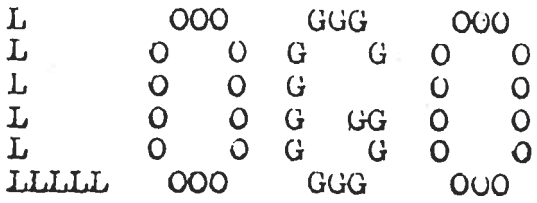


pro ZX Spectrum

Ing. Julius T í m a r

## Kapitola 1
## Logo pro počítač ZX Spectrum

### Úvod

Vítáme vás v LOGU, počítačovém jazyku, který vám dovolí používat váš počítač ke kreslení obrázků, psaní textů, hraní her nebo k počítání. Tento manuál vás naučí, jak se toto vše provádí.

Na rozdíl od takových jazyků, jako je angličtina nebo němčina, Logo nemá mnoho slov nebo gramatických pravidel. Je zde však určitý počet slov, budeme je nazývat základní procedury nebo primitiva, kterým Logo rozumí. Tato primitiva vám dovolí programovat Spektrum pro různé účelyMůžete napsat programy, které kreslí, nebo které manipulují se slovy a se seznamy. Ale nejen to, vy můžete také rozšířit slovník jazyka Logo. Můžete používat základní procedury, které již existují a používat je k vytváření nových procedur. Potom můžete používat tyto nové procedury k sestavování velmi složitých programů.

Tento manuál soustřeďuje programy, které vytvářejí počítačovou grafiku, tj. obrázky na obrazovce počítače. Počítačová grafika vám dovolí jasně vidět to, co se dělá, když se má něco dělat, a tím je proto dobrým úvodem do programování.

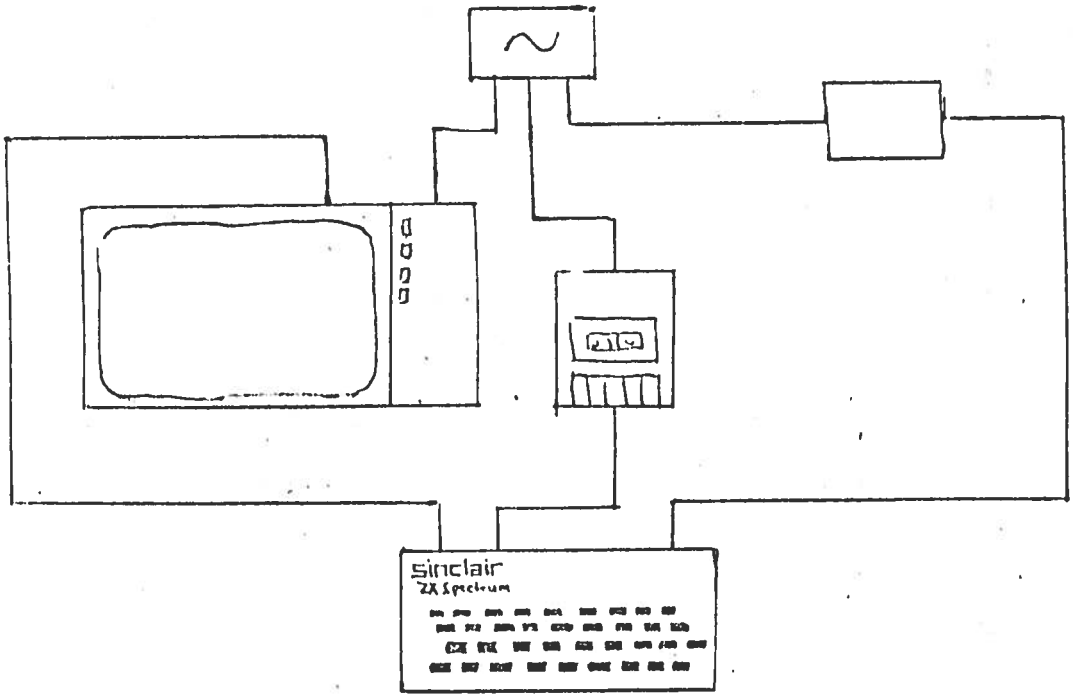
Tento manuál není úplná příručka pro užití, ale ukáže vám, jak začít programovat a opravovat programy, a jak ukládat programy a opět je používat. Při čtení tohoto manuálu můžete také řešit různé problémy, neboť na konci kapitol jsou uvedeny části nazvané Návody, které vám pomohou.

Co potřebujete, abyste mohli začít

K tomu, abyste mohli používat Sinclair LOGO, potřebujete tyto čtyři věci:

1) počítač ZX Sinclair Spectrum s pamětí 48K,
2) televizní přijímač,
3) kazetový magnetofon,
4) program Logo na kazetě.

Zapojte uvedené přístroje podle obrázku.




Zapojte televizor a nastavte 36 kanál. Připojte počítač na síť a po vyladění se vám zobrazí tato zpráva

(C) 1982 Sinclair Research Ltd.

která vám oznamuje, že je vše připojeno v pořádku.

Nyní nahrajeme Logo do počítače. Do magnetofonu vložíme kazetu a hlasitost nastavíme asi na 2/3. Na počítači stiskneme tlačítko J a tím se na obrazovce objeví slovo LOAD. Nyní stiskneme tlačítko SYMBOL SHIFT (dále budeme psát pouze SYS) a dvakrát stiskneme tlačítko P (tlačítko SYS stále stisknuté). Na obrazovce se nyní objeví

LOAD ""

Pokud omylem stiskneme jiné tlačítko, můžeme chybně napsané písmeno nebo slovo smazat tím, že stiskneme tlačítko CAPS SHIFT (dále budeme psát pouze CAPS) a tisknutím číslice Ø mažeme. Když jsme dobře napsali LOAD "" , stiskneme tlačítko ENTER. Nápis na obrazovce zmizí a počítač nyní může číst program z magnetofonu. Na magnetofonu stiskneme tlačítko PLAY. Za chvilku se na obrazovce objeví vodorovné proužky. To znamená, že se program nahrává. Potom se na obrazovce začne kreslit obrázek a bliká nápis LOADING LOGO. Až se celý program dobře nahraje, obrázek zmizí a objeví se text:

WELCOME TO SINCLAIR LOGO
(C) LCSI/SOLI 1984 VER. 1.6

Nyní můžete vypnout magnetofon. Tento text je známkou toho, že se program dobře nahrál a oznamuje vám, že jste vítáni v jazyku Logo. Pokud se tato zpráva neobjeví, došlo k chybě při nahrávání. Zkuste převinout pásek na začátek a zkuste vše znovu, případně změňte hlasitost. Nejmenší hlasitost je ta, která je ještě z počítače slyšet.

Logo používá otazník ? jako označení, že čeká na váš příkaz. Jako kursor se používá blikající čtvereček. Jakmile začneme něco psát, tento čtvereček se pohybuje a ukazuje nám, kam se bude dále psát.

Klávesnice

Klávesnice je popsána v manuálu, který je u počítače ZX Spectrum Zde si popíšeme některé klávesy, které budeme potřebovat v Logu.

BREAK
SPACE — Po stisknutí této klávesy se napíše mezera.

ENTER — Stisknutím této klávesy oznamuje systému Logo, že jsme ukončili psaní textu a chceme, aby se toto provedlo, nebo aby se pokračovalo na novém řádku. Po každém napsaném příkazu musíme stisknout tlačítko ENTER. Instrukce v Logu začíná ? a může mít nejvýše 250 znaků - tj. skoro 8 řádků.

SYS — Pro tlačítko SYMBOL SHIFT budeme používat tuto zkratku.

příklad: — Současným stisknutím tlačítek SYS a P získáme znak pro uvozovky: " .

CAPS — Pro tlačítko CAPS SHIFT budeme používat tuto zkratku. Stisknutím tlačítka CAPS a písmena dostaneme velké písmeno.

C MODE — Mód pro psaní velkých písmen. Stisknutím tlačítek CAPS a 2 dostaneme tento mód. V pravém dolním rohu se objeví písmeno C, které oznamuje C Mód. Programy v Logu budeme psát v C Módu.

L MODE — Mód pro psaní malých písmen. Stisknutím tlačítek CAPS a 2 po druhé se vrátíme do L módu.

DELETE — Když stiskneme tlačítko CAPS a 0, Logo vymaže jeden znak, který byl vlevo před kurzorem.

← — Když stiskneme tlačítka CAPS a 5, Logo posune kurzor o jedno místo doleva.

↓ — Když stiskneme tlačítka CAPS a 6 současně v tzv. EDIT módu (budeme probírat v kapitole 5), Logo posune kurzor o jeden řádek dolu.

↑ — Když stiskneme tlačítka CAPS a 7 současně v tzv. EDIT módu (budeme probírat v kapitole 5), Logo posune kurzor o jeden řádek nahoru.

→ — Když stiskneme tlačítka CAPS a 8 současně, Logo posune kurzor o jeden znak doprava.

E MODE — Když stiskneme tlačítka CAPS a SYS současně, objeví se v pravém dolním rohu písmeno E. V tomto módu můžeme některá tlačítka využít zvláštním způsobem.

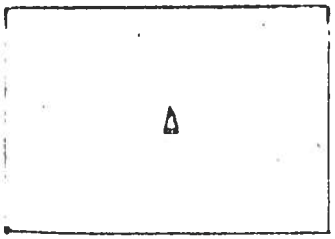
Kapitola 2

Začneme kreslit

Úvod

Nejlépe se naučíme Logo tím, že experimentujeme!

Budeme se učit programování tím, že budeme kreslit obrázky. Budeme řídit želvičku, malý trojúhelníček, který se objeví na obrazovce. Pro ovládání želvičky je mnoho příkazů. V této kapitole si uvedeme nejdůležitější z nich. Napíšeme na klávesnici ?SHOWTURTLE (a stiskneme ENTER)



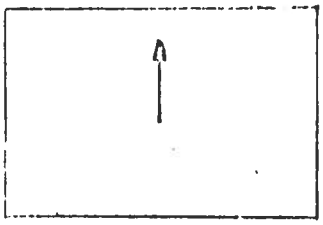
Tento příkaz napíšeme, když chceme, aby se želvička objevila. Na obrazovce se ~~xxxxx~~ uprostřed objeví podlouhlý trojúhelníček ukazující špičkou nahoru. To je želvička, která nám bude kreslit obrázky. Současně se také změnilo rozdělení obrazovky. Pro psaní textů jsou určeny spodní dva řádky, zbytek obrazovky je pro kreslení.

Změna stavu želvičky

Nyní si ukážeme některé příkazy, které mění stav želvičky. Stavem rozumíme polohu želvičky a její nasměrování. Mnohé z těchto příkazů mají zkratky, aby se rychleji napsaly. Když budeme používat nový příkaz, ukážeme si i zkratku.

Dále budeme uvádět určité příkazy, např. ~~xx~~ FORWARD 50, ale vy můžete ~~xxxxxxxxxxxxx~~ provádět tyto příkazy s jinými čísly a sledovat, jak se změní výsledek. Můžete psát velká nebo malá písmena, ale Logo s nimi bude pracovat jako s velkými písmeny. Proto je lepší psát příkazy velkými písmeny. Nezapomeňte po skončení příkazu stisknout tlačítko ENTER, teprve potom Logo vykoná žádanou činnost.

?FORWARD 5Ø nebo pouze FD 5Ø

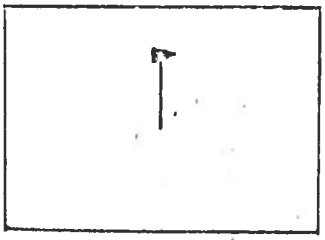


FORWARD je příkaz, který nutí želvičku jít dopředu, a musí mít ještě další vstupní informaci, o kolik kroků dopředu se má želvička posunout. V tomto případě je to číslo 50. Můžete použít i jiné číslo. Pokud to není správné, Logo vám to oznámí.

Mezera mezi slovem FORWARD a číslem 5Ø je velice důležitá, protože Logo rozlišuje příkaz FORWARD 5Ø a slovo FORWARD5Ø. Naopak, napíšete-li více mezer mezi příkazem a vstupní informací, Logo je vynechá.

Ještě upozorňujeme, že želvička změnila svoji polohu, ale nezměnila svoje nasměrování, tj. směr, kterým ukazuje šipka.

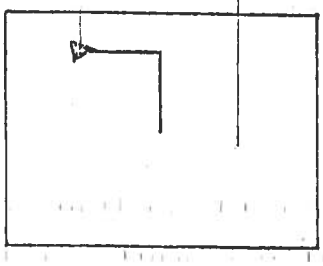
?RIGHT 9Ø nebo pouze RT 9Ø



RIGHT je příkaz, který otáčí želvičkou doprava a musí mít ještě vstupní informaci, o kolik stupňů se má želvička otočit. Otočení o 90 znamená otočení o pravý úhel. Podobně příkaz LEFT 9Ø nebo pouze LT 9Ø otočí želvičku o 90 stupňů vlevo.

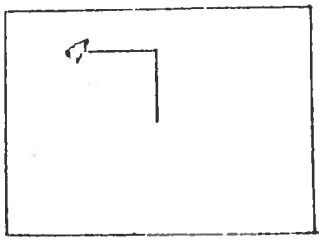
Ještě upozorňujeme, že v tomto případě želvička změnila svoje nasměrování, ale nezměnila svoji polohu.

?BACK 5Ø nebo pouze BK 5Ø

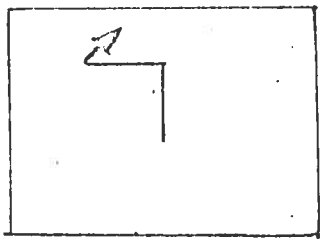


BACK je příkaz podobně jako FORWARD, který nutí želvičku couvnout zpět, takže změní svoji polohu, ale nezmění nasměrování.

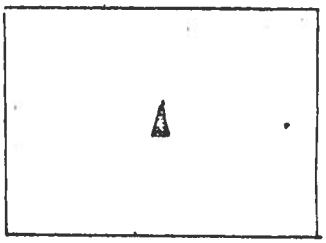
?LEFT 45 nebo pouze LT 45



Želvička se otočí o 45 stupňů vlevo se směru, do kterého byla natočena. Můžete to vidět zřetelněji, když želvičku posunete vpřed příkazem FORWARD 25.



?CLEARSCREEN nebo pouze CS



Jestliže chcete vyčistit obrazovku a začít znovu uprostřed, napíšete příkaz CLEARSCREEN nebo pouze CS. Tento příkaz smaže vše, co bylo nakresleno a vrátí želvičku doprostřed obrazovky a nasměruje ji směrem nahoru.

Návody

Jestliže máte v počítači Logo, můžete zkoušet různé příkazy. Želvička občas nedělá to, co chcete. Je to často zaviněno tím, že jste udělali chybu; v počítačovém jazyce se říká, že jste udělali vešku (anglicky bug).

Nejčastejší chyba u začínajících programátorů je v tom, že zapomenou udělat mezeru mezi příkazem a vstupem. Například FORWARD 5Ø je příkaz v Logu, ale FORWARD5Ø je slowo, které jste mohli už dříve (nebo později) definovat.

~~Odděleny~~ Dva příkazy jsou od sebe odděleny pouze mezerou mezi slovy. Rozdíl mezi slovy FRWARD a FORWARD je pouze v písmenku O, ale pro Logo je to rozdíl mezi tím, že vykoná nějakou činnost, nebo tím, že vám napíše chybovou zprávu.

Jestliže napíšete bez mezery

FORWARD5Ø

Logo vám napíše zprávu

I don't know how to FORWARD5Ø

tj. Logo neví, co znamená příkaz FORWARD5Ø. Takovýmto způsobem dává Logo na vědomí, že v daném příkazu je chyby, ale také se snaží vyjádřit, jaká chyba to asi je. Jestliže chybová zpráva je delší než jeden řádek, Logo na konci řádku přestane psát a vpravo dole se objeví blikající šipka. Po stisknutí ENTER nebo jiné klávesy se objeví zbytek zprávy.

Kapitola 3

První procedura

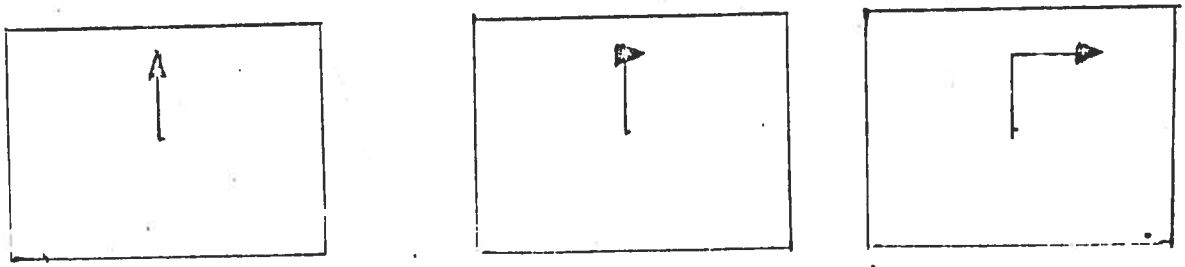
Naučíme želvičku nakreslit čtverec

V Logu je určitý počet slov, kterým Logo rozumi, tj. ví, co má dělat. Tato slova, jako např. FORWARD, RIGHT atd. nazýváme základní procedury nebo též primitiva. Od toho okamžiku, kdy nahrajeme Logo do počítače, ví, co má dělat po příkazu FD 5Ø, ale neví, co dělat po příkazu CTVEREC.

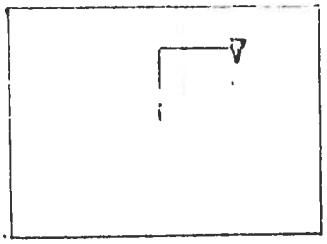
Logo se ale může naučit novým procedurám. Například chcete, aby Logo rozumělo slovu CTVEREC v takovém smyslu, že spojením slov, která už Logo zná, nakreslí na obrazovce čtverec. Tím vytvoříte novou proceduru CTVEREC. Novou proceduru můžete nazvat libovolným jménem, ale nesmí se jmenovat jako některá základní procedura.

Když použijeme příkazy FORWARD a RIGHT, můžeme naučit Logo nakreslit čtverec.

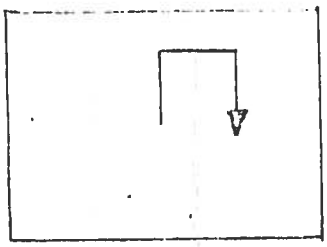
?FD 3Ø ?RT 9Ø ?FD 3Ø



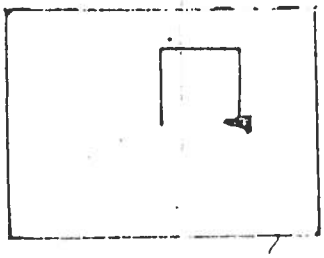
?RT 90



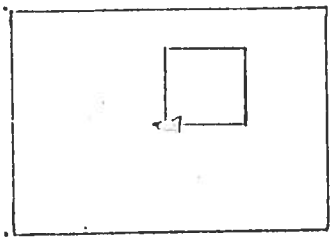
?FD 30



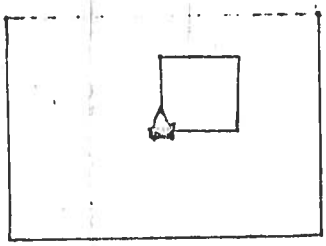
?RT 90



?FD 30



?RT 90



V tomto případě jsme si zvolili číslo 30 jako vstup pro příkaz FD, ale mohli jsme zvolit i jiné číslo. Úhle otočení avšak musí být $90^{o}$, jinak bychom nedostali čtverec!

Než budeme definovat novou proceduru, musíme si pro ni zvolit jméno. K Proceduru, která bude kreslit, nazveme CTVEREC. Pro definici nové procedury použijeme slovo TO a dále jméno procedury, kterou chceme definovat. Logo se přepne do zvláštního módu, kdy můžeme definovat proceduru. Napíšeme

```
XTMXTMW
?TO CTVEREC
```

A nyní budeme definovat, jak nakreslit čtverec:

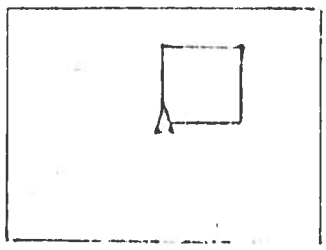
```
>FD 30 RT 90
>FD 30 RT 90
>FD 30 RT 90
>FD 30 RT 90
>END
```

V tomto módu užívá Logo znak > místo znaku ? , a tím nám dává na vědomí, že jsme v módu definování procedury. To znamená, že Logo neprovádí napsané příkazy, nýbrž si je ukládá do paměti. Slovo END, které napíšeme nakonec, znamená pro Logo, že má ukončit mód definování procedury. Logo ukončí definování procedury a oznámí nám

```
RØW CTVEREC defined
?
```

Od této chvíle můžeme používat proceduru CTVEREC a Logo ví, co má při této proceduře dělat. Znak ? na začátku řádku oznamuje, že Logo čeká na příkaz, který bude ihned provádět. Vyzkoušíme nadefinované slovo tím, že ho napíšeme:

?CTVEREC



Návody

Předpokládejme, že procedura CTVEREC nepracuje správně, např. při definování jsme napsali chybný příkaz. Později se naučíme opravovat definovanou proceduru. Nyní si ukážeme, jak se ruší definovaná procedura, použijeme příkaz ERASE. V našem případě napíšeme

?ERASE "CTVEREC

kde před jméno procedury napíšeme znak " , tj. uvozovky. Pozor, Logo nám neoznámí, že proceduru vymazalo, pouze napíše znak ? a čeká na další příkaz. O tom, že Logo nezná proceduru CTVEREC se přesvědčíme např. tím, že tuto proceduru vyvoláme. Napíšeme ?CTVEREC a Logo nám vypíše zprávu

I don't know how to CTVEREC

tj. Logo neví, co je to CTVEREC.

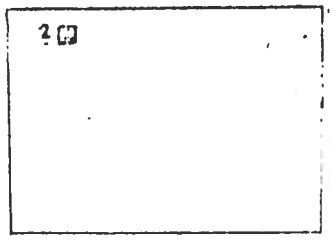
Kapitola 4

TEXTSCREEN, PRINT a REPEAT

TEXTSCREEN

Jestliže chceme kreslit, Logo vytvoří grafickou obrazovku, kde pro obrázky je reservováno horních 20 řádků, a pro texty spodní 2 řádky. Jestliže chceme psát delší texty, pak v grafické obrazovce bychom viděli pouze poslední dva napsané řádky, a to je někdy málo. Když chceme použít celou obrazovku na psaní textu, použijeme příkaz TEXTSCREEN nebo zkráceně TS. Obrazovka se vymaže a kurzor se nastaví do levého horního rohu.

```
?TEXTSCREEN
```




Logo má příkaz, kterým na obrazovku píše text. Je to příkaz PRINT . Napíšeme např.

```
?PRINT [JAK SE MAS?]
```

kde znak [ napíšeme současným stisknutím tlačítek SYS a Y, a pro napsání znaku ] stiskneme tlačítka SYS a U. Když po napsání PRINT [JAK SE MAS?] stiskneme ENTER, Logo provede

```
JAK SE MAS?
?
```

a čeká na další příkaz. Při psaní příkazu PRINT nesmíme vynechat mezeru před vstupní informací.

Může se stát, že při psaní uděláme chybu. Napíšeme např.

```
?PRINT [JK SE MAS?]
```

Než stiskneme tlačítko ENTER, přečteme si příkaz, který jsme napsali, a vidíme, že tam je chyba. Stiskneme proto DELETE, tj. tlačítko CAPS a Ø , a tím se poslední znak smaže. Tiskneme Ø tak dlouho, až na obrazovce zbude

```
?PRINT [J
```

a nyní napíšeme správně

```
?PRINT [JAK SE MAS?]
```

a stiskneme ENTER

```
JAK SE MAS?
```

Tlačítko DELETE , t.j. CAPS a Ø je jedno z mnoha tlačítek, které se užívají při editaci, tj. opravách a změnách. Více si o editaci povíme v následující kapitole. Nyní můžeme experimentovat s příkazem PRINT . Jestliže chceme vypsat více slov, napíšeme je do hranatých závorek [ ]. Když chceme napsat pouze jedno slovo, použijeme uvozovky jako v tomto případě:

```
?PRINT "AHOJ
AHOJ
```

Vymazání textové obrazovky

Pro vymazání textové obrazovky slouží příkaz CLEARTEXT nebo CT. Vyzkoušíme si to:

```
?PRINT [JAK SE MAS?]
JAK SE MAS?
?PRINT "AHOJ
AHOJ
?CLEARTEXT
```

```
?
```

Když ale pracujeme v grafickém obrazovce, potom příkaz CT vymaže spodní dva řádky textu.

Nyní napíšeme program, který používá příkaz PRINT.

```
?TO POZDRAV
>SHOWTURTLE
>PRINT [DOBRY DEN]
>END
POZDRAV defined
?POZDRAV
```

příkaz REPEAT

Jestliže chceme v Logu nějakou část vícekrát opakovat, můžeme použít příkaz REPEAT. Za tento příkaz dáme číslo, které značí počet opakování a dále do hranatých závorek seznam příkazů, které chceme opakovat. Tlačítko ENTER stiskneme až po napsání všech příkazů, tj. vše píšeme jakoby na jeden řádek. Zkusíme

```
?REPEAT 4 [POZDRAV]
```

a Logo provede

```
DOBRY DEN
DOBRY DEN
DOBRY DEN
DOBRY DEN
```

Logo může opakovat i vícekrát. Můžeme zkusit

```
?TS
?REPEAT 1ØØ [PR [JA JSEM NEJLEPSI]]
```

```
JA JSEM NEJLEPSI
JA JSEM NEJLEPSI
JA JSEM NEJLEPSI
JA JSEM NEJLEPSI
JA JSEM STOPPED!!!
```

Na obrazovce se píše text, až se celá obrazovka zaplní. Chceme-li zastavit výpis, stiskneme tlačítka CAPS a BREAK/SPACE současně a chvilku podržíme. Logo přeruší provádění programu a vypíše zprávu

```
STØPPED!!!
```

Zvláštní tlačítka

| | |
|---|---|
| SYS P | " |
| CAPS BREAK/SPACE | STOPPED!!! |
| CAPS Ø | DELETE |
| SYS Y | [ |
| SYS U | ] |

Kapitola 5

Editor pro Sinclair Logo

Úvod

Když definujete novou proceduru pomocí příkazu TO, často zjistíte, že jste někde v předcházejícím řádku udělali chybu. Chcete ji opravit, ale nejde to, museli byste celou proceduru vymazat a napsat znovu. To by ale nebylo efektivní.

Jestliže chcete opravit již napsaný program, použijeme Editor. Logo vám vypíše celý program do textové obrazovky. Nyní můžete posunovat kurzor po celé obrazovce, opravíte chybné místo a definujete proceduru znova. Tak postupujeme, až program funguje správně. Při použití Editoru se ale smaže obrázek nebo text na obrazovce, neboť Editor používá svoji editorovou obrazovku.

EDIT

Pro vyvolání Editoru se používá příkaz EDIT nebo zkráceně ED. Za tímto příkazem píšeme jméno procedury, kterou chceme opravit buď v hranatých závorkách, nebo ve předu s uvozovkami. Mezi uvozovkami a jménem procedury neděláme mezeru.

Poznámka: Jméno procedury nemůže obsahovat mezeru, ale může obsahovat i číslice. Jméno procedury může být libovolně dlouhé, ale nesmí to být jmeno základní procedury.

Zkusíme si opravit proceduru CTVEREC. Pokud ji máme v paměti počítače, napíšeme

```
?EDIT "CTVEREC
```

```
TO CTVEREC
FD 3Ø RT 9Ø
FD 3Ø RT 9Ø
FD 3Ø RT 9Ø
FD 3Ø RT 9Ø
END
```

Vidíme, že počítač vypsal proceduru tak, jak jsme ji nadefinovali. Kurzor je připraven v levém horním rohu. Na spodním řádku obrazovky je text

```
LOGO EDITOR (C) SOLI/LCSI
```

který nám oznamuje, že jsme v editorové obrazovce.

Nyní, když máme nastavenu editorovou obrazovku, můžeme pomocí šipek posunovat kurzor, a měnit příkazy. Pokud chceme nějaký příkaz přidat, pouze si nastavíme kurzor na příslušné místo a příkaz napíšeme. Nesmíme zapomenout na mezeru za příkazem. Když stiskneme tlačítko ENTER, pak vše, co je za kurzorem, se napíše na nový řádek.

Když chceme nějakou část programu smazat , najedeme kurzorem za tuto část a tisknutím DELETE mažeme jeden znak za druhým. Nesmíme to dělat rychle, protože Editor maže pomalu. Nejsložitější oprava je, jestliže chceme místo jednoho příkazu napsat příkaz jiný. Pak je nejlepší starý příkaz smazat, a nyní může na toto místo přidat příkaz nový. Např. můžeme v proceduře CTVEREC všude místo 3Ø napsat 5Ø, tj. smažeme 3 a napíšeme 5.

Editor nám ale umožňuje ještě jednu věc, která je občas velmi potřebná. Někdy se rozhodnete provádět opravy, s chutí se do toho pustíte, a najednou zjistíte, že všechno je jinak. Co teď? Takto opravený program se vám nelíbí, chtěli byste se znovu podívat na původní program. V tomto případě stisknete tlačítka CAPS a BREAK/SPACE. Editor smaže obrazovku a napíše

STOPPED!!!

a ponechá váš původní program beze změny. Zkuste si v proceduře CTVEREC něco změnit a stiskněte CAPS a BREAK/SPACE. Nyní si znovu vyvoláme proceduru CTVEREC tím, že napíšeme

ED "CTVEREC

a skutečně vidíme, že původní program zůstal beze změny. Protože tento program pracuje dobře, nebudeme na něm nic měnit a ukončíme editaci tím, že stiskneme tlačítka CAPS a BREAK/SPACE.

Editor můžeme použít také k definování nové procedury. Výhoda proti definování pomocí příkazu TO je v tom, že během definování můžeme provádět libovolné změny. Zkusíme si to.

?ED "CTVEREC1

Logo vyčistí obrazovku a napíše

TO CTVEREC1

a protože tato procedura není definována, Logo nic nenapíše a čeká na vaše příkazy.

Nyní píšeme příkazy tak, aby Logo mohlo provést proceduru CTVEREC1

```
FD 40 RT 90
FD 40 RT 90
FD 40 RT 90
FD 40 RT 90
END
```

Nejprve nastavíme kurzor na nový řádek pomocí tlačítek CAPS a 6, potom píšeme jeden řádek za druhým, a na konci stiskneme ENTER, tím se nám kurzor posune na nový řádek. Proceduru ukončíme příkazem END. Tím se ale ještě procedura CTVEREC1 nedefinovala, neboť pořád jsme ještě jsme v Editoru.

Pozor: Nyní neukončíme činnost Editoru tím, že stiskneme CAPS a BREAK/SPACE. Tím bychom proceduru CTVEREC1 nedefinovali. A pokud bychom to náhodou učinili, a napsali ED "CTVEREC1, opět bychom dostali prázdnou obrazovku. Ale kdybychom napsali pouze EDIT nebo ED, vrátí nám Editor poslední proceduru, kterou jsme editovali.

Vrátíme se k naší proceduře CTVEREC1. Zatím je v Editoru, a my ji chceme dostat do Loga. K tomu slouží tlačítko E MODE C, tj. musíme nejprve stisknout tlačítka CAPS a SYS tak, aby v pravém dolním rohu bylo písmeno E, a nyní stiskneme tlačítko C. Text na obrazovce se smaže, a Logo nám oznámí

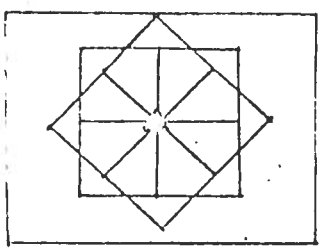
```
CTVEREC1 defined
```

a tím nám dává na vědomí, že proceduru CTVEREC1 již zná. Můžeme se o tom přesvědčit, když napíšeme

```
?CTVEREC1
```

a Logo nakreslí čtverec se stranou 40 jednotek. Když znovu napíšete CTVEREC1, želvička znovu nakreslí čtverec, ale protože už je nakreslený, tak ho objede. Zkusíme proto želvičku otočit o 45 stupňů vpravo. Napíšeme

```
?REPEAT 8 [CTVEREC1 RT 45]
```



a dostaneme takovýto obrázek. Napíšeme proceduru, která tento obrázek nakreslí a nazveme ji HVEZDA. Napíšeme ji v Editoru:

```
?EDIT "HVEZDA
TO HVEZDA
REPEAT 8 [CTVEREC1 RT 45]
END
```

Nyní stiskneme tlačítko E MODE C (tj. CAPS a SYS - aby byl E mód, a potom C). Logo odpoví

```
HVEZDA defined
```

a hned tento program vyzkoušíme:

```
?HVEZDA
```

a vidíme, že program správně pracuje. Želvičku, která je uprostřed obrázku, ~~xkwxá~~ schováme příkazem

```
?HIDETURTLE    nebo pouze HT
```

Když opět chceme zobrazit želvičku, napíšeme SHOWTURTLE, nebo pouze ST.

SETSCRUNCH

Jestliže nakreslíte čtverec, který na obrazovce vypadá jako obdélník, je chyba ve vašem televizoru, a nikoliv v Logu. Ale pomocí příkazu SETSCRUNCH nebo pouze SETSCR můžeme změnit rozměry obrázku. Napíšeme

```
?SETSCR [5Ø 1ØØ]
?CS CTVEREC1
```

a Logo nyní nakreslí čtverec skutečně jako obdélník. Zkuste dát do příkazu SETSCR jiná čísla do hranatých závorek a uvidíte, co se stane. Můžete zkusit např.:

```
?SETSCR [5ØØØ 5ØØØ]
?CS HVEZDA
```

Při tomto nastavení želvička vyběhla z obrazovky, ale z druhé strany se do ní opět vrátila. Tento režim si vysvětlíme později.

K normálnímu zobrazení se vrátíme příkazem

```
?SETSCR [1ØØ 100]
```

Velká a malá písmena

Program v Logu můžete psát pomocí malých nebo velkých písmen. Písmena se přepínají pomocí tlačítka CAPS 2. Pokud vpravo dole je písmeno malé l , budou se psát malá písmena, pokud je dole písmeno velké C , budou se psát velká písmena.

Program můžeme psát malými písmeny, ale Logo si při definici procedury převede malá písmena na velká, takže veškeré názvy procedur budou velkými písmeny. Pouze text, který se má tisknout po instrukci PRINT zůstane zachován. Zkusíme si to na malém příkladu:

```
?to usmev
>pr [pr "zert]
>end
```

Logo definuje proceduru a napíše

```
USMEV defined
```

t.j. název převedlo na velká písmena. Vypíšeme si nyní tuto proceduru

```
?ed "usmev
```

a Logo napíše

```
TO USMEV
PR [pr "zert]
END
```

Protože Logo si příkazy převádí na velká písmena, bude lépe, když od počátku budeme psát procedury velkými písmeny.

Poznámky: Jestliže zadáme příkaz pouze EDIT nebo ED, Editor ukáže poslední editovanou proceduru. To je výhodné, jestliže některou proceduru opravujeme vícekrát za sebou, podruhé stačí napsat pouze ED. Jestliže jsme dosud žádnou proceduru needitovali a napíšeme ED, Editor zobrazí prázdnou obrazovku. Pokud se nám zobrazí jiná procedura, než chceme editovat, stiskneme CAPS a BREAK/SPACE a můžeme zadat k editaci správnou proceduru.

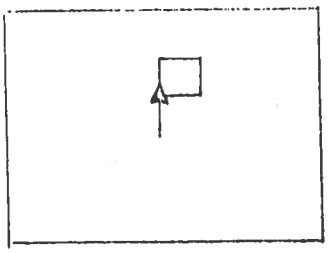
Vytváření vlastních procedur

Jestliže jste vytvořili nějakou proceduru, můžete ji dále používat jako kteroukoliv jinou základní proceduru, např. okamžitě ji použít v další proceduře. Toto jednoduché používání vlastních procedur dělá z Loga velmi silný prostředek.

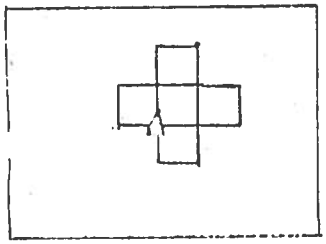
PRAPOREK, KRIZEK, PRAPOREKZPET, PRAPORKY, MNOHOPRAPORKU

Nyní si ukážeme několik procedur, které využívají proceduru CTVEREC4.

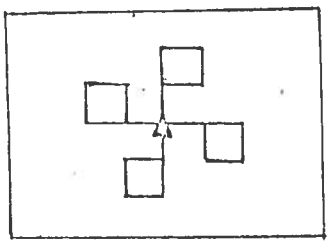
```
?TO PRAPOREK
>FD 30
>CTVEREC4
>END
PRAPOREK defined
?PRAPOREK
```



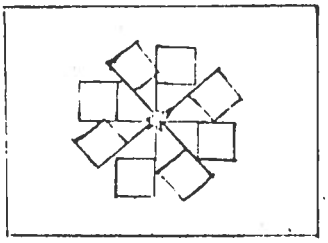
```
?TO KRIZEK
>REPEAT 4 [PRAPOREK RT 90]
>END
KRIZEK defined
?KRIZEK
```



```
?TO PRAPOREKZPET
>PRAPOREK
>BK 30
>END
PRAPOREKZPET defined
?TO PRAPORKY
>REPEAT 4 [PRAPOREKZPET RT 90]
>END
PRAPORKY defined
?PRAPORKY
```



```
?TO MNOHOPRAPORKU
>PRAPORKY
>RT 45
>PRAPORKY
>END
MNOHOPRAPORKU defined
?MNOHOPRAPORKU
```



Procedury PRAPOREK a PRAPOREKZPET nutí želvičku, aby nakreslila stejný obrázek, ale želvička je v jiném stavu. Želvička má stejné nasměrování, ale jinou polohu.

PRAPOREKZPET vrátí želvičku na to místo, z kterého začala. Rozdíl v poloze želvičky je vidět na procedurách KRIZEK a PRAPORKY. Procedura KRIZEK používá proceduru PRAPOREK čtyřikrát, zatímco procedura PRAPORKY používá proceduru PRAPOREKZPET také čtyřikrát.

Jestliže nyní vypnete počítač, ztratí se vám všechny nadefinované procedury. V další kapitole si popíšeme, jak nahrát programy na kazetu.

Zvláštní tlačítka:

| | |
|---|---|
| CAPS 5 | ← |
| CAPS 6 | ↓ |
| CAPS 7 | ↑ |
| CAPS 8 | → |
| CAPS Ø | DELETE |
| CAPS BREAK/SPACE | opuštění Editoru |
| E MODE C | definování procedury z Editoru |

Kapitola 6

Nahrávání na magnetofon

Úvod

Když definujete procedury na počítači, Logo si tyto procedury pamatuje a vy je můžete používat. Jakmile ale vypnete počítač, všechny programy se smažou.

Chcete-li si uchovat definované procedury, musíte je nahrát na magnetofonovou kazetu. K tomu slouží příkaz SAVE. Později si můžete tyto procedury znovu přehrát do počítače pomocí příkazu LOAD. Tímto způsobem můžete jednou definované procedury kdykoliv později používat.

SAVE, SAVEALL

K nahrávání programů na kazetu můžete použít libovolný kazetový magnetofon, třeba ten, z kterého jste nahrávali Logo do počítače. Je dobré, když má magnetofon zabudovaný mikrofon a když má magnetofon počítadlo. Před tím, než budete nahrávat program, vytáhněte kablík ze zdířky EAR a zapněte kablík do zdířek MIC počítače a MIC magnetofonu. Pokud máte nějaké pochybnosti, podívejte se do příručky k počítači Spectrum.

Při nahrávání programů na kazetu vlastně vytváříme soubory, neboli file (čti fajl). Do tohoto souboru si můžeme nahrát buď jen předem určené procedury, nebo všechny procedury, které jsou v počítači. Pro nahrávání jenom některých procedur slouží příkaz SAVE, pro nahrání všech procedur slouží příkaz SAVEALL.

Před vlastním nahráváním si musíme připravit kazetu. Najdeme si na kazetě místo, které je volné. Pokud máme mikrofon, nahrajeme si na pásek jmeno nahrávky, tj. jméno souboru, jak jej budeme za chvíli nahrávat z počítače. Vynulujeme počítadlo. Nyní napíšeme na počítači:

?SAVE "SOUBOR "CTVEREC

což znamená, že do souboru se jménem SOUBOR budeme nahrávat proceduru CTVEREC. Dále stiskneme na magnetofonu tlačítka REC a PLAY, a na počítači tlačítko ENTER. Jakmile se program nahrává, objevují se na obrazovce pruhy. Po ukončení nahrávání se na obrazovce objeví znak ? .

Chceme-li nahrávat na kazetu více procedur, musíme jména procedur dát do hranatých závorek, např.

?SAVE "SOUBOR [CTVEREC PRAPOREK]

a Logo nahraje na kazetu procedury CTVEREC a PRAPOREK. Nesmíme však zapomenout, že procedura PRAPOREK potřebuje proceduru CTVEREC, a kdybychom ji nenahráli, nemohla by procedura PRAPOREK pracovat - ledaže bychom předem vytvořili novou proceduru CTVEREC. Z toho důvodu, abychom na něco nezapomněli, je dobré nahrávat na magnetofon všechny procedury. Napíšeme:

?SAVE "SOUBOR

kde píšeme jenom jméno souboru. Jakmile se nahrává soubor, obrazovka bliká, po skončení se objeví znak ? a můžeme vypnout magnetofon.

Jméno souboru, který nahráváme na kazetu, volíme tak, abychom poznali, oč v tomto souboru jde. Např. můžeme zvolit i jméno některé procedury, která v tomto souboru je zahrnuta. Jméno souboru nemá nic společného se jmény procedur, slouží pouze k označení souboru na kazetě. Programy, které jsme doposud vytvořili, můžeme nazvat např. CTVEREC nebo PRAPORKY.

LOAD

Pro čtení souboru z kazety slouží příkaz LOAD, za kterým uvádíme jméno souboru. Pokud nevíme jmeno souboru, můžeme napsat jakékoliv jiné jméno, neboť Logo ukazuje jméno přečteného souboru. Proto když Logo nalezne nějaký soubor a vypíše jeho jméno, zastavíme počítač tlačítkem CAPS BREAK/SPACE, vrátíme magnetofon a čteme kazetu se správným jménem. Pro náš případ napíšeme

?LOAD "SOUBOR

jestliže chceme číst z kazety soubor s názvem SOUBOR.

Při čtení z kazety musíme mít připojen kablík EAR - EAR, nastavenou kazetu před nahraným souborem. Stiskneme ENTER a pustíme magnetofon tlačítkem PLAY. Logo hledá začátek souboru, a když ho najde, napíše

SOUBOR LOG

kde LOG znamená, že se jedná o soubory pro LOGO. Obrazovka začne blikat a po přečtení jednotlivých bloků se objevují jména procedur, která jsou v souboru nahrané.

Upozornění: Někdy je pausa mezi bloky malá a Logo nestačí blok zpracovat. Proto je nutné mezi bloky zastavovat asi na 1 s magnetofon.

Kapitola 7

Želvička má pero a barvy

Úvod

Když se želvička pohybuje, zanechává za sebou čáru, protože má pero, které kreslí. Jestliže chceme želvičku někam posunout, aniž by kreslila, musíme zvednout pero. Můžeme si také xxxxx volit barvu inkoustu, kterým želvička kreslí, dále můžeme volit barvu pozadí, na které se kreslí (tj. barvu papíru), a konečně i barvu rámečku, tzv. border. Tato kapitola pojednává o tom, jak užívat pero a barevnou grafiku.

Příkazy pro ovládání pera

Aby želvička nekreslila, musíme zvednout pero příkazem PENUP nebo pouze PU:

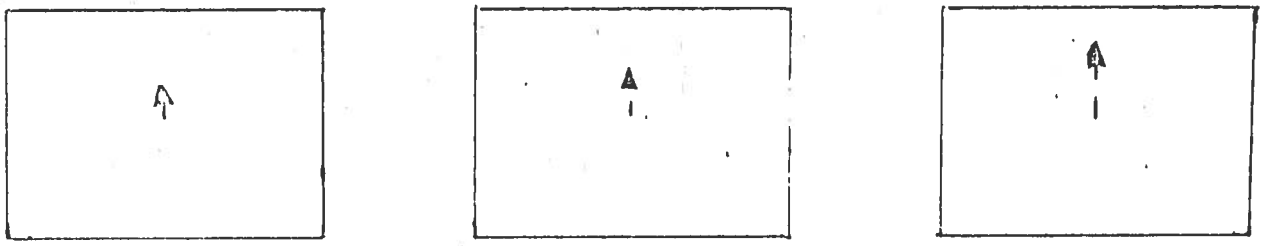
```
?PENUP     nebo pouze PU
```

Aby želvička opět kreslila, napíšeme

```
?PENDOWN     nebo pouze PD
```

Uděláme si příklad na tyto dva příkazy:

```
?FD 2Ø                ?PU FD 2Ø                ?PD FD 2Ø
```

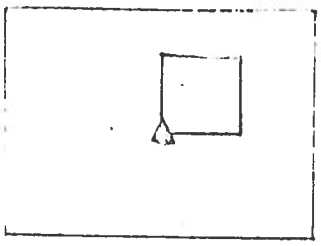


Na obrázku vidíme, že želvička udělala čáru dlouhou 20 jednotek, potom kousek vynechala, a opět udělala čáru 20 jednotek.

Spolu s příkazy PENUP a PENDOWN máme ještě dva příkazy pro ovládání pera. Jsou to příkazy PENERASE (PE) a příkaz PENREVERSE (PX).

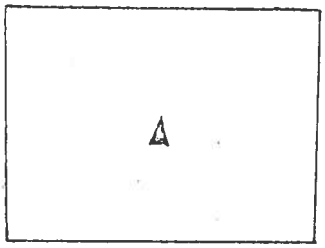
příkaz PENERASE, zkráceně PE nutí želvičku mazat nakreslenou čáru, přes kterou přejíždí. Chceme-li, aby želvička pokračovala v kreslení, dáme příkaz PENDOWN nebo PD. Ukážeme si to na příkladu:

```
?CS PD
?CTVEREC
```



Nyní napíšeme

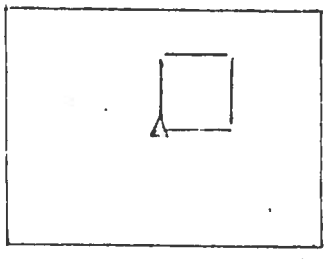
```
?PE CTVEREC
```



a želvička smazala nakreslený čtverec.

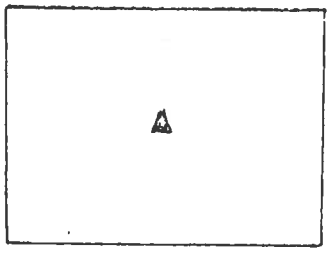
PENREVERSE je trochu zvláštní příkaz, je to kombinace příkazů PD a PE. Jestliže pod želvičkou žádná čára není, potom příkaz PE funguje jako PD, tj. želvička kreslí čáru. Pokud ale přijede na čáru, potom příkaz funguje jako PE, tj. želvička čáru maže. Ukážeme si to na příkladu:

```
?PX
?CTVEREC
```



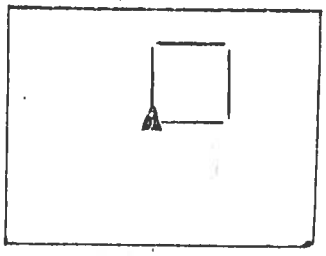
a dále

```
?CTVEREC
```



a opět pokračujeme

```
?CTVEREC
```



Na prvním obrázku želvička nakreslila čtverec, na kterém ale vynechala rohy, přes které jela dvakrát. Na druhém obrazku čtverec smazala, a na třetím ho zase nakreslila. Kdybychom napsali příkaz PD CTVEREC PE CTVEREC , pak želvička nakreslí jenom rohy čtverce. Zkuste si to.

Účinek příkazu PE se ruší každým dalším příkazem, buď PD, PU nebo PR.

Sinclair Logo používá barevnou grafiku

Tato část se dá zcela využít pouze tehdy, máme-li barevný televizor. Na černobílém televizoru vidíme pouze odstíny šedé barvy.

Existují tři druhy barevných změn. Nejprve se může m nit barva pozadí, na kterém se želvička pohybuje příkazem SETBG. Dále můžeme měnit barvu čáry, kterou želvička zanechává příkazem SETPC a konečně můžeme měnit okolí a textové části v grafické obrazovce (spodní dva řádky) příkazem SETBORDER, zkráceně SETBR.

Každý tento příkaz ještě potřebuje vstupní informaci, a tou je číslo, které udává žádanou barvu. Číslo barvy odpovídá názvům, které jsou u tlačítek na počítači ZX Spectrum.

0 black ... černá
1 blue ... tmavě modrá
2 red ... červená
3 magneta ... fialová
4 green ... zelená
5 cyan ... světle modrá
6 yellow ... žlutá
7 white ... bílá

Můžeme si vyzkoušet měnit barvu pozadí:

```
?SETBG 0    dostaneme černou obrazovku
?SETBG 1              tmavě modrá
?SETBG 2              červená
?SETBG 3              fialová
?SETBG 4              zelená
?SETBG 5              světle modrá
?SETBG 6              žlutá
?SETBG 7              bílá
```

Nebo můžete napsat program, který vyzkouší všechny tyto barvy. Použijeme při tom příkaz WAIT 50 který nutí Logo čekat asi 1 s před vykonáním dalšího příkazu.

```
?TO B.POZ
>SETBG Ø WAIT 5Ø
>SETBG 1 WAIT 5Ø
>SETBG 2 WAIT 5Ø
>SETBG 3 WAIT 5Ø
>SETBG 4 WAIT 5Ø
>SETBG 5 WAIT 5Ø
>SETBG 6 WAIT 5Ø
>SETBG 7 WAIT 5Ø
>END
B.POZ defined
```

A nyní tento program vyzkoušíme

```
?REPEAT 3 [B.POZ]
```

příkaz BACKGROUND, zkráceně BG nám dá jako výstup odpovídající číslo barvy pozadí. Zkusíme si to

```
?PR BG
7
```

tj. barva pozadí má číslo 7, je bílé.

Můžeme pozměnit náš program

```
?TO POZADI
>SETBG BG + 1 WAIT 5Ø
>END
POZADI defined

?TO BA.PO
>REPEAT 7 [POZADI]
>END
BA.PO defined
```

Změna barvy pera

Chceme-li změnit barvu čáry, kterou želvička kreslí, provedeme to příkazem SETPC a číslo barvy, která je stejná jako u pozadí. Jestliže změníme barvu pera a dáme příkaz pro textovou obrazovku TEXTSCREEN nebo pouze TS, Logo bude psát text změněnou barvou. Ale text v grafické obrazovce, tj. spodní dva řádky se příkazem SETPC č. barvy nemění. Zkusíme si to v následujícím příkladu.

```
?SETBG Ø
?CS
?SETPC 2 CTVEREC
?RT 45 SETPC 3 CTVEREC
?RT 45 SETPC 4 CTVEREC
```

a nyní napíšeme

```
?RT 45 SETPC Ø CTVEREC
```

a po stisknutí Enter se žádný čtverec nenakreslí, naopak část obrazku se "smaže". Je to proto, že barva pera a barva pozadí jsou stejné - černé, takže nakreslený čtverec není vidět!

Příkaz PENCOLOUR (zkráceně PC) dá jako výstup odpovídající číslo barvy pera. Zkusíme si to

```
?PR PC
Ø
```

tj. barva pera je černá.
Zkusíme si další proceduru

```
?TO TOTO
>REPEAT 8 [FD 3Ø RT 45]
>END
TOTO defined

?CS SETBG 7
?SETPC 1
?TOTO          nakreslí se osmiúhelník

?TO TOTAL                    procedura pro nakreslení kytičky
>CS REPEAT 18 [TOTO RT 2Ø]
>HT
>END
TOTAL defined
```

Nyní budeme měnit barvu pozadí a barvu pera

```
?TO ZM.BAR
>SETPC PC + 1
>SETBG BG + 1
>REPEAT 4 [FD 3Ø RT 45]
>RT 2Ø
>END
ZM.BAR defined
```

Nyní si program vyzkoušíme

?REPEAT 8 [ZM.BAR]

na obrazovce se nakreslí několikanásobný vícebarevný devítiúhelník.

Změna barvy okolí

Příkaz SETBORDER, zkráceně SETBR a číslo mění barvu okolí, tj. okraj části obrazovky, na kterou Logo kreslí. Číslo barvy je stejné, jako v předcházejících případech. Zkusíme

?SETBR 1

a okolí na obrazovce bude mít barvu tmavě modrou.

Návody

Při změně barvy pozadí někdy zjistíte, že to má vliv i na barvu nakreslené čáry. To se nejvíce projevuje, když používáte příkazy PENERASE (mazání) a PENREVERSE (změna čáry).

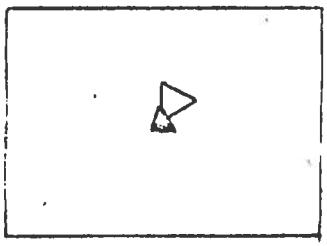
Barvy, které dostanete na obrazovce, záleží velmi na typu použitého televizoru a dále na nastavení barev.

Kapitola 8
Další pohled na editování procedur .

Úvod
Editor pro Logo má velmi široké použití. Můžeme jím opravovat procedury, které jsme již v systému definovali, ale můžeme jím také definovat procedury nové. Tento způsob definování má velké výhody v tom, že okamžitě můžeme opravovat vzniklé chyby, nebo vytvářet proceduru "za chodu". Např. chceme definovat trojúhelník. Normálně píšeme

```
?TO TROJÚHELNIK
>FD 45 RT 12Ø
>FD 45 RT 12Ø
>FD 45 RT 12Ø
>END
TROJUHELNIK defined
```

?TROJUHELNIK



Zahájení činnosti Editoru
Zahájit činnost Editoru můžete několika způsoby, a tím dostanete i poněkud odlišné výsledky. Když napíšeme EDIT nebo pouze ED a dále nenapíšeme jméno procedury, Logo nám vypíše tu proceduru, kterou jsme editovali naposledy. Pokud jsme ještě žádnou proceduru needitovali, Logo vypíše prázdnou editorovou obrazovku.

Když chceme editovat např. proceduru TROJUHELNIK, napíšeme

ED "TROJUHELNIK

a Editor vypíše proceduru TROJUHELNIK . Editor nám umožňuje editovat i více procedur najednou, když napíšeme seznam procedur. Např.

ED [TROJUHELNIK CTVEREC]

a Logo vypíše obě procedury. Pokud napíšeme jméno procedury, která ještě nebyla definována, Logo na obrazovku napíše pouze

TO název

a zbytek nechá prázdný, takže proceduru můžeme hned doplnit. Chceme-li, aby Logo vypsalo prázdnou editorovou obrazovku, napíšeme

ED []

což je vlastně prázdný seznam. Logo nám předloží prázdnou editorovou obrazovku.

Ukážeme si definování nové procedury pomocí Editoru. Budeme definovat proceduru TROJUHELNIK. Pokud ji už máme definovanou, smažeme ji příkazem

```
?ER "TROJUHELNIK
?
```

Pozn.: Logo nám nedává žádnou zprávu, že proceduru smazalo. Teprve pokud o smazání neexistující procedury dá chybovou zprávu. Zkuste znovu napsat ER "TROJUHELNIK.

A nyní můžeme definovat proceduru

```
?ED "TROJUHELNIK
```

Logo nám předloží čistou editorovou obrazovku, pouze nahoře je nápis. Pokračujeme v definici

```
TO TROJUHELNIK
FD 45 RT 12Ø
FD 45 RT 12Ø
FD 45 RT 12Ø
END
```

Na začátku je kurzor na prvním písmenku slova TO. Pomocí tlačítka CAPS 6 najedeme na nový řádek a píšeme FD 45 RT 12Ø, potom stiskneme ENTER a pokračujeme dále. Nakonec napíšeme END (ale nemusíme, Logo to při ukončení editace samo dopíše END), a ukončíme činnost Editoru tím, že definujeme proceduru stisknutím tlačítek E MODE (tj. stiskneme současně CAPS a SYS tak, aby vpravo dole bylo písmeno E) a C. Logo smaže editorovou obrazovku, nastaví textovou obrazovku a napíše

```
TROJUHELNIK defined
```

Ukončení činnosti Editoru

Ukončení činnosti Editoru je možné dvojím způsobem. Když stiskneme tlačítka E MODE C, přerušíme opravování v Editoru a chceme opravenou proceduru (příp. více procedur) definovat. Logo tuto skutečnost potvrdí zprávou, např.

```
TROJUHELNIK defined
```

K opuštění Editoru máme ještě jednu možnost. Jestliže se během oprav rozhodneme, že chceme ponechat původní proceduru beze změny (např. proto, že změny vlastně nemají smysl, nebo chceme provést změny znovu od začátku), opustíme Editor stisknutím tlačítek CAPS a BREAK/SPACE. Logo opustí Editor a zanechá procedury v tom stavu, v jakém byly před startem Editoru.

Souhrn Editačních tlačítek

Pokud jsme v Editoru, máme k dispozici několik kombinovaných tlačítek, které slouží pro práci v Editoru. Nyní si je stručně popíšeme.

| | |
|---|---|
| CAPS 5 | Posune kurzor o jeden znak vlevo |
| CAPS 6 | Posune kurzor o jeden řádek dolů |
| CAPS 7 | Posune kurzor o jeden řádek nahoru |
| CAPS 8 | Posune kurzor o jeden znak vpravo |
| CAPS Ø | Smaže jeden znak před kurzorem |
| E MODE CAPS 5 | Posune kurzor na začátek řádku |
| E MODE CAPS 6 | Posune kurzor na konec obrazovky |
| E MODE CAPS 7 | Posune kurzor na začátek obrazovky |
| E MODE CAPS 8 | Posune kurzor na konec řádku |
| E MODE B | Posune kurzor na začátek textu |
| E MODE E | Posune kurzor na konec textu |
| E MODE N | Posune kurzor na další stránku |
| E MODE P | Posune kurzor na předcházející stránku |
| E MODE Y | Smaže (yanks) zbytek řádku za kurzorem, ale uloží tuto část do vnitřní paměti |
| E MODE R | Vypíše obsah vnitřní paměti |
| E MODE Z | Zapne/vypne pípnutí při stisknutí tlačítka |

Použití editačních tlačítek mimo Editor

Mnoho z těchto tlačítek pracuje i mimo Editor, v tom případě ale pracují pouze na jednom řádku. Řádek v Logu začíná znakem ? a končí tehdy, když stiskneme tlačítko ENTER, a může obsahovat 250 znaků. Než si zkusíme editovat mimo Editor, nejprve opravíme proceduru TROJUHELNIK. Napíšeme

```
?EDIT "TROJUHELNIK
```

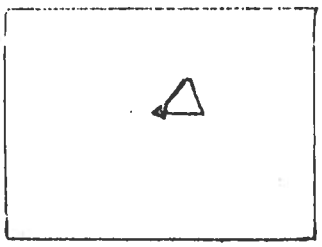
Posuneme kursor na další řádek (CAPS 6) a napíšeme

```
RT 3Ø  ENTER
```

a ukončíme činnost Editorutím, že stiskneme E MODE C.

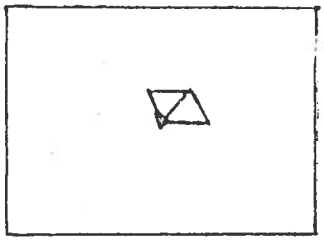
Logo odpoví TROJUHELNIK defined, a nyní tuto opravenou proceduru vyzkoušíme.

?TROJUHELNIK

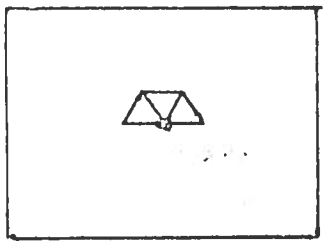


Nyní stiskneme tlačítka E MODE R, Logo vypíše obsah vnitřní paměti, tj. TROJUHELNIK. Do vnitřní paměti se mimo Editor ukládá každý příkaz, který napíšeme. Kurzor je připraven na začátku řádku. Nyní napíšeme

?LT 9Ø SPACE ENTER a Logo nakreslí



Tento postup můžeme opakovat. Stiskneme F MODE R a potom ENTER. Logo nakreslí tento obrázek.



Kapitola 9
Pracovní paměť

Úvod

Pracovní paměť počítače obsahuje všechny procedury, které jste definovali. Logo obsahuje několik základních procedur, které vám umožňují organizovat tuto paměť. Můžete si např. nechat vytisknout seznam procedur, které se v této paměti nacházejí, nebo si vypsat definice některých nebo i všech procedur.

Vypsání procedur

Pro výpis názvů procedur, které jsme definovali do pracovní paměti, slouží příkaz POTS (Print Out TitleS). Nejprve si ale nastavíme textovou obrazovku.

```
?TS
?POTS
TO TROJUHELNIK
TO ZM.BAR
TO TOTAL
  ...
```

Na obrazovce se nyní vypisují jednotlivé názvy procedur, které jsme během práce s Logem nadefinovali. Jestliže chceme zastavit výpis, stiskneme tlačítka SYS S. Po stisknutí libovolného tlačítka pokračuje výpis dále. Druhá možnost, jak zastavit výpis, je stisknutí tlačítek CAPS a BREAK/SPACE. V tom případě Logo oznámí

```
STOPPED!!!
?
```

a čeká na další příkaz (výpis se ukončil).

Pro výpis procedury, tj. příkazů tak, jak byla definována, slouží příkaz POPS (Print Out ProcedureS). Tím se vypisují definice všech procedur. Zkusíme si to:

```
?POPS
TO TROJUHELNIK
RT 30
FD 45 RT 120
FD 45 RT 120
FD 45 RT 120
END
```

```
TO ZM.BAR
SETPC PC + 1
SETBG BG + 1
REPEAT 4 [FD 3Ø RT 45]
RT 2Ø
END
atd.
```

Výpis pozastavíme stisknutím tlačítek SYS S, nebo úplně zastavíme stisknutím tlačítek CAPS BREAK/SPACE.

Když si chceme vypsat pouze jednu proceduru, např. CTVEREC, napíšeme

```
?PO "CTVEREC
TO CTVEREC
FD 3Ø RT 9Ø
FD 3Ø RT 9Ø
FD 3Ø RT 9Ø
FD 3Ø RT 9Ø
END
```

Můžeme si nechat vypsat více procedur najednou, např.

```
?PO [CTVEREC HVEZDA TROJUHELNIK]
```

Logo vypíše všechny tyto procedury. Pamatujme, že tyto příkazy používáme v textové obrazovce, neboť v grafické obrazovce bychom viděli pouze poslední dva řádky. Pokud použijeme tento příkaz v grafické obrazovce, stiskneme tlačítka E MODE R a napíšeme TS SPACE a stiskneme ENTER, a vidíme výpis na čisté textové obrazovce.

Mazání procedur z pracovní paměti

Během práce s Logem někdy zjistíte, že urcitou proceduru již nepotřebujete a chcete ji smazat. Ale buďte opatrní! Jestliže jste si určitou proceduru nenahráli na magnetofon, a smažete ji, pak vám nezbude, než ji znovu napsat. Procedura se maže příkazem ER s názvem procedury, např.:

```
?ER "TROJUHELNJK
```

Chceme-li smazat víc procedur, napíšeme je do seznamu

?ER [TROJUHELNIK CTVEREC PRAPOREK]

a Logo vymaže uvedené procedury. Jestliže uvedené procedury neexistují, Logo napíše zprávu, např.

ER doesn't like TROJUHELNIK as input

to znamená, že procedura ER nemůže vymazat proceduru TROJUHELNIK.

Když chceme uvolnit celou pracovní paměť, tj. vymazat všechny definované procedury, napíšeme

?ERPS

(ERase ProcedureS), a tím se všechny procedury vymažou. Tím se ale nemaže paměť Editoru, takže napíšeme-li ED, vypíšou se na obrazovku naposledy procedury, které byly naposledy editované.

Výpis procedur na tiskárnu

Jestliže k počítači ZX Spectrum máme připojenou tiskárnu, můžeme si procedury nechat vytisknout na papír. Nejdříve musíme programově zapojit tiskárnu příkazem

?PRINTON
?PO "TROJUHELNIK

a tím se vytiskne definice procedury TROJUHELNIK. Chceme-li si nechat vytisknout více procedur, napíšeme jejich názvy do seznamu

?PRINTONN
?PO [TROJUHELNIK CTVEREC]

Po skončení tisku vypneme programově tiskárnu příkazem

?PRINTOFF

Pomocí tisknárny si můžeme nechat na papír vytisknout i obrázky, které jsme nakreslili na obrazovku. K tomu použijeme příkaz COPYSCREEN, např.:

?TOTAL
?COPYSCREEN

a Logo vytiskne vše, co je na obrazovce, kromě spodních dvou řádků.

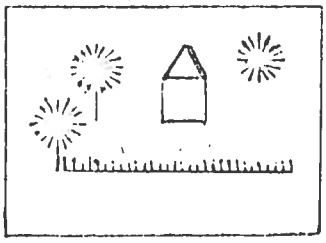
Kapitola 10

První projekt: Kreslíme zahradu

Úvod

Jako první projekt, který provedeme v Logu, bude nakreslení ZAHRADY, která obsahuje DUM, dva STROMY, TRAVNIK a SLUNCE.

Logo má tu vlastnost, že nám dovolí rozdělit tento velký projekt na řadu malých částí. Nejdříve si nakreslíme tento projekt na papír.



Vidíme, že ZAHRADA se skládá z těchto částí:

1) DUM
2) dva STROMY
3) TRAVNIK
4) SLUNCE

Pro každou tuto část budeme definovat zvláštní procedury. Potom je dáme dohromady.

1) DUM

Nejdříve musíme rozdělit si DUM na jednotlivé části. Vidíme, že musíme nakreslit CTVEREC a potom TROJUHELNIK. Napíšeme si nyní tyto procedury. První proceduru nazveme CTVEREC2.

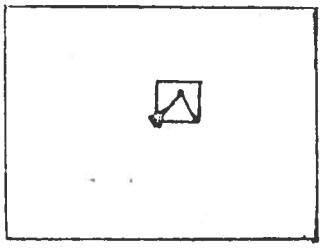
```
?TO CTVEREC2
>REPEAT 4 [FD 45 RT 90]
>END
CTVEREC2 defined
```

Nyní si vypíšeme proceduru TROJUHELNIK, kterou jsme si definovali již dříve:

```
?PO "TROJUHELNIK
TO TROJUHELNIK
RT 3Ø
FD 45 RT 12Ø
FD 45 RT 12Ø
FD 45 RT 12Ø
END
```

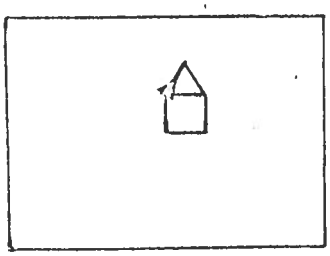
Nyní dáme tyto dvě procedury dohromady a nakreslíme dům:

```
?TO DUM
 CTVEREC2
 TROJUHELNIK
 END
DUM defined
?DUM
```



Ale to není to, co chceme! Provedeme opravu

```
?ED "DUM
CTVEREC2
FD 45
TROJUHELNIK
END
DUM defined
?DUM
```

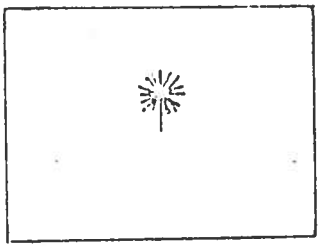


Toto je již lepsí!

2) Dva STROMY

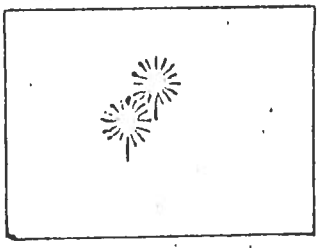
Nejdříve nakreslíme jeden STROM

```
?TO STROM
>FD 5Ø
>REPEAT 36 [FD 3Ø BK 3Ø RT 1Ø]
>BK 5Ø
>END
STROM defined
?HT STROM
```



Nyní můžeme psát proceduru STROMY, která nakreslí strom, posune želvičku a nakreslí druhý strom.

```
?TO STROMY
>STROM
>PU LT 9Ø FD 3Ø LT 9Ø
>FD 5Ø RT 18Ø PD
>STROM
>END
STROMY defined
?STROMY
```



3) TRAVNIK

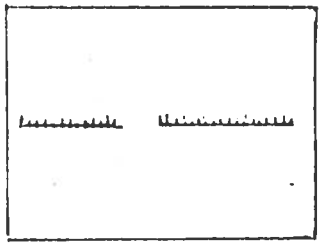
Nyní nakreslíme TRAVNIK, který bude tvořit spodní okraj naší ZAHRADY.

```
?TO TRAVNIK
>REPEAT 45 [FD 1Ø BK 1Ø RT 9Ø
D 5 LT 9Ø]
>END
TRAVNIK defined
```

Poznámka k definici procedury TRAVNIK: Když píšeme cykl REPEAT, dáváme to, co se má opakovat, do závorek [ a ]. Uvnitř nesmíme stisknout tlačítko ENTER, t.j. musíme vše psát na jeden "programový řádek", což ve skutečnosti může být na obrazovce více řádku. Logo na konec obrazovkového řádku dá značku ! , a tím nám dává na vědomí, že programový řádek ještě pokračuje.

Nyní si xxxxx tuto proceduru vyzkoušíme:

```
?TRAVNIK
```

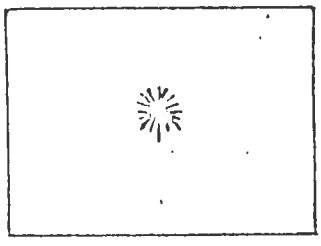


Obrázek se kreslí správně, ale není správně umístěn. To provedeme až nakonec.

4) xxxxx SLUNCE

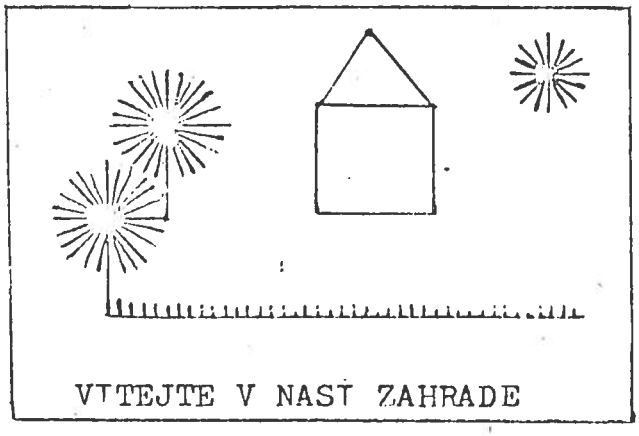
nakonec naprogramuje SLUNCE, které svítí na naši ZAHRADU.

```
XXXXXXX
XXXXXXXXXXXX
?TO SLUNCE
>HT
>REPEAT 26 [FD 15 BK 15 RT 18]
>END
SLUNCE defined
?SLUNCE
```



Nyní budeme přemýšlet, jak spojit tyto procedury dohromady, abychom nakreslili naši ZAHRADU. Musíme být opatrní na konci každé procedury a uvědomit si, v jakém stavu bude želvička, a do jakého stavu se má dostat před následující procedurou. Zkuste nakreslit ZAHRADU, než budete číst dále.

```
?ED "ZAHRADA
TO ZADRADA
DUM
PU LT 15Ø FD 8Ø RT 12Ø PD
STROMY
TRAVNIK
PU FD 1ØØ LT 9Ø FD 15 RT 9Ø PD
SLUNCE
PR [VITEJTE V NASI ZAHRADE]
END
ZAHRADA defined
?ZAHRADA
```




Zkuste nyní nakreslit jednotlivé části v různých barvách!

Kapitola 11
Jednoduchá želví geometrie

Mnohoúhelníky

Když jsme psali proceduru DUM, použili jsme rovnostranný trojúhelník, t.j. trojúhelník, jehož všechny tři strany jsou stejně dlouhé, a jehož tři úhly jsou shodné.

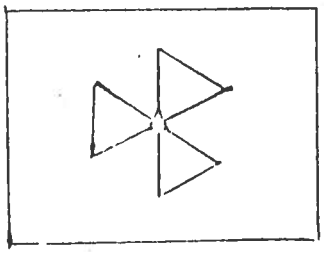
Při kreslení trojúhelníku želvička popojela dopředu a pak se otočila vpravo o $120^{\circ}$ v každém rohu. Můžete se zeptat proč? Je to proto, že když želvička nakreslí obrázek, musí se otočit kolem dokola, t.j. o $360^{\circ}$, a při tom se otáčí třikrát, t.j. 360 / 3 = 120. Podobně při kreslení čtverce se želvička otočí čtyřikrát, pokaždé o 360 / 4 = 90 stupňů. Podobně můžeme pokračovat dále, a nakreslit pětiúhelní, šestiúhelník atd.

```
REPEAT 3 [FD 3Ø RT 12Ø]  --> TROJUHELNIK
REPEAT 4 [FD 3Ø RT 9Ø]   --> CTVEREC
REPEAT 5 [FD 3Ø RT 72]   --> PETIUHELNIK
REPEAT 6 [FD 3Ø RT 6Ø]   --> SESTIUHELNIK
```

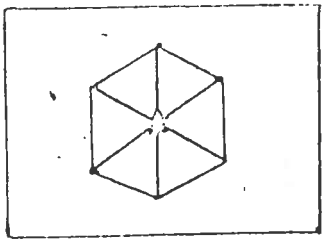
Zkusíme si nyní novou proceduru pro trojúhelník, kterou nazveme TRI, podle předcházejících instrukcí, a budeme si s ní hrát.

```
?TO TRI
>REPEAT 3 [FD 3Ø RT 12Ø]
>END
TRI defined

?REPEAT 3 [TRI RT 12Ø]
```

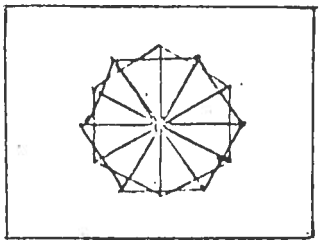


```
?REPEAT 6 [TRI RT 6Ø]
```



Takto můžeme pokračovat dále. Chceme-li trojúhelník nakreslit n-krát, musíme se po každém nakreslení ↑ TRI otočit o 360 / n stupňů. Nebo naopak, otočíme-li se po každém nakreslení TRI o n stupňů, musíme celý postup opakovat XXXXXX 360 / n krát. Např. pro otočení o $30^{\circ}$ musíme opakovat 360/30, tj. 12 krát. Protože ale Logo umí aritmetické výpočty, necháme ho počítat za nás.

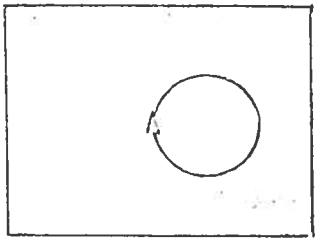
```
?REPEAT 36Ø/3Ø [TRI RT 3Ø]
```



KRUH

Vidíte, že když my kreslíme víceúhelníky, že kresba čím dál více připomíná kružnici? Kdybychom zkoušeli kreslit mnohoúhelníky s čím dál tím větším počtem stran, nakreslili bychom kružnici. Zkusíme si to!

```
?REPEAT 36Ø [FD 1 RT 1]
```



Tato kružnice vypadá pěkně, ale její kreslení trvá dlouho. To proto, že opakujeme kreslení jedné strany 360 krát. Můžeme nakreslit pěkně vypadající kružnici rychleji?

Kapitola 12

Úvod do použití proměnných

Úvod

Nyní jste již seznámeni se zápisem vstupní informace, tj. zvláštní formou zadávání informace pro základní procedury, jako je např. FORWARD nebo RIGHT. Tyto procedury ke své činnosti potřebují ještě další informaci, např. o kolik jednotek postoupit -procedura FORWARD, nebo o kolik stupňů se otočit - procedura RIGHT. Procedury, které napíšete, mohou mít také tuto vstupní informaci. Protože ale tuto informaci můžete případ od případu měnit, budeme ji nazývat proměnná.

Velký čtverec a malý čtverec

Můžeme ~~nnnnxxnnnkxnnsxixxx~~ chtít, aby želvička nakreslila čtverec, jehož strana bude mít délku 50, nebo 60, nebo 80 nebo jen 10 jednotek. Jedna z možností je tak, že pro každou velikost nadefinujeme novou proceduru, např.:
CTVEREC5Ø , CTVEREC6Ø, atd. Ale to by bylo trochu složité. Můžeme napsat takovou proceduru CTVEREC, která bude mít vstupní proměnnou. Potom budete vždy udávat, jak velký čtverec chcete nakreslit. Napíšeme např.:

```
?CTVEREC 5Ø
?CTVEREC 6Ø  atd
```

a želvička nakreslí čtverec příslušně veliký. Nadefinujeme si novou proceduru CTVEREC s proměnnou. Pokud již máme proceduru CTVEREC, můžeme ji editovat. Napíšeme:

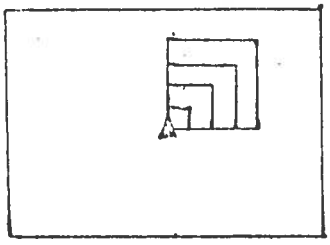
```
?ED "CTVEREC
TO CTVEREC :STRANA
REPEAT 4 [FD :STRANA RT 9Ø]
END
CTVEREC defined
```

Protože chceme, aby se kreslil čtverec s námi zadanou délkou strany, napíšeme za název procedury CTVEREC ještě název proměnné - STRANA -, kterou budeme používat, ale ~~xxx~~ protože je to proměnná, napíšeme před ní znak : , tedy napíšeme :STRANA.

Tuto proměnnou pak dále použijeme tam, kde to budeme potřebovat, tj. v instrukci FD :STRANA . Jako vstupní proměnnou můžeme obecně použít cokoliv, např. číslo, nebo slovo, nebo seznam či seznam seznamů. V našem případě proměnná :STRANA musí být číslo, protože tato proměnná bude dále použitá v příkazu FORWARD :STRANA, a procedura FORWARD potřebuje jako vstupní informaci číslo.

Nyní můžeme použít proceduru CTVEREC a volit si za proměnnou STRANA libovolné číslo podle toho, jak velký čtverec chceme nakreslit. Zkusíme

```
?CS
?CTVEREC 1Ø
?CTVEREC 2Ø
?CTVEREC 3Ø
?CTVEREC 4Ø
```



Jestliže nyní napíšeme pouze

```
?CTVEREC
```

Logo nám napíše chybovou zprávu

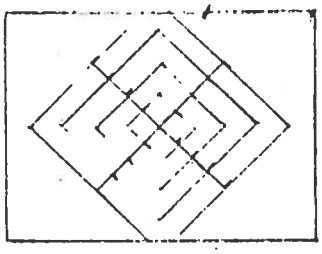
```
Not enough inputs to CTVEREC
```

tj. Není dostatek vstupů pro CTVEREC. Když ale napíšete CTVEREC 1Ø, Logo si vloží číslo 1Ø do proměnné STRANA. Ale pozor: Jakmile procedura CTVEREC ukončí svoji činnost, proměnná STRANA již není dostupná.

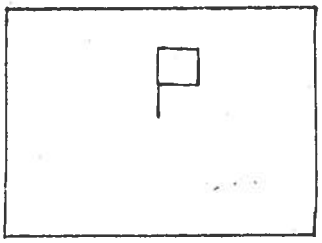
Další procedury, které užívají proceduru CTVEREC :STRANA

```
?TO CTVERCE
>CTVEREC 1Ø
>CTVEREC 2Ø
>CTVEREC 3Ø
>CTVEREC 4Ø
>END
CTVERCE defined
```

```
?TO DIAMANT
>RT 45
>REPEAT 4 [CTVERCE RT 90]
>HT
>END
DIAMANT defined
?DIAMANT
```



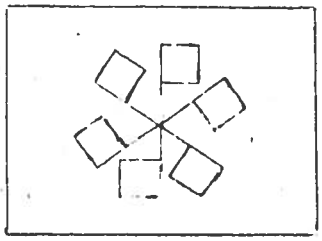
```
?TO PRAP :VEL
>FD :VEL
>CTVEREC :VEL
>BK :VEL
>END
PRAP defined
?PRAP 30
```



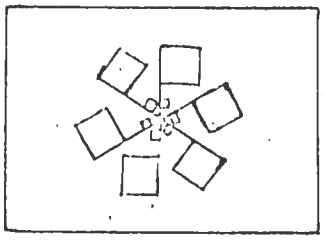
Poznamenáváme, že naše procedura pro kreslení čtverce se jmenuje CTVEREC :STRANA, a my ji nyní vyvoláváme příkazem CTVEREC :VEL, neboť proměnnou, s kterou chceme kreslit velikost čtverce, je nyní proměnná xVR :VEL.

```
?TO 6PRAP :VEL
>REPEAT 6 [PRAP :VEL RT 60]
>END
6PRAP defined
```

```
XXXXXXXØ
?6PRAP 3Ø
```



```
?TO SPINPRAP :VEL
>6PRAP :VEL
>6PRAP :VEL - 2Ø
>END
SPINPRAP defined
?SPINPRAP 3Ø
```



Návody

Zkuste si následující příklady a dívejte se, co se stane.

1) Vynechte mezeru před znakem :.
2) Napište mezeru mezi znakem : a VEL
3) Napište znak : před číslo
4) Uvnitř procedury napište :VL místo :VEL
5) Vynechte znak : a napište pouze VEL
6) Zkuste vložit do procedury další instrukci
7) Náhodou vymažte některou instrukci

Kapitola 13
Čísla a aritmetika

Infix a prefix
Jak uvidíme dále na příkladech, Sinclair Logo může používat aritmetické operace. K počítání používáme následující operace:

| | |
|---|---|
| / | dělení |
| * | násobení |
| - | odečítání |
| + | sčítání |

Tyto znaky se píšou mezi čísla, proto se nazývají infix operace.

Jestliže spojíme několik operací, potom dělení má přednost před násobením, dělení i násobení mají přednost před odečítáním, a dělení, násobení a odečítání má přednost před sčítáním. Zkusíme si to na příkladech:

```
?PR 5 + 3
8
?PR 4 * 23
92
?PR 345 - 32
313
?PR 25/5
5
```

Zkusíme dále:

```
?PR 3 + 4 * 2
11
```

ale

```
?PR (3 + 4) * 2
14
```

Závorky ( ) znamenají, že tato činnost se provádí nejdříve. To znamená, že operace v závorkách se provede první, ostatní operace později. V našem případě se nejprve provede 3+4=7, teprve potom se provede 7 * 2 = 14.

Pro aritmetické operace můžeme také použít procedury nazvané jmény DIV, pro dělení, PRODUCT pro násobení, SUM pro sčítání, a xx potom napíšeme čísla, s kterými tyto procedury budou pracovat. Protože se typ operace píše před čísly, nazývají se tyto operace prefix operace. Poznamenáváme, že zde není procedura pro odečítání, k tomu se používá operace SUM pro sčítání, kde druhé číslo dáme se znaménkem minus. Ukážeme si několik příkladů:

```
?PR SUM 3 4
7
?PR DIV 12 6
2
?PR PRODUCT 4 4
16
```

Čísla v Sinclair Logu

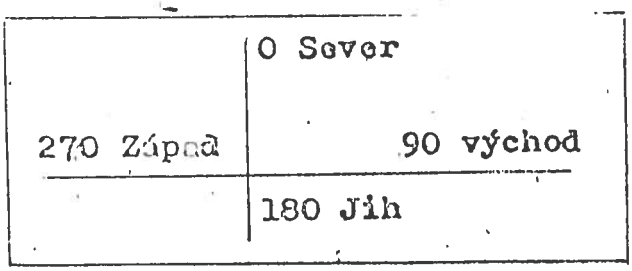
Logo může pracovat s čísly celými i desetinnými.

```
?PR 25/6
4.1666667
?PR 4 * 2.3
9.2
?PR 19 - -2.5
21.5
```

Poznamenáváme, že v posledním příkladu je důležitá mezera před číslem -2.5

Kapitola 14
Pole pro želvičku

Úvod
Želvička se vždy nachází v nějakém stavu, který popíšeme polohou a směrem. Směr je dán podobně jako na kompasu, kdo $0^{\circ}$ znamená směr na sever, $90^{\circ}$ východ, $180^{\circ}$ jih a $270^{\circ}$ západ. Znázorníme si to na obrázku




Jestliže Logo začíná, pak směr želvičky je 0. Po instrukci CS se nastaví směr 0. Můžeme si to vyzkoušet

```
?CS
?RT 90
?PR HEADING
90
```

Příkaz HEADING vydá hodnotu směru, ve kterém se želvička nachází. Protože HEADING je prvním základní procedura, která vydává hodnotu, je to tedy funkce. Tento výstup musíme nějak využít, např. PRINT, nebo jako vstup do další procedury. Jestliže to neuděláme, Logo nám dá chybovou zprávu. Např.:

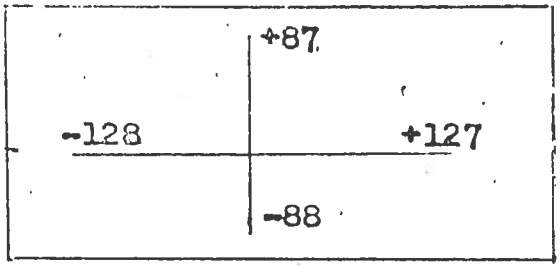
```
?CS RT 90
?HEADING
You don't say what to do with 90
```

tj. systém neví, co má dělat s výstupní hodnotou 90.

Stav želvičky se kromě směru popisuje ještě polohou, což jsou dvě čísla, která ukazují pozici želvičky v souřadnicích x a y. Tyto dvě čísla se zadávají jako seznam. Když Logo začíná, nebo po příkazu CS je tato poloha nastavena na hodnotu [0 0].

První číslo znamená pozici želvičky ve vodorovném směru neboli ve směru osy x od začátku, který je uprostřed obrazovky. Je-li želvička na východ, doprava, číslo je

kladné, když je želvička na západ od středu, nebo doleva, číslo je záporné. Druhé číslo znamená pozici želvičky ve svislém směru neboli ve směru osy y. Je-li želvička na sever nebo nahoru, číslo je kladné, je-li želvička na jih nebo dolů, číslo je záporné. Rozsah je znázorněn na obrázku:




Zde je znázorněn souřadnicový systém, ve kterém se želvička může pohybovat. Je-li želvička uprostřed, obě hodnoty XCOR a XCOR, které vyjadřují polohu želvičky, jsou Ø. Na polohu želvičky se můžeme ptát pomocí funkce POS.

```
?CS LT 9Ø FD 3Ø
?PR POS
```

dostaneme odpověď:

```
-3Ø Ø
```

Napíšeme-li dále

```
?BK 6Ø
?PR POS
```

dostaneme odpověď:

```
3Ø Ø
```

Můžeme se také ptát na jednotlivé souřadnice:

```
R ?PR XCOR
  3Ø
  ?PR YCOR
  Ø
```

což nám udává, že želvička má x-ovou souřadnici 30 a y-ovou souřadnici 0.

Příkaz SETPOS, což je zkratka SET POSition, je příkaz, který nastavuje želvičku do definovaného místa. Na rozdíl od příkazů FORWARD nebo BACK nezáleží na počáteční pozici. Příkaz SETPOS nemění směr. Např.

```
?SETPOS [5Ø -52]
```

nastaví želvičku do pozice, kdy XCOR = 5Ø, YCOR = -52.

Musíme si uvědomit, že parametry, které se udávají do příkazu SETPOS se zadávají jako seznam dvou čísel, tj. v hranatých závorkách. Mezi čísly 50 a -52 musíme udělat mezeru, ale není znaménkem - a hodnotou 52 nesmíme udělat mezeru! Jinak nám systém napíše chybové hlášení

SETPOS doesn't like - as input

Nastavení režimu pro obrazovku: WRAP, FENCE a WINDOW

Jestliže kreslíme na obrazovce, můžeme si pro kreslení nastavit tři režimy práce s obrazovkou. Režim WRAP je takový, že čára kreslená mimo rozměry obrazovky se ukáže na druhé straně a normálně pokračuje, tj. z nakresleného obrázku se nic nestratí. Současně se také upravují hodnoty souřadnic tak, aby byly v rozmezí obrazovky. Zkusíme:

```
?CS
?FD 500
?PR POS
0 -28
```

tj. želvička není ve vzdálenosti 500 od středu, nýbrž pouze -28 dolů.

Jestliže nastavíme režim FENCE, potom pokus o kreslení mimo obrazovku způsobí chybové hlášení a zastavení programu. Zkusíme:

```
?FENCE
?CS
?FD 500
```

Logo napíše hlášení:

Turtle out of bounds

tj. Želvička je mimo rozsah. Režim FENCE zůstane zachován, dokud nenapíšeme WRAP nebo WINDOW.

Režim WINDOW je takový režim, kdy se kreslí na obrazovku pouze to, co je v rozsahu obrazovky. Chceme-li kreslit mimo obrazovku, nic se nekreslí a želvička je mimo obrazovku. Rozsah pohybu želvičky je od -32768 do +32767. Pokud se chceme pohybovat mimo tento rozsah, Logo vydá chybové hlášení. Například:

```
?CS
?FD 5ØØØØ
FD doesn't like 5ØØØØ as input
```

Použití polohy želvičky pro kreslení

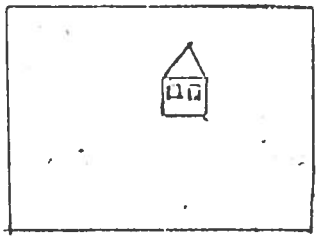
Nyní, když umíme nastavovat polohu želvičky a umíme pracovat s proměnnými, můžeme můžeme vylepšit náš program ZAHRADA.

1) Nejdříve si opravíme proceduru DUM tak, že nakreslíme do domu také okno.

```
?ED "DUM
CTVEREC 45 FD 45 TROJUHELNIK
PU SETPOS [1Ø 26] PD
SETH Ø
CTVEREC 1Ø
PU SETPOS [25 26] PD
CTVEREC 1Ø
END
DUM defined
```

a hned si to vyzkoušíme

```
?DUM
```



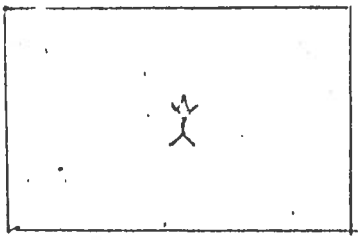
2) Můžeme si také nakreslit malého panáčka, který bude žít v naší zahradě. Nakreslíme

```
?TO V :VEL
>LT 5Ø
>KRESLI :VEL
>RT 1ØØ
>KRESLI :VEL
>LT 5Ø
>END
V defined
```

```
TO KRESLI :VEL
>FD :VEL
>BK :VEL
>END
KRESLI defined
TO OSOBA :VEL
>SETH 180
>V :VEL
>RT 180
>FD :VEL
>V :VEL
>FD :VEL/2
>END
OSOBA defined
```

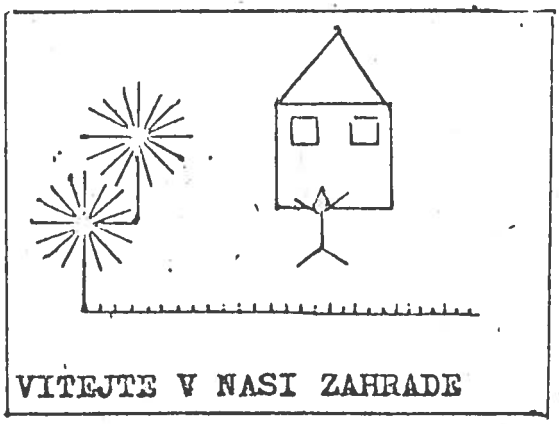
A hned si to vyzkoušíme:

```
?OSOBA 10
```



3) Opravíme náš program ZAHRADA

```
TED "ZAHRADA
TO ZAHRADA
WINDOW
DUM
PU SETPOS [-50 15] SETH 0 PD
SEROMK
PU SETPOS [-90 -45] PD
TRAVNIK
PU SETPOS [90 60] SETH 0 PD
SLUNCE
PU SETPOS [20 0] PD
OSOBA 7
PR "|VITEJTE V NASI ZAHRADE|
END
```




A hned si tento program vyzkoušíme.

Kapitola 15

## Procedura MAKE přiřazuje proměnným hodnotu

Úvod

Chceme-li některé proměnné přiřadit určitou hodnotu, použijeme k tomu proceduru MAKE. Musíme si přitom uvědomit, že číslo je v Logu jako slovo, které se skládá pouze z číslic. V proměnných se ale může uchovávat nejen číslo, ale i slovo nebo dokonce celý seznam. Hodnotu proměnných můžeme dále používat. Ukážeme si to:

```
?MAKE "VEK 8
?PR :VEK
8
?
```

Procedura MAKE dává do proměnné, kterou si označíme slovem "VEK hodnotu, v našem případě číslo 8. Procedura PR tiskne obsah proměnné, proto musíme před název proměnné napsat znak : .

Použití procedury MAKE při kreslení

Procedura MAKE můžeme použít např. při kreslení pravoúhlého trojúhelníka, jestliže známe dvě strany u pravého úhlu. Použitím procedury MAKE si zapamatujeme počáteční polohu, do které se později želvička vrátí.

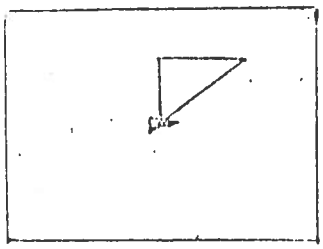
```
?CS
?MAKE "START POS
?PR :START
0 0
```

Do proměnné "START jsme si uložili počáteční polohu želvičky. Nyní můžeme nakreslit trojúhelník:

```
?FD 33
?RT 90
?FD 42
?SETPOS :START
```

Poslední příkaz nutí želvičku, aby se vrátila do polohy :START, tj. do počáteční polohy.

PO těchto příkazech jsme dostali takovýto obrázek:



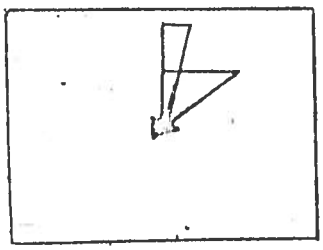
Protože pero bylo dole, želvička při příkazu SETPOS :START se posunula do počáteční polohy a nakreslila při tom čáru. Směr želvičky byl ale takový, jako při posledním příkazu.

Tímto způsobem můžeme definovat novou proceduru pro kreslení trojúhelníku. Nazvěme ji TROJ:

```
?TO TROJ :STRANA1 :STRANA2
 MAKE "START POS
 FD :STRANA1
 RT 90
 FD :STRANA2
 SETPOS :START
 END
TROJ defined
```

a hned si tuto proceduru vyzkoušíme:

```
?CS
?TROJ 40 50
?SETH 0
?TROJ 75 20
```



Kapitola 16

Další kreslení kruhů a oblouků

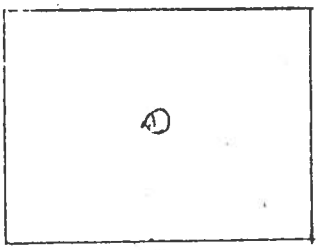
Kružnice

Napíšeme si proceduru pro kreslení kružnice, která bude kreslit různě veliké kruhy:

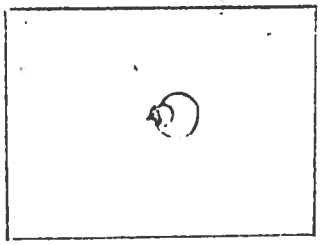
```
?TO KRUH :KROK
>REPEAT 36 [FD :KROK RT 10]
>END
KRUH defined
```

Nyní si tuto proceduru vyzkoušíme pro různé vstupní hodnoty:

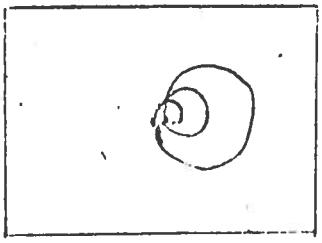
?KRUH 1



?KRUH 5



?KRUH 10



Vidíme, že velikost nakreslené kružnice je v poměru k zadané vstupní hodnotě. Není to náhodou, neboť při kreslení všech kružnic se používá stejného počtu příkazů FD, pouze s jinou vstupní hodnotou. Tato hodnota určuje velikost nakreslené kružnice.

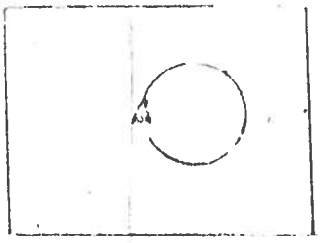
Poloměr kružnice

Jsme zvyklí určovat velikost kružnice pomocí poloměru, tj. vzdáleností středu kružnice od bodu na kružnici. Při volání procedury KRUH bychom museli tento poloměr vypočítat. Ale tohoto výpočet může za nás udělat Logo. V proceduře KRUH jsme zadali parametr :KROK a nakreslená kružnice měla délku 36 * :KROK ,což je vlastně 2 * 3.14 * POLOMER . Zadáme-li POLOMER, můžeme odtud vypočítat velikost KROK. Nadefinujeme si proceduru KRUHPOL, které zadáme jako vstupní hodnotu velikost nakreslené kružnice:

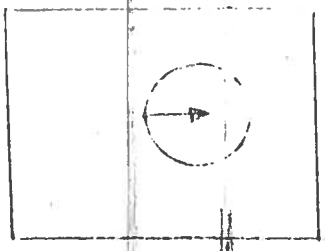
```
?TO KRUHPOL :POLOMER
>KRUH 2 * 3.14 * :POLOMER / 36
>END
KRUHPOL defined
```

Nyní si tuto proceduru vyzkoušíme:

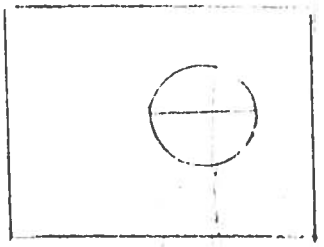
?KRUHPOL 3Ø



?RT 9Ø FD 3Ø

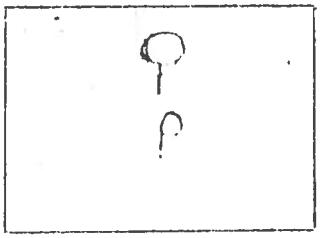


?FD 3Ø HT

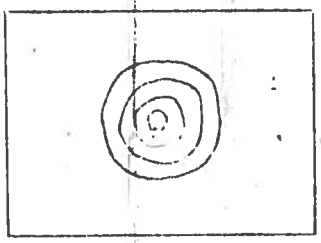


Zde si ukážeme několik obrázků, které jsou nakreslit pomocí kružnic. Zkuste je sami!

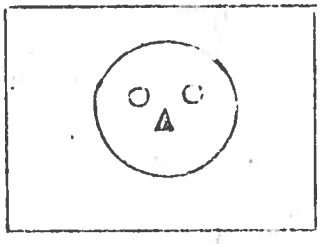
KYTKA



TERC



HLAVA



Oblouky

Oblouky jsou části kružnic. Mnoho projektů používá při kreslení oblouky, a proto si nadefinujeme proceduru pro kreslení oblouků. Při kreslení kružnice pomocí procedury KRUH jsme část kružnice opakovali 36 krát, podobně v procedure OBLOUK budeme opakovat tak tolikrát, jakou část kružnice chceme nakreslit. Definici nové procedury provedeme tak, že opravíme starou proceduru, které změníme i název.

```
TED "KRUH
TO KRUH :KROK
REPEAT 36 [FD :KROK RT 1Ø]
END
```

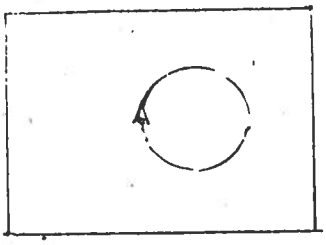
změníme na

```
TO OBLOUK :KROK :POCET
REPEAT :POCET [FD :KROK RT 1Ø]
END
OBLOUK definován
```

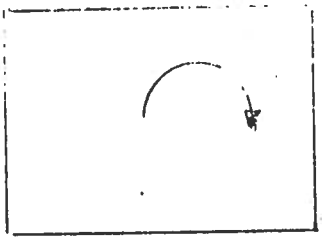
tj. změnili jsme název a přidali jeden parametr, a dále jsme opravili počet opakování v příkazu REPEAT.

Nyní můžeme vyzkoušet:

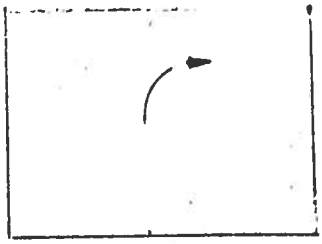
?OBLOUK 1Ø 36



?CS OBLOUK 1Ø 18



?CS OBLOUK 1Ø 9

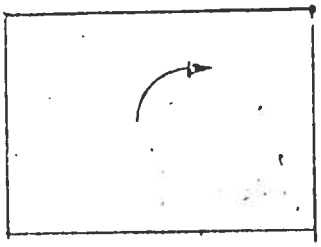


Je tu ale jedna nepříjemnost: Bylo by lepší vyjadřovat délku oblouku ve stupních, nikoliv počtem opakování. Proto provedeme editaci našeho programu:

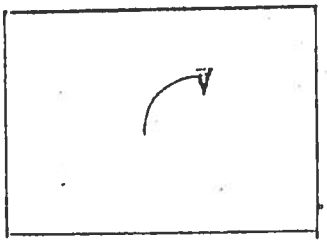
```
?ED "OBLOUK
TO OBLOUK :KROK :STUPNE
REPEAT :STUPNE/1Ø [FD :KROK RT 1Ø]
END
OBLOUK defined
```

A nyní můžeme tuto proceduru vyzkoušet:

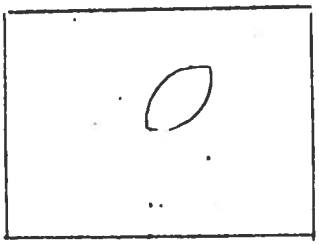
?OBLOUK 6 9Ø



?RT 9Ø



?OBLOUK 6 9Ø HT



Tím jsme nakreslili lístek. Můžeme zkusit nakreslit celou kytičku.

Kapitola 17

## Zkoumání polygonů a spirál

### Polygony

Stejně tak dobře, jako měníme délku, kterou se želvička pohybuje, tak můžeme měnit i její otáčení. Jestliže začneme měnit tyto dvě hodnoty, dostaneme mnoho různých zajímavých kreseb. Zkusíme si to:

```
?TO POLY :KROK :UHEL
>FD :KROK
>RT :UHEL
>POLY :KROK :UHEL
>END
POLY defined
```

A hned si to vyzkoušíme, např.:

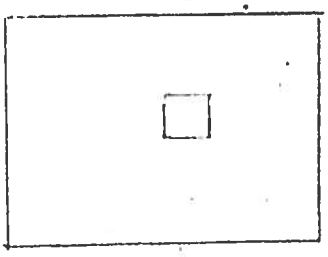
```
?POLY 30 120
```

Želvička bude kreslit a kreslit, zastavit ji můžeme stisknutím tlačítek CAPS a BREAK/SPACE.

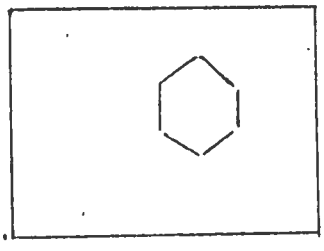
Co se zde stalo? Želvička udělala pohyb vpřed o 30 jednotek, potom se otočila vpravo o 120 stupňů. A nyní procedura volá sama sebe, tj. želvička tento postup stále opakuje, dokud ji nezastavíme. Tento způsob používání se nazývá rekurse.

Tuto proceduru můžeme zkoušet pro různé vstupní hodnoty. Je dobré před každým spuštěním vymazat obrazovku příkazem CS. Zkusíme si pár příkladů:

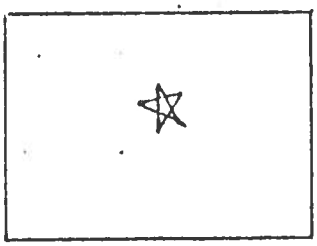
?POLY 30 90



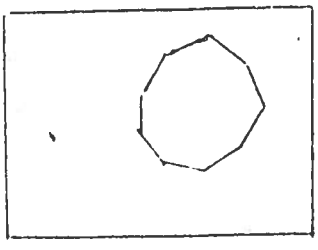
?POLY 30 60



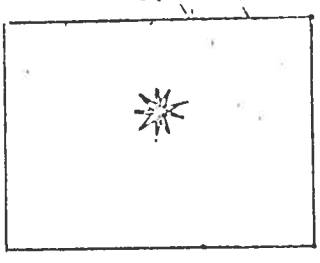
?POLY 30 144



?POLY 30 40



?POLY 30 160



Nyní nadefinujeme polygon, který se bude otáčet a měnit při tom své barvy:

```
?TO POLYT :N :STRANA :UHEL
>POLY1 :N :STRANA
>RT :UHEL
>SETPC PC + 1
>POLYT :N :STRANA :UHEL
>END
POLYT defined

?TO POLY1 :N :STRANA
>REPEAT :N [FD :STRANA RT 360 / :N]
>END
POLY3 defined
?POLYT 6 40 30
```

Poznámka: jestliže barva pera je stejná jako barva pozadí, pak nakreslená čára není vidět.

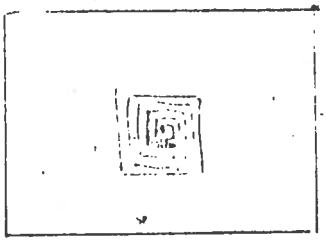
Spirály

Procedura POLY kreslí uzavřené obrázky. Želvička se pohybuje dopředu a otáčí se tak, že po určité době se vrátí na své výchozí místo, a pak se opakuje.

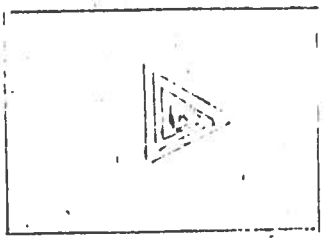
Jestliže chceme kreslit spirály, musí se želvička vrátit na původní místo, ale o kousek dále, po dalším návratu ještě dál atd. Toto popojití způsobíme tak, že k proměnné :KROK přičteme nějakou hodnotu. Protože se volání rekursivně opakuje, bude se opakovat i toto přičítání.

```
?TO SPI :KROK :UHEL
>FD :KROK
>RT :UHEL
>SPI :KROK + 6 :UHEL
>END
SPI defined
?HT SETSCRUNCH [50 50]
```

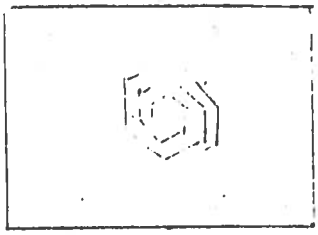
?SPI 5 9Ø



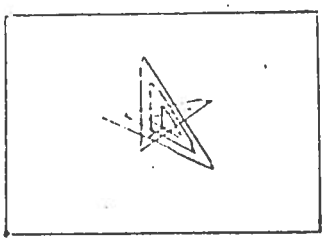
?SPI 5 12Ø



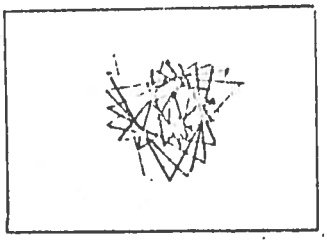
?SPI 5 6Ø



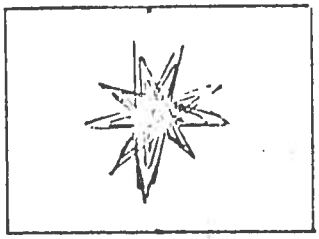
?SPI 5 144



?SPI 5 125



?SPI 5 16Ø



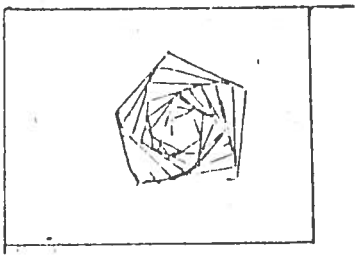
Nyní opravíme procoduru SPI tak, že k ní přidáme třetí vstup, který říká, o kolik se bude zvětšovat krok. To uděláme proto, abychom mohli ~~kmdmchmmk~~ toto zvětšování měnit.

```
TED "SPI
TO SPI :KROK :UHEL :KR
FD :KROK
RT :UHEL
SPI :KROK + :KR :UHEL :KR
END
SPI 'definod
```

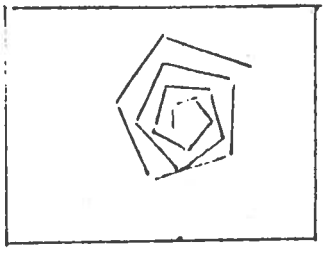
a hned si to vyzkoušíme:

```
?SMTSCHUNCH [1ØØ 1ØØ]
```

?SPI 5 75 1



?SPI 5 75 2



Želvička pořád kreslí a kreslí, zastavíme ji stisknutím tlačítek CAPS a BREAK/SPACE.

Jestliže použijeme příkaz FENCE, želvička se při pohybu mimo obrazovku zastaví. Napíšeme:

```
?CS
?FENCE
?SPI 5 125 2
```

Napíšeme-li příkaz WINDOW, želvička se může pohybovat i mimo obrazovku. Napíšeme:

```
?CS
?WINDOW
?SPI 5 125 2
```

Želvička začne kreslit postupně po celé obrazovce, až se dostane úplně mimo obrazovku. Program zastavíme stisknutím tlačítek CAPS a BREAK/SPACE.

Jestliže použijeme příkaz WRAP, želvička bude kreslit stále po obrazovce, a pokud by někdo vyjela mimo obrazovku, vrátí se z opačné strany. Můžeme si to zkusit:

```
?CS
?WRAP
?SPI 5 125 2
```

Máme pořád jednu proceduru se stejnými parametry, ale pokaždé dostáváme jiný obrazek, podle nastavení režimu obrazovky.

Kapitola 18

## Zkoumání rekurzivních procedur

### Úvod

Jedním z velmi silných rysů Loga je to, že může složité úlohy rozložit na jednodušší procedury, z nichž každá má své jméno a řeší svou část úlohy bez ohledu na práci ostatních procedur. Přitom procedury pro řešení úkolů mohou vyvolávat samy sebe. Tyto procedury nazýváme rekurzivní. Je to např. procedura POLY:

```
?TO POLY :KROK :UHEL
>FD :KROK
>RT :UHEL
>POLY :KROK :UHEL    - toto je rekurzivní volání
>END
POLY defined
```

Procedura POLY volá proceduru POLY jako část své definice

Rekurzivní volání umožní opakovat proceduru vícekrát. Rekurzivní volání může být přímé, tak jako v proceduře POLY, nebo může být nepřímé, např.:

```
?TO JEDNA              ?TO DVE
>FD 1Ø                 >PE BK 1Ø FD
>DVE                   >JEDNA
>END                   >END
```

Procedura JEDNA volá procedura DVE a procedura DVE volá procedura JEDNA ... a tím se dostáváme do uzavřeného kruhu. Takto vyvolaná procedura se dá zastavit pouze stisknutím tlačítek CAPS a BREAK/SPACE.

Ne všechny procedury pracují tímto způsobem, některé se musí zastavit. Tato část procedury, kde dochází k zastavení, neboli k "vyskočení" z kolotoče rekurze, je velmi důležitou částí procedury. Ukážeme si některá pravidla, jak toho docílnout.

Zastavení rekurzivních procedur

Nejdříve se podíváme na některé možnosti zastavení rekurzivních procedur uvnitř programu.

Příklad 1: Zastavení procedury SPI

```
TO SPI :KROK :UHEL :KR
>IF :KROK > 150 [STOP]
>FD :KROK
>RT :UHEL
>SPI :KROK + :KR :UHEL :KR
>END
```

V tomto příkladu jsme naprogramovali zastavení (STOP), jestliže krok je větší než 150. Příkaz IF :KROK > 150 [STOP] se dá vyjádřit takto: Jestliže je hodnota :KROK větší než 150, pak se má program zastavit a vrátit se tam, odkud byl volán, jinak má následovat další příkaz, v tomto případě se procedura opakuje. Vyzkoušíme si tuto opravenou proceduru:

```
SPI 5 125 10
```

Po určité době se činnost procedury zastaví.

Ještě vysvětlení: Procedura IF očekává jako vstupní hodnotu buď TRUE (pravdivý) nebo FALSE (nepravdivý), a podle toho řídí další činnost. Je-li hodnota [illegible] TRUE, pokračuje se příkazem, který následuje na tomto řádku, je-li FALSE, další příkaz na řádku se přeskočí. Operátor > je zvláštní druh operace, která dává výsledek buď TRUE nebo FALSE. Je to predikát, který se používá jako první vstupní hodnota pro IF.

Příklad 2: Tato rekurzivní procedura používá dvě nové základní procedury: FIRST a BUTFIRST.

Procedury FIRST a BUTFIRST rozdělí objekt (tj. slovo nebo seznam), s kterým pracují. Procedura FIRST vybírá první prvek, tj. první písmeno ze slova nebo první prvek ze seznamu, procedura BUTFIRST vybírá právě naopak, tj. slovo bez prvního písmena ze slova, nebo seznam bez prvního prvku ze seznamu. Protože seznam je tvořen ze slov, seznamů nebo slov i seznamů, může být první prvek seznamu buď slovo, nebo seznam.

V Logu je více primitivních procedur, které pracují se slovy nebo seznamy, tj. spojují je, rozdělují, nebo je zkoumají. Tyto procedury jsou uvedeny v seznamu procedur. Nyní si uvedeme příklad, který pracuje s procedurami FIRST a BUTFIRST:

```
?TO SVISLE :SLOVO
 IF :SLOVO = " [STOP]
 PR FIRST :SLOVO
 SVISLE BUTFIRST :SLOVO
 END
SVISLE dofined
?SVISLE "AHOJ
A
H
O
J
?
```

Co se stalo, když jsme zkusili proceduru SVISLE "AHOJ ?

a) Příkaz IF :SLOVO = " [STOP] nutí Logo ukončit proceduru, jestliže hodnota slova SLOVO je prázdná (angl. empty), tj. SLOVO neobsahuje ještě ani jedno písmeno či znak.

b) Jestliže tomu tak není, pak ve slově SLOVO je nějaké písmeno či znak. Potom druhý příkaz, PRINT FIRST :SLOVO vytiskne první písmenný znak slova SLOVO, v našem případě je to písmeno A.

c) Třetí příkaz, SVISLE BUTFIRST :SLOVO je rekurzivní volání, ale v našem prvním případě se jako vstupní informace zasílá obsah slova SLOVO bez prvního znaku, tj. pouze HOJ.

d) Opět se opakuje první příkaz, viz bod a). Obsah slova SLOVO se postupně zmenšuje na HOJ, pak OJ, pak J a při dalším volání je obsah slva SLOVO prázdný. Tím se ukončí činnost procedury SVISLE.

Příklad 3: Rekurzivní otáčení

Jednoduchá rekurze je úplně jednoduchá. Ale některá rekurze může být úplně složitá, ba dokonce i záludná! Chcete-li se

podívat na jednu takovou rafinovanost. Podívejte se následující příklad. Pokud ne, můžete přeskočit na následující kapitolu.

Máme dvě procedury, které vypadají zcela stejně

```
?TO POCITEJ :N
 IF :N = Ø [STOP]
>PRINT :N
>POCITEJ :N - 1
>END
POCITEJ defined
```

```
?TO PREPOCITEJ :N
>IF :N = Ø [STOP]
>PREPOCITEJ :N - 1
>PRINT :N
>END
PREPOCITEJ defined
```

a obě procedury vyzkoušíme:

```
?POCITEJ 3
3
2
1
```

```
?PREPOCITEJ 3
1
2
3
```

a vidíme, že každá dává jiné výsledky.

Procedura POCITEJ :N se zastaví, pokud je :N rovno Ø, jinak tiskne hodnotu N a znovu se opakuje, ovšem s číslem N o jednotku menším. Procedura tedy tiskne hodnoty 3, 2, 1 a pak se zastaví. Tato procedura je průhledná a podobná ostatním procedurám, které jsme uváděli.

Procedura PREPOCITEJ, ačkoliv na to nevypadá, je daleko, daleko složitější.

a) První příkaz zjišťuje, zda hodnota :N je Ø. Jestliže ano, procedura končí. Jestliže ne, pokračuje se dalším příkazem na dalším řádku.

b) Druhý příkaz volá znovu proceduru PREPOCITEJ, ale s hodnotou :N o jednotku menším. Ale co se děje? Tento příkaz je rekurzivní volání, tj. znovu se vracíme na začátek, a vůbec se nedostaneme na další příkaz, ale pozor: dokud je hodnota :N různá od nuly. Jakmile je :N = Ø, a to se jednou stane, pak právě vyvolaná procedura PREPOCITEJ skončí, a vrací se tam, odkud byla vyvolaná, a to je na příkaz PRINT :N.

c) PRINT :N je příkaz, na který se vrací program po volání PREPOCITEJ :N - 1 , kdy hodnota :N - 1 byla rovna Ø, tj. hodnota :N je rovna 1. To se má tisknout, a tudíž program vytiskne 1. A [illegible] přesune se na další příkaz.

d) Poslední příkaz je END, kterým se končí činnost procedury, a [illegible] program se vrací tam, odkud byla procedura vyvolaná. Protože se jedná o rekurzivní volání, program se vrací zdánlivě na stejné místo, na příkaz PRINT :N, ale s jinou hodnotou :N, v tomto případě je to 2. Proto se vytiskne hodnota 2 a znovu se přechází na poslední instrukci, což je END. A znovu se program vrací tam, odkud byl vyvolán, tj. na příkaz PRINT :N, ale [illegible] :N je rovno 3, proto se tiskne 3. A opět se přechází na příkaz END a program se vrací tam, odkud byl vyvolán. A teprve nyní se vrací na volání

TPREPOCITEJ 3

a LOGO zjistí, že za tímto voláním již žádný příkaz nenásleduje, tj. činnost končí a Logo napíše

?

což znamená, že se úspěšně ukončila žádaná činnost a očekává se zadání další činnosti.

Můžeme říci, že při rekurzivním volání uprostřed programu, kdy po volání následují ještě nějaké příkazy, můžeme dostat od procedury více výsledků. V tomto případě naposledy volaná procedura dá výsledky nejdříve, a procedura volaná jako první, dá výsledky až nakonec.

Pokud si uvědomíme, že každá volaná procedura po svém ukončení se vrací na místo vyvolání, a tam [illegible] pokračuje v programu, můžeme snadněji pochopit způsob činnosti. Jiná věc ovšem je, jak si označit to které volání procedury, buď čísly, nebo barvičkami, nebo nějakou záložkou či papírkem. A v tom je asi ten největší problém: počítač si pamatuje, odkud byla která procedura volána, my si to ale někdy popletome.

Kapitola 19

Projekt hra

Návrh hry

Zkusme si sestrojit takovouto hru: Někde na obrazovce se objeví cíl, a hráč má pomocí tlačítek pohybovat želvičku tak, aby co nejmenším počtem kroků dosáhla cíle.

V první verzi budeme řídit želvičku pomocí příkazů, např. LT 45 nebo FD 80. Později použijeme k pohybu želvičky určená tlačítka. S tím, jak se bude vyvíjet návrh hry, se bude upravovat i projekt.

Nejdříve potřebujeme umístit na obrazovku cíl. K tomu si nadefinujeme proceduru UMISTI. Tato procedura umístí náhodně na obrazovku želvičku tak, že bude směrovat nahoru.

```
?TO UMISTI
>PU
>RT RANDOM 360
>FD RANDOM 85
>SETHEADING 0
>PD
>END
UMISTI defined
```

Operace RANDOM, která se v této proceduře používá, dává jako výstup číslo mezi 0 a číslem o jednotku nižším, než je vstupní hodnota, kterou procedura RANDOM potřebuje.

V proceduře UMISTI tedy se nejprve želvička otočí doprava o úhel mezi 0 a 359. Tento úhel se po každém volání znovu počítá a je pokaždé jiný. Dále se želvička posune dopředu o 0 až 84 kroků a nasměruje se na Sever, potom se spustí pisátkem pero. Tuto proceduru budeme používat, když budeme chtít někam nakreslit cíl, a tudíž i želvičku. Je dobré ale nejprve umístit želvičku doprostřed. K tomuto účelu budeme definovat proceduru NASTAVHRU.

```
?TO NASTAVHRU
>CS UMISTI
>CIL
>PR [ZKUS ZASAHNOUT CIL]
```

```
>PU SETPOS [Ø Ø]
>UMISTI
>END
NASTAVHRU defined

?TO CIL
>ČTVER 1Ø
>END
CIL defined

?TO CTVER :STR
>REPEAT 4 [FD :STR RT 9Ø]
>END
CTVER defined
```

Zkusíme si tento program. Napíšeme

```
?NASTAVHRU
?RT 45
?FD 1ØØ
```

a patrně jsme se netrefili. Zkusíme trefit se znovu.

Použití tlačítek k řízení hry

Můžeme napsat mnoho různých interaktivních programů. Můžeme klást otázky a získat odpověď ve slovech nebo ve spojení slov, nebo můžeme pohybovat želvičkou tím, že pouze stiskneme určité tlačítko. Za tímto účelem použijeme operaci READCHAR, zkráceně RC. Např. zkusíme

```
?PR RC     (ENTER)
```

Logo nyní čeká na stisknutí libovolného tlačítka. Stiskneme např. A . Procedura RC přijme tento znak a pak vydá ho jako výstup procedury PRINT. Tato procedura potom vytiskne A

```
?PR RC
A
?
```

Jestliže napíšeme pouze samotné RC, Logo čeká na stisknutí tlačítka, ale potom vydá chybovou zprávu

```
?RC
nyní stiskneme T
You don't say what to do with T
```

RC je funkce, podobně jako HEADING nebo POSITION, tj. vydává hodnotu, která se používá jako vstup pro jiné příkazy. Tento výstup můžeme např. uložit pomocí procedury MAKE, např.:

```
?MAKE "TL RC
```

Logo nyní čeká na stisknutí tlačítka. Stiskneme např. Z, potom proměnná :TL bude obsahovat písmeno Z. Ověříme si to:

```
?PRINT :TL
Z
```

Tuto myšlenku použijeme při vytváření procedury, kterou nazveme HRA:

```
?TO [illegible] HRAJ
>MAKE "ODPOVED RC
>IF :ODPOVED = "F [FD 10]
>IF :ODPOVED = "R [RT 15]
>IF :ODPOVED = "L [LT 15]
>[illegible] HRAJ
>END
[illegible] HRAJ defined
```

Nyní máme definována tato tlačítka:

F pro posun želvičky o 10 kroků dopředu
R pro otočení vpravo o 15°
L pro otočení vlevo o 15°

V proceduře HRAJ se do proměnné ODPOVED ukládá hodnota výstupu procedury RC. Dále se tato hodnota porovnává s předepsanými slovy. Je-li shoda, tj. hodnota porovnání je TRUE, vykoná se příkaz, který následuje na řádku, jinak se přechází na další řádek příkaz.

Tato procedura HRAJ je rekurzivní, neboť se znovu vyvolává. Vyvolává se neustále, proto ji musíme ukončit stisknutím tlačítek CAPS a BREAK/SPACE.

Rozšíření návrhu hry

Jestliže jsme si vyzkoušeli předcházející hru HRAJ, můžeme přistoupit k dalšímu rozšíření návrhu hry. Nadefinujeme si novou proceduru HRA, která využívá procedury NASTAVHRU a proceduru HRAJ.

```
?TO HRA
>NASTAVHRU
>HRAJ
>END
HRA definod
```

Zkus proceduru HRA

Při zkoušení procedury HRA, hlavně někým novým, se neví, co se má dělat. Proto by bylo vhodné napsat pravidla. Opravíme proto proceduru HRA:

```
?ED "HRA
 PRAVIDLA
 NASTAVHRU
 PD
 HRAJ
 END
 HRA definod
```

a dopíšeme proceduru PRAVIDLA:

```
?TO PRAVIDLA
>PR [ZASAHNI CIL ZELVICKOU]
>WAIT 100
>PR [TISKNI R NEBO L PRO OTOCENI]
>PR [A F PRO POHYB VPRED]
>WAIT 100
>END
PRAVIDLA definod
```

Procedura HRA nyní

Nyní HRA vypadá daleko lépe, ale stále je zde co vylepšovat. Hra probíhá pomalu. Hráč může ovládat želvičku a pokud se netrefí, může znovu a znovu, a hra se neukončí. Hráč by měl sám rozhodnout, kdy je dosaženo cíle a hru ukončit. Zkusíme proto navrhnout novou proceduru HRA:

```
?TO HRA
/PRAVIDLA
>NASTAVHRU
>HRAJ
```

```
>WAIT 100          (Logo čeká před novým spuštěním)
>HRA
>END
HRA defined
```

Nyní opravíme proceduru HRAJ tak, že dáme hráči pouze jednu možnost zasáhnutí do cíle. Vtip je v tom, že po natočení musí hráč zvolit vzdálenost, o kterou se želvička posune. Tato možnost se volí písmenem T.

```
?TO HRAJ
>MAKE "ODPOVED RC
>IF :ODPOVED = "R [RT 15]
>IF :ODPOVED = "L [LT 15]
>IF :ODPOVED = "T [ZASAHNI STOP]
>HRAJ
>END
HRAJ defined
```

Nyní ale musíme opravit proceduru PRAVIDLA:

```
?ED "PRAVIDLA
TO PRAVIDLA
PR [ZASAHNI CIL ZELVICKOU]
WAIT 100
PR [TISKNI R NEBO L PRO OTOCENI]
PR [T PRO ZASAH]
WAIT 100
END
```

Tento návrh provádíme metodou shora-dolu (top-down) postupným přibližováním; v proceduře HRAJ voláme proceduru ZASAHNI, kterou teprve musíme definovat:

```
?TO ZASAHNI
>PR [JAK DALEKO JE CIL?]
>ED READWORD
>END
ZASAHNI defined
```

```
?TO READWORD
>OUTPUT FIRST READLIST
>END
READWORD defined
```

Funkce READWORD je podobná jako READCHAR, jenomže v tomto případě můžeme napsat celé slovo místo jednoho znaku. Procedura čeká na stisknutí tlačítka ENTER. Procedura READWORD vybere první slovo, které bylo napsáno, a vydá ho jako výstup. Procedura READLIST je základní procedura, která čeká na zadání z klávesnice ukončené tlačítkem ENTER, a jako výstup vydává seznam.

Nyní máme nadefinovanou celou hru. Zkusíme ji tím, že napíšeme

HRA

potom tlačítky R a L otáčíme želvičkou a tlačítkem T spustíme pokus o zasáhnutí cíle. Dále udáme vzdálenost, a buď se trefíme, či nikoliv.

Zde byl popsán detailní návrh hry, podle vašich představ nyní můžete hru buď vylepšit, nebo napsat úplně jinak.

Hra NIM

Jako poslední projekt si ukážeme program pro hru NIM, kde si také ukážeme možnosti, které nám dává Logo.

Hra NIM má tato pravidla: Na hromádce je určitý počet zápalek, hrají dva hráči, První hráč bere 1, 2 nebo 3 zápalky, potom totéž druhý hráč, atd. Kdo vezme poslední zápalku, vyhraje. Program navrhneme metodou shora dolů. Nejprve navrhneme vlastní hru:

```
?TO HRA.NIM :POCET :HRAC :SOUPER
>PRINT []
>(PRINT [POCET ZAPALEK JE] :POCET )
[illegible]
>(PRINT [NA TAHU JE] :HRAC )
>PRINT [KOLIK ZAPALEK BERES?]
>MAKE "NOVY.POCET :POCET - UDELEJ.TAH :HRAC :POCET
>IF :NOVY.POCET = 0 [(PRINT [VYHRAL] :HRAC ) STOP]
>HRA.NIM :NOVY.POCET :SOUPER :HRAC
>END
[illegible] defined
```

Program HRA.NIM potřebuje proceduru UDELEJ.TAH

```
?TO UDELEJ.TAH :HRAC :POCET
>IF :HRAC = "POCITAC [MAKE "TAH CHYTRY.TAH :POCET]
     [PRINT [MUZES BRAT 1, 2 NEBO 3 ZAPALKY.]
     MAKE "TAH CTI.CISLO]
>IF NOT MEMBERP:TAH [1 2 3] [OUTPUT UDELEJ.TAH
          :HRAC :POCET]
>IF :TAH > :POCET [OUTPUT UDELEJ.TAH :HRAC :POCET]
>(PRINT :HRAC [BERE] :TAH )
>OUTPUT :TAH
>END
UDELEJ.TAH dofined
```

Tato procedura je poněkud složitější. Potřebujo dvě vstupní hodnoty: :HRAC a :POCET . Je-li jako :HRAC slovo "POCITAC, dosadí se do proměnné výstup procedury CHYTRY.TAH - tj. tah počítače (bude definováno dále), jinak se do proměnné "TAH dá číslo, které zadá hráč. Nyní se testuje, je-li :TAH číslo 1, 2 nebo 3 - to dělá procedura MEMBERP. Není-li tomu tak, procedura UDELEJ.TAH se opakuje (rekurzivně). Dále se testuje, zda :TAH je menší než :POCET. Pokud je vše v pořádku, vytiskne se, který hráč bere kolik zápalek a procedura končí. Ještě poznámka: Příkazy IF se musí psát na jeden řádek.

```
?TO CTI.CISLO
>OUTPUT FIRST READLIST
>END
CTI.CISLO dofinod
```

a nakonec nadefinujeme proceduru pro CHYTRY.TAH

```
?TO CHYTRY.TAH :POCET
>MAKE "TAH REMAINDER :POCET 4
>IF :TAH = 0 [MAKE "TAH 1 + RANDOM 3]
>OUTPUT EK :TAH
>END
CHYTRY.TAH defined
```

a program vyzkoušíme např.:

```
?HRA.NIM 15 "JIRKA "POCITAC
```

a dále postupujeme podle pokynů. Podobně spustíme novou hru. Nohou proti soubě hrát dva hráči, např. JIRKA a PAVEL, nebo i počítač proti počítači.

O B S A H

Logo - stručný návod

Nahrajeme program LOGO - LOAD "" Enter

V Logu nastavíme velká písmena - stiskneme CAPS + 2,
aby bylo v pravém dolním rohu písmeno velké C

napíšeme písmeny LOAD "LOGO1 Enter

pustíme magnetofon - PLAY a čekáme, až se Logo najde program označený LOGO1

Po nahrání zastavíme magnetofon.

V Logu napíšeme: REM1 Enter a čteme, pak
REM2 Enter
REM3 Enter
REM4 Enter

potom vše smažeme: ERALL
nahrajeme další část: LOAD "LOGO2 Enter
pustíme magnetofon.
Pak vše smažeme a nahrajeme LOGO3 a nakonec LOGO8

Můžeme také nahrát program KLAUN a spustit ho příkazem KLAUN.
Nebo nahrajeme program TRIROZ a spustíme příkazem PRIKLAD.
Nebo program ZXUSER, spustíme příkazem DILO.
Mnoho zábavy s Logem přeje

Ing. Julius Timar tel. zam. 25111/275
Hronovická 1034 tel. domů 22376
530 00 Pardubice

Kdykoliv můžete žádat o radu, 24 hod denně, 7 dní v týdnu.