# KURS JAZYKA SINCLAIR LOGO
## ver. 1.6



"Umět tak současně přemýšlet jako dítě a myslit jako počítač!"
Seymour Papert, MIT, 1964

## I. LOGO - hříčka nebo vážná alternativa?

Programovací jazyk LOGO je tak jednoduchý, že s ním může pracovat i malé dítě. Přesto LOGO není dětským jazykem. Původně vznikl na Massachusetts Institute of Technology v rámci pokusu o zapojení dětí do práce s velkým počítačem. Před několika lety byl interpreter LOGO vytvořen pro Commodore 64 v USA a dnes, kdy počítače vstoupily do domácností, je LOGO přístupné každému. Varianta Sinclair LOGO se maličko liší od jiných variant LOGO, ale to není na závadu.

Jazyk LOGO má tyto zvláštnosti:

- grafika želvy
- široká sada příkazů
- možnost zpracování Seznamů
- velmi dobré vedení uživatele
- možnost rekurze
- příkazy strukturovaného programování, procedury a logické vlastnosti
- relativně vysoká rychlost zpracování
- lehce zvládnutelný jazyk
- univerzální použitelnost v programování

U verze Sinclair LOGO byla odstraněna nevýhoda jiných verzí, které neumějí vydávat hudební tóny. Zůstala však nevýhoda extrémních nároků na objem paměti.

### Snadno řiditelná želva

Pro většinu uživatelů bude nejpřitažlivější tzv. želví grafika, tj. kreslení na obrazovku řízením želvy, která za sebou zanechává stopu. Pro kreslení je tu široká sada příkazů.

Prohlédněte si obraz KVADRAT, přiložený k tomuto textu. Byl nakreslen procedurou KVADRAT bez souřadnicového systému a bez komplikovaných matematických vztahů. Nahrajte Sinclair LOGO a zadejte: (na velká písmena přepněte CAPS LOCK)

```
TO KVADRAT :D
FENCE
FORWARD :D RIGHT 90 FORWARD :D RIGHT 90
FORWARD :D RIGHT 90 FORWARD :D FORWARD :D
KVADRAT :D+1
END
```

Po vložení odstartujte zadáním KVADRAT 2.

Všimněte si, že řádky programu nemají čísla, příkazy se provádějí jak jdou za sebou. LOGO má též Editor, který umožňuje tvořit a opravovat procedury bez očíslovaných řádků.

Popišme si nyní listing procedury KVADRAT:

Začíná se vždy jménem procedury, před kterým musí být TO. (Nezapomeňte dělat mezery mezi slovy a příkazy!)

Následuje příkaz FENCE, jenž nastaví překladač tak, aby hned, jakmile se želva dostane na hranici papíru, vydal chybové hlášení a zastavil výkon procedury.

Želva se může vždy pohybovat buď kupředu (FORWARD možno zkrátit na FD) nebo dozadu (BACK možno zkrátit na BK) o počet jednotek, který je dán číslem za příkazem. Přitom za sebou zanechává stopu (PENDOWN nebo PD) nebo stopu nezanechává (PENUP nebo PU znamená PERO ZDVIHNOUT). Též je možno želvu na jednom místě natáčet doprava (RIGHT nebo RT) o zadaný počet úhlových stupňů od dosavadní pozice nebo doleva (LEFT nebo LT). Želva se dá řídit i souřadnicemi, ale náš program se bez nich obešel.

Máme však zavedenou proměnnou :D (proměnná se označuje dvojtečkou), která je délkou posunutí pro příkaz FORWARD. Takže FORWARD :D znamená posunutí želvy o D kroků kupředu ve směru natočení její hlavy a záleží na momentální hodnotě D. Příkazem RIGHT 90 se otočí želva o 90 stupňů doprava, tedy o pravý úhel.

Další příkazy nutno vysvětlit podrobněji: Příkaz END označuje konec procedury a je nutné jej vždy na konci definování procedury napsat. Jak si ale vysvětlit předposlední řádek? Každá procedura musí mít své jméno, ta naše se jmenuje KVADRAT. Po ukončení definování příkazem END je procedura pod tímto jménem uložena do paměti a podobně jako v jazyce PASCAL je možno ji tímto jménem kdykoli vyvolat k činnosti. KVADRAT :D+1 tedy znamená, že se začne vykonávat procedura KVADRAT znovu od začátku, ale s hodnotou proměnné D zvětšenou o jednu. Vidíme, že v našem programu procedura KVADRAT vyvolává sama sebe a to je princip všech zázraků - skvělá vlastnost moderních programovacích jazyků, zvaná "rekurze".

LOGO obsahuje i další příkazy pro strukturované programování: IF, THEN, ELSE, IFTRUE, IFFALSE, REPEAT (JESTLIŽE, PAK, JINAK, KDYŽ PRAVDA, KDYŽ NEPRAVDA, OPAKUJ). Další možnosti jsou určeny pro ovládání jednotky disketové paměti (DOS). Grafika dovoluje užívat dříve vytvořené procedury nebo obrazy, nahrané na pásku. Překladač ale vyžaduje, aby různé procedury měly různá jména, tedy KVADRAT smí být použito jenom jednou.

## Srozumitelná chybová hlášení

LOGO je vybaven komfortním vedením uživatele. Programátoru se nestane, že by byl jako v BASICu odbyt lapidárním hlášením "Synatx error". Chybová hlášení v LOGO obsahují:

Hlášení řádek procedura úroveň.

Pod terminem "úroveň" se rozumí toto: mód přímých příkazů je označen číslem 0, vyskytne-li se chyba uvnitř procedury, bude hlášení 1. Je-li chyba odvozena od jiné, má hodnotu 2, atd.

## Slova a Seznamy

LOGO poskytuje dvě možnosti zpracování textů: Slova a Seznamy. Obě formy spolu těsně souvisí. Data mohou být převáděna z jedné formy na druhou, čísla mohou být brána jako Slova (vlastně řetězce) a mohou se se Slovy společně vyskytovat.

Slovo je v jazyku LOGO definováno jako řetězec. K jeho označení je nutno před ně napsat uváděcí značku (uvozovky). Tím je Slovo rozpoznáno od jména procedury nebo od příkazu. Chceme-li vytisknout Slovo "ahoj" na obrazovku, zadáme:

PRINT "ahoj

Konec Slova se neoznačuje uvozovkami, ale mezerou nebo ukončením řádku.

Se Slovy lze manipulovat vhodnými procedurami, např. WORD připojí jedno Slovo k druhému:

PRINT WORD "ČARY "MARY dá výsledek ČARYMARY.

Zatímco Slovo označené v řadě uváděcí značkou (uvozovkami) je samostatným slovem, společná řada Slov dává tzv. Seznam. Ten se dá opět různě zpracovávat. Seznam se označuje tak, že se jeho slova uzavřou do hranatých závorek:

PRINT "[SINCLAIR ZX SPECTRUM] dá SINCLAIR ZX SPECTRUM

Zpracováním Seznamu dostaneme opět operační řádek. Např. příkaz SENTENCE spojí více Seznamů v jeden. Další operace jsou:

FIRST vydá první slovo Seznamu LAST vydá poslední Slovo
BUTFIRST dává zbytek Seznamu bez prvního slova
BUTLAST dává zbytek Seznamu bez posledního slova.

Tyto a další operace mohou být kombinovány. Např. si můžeme nechat vytisknout

LAST BUTLAST [POČÍTAČ TELEVIZOR TISKÁRNA]

a dostaneme Slovo TELEVIZOR, protože je to poslední slovo zbytku Seznamu bez posledního slova. Konečně můžeme také složit Seznam ze samých Seznamů - tedy místo Slov bude Seznam obsahovat jiné seznamy jako své elementy:

[POČÍTAČ TELEVIZOR TISKÁRNA] [KNIHA PERO PAPÍR]

Zacházením se seznamy jako s velmi flexibilními datovými strukturami přicházíte zde na principy moderního jazyka LISP, který se používá v bádáních o umělé inteligenci.

## LOGO podporuje logické myšlení

LOGO je velmi snadno použitelný jazyk, disponující velmi dobrou sadou příkazů. Obzvláště jeho mnohostrannost je pozoruhodná. Nehledě k tomu, že želví grafika poskytuje mnoho zábavy, podporuje LOGO rozvoj logického a strukturovaného myšlení. Je to jazyk nejen pro děti, ale i pro dospělé, kteří se poprvé setkávají s počítačem a s jeho pedagogickou hodnotou. Také pro ty, kteří si rádi hrají, poskytuje mnoho příležitostí. S pomocí LOGO se vbrzku stanete programátory.

## Hrajeme si s obrazem na obrazovce

Již jsme dlouho mluvili vážně, odpočiňme si při hraní. Naprogramujeme si velmi krátkou, ale podivuhodnou proceduru se jménem TRAPEZ. Ta pracuje s třemi proměnnými, jejichž dosazováním dle vlastní vůle získáte přerozmanité grafické vzory které můžete přes sebe skládat a neustále měnit. Zahrajte si na návrháře koberců - prohlédněte si připojený obrázek TRAPEZ, není to vlastně pěkný koberec? Program TRAPEZ vyvolává stále sám sebe a můžete ho kdykoli zastavit pomocí BREAK klávesy, zadat nové hodnoty proměnných a zase spustit:

```
TO TRAPEZ :X :Y :D
PENUP SETX :X SETY :Y
PENDOWN FD :D RT 90
TRAPEZ :X-1 :Y :D+1
END
```

Startujte např. TRAPEZ 127 0 87, potom TRAPEZ 0 0 -87 ,pak TRAPEZ -50 0 50. Pak, co Vás napadne. Povede se Vám koberec? Obrazy na obrazovce budou daleko hezčí než náš obrázek vyvedený tiskárnou!

## II. Želva se učí běhat a kreslit

Jako starý skvělý Leonardo maloval svoji Monu Lisu, můžete i Vy namalovat svůj obraz. Musíte přitom provádět mnoho jednotlivých tahů štětcem a mezi jednotlivými tahy štětec z plátna zvednout a přenést na jiné místo.

Chcete-li nakreslit např. dům s oknem a dveřmi, musíte si uvědomit, že to nelze nakreslit jedním tahem. Bude třeba kreslit zvlášť DUM, DVERE a OKNO. Každý útvar je pak nutno začínat kreslit ve správné pozici, na kterou se musí s želvou nejprve dojít, aniž by želva přitom kreslila. Pero (Pen) musí být přitom zvednuto (up), což provedeme příkazem PENUP (možno zkrátit na pouhé PU). Vymažte obrazovku pomocí CS a začneme.

Sekvencí příkazů CS PU FD 50 se želva posune o 50 pozic nahoru, aniž by kreslila (FD je zkrácené FORWARD). Když nyní zadáte opět CS, obraz se smaže a tím se pero opět spustí dolů ke kreslení. Jenže takto přijdeme o obraz. Proto se ke spuštění pera používá příkaz PENDOWN (zkráceně PD). Ten dovolí spustit pero, aniž by se obraz smazal.

Teď už můžeme napsat proceduru DUM:

```
TO DUM                          jméno procedury
PD                              pero dolů
RT 90 FD 40 LT 90 FD 40         spodek domu
LT 90 FD 40 LT 90 FD 40         (tj. obdélník)
RT 180 FD 40                    přechod ke střeše
RT 30 FD 40 RT 120 FD 40        střecha
PU                              pero zvednout
END                             konec procedury
```

### Kde je pero?

Všimněte si, že hned na začátku programu je PD a program končí s PU. To je pochopitelné. Má-li být nakreslen dům, musí být pero spuštěno - PD - a po nakreslení domu musí být zvednuto, neboť je nutno želvu převést na jinou počáteční pozici pro kreslení např. dveří. To nám dovolí PU. My jistě chceme uvidět dům s dveřmi i oknem. Musíme tedy sestavit procedury na jejich nakreslení.

Na kresbu dveří stačí tři čáry. Chceme např. dveře vysoké 15 jednotek a široké 10 jednotek:

```
TO DVERE
PD FD 15 RT 90 FD 10 RT 90 FD 15 PU
END
```

## Dveře musí vést do domu

Nyní již Můžeme nakreslit DUM i DVERE, ovšem odděleně od sebe. Zadejte CS a otočte želvu o 90 stupňů doleva pomocí LT 90. Potom ji posuňte kupředu o 75 jednotek pomocí PU FD 75. Znovu ji otočíme nahoru pomocí RT 90, protože naše procedury začínají vždy s želvou orientovanou nahoru. Zadáme příkaz DUM.

Dům je nyní nakreslen. Natočte želvu o 60 stupňů doleva, pak ji posuňte o 30 a orientujte opět nahoru. Zadejte příkaz DVERE.

Dveře jsou nyní nakresleny, ale na jiném místě, než mají být. Musíme tedy před kresbou dveří převést želvu dovnitř domu. A to na správnou počáteční pozici pro kresbu dveří. Takže nyní želvu otočte a zkuste ji převést tam, kam patří - nezapomeňte PU. Pak zadejte znovu DVERE a ony se nakreslí tam, kde mají být.

Nyní potřebujeme nakreslit OKNO. Použijeme stejný způsob programování procedury (variant je ovšem více):

```
TO OKNO
PD
FD 16 RT 90 FD 16 RT 90              čtverec okna
FD 16 RT 90 FD 16 RT 90
FD 8 RT 90 FD 16 RT 90               vnitřní kříž
FD 8 RT 90 FD 8 RT 90 FD 16
PU
END
```

Umístěte nyní toto OKNO dovnitř domu. Výsledek by se měl shodovat s naším obrázkem, nazvaným VSECHNO.

## Dům s dveřmi a oknem

Pochopitelně, že chceme sestavit takový program, který nakreslí najednou celý dům, s dveřmi i oknem. Použijme tedy již sestavených procedur a sestavme proceduru VSECHNO:

```
TO VSECHNO
DUM DVERE OKNO
END
```

Natypujte příkaz VSECHNO a uvidíte podivný výsledek. Dveře a okno jsou umístěny zcela nesprávně, protože jsme zapomněli mezi jednotlivými procedurami převést želvu do správné počáteční pozice ke kreslení. Abychom počítač nezmátli, musíme chybnou proceduru VSECHNO vymazat. K tomu slouží příkaz

```
ERASE [VSECHNO]
```

což znamená vymazat proceduru VSECHNO.

Zkuste nyní zadat příkaz VSECHNO. LOGO Vám odpoví hlášení mI don't know how to VSECHNO, Nevím, co je to VSECHNO. Pochopitelně, procedura je přece vymazána. Můžeme ji tedy zadat znovu, tentokrát správně. Nejprve je třeba nakreslit dům, pak želvu převést na počáteční pozici pro kresbu dveří. Po skončení kresby domu je želva v jeho pravém horním rohu a je odkloněna o 30 stupňů k vodorovné rovině. Musíme tedy použít sekvenci RT 30 FD 40 RT 90 FD 30 RT 90, aby želva byla před kreslením orientována nahoru. Pak se mohou kreslit DVERE a po jejich nakreslení převedeme želvu do počáteční pozice pro kreslení okna sekvencí RT 180 FD 20 . A ovšem OKNO nakreslíme. Teď vidíte užitečnost umístění PD a PU uvnitř procedur. Program procedury VSECHNO lze tedy natypovat:

```
TO VSECHNO
DUM
RT 30 FD 40 RT 90 FD 30 RT 90
DVERE
RT 180 FD 20
OKNO
END
```

Příkazem VSECHNO nyní nakreslíte dům tak, jak je na našem obrázku. Chcete-li se procvičit, nakreslete na téže obrazovce více stejných domů, když po každém příkazu VSECHNO převedete ručně želvu do nové počáteční pozice a opět zadáte VSECHNO. Zkuste napsat proceduru OSADA, která nakreslí 3 domy vedle sebe. Potom proceduru SAMOTY, která nakreslí 3 domy na různých místech obrazovky hodně daleko od sebe.

## Želva má barevné tužky

Váš počítač generuje obraz barevný a byla by proto škoda kreslit jen černobíle. LOGO má příkazy na volbu barev v obraze.

Barvu papíru si zvolíte pomocí SETBG a barvu inkoustu pomocí SETPC, v obou případech nutno doplnit číslo barvy z barevné stupnice ZX Spectrum. Pomocí sekvence

```
CS SETBG 5 SETPC 2 VSECHNO
```

tedy nakreslíte náš dům červenou barvou na bledëmodrém papíře. Potom aniž byste něco kreslili, můžete změnit barvu papíru již hotového výkresu opět pomocí SETBG, např. SETBG 4 dá zelený papír. Stejně se dá změnit barva inkoustu.

## Pozice želvy je nejdůležitější

Změnu pozice želvy jsme museli doposud provádět poměrně složitě pomocí natáčení RT nebo LT a posunu kupředu FD či vzad BK. Existuje však jednodušší způsob nastavení želvy na potřebnou pozici, a to pomocí souřadnicového systému. Počátek souřadnic je v základní pozici želvy, kde se nachází po zadání CS nebo ji tam lze vrátit příkazem HOME (domů). Tato pozice má souřadnice [0 0] a vedou z ní pomyslné osy vodorovně a svisle, jak je to obvyklé v matematice. Vzhledem ke grafice ZX Spectrum má tedy vodorovná osa polohy želvy rozsah od -127 do +127 a svislá osa od -87 do +87. Jak nyní docílíme posun želvy z nulové pozice do pozice se souřadnicemi x= 100 a y= 50?

I pomocí nám známých příkazů to dokážeme, ale bude to zdlouhavé. Zadáme FD 50, čímž se želva dostane na y=50. Pak ji otočíme o 90 stupňů vpravo RT 90 a posuneme FD 100, čímž je na konečné pozici s x=100 a y=50. Jak ji nyní dostat do nové pozice s x= -60 a y= 10? Protože x máme rovno 100 a y rovno 50, musíme s želvou o 160 pozic doleva a o 40 pozic dolů. To vykonáme např. pomocí RT 180 FD 160 LT 90 FD 40. Chceme-li nyní dostat želvu zpět do pozice nulové, stačí k tomu FD 10 LT 90 FD 60 anebo již zmíněný příkaz HOME. Teď jsme prováděli tzv. relativní posuny.

## Přímý přesun

Ovšem popsaný způsob je zbytečně složitý. Přesun lze uskutečnit jedním příkazem SETPOS (nastav pozici), kterým se provádí absolutní určování pozice. Tedy SETPOS [100 50] za nás vyřeší první výše uvedený posun a SETPOS [-60 10] posun druhý.

Pokud dáte před SETPOS příkaz PD, spojí želva svoji starou a novou pozici přímou čarou.

## Trojí absolutní posouvání

Přesun lze uskutečnit ještě trojím způsobem. Příkazem SETX 50 posuneme želvu na x= 50, přičemž y zůstane zachováno. Podobně SETY a číslo posunou želvu na dané y a x zůstane zachováno. Třetí způsob je natočení želvy v dané pozici. SETHEADING zkráceně SETH 30 natočí želvu o 30 stupňů doprava vzhledem ke svislé ose. SETH 0 natočí želvu nahoru.

Pomocí těchto příkazů a procedury všechno lze nakreslit celé město, když udáme souřadnice počátečních bodů každého domu:

| Dům | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| x | -130 | -140 | -20 | 30 | 40 | -30 |
| y | -80 | 20 | -50 | -90 | 50 | 55 |

Odpovídající program bude:

```
TO MESTO
PU
SETPOS [-130 -80] PD VSECHNO PU
SETPOS [-140 20] PD VSECHNO PU
SETPOS [-20 -50] PD VSECHNO PU
    atd. až po
SETPOS [-30 55] PD VSECHNO PU
END
```

Když jste zvládli město, pokuste se sestavit program KOSTEL podle našeho obrázku. Rozdělte si úlohu na jednotlivé části kostela, pro které sestavte procedury a nakonec z nich sestavte celkovou proceduru KOSTEL.

## Přehled příkazů grafiky želvy

Projděte si pečlivě tento přehled a vyzkoušejte si příkazy, o kterých jsme se nezmiňovali.

| | |
|---|---|
| BK (BACK) číslo | chod zpět |
| BRIGHT | zjasnění |
| BG (BACKGROUND) | informace o barvě papíru (PRINT BG) |
| CLEAN | smaže obraz bez návratu želvy |
| CS (CLEARSCREEN) | nový start grafiky |
| FD (FORWARD) číslo | chod kupředu |
| FENCE | vydá chyb.hlášení,je-li želva na kraji |
| HT (HIDETURTLE) | želva neviditelná, kreslí však dále |
| HEADING | dá natočení želvy, např. PRINT HEADING |
| HOME | vrátí želvu do nulové pozice |
| LT (LEFT) číslo | otočí želvu doleva o daný počet stupňů |
| PC (PENCOLOUR) | dá číslo barvy inkoustu, např.PRINT PC |
| PD (PENDOWN) | spustí pero na papír |
| PU (PENUP) | zvedne pero z papíru |
| PE (PENERASE) | smaže stopu želvy |
| PX (PENREVERSE) | obnoví smazanou stopu |
| POS | dá pozici želvy, např. PRINT POS |
| RT (RIGHT) číslo | otočí želvu o daný počet stupňů |
| ST (SHOWTURTLE) | učiní zmizelou želvu viditelnou |
| SETH (SETHEADING) číslo | absolutní natočení želvy (0 = nahoru) |
| SETBG číslo | nastaví barvu papíru |
| SETX číslo | nastaví x-ovou souřadnici želvy |
| SETY číslo | nastaví y-ovou souřadnici želvy |
| SETBR (SETBORDER) číslo | nastaví barvu borderu |
| SHOWNP | závěr informace o stavu želvy BG+PC |
| SETPC číslo | nastaví barvu inkoustu |
| SETSCR (SETSCRUNCH) [x y] | zmenší nebo zvětší měřítko souřadnic (xkrát na délku, y krát na výšku) |
| SCRUNCH | s PRINT dá zkreslení měřítka |
| TOWARDS [x y] | natočí želvu k bodu o souřadn. x,y |
| WINDOW | přepne na obraz s želvou |
| WRAP | želva může vystoupit z obrazu |
| XCOR | např. s PRINT dá x souřadn. želvy |
| YCOR | dá y souřadnici želvy |
| TS (TEXTSCREEN) | přepne z obrazu želvy na textový obraz |
| DOT | umisťuje bod |

## III. Želva nám roste

Mezitím co jsme učili želvu kreslit, trochu povyrostla, takže ji můžeme naučit i obtížnější úkony, jako jsou manipulace s proměnnými. V předchozí kapitole jsme stavěli město tím, že jsme zadávali rozličné počáteční pozice pro kreslení a pak dávali příkaz VSECHNO, např.

PU SETPOS [30 50] VSECHNO nebo PU SETPOS [45 67] VSECHNO.

Zadávat vždy a vždy nové hodnoty počátečních pozic je zdlouhavé. Programátor, jenž chce být s programem dříve hotov, by proto chtěl zadat souřadnice jako proměnné, asi takto:

PU SETPOS [x y] VSECHNO

čímž by nahradil jedním programovým řádkem množství řádků dřívějších. Pak ale musí před výkonem programu určit počítači hodnoty proměnných x= 30 a y= 50.

### Tipy pro pohodlného programátora

Jak to ale uskutečnit?, ptá se náš pohodlný programátor. Víme už, že definice procedury musí začínat symbolem TO, takže naši novou proceduru nadefinujeme TO PDUM. Pak následuje sled příkazů jako v proceduře VSECHNO, s některými změnami. Nejprve PU pro zdvih pera. Pak by měl následovat SETPOS, jenže tento příkaz vyžaduje dvě proměnné, a to Sinclair LOGO neumí. Proto musíme SETPOS [x y] nahradit dvěma příkazy, kde každý pracuje jen s jednou proměnnou. To jde, SETPOS x y lze nahradit SETX x a SETY y. Ale LOGO musí nějak poznat, že jde o proměnnou x a o proměnnou y. K tomu slouží zaváděcí značka proměnné - dvojtečka těsně před jménem proměnné, např. :X nebo :Y

Náš pohodlný programátor tedy řádek nadefinuje takto:

PU SETX :X SETY :Y VSECHNO

V tomto programu bude LOGO posouvat želvu na takovou souřadnici x, jakou má momentálně proměnná :X hodnotu. Podobně i proměnná y.

### Hodnoty proměnných se musí určit předem

LOGO vyžaduje jména proměnných označovat dvojtečkou, jinak je jedno, jak proměnnou pojmenujete. Místo :X a :Y si můžete zvolit třeba :SOURADNICEX a :SOURADNICEY nebo i :STREJDA a :TETA protože jazyku LOGO je to jedno. Je zajisté lepší užívat jmen kratších.

Jak nyní sdělit počítači, jaké hodnoty proměnných jsou právě nyní platné? Základní možností zavedení konkrétní hodnoty pro danou proměnnou je její zadání při zadávání procedury, tedy pro náš příklad by to bylo PDUM 30 50. Ale LOGO musí nějak poznat, které proměnné která hodnota patří.

Pozná to snadno, jestliže při definování jména procedury uvedeme za jménem, s kterými proměnnými bude procedura pracovat, tedy pro náš příklad:

```
TO PDUM :X :Y
```

Pak ale musíme při zadávání výkonu procedury zadat hodnoty proměnných ve správném pořadí. Např. pro x=30 a y=50 je nutno zadat:

```
PDUM 30 50
```

V okamžiku stisku ENTER přiřadí LOGO první proměnné hodnotu 30 a druhé proměnné hodnotu 50. Můžeme již sestavit program PDUM:

```
TO PDUM :X :Y
PU SETX :X SETY :Y VSECHNO
END
```

## Zelva protestuje

Zkuste nyní zadat příkaz PDUM bez hodnot proměnných, tedy jen PDUM. Příkaz se nevykoná a LOGO protestuje chybovým hlášením Not enough inputs to PDUM, v české verzi LOGO je to hlášení Není dost vstupů pro PDUM. Totéž hlášení obdržíte, zapomenete-li např. volit číslo barvy u příkazu volby barvy inkoustu SETPC.

Zkuste nyní napsat program BDUM tak, aby bylo možno volit nejen kde bude dům stát, ale i jeho barvu. Např. aby byl příkazem BDUM 30 10 2 nakreslen červený dům na pozici [30 10].

## Opakování je nejen matkou moudrosti

Podívejte se na přiložený obraz kompozice. Není to nic jiného než samé obdélníky. Nevěříte tomu? Logo zde pracuje s jedním obdélníkem, kresleným stále v různých pozicích. Je nabíledni, že nakreslit takový obraz vyžaduje množství jednotlivých kreslicích příkazů. Jak to zvládnout pohodlněji? Naštěstí je LOGO vybaveno speciálním příkazem, jenž dovoluje opakovat daný příkaz nebo proceduru, kolikrát chcete.

Je to příkaz REPEAT (opakuj). Nejprve si ale nakreslíme

```
TO OBDELNIK
FD 10 RT 90 FD 60 RT 90 FD 10 RT 90 FD 60 RT 90
END
```

Nyní si položíme úkol nakreslit tento obdélník desetkrát. Ale kreslit ho stále na jednom místě nemá smysl. Proto musíme po každém nakreslení obdélníka želvu o něco posunout, aby nakreslila další obdélník jinde. Program tedy musí vykonat tyto úkony (desetkrát za sebou):

1) Nakreslit obdélník, 2) PENUP, pero nahoru, 3) např. FD 7, 4) otočit o 90 vpravo, 5) FD 7, 6) otočit o 90 vlevo, 7) PENDOWN

Odpovídající příkazový řádek je tedy

```
OBDELNIK PU FD 7 RT 90 FD 7 LT 90 PD
```

Trik nyní spočívá v tom, že celý postup musíme 10krát opakovat, což zajistíme příkazem REPEAT 10 a uzavřením toho, co se má opakovat, do hranatých závorek:

```
TO UHELNIK
REPEAT 10 [OBDELNIK PU FD 7 RT 90 FD 7 LT 90 PD]
END
```

Výsledek musí odpovídat našemu obrázku UHELNIK.

Vraťme se ale k obrazu KOMPOZICE. Je tak jednoduchý, jako je jednoduché posunout obdélník. Jenom se zde musí obdélník nejen posouvat, ale vždy také pootočit. Na jeden okruh okolo středu otáčení je 36 pootočení obdélníka o 10 stupňů. Po každém pootočení pak následuje posunutí. Procedura by mohla vypadat:

```
TO KOMPOZICE
PU SETPOS [-60 0] PD
REPEAT 36 [OBDELNIK FD 14 RT 10]
END
```

Po vyzkoušení vidíte, že výsledek neodpovídá ještě našemu obrazu. Kresba kolem dokola je tam zopakována třikrát, pokaždé s jiným středem otáčení a kratším posunutím (14 - 12 - 10). Před začátkem nového okruhu je provedeno nastavení do nové pozice a natočení želvy nahoru příkazem SETH 0. Dobré je také rozlišit každý okruh jinou barvou. Ten největší např. nakreslíme světle modrý, prostřední žlutý a vnitřní zelený, to vše pro efekt na černém papíře.

Teď už můžeme program napsat:

```
TO KOMPOZICE
SETBG 0 SETPC 5 PU SETPOS [-60 0] PD
REPEAT 36 [OBDELNIK FD 14 RT 10]
SETPC 6 PU SETPOS [-58 0] SETH 0 PD
REPEAT 36 [OBDELNIK FD 12 RT 10]
SETPC 4 PU SETPOS [-55 0] SETH 0 PD
REPEAT 36 [OBDELNIK FD 10 RT 10]
END
```

## Výpočty s LOGO

LOGO neslouží jen ke kreslení zajímavé grafiky. Můžete též zpracovávat matematické vzorce i funkce. Vyzkoušejte si to!

Na zadání 4 + 5 odpoví počítač RESULT : 9 . Na úkol 5 * 876 opět odpoví RESULT 4380. Stejně tak i na další podobná zadání.

Lze též početní výkony kombinovat:

```
56 * 87 - 87/6 + 65            RESULT: 4922.5
     Lze užívat závorky:
56 * (67 - 6)/45               RESULT: 75.9111
     Bez závorek pracuje LOGO s předností početních výkonů:
56 * 67 - 6/45                 RESULT: 3751.8667
```

LOGO nemá žádný příkaz pro umocňování, které musíte počítat násobením. Např. na výpočet páté mocniny libovolného čísla si sestavte program:

```
TO NAPATOU :C
PRINT :C * :C * :C * :C * :C
END
```

Příkaz PRINT zde není příkazem grafickým, ale příkazem tisku na obrazovku. Zadáte-li nyní NAPATOU 4.55, dostanete pátou mocninu čísla 4.55, tedy RESULT: 1950.1005.

Ale tato procedura počítá výhradně pátou mocninu a žádnou jinou. Bylo by praktičtější definovat funkci, která by akceptovala dvě čísla - základ mocniny c a exponent e. Umocňování je vlastně opakované násobení, šlo by tedy použít

REPEAT, např.:

```
REPEAT 3 [PRINT 3*3]
```

Jak to uskutečnit? Odpověď hledejme opět v užití proměnných a jejich přidělených hodnot. Zadáním TO MOCNINA :C bude definována proměnná :C, která bude mít konstantní hodnotu tak dlouho, dokud jí nezadáme hodnotu novou. Příkaz

```
REPEAT 3 [PRINT :C* :C]
```

však nedá proměnné žádnou hodnotu neboť není příkazem přiřazovacím.

Musíme tedy zkusit proceduru TO ...., :PROMěNNá ještě jednou, ale připustíme přiřazení hodnoty k proměnné až UVNITř programu. LOGO má na to speciální přiřazovací příkaz MAKE (udělej). Jeho efekt spočívá v tom, že hodnota jedné proměnné je "udělána" hodnotou druhou. Jméno nové proměnné musí přitom začínat zaváděcí značkou " (uvozovky).

Takže

MAKE "C 5  přiřadí proměnné :C hodnotu 5.

MAKE "C :C* :C  přiřadí proměnné :C hodnotu odvozenou od její předešlé hodnoty a udanou příkazem na konci, tedy zde umocní C na druhou. Momentální hodnotu proměnné zjistíme např. PRINT :C .

To vše nyní využijeme a sestavíme konečný program:

```
TO MOCNINA :C :E
MAKE "D 1
REPEAT :E [MAKE "D :C* :D] PRINT :D
END
```

## Chyby se musí opravit

Velmi často se Vám zpočátku stane, že se při definování nové procedury dopustíte chyby. LOGO Vám to oznámí chybovým hlášením. Chybnou definici lze opravit a nemusí se vymazávat definice stará. Opravy a úpravy dříve definovaných slov se provádějí v tzv. editoru. Do editoru přejdete pomocí příkazu EDIT, který lze zkrátit na ED. Za příkazem nutno napsat jméno procedury, kterou chcete opravit. Samotné ED zobrazí do editoru posledně definovanou proceduru.

Na obrazovce se dole objeví nápis LOGO EDITOR, nahoře je výpis definice a blikající kurzor. Ten můžete pomocí šipkových kláves libovolně posouvat po definici, chybné místo vymazat pomocí DELETE a znovu jej napsat. Lze vkládat i celá slova a to tak, že je na příslušném místě napíšete, původní text se rozestoupí. Nový řádek lze vložit tak, že na konci původního řádku, za kterým má být nový, stisknete ENTER.

Máte-li vše opraveno, vrátíte se do výkonného režimu takto: stiskem obou shiftů nastavíte extended mód, vpravo dole se musí objevit E. Pak stisknete C a návrat je proveden. Od toho okamžiku platí opravená definice.

## Pozor na proměnné!

Při zacházení s proměnnými dejte pozor na to, aby Vám nevznikl omyl díky charakteristické vlastnosti LOGO: dává se přednost lokálnosti proměnné před její globálností. Lokální proměnná je taková, která existuje jen uvnitř procedury, pro níž je nadefinována. Jakmile z procedury vystoupíte, Logo se už k dané proměnné nehlásí! Zkuste si to na příkladu:

```
TO PRIDEJ :X
MAKE "X :X+ 1
END
```

Tato procedura zvětší hodnotu proměnné o jednu. Ale proměnná X se zvětší jen uvnitř procedury PRIDEJ a ven z ní nevystoupí! Zadejte PRIDEJ 4. Procedura se vykoná a X má hodnotu 5. Ale zkuste si ji nechat vytisknout pomocí PRINT :X - místo 5 se objeví chybové hlášení "X nemá hodnotu", LOGO se k proměnné X nezná!

Proto je LOGO vybaven zvláštní pamětí, do které lze hodnotu proměnné uložit před výstupem z procedury a ona tam i po výstupu zůstane. Uložení se provede příkazem OUTPUT proměnná. Přeneste si do editoru definici slova PRIDEJ a před END doplňte nový řádek: OUTPUT :X Pak se z editoru vraťte a zadejte PRIDEJ 4. Objeví se chybové hlášení "Neudal jste, co udělat s 5". To znamená, že hodnota 5 z procedury vystoupila (OUTPUT znamená výstup), ale nebylo udáno, co s ní udělat. Zadejte tedy znovu: PRINT PRIDEJ 4 a hodnota 5 se vytiskne. Zkuste nyní PRINT :X a opět zjistíte, že LOGO proměnnou X nezná.

Ale i to lze napravit. Znovu přejděte do editoru pomocí příkazu ED PRIDEJ a opravte v definici MAKE "X na MAKE "Y. Tím přiřadíte zvětšenou hodnotu proměnné Y, se kterou procedura nepracuje. Vystupte z editoru a zadejte znovu PRINT PRIDEJ 4. Když potom zadáte PRINT :Y, hodnota proměnné Y se vytiskne. LOGO tuto proměnnou i její hodnotu zná i po výstupu z procedury. Takovou proměnnou nazýváme globální.

Při užívání OUTPUT dejte pozor; cokoli za OUTPUT se již neprovede. OUTPUT musí být v posledním řádku před END!

## Program na závěr

Na závěr tohoto stručného úvodního kursu uvedeme příklad, jenž názorně uvede programovací možnosti LOGO.

Nejprve nadefinujte TROJUHELNIK o straně :N

```
TO TROJ :N
REPEAT 3 [FD :N LT 120]
END
```

Vyzkoušejte zadáním TROJ 50, uvidíte rovnostranný trojúhelník. Podobně můžete nadefinovat OBDELNIK, CTVEREC apod. Nyní nadefinujeme proceduru ZNACKA, která zavede proměnné TVAR a VEL (velikost) jako seznamy, pomocí SENTENCE SE tyto Seznamy spojí a jejich spojení provede, tedy nakreslí zadaný tvar se zadanou velikostí:

```
TO ZNACKA :TVAR :VEL
RUN SE :TVAR :VEL
END
```

Odzkoušíme pomocí ZNACKA "TROJ 50 nebo ZNACKA [TROJ] 50. Nakreslí se opět trojúhelník. Zde už vidíte, že si můžeme zvolit druh obrazce, místo "TROJ lze zadat i "OBD apod. Nyní definujeme PRAPOREK, kresbu praporku o tvaru značky:

```
TO PRAPOREK :TVAR :VEL
FD :VEL/2
ZNACKA :TVAR :VEL/2
BK :VEL/2
END
```

Proceduru vyzkoušejte PRAPOREK "TROJ 50. Nakreslí se Vám praporek tvaru trojúhelníka, poloviční velikosti než předtím. Další procedura bude kreslit 4 praporky vzájemně pootočené o pravý úhel:

```
TO OBRAZEK :TVAR :VEL
REPEAT 4 [PRAPOREK :TVAR :VEL RT 90]
END
```

Odzkoušejte pomocí OBRAZEK "TROJ 80 (80 je dobře dělitelno dvěma). A nyní můžete vychutnat krásu programování LOGO: smažte obrazovku pomocí CS a zadejte nejprve

```
OBRAZEK [OBRAZEK [TROJ]] 80          a potom
OBRAZEK [OBRAZEK [OBRAZEK [TROJ]]] 80
```

S podivem zjistíte, co lze zadáním tvaru a počáteční velikosti nakreslit. Zde vidíte výhodu užívání Seznamů, proměnnou je zde tvar obrazce! a jeho počáteční velikost, která se ale vůbec nenakreslí. Program je převzat ze ZX-User Club 7-8/85, úpravy provedl ing J. Timar, CSc.

## Závěrem

Tento kurs je míněn jako stručný a úvodní, pro rychlé seznámení se zvláštnostmi a výhodami jazyka LOGO pro ty, kteří nemají k dispozici manuál. Nemůže tedy nahradit manuál programu nebo učebnici jazyka LOGO. Ty nejsou bohužel v češtině k dispozici. Začátečníky upozorňujeme, že zvláště problematika programování s použitím Seznamů je velmi složitá a zde byly postiženy jen její základy.

Jazyk LOGO však považujeme za velmi vhodný pro výuku mládeže a dětí, kde tvoří logické pokračování kursu programování začínajícího jazykem KAREL.

Všem příznivcům jazyka LOGO přejeme hodně úspěchů.

Autor a Klub elektroniky Svazarmu, pošt.schr.23, Chotěboř 583 01

Natištěno pro ZO SVAZARM Karolinka, květen 1989.

Abecední seznam příkazů jazyka Sinclair LOGO ver. 1.6 pro počítač Sinclair ZX Spectrum 48K

---

Příkazy grafiky želvy jsou uvedeny bez jejich významů, které jsou popsány na straně 7. Příkazy, v jejichž významech je uvedeno, že něco zkouší, dávají obvykle výsledek TRUE (pravda) nebo FALSE (nepravda), což si můžete ověřit PRINT příkaz.

| | |
|---|---|
| AND | zkouší pravdivost dvou výrazů |
| ARCSIN | |
| ARCCOS | inverzní trigonom. funkce, např. |
| ARCTAN | PRINT ARCSIN 0.56 |
| ARCCOT | |
| ASCII | vydá číslo kódu znaku v ASCII kódu, např. PRINT ASCII "a |
| BK (BACK) | |
| BRIGHT | |
| BG (BACKGROUND) | |
| BF (BUTFIRST) | vydá zbytek Seznamu bez prvního Slova |
| BL (BUTLAST) | vydá zbytek seznamu bez posledního Slova |
| BYE | přechod do BASICu (návrat RUN či GOTO 1) |
| .BSAVE "jméno adresa délka | SAVE stanoveného rozsahu paměti |
| .BLOAD "jméno adresa | LOAD bytů na určenou adresu |
| CLEAN | |
| CS (CLEARSCREEN) | |
| CT (CLEARTEXT) | smazání textového obrazu |
| CURSOR | vydá pozici kurzoru, např. PRINT CURSOR |
| COPYDEF | kopie definice do jiného Slova |
| COPYSCREEN | kopie obrazovky na připojené tiskárně |
| COT (COTANGENT) | trigonometrické |
| COS (COSINE) | funkce |
| .CONTENTS | s PRINT dá všechny dosud vložené výrazy |
| COUNT [jméno] | dá počet stanovišť v Seznamu |
| .CALL | volá rutinu ve strojovém kódu |
| CATALOG | listuje daty na disketě |
| DEFINE | pojmenovává Seznam |
| DEFINEDP jméno | zkouší, zda je procedura v primitivech |
| DOT [x y] | umístí bod na souřadnicích x,y |
| DIV | děleno PRINT DIV 10 5 je 10/5 |
| .DEPOSIT | uloží datový byte na určenou adresu |
| END | zakončuje definici procedury |
| ED (EDIT) jméno | přenese definici do editoru |
| EQUALP | rovnost dvou výrazů |
| EDNS | smaže program v editoru |
| EMTYP | zkouší, zda je slovo nebo Seznam prázdný |
| ERNS | vymaže všechny jména |
| ERPS | vymaže všechny procedury |
| ERALL | vymaže všechno vložené |
| ER (ERASE) [jméno] | vymaže slovo z pracovní paměti |
| ERN "jméno | vymaže jméno slova |
| .EXAMINE | vydá hodnotu bytu na dané adrese |
| .ERASEFILE | vymaže pole dat |

| | |
|---|---|
| FALSE | nepravdivost výrazu |
| FD (FORWARD) | |
| FENCE | |
| FIRST | vydá obsah prvního stanoviště v Seznamu |
| FPUT "slovo [seznam] | připojí slovo k začátku Seznamu |
| FLASH | blikání grafiky |
| HT (HIDETURTLE) | |
| HEADING | |
| HOME | |
| CHAR číslo | k danému ASCII kódu vydá příslušný znak |
| IF podmínka [akce 1] [akce 2] | podmíněný příkaz: jestliže platí podmínka, pak akce 1, jinak akce 2 |
| IF podmínka [STOP] | vyskočí z podprogramu (místo akce 2 slouží další řádky programu) |
| IF podmínka [OUTPUT něco] | naplní se výstup, skončí procedura, akce 2 = další řádky |
| ITEM | vydá určené stanoviště v Seznamu |
| INT | integer, celočíselná část čísla |
| INVERSE | inversní tisk |
| KEYP | stejné jako INKEYS v Basicu |
| LIST | spojuje Seznamy k sobě |
| LISTP [jméno] | zkouší, zda je výraz Seznamem |
| LT (LEFT) | |
| LAST | vydá obsah posledního stanoviště Seznamu |
| LPUT | připojí stanoviště k poslednímu Seznamu |
| LOAD "jméno | LOAD programu |
| LOADD "jméno | LOAD dat |
| MAKE "jméno hodnota | přiřadí hodnotu proměnné |
| MEMBERP | zkouší, zda výraz má další hodnotu |
| NUMBERP [výraz] | zkouší, zda výraz je číslo |
| NOT | zkouší pravdivost výrazu |
| NAMEP | zkouší, zda výraz má hodnotu |
| NODES | vydá rozsah volné paměti |
| NORMAL | zruší INVERSE |
| OP (OUTPUT) | uloží výsledek programu do přechod.paměti a ukončí proceduru |
| OR | logické NEBO |
| OVER | pro tisk přes sebe |
| PRIMITIVEP | zkouší, zda výraz je primitivem |
| PC (PENCOLOUR) | |
| PD (PENDOWN) | |
| PU (PENUP) | |
| PE (PENERASE) | |
| PX (PENREVERSE) | |
| POS (POSITION) | |

| | |
|---|---|
| PR (PRINT) "text | vytiskne text |
| PRINTON | nastaví pohotovost tiskárny |
| PRINTOFF | vypne pohotovost tiskárny |
| PRODUCT | PRODUCT 10 5 je 10 * 5 |
| .PRIMITIVES | vytiskne všechna primitiva LOGO |
| PO [jméno] | vytiskne definici procedury |
| POALL | vytiskne definice všech vytvořených slov |
| PONS | vypíše jména vytvořených procedur |
| POPS | vypisuje všechny definice procedur |
| POTS | vypíše jména procedur a parametrů |
| REPEAT číslo [...] | opakuje provádění Seznamu [...] |
| RT (RIGHT) | |
| RUN [jméno] | startuje provádění seznamu |
| RANDOM | vydá pseudonáhodné číslo |
| RL (READLIST) | čeká na vstup řádku zakončený ENTER |
| REMAINDER | vydá celočíselný zbytek čísla |
| ROUND | zaokrouhlí na nejbližší celé číslo |
| RC (READCHAR) | čte poslední natypovaný znak |
| RECYCLE | uvolňuje paměť od soustředěných zbytků |
| .RESERVE | reservuje paměť v prázdném LOGu |
| .RESERVED | vydá adresu začátku reservované paměti |
| ST (SHOWTURTLE) | |
| SETH (SETHEADING) | |
| SETSCR (SETSCRUNCH) | |
| SETBG | |
| SETCUR [x y] | nastaví kurzor do pozice |
| SETTC číslo | volba barvy pozadí textu |
| SETX , SETY | |
| SETPOS [x y] | nastavení želvy do pozice |
| SETBR | |
| SHOWNP | |
| STARTROBOT, STOPROBOT | příkazy na ovládání robota |
| SE (SENTENCE) | spojí Seznamy k sobě |
| SUM | sčítá čísla |
| SQRT | druhá odmocnina |
| SIN (SINE) | sinus |
| SHOW | provede slovo a zobrazí jeho výstup |
| SOUND [x y] | zahraje tón x,y |
| SAVEALL "jméno | SAVE slov a proměnných |
| SAVESCR "jméno | SAVE obrazovky |
| SAVED "jméno adresa délka | SAVE bytů |
| SAVE "jméno "jméno procedur | SAVE procedur |
| .SERIALIN, .SERIALOUT, .SETSERIAL | pro Interface 1 |
| SETDRIVE | číslo diskety |
| SETPC | |
| SCRUNCH | |
| STOP | zastaví chod programu |

| | |
|---|---|
| TO | začátek definice procedury |
| TEXT | dá text programu jako Seznam |
| THING | vydá hodnotu výrazu |
| TC (TEXTCOLOUR) | barva psaného textu |
| TRUE | pravdivý výraz |
| TOPLEVEL | vyskočí z procedury do módu příkazů |
| TS (TEXTSCREEN) | přechod z obrazu želvy do text. módu |
| TAN | tangens |
| TOWARDS | |
| TYPE | jako PRINT, ale tiskne za sebou |
| WORD | připojí Seznam ke Slovu |
| WORDP | zkouší, zda výraz je Slovo |
| WINDOW | |
| WRAP | |
| WAIT číslo | čeká daný počet sekund |
| XCOR | |
| YCOR | |

aritmetická znaménka: ( ) = + - * / < >

Kurs jazyka Sinclair LOGO ver. 1.6 * Vydal Klub elektroniky Svazarmu Chotěboř jako metodický materiál jen pro vnitřní potřebu svých členů * (c) JIPOSOFT 1985 *

Natištěno pro ZO SVAZARM Karolinka, květen 1989.