při vytvoření nemá hodnotu).
`left(_)` `lt`	–Otočí želvu o udaný úhel doleva (proti směru hodinových ručiček).
`ln(_)`	=Vrací přirozený logaritmus parametru.
`log(_)`	=Vrací logaritmus parametru při základu deset.
`log2(_)`	=Vrací logaritmus parametru při základu dvě.
`logicp(_)`	=Vrací pravdivou logickou hodnotu, je-li hodnota logického typu. Jinak vrací nepravdivou hodnotu.
`lput(_1,_2)`	=Vrací seznam vzniklý vložením prvního parametru na konec druhého parametru (seznamu).
`make(_1,_2)`	–Změní hodnotu proměnné s názvem udaným prvním parametrem na hodnotu udanou druhým parametrem. Není-li proměnná dosud definována, vytvoří novou globální proměnnou.
`memberp(_1,_2)`	=Vrací true, je-li prvek _1 obsažen v seznamu _2, jinak vrací false.
`mod-5(_1,_2)`	=Vrací zbytek po dělení parametrů (čísel).
`namep(_)`	=Vrací pravdivou logickou hodnotu, je-li hodnota názvem proměnné. Jinak vrací nepravdivou hodnotu.

`not(_)`	=Vrací logickou hodnotu opačnou k hodnotě parametru.
`notrace`	–Končí mód trasování programu.
`num(_)`	=Vrací parametr (textový) převedený na číslo.
`numberp(_)`	=Vrací pravdivou logickou hodnotu, je-li hodnota číslem. Jinak vrací nepravdivou hodnotu.
`or-1(_1,_2)`	=Vrací logický součet parametrů (logických).
`output(_)` `op`	–Ukončuje provádění funkce a nastavuje její výstupní hodnotu na hodnotu _.
`pendown` `pd`	–Nastavuje grafický mód *pendown*, želva za sebou nechává světlou stopu.
`penerase` `pe`	–Nastavuje grafický mód *penerase*, želva za sebou nechává tmavou stopu.
`penup` `pu`	–Nastavuje grafický mód *pen up* želva za sebou nenechává žádnou stopu.
`po(_)`	–Vypíše funkci, proceduru, nebo jejich seznam.
`pops`	–Vypíše všechny funkce a procedury.
`position`	=Vrací momentální pozici želvy na obrazovce jako pos seznam [X Y].
`pots`	–Tiskne názvy všech definovaných procedur a funkcí.
`preverse` `px`	–Nastavuje grafický mód *preverse*, želva za sebou nechává negovanou stopu.
`primitivep(_)`	=Vrací true, je-li parametrem standardní funkce nebo procedura.
`primitives`	=Vrací seznam názvů všech standardních funkcí a procedur.
`print(_)` `pr`	–Zobrazuje hodnotu parametru. Za hodnotou zobrazí navíc vždy jednu mezeru a konec řádku. Je-li procedura print uzavřena s několika parametry do kulatých závorek, končí se řádek až po vyčerpání všech

parametrů. U seznamů nezobrazuje na první úrovni zanoření "[" a "]".

`printon` – Zahajuje kopírování výstupu na tiskárnu.

`printoff` – Ukončuje kopírování výstupu na tiskárnu.

`random(_)` = Vrací celé náhodné číslo od 0 do _ .

`readchar` `rc` = Vrací znak napsaný na klávesnici. Je-li stisknuta řídicí klávesa, je hlášena chyba.

`readcode` = Vrací kód stisknuté klávesy. Není-li stisknuta žádná klávesa, čeká na její stisk.

`readlist` `rl` = Vrací řádek přečtený z klávesnice jako seznam. Pokud byl kursor v okamžiku zahájení editace na 1. pozici, tiskne uvozovací znak ':'.

`readquote` `rq` = Vrací hodnotu výrazu zadaného z klávesnice. Způsob zadávání výrazu je obdobný jako na příkazovém řádku. Pokud by byl kursor v okamžiku zahájení editace na 1. pozici řádku, tiskne se uvozovací znak ':' .

`repeat(_)` – Parametr (číselný) udává, kolikrát se bude následující programový blok opakovat.

`rerandom` – Nastaví generátor náhodných čísel na počáteční hodnotu.

`right(_)` `rt` – Otočí želvu doprava (ve směru hodinových ručiček) o úhel udaný parametrem.

`rnd` = Vrací náhodné číslo v rozsahu ,1).

`round(_)` = Vrací zaokrouhlený parametr _ .

`run(_)` – Spustí funkci nebo proceduru, jejíž název je parametrem. Parametry funkce se čtou z textu za 'run'.

`save(_1,_2)` – Uloží na vnější paměť funkci, proceduru nebo jejich seznam (druhý parametr) do souboru (jehož jméno je prvním parametrem). Vnější pamětí je buď magnetofon nebo disk (disk v diskových verzích na C 2717).

`saveall(_)`	–Uloží na mgf. pásku všechny procedury a funkce.
`scrunch`	=Vrátí nastavené měřítko jako seznam [X Y]. Základní nastavení je [1 1].
`sentence(_1,_2)` `se`	=Vrací parametry spojené do seznamu. (*se [1][1 2]=[1 2 3], se [1] 2 =[1 2] ...*)
`setcursor(_)` `setcur`	–Nastaví kursor do polohy zadané parametrem ve tvaru [X Y]. Poloha je ve znacích, levý horní roh má souřadnice [0 0].
`setheading(_)` `seth`	–Nastaví želvu do směru udaného parametrem (směr 0 je svisle nahoru).
`setpos(_)`	–Přesune želvu do pozice udané parametrem (seznam souřadnic [X Y]). Souřadnice [0 0] je ve středu obrazovky. Souřadnice X roste doprava, Y nahoru. Jeden bod obrazovky odpovídá nárůstu souřadnice o 1 při měřítku [1 1].
`setscrunch(_)` `setscr`	–Nastaví měřítko. Parametrem je seznam měřítek souřadnic [X Y].
`setX(_)`	–Nastaví X-ovou souřadnici želvy na zadanou hodnotu.
`setY(_)`	–Nastaví Y-ovou souřadnici želvy na zadanou hodnotu.
`show(_)`	–Zobrazí hodnotu parametru. Za hodnotu přidá jednu mezeru. U textových řetězců zobrazuje před a za hodnotou znak ' .
`shownp`	=Vrátí pravdivou logickou hodnotu, je-li želva zobrazena, jinak vrátí nepravdivou logickou hodnotu.
`showturtle` `st`	–Zobrazí želvu.
`sin(_)`	=Vrací sinus parametru (ve stupních).
`sqrt(_)`	=Vrací druhou odmocninu parametru (číselného).
`stop`	–Ukončí provádění procedury.
`str(_)`	=Vrací parametr (číselný) převedený na text.

`system`	–Ukončuje práci systému MicroLOGO a vrací řízení monitoru.
`tan(_)`	=Vrací hodnotu tangens parametru (ve stupních).
`ten(_)`	=Vrací hodnotu 10 ^ _ (mocnina deseti).
`textscreen` `ts`	–Přepne obrazovku do textového režimu a smaže ji.
`thing(_)`	=Vrací hodnotu proměnné, jejíž název je parametrem.
`To(_)`	–Zahajuje definici funkce nebo procedury, jejíž název je parametrem.
`towards(_)`	=Vrátí úhel, na který se musí želva nastavit (příkazem seth), aby mířila do zadaného bodu.
`trace`	–Zahajuje mód trasování programu. V tomto módu se při vstupu a výstupu do a z funkce nebo procedury vypisuje její název a seznam lokálních proměnných i s hodnotami.
`true`	=Vrací hodnotu true.
`two(_)`	=Vrací hodnotu 2 ^ _ (mocnina 2).
`type(_)`	–Zobrazí hodnotu parametru. U seznamů nezobrazuje na první úrovni zanoření znaky [a].
`window`	–Nastavuje mód "window" (překročí- li želva okraj obrazovky, nezobrazuje se).
`while(_)`	–Dokud je splněna logická podmínka (parametr), provádí se následující blok programu.
`word(_1,_2)`	=Vrací textový řetězec vzniklý spojením obou parametrů (textových řetězců).
`wordp(_)`	=Vrací pravdivou logickou hodnotu, má- li proměnná, jejíž název je parametrem, typ textový řetězec (slovo). Jinak vrací nepravdivou hodnotu.

`wrap`	–Nastavuje mód "wrap" (překročí- li želva okraj obrazovky, začne se zobrazovat na opačném okraji obrazovky "modulo").
`Xcor`	=Vrátí X-ovou souřadnici želvy.
`Xscr`	=Vrátí X-ové měřítko.
`Ycor`	=Vrátí Y-ovou souřadnici želvy.
`Yscr`	=Vrátí Y-ové měřítko.
`+~4(_1,_2)`	=Vrací součet parametrů (číselných).
`-~4(_1,_2)`	=Vrací rozdíl parametrů (první druhý).
`*~5(_1,_2)`	=Vrací součin parametrů (číselných).
`/~5(_1,_2)`	=Vrací podíl parametrů (první / druhý).
`~6(_1,_2) **`	=Vrací obecnou mocninu (první ^ druhý).
`>~3(_1,_2)`	=Vrací pravdivou logickou hodnotu, je-li první parametr větší než druhý (číselné).
`<3(_1,_2)`	=Vrací pravdivou logickou hodnotu, je- li první parametr menší než druhý (číselné).
`>=~3(_1,_2)`	=Vrací pravdivou logickou hodnotu, je- li první parametr větší nebo roven druhému (číselné).
`<=3(_1,_2)`	=Vrací pravdivou logickou hodnotu, je- li první parametr menší nebo roven druhému (číselné).
`=~3(_1,_2)`	=Vrací pravdivou logickou hodnotu, je- li typ i hodnota prvního parametru rovna druhému.
`<>~3(_1,_2)` `#`	=Vrací pravdivou logickou hodnotu, není-li první parametr roven druhému.
`.primitives`	–Tiskne názvy všech standardních funkcí a procedur.

7. 1. Tabulka kódů KOI-8čs

	20	30	40	50	60	70	C0	D0	E0	F0
0	SP	0	@	P	‘	p		ô		Ô
1	!	1	A	Q	a	q	á	ä	Á	Ä
2	"	2	B	R	b	r		ř		Ř
3	#	3	C	S	c	s	č	š	Č	Š
4	$	4	D	T	d	t	ď	ť	Ď	Ť
5	%	5	E	U	e	u	ě	ú	Ě	Ú
6	&	6	F	V	f	v	í		Ŕ	
7	’	7	G	W	g	w	ch	é	CH	É
8	(	8	H	X	h	x	ü		Ü	
9	)	9	I	Y	i	y	í	ý	Í	Ý
A	*	:	J	Z	j	z	ů	ž	Ů	Ž
B	+	;	K	[	k	{	Ĺ			
C	,	>	L	\	l	|	Ľ			
D	-	=	M	]	m	}	ö		Ö	
E	.	>	N	~	n	~		Ň		
F	/	?	O	_	o		ó		Ó	

8. Literatura

[1] Abelson H., diSessa A. A.: Turtle Geometry, 1984, MIT Press.

[2] Timar J.: LOGO pro ZX-Spectrum, 1986, OPS Pardubice.

Ing. Pavel Zemčík

LOGO - učebnice

Jazyková úprava: Eva Rocheltová
Obálka:Vladimír Nohejl
Vydalo ©ZENITCENTRUM, Hostímská 703, 266 01 Beroun, jako příručku k programu LOGO. Číslo povolení 523-403/16-88. Samostatně neprodejné.

VYDÁVÁ
ZENITCENTRUM
POBOČKA BEROUN