COMMODORE 64

uživatelská příručka

Obsah

0. Úvod

Váš nový počítač COMMODORE 64 je dnes nejlepším domácím počítačem, který můžete využívat pro seriózní práci i pro zábavu. Mikropočítač COMMODORE 64 má dostatečně velkou paměť (64KB), barevný obrazový i akustický výstup.

Tento snadno pochopitelný návod k použití vám podá všechny informace, které budete potřebovat k obsluze vašeho počítače. Po prostudování tohoto návodu budete umět vytvořit i vlastní jednoduché programy v jazyku Basic. Návod samozřejmě nemůže obsáhnout celou problematiku počítačů a programování, ale může vám pomoci při výběru literatury, která se těmito problémy zaobírá podrobněji.

Ti z vás, kteří chtějí pouze využívat již hotové programy a nechtějí tvořit vlastní, nemusí ani dočíst návod do konce, protože pro obsluhu počítače stačí pochopit pokyny prvních kapitol.

Co všechno váš COMMODORE 64 umí?

COMMODORE 64 používá tzv. SPRITE grafiku, která umožňuje vytvoření až osmi tříbarevných obrázků (spritů), s kterými lze plynule pohybovat po celé obrazovce, lze zdvojnásobovat jejich šířku i výšku, lze určit, který z nich bude na obrazovce překrývat druhý a lze i automaticky indikovat jejich kolizi.

COMMODORE 64 může svými schopnostmi směle konkurovat i hudebním syntetizátorům. Má zabudovány tři nezávislé tónové generátory s rozsahem plných devět oktáv. Dále je možno naprogramovat čtyři různé tvary vln, obalovou křivku tónu, pásmové filtry, rezonanci a samozřejmě hlasitost. Pokud chcete dokonale využít těchto schopností, můžete svůj počítač připojit ke kterékoliv HIFI soupravě.

Chcete-li svého počítače COMMODORE 64 plně využívat, můžete jej doplnit o další zařízeni. Pro záznam dat lze využít buď kazetový magnetofon nebo disketovou jednotku, kterých může COMMMODORE 64 obsluhovat najednou až pět. Dále lze k vašemu počítači připojit jehličkovou tiskárnu, souřadnicový zapisovač (plotter), telefonní modem, barevný monitor a pod.

Přejeme vám mnoho úspěchů při programovaní. Doufáme, že i tento návod přispěje k rozšíření vašich znalostí.

1. Uvedení počítače do provozu

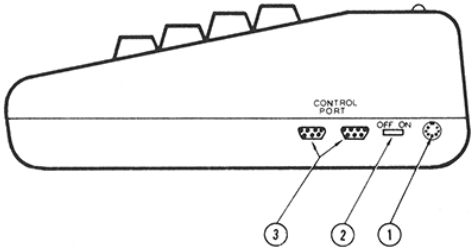
1.1. Vybalení počítače

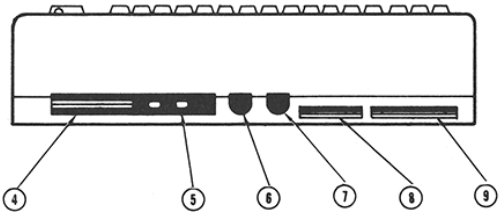
Dříve než se dostaneme k tomu jak počítač uvést do provozu, zkontrolujme obsah balení. Krabice by měla obsahovat:

- návod k obsluze
- počítač COMMODORE 64
- zdroj s propojovacími kabely
- kabel pro připojení k televizoru

Neobsahuje-li balení všechny tyto součásti, výrobek reklamujte.

Co a jak lze k počítači připojit





Konektory na postranním panelu:

1 - konektor napájení, sem se připojí konektor zdroje
2 - hlavní vypínač počítače
3 - konektory pro připojení joysticků, světelného pera apod.

Konektory na zadním panelu:

4 - konektor pro zásuvné moduly
5 - přepínač pro volbu televizního kanálu a konektor TV pro propojení počítače s televizorem

6 - audio a video výstup pro připojení počítače k monitoru nebo k HIFI zařízení
7 - sériový výstup pro připojení tiskárny a disketové jednotky
8 - konektor pro připojení magnetofonu
9 - programovatelný uživatelský konektor pro připojení zásuvných modulů, modemů apod.

1.2. Instalace

Připojení televizoru

1. Jedním koncem připojte přiložený koaxiální kabel do konektoru č.5 na zadní stěně počítače.
2. Druhý konec kabelu připojte k anténnímu vstupu vašeho televizoru.
3. Nyní zapněte počítač vypínačem na pravé straně a na televizoru nalaďte 36. kanál.

Připojení monitoru

Lepšího obrazu docílíte připojením vašeho počítače k barevnému monitoru. K tomuto propojení slouží konektor č.6 na zadní stěně počítače. Má-li váš televizor video konektor, můžete jej použít jako monitor (POZOR!!! Do počítače se nesmí dostat napětí 12V, které přepíná televizor do režimu video).

Počítač lze připojit pomocí konektoru k jakémukoliv HIFI zařízení.

Popis jednotlivých pinů audio/video konektoru naleznete v příloze B.

Popis připojení tiskárny, disketové jednotky a dalších periférií naleznete v návodech, které jsou součástí těchto zařízení.

1.3. Uvedení do provozu

1. Zapněte počítač vypínačem na pravé straně
2. Po několika sekundách se na obrazovce objeví hlavička a blikající kurzor (viz obr.).
3. Dolaďte televizor tak, aby měl optimálně ostrý obraz.
4. Nastavení barev a jasu je popsáno v další kapitole. Nyní by obrazovka měla být tmavě modrá se světle modrým okrajem a světle modrými znaky.

Pokud se počítač nehlásí výše uvedeným způsobem, přezkoušejte ještě jednou zapojení podle následujícího návodu:

příznak	příčina	odstranění
kontrolka "power" nesvítí	počítač není zapnut	zapněte vypínač na pravé straně do polohy ON
	není zapojena síť	zkontrolujte napětí v zásuvce; zkontrolujte připojení do zásuvky
	porucha pojistky	vyměňte pojistku zdroje
na obrazovce není obraz	není zvolen správný kanál	zkontrolujte správné nastavení kanálového voliče
	závada v propojení s televizorem	zkontrolujte kabel a jeho zapojení
zmatené znaky při zapojeném modulu	modul je nesprávně zasunut	vypněte počítač a zkontrolujte správné zasunuti modulu
silný šum	je nastavena příliš velká hlasitost	
nefunguje zvuk	je nastavena příliš malá hlasitost	

	audiovýstup není zapojen do zesilovače	zkontrolujte zapojení

Pozn.:
Kurzor (cursor) je blikající čtverec pod hlášením READY. Ukazuje polohu, na které se objeví znak zadaný z klávesnice.

1.4. Nastavení barev

Pro optimální nastavení barev si nejprve vytvoříme na obrazovce obrázek. Použijeme k tomu klávesu "CONTROL", která musí být stisknuta současně s jinou klávesou (0-9).

Stiskněte současně klávesu CONTROL a klávesu 9. Tímto jste přepnuli počítač do režimu inverzního zobrazení. To znamená, že se znaky zobrazí v barvě podkladu a podklad bude mít barvu původně určenou pro písmo.
Nyní stiskněte mezerník a na obrazovce se začne objevovat světlý pruh.
Nyní změníme barvu písma na žlutou. Stiskněte současně klávesu CONTROL a číslici 8. Po stisknutí mezerníku se začne objevovat žlutý pruh.
Tímto způsobem (použitím klávesy CONTROL a číslice) si vytvořte na obrazovce několik různých pruhů.
Podle těchto pruhů nastavte barvy svého televizoru tak, abyste získali co nejkvalitnější obraz.

Nyní je váš systém správně nastaven.

V dalších, kapitolách se seznámíte s programovacím jazykem BASIC. Rozhodnete-li se však využívat již hotové programy, nebudete tyto znalosti potřebovat. Každý hotový program je totiž doplněn podrobným návodem k použití.
Přesto vám doporučujeme prostudovat alespoň několik dalších kapitol, abyste se seznámili se základními vlastnostmi systému vašeho nového počítače.

2. Začátky s počítačem

2.1. Klávesnice

Klávesnice je nejdůležitějším dorozumívacím prostředkem mezi vámi a vaším počítačem. Proto se s ní musíme dokonale seznámit. Klávesnice počítače Je velmi podobná klávesnici psacího stroje. Má však několik speciálních tlačítek, jejichž funkci si musíme nejprve objasnit.

RETURN

Klávesou RETURN se informace, kterou jste pomocí klávesnice napsali a která se objevila na obrazovce, uloží do paměti počítače.

SHIFT

Klávesa SHIFT má podobnou funkci jako přepínač velkých a malých písmen u psacího stroje. Se stisknutou klávesou SHIFT se na obrazovku píší velká písmena nebo se zapíná funkce napsaná v horní polovině klávesy. V režimu velká písmena/grafika se pak zobrazují znaky pravé poloviny klávesy.

Klávesy určené pro opravy a editaci

CRSR

Co to je kurzor jsme si již vysvětlili. Dvojicí tlačítek označených CRSR (od slova cursor) můžeme tímto posouvat po celé obrazovce. Levou klávesou posouváme kurzor dolů (stiskneme-li současně SHIFT tak nahoru) a pravou klávesou posouváme kurzorem vpravo (resp. vlevo).

INST/DEL

Označení těchto kláves Je odvozeno od slov DELete (smazat) a INSert (vkládat). Klávesou DEL můžeme smazat znak před (vlevo) kurzorem, zbytek řádky se pak posune o jeden znak vlevo. Stiskneme-li současně s klávesou INST/DEL klávesu SHIFT, zapneme funkci vkládání. Všechny znaky v řádce, kde se nachází kurzor, se posunou vpravo a místo pod kurzorem se uvolní pro vkládaný znak.

CLR/HOME

Stlačením klávesy HOME přesuneme kurzor do levého horního rohu obrazovky. Stiskneme-li současně s touto klávesou klávesu SHIFT, smaže se navíc celá obrazovka.

RESTORE

Tato klávesa se používá současně s klávesou RUN/STOP a navrací počítač zpět do výchozího stavu. Jaké to má praktické využití, si povíme později.

Funkční klávesy

Funkčním klávesám lze v programu přiřadit libovolný význam. Podrobně se o tom zmíníme v 5. kapitole.

CTRL

Klávesa CTRL (control=řízení) se používá ke změně barvy písma a k zapnutí inversního zobrazování. Používá se současně s klávesami 1 až 8 (pro změnu barvy) a s klávesami 9 a 0 (zapnutí/vypnutí inversního zobrazení).

RUN/STOP

Klávesa RUN/STOP slouží k zastavení běžícího basicového programu. Stiskneme-li klávesu RUN/STOP společně s klávesou SHIFT, automaticky se do počítače nahraje program z magnetofonu a odstartuje se.

C= (Commodore)

Klávesa se symbolem COMMODORE(C=) má několik funkcí. Stiskneme-li klávesu C= spolu s jinou klávesou, zobrazí se znak uvedený v levé polovině klávesy. Stiskneme-li klávesu C= spolu s klávesou SHIFT, přepneme tím sadu používaných znaků. Můžeme tedy používat buď malá písmena, velká písmena a znaky na levé polovině kláves nebo velká písmena, znaky uvedené na pravé polovině kláves (spolu s klávesou SHIFT) a znaky uvedené na levé polovině kláves (spolu s klávesou C=).

Stiskneme-li klávesu C= společné s klávesami 1 až 8, změníme tak barvu písma podobně jako při použití klávesy CTRL. Barvy se zapínají podle následujícího klíče:

C= 1 oranžová	C= 5 šedá 2
C= 2 hnědá	C= 6 světle zelená
C= 3 světle červená	C= 7 světle modrá
C= 4 šedá 1	C= 8 šedá 3

2.2. Nahrávání programů - LOAD

Programy pro Váš počítač COMMODORE 64 mohou být uloženy buď na disketách nebo na kazetách nebo v paměti zásuvných modulů. Vaše programy pak můžete ukládat buď na diskety nebo na kazety, podle toho jaké záznamové zařízení vlastníte.

Nezapomeňte vždy před zapnutím počítače zkontrolovat připojení záznamového zařízení.

Nahrávaní programu ze zásuvného modulu

Na zásuvných modulech bývají uloženy jak uživatelské programy, tak různé hry.

Postup:

1. Vypněte počítač!!!
   Jestliže zasouváte do počítače modul, musí být počítač vypnutý. V opačném případě hrozí poškození modulu i počítače.
2. Zasuňte modul do konektoru pro zásuvné moduly.
3. Zapněte počítač
4. Odstartujte program dle přiloženého návodu.

Nahrávání programu z kazety

a) Nahrávání zakoupených programů:

Na zakoupených kazetách bývá zpravidla uložen jeden program. Proto při nahrávání do počítače postupujeme následujícím způsobem:

1. Vložte kazetu do magnetofonu.
2. Stiskněte tlačítko REW a přetočte kazetu na začátek první strany.
3. Napište příkaz LOAD a stiskněte klávesu RETURN. Na obrazovce se objeví nápis PRESS PLAY ON TAPE.
4. Stiskněte tlačítko PLAY na magnetofonu. Nyní počítač hledá hlavičku programu. Po jejím nalezení vypíše FOUND a jméno programu, který nalezl.
5. Stiskněte klávesu C= a program se začne nehrávat do počítače. Nahrávaní můžete přerušit stisknutím klávesy RUN/STOP.
6. Takto nahraný program se zpravidla sám automaticky spustí. Pokud ne, spustíme jej příkazem RUN.

b) Nahrávání vlastních programů:

Program, který jste na kazetu uložili, můžete do počítače nahrát následujícím způsobem:

1. Vložíme kazetu do magnetofonu a přetočíme ji na začátek nebo alespoň před program, který chceme nahrát.
2. Napíšeme na klávesnici příkaz LOAD"jméno programu". Pokud neznáme přesný název, můžeme napsat pouze LOAD a počítač nahraje první program, který nalezne.
3. Stiskneme klávesu RETURN a na obrazovce se objeví nápis PRESS PLAY TO TAPE.
4. Stiskneme tlačítko PLAY na magnetofonu. Počítač začne hledat žádaný program a po jeho nalezeni se ohlásí FOUND jméno programu.
5. Stiskneme klávesu C= a počítač program nahraje. Po nahrání programu se na obrazovce objeví nápis
   READY
   a blikající kurzor. Program spustíme příkazem RUN.

Pozn: Během vyhledávání a nahrávání programu je obrazovka prázdná.

Jestliže do počítače nahrajeme nový program, je program původně uložený v počítači smazán.

Nahrávání programu z diskety

Nahrávání programů z diskety je stejně jednoduché jako nahrávání z kazety. Vyhledávání programů a jejich nahrávání z diskety je však podstatně rychlejší.

Postupujeme následovně:

1. Otevřete dvířka disketové jednotky a disketu zasuňte tak, aby byla její etiketa otočena směrem nahoru a směrem k vám. Všimněte si zářezu na disketě (může být i přelepen páskou). Pokud disketu zasouváte správně, musí být tento zářez na levé straně.
2. Nyní opatrně zavřete dvířka disketové jednotky.
3. Napište příkaz
   LOAD"jméno programu",8
   Chcete-li nahrát první program, který je uložen na disketě, můžete použít příkazy
   LOAD"*",8
   Číslice 8 udává adresu zařízení, ze kterého má být program nahrán, tedy disketovou jednotku.
4. Stiskněte klávesu RETURN. Ha obrazovce se objeví nápis
   SEARCHING FOR jméno programu
   LOADING

READY
a blikající kurzor.

2.3. Jak naformátovat novou disketu

Chceme-li použít pro záznam programu úplně novou nenaformátovanou disketu, musíme ji nejprve naformátovat. To znamená opatřit disketu hlavičkou a definovat místo pro jednotlivé sektory, do kterých se pak budou zapisovat data. Formátování provádíme verzí basicového příkazu OPEN:

```
OPEN 1,8,15,"N0:jméno,id"
```

N0 - tato zkratka říká, že se má vytvořit nová hlavička (N = new) na disketě v disketové jednotce 0
jméno - jméno diskety, které bude figurovat při výpisu adresáře diskety, může obsahovat maximálně 16 znaků
id - identifikační kód diskety složený ze dvou libovolných znaků

Zhasne-li kontrolka "drive" na disketové jednotce, můžeme ukončit formátovaní příkazem:
CLOSE 1
a stisknutím klávesy RETURN.

POZOR!!!
Formátováním se zničí veškeré informace uložené na disketě, proto se vždy ujistěte, že disketa je prázdna.

Pozn:
Podrobněji je o příkazu OPEN pojednáno v příloze originálního manuálu.

2.4. Uschování programů - SAVE

Uschování programů na kazetu

Chcete-li uložit program, který máte v paměti počítače na kazetu, použijte následující postup:

1. Napište příkaz
   SAVE"jméno programu"
   (jméno programu může obsahovat až 18 libovolných znaků)
2. Stiskněte klávesu RETURN a na obrazovce se objeví nápis
   PRESS PLAY AND RECORD ON TAPE
3. Stiskněte současně tlačítka REC a PLAY na magnetofonu. Obrazovka se vyprázdní a programu se uschová. Po uschování programu je počítač připraven k další spolupráci.

Uschování programů na disketu

Chcete-li uschovat program na disketu, založte do disketové jednotky disketu (podle stejných pravidel jako při nahrávání), a postupujte následujícím způsobem:

1. Napište příkaz
   SAVE"jméno programu",8
   (číslice 8 označuje zařízení, na které má být program uschován)
2. Stiskněte klávesu RETURN a na obrazovce se objeví nápis
   SAVING"jméno programu"
   OK
   READY
   a blikající kurzor.

Program je uložen na disketě.

2.5. Vypsání seznamu programů na disketě - DIRECTORY

Chceme-li zjistit jaké programy máme uloženy na disketě, můžeme si nechat vypsat jejich seznam tzv. directory.

Postupujeme následovně:

1. Napíšeme příkaz
   LOAD"$",8
   na obrazovce se objeví nápisy
   SEARCHING FOR $
   LOADING
   READY
   a blikající kurzor.
   Počítač vyhledal a nahrál seznam programů uložených na disketě.

2. Nahraný seznam se vypíše na obrazovku příkazem LIST
   a stisknutím klávesy RETURN.

3. Začátky v Basicu

3.1. Psaní a počítání

V této kapitole začneme pronikat do tajů programovacího jazyka BASIC.

Nejdůležitějším a nejpoužívanějším příkazem je příkaz PRINT. Pomocí tohoto příkazu můžeme napsat cokoliv na obrazovku.

Postupujte podle následujícího návodu:

1. Napište příkaz
   PRINT "COMMODORE 64"
2. Stiskněte klávesu RETURN. Počítač provede příkaz PRINT a vypíše na obrazovku znaky uvedené v uvozovkách.
3. Na obrazovce se tady objeví
   COMMODORE 64

Pokud při psaní uděláte chybu, použijte k opravě klávesu INST/DEL.

Jestliže počítač místo požadovaného textu vypíše:

? SYNTAX ERROR

pak jste se dopustili někde omylu. Buď není správně napsán příkaz nebo chybí uvozovky. Počítač pracuje velmi přesně a zpracovává pouze příkazy, které přesně odpovídají předepsanému schématu.

Použití příkazu PRINT pro výpočty

Jak je důležité psát uvozovky všude tam, kde mají být si ukážeme na následujícím příkladu.

Napište příkaz:

```
PRINT 12 + 12
```

stiskněte klávesu RETURN a na obrazovce se objeví číslo 24. tedy výsledek zadaného výrazu.

Zkuste ještě zadat příkaz:

```
PRINT "12 + 12"
```

na obrazovce se objeví

```
12 + 12
```

Přesvědčili jsme se tady, počítač při příkazu PRINT opisuje znaky uvedené v uvozovkách a píše skutečnou hodnotu výrazů uvedených bez uvozovek.

Podobně jako jsme v tomto případě použili matematickou funkci sčítání, lze použit i ostatní matematické funkce uvedené v další kapitole.

Je třeba si ale uvědomit, že chceme-li zobrazit výsledek na obrazovce, je nutno před matematický výraz napsat příkaz PRINT. Příkaz je pak vykonán po stisknutí klávesy RETURN.

3.2. Matematické funkce

FUNKCE	SYMBOL	příklad
sčítání	+	12 + 12
odečítání	-	12 - 1
násobení	*	12 * 12
dělení	/	144 / 12
umocňování	↑	12 ↑ 5

Pozn:
Existuje celá řada zkratek pro příkazy jazyka Basic, najdete Je v příloze originální příručky. Zkratka pro příkaz PRINT Je otazník <?>.
Příkaz:

```
PRINT 3*5-7+2
```

lze tedy napsat takto

```
? 3*5-7+2
```

3.3. Vícenásobné operace na jedné řádce

Zatím jsme jako příklady uváděli velice jednoduché výpočty. Váš počítač COMMODORE 64 je však schopen vyřešit i daleko složitější výrazy. I když je váš počítač velmi přesný a dokáže počítat s velmi vysokými čísly, přece jen nějaké omezení má.

COMMODORE 64 provádí všechny výpočty s přesností na 10 míst, z nichž zobrazuje pouze 9.

Zobrazení čísel v rozsahu od 0.01 do 999 999 999 je ve standardním tvaru, čísla větší a menší se zobrazují ve vědecké notaci (tj. pomocí mocnin čísla 10).

Příklad: zadáme-li příkaz

```
PRINT 123000000000000000
```

zobrazí počítač

```
1.23E+17
```

tedy 1.23 * 10 ↑ 17

I při zápisu čísel ve vědecké notaci existují hranice, mezi nimiž musí ležet čísla, mají-li být zpracována počítačem.
Horní hranice je ± 1.70141183E +38
Dolní hranice je ± 2.93873588E -39

3.4. Kombinace různých matematických operací

Jestliže počítači uložíme vyřešit matematický výraz, který obsahuje různé druhy matematických operací, musíme, chceme-li dostat správný výsledek, znát postup, jakým počítač řeší složité operace. Proto si pozorně prohlédněte následující tabulku, ve které je uvedena priorita matematických operací. Počítač tedy řeší operace v následujícím pořadí:

1. znaménko - před číslem (nikoliv odečítání)
2. znaménko ↑ pro umocňování
3. znaménko / a * pro dělení a násobení
4. znaménko + a - pro sčítání a odečítání

Chceme-li, aby počítač řešil nejprve operace s nižší prioritou, musíme tyto uzavřít do závorek.

Příklad:

```
?(30+15)*(2-3)
```

Závorek muže být použito libovolné množství a mohou být i libovolně vkládány.

Příklad:

```
?30+(15*(2-3))
```

V tomto případě řeší počítač nejprve operace uvnitř závorek a to tak, že jako první řeší vloženou závorku (tedy operaci 2-3).

3.5. Kombinovaný výpis

Použitím správné kombinace uvozovek u příkazu PRINT můžeme dosáhnout například toho, aby byl na obrazovce znázorněn jak zadaný příklad, tak i jeho výsledek.

Příklad:

```
?"5 * 9";5 * 9
```

Na tento příkaz bude počítač reagovat následovně. První část ("5 * 9") vytiskne na obrazovku jako řetězec znaků a druhou část za středníkem vypočte a vytiskne výsledek.

Stejně jako středníku je možno jako oddělovací znaménko použít čárku. Rozdíl uvidíte na první pohled. Zatím co při použití středníku se výsledek vytiskne hned za zadáním, ve druhém případě vznikla mezi zadáním a výsledkem mezera. Důvodem je to, že počítač COMMODORE 64 používá čárku tabulátoru.

Obrazovka vašeho počítače obsáhne 40 znaků na jednom řádku. Každý řádek je rozdělen na čtyři díly po deseti znacích. Použijeme-li jako oddělovací znaménko čárku, napíše se výraz uvedený za čárkou na první pozici následujícího dílu obrazovky (tzn. do sloupce 1, 11, 21 nebo 31). Této skutečnosti lze při různých výpisech využít.

4. Psaní jednoduchých programů v BASICu

4.1. Číslování řádků

Až doposud jsme se spokojili s tím, že jsme počítači zadávali jednotlivé příkazy a on je plnil okamžitě po stisknutí klávesy RETURN. Tomuto způsobu provádění příkazů se říká práce v přímém módu (direct mod).

Počítač však umí více. Je schopen provádět mnoho po sobě jdoucích příkazů, které mu zadáme. Tyto příkazy, které zadáváme pomocí očíslovaných řádků, nazýváme programem.

Nejprve si vyzkoušíme velice jednoduchý program. Smažte obrazovku stisknutím kláves SHIFT a CLR/HOME. Nyní napište příkaz NEW a stiskněte klávesu RETURN. Tím se vymaže z paměti vše, co jste tam během vašich předcházejících pokusů zapsali.

Nyní opište následující řádky a nezapomeňte každý z nich ukončit klávesou RETURN.

```
10 PRINT "COMMODORE 64"
20 GOTO 10
```

Program je tady napsán a můžeme je] spustit. Napište tedy příkaz RUN a stiskněte klávesu RETURN. Na obrazovce se začne donekonečna vypisovat nápis COMMODORE 64. Program můžete ukončit stisknutím klávesy RUN/STOP.

Na tomto krátkém příkladu jsme si ukázali několik věcí, které jsou pro programování velice důležité. Podívejme se na ně podrobněji.

Všimněte si, že každá řádka, ve které je napsán příkaz, začíná číslem. Podle těchto čísel postupuje počítač při řešení programu. Jako čísla řádků mohou být použita všechna celá čísla v rozsahu od 0 do 63999. Je dobrým zvykem číslovat řádky programu s odstupem 10. To proto, aby bylo možno dodatečně do programu vkládat řádku.

4.2. Příkaz GOTO

Jak jste si jistě všimli, vyskytuje se v našem programu kromě již známého příkazu PRINT ještě příkaz GOTO. Tento příkaz říká počítači, na kterém řádku má v řešení programu pokračovat.

Podívejme se nyní, jak tedy probíhal náš program:

1. Počítat podle příkazu na řádku 10 vypsal text COMMODORE 64 na obrazovku.
2. Postoupil v řešení na další řádek (tedy 20) a zde dostane příkaz jdi na řádek 10 (GOTO 10).
3. Vrátí se tedy na řádek 10 a vykoná příkaz PRINT.
4. Tuto smyčku opakuje tak dlouho, dokud ji nepřerušíme klávesou RUN/STOP.

4.3. Příkaz LIST

Používáme-li přímého módu, není možno příkaz, který již není na obrazovce, znovu vyvolat. V programovém módu je tomu jinak. Program, který jsme do počítače vložili, můžeme kdykoliv vyvolat napsáním příkazu LIST a stisknutím klávesy RETURN.

Program, který se po příkazu LIST vypíše na obrazovce lze samozřejmě znovu spustit, upravit nebo uložit na vnější paměť (magnetofon nebo disketu).

4.4. Příkazy pro editaci

Uděláte-li při zápisu programu nějakou chybu, můžete ji opravit a to hned několika způsoby:

1. Můžete řádek napsat znovu se stejným číslem. Po odeslání klávesou RETURN se starý řádek nahradí novým.
2. Napíšete-li pouze číslo řádku a stisknete RETURN, v programu se vymaže řádek s tímto číslem.
3. Máte-li výpis programu na obrazovce, můžete řádek použitím kláves pro pohyb kurzoru a editovacích kláves libovolně opravovat. Po stisknutí klávesy RETURN se a opravený řádek uloží do paměti.

Nyní se pokusíme opravit náš původní program. Nechte jej vypsat na obrazovku příkazem LIST. Nyní pomocí kláves pro pohyb kurzoru kláves SHIFT a INST/DEL opravte první řádek na tvar:

```
10 PRINT "COMMODORE";
```

Spusťte program příkazem RUN. Zjistíte, že počítač vypisuje své jméno do řádku vedle sebe. To proto, že jsme v příkazu PRINT použili jako oddělovač středník (;).

4.5. Proměnné

Jedním z nejdůležitějších pojmů všech programovacích jazyků jsou proměnné. Proto je velmi důležité seznámit se s jejich vlastnostmi a využitím.

Každou proměnnou si můžeme představit jako zásuvku, kterou si můžeme libovolně pojmenovat, a do níž můžeme během programu ukládat různé hodnoty.

Vyzkoušejme si to na příkladu:

```
10 X = 5
20 Y = X + 8
30 PRINT "VYSLEDEK:";Y
```

V prvním řádku programu jsme proměnné X přiřadili hodnotu 5. Ve druhém řádku jsme proměnné Y přiřadili hodnotu X+8 tedy 5+8. Výsledek tedy bude 13.

Proměnným nemusíme přiřazovat pouze číselné hodnoty. Proměnné mohou obsahovat například i řetězce znaků. Podle toho, co proměnné obsahují, dělíme je na tři základní kategorie:

1. Celočíselné proměnné (integer)
   označují se znakem % za jménem (např. A%, X%, M1% apod.) a mohou obsahovat pouze celá čísla.
2. Řetězcové proměnné (string)
   označují se znakem $ za jménem (např. A$, X$, M1$ apod.) a mohou obsahovat libovolné znaky.
3. Proměnné typu real (floating point)
   nemají za svým jménem žádný znak (např. A, X, Ml) a mohou obsahovat libovolná reálná čísla.

Při výběru jména pro proměnné musíme dodržet několik následujících zásad:

1. Jméno proměnné musí obsahovat jeden nebo dva znaky, nepočítaje v to znak pro rozlišení řetězcových a celočíselných proměnných.
2. Abychom se mohli lépe orientovat v programu, je možno skládat jméno proměnné z více znaků (např. DEN, MESIC$ apod.)
3. Obsahuje-li jméno více než dva znaky, musíme počítat s tím, že počítač identifikuje proměnnou podle prvních dvou znaků a jména. Není tedy schopen rozlišit proměnnou VYSLEDEK a VYPOCET.

4. V jednom programu je možno používat jedno jméno u různých typů proměnných. Je tedy možno v jednom programu použít proměnné X, X$ a X%.
5. První znak jména proměnné musí být písmeno. Druhý znak je pak libovolný. Znak rozlišující typ proměnné musí být poslední.
6. Jméno proměnné nesmí obsahovat klíčová slova jazyka BASIC (např. GOTO, TO, THEN apod.). Seznam těchto slov je uveden v příloze originálního manuálu.

4.6. Příkaz smyčky FOR/NEXT

Opakovaný výpis, který jsme si ukázali v minulé kapitole, jde realizovat také pomocí smyčky FOR - NEXT. Výhodou použití této smyčky je to, že můžeme určit počet výpisů.

Zavedeme si jednu proměnnou, kterou budeme používat pro registrování počtu výpisů. Nazveme ji třeba POCET. Budeme-li chtít, aby počítač vypsal své jméno na obrazovku čtyřikrát, budeme při každém výpisu zvyšovat hodnotu proměnné o 1 a kontrolovat její hodnotu. Dosáhne-li proměnná hodnoty 4, ukončíme program.

Pozorně opište následující program a spusťte jej příkazem RUN:

```
10 FOR POCET=1 TO 4
20 ?"COMMODORE 64"
30 NEXT POCET
```

Výsledkem by měl být výpis:

```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Jak vlastně program probíhal?

1. V řádku 10 byla proměnné přiřazena hodnota 1.
2. V řádku 20 počítač vypsal na obrazovku požadovaný text.
3. V řádku 30 byla hodnota proměnné zvýšena o 1. Počítač zkontroloval, zda nebyl překročen rozsah definovaný pro proměnnou POCET v řádku 10 a program se vrátil na začátek smyčky (tedy na řádek 10).
4. Tímto způsobem proběhl výpis celkem čtyřikrát. Po čtvrtém průchodu měla proměnná POCET hodnotu 4. V řádku 30 byla opět zvýšena o 1. Její hodnota tím přesáhla předepsaný rozsah, proto počítač smyčku ukončil.

O tom, jakou hodnotu má proměnná POCET během programu, se můžete přesvědčit, upravíte-li řádek 20 na tvar:

```
20 ?"COMMODORE 64","POCET ="; POCET
```

Při použití smyčky FOR - NEXT není nutné zvyšovat hodnotu proměnné vždy o 1. Krok (step), o který bude hodnota proměnné zvyšována, si můžeme zvolit. Vyzkoušejte si to na následujících příkladech:

```
10 FOR A=l TO 10 STEP .5
20 ?A
30 NEXT A

10 FOR A=2 TO 8 STEP 2
20 ?A
30 NEXT A

10 FOR A=8 TO 2 STEP -2
20 ?A
30 NEXT A
```

4.7. Příkaz IF/THEN

Příkaz IF/THEN se používá především pro podmíněné větvení programu. Jestliže počítač narazí při běhu programu na příkaz IF/THEN, prověří, zda je podmínka uvedená za slovem IF splněna. Jestliže ano, vykoná příkaz uvedený za slovem THEN, jestliže splněna není, pokračuje program na následujícím řádku:

Ukážeme si to na příkladu:

```
10 X=60
20 X=X+1
30 ?"COMMODORE 64"
40 IF X=64 THEN END
50 GOTO 20
```

Výsledkem tohoto programu bude opět čtyřnásobný výpis textu COMMODORE 64. Program proběhl následovně:

1. V řádku 10 byla proměnné X přiřazena hodnota 60.
2. V řádku 20 byla tato hodnota zvýšena o 1.
3. V řádku 30 byl proveden výpis.

4. V řádku 40 byla prozkoumána hodnota proměnné X a protože nebyla splněna předepsaná podmínka, pokračuje na další řádek.
5. V řádku 50 je příkaz skoku na řádek 20.
6. Program proběhne tímto způsobem celkem čtyřikrát. Hodnota proměnné X se postupně zvýší až na hodnotu 64.
7. Postoupí-li program na řádek 40 a hodnota proměnné X je 64, pak je splněna podmínka předepsaná v tomto řádku a program je ukončen příkazem END, který je napsán za slovem THEN.

Při používání příkazu IF/THEN můžeme použit následujících podmínek:

symbol	význam
<	je menší než
>	je větší než
=	je roven
<>	není roven
<=	je menší nebo roven
>=	je větší nebo roven

5. Pokračování BASICu

5.1. Úvod

Následující kapitoly jsou určeny pro ty z Vás, kteří již mají zkušenost s programovacím jazykem BASIC. Popisuje postupy, které se používají pro psaní náročnějších programů.

Pro ty z vás, kteří následujícím kapitolám zcela neporozumí, jsme připravili alespoň několik ukázkových programů v kapitolách 6 a 7 a seznam literatury pro začátečníky v příloze originálního manuálu.

5.2. Pohyb na obrazovce

Pozorně si opište následující program, který využívá příkazů, které jsme se již naučili. Novinkou je pouze využití příkazu PRINT pro pohyb kurzoru.

Chceme-li v přímém módu například pohnout kurzorem vlevo, stiskneme příslušnou klávesu CRSR. Chceme-li vymazat obrazovku, stiskneme klávesy SHIFT a CLR/HOME. Jak to ale udělat přímo z programu? Velice snadno. Použijeme příkazu PRINT, otevřeme uvozovky a stiskneme příslušnou klávesu. Stiskneme-li např. SHIFT a CLR/HOME, obrazovka se nám nevymaže, ale mezi uvozovkami se objeví inverzní znak srdce. Dojde-li počítač v programu na toto místo, smaže obrazovku. Podobně je to i s pohybem kurzoru.

```
  10 REM SKAKAJICI MIC
  20 PRINT "<SHIFT><CLR/HOME>"
  25 FOR X=1 TO 10: PRINT"<CRSR DOLU>";: NEXT
  30 FOR BL=1 TO 40
  40 PRINT "<SPACE><SHIFT>Q<CRSR VLEV>";
  50 FOR TM=l TO 5
  60 NEXT TM
  70 NEXT BL
  75 REM POHYB MICKEM DOLEVA
  80 FOR BL=40 TO 1 STEP -1
  90 PPINT "<SPACE><CRSR VLEVO><CRSR VLEVO><SHIFT>Q<CRS
VLEVO>";
 100 FOR TM=1 TO 5
 110 NEXT TM
 120 NEXT BL
 130 GOTO 20
```

Pozn: Povšimněte si, že lze napsat na jeden řádek i více příkazů, je však nutné je oddělit dvojtečkou.

Pokud jste program správně napsali, objeví se po spuštění příkazem RUN na obrazovce míček, který se bude pohybovat mezi levým a pravým krajem obrazovky. Podívejme se nyní na program podrobněji.

1. V řádku 10 je za REM (REMark = poznámka) napsán komentář, který obsahuje název programu. Tento komentář předznamenaný REM nemá žádný vliv na zpracování programu.
2. V řádku 20 se vymaže obrazovka.
3. V řádku 25 se desetkrát zopakuje příkaz posun kurzoru dolů.
4. V řádku 30 začíná smyčka pro pohyb míčku zleva doprava (od sloupce 1 do sloupce 39).
5. V řádku 40 se vytiskne nejprve mezera (SPACE), pak se vytiskne míček a kurzor se vrátí o jednu pozici zpět. Pohybem zpět je zajištěno, aby při příštím průchodu byl míček mezerou smazán.
6. Smyčka na řádku 50 a 60 zpomaluje pohyb míčku. Bez této smyčky by byl míček sotva viditelný.
7. V řádku 70 je ukončena smyčka pro pohyb míčku vpravo. Program běží v této smyčce, dokud míček nedosáhne 39 sloupce.
8. V řádkách 80 až 120 je podobný program pro pohyb míčku zprava doleva.
9. Dosáhne-li míček levého okraje, tedy sloupce 0, vrací se program zpět na řádek 20.
10. Provázání výše popsaných smyček je velmi dobře patrno z obrázku anglického manuálu.

Používáme-li v programu více vnořených smyček, je bezpodmínečně nutné dodržet následující pravidlo. První otevřená smyčka musí být uzavřena jako poslední a poslední otevřená smyčka musí být uzavřena jako první. Spojíme-li tedy ve výpisu začátky a konce smyček (podobně jako na obr. v originálním manuálu), nesmí se nám čáry křížit.

5.3. Příkaz vstupu INPUT

Do programů, o nichž byla dosud řeč, nebylo možno během výpočtů nikterak zasáhnout. Použijeme-li však příkaz INPUT, můžeme do programu vkládat data, aniž bychom jej museli zastavit.

Následující jednoduchý program vám umožní udělat si představu o tom, jak se pracuje s příkazem INPUT.

```
10 INPUT A$
20 PRINT "VLOZILI JSTE:";A$
30 PRINT
40 GOTO 10
```

Spustíte-li program příkazem RUN, zastaví se na řádku 10 a zde vypíše otazník. Nyní očekává vstup z klávesnice. Zadejte tedy libovolnou kombinaci znaků (např. COMMODORE) a ukončete vstup klávesou RETURN. Program postoupí na další řádek a vypíše vámi vložený řetězec znaků.

Chcete-li, aby váš program byl přehlednější, můžete každý příkaz INPUT doplnit komentářem. Komentář však nesmí obsahovat více jak 38 znaků.

Použijeme-li komentáře, bude řádek 10 našeho programu vypadat následovně:

```
10 INPUT "VLOZTE RETEZEC:",A$
```

Pomocí příkazu INPUT lze do programu vkládat jak řetězcové tak numerické údaje. Záleží pouze na proměnné, která po příkazu INPUT následuje.

Využití příkazu INPUT ukazuje následující program:

```
1   REM PPOGRAM PRO PREPOCET TEPLOTY
5   PRINT "<SHIFT><CLR/HOME>"
10  PRINT "PREPOCET Z FANHREITA NEBO CELSIA (F/C)":INPUT A$
20  IF A$="" THEN 10
30  IF A$="F" THEN 100
40  IF A$<>"C" THEN END
50  INPUT "VSTUP VE STUPNICH CELSIA:";C
60  F=(C*9)/5+32
70  PRINT C;"STUPNU CELSIA =";F;"STUPNU FAHRENHEITA"
80  PRINT
90  GOTO 10
100 INPUT "VSTUP VE STUPNICH FAHRENHEITA:";F
110 C=(F-32)*5/9
120 PRINT F; "STUPNU FAHRENHEITA =";C;"STUPNU CELSIA"
130 PRINT
140 GOTO 10
```

Jak vlastně tento program pracuje?

1. V řádku 10 se vypíše text a počítač čeká na vstup.
2. V řádcích 20, 30 a 40 se tento vstup ověřuje. Nebyl-li vložen žádný znak (pouze stisknuta klávesa RETURN) vrátí se program na řádek 10. Byl-li vložen znak F,

postoupí program na řádek 100. Byl-li vložen znak C, pokračuje program na řádku 50. Vložení jiného znaku má za následek zastavení programu v řádku 40.

3. V řádku 50 se vypíše komentář u příkazu INPUT a počítač opět čeká na vstup (tentokrát numerický).
4. V řádku 60 se provede žádaný výpočet.
5. V řádku 70 se provede výpis zadané hodnoty i výsledku.
6. Příkaz PPINT v řádku 80 má za následek vynechání řádku při výpisu na obrazovku a příkaz GOTO v řádku 90 vrací program zpět.
7. Obdobně se provádí opačný přepočet na řádkách 100 až 140.

5.4. Příkaz vstupu GET

Pomoci příkazu GET je možno vkládat podobně jako pomocí příkazu INPUT. Počítač však nečeká na potvrzení vstupu klávesou RETURN a čte pouze klávesu, která je stisknuta v okamžiku, kdy je příkaz GET vykonáván. Vstup dat lze tedy pomocí příkazu GET urychlit, ale je potřeba program náležitě ošetřit.

Příkaz GET přiřazuje hodnotu právě stisknuté klávesy proměnné, která je uvedena za tímto příkazem. Vysvětlíme si to opět na příkladu:

```
10 GET A$: IF A$="" THEN 10
20 PRINT A$
30 GOTO 10
```

Vzhledem k tomu, že příkaz GET pracuje bez přerušení, je nutno ošetřit program tak, aby nepostoupil na další řádek, dokud nebyla stisknuta žádná klávesa (viz řádek 10). Stiskneme-li libovolnou klávesu, není již splněna podmínka ve druhém příkazu v řádku 10 (řetězcová proměnná již není prázdná) a vypíše se námi vložený znak. Všimněte si, že znak je vypisován až příkazem PRINT a nikoliv příkazem GET.

Podle právě získaných vědomostí můžeme upravit program pro přepočet teplot z minulé kapitoly. Opravíme řádky 10 a 20:

```
10 PRINT "PREPOCET Z FAHRENHEITA NEBO CELSIA(F/C)"
20 GET A$: IF A$="" THEN 10
```

Nyní již nemusíme vstup potvrzovat klávesou RETURN.

S výhodou lze použít příkazu GET ve spojení s tzv. funkčními klávesami, které jsou umístěny na pravé straně klávesnice. Stejné jako všechny ostatní klávesy mají i tyto definován svůj ASCII kód (viz příloha A). Použití si opět ukážeme na příkladu:

```
10 GET A$: IF A$="" THEN 10
20 IF A$=CHR$(133) THEN PRINT "COMMODORE 64"
30 GOTO 10
```

Jak se dá z přílohy A snadno zjistit, má funkční klávesa F1 ASCII kód 133. Stiskneme-li ji tedy, přiřadí se proměnné A$ hodnota tohoto kódu a počítač vypíše požadovaný text.

5.5. Generování náhodných čísel

Váš počítač COMMODORE 64 má celou řadu funkcí, které můžete využívat ve svých programech. Jedna z těchto funkcí je generování náhodného čísla RND (RaNDom = náhodný). Jakým způsobem tato funkce pracuje, si ukážeme na následujícím příkladu:

```
10 FOR X=1 TO 10
20 PRINT RND(1)
30 NEXT
```

Po skončení programu by se mělo na vaší obrazovce objevit deset náhodných čísel z intervalu od 0 do 1. O tom, že jsou tato čísla opravdu náhodná, se můžete přesvědčit několikanásobným spuštěním programu. Pokaždé budou čísla na obrazovce jiná.

Chceme-li tuto funkci využít například pro simulování hry v kostky, musíme nějakým způsobem upravit interval, ve kterém se nahodilá čísla pohybují. Upravme tedy řádek 20 následujícím způsobem:

```
20 PRINT 6*RND(1)
```

Nyní již leží čísla v požadovaném rozsahu, ale stále to nejsou čísla celá. I s tímto problémem si však umíme poradit. Použijeme další z funkcí našeho počítače, a sice funkci INT (INTeger = celé číslo). Opět tedy provedeme změnu v řádku 20:

```
20 PRINT INT(6*RND(1))
```

Funkce INT nám odřízla od vygenerovaných čísel všechna desetinná místa. Nyní již dokážeme generovat čísla celá ve

správném rozsahu, ale generují se od čísla 0 do 5 místo od 1 do 6. I zde je snadná pomoc. Provedeme tedy poslední změnu v řádku 20:

```
20 PRINT INT(6*RND(1))+1
```

Nyní můžeme být konečně se svým programem spokojení.

Obecně tedy platí, že velikost rozsahu generovaných náhodných čísel je dána číslem, kterým násobíme funkci RND. Posun tohoto rozsahu docílíme přičtením patřičného čísla k výsledku. Chceme-li, aby výsledek bylo číslo celé, použijeme funkci INT. Schematicky lze toto vyjádřit následovně:

```
Náhodné číslo = INT(rozsah * RND(1)) + posuv
```

5.6. Hra na uhádnutí čísla

Nyní si ukážeme, jak lze využít náhodných čísel při programování her. Kromě toho nám následující program ukáže některé nové programátorské triky.

```
1   REM HRA HADANI CISEL
2   PRINT "<SHIFT><CLR/HOME>"
5   INPUT "HORNI HRANICE CISLA:";LI
10  NM=INT(LI*RND(1))+1
15  CN=0
20  PRINT "A TAK MUZEME ZACIT!"
30  INPUT "VAS TIP:";GU
35  CN=CN+1
40  IF GU>NM THEN PRINT "ME CISLO JE MENSI":PRINT:GOTO 30
50  IF GU<NM THEN PRINT "ME CISLO JE VETSI":PRINT:GOTO 30
60  IF NM-GU THEN PRINT "DOBŘE - UHADL JSTE SPRAVNE CISLO"
65  PRINT "NA";CN;"POKUSU":PRINT
70  PRINT "CHCETE JEŠTĚ HRAT? (A/N)";
80  GET AN$:IF AN$="" THEN 80
90  IF AN$="A" THEN 2
100 IF AN$="N" THEN 80
```

Spustíte-li tento program, počítač se vás zeptá na horní hranici intervalu, ve kterém se má náhodné číslo vyskytovat. Vygeneruje toto číslo, porovnává jej s vašimi tipy a napovídá, zda váš tip je větší nebo menší než náhodné číslo. Pokuste se sami upravit program tak, aby bylo možno určit i dolní hranici intervalu náhodného čísla.

Počet vašich pokusů se ukládá do proměnné CN, jejíž hodnota se po každém vašem pokusu zvýší o 1.

Povšimněte si v programu několika zajímavostí. V řádcích 40 a 50 jsou za příkazem THEN uvedeny různé další příkazy oddělené dvojtečkou. Tento způsob zápisu šetří jednak práci při psaní programu a jednak i paměť. Nesmíme ovšem zapomenout na to, že všechny příkazy uvedené za příkazem THEN se vykonávají pouze při splněné podmínce IF.

5.7. Hra v kostky

Další program simuluje hru v kostky se dvěma kostkami.

```
5  PRINT "ZKUSIS SVE STESTI?"
10 PRINT "CERVENA KOSTKA";INT(6*RND(1))+1
20 PRINT "BILA KOSTKA";INT(6*RND(1))+1
30 PRINT "STISKNI MEZERU PRO DALSI HOD":PRINT
40 GET A$:IF A$="" THEN 40
50 IF A$=CHR$(32) THEN 10
```

Ověřte si své znalosti o programování a pokuste se porozumět tomu, jak je tento program postaven.

5.8. Náhodné kresby

Pomocí funkce RND je možno generovat i náhodné znaky. Dokážeme si to na následujícím krátkém programu:

```
10  PRINT "<SHIFT><CLR/HOME>"
20  PRINT CHR$(205.5+RND(1));
30  GOTO 20
```

Nejdůležitější funkcí v tomto programu je funkce CHR$. Každému číslu od 0 do 255 uvedenému v závorce za touto funkcí odpovídá nějaký znak. Generujeme-li toto číslo funkcí RND, zobrazuje se na obrazovce pokaždé jiný náhodný znak.

Jaký znak odpovídá určitému číslu tzv. ASCII kódu, můžeme zjistit v příloze A.

Pro generování náhodného znaku jsme použili vzorec (205.5+RND(1)), proto se nám budou generovat náhodná čísla v rozsahu od 205.5 do 206.5. Vhledem k tomu, že funkce CHR$ zanedbává desetinná čísla, budou znaky s ASCII kódem 205 a 206 vypisovat se stejnou pravděpodobností.

Potřebujeme-li zjistit, jakou má hodnotu ASCII kód určitého znaku, použijeme funkci inverzní k funkci CHR$. Touto funkcí je funkce ASC.

```
PRINT ASC("X")
```

V tomto případě X je znak, jehož ASCII kód nás zajímá. Může to být libovolný tisknutelný znak.

6. Barevná grafika

6.1. Barva a grafika

Na jednoduchých programech jsme již poznali některé z bohatých programových možností počítače COMMODORE 64. Jednou z nejvíce fascinujících je možnost vytvářet obrazy v barevné grafice. V této části vám ukážeme, jak můžete vytvořit barevnou grafiku a jak ji ve vlastních programech a hrách využít.

Dosud jsme se soustředili na výpočetní možnosti našeho počítače a omezili jsme se na standardní barvy obrazovky (světle modré písmo na tmavě modrém podkladu a světle modrý okraj obrazovky). V této kapitole si ukážeme, jak lze tyto barvy měnit a jak lze využívat velkého množství grafických znaků, které nám náš počítač nabízí.

6.2. Zadávání barev příkazem PRINT

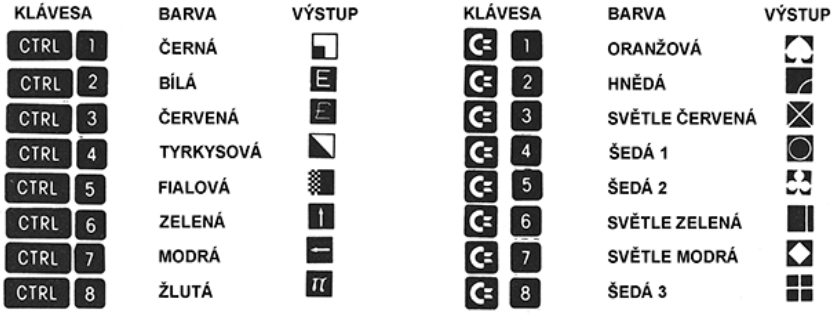
Pokud jste dělali zkoušku nastavení barev v kapitole 1.4, tak již víte, že lze měnit barvu písma současným stisknutím kláves CTRL a číslice 1 až 8. Jak je však možno měnit barvy v programu?

Jak jsme se mohli přesvědčit již v programu "Skákající míč", je možno některé speciální funkce klávesnice zadat do programu pomocí příkazu PRINT. K dispozici máme následujících 16 barev. Některé z nich se zadávají současným stisknutím klávesy CTRL a číslice a některé se zadávají současným stisknutím klávesy C= a číslice.

CTRL 1	černá	C= 1	oranžová
CTRL 2	bílá	C= 2	hnědá
CTRL 3	červená	C= 3	světle červená
CTRL 4	tyrkysová	C= 4	šedá 1
CTRL 5	fialová	C= 5	šedá 2
CTRL 6	zelená	C= 6	světle zelená
CTRL 7	modrá	C= 7	světle modrá
CTRL 8	žlutá	C= 8	šedá 3

Zadejte nyní příkaz NEW a udělejte následující experiment. Zadejte do řádku 10 příkaz PRINT, otevřete uvozovky, stiskněte současně klávesy CTRL a číslici 1. Nyní vložte písmeno S, stiskněte současně klávesy CTRL a číslici 2, napište písmeno P a takto pokračujte. Do uvozovek příkazu

PRINT tímto způsobem vložte slovo SPECTRUM a před každým písmenem změňte barvu písma. Stejně jako při řízení pohybu kurzoru v programu "Skákající míč" budou i zde řídící znaky pro změnu barvy představovány grafickými symboly (viz tabulka).



Pozn.: po skončení programu zůstane barva písma nastavena podle posledního příkazu. Do původního stavu lze počítač uvést současným stisknutím kláves RUN/STOP a RESTORE.

6.3. CHR$ kódy barev

Nejen kódy barev, ale i kódy pro posun kurzoru, vymazání obrazovky a další nalezneme v již několikrát vzpomínané příloze originální příručky. Při programování pak lze používat tyto kódy následujícím způsobem.

```
PRINT "<SHIFT><CLR/HOME>"
```

lze nahradit příkazem

```
PRINT CHR$(147)
```

Následující program vytvoří na obrazovce barevné pruhy. Všimněte si, že se řádky 40 až 190 liší pouze argumenty funkce CHR$. Můžete tedy při psaní programu napsat pouze řádek 40 a další řádky vytvořit pouze opravou tohoto řádku (viz pokyny pro editaci).

```
1   REM BAREVNE PRUHY
5   PRINT CHR$(147):REM VYMAZANI OBRAZOVKY
10  PRINT CHR$(18), "          ";:REM INVERZNI PRUH
20  CL=INT(8*RND(1))+1
30  ON CL GOTO 40,50,60,70,80,90,100,110
```

```
40  PRINT CHR$(5);:GOTO 10
50  PRINT CHR$(29);:GOTO 10
60  PRINT CHR$(30);:GOTO 10
70  PRINT CHR$(31);:GOTO 10
80  PRINT CHR$(144);:GOTO 10
90  PRINT CHR$(156);:GOTO 10
100 PRINT CHR$(158);:GOTO 10
110 PRINT CHR$(159);:GOTO 10
```

Pokyny pro editování

Text programu až do řádku 40 vkládejte běžným způsobem. Nyní se pomocí kláves pro řízení kurzoru vraťte kurzorem na řádek 40. Opravte číslo řádky z 40 na 50, pomocí kláves SHIFT a INS/DEL si vytvořte místo pro dvojciferný argument funkce CHR$, opravte tento argument a klávesou RETURN odešlete takto opravený řádek do paměti. Necháme-li si nyní vypsat program příkazem LIST, zjistíme, že obsahuje jak řádek 40 tak i řádek 50. Podobným způsobem můžeme vytvořit i ostatní řádky programu.

Nyní se podívejme, jak vlastně tento program funguje.

1. V řádku 5 se vymaže obrazovka (CHR$(147) odpovídá SHIFT CLR/HOME)
2. V řádku 10 se zapne inverzní mód (CHR$(18) odpovídá CTRL 9) a vypíše 10 inverzních mezer.
3. V řádku 20 se proměnné CL přiřadí hodnota náhodného čísla v rozsahu od 1 do 8.
4. V řádku 30 je použit příkaz ON – GOTO, který umožňuje větvení programu. Počítač pokračuje v řešení na řádku udaném za GOTO, podle hodnoty proměnné CL.
5. Následující řádky programu mění aktuální barvu písma v závislosti na hodnotě proměnné CL. Po změně barvy písma se program vrací zpět na řádek 10.

6.4. PEEK a POKE

Nyní se seznámíme s metodou, pomocí které lze zjistit obsah kteréhokoliv paměťového místa nebo naopak na toto místo zapsat libovolnou hodnotu.

Stejně jako proměnnou lze kterékoliv paměťové místo v našem počítači přirovnat k zásuvce informací. Obsah některých zásuvek je pro běh počítače velmi důležitý. Existují místa v paměti, kde počítač zjistí, jakou barvu má mít rámeček, podklad obrazovky nebo právě tisknutý znak. Změníme-li obsah

těchto paměťových míst, změní se barvy na obrazovce. Podobným způsobem, tedy vložením určité hodnoty na určité paměťové místo, se nechají generovat tóny, zobrazovat znaky apod.

Každé paměťové místo má své číslo, kterému říkáme adresa. Paměťové místo, na kterém je uložena informace o barvě okraje obrazovky má adresu 53280.

```
POKE 53280,x
```
Kde 'x' je kód barvy.

Podobně lze změnit i barvu podkladu obrazovky příkazem:

```
POKE 53281,x
```

Každé paměťové místo může obsahovat kterékoliv číslo z rozsahu od 0 do 255.

Vzhledem k tomu, že náš počítač využívá 16 barev, jsou pro jejich záznam využita pouze čísla od 0 do 15. Zapsání jiného čísla na tyto adresy nemá význam.

V následující tabulce jsou uvedena čísla jednotlivých barev:

0	černá	8	oranžová
1	bílá	9	hnědá
2	červená	10	světle červená
3	tyrkysová	11	šedá 1
4	fialová	12	šedá 2
5	zelená	13	světle zelená
6	modrá	14	světle modrá
7	žlutá	15	šedá 3

A nyní se podívejme na různé kombinace barev okraje a pozadí obrazovky. Všechny kombinace nám předvede následující program:

```
10 FOR BO=0 TO 15
20 FOR BA=0 TO 15
30 POKE 53280,BA
40 POKE 53281,BO
50 FOR X=1 TO 500:NEXT X
60 NEXT BA:NEXT BO
```

Smyčky tohoto programu jsou do sebe vnořeny takovým způsobem, že se nám na obrazovce vystřídají všechny možné kombinace barev. Přídavná smyčka na řádku 50 celý program jen trochu zpomaluje.

Po skončení programu zadejte počítači příkaz:

```
PRINT PEEK(53280) AND 15
```

Pomocí tohoto příkazu zjistíme, jakou hodnotu má nyní paměťové místo 53280. Pokud program proběhl bez chyby, měla by se nám na obrazovce objevit hodnota 15.

Logickou operací AND mezi obsahem paměťového místa a číslem 15 jsme vyloučili z výpisu všechna čísla větší než 15. Podrobněji se budeme logickými operacemi zabývat v následující kapitole.

6.5. Grafika na obrazovce

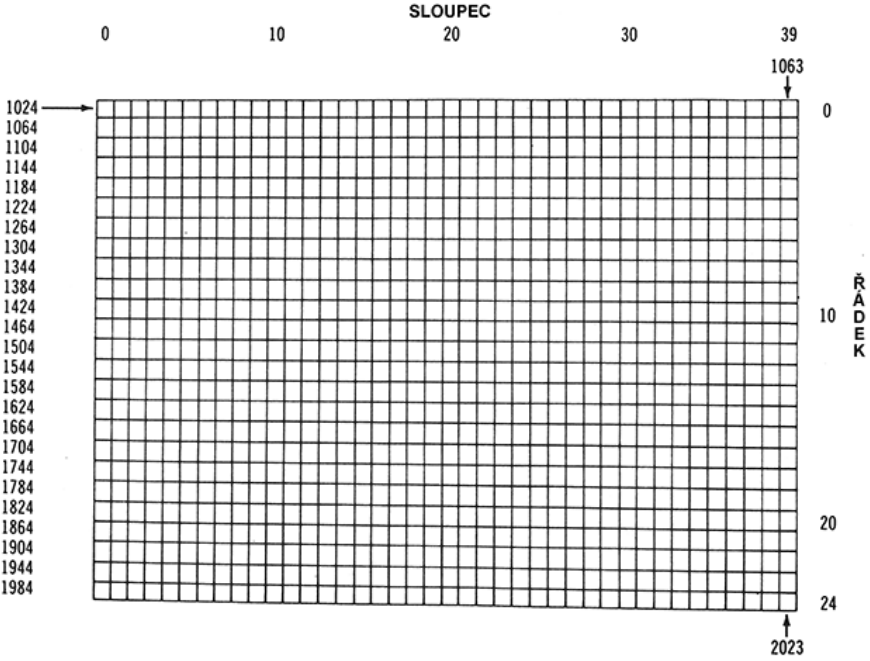
Až dosud jsme tiskly znaky na obrazovku pomocí příkazu PRINT, tedy jeden po druhém. Přestože jsme se naučili tímto příkazem řídit i kurzor a tak dosáhnout libovolného místa na obrazovce, je tato metoda pomalá a náročná na cenné místo v paměti.

Stejně jako existují v paměti místa, které určují barvy obrazovky, existují i místa odpovídající jednotlivým pozicím na obrazovce, kam je možno uložit kód znaku, který má být zobrazen.

6.6. Paměť pro obrazovku

Vzhledem k tomu, že obrazovka našeho počítače má 25 řádků po 40 znacích, můžeme na ni tedy zaznamenat až 1000 znaků. Každé pozici na obrazovce musí odpovídat jedno paměťové místo v paměti našeho počítače.

Obrazovku našeho počítače si můžeme představit jako pravoúhlou síť, jejíž každé políčko odpovídá jednomu paměťovému místu neboli jedné znakové pozici na obrazovce (viz obr.).



Do těchto paměťových míst můžeme pomocí příkazu POKE zapsat libovolnou hodnotu v rozsahu od 0 do 255. To, jaký znak odpovídá té které hodnotě, zjistíme z tabulky v příloze originální příručky.

Jak jsme si již řekli, obsahuje paměť určená pro obrazovku 1000 paměťových míst. Adresa prvního paměťového místa obrazovky (pozice v levém horním rohu) je 1024. Z toho vyplývá, že adresa posledního místa obrazovky (pozice v pravém dolním rohu) je 2023.

Nyní se vraťme k našemu skákajícímu míči a předpokládejme, že chceme, aby se míč objevil asi uprostřed obrazovky, tj. na řádku 12 ve sloupci 20. Adresu paměťového místa obrazovky vypočteme obecně následujícím způsobem:

adresa pozice = 1024 + sloupec + 40 * řádka

Adresa paměťového místa bude tedy v našem případě:

1024 + 20 + 40 * 12 = 1524

Z tabulky v příloze originální příručky zjistíme, že kód pro znak, který používáme pro znázornění míče, je 81. Smažeme tedy obrazovku příkazem SHIFT CLR/HOME a zadáme příkaz:

POKE 1524,81

Proč se na požadovaném místě obrazovky nyní neobjevil náš míč, se dozvíme v následující kapitole.

6.7. Paměť pro barvy znaků

Dosáhli jsme toho, že jsme vytiskli znak uprostřed obrazovky, aniž bychom použili příkaz PRINT. Zatím však znak nevidíme, protože má stejnou barvu jako pozadí. Musíme tedy změnit barvu znaků.

Stejně jako je v paměti našeho počítače rezervováno 1000 míst pro znaky obrazovky, je zde rezervováno i 1000 míst pro jejich barvu.

Paměťová místa, ve kterých je uchována informace o barvě znaku na obrazovce, začínají od adresy 55296. Určitou pozici na obrazovce vypočteme obdobným způsobem jako v případě znaků.

Adresa pozice = 55296 + sloupec + 40 * řádka

Vypočteme tedy podle uvedeného vzoru adresu našeho znaku:

55269 + 12 + 40 * 20 = 55796

Na tuto adresu musíme uložit příkazem POKE kód barvy, který znak má mít. Kódy barev jsou opět v rozsahu od 0 do 15 a odpovídají těm, které jsme použili pro změnu barvy podkladu a okraje obrazovky.

Příkaz pro nastavení červené barvy míče bude mít tedy tvar:

POKE 55796,2

6.8. Další hra se skákajícím míčem

Program uvedený v této kapitole je velmi podobný již dříve uvedené hře s míčem. Na rozdíl od původní verze však nepoužívá příkazu PRINT, ale příkazu POKE, který umožňuje pružněji měnit polohu míče.

```
10  PRINT "<SHIFT><CLR/HOME>"
20  POKE 53280,7: POKE 53281,0
30  X=1: Y=1
40  DX=1: DY=1
50  POKE 1024+X+40*Y,32
```

```
55  POKE 55296+X+40*Y,1
60  FOR T=1 TO 10: NEXT
70  POKE 1024+X+40*Y,32
80  X=X+DX
90  IF X<=0 OR X>=39 THEN DX=-DX
100 Y=Y+DY
110 IF Y<=0 OR Y>=24 THEN DY=-DY
120 GOTO 50
```

Popis programu:

1. V řádku 10 se vymaže obrazovka.
2. V řádku 20 se zvolí barva rámečku žlutá a barva podkladu černá.
3. Proměnné X a Y obsahují souřadnice obrazovky, ve kterých se právě nachází míč.
4. Proměnné DX a DY obsahují hodnoty, o které se má míč posunout. Kladná hodnota DX představuje pohyb míče vpravo, záporná vlevo. Kladná hodnota DY představuje pohyb dolů, záporná nahoru.
5. V řádku 50 a 55 se míč vytiskne (včetně informace o barvě).
6. V řádku 60 je nám již dobře známá zpožďovací smyčka.
7. V řádku 70 je míč přepsán mezerou (SPACE = 32).
8. V řádku 80 se přičtením proměnné DX vypočte nová souřadnice X zobrazení.
9. Jestliže se míč dotkl levého či pravého okraje obrazovky, změní se v řádku 90 znaménko proměnné DX.
10. V řádcích 100 a 110 je stejným způsobem ošetřen i pohyb míčku nahoru a dolů.
11. V řádku 120 je instrukce skoku na řádek 50, kde se míč zobrazí na obrazovce v nově vypočtené poloze.

Doplněním programu o další řádky se stane tento ještě dokonalejším:

```
21  FOR L=1 TO 10
25  POKE 1024+INT(RND(1)*1000),166
27  NEXT L
115 IF PEEK(1024+X+40*Y)=166 THEN DX=-DX:GOTO 80
```

Vysvětlení:

1. V řádcích 21, 25 a 27 se na náhodně vygenerovaných pozicích zobrazí překážky.
2. V řádku 115 se pomocí funkce PEEK přezkoumá, zda míč narazil na překážku a jestliže ano, změní směr pohybu míče.

7. Jemná grafika- SPRITE

7.1. Úvod ke SPRITům

V předchozích kapitolách jsme se naučili používat příkazu PRINT pro formátovaný výpis na obrazovku a příkazu POKE pro znázornění libovolného znaku na libovolné pozici obrazovky.

Tyto příkazy však nestačí pro konstrukci pohybujících se obrázků. Jejich konstrukci znesnadňuje omezený počet grafických symbolů našeho počítače a jejich pohyb vyžaduje velké množství matematických operací a příkazů.

Použitím SPRITů odpadá většina uvedených problémů. SPRITE je libovolně definovaný objekt v jemné grafice, jehož polohu na obrazovce lze snadno měnit pomocí jednoduchých příkazů. Všechny potřebné výpočty pro pohyb SPRITů si řeší počítač sám, netřeba je zadávat.

Zobrazení pomocí SPRITů má i řadu dalších výhod. Snadno lze měnit jejich barvu a velikost, vzájemnou kolizi dvou SPRITů lze snadno registrovat a lze určit, který ze dvou obrázků bude na obrazovce překrývat druhý.

Použití SPRITů má ale také jednu nevýhodu a tou je poměrně obtížné programování. Abychom toto programování bezvadně zvládli, je potřeba pochopit ještě několik funkcí našeho počítače.

7.2. Bity a byty

Detailní popis toho, jakým způsobem zpracovává počítač čísla a informace, by daleko přesáhl rozsah tohoto návodu. Abychom však mohli pochopit některé operace, musíme si vysvětlit alespoň několik základních pojmů.

Nejmenší jednotkou informace je jeden BIT. Bit je možno si představit jako přepínač nebo klopný obvod. Bit totiž může nabývat pouze dvou hodnot, a sice hodnoty 0 (vypnuto) a hodnoty 1 (zapnuto). Proto hovoříme o tom, že počítače využívají dvouhodnotové neboli binární aritmetiky.

Osm bitů tvoří v našem počítači jeden BYTE (čti bajt). Jeden byte obsahuje tedy 8 přepínačů a podle jejich kombinace můžeme rozlišit 256 různých stavů. Z toho vyplývá, že jeden byte může nabývat hodnoty od 0 do 255. Byte je také nejmenší jednotkou informace, kterou lze dosáhnout pomocí příkazů Basicu (např. POKE).

REGISTRY jsou paměťová místa, kterých se využívá vždy ke stejné funkci, a obsahují zpravidla jeden byte. Při využívání

SPRITů se setkáme s registry, ve kterých je zaznamenána například barva obrázku, jeho souřadnice a podobně.

Již jsme se zmínili o tom, že počítače využívají pro svou práci binární čísla. Povězme si tedy, jakým způsobem převádíme čísla binární na dekadická a opačně.

Pomůže nám k tomu následující tabulka:

128	64	32	16	8	4	2	1		
0	0	0	0	0	0	0	1	=1 * 2↑0	=1
0	0	0	0	0	0	1	0	=1 * 2↑1	=2
0	0	0	0	0	1	0	0	=1 * 2↑2	=4
0	0	0	0	1	0	0	0	=1 * 2↑3	=8
0	0	0	1	0	0	0	0	=1 * 2↑4	=16
0	0	1	0	0	0	0	0	=1 * 2↑5	=32
0	1	0	0	0	0	0	0	=1 * 2↑6	=64
1	0	0	0	0	0	0	0	=1 * 2↑7	=128

Podle této tabulky můžeme poměrně snadno převést binární číslo na dekadické. Tak například:

10110101 = 1*2↑7 + 0*2↑6 + 1*2↑5 + 1x2↑4 + 0*2↑3 + 1*2↑2 + 0*2↑1 + 1*2↑0 = 128 + 32 + 16 + 4 + 1 = 181

Následující jednoduchý program převádí binární čísla na dekadická.

```
5  REM PREVOD BINARNICH CISEL NA DEKADICKA
10 INPUT "ZADANI 8-BITOVEHO CISLA:";A$
12 IF LEN (A$)<>8 THEN PRINT "8 BITU PROSIM": GOTO 10
15 TL=0: C=0
20 FOR X=8 TO STEP -1: C=C+1
30 TL=TL+VAL(MID$(A$,C,1))*2↑(X-1)
40 NEXT X
45 PRINT: PRINT A$;" BINARNE =";TL;"DEKADICKY"
55 PRINT
60 GOTO 10
```

Program probíhá následovně:

1. Binární číslo se uloží pomocí příkazu INPUT do proměnné A$.
2. Pomocí funkce LEN se zkontroluje délka.
3. Pomocí funkce MID$ se A$ čte znak po znaku a zjišťuje se funkcí VAL, zda tento znak odpovídá hodnotě 1 nebo 0. Tato hodnota se pak násobí příslušnou mocninou čísla 2.
4. Zadání se pak společně s výsledkem vypíše na obrazovku.

7.3. Konstrukce SPRITů

SPRIty jsou ovládány zvláštními obvody počítače COMMODORE 64. Tyto obvody nazýváme VIC (video interface CHIP). Obstarávají veškerou práci, která je spojena s ovládáním SPRITů, řízením barvy, velikosti, polohy a podobně.

Informace o každém SPRITu jsou uloženy ve 46 paměťových místech – registrech. Každý registr obsahuje jeden byte informace. Které informace to jsou, se dozvíme z následující tabulky:

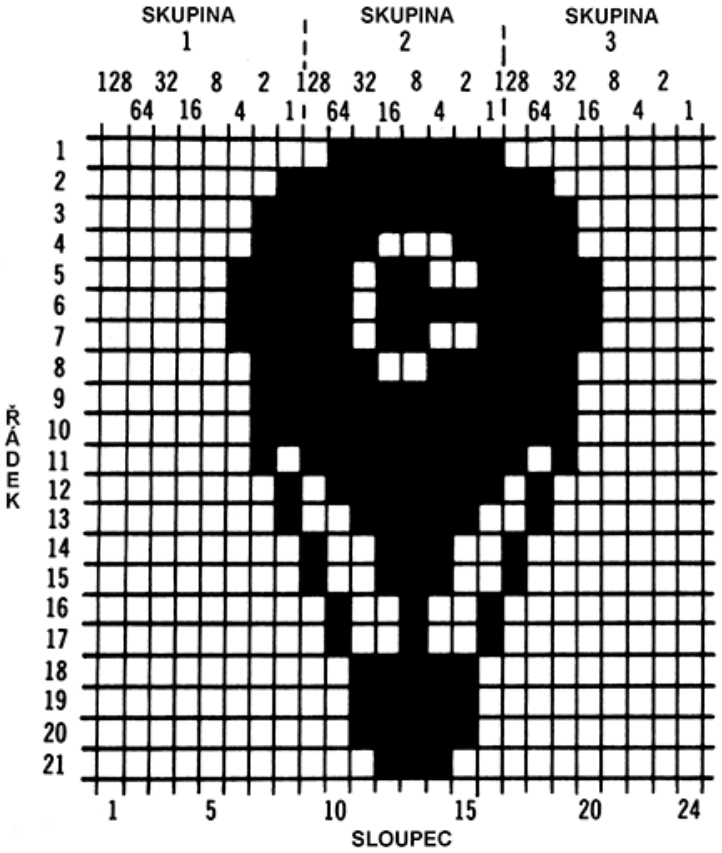
registr	význam
0	souřadnice x spritu 0
1	souřadnice y spritu 0
2	souřadnice x spritu 1
3	souřadnice y spritu 1
4	souřadnice x spritu 2
5	souřadnice y spritu 2
6	souřadnice x spritu 3
7	souřadnice y spritu 3
8	souřadnice x spritu 4
9	souřadnice y spritu 4
10	souřadnice x spritu 5
11	souřadnice y spritu 5
12	souřadnice x spritu 6
13	souřadnice y spritu 6
14	souřadnice x spritu 7
15	souřadnice y spritu 7
21	viditelnost spritů (1/0)
23	dvojnásobný rozměr ve svislém směru
29	dvojnásobný rozměr ve vodorovném směru
27	nastavení priority zobrazení
39-46	nastavení barev spritů

Podrobněji jsou jednotlivé registry popsány v příloze originálního manuálu.

7.4. Jak se tedy SPRITy vytvářejí?

Jak již víme, může obrazovka našeho počítače obsahovat maximálně 25 řádků po 40 znacích, tedy maximálně 1000 znaků. Každý znak se skládá ze 64 bodů(z matice 8 x 8). Použití SPRITů, tedy jemné grafiky, nám zpřístupňuje každý tento bod. Máme tedy k dispozici 320 x 200 bodů.

Pro každý SPRITE můžeme použít maximálně pole 24 bodů široké a 21 bodů vysoké (viz obr.). Nejlépe se tvar SPRITE navrhuje na čtverečkovaném papíru. Nakreslete si pole 24 čtverečků široké a 21 vysoké. Navrhněte tvar obrázku a vybarvěte příslušné čtverečky. Nyní doplňte vaši kresbu o rastr, který vám umožní zadat obrázek do počítače ve formě dat. Stejně jako na obrázku rozdělte kresbu na tři sloupce (skupiny) po osmi bodech a jednotlivé sloupce bodů označte zprava doleva mocninami čísla 2, tedy 1, 2, 4, 8, 16, 32, 128. Očíslujte také řádky od 1 do 23.



Podívejme se, jak je vytvořen soubor dat pro kresbu balónu. V prvním řádku v první skupině odpovídají hodnoty všech bodů 0. Tedy i konečný součet bude 0.

Druhý sloupec (skupina) obsahuje jeden bod s hodnotou 0 a ostatní s hodnotou 1. Celkovou hodnotu tedy vypočteme následovně:

```
128*0 + 64*1 + 32*1 + 16*1 + 8*1 + 4*1 + 2*1 + 1*1 =127
```

Poslední sloupec (skupina) první řádky pak bude mít opět hodnotu 0.

První řádek našeho SPRITu můžeme tedy vyjádřit pomocí dat následovně:

```
DATA 0,127,0
```

Obdobným způsobem vypočteme hodnoty dat pro všechny další řádky a sloupce. Musíme získat celkem 63 hodnot dat.

Jak využít takto získaná data si ukážeme v následujícím programu. Objeví se v něm nové příkazy DATA a READ. Příkaz READ přiřazuje určité proměnné hodnoty, které jsou uvedeny v řádcích označených jako DATA. Podrobněji bude tento příkaz vysvětlen v kapitole 9.

```
1   REM NAHORU, NAHORU A PRYC
5   PRINT "<SHIFT><CLR/HOME>"
10  V=53248: REM ZAKLADNI ADRESA VIC
11  POKE V+21,4: REM AKTIVACE SPRITU 2
12  POKE 2042,13: REM DATA PRO SPRITE 2 Z BLOKU 13
20  FOR N=0 TO 62: READ Q: POKE 832+N,Q: NEXT
30  FOR X=0 TO 200
40  POKE V+4,X: REM NOVA SOURADNICE X
50  POKE V+5,X: REM NOVA SOURADNICE Y
60  NEXT X
70  GOTO 30
199 REM DATA PRO PRIKAZ READ Q
200 DATA 0,127,0,1,255,192,3,255,224,3,231,224
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

Pokud jste program zadali správně, poletí váš balón šikmo vzhůru. Abychom tomuto programu dokonale porozuměli, rozebereme si ho podrobněji.

1. V řádku 10 je použit výraz V=53248. Proměnná V obsahuje počáteční adresu VICu. Budeme-li tedy obsazovat jednotlivé registry tohoto obvodu, použijeme příkazu POKE V+č. registru, hodnota.

2. V řádku 11 zapisujeme do registru č. 21 hodnotu 4. Registr 21 VICu je vyhrazen pro informace o vybavení (resp. zviditelnění) SPRITu. Každý z osmi programovatelných SPRITů (0 až 7) v něm má vyhrazen jeden bit. Zapíšeme-li tedy do tohoto registru číslo 4 (binárně 00000100), bude vybaven na obrazovku SPRITE č. 2. Kdybychom do tohoto registru zapsali například číslo 11 (binárně 00001011), na obrazovce by se objevily SPRITy 0, 1 a 3.
3. V řádku 12 určujeme, kde budou uložena data pro vytvoření SPRITu. Každý SPRITE má v paměti rezervováno jedno paměťové místo, kam se ukládá tzv. SPRITE POINTER neboli ukazatel začátku dat. Tyto ukazatele jsou uloženy na následujících adresách:

   2040 SPRITE POINTER 0
   2041 SPRITE POINTER 1
   2042 SPRITE POINTER 2
   2043 SPRITE POINTER 3
   2044 SPRITE POINTER 4
   2045 SPRITE POINTER 5
   2046 SPRITE POINTER 6
   2047 SPRITE POINTER 7

   Pro definování každého SPRITu je jak víme potřeba 63 dat, tedy 63 bytů. Adresu prvního bytu zjistíme, vynásobíme-li SPRITE POINTER číslem 64. Používáme-li tedy SPRITE 2 a chceme-li ukládat data pro jeho vytvoření od adresy 832, uložíme do SPRITE POINTERu 2 (na adresu 2042) hodnotu 13 (13*64=832).
4. V řádku 20 čteme ve smyčce (FOR – NEXT) postupně všech 63 dat (READ Q) a ukládáme je do paměti od adresy 832 (POKE 832+N,Q) dle předcházejícího výpočtu.
5. V řádcích 30 až 60 je realizován pohyb balónu. Souřadnice X i Y se postupně zvyšují ve smyčce FOR – NEXT od 0 do 200. Tato hodnota se zapisuje do registru č. 4 a 5, které řídí pohyb SPRITu.
6. V řádku 70 je příkaz pro návrat na řádek 30, čímž se celý program zopakuje.
7. V řádcích 200 až 240 jsou uložena data pro tvorbu SPRITu.

Abychom si na příkladech ukázali všechny způsoby využití VICu pro práci se SPRITy, budeme postupně náš program s létajícím balónem rozšiřovat.

```
25 POKE V+23,4: POKE V+29,4
```

8. Po doplnění tohoto řádku do programu se velikost balónu zvětší na dvojnásobek. Je to proto, že jsme v registru 23 (zvětšení ve svislém směru) a v registru 29 (zvětšení ve vodorovném směru) přepsali bit určující zvětšení SPRITu 2 z 0 na 1.

```
11 POKE V+21,12
12 POKE 2042,13: POKE 2043,13
30 FOR X=1 TO 190
45 POKE V+6,X
55 POKE V+7,190-X
```

9. Zapsáním hodnoty 12 (binárně 00001100) do registru 21 jsme kromě SPRITu 2 zaktivovali i SPRITE 3.
10. V řádku 12 jsme oběma zaktivovaným SPRITům přidělili stejná data, tedy i stejný tvar.
11. V řádcích 45 a 55 je naprogramován pohyb SPRITu č. 3, jeho souřadnice X se bude postupně zvyšovat od 0 do 190 a souřadnice Y bude klesat od 190 do 0.

```
11 POKE V+21,28
12 POKE 2042,13: POKE 2043,13: POKE 2044,13
25 POKE V+23,12: POKE V+29,12
48 POKE V+8,X
58 POKE V+9,100
```

12. V řádku 11 jsme zapsáním hodnoty 28 (binárně 00011100) aktivovali SPRITE 2, 3, i 4.
13. V řádku 12 jsme určili, že všechny SPRITy budou mít stejná data.
14. V řádku 25 jsme do registrů pro zvětšení SPRITů zapsali hodnotu 12 (binárně 00001100). Proto bude SPRITE 2 a 3 zvětšen.
15. V řádku 48 je naprogramován pohyb ve směru X pro SPRITE 4. Podle řádku 58 bude mít tento SPRITE konstantní souřadnice Y=100. Proto se balón bude pohybovat pouze vodorovně.

```
15 POKE V+41,13: POKE V+42,6: POKE V+43,8
```

16. Podobně jako barva jednotlivých znaků na obrazovce lze měnit i barvu jednotlivých SPRITů. V řádku 15 jsme určili pro SPRITE 2 barvu světle zelenou, pro SPRITE 3 modrou a SPRITE 4 oranžovou. Pro volbu barev SPRITů jsou určeny registry 39 až 46. Kódy barev jsou stejné jako pro znaky a jsou uvedeny v příloze originálního manuálu.

```
30 FOR X=0 TO 255
```

```
50 POKE V+5,150
55 POKE V+7,50
```

17. Po této úpravě programu budou naše balóny létat vodorovně asi do tří čtvrtin obrazovky. Nemohou dolétnout až k pravému okraji, protože šířka obrazovky je 320 bodů a registr může obsahovat pouze hodnoty od 0 do 255 (binárně 00000000 až 11111111). Abychom mohli posouvat SRITy po celé obrazovce, musíme použít registr 16, ve kterém je uložen nejvyšší (tedy devátý) bit souřadnice X pro každý SPRITE. Program tedy musíme upravit takto:

```
65  POKE V+16,12
70  FOR X=0 TO 63
80  POKE V+4,X
85  POKE V+6,X
88  POKE V+8,X
90  NEXT X
100 POKE V+16,0
110 GOTO 30
```

V řádku 65 se do registru 16 uloží hodnota 12 (binárně 00001100). Znamená to, že jsme nastavili devátý bit souřadnice X SPRITu 2 a 3 na hodnotu 1. Při prvním průchodu smyčkou (řádky 70 až 90) bude mít tedy souřadnice X těchto SPRITů binární hodnotu 100000000 (1 z registru 16, 00000000 z registrů 4 a 6), tedy decimálně 256. Hodnoty registrů 4, 6 a 8 se ve smyčce zvyšují již známým postupem. Na konci smyčky se hodnota registru 16 nuluje, aby let balónů mohl začít opět u levého okraje obrazovky.

7.5. Priorita zobrazení SPRITů

Necháme-li nakreslit dva SPRITy na stejné místo obrazovky, bude jeden překrývat druhý. Více "vpředu" (tedy viditelný) bude vždy SPRITE s nižším číslem. Platí tedy, že SPRITE 0 má prioritu nejvyšší (nemůže být žádným SPRITem překryt) a SPRITE 7 nejnižší (bude překryt kterýmkoliv SPRITem).

Vzájemný vztah (prioritu zobrazení) mezi běžnými znaky na obrazovce a SPRITy upravuje registr 27. Jeho počáteční hodnota je 0, to znamená, že všechny SPRITy mají vyšší prioritu zobrazení než znaky. Znaky budou tedy SPRITy překryty.

Nastavíme-li však některý bit registru 27 na hodnotu 1, změníme prioritu zobrazení příslušného SPRITu. Tento SPRITE pak bude překryt ostatními znaky na obrazovce.

Ukážeme si to na následujícím příkladu.

```
POKE 53248+27,16
```

Tímto příkazem dosáhneme toho, že se SPRITE 4 bude objevovat za ostatními znaky na obrazovce.

8. Zvuky

8.1. Vytváření zvuků a hudby

Počítač COMMODORE 64 může svými akustickými možnostmi směle konkurovat elektronickým syntetizátorům. Stejně jako pro jemnou grafiku, má COMMODORE 64 i pro generování zvuku speciální obvod. Tento obvod se nazývá Sound Interface Device (zkráceně SID).

SID obsahuje 29 osmibitových registrů, ze kterých je 25 určeno pro zápis dat. Data pro generování zvuků se do registrů zapisují podobně, jako tomu bylo u registrů VIC. Adresa registru 0 je 54272 a adresa nejvyššího registru je tedy 54272 + 24 = 54296.

Abychom mohli vytvořit zvuk přesně podle svých představ, musíme počítači zadat všechny následující parametry:

1. hlasitost zvuku
2. číslo generátoru
3. tvar generovaných vln
4. kmitočet
5. tvar obalové křivky tónu

Jak a proč je potřeba všechny tyto parametry zadat si nyní rozebereme podrobněji.

Nastavení hlasitosti

Váš COMMODORE 64 může generovat zvuk v šestnácti úrovních hlasitosti. Informace o nastavení hlasitosti se zapisuje do dolních čtyř bitů registru 24. Zapíšeme-li tedy do tohoto registru 0, bude hlasitost nejmenší a naopak, zapíšeme-li hodnotu 15 (binárně xxxx1111) bude hlasitost nejvyšší.

Nastavení hlasu

COMMODORE 64 má pro generování zvuků k dispozici tři nezávislé tónové generátory. K programování těchto tří generátorů používáme prvních 21 registrů.

1. hlas registry 0 až 6
2. hlas registry 7 až 13
3. hlas registry 14 až 20

U každého generátoru můžeme nastavit kmitočet tónu, šířku pulsu, tvar vlny a obalovou křivku tónu. Pro zápis těchto údajů používáme následujících registrů:

1. hlas	2. hlas	3. hlas	význam registru
0	7	14	kmitočet tónu (dolní byte)
1	8	15	kmitočet tónu (horní byte)
2	9	16	šířka pulsu (dolní byte)
3	10	17	šířka pulsu (horní byte)
4	11	18	řídící registr
5	12	19	nastavení obalové křivky tónu
6	13	20	nastavení obalové křivky tónu

Nastavení kmitočtu tónu

Kmitočet každého generovaného tónu se může pohybovat v rozsahu od 0 do 65000. Jak víme, může jeden osmibitový registr obsahovat hodnoty od 0 do 255, proto musíme pro zápis kmitočtu použít dvou registrů, čímž se nám rozsah rozšíří od 0 do 65535 (binárně 1111111111111111). Do prvního registru zapisujeme hodnotu dolních (méně mocných) osmi bitů a do druhého registru zapisujeme hodnotu horních (mocnějších) osmi bitů. Mluvíme pak o tzv. dvoubytovém (čti dvoubajtovém) uložení systémem LO-HI (low = dolní, high = horní).

Chceme-li tedy navolit pro první hlas kmitočet F, použijeme příkazů:

```
POKE 54272+1,HI … kde HI=INT(F/256)
POKE 54272,LO … kde LO=F-(256*HI)
```

Nastavení šířky pulsu

Do následujících dvou registrů se zapisuje šířka pulsu. Je to v podstatě poměr mezi zapnutím a vypnutím u pravoúhlého signálu. Pro zápis do těchto dvou registrů platí stejná pravidla jako pro zápis kmitočtu.

Registr řízení

V prvním bitu tohoto registru je uložena nejdůležitější informace, a sice informace o tom, zda je příslušný hlas zapnut, či nikoliv. Proto se v programu zadávají data do tohoto registru jako poslední, poté co jsou již ostatní parametry nastaveny.

Kromě toho obsahuje řídící registr informaci o tom, jaký tvar vlny má být generován. SID může generovat signál skládající se z trojúhelníkových, pilových, pravoúhlých nebo šumových vln.

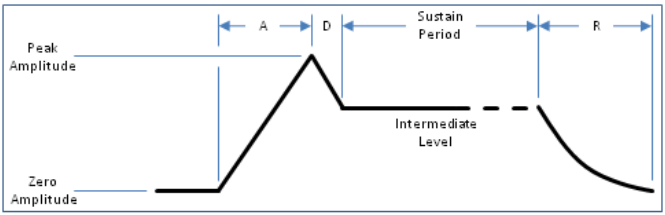
bit	význam
0	zapnutí hlasu
1	nevyužit
2	nevyužit
3	nevyužit
4	trojúhelníková vlna
5	pilová vlna
6	pravoúhlá vlna
7	šumová vlna

Chceme-li tedy zapnout 2. hlas a zvolit pro něj signál s pilovým průběhem, použijeme příkazu:

```
POKE 54272+11,32+1
```

Volba obalové křivky tónu

Obalová křivka tónu se skládá ze čtyř částí. Grafické znázornění těchto částí najdeme na obrázku. První část označujeme jako nasazení (A = attack), druhou část jako útlum (D = decay), třetí jako výdrž (S = sustain) a čtvrtou jako dozvuk (R = release). Tyto čtyři parametry tónu, které určují jeho zabarvení, jsou uloženy v další dvojici registrů.



Každý z uvedených parametrů může nabývat hodnot od 0 do 15. Pro každý jsou tedy rezervovány čtyři bity příslušného registru.

Pro první hlas tedy platí:

registr 5	bit 0 – 3	nasazení
registr 5	bit 4 – 7	útlum
registr 6	bit 0 – 3	výdrž
registr 6	bit 4 – 7	dozvuk

Obdobně toto platí i pro druhý a třetí hlas. Chceme-li tedy zvolit čas nasazení pro druhý hlas 5, útlum 3, výdrž 12 a dozvuk 7, použijeme pro zápis do registrů následující příkazy:

```
POKE 54272+12,5+48 (binárně 0011 0101)
POKE 54272+13,12+112 (binárně 0111 1100)
```

Nastavení filtrů

Další možností jak modulovat zvuk generovaný naší počítačem COMMODORRE 64 je použití filtrů. Akustický filtr je zařízení, které propouští jen některé kmitočty a ostatní potlačuje.

SID má zabudovány tři typy filtrů:

1. HIGH-PASS FILTR
   Propouští kmitočty vyšší než je nastavená frekvence a potlačuje kmitočty nižší.
2. LOW-PASS FILTR
   Propouští kmitočty nižší než je nastavená frekvence a potlačuje kmitočty vyšší.
3. BAND-PASS FILTR
   Propouští pouze úzké pásmo kmitočtů kolem nastavené frekvence, ostatní kmitočty potlačuje.

Mezní frekvence, podle které všechny filtry pracují, ze zadává do registrů 21 a 22 jako jedenáctibitové číslo. Jeho hodnota se může pohybovat v rozsahu od 0 do 2047.

Obsazení registrů

registr	bit	význam
21	0-2	bity 0-2 mezní frekvence
21	3-7	nevyužity
22	0-7	bity 3-10 mezní frekvence
23	0	filtr pro hlas 1
23	1	filtr pro hlas 2
23	2	filtr pro hlas 3
23	3	filtr externí vstup
23	4-7	filtr ozvěny (2-15)
24	0-3	nastavení hlasitosti
24	4	LOW-PASS filtr
24	5	BAND-PASS filtr
24	6	HIGH-PASS filtr

Podrobně je popsáno obsazení jednotlivých bitů registrů SIDu popsáno v příloze originálního manuálu.

8.2. Příklad hudebního programu

Na závěr této kapitoly vytvoříme program, podle kterého zahraje počítač stupnici, a na kterém si v praxi ukážeme programování jednotlivých registrů SIDu.

```
     NEW

5    REM STUPNICE
7    FOR L=54272 TO 54296: POKE L,0: NEXT
10   POKE 54296,15
20   POKE 54277,7: POKE 54278,133
50   READ A
55   IF A=-1 THEN END
60   READ B
80   POKE 54273,A: POKE 54272,B
85   POKE 54276,17
90   FOR T=0 TO 250: NEXT: POKE 54276,16
95   FOR T=0 TO 50: NEXT
100  GOTO 20
110  DATA 16,195,18,209,21,31,22,96
120  DATA 25,30,28,49,31,165,33,135
999  DATA -1
```

1. Nejprve vymažeme paměť počítače příkazem NEW.
2. V řádku 7 zapíšeme ve smyčce FOR – NEXT do všech registrů SIDu hodnotu 0.

3. V řádku 10 jsme zapsali do registru 34 hodnotu 15, čímž jsme nastavili hlasitost na maximum.
4. V řádku 20 jsme zapsali do registrů 5 a 6 hodnoty 7 a 133 (binárně 00000111 a 10000101). Tímto jsme nastavili pro obalovou křivku tónu hodnoty:
   nasazení 0
   útlum 7
   výdrž 8
   dozvuk 5
5. V řádku 50 přiřadíme proměnné A hodnotu, která je uvedena na prvním místě v datovém řádku 110.
6. V řádku 55 otestujeme proměnnou A. Je-li její hodnota rovna -1, program je ukončen příkazem END.
7. V řádku 60 přiřadíme proměnné B druhou hodnotu v datovém řádku 110.
8. V řádku 80 zapíšeme do registru 0 hodnotu proměnné B a do registru 1 hodnotu proměnné A. Proměnná A představuje v našem případě horních osm bitů hodnoty kmitočtu a proměnná B dolních osm bitů (binárně 00010000 11000011). Skutečnou hodnotu vkládaného kmitočtu tedy zjistíme následujícím výpočtem:
   256 * A + B = 256 * 16 + 195 = 4291
   Podle přílohy originální příručky zjistíme, že tento kmitočet odpovídá notě C.
9. V řádku 85 zapíšeme do řídícího registru 4 hodnotu 17 (binárně 00010001). Tím jednak zvolíme tvar signálu (trojúhelníkové vlny) a jednak zapneme hlas 1.
10. Smyčka FOR – NEXT v řádku 90 udává dobu znění tónu. Po skončení této smyčky se do řídícího registru zapíše hodnota 16 (binárně 00010000). Tvar signálu zůstává navolen, ale vypne se 1. hlas.
11. Smyčka v řádku 95 představuje prodlevu mezi jednotlivými tóny.
12. Podle příkazu v řádku 100 se program vrací zpět na řádek 20 a probíhá znovu. V řádcích 50 a 60 se však proměnným A a B přiřazují další hodnoty z datových řádků. Program končí, načte-li se do proměnné A hodnota -1 z datového řádku 999.

Nyní zkusíme upravit v programu některé hodnoty registrů a pozorujte, jak se bude měnit zvuk generovaný naším počítačem:

```
85 POKE 54276,33
90 FOR T=0 TO 250: NEXT: POKE 54276,32
```

13. V řádku 85 jsme zapsali do řídícího registru 4 hodnotu 33 (binárně 00100001). Tím jsme změnili tvar signálu na pilový a zapnuli 1. hlas.

```
20 POKE 54277,3: POKE 54278,0
```

14. Touto změnou v řádku 20 jsme změnili tvar obalové křivky tónu. Zcela jsme potlačili nasazení, výdrž a dozvuk, pouze útlum je nastaven na hodnotu 3. Nyní má náš počítač zvuk podobný zvuku banja.

V anglické verzi návodu je uvedeno ještě několik dalších příkladů, podle kterých můžeme vytvořit programy generující hudbu nebo různé zvukové efekty. Programy jsou sestaveny podobně, jako výše uvedený program, rozdíl je pouze v hodnotách, které zapisujeme do jednotlivých registrů.

9. Manipulace s daty

9.1. READ a DATA

Dosud jsme poznali dva způsoby, jakými lze přiřadit určité proměnné její hodnotu. Jednak přímým přiřazením v programu (A = 2) a jednak pomocí příkazů vstupu (INPUT a GET).

V některých případech, zvláště při zpracování většího počtu dat, je ale výhodnější použít příkazu READ.

```
10 READ X
20 PRINT "X JE NYNI:";X
30 GOTO 10
40 DATA 1,34,10.5,16,234.56
```

Po spuštění programu příkazem RUN se na obrazovce objeví:

```
X JE NYNÍ: 1
X JE NYNÍ: 34
X JE NYNÍ: 10.5
X JE NYNÍ: 16
X JE NYNÍ: 234.56

? OUT OF DATA ERROR IN 10
READY
```

Podívejme se, jak program probíhá. V řádku 10 se proměnné X přiřadí hodnota, která je uvedená jako první v datovém řádku 40. Proměnná X má nyní hodnotu 1 a ukazatel (POINTER) ukazující na dosud nepřečtená data se posune o jedno místo doprava. V řádku 20 se provede výpis a program se vrátí na řádek 10. Program probíhá znovu, proměnné X je přiřazena hodnota 34 atd.

Program končí výpisem chybového hlášení v okamžiku, kdy požadujeme čtení dalších data a ukazatel je již na konci datového řádku. Předejít této chybě můžeme následujícími způsoby:

1. čtením dat ve smyčce FOR - NEXT

```
10 FOR X=1 TO 3
15 READ A$
20 PRINT "A$ JE NYNI:";A$
30 NEXT
40 DATA ZDRAVI,VAS,COMMODORE 64
```

2. označením konce dat značkou

```
10 READ A
15 IF A<0 THEN END
20 DATA 13,35,29,-999
25 PRINT "DATA =";A
30 GOTO 10
```

Značkou (FLAGem) se v tomto případě rozumí znak nebo číslo, které se v našem souboru dat nemůže vyskytnout a které lze snadno testovat příkazem IF – THEN.

Jak jste si všimli, můžeme příkazem RAED číst nejen číselné, ale i řetězcové proměnné. Záleží pouze na typu proměnné, které mají být data přiřazena.

Pravidla pro zápis datových řádek:

1. Datové řádky mohou obsahovat:
   - čísla celá
   - reálná čísla
   - čísla zapsaná ve vědecké notaci
   - znakové řetězce
2. datové řádky nesmí obsahovat:
   - proměnné
   - aritmetické operace
3. Jednotlivá data jsou od sebe oddělena čárkou. (Pro zápis desetinných čísel používáme zásadně desetinné tečky.)
4. Datový řádek musíme vždy označit instrukcí DATA. Takto označený řádek může být zapsán na kterémkoliv místě programu. (Zpravidla se datové řádky umisťují na konec programu.)

Chceme-li několikrát přečíst příkazem READ stejná data, můžeme vrátit ukazatel (POINTER) na začátek datové řádky příkazem RESTORE. Vyzkoušejte si to na příkladu:

```
10 FOR X=1 TO 3
15 READ A$
20 PRINT "A$ JE NYNI:";A$
30 NEXT
40 DATA ZDRAVI,VAS,COMMODORE 64
45 PRINT: RESTORE: GOTO 10
```

9.2. Výpočet střední hodnoty

Praktické využití příkazu READ si ukážeme na následujícím příkladu, kde z čísel uvedených v datovém řádku počítač určí střední hodnotu.

```
5  T=0: CT=0
10 READ X
20 IF X=-1 THEN 50
25 CT=CT+1
30 T=T+X
40 GOTO 10
50 PRINT "BYLO NACTENO";CT;"HODNOT."
60 PRINT "CELKOVY SOUCET JE";T
70 PRINT "STREDNI HODNOTA JE";T/CT
80 DATA 75,80,62,91,87,93,78,-1
```

```
RUN
BYLO NACTENO 7 HODNOT
CELKOVY SOUCET JE 566
STREDNI HODNOTA JE 80.8571429
```

Program si jako vždy rozebereme:

1. V řádku 5 jsme vynulovali obě používané proměnné T a CT.
2. V řádku 10 je proměnné X přiřazena hodnota z datového řádku.
3. V řádku 20 otestuje, zda X neobsahuje značku (FLAG) označující konec dat.
4. V řádku 25 se zvýší o jednu hodnota čítače počtu dat CT.
5. V řádku 30 se hodnota proměnné X přičte k hodnotě proměnné T, ve které je uložen celkový součet.
6. Program opakuje postup do té doby, než je do proměnné X načtena hodnota -1.
7. Obsahuje-li proměnná X hodnotu -1 (značka konce dat), program postoupí podle příkazu IF – THEN v řádku 20 na řádek 50.
8. Podle příkazu v řádku 50 se na obrazovku vypíše počet dat, ze kterých se střední hodnota počítala.
9. Podle příkazu v řádku 60 se vypíše celkový součet dat.
10. V řádku 70 se spočítá a vypíše střední hodnota dat.

9.3. Indexované proměnné

Až dosud jsme používali pouze jednoduché proměnné, jejichž jméno se skládalo z jednoho znaku nebo z kombinace dvou a více znaků.

Ve složitějších programech je však výhodnější používat tzv. indexovaných proměnných. Jméno indexované proměnné se skládá ze jména, pro které platí stejná pravidla jako pro jméno jednoduché proměnné, a z indexu, který se uvádí v závorce.

```
A(1)       A = jméno
           (1) = index
```

Index proměnné nemusí vždy obsahovat pouze číslo, ale může obsahovat aritmetický výraz nebo jinou proměnnou.

Příklady jmen indexovaných proměnných:

S(1), CT(9), A0(N), B$(1), JMENO$(A*3), ABC$(X+4) apod.

Abychom mohli pochopit, co to vlastně indexované proměnné jsou a jak se jich používá, musíme si objasnit některé nové pojmy.

Když jsme s proměnnými začali pracovat, řekli jsme si, že je možné představit si každou proměnnou jako zásuvku, do které můžeme ukládat různé hodnoty. Představme si nyní, že máme těchto zásuvek několik pod sebou. Tomuto sloupci zásuvek budeme říkat POLE (ARRAY). Nejvyšší zásuvka bude mít např. jméno A(0) a poslední A(n). Takovému poli, ve kterém jsou uloženy proměnné s jedním indexem, říkáme pole jednorozměrné.

Jak využít indexovaných proměnných v praxi si ukážeme na následujícím programu. Opět budeme určovat střední hodnotu několika čísel.

```
5  PRINT CHR$(147)
10 INPUT "KOLIK CISEL:";X
20 FOR A=1 TO X
30 PRINT "VSTUP CISLO";A;:INPUT B(A)
40 NEXT
50 SU=0
60 FOR A=1 TO X
70 SU=SU+B(A)
80 NEXT
90 PRINT: PRINT "STREDNI HODNOTA JE";SU/X
RUN

KOLIK CISEL: ? 5
```

```
VSTUP CISLO 1 ? 125
VSTUP CISLO 1 ? 167
VSTUP CISLO 1 ? 189
VSTUP CISLO 1 ? 167
VSTUP CISLO 1 ? 158

STREDNI HODNOTA JE 161.2
```

Podívejme se opět, jak tento program probíhal:

1. Příkazem v řádku 5 jsme vymazali obrazovku.
2. V řádku 10 jsme pomocí příkazu INPUT zadali počet čísel, ze kterých se střední hodnota bude počítat. Tato hodnota je uložena v proměnné X.
3. Počet průchodů smyčkou FOR – NEXT v řádcích 20 až 40 je určen proměnnou X.
4. V řádku 30 se indexované proměnné B(A) přiřadí postupně jednotlivá čísla je zpracování. Hodnota A je závislá na počtu průchodů smyčkou FOR – NEXT. Po pátém průchodu bude mít tedy indexovaná proměnná B(A) tyto hodnoty:
   B(1) 125
   B(2) 167
   B(3) 189
   B(4) 167
   B(5) 158
5. V řádku 50 se vynuluje proměnná SU.
6. V řádcích 60 až 80 se postupně ve smyčce sečtou všechna zadaná čísla.
7. V řádku 90 se vypočte a vytiskne požadovaná střední hodnota.

9.4. Dimenzování polí

Použijeme-li jakoukoliv indexovanou proměnnou, rezervuje si počítač v paměti pole pro jedenáct hodnot. Lze tedy použít indexovanou proměnnou s indexem od 0 do 10.

Chceme-li uložit do jedné proměnné více hodnot, musíme předem dimenzovat velikost pole, které si má počítač v paměti rezervovat. Budeme-li například používat proměnnou A s indexy od 0 do 25, musíme předem zadat příkaz:

```
DIM A(25)
```

Budeme-li v programu využívat více indexovaných proměnných, lze jejich dimenzování provést najednou příkazem:

```
DIM A(25), B(16), A$(122)
```

9.5. Simulace hry v kostky

Některé složité programy lze použitím indexovaných proměnných zjednodušit a zkrátit. Následující ukázkový program simuluje hru v kostky a indexovaných proměnných využívá k registraci toho, kolikrát padlo které číslo.

```
1  PRINT CHR$(147)
10 INPUT "POČET HODU:";X
20 FOR L=1 TO X
30 R=INT(6*RND(1))+1
40 F(R)=F(R)+1
50 NEXT L
60 PRINT "CISLO","POČET HODU"
70 FOR C=1 TO 6: PRINT C,F(C):NEXT
```

Jak tento program funguje?

1. V řádku 1 se vymaže obrazovka.
2. V řádku 10 zadáme pomocí příkazu INPUT počet hodů. Tato hodnota se přiřadí proměnné X.
3. V následujících řádkách se ve smyčce X krát generuje náhodné číslo od 1 do 6. Podle toho, které číslo bylo vygenerováno, se zvýší o 1 jedna z proměnných F(1) až F(6).
4. Na závěr se vypíše celková bilance hodů, která bude vypadat takto:

```
POCET HODU:? 1000
CISLO   POCET HODU
1       148
2       176
3       178
4       166
5       163
6       169
```

V anglické verzi návodu najdete program, který má stejnou funkci jako program předešlý, ale nepoužívá indexované proměnné. Srovnejte, o kolik je delší a složitější.

9.6. Dvojrozměrná pole

Když jsme hovořili o indexovaných proměnných, představovali jsme si jednorozměrné pole jako sloupec zásuvek. Představíme-li si několik takových sloupců vedle sebe, vznikne pole dvojrozměrné. Určitou zásuvku pak musíme označit jak číslem sloupce, tak i číslem řady. Musíme tedy použít dvou indexů. Zápis takové proměnné pak vypadá asi takto:

```
A(4,6)     A = jméno proměnné
           4 = index udávající řádek
           6 = index udávající sloupec
```

Pro dvojrozměrné pole platí stejná pravidla i dimenzování jako pro pole jednorozměrné. Dvojrozměrné pole dimenzujeme například příkazem:

```
DIM A(20,30)
```

Abychom získali představu o tom, jakým způsobem lze dvojrozměrná pole využívat v praxi, prostudujte si následující program. V tomto programu se registrují odpovědi na čtyři otázky. Na závěr programu se vyhodnotí, kolikrát bylo na kterou otázku odpovězeno ANO, kolikrát NE a kolikrát SNAD.

```
10  REM OTAZKY
20  PRINT "<SHIFT><CLR/HOME>"
30  FOR R=1 TO 4
50  PRINT "OTAZKA CISLO:";R
60  PRINT "1-ANO 2-NE 3-SNAD"
61  GET C: IF C<1 OR C>3 THEN 61
65  PRINT C: PRINT
70  A(R,C)=A(R,C)+1
80  NEXT R
85  PRINT
90  PRINT"CHCETE ODPOVIDAT ZNOVU? (Y/N) ": PRINT
100 GET A$: IF A$="" THEN 100
110 IF A$="Y" THEN 20
120 IF A$<>"N" THEN 100
130 PRINT "<SHIFT><CLR/HOME>";
131 PRINT "ZAVERECNE VYHODNOCENI ODPOVEDI": PRINT
140 PRINT SPC(18); "ODPOVEDI"
141 PRINT "OTAZKA","ANO","NE","SNAD"
142 PRINT "------   -------------------------"
150 FOR R=1 TO 4
```

```
160 PRINT R,A(R,1),A(R,2),A(R,3)
170 NEXT R
```

1. V řádcích 30 až 65 se napíše číslo otázky a příkazem GET se proměnné C přiřadí číslo odpovědi.
2. V řádku 70 se zvýší o 1 jedna z proměnných A(1,1) až A(4,3) dvojrozměrného pole.
3. V řádcích 90 až 120 je testována odpověď na otázku, zda budeme v programu pokračovat či nikoliv. Rozhodneme-li se pokračovat, vrací se program zpět na řádek 20.
4. V řádcích 130 až 170 jsou příkazy pro závěrečné zhodnocení a výpis počtu odpovědí.

Spustíme-li program příkazem RUN, objeví se na obrazovce:

```
OTAZKA CISLO:1
1-ANO 2-NE 3-SNAD
VASE ODPOVED:1

OTAZKA CISLO:2
1-ANO 2-NE 3-SNAD
VASE ODPOVED:1

OTAZKA CISLO:3
1-ANO 2-NE 3-SNAD
VASE ODPOVED:2

OTAZKA CISLO:4
.
.
```

Podobný výpis se bude opakovat, dokud nebudeme chtít závěrečné zhodnocení. Pak se obrazovka smaže a objeví se výpis:

```
ZAVERCNE VYHODNOCENI ODPOVEDI

            ODPOVED
OTAZKA   ANO   NE    SNAD
-------------------------
1        6     1     0
2        5     2     0
3        7     0     0
4        2     4     1
```

Přílohy

A. Kódy ASCII A CHR$

Tato příloha ukazuje pro všechna X, jaký znak se objeví na obrazovce, když pošlete PRINT CHR$(X). Naopak můžete také zjistit výsledek pro PRINT ASC("x"), přičemž "x" odpovídá příslušné stisknuté klávese. To je zvláště potřebné, aby se zjistil znak přijatý v nějakém výrazu GET, aby se přepnulo mezi velkým a malým písmem a aby se poslaly řídicí znaky, které nelze tisknout jako například přepnutí velkého a malého písma, které nemohou být v nějakém stringu uzavřeny do uvozovek.

Znak	CHR$	Znak	CHR$	Znak	CHR$	Znak	CHR$
	0	CRSR ▼	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
WHT	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
blokovaný SHIFT CTRL	8		25	*	42	;	59
otevřený SHIFT CTRL	9		26	+	43	<	60
	10		27	,	44	=	61
	11	RED	28	-	45	>	62
	12	CRSR →	29	.	46	?	63
RETURN	13	GRN	30	/	47	@	64
PŘEPNUTÍ NA MALÁ PÍSMENA	14	BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

Znak	CHR$	Znak	CHR$	Znak	CHR$	Znak	CHR$
D	68		97		126		155
E	69		98		127		156
F	70		99		128		157
G	71		100		129		158
H	72		101		130		159
I	73		102		131		160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	f8	140		169
S	83		112		141		170
T	84		113		142		171
U	85		114		143		172
V	86		115		144		173
W	87		116		145		174
X	88		117		146		175
Y	89		118		147		176
Z	90		119		148		177
[	91		120		149		178
£	92		121		150		179
]	93		122		151		180
↑	94		123		152		181
←	95		124		153		182
	96		125		154		183

Znak	CHR$	Znak	CHR$	Znak	CHR$	Znak	CHR$
	184		186		188		190
	185		187		189		191

Kódy	**192-233**	**jako kódy**	**96-127**
Kódy	**224-254**	**jako kódy**	**160-190**
Kódy	**255**	**jako kódy**	**126**

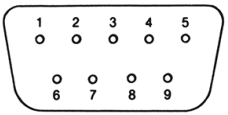
B. Zapojení vstupně/výstupních konektorů

V této příloze je popsáno zapojení konektorů C64:

1. Herní konektor (Game Port)
2. Slot na cartidge (Cartidge Slot)
3. Audio/Video konektor (Audio/Video Connector)
4. Sériový konektor (Serial Port)
5. TV konektor (TV Connector)
6. Kazetové rozhraní (Cassette Interface)
7. Uživatelský konektor (User Port)
8. Napájecí konektor (Power socket)

Herní konektor 1

Pin	Typ	Poznámka
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY	
6	BUTTON A/LP	
7	+5V	Max. 100mA
8	GND	
9	POT AX	



Herní konektor 2

Pin	Typ	Poznámka
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY	
6	BUTTON B	
7	+5V	Max. 100mA
8	GND	
9	POT BX	

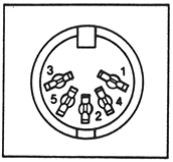
Slot na cartidge

Pin	Typ	Pin	Typ
1	GND	A	GND
2	+5V	B	/ROMH
3	+5V	C	/RESET
4	/IRQ	D	/NMI
5	/CR/W	E	02
6	Dot Clock	F	A15
7	I/O 1	H	A14
8	/GAME	J	A13
9	/EXROM	K	A12
10	I/O 2	L	A11
11	/ROML	M	A10
12	BA	N	A9
13	/DMA	P	A8
14	D7	R	A7
15	D6	S	A6
16	D5	T	A5
17	D4	U	A4
18	D3	V	A3
19	D2	W	A2
20	D1	X	A1
21	D0	Y	A0
22	GND	Z	GND

Audio/Video konektor

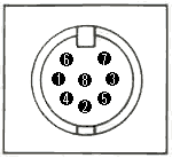
Varianta s 5-pinovým konektorem

Pin	Typ
1	LUMINANCE
2	GND
3	AUDIO OUT
4	VIDEO OUT
5	AUDIO IN



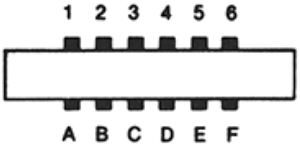
Varianta s 8-pinovým konektorem

Pin	Typ
1	AUDIO OUT
2	GND
3	LUMINANCE
4	AUDIO IN
5	VIDEO OUT
6	NC
7	NC
8	CHROMA OUT



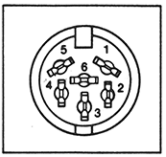
Kazetové rozhraní

Pin	Typ
A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SENSE

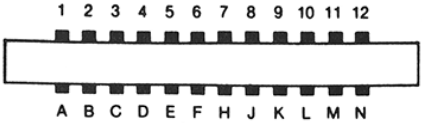


Sériový konektor

Pin	Typ
1	SERIAL /SRQIN
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	/RESET



Uživatelský konektor



Pin	Typ	Poznámka
1	GND	
2	+5V	MAX. 100mA
3	/RESET	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	/PC2	
9	SER. ATN OUT	
10	~9V	MAX. 100mA
11	~9V	MAX. 100mA
12	GND	

Pin	Typ	Poznámka
A	GND	
B	/FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	

Napájecí konektor

Pin	Typ
1	GND
2	GND
3	GND
4	+5V
5	+5V
6	~9V
7	~9V

Pozn: Ačkoliv jsou aktuálně použity jen 4 piny, napájecí konektor je 7-pinový DIN. Je použit pin 5 pro +5V, pin 2 pro GND a na pinech 6 a 7 je ~9V.

Verze z 26.7.2011
Skenování a úprava textu: Solaris104
Skenování a úprava obrázků: Jack
www.C64.cz