Osobní počítač

128

128D

Systémový manuál

O B S A H

Systémový manuál C 128

KAPITOLA 1

ÚVOD

Průvodce manuálem

Tento systémový manuál Commodoru 128 byl sestaven, aby umožnil uživateli využít plně pokročilé funkce počítače Commodore 128. Dále vysvětlíme, jak manuál používat.

Dříve než budete pokračovat ve čtení tohoto Systémového manuálu doporučujeme vám přečíst si druhou knížku, dodávanou s počítačem, Základní průvodce osobního počítače Commodore 128, která obsahuje důležité informace jak připravit a uvést do chodu Commodore 128.

Těm, kteří mají zájem hlavně o využití jazyka BASIC při sestavování a využívání vlastních programů, doporučujeme přečíst nejprve Kapitolu 2. - Využití modu C 128. Tato kapitola je úvodem do programovacího jazyka BASIC, používaného v modech C 128 a C 64; obsahuje pokyny, jak využívat příkazy v modech C 128 a C 64, popisuje klávesnici Commodoru 128, vysvětluje jak využívat numerické a řídící příkazy BASICu (včetně příkazů pro barvy, grafiku a zvuk), používané pouze v modu C 128.

O možnostech využití BASICu v modu C 64 informuje Kapitola 3 - Využití modu C 64.

O používání CP/M v Commodoru 128 čtěte v Kapitole 4 - Využití modu CP/M. V této kapitole se vysvětluje, jak uvést do chodu a používat CP/M v Commodoru 128. V tomto modu lze používat tisíce softvérových prostředků tvořících řadu PERFECT (PERFECT WRITER, PERFECT CALC, PERFECT FILER) a rovněž vytvářet vlastní programy v CP/M.

Další podrobnosti o příkazech BASIC 7.0 naleznete v kapitole 5., Encyklopedie BASICu 7.0. Tato kapitola obsahuje informace o tvaru a o použití všech příkazů, instrukcí a funkcí BASIC 7.0.

Jestliže i po pročtení Kapitol 1.-5. potřebujete další technické informace o speciálních otázkách a záležitostech týkajících se Commodoru 128, obraťte se k Dodatkům tohoto Systémového manuálu. Dodatky obsahují mnoho informací, například seznam chybových hlášení BASICu a DOSu a souhrn příkazů pro disk. Bezprostředně za dodatky naleznete "Slovníček", obsahující některé definice termínů z oblasti informatiky.

Doplňující technické informace charakterizující Commodore 128 naleznete v "Manuálu programátora Commodoru 128".

Kapitola 2

POUŽITÍ MODU C 128

ČÁST 2

Všeobecné informace o osobním počítači Commodore 128

Všeobecné informace o osobním počítači Commodore 128

Osobní počítač Commodore 128 nabízí tři základní funkční mody:

- mod C 128
- mod C 64
- mod CP/M

Uvedeme charakteristiku každého modu:

Mod C 128

V modu C 128 osobní počítač Commodore 128 umožňuje přístup k 128 K RAM a k účinnému jazyku BASIC, nazývanému BASIC 7.0. Basic 7.0 disponuje více než 140 příkazy, instrukcemi a funkcemi. Mod C 128 nabízí rovněž možnost pracovat se 40 nebo 80 sloupci, s jedinou klávesnicí s 92 tlačítky. Zabudovaný monitor strojového kodu umožňuje vytvářet a používat vlastní programy ve strojovém kódu. V modu C 128 lze používat řadu nových periferijních zařízení Commodoru, včetně nového rychlého sériového diskdrajvu (model 1570), "myš" a videomonitor RGBI (mod. 1901) s 40/80 sloupci a navích všechny normální sériové periferie Commodoru.

Mod C 64

V modu C 64 funguje Commodore 128 přesně stejně jako počítač Commodore 64, umožňujíc používat

širokou paletu programů Commodoru 64. Mimoto je zcela kompatibilní s periferiemi Commodoru 64.
Mod C 64 nabízí jazyk BASIC 2.0, výstup 40 sloupců a přístup k 64 k RAM.

Mod CP/M

V modu CP/M je zabudovaný mikrpprocesor Z80 vybavený všemi charakteristikami CP/M, vypracovaného Digital Research, verze 3.0, dodatečně k dalším kapacitám, poskytovaným Commodorem. Softvér CP/M Commodoru 128, nyzývaný CP/M Plus, má k dispozici 128 K paměti RAM, 40 a 80 sloupců výstupu, přístup k celé klávesnici, numerické klávesničce a k specialním tlačítkům, přístup k novému rychlému seriovému disku 1570 a jiným standartním sériovým periferiím.
Kapitoly 2., 3. a 4., obsahující části od 3. do 15., uvádějí informace o přístupu a použití všech tří možných mnohostranných operačních modů osobního počítače Commodore 128.

Přechod z jednoho modu do jiného

Následující tabulka ukazuje, jak přecházet z jednoho modu do jiného

pnuto - C 128, 40 sl.

. Zkontrolujte, zda je tlačítko 40/80 v horní poloze

2. Zapněte počítač

C 128, 80 sl. - C 128, 40 sl.

1. Stiskněte tlačítko ESC.
2. Stiskněte tlačítko X

NEBO

1. Prověřte zda tlačítko 40/80 je nahoře
2. Stiskněte tlačítko RESET

C 64 - C 128, 40 sl.

1. Prověřte zda je tlačítko 40/80 nahoře
2. Vypněte a zapněte počítač

CP/M, 40 sl. - C 128, 40 sl.

1. Prověřte, zda je tlačítko 40/80 nahoře
2. Vypněte a zapněte počítač

CP/M, 80 sl. - C 128, 40 sl.

1. Prověřte, zda je tlečítko 40/80 nahoře
2. Vypněte a zapněte počítač

Vypnuto - C 128, 80 sl.

1. Stiskněte tlačítko 40/80
2. Zapněte počítač

C 128, 40 sl. - C 128, 80 sl.

1. Stiskněte tlačítko ESC
2. Stiskněte tlačítko X

NEBO

1. Stiskněte tlačítko 40/80

2. Stiskněte tlačítko RESET

C 64 - C 128, 80 sl.

1. Stiskněte tlačítko 40/80
2. Vypněte a zapněte počítač

CP/M, 40 sl. - C 128, 80 sl.

1. Stiskněte tlačítko 40/80
2. Vyndejte systémový disk CP/M z drajvu, jestlize je to nutné
3. Vypněte a zapněte počítač

CP/M, 80 sl. - C 128, 80 sl.

1. stejně jako v předchozím případě

Vypnuto - C 64

1. Podržte tlačítko Cx
2. Zapněte počítač

NEBO

1. Zaveďte kasetu C 64
2. Zapněte počítač

C 128, 40 sl. - C 64

1. Napište GO 64, stiskněte tlačítko RETURN
2. Objeví se hlášení ARE YOU SURE? (jste si jistý?). Napište Y a stiskněte RETURN

C 128, 80 sl. - C 64

Tentýž postup

CP/M, 40 sl. - C 64

1. Vypněte počítač
2. Prověřte, zda je tlačítko 40/80 nahoře
3. Podržte tlačítko Cx a zapněte zároveň počítač

NEBO

1. Vypněte počítač
2. Zasuňte kasetu C 64
3. Zapněte počítač

CP/M, 80 sl. - C 64

Tentýž postup

Vypnuto - CP/M, 40 sl.

1. Zapněte diskdrajv
2. Založte systémový disk CP/M do drajvu
3. Prověřte, zda je tlačítko 40/80 nahoře
4. Zapněte počítač

C 128, 40 sl. - CP/M, 40 sl.

1. Zapněte diskdrajv
2. Založte systémový disk CP/M do drajvu
3. Prověřte, zda je tlačítko 40/80 nahoře
4. Napište BOOT
5. Stiskněte RETURN

C 128, 80 sl. - CP/M, 40 sl.

Tentýž postup

C 64 - CP/M, 40 sl.

1. Prověřte, zda je tlačítko 40/80 sl. nahoře
2. Zapněte drajv
3. Založte systémový disk CP/M do drajvu
4. Vypněte a zapněte počítač

CP/M, 80 sl. - CP/M, 40 sl.

1. Založte disk CP/M do drajvu
2. Po stisknutí A napište DEVICE CONOUT=40sl.
3. Stiskněte RETURN

Vypnuto - CP/M 80 sl

1. Zapněte diskdrajv
2. Založte systémový disk do drajvu
3. Stiskněte tlačítko 40/80
4. Zapněte počítač

C 128, 40 sl. - CP/M, 80 sl.

1. Zapněte diskdrajv
2. Zapožte systémový disk CP/M do drajvu
3. Stiskněte tlačítko 40/80
4. Napište BOOT
5. Stiskněte RETURN

C 128, 80 sl. - CP/M, 80 sl.

1. Zapněte diskdrajv
2. Zapožte systémový disk CP/M do drajvu
3. Zkontrolujte, zda je tlačítko 40/80 nahoře
4. Napište BOOT
5. Stiskněte RETURN

C 64 - CP/M, 80 sl.

1. Stiskněte tlačítko 40/80
2. Zapněte diskdrajv
3. Založte systémový disk CP/M do diskdrajvu
4. Vypněte a zapněte počítač

CP/M, 40 sl - CP/M, 80 sl

1. Vložte disk CP/M do drajvu
2. Po stisknutí A napište DEVICE CONOUT= 40 col.
3. Stiskněte RETURN

POZNÁMKA: Jestliže používáte duální monitor Commodore 1901, nezapomeňte přemístit vypínač video z SLOŽENÝ nebo ZVLÁŠTNÍ do RGBI jestliže přecházíte z 40 na 80 sloupců; proveďte tuto operaci naopak, jestliže si přejete přejít z 80 na 40 sloupců. Mimoto, při přechodu z jednoho modu do druhého, vyjměte kasetu z hnízda pro rozšíření, případně disk z drajvu

Č Á S T 3

Všeobecné informace o programování v Basicu

Programovací jazyk Basic

Programovací jazyk BASIC je speciální jazyk, který umožňuje dialog s Commodorem 128. BASIC je prostředek užívaný pro sdělení počítači, jaké operace má provést.

BASIC má vlastní slovník (sestávající z příkazů, instrukcí a funkcí) a vlastní sytaktická pravidla. Slovník a syttax BASICu lze použít pro vytváření souhrnů instrukcí, nazývaného programem, které by počítač byl schopný provést.

Prostřednictvím BASICu je možné komunikovat s Commodorem 128 dvěma způsoby: prostřednictvím programu nebo přímo (neprogramově).

Přímý mod

Po zapnutí je Commodore 128 okamžitě připraven přijímat příkazy BASIC v přímém modu. V tomto modu se příkazy natisknou na klavesnici a zavádějí se do počítače stisknutím tlačítka RETURN. Počítač provede celý daný příkaz v přímém modu okamžitě po stlačení tlačítka RETURN. Většina příkazů Commodoru 128 může být použita jak v přímém modu, tak uvnitř programu.

Programovací mod

V programovacím modu se zavádí řada příkazů pro řešení specifického problému. Každý příkaz patří do jednoho řádku programu. Každý

příkaz programu může sestávat maximálně ze 160 znaků, což je ekvivalentní čtyřem řádkům na obrazovce při formátu 40 sloupců, nebo dvěma řádkům na obrazovce s formátem 80 sloupců.

Po zavedení programu jej lze okamžitě použít, jestliže napíšeme příkaz RUN (běž) následovaný RETURN. Jinak je možné program zaznamenat na disk nebo na pásku pomocí příkazu DSAVE (nebo SAVE). Později je možné program opět přivolat z disku nebo pásky pomocí příkazu DLOAD (nebo LOAD). Tento příkaz program z disku nebo z pásky zkopíruje a přenese do paměti Commodoru 128. Program je pak možne znovu provést příkazem RUN. Všechny tyto příkazy budou podrobně vysvětleny později v této části manuálu. Počítač se téměř vždy používá s programem, buď osobně vytvořeným nebo již hotovým k použití. Unikalní cena možnosti pracovat v přímém modu se vyjeví při potřebě změnit program pomocí příkazů LIST, LOAD, SAVE a RUN. Jediným rozdílem přímého a programovacího modu je, že v přímém modu před příkazem nemusí být číslo řádku.

Používání klávesnice

Na následující fotografii je ukázána klávesnice Commodoru 128

BASIC je stejný v modu C 64 a Modu C 128. Většina tlačítek a příkazů se používá pro programování v obou modech. Tlačítka označená na obrázku stínovaně se mohou používat v modu C 64. V modu C 128 se používají všechna tlačítka klávesnice.

Znaky na klávesnici

Klávesnice Commodoru 128 nabízí dva soubory různých znaků:

- velká písmena a grafické znaky
- velká a malá písmena

Ve formátu 80 sloupců jsou oba soubory k dispozici současně. Lze tedy zobrazit na obrazovce 512 různých znaků. S formátem 40 sloupců lze používat současně pouze jediný soubor znaků. Po zapnutí Commodoru 128 s gormátem 40 sloupců je klávesnice připravena pro soubor znaků velká písmena/grafické znaky, takže všechna tištěná písmena budou velká. Abyste přešli od jednoho souboru znaků k druhému, stiskněte tlačítko SHIFT současně s tlačítkem Cx (tlačítko COMMODORE). Zkuste použít oba soubory znaků: zapněte počítač a natiskněte

nějaká písmena nebo grafické znaky, potom stiskněte tlačítko SHIFT a Cx (Commodore). Na obrazovce zaznamenáme změnu na velká a malá písmena. Stiskněte znovu SHIFT a Cx, abyste se vrátili k souboru znaků velká písmena/grafické znaky.

Použití příkazových tlačítek

Příkazová tlačítka jsou ta, kterými posíláme zprávy počítači. Některá příkazová tlačítka (jako tlačítko RETURN) se používají samotná, kdežto jiná (jako SHIFT, CTRL, Cx a RESTORE) se používají v kombinaci s jinými tlačítky. Nyní vysvětlíme použití každého tlačítka. Tlačítka používaná v modu C 128 budou popsána v části 5.

Return

Stisknutím tlačítka RETURN (návrat) dosáhneme toho, že vytištěné znaky jsou zaslány do paměti Commodoru. Stlačení tohoto tlačítka rovněž způsobí přemístění kursoru (malý blikající čtvereček, který ukazuje, kde se objeví další vytištěný znak) na následující řádek. Jestliže stiskneme nějaký příkaz chybným způsobem, nebo jestliže zavedeme cokoliv, čemu počítač nerozumí, pak se po stisknutí RETURN objeví hlášení (např. SYNTAX ERROR - syntaktická chyba). Nazýváme jej chybovým hlášením. V Dodatku A naleznete seznam všech chybových

hlášení spolu s radou, jak tyto chyby opravit. POZNÁMKA: V příkladech, uvedených v tomto manualu o nezbytnosti stisknout tlačítko RETURN informuje symbol RETURN

SHIFT

V dolní řadě tlačítek na klávesnici se nacházejí dvě tlačítka SHIFT, jedno vlevo, druhé vpravo, jako na normálním psacím stroji. Tlačítko SHIFT se používá třemi způsoby:

1. Spolu se souborem znaků velká/malá písmena se tlačítko SHIFT používá jako tlačítko pro velká písmena na normálním psacím stroji. Jestliže je tlačítko SHIFT stlačeno, objeví se velké písmeno, nebo znak uvedený v horní půlce tlačítka se dvěma znaky.
2. Tlačítko SHIFT se používá spolu s jinými příkazovými tlačítky pro provedení zvláštních úkonů.
3. S klávesnicí ve stavu velká písmena/grafické znaky se tlačítko SHIFT používá pro vytištění grafických symbolů nebo těch znaků, které jsou uvedené na přední straně některých tlačítek. Další podrobnosti naleznete v oddíle "Zobrazení grafických znaků" na konci této části.

SHIFT LOCK

Jestliže toto tlačítko stiskneme, zůstane stisknuté. Poté všechny tištěné znaky budou buď velká písmena, nebo znaky uvedené v horní půli tlačítka se dvěma znaky. Pro odblokování tlačítka SHIFT LOCK je musíte stisknout ještě jednou.

Přemístění kursoru

V modu C 128 se kursor bude posouvat jednak po použití čtyř tlačítek s šipkami, umístěnými napravo nahoře na hlavní klávesnici, jednak po použití dvou tlačítek CRSR vpravo v řadě vnitřních tlačítek hlavní klávesnice.

Použití čtyř tlačítek s šipkami

V modu C 128 se kursor přemisťuje v kterémkoliv směru nejjednodušeji pomocí tlačítek s šipkami (nacházejí se v horní řadě tlačítek), které indikují směr pohybu kursoru (tato tlačítka nelze použít v modu C 64).

Použití tlačítek CRSR

Pro přemístění kursoru, jestliže se nacházíme v modu C 128 i C 64, můžeme použít dvě tlačítka, umístěna vpravo v řadě vnitřních tlačítek hlavní klávesnice:

- Stisknutí samotného tlačítka CRSR ↑↓ posunuje kursor dolů
- Stisknutí CRSR ↑↓ spolu s tlačítkem SHIFT posunuje kursor nahoru

- Stisknutí samotného tlačítka CRSR posunuje kursor vpravo
- Stisknutí CRSR spolu s tlačítkem SHIFT posunuje kursor vlevo

Není nutné opakovaně tisknout tlačítko pro pohyb kursoru, aby se posunul o více míst. ve skutečnosti postačí držet tlačítko stisknuté dokud se kursor nedostane do požadované polohy. Upozorňujeme, že kursor po dosažení pravého okraje obrazovky se sám vrátí na počátek a přemístí se na následující řádek. Jestliže přemisťujeme kursor vlevo, pohybuje se po řádku až dosáhne levého okraje a poté přeskočí na konec předchozího řádku. Radíme vám, abyste se důvěrně seznámili s tlačítky pro přemisťování kursoru, protože tato tlačítka usnadňují programování. S určitou praxí budete přemisťovat kursor automaticky, bez přemýšlení.

Inst/Del

Toto tlačítko má dvě funkce. INST slouží pro dosazování, DEL pro mazání.

Dosazování znaků

Pro dosazení znaku do nějakého řádku musíme použít tlačítko SHIFT spolu s tlačítkem INST/DEL. Například, chybně jsme v řádku vytiskli znaky, jako v následujícím příkladu:

JITKA OTONOVÁ

především, abychom mohli doplnit chybějící

znaky, musíme přemístit kursor na místo, v němž je chyba

JITKA ▨OTONOVA

poté podržte tlačítko SHIFT, stiskněte tlačítko INST/DEL a držte je, dokud se nevytvoří dostatečné místo na doplnění znaků:

JITKA ▨ TONOVA

Všimněte si, že INST nepřemisťuje kursor, ale pouze vytvoří místo za kursorem a posune znaky doprava. Abychom dokončili opravu, natiskneme S a O

JITKA SOTONOVA

Vymazání znaku

Stiskněte tlačítko DEL a kursor se přemístí o jedno místo směrem vlevo a vymaže znak, který se nacházel v této poloze. To znamená, že jestliže si přejeme cokoliv vymazat, dosáhneme toho přemístěním kursoru vpravo od znaku, který chceme vymazat

Například, chybně jsme natiskli:

PRINT "HONZEA"

chtěli jsme vytisknout HONZA, ne HONZEA; abychom vymazali E před A umístíme kursor na A a stiskneme tlačítko DEL. Znak napravo od kurzoru se posune o jedno místo vlevo. Objeví se správné slovo:

PRINT "HONZA"

Současné použití INST a DEL

Během opravování lze současně používat INST i DEL. Především přemístíme kursor na opravované znaky a stiskneme tlačítko INST/DEL, abychom vymazali přebytečné znaky. Dále současně tiskneme SHIFT a INST/DEL, abychom si vytvořili potřebné volné místo. Poté vytiskneme opravy. Je rovněž možné tisknout přes chybné znaky a potom použít INST, abychom získali potřebné místo.

CTRL

Tlačítko CTRL /control - řídit/ se používá v kombinaci s jinými tlačítky, abychom provedli některé speciální operace, nazývané řídící funkce. Abychom provedli jednu z těchto operací, přidržíme tlačítko CTRL současně s dalším tlačítkem. Úplný seznam řídících funkcí naleznete v Dodatku I. Řídící funkce se často používají v hotových programech připravených k použití, jako například v programech pro úpravu textů.

Z řídících funkcí se nejčastěji používají ty, které stanoví barvu znaků a kursoru. Abychom zvolili barvu podržte tlačítko CTRL a současně stiskněte jedno z číselných tlačítek /od 1 do 8/, umístěných na horním okraji klávesnice. K dispozici je osm barev, které lze zvolit

pomocí tlačítka Cx, jak bude vysvětleno později.

RUN/STOP

Toto tlačítko má dvě funkce. V některých případech funkce RUN tohoto tlačítka bude dosažena současným stlacením SHIFT a RUN/STOP. Mimoto lze používat operaci STOP tohoto tlačítka, abychom přerušili provádění programu nebo tisku během provádění vlastního programu. V každém případě, ve většině hotových programů je funkce STOP tlačítka RUN/STOP záměrně deaktivována. Měli bychom se vyhýbat přerušování programu dříve než je plně proveden. Jestliže tato funkce není vyřazena z činnosti vydáváme se v nebezpečí, že přijdeme o data v zadrženém programu.

RESTORE

Tlačítko RESTORE se používá spolu s tlačítkem RUN/STOP, abychom uvedli počítač do výchozího stavu.

Ve většině hotových programů je tato funkce vyřazena z provozu ze stejných důvodů, jako funkce tlačítka RUN/STOP: abychom se vyhnuli ztrátě cenných údajů.

CLR/HOME

CLR značí vymazání, zrušení. HOME je jméno, které dáváme levému hornímu rohu obrazovky. Stlačení samotného tohoto tlačítka způsobí návrat kursoru do polohy HOME /domů/, kdežto stisknutí současně s tlačítkem SHIFT vymazává celou obrazovku a kursor se přemístí do polohy HOME.

Tlačítko Commodore /Cx/

Tlačítko Cx /nazývané rovněž tlačítkem Commodore/ má mnoho funkcí, mezi nimi:

1. Tlačítko Cx umožňuje přechod od znaků velká/malá písmena k souboru znaků velká písmena/grafické znaky. Abychom přešli od jesnoho souboru k druhému stiskněte současně tlačítka Cx a SHIFT.
2. Jeden ze dvou modů tlačítka Cx působí jako tlačítko SHIFT pro dosažení grafických znaků uvedených na levém boku každého tlačítka. Stiskněte prostě tlačítko Cx spolu s požadovaným grafickým tlačítkem.
3. Tlačítko Cx kromě toho umožňuje použít řadu osmi dodatečných barev kursoru. Abyste dosáhli těchto barev, podržte tlačítko Cx a a stiskněte jedno z numerických tlačítek /od 1 do 8/, nacházejících se na horním okraji klávesnice.
4. Abychom zpomalili zobrazení průběhu programu, podržíme tlačítko Cx. V tomto modu bude průběh programu viditelně zpomalen. Abychom se vrátili k normální rychlosti provádění, uvolníme tlačítko.
5. Podržte stisknutým tlačítko Cx zatímco zapínáte počítač - přejdete do modu C 64.

Funkční tlačítka

Čtyři tlačítka umístěná nahoře na numerické klávesnice /označená F1, F3, F5 a F7 na horní straně a F2, F4, F6 a F8 na přední straně/ nazýváme funkčními tlačítky. V obou modech C 128 a C 64 mohou být funkční tlačítka programována /viz popis příkazu KEY v Části 5. Kapitoly 2. a v Kapitole 5. - Encyklopedie BASICu 7.0/. Tato tlačítka se často používají v hotových programech pro provedení určitých operací stlačením jediného tlačítka.

Zobrazení grafických znaků

Abychom zobrazili grafické znaky, znázorněné na přední části tlačítek, podržte stlačeným tlačítko SHIFT současně se zajímajícím nás tlačítkem. Grafické znaky napravo mohou být zobrazeny pouze jestliže je klávesnice uvedena do stavu velká písmena/grafické znaky /tento soubor znaků je aktivován po zapnutí počítače/. Abycho znázornili grafické znaky v levé půlce přední strany tlačítka, stiskněte tlačítko Cx

spolu se zajímajícím vás tlačítkem. Tento znak může být zobrazen v obou souborech znaků klávesnice.

Pravidla pro psaní programů v jazyce BASIC

Programy v jazyce BASIC mohou být zaváděny a používány, aniž byste znali jazyk BASIC. Jestliže nebudete věnovat náležitou pozornost psaní, chyby mohou způsobit, že počítačem nebude zaváděná informace přijata. Následují rady, jak napravit chyby při zavádění nebo opisování programů

1. Mezery mezi slovy nejsou důležité; například FORT=1TO10 je rovnocenné FOR T=1 TO 10. Jedno klíčové slovo BASICu však nesmí být přerušeno mezerami /viz Encyklopedie BASICu 7.0, Kapitola 5., seznam klíčových slov BASICu/.
2. Všechna znaky mohou být uzavřeny do uvozovek, Některé znaky však mají speciální funkci, jestliže jsou uzavřeny do uvozovek. Tyto funkce budou vysvětleny později v tomto manuálu.
3. Věnujte pozornost rozdělujícím znaménkům. Uvozovky, dvojtečky a středníky mají speciální účel, který bude ilustrován v později v této části.
4. Stiskněte vždy tlačítko RETURN /značené jako RETURN v tomto manuálu/ po dokončení každého číslovaného řádku.

5. Nepřekračujte 160 znaků v jednom řádku programu. 160 řádků odpovídá 4 úplným řádkům na obrazovce ve formátu 40 sloupců a dvěma úplným řádkům na obrazovce ve formátu 80 sloupců. Viz Část 8. s dalšími podrobnostmi o formátech 40 a 80 sloupců.
6. Dávejte pozor, abyste nezaměnili písmeno l s číslem 1 a písmeno O s číslem 0.
7. Počítač ignoruje vše, co následuje za nápisem REM v jednom řádku programu. REM znamená poznámku /remark/. Příkaz REM slouží ke vkládání komentářů do programu, informací o tom, co se provádí v určité části programu.

Během zavádění programů uvedených v této části manuálu se řiďte radami, uvedenými výše.

Příkaz PRINT

Příkaz PRINT /tiskni/ sděluje počítači, že má zobrazit informace na obrazovce. Umožňuje tisknout jak čísla, tak text /písmena/, avšak pro každý z obou případů existují svoje pravidla, která budou vysvětlena níže.

Tisk čísel

Pro tisk čísel používáme příkaz PRINT následovaný číslem, nebo čísly, která chceme tisk-

nout. Napište

PRINT 5

a poté stiskněte tlačítko RETURN. Uvidíte, že číslo 5 je zobrazeno na obrazovce.

Nyní napiště, následované RETURN:

PRINT 5,6

V tomto příkazu PRINT čárka informuje Commodore 128, že se požaduje vytisknout více než jedno číslo. Jestliže počítač nalezne čárku v řetězci číslic v příkazu PRINT pak každé číslo za čárkou bude vytištěno o 10 mezer vpravo za předchozím číslem. Jestliže si nepřejete mít nadbytečné mezery, použijte středník /;/ místo čárky v příkazu PRINT. Středník přikazuje počítači tisknout čísla těsně za sebou. Jestliže je číslo vytištěno, bude před ním jedna mezera nebo znaménko minus. Vypište následující příklady a podívejte se na výsledek:

PRINT 5;6 RETURN

PRINT 100;-200;300;-400;500 RETURN

Použití otazníku pro zkrácení příkazu PRINT

Peo zkrácení zápisu příkazu PRINT lze použít otazník /?/. V řadě příkladů v této části manuálu používáme symbol ? místo slova PRINT. Většina příkazů BASICu může být zkrácena. Zkratky příkazů BASIC jsou uvedeny v Dodatku K tohoto manuálu.

Tisk textu

Poté, co jsme se naučili jak tisknout čísla, se pokusíme tisknout text. Všechna slova nebo znaky, které chceme zobrazit, musí být vytištěna na klávesnici s uvozovkami na začátku a na konci řetězce znaků. Řetězec nebo string je v BASICu název pro libovolnou posloupnost znaků, uzavřených v uvozovkách. Uvozovky dostaneme stisknutím SHIFT spolu s číselným tlačítkem 2, nacházejícím se u horního okraje klávesnice /ne tlačítko 2 na numerické klávesnice/. Napiště:

?"COMMODORE 128" RETURN

?"4*5" RETURN

Všimněte si, že po stisknutí RETURN počítač zobrazí znaky uzavřené v uvozovkách. Všimněte si rovněž, že v druhém příkladu se nevytiskne výsledek 4x5, protože v tomto případě jsou číslice chápány jako řetězec, ne jako matematická operace násobení. Pro výpočet výsledku 4*5 použijte následující příkaz:

? 4*5 RETURN

Takto lze tisknout jakýkoliv řetězec znaků, za použití příkazu PRINT a uzavírajících tištěné znaky do uvozovek. Abyste skombinovali text a matematické operace v jednom příkazu PRINT použijte jej v následujícím tvaru:

? "4*5="4*5 RETURN

Vidíte, že počítač natiskl znaky v uvozovkách, provedl operaci a vytiskl výsledek. Pořadí textu a operací není důležité, je možné použít obě pořadí v příkazu PRINT. Napište následující příkaz:

?4*(2+3)"SE ROVNA"4*5 RETURN

Vidíte, že na obrazovce se zobrazí rovněž mezera uvnitř uvozovek. Natiskněte:

?" ZDE"

Tisk v různých barvách

Commodore 128 zobrazuje 16 různých barev, které lze snadno měnit. Abychom dostali určitou barvu postačí stisknout tlačítko CTRL současně s některým numerickým tlačítkem od jedné do osmi v horní části hlavní klávesnice. Kursor bude mít barvu podle toho, které numerické tlačítko jsme stiskli. Také všechny znaky budou mít zvolenou barvu podle kursoru. Podržte tlačítko Commodore a současně stiskněte číselné tlačítko od jedné do osmi a obdržíte osm dalších barev.

V tabulce 3-1 jsou uvedeny barvy, které jsou k dispozici v modu C 128 pro formáty 40 a 80 sloupců. V tabulce je rovněž uvedeno pořadí tlačítek /tlačítko CONTROL plus číselné tlačítko nebo tlačítko CX plus číselné tlačítko/

používané pro volbu dané barvy.

Control+	Barva	CX	Barva
1	černá	1	pomerančově žlutá
2	bílá	2	kaštenová
3	červená	3	světlečervená
4	modrá	4	tmavošedivá
5	nachová	5	šedivá
6	zelená	6	světlezelená
7	tmavomodrá	7	světlemodrá
8	žlutá	8	světlešedá

Formát 40 sloupců

Control+	Barva	Cx	Barva
1	černá	1	tmavopurpurová
2	bílá	2	kaštenová
3	tmavočervená	3	světlečervená
4	světlemodrá	4	tmavomodrá
5	světlepurpurová	5	šedivá
6	tmavozelená	6	světlezelená
7	tmavomodrá	7	světlemodrá
8	světležlutá	8	světlešedivá

Formát 80 sloupců

Použití tlačítek kursoru /CRSR/ uvnitř závorek v příkazu PRINT

Při použití tlačítek pro pohyb kursoru uvnitř uvozovek se na obrazovce zobrazí grafické znaky znázorňující tlačítka. Tyto znaky se neobjevují na obrazovce po stisknutí tlačítka RETURN. Vytiskněte otazník, uvozovky /číselné tlačítko 2 současně s tlačítkem SHIFT/, poté stiskněte jedno z tlačítek kursoru desetkrát

"za sebou", napište slovo ZDE a uzavřete uvozovky. Mělo by se objevit následující:

?" ZZZZZZZZZZ ZDE"

Poté stiskněte RETURN. Commodore 128 zobrazí 10 prázdných řádků a na jedenáctém řádku se zobrazí "ZDE". Jak ukazuje předchozí příklad, je možné komunikovat s počítačem také pomocí tlačítka pro řízení kursoru uvnitř uvozovek, když se tiskne na obrazovku.

Jak začít programovat

Až dosud se většina projednávaných příkazů prováděla v přímém modu. To znamená, že příkaz se provedl bezprostředně po stisknutí tlačítka RETURN. Většinu příkazů a funkcí BASICu je však možné použít v programu.

Co je to program

Program je posloupnost očíslovaných příkazů v jazyce BASIC, které ukazují počítači, jaké činnosti má vykonat. Tyto číslované instrukce se nazývají příkazy nebo řádky příkazů.

Čísla řádků

Řádky příkazů jsou číslovány tak, aby poznal v jakém pořadí je má provádět. Počítač provádí řádky programu a řídí se přitom číselným pořadím, pokud mu sám program nepřikáže něco jiného. Pro číslování řádků se používají čísla

od 0 do 63999. V čísle řádku nikdy nepoužívejte čárku /na oddělení tisíců např./. Většina příkazů použitých až dosud byly příkazy používané v přímém modu, ale mohou být použity i jako příkazy programu. Například, napiště

10 ?"COMMODORE 128" RETURN

Vidíte, že počítač nezobrazí na obrazovce nápis COMMODORE 128 po stisknutí tlačítka RETURN, jak by to udělal po použití příkazu PRINT v přímém modu. Je to proto, že číslo 10 před symbolem PRINT /?/ sděluje počítači, že je zaváděn program v BASICu. Počítač si zapamatuje očíslovaný příkaz a čeká na následující.

Nyní napište RUN /běž/ a stiskněte RETURN. Počítač zobrazí slovo COMMODORE 128. Tato operace nemá týž účinek, jako při použití příkazu PRINT v přímém modu. Program je stále přítomen v paměti počítače a může být proveden pokaždé, když budete chtít.

Zobrazení programu - příkaz LIST

Program sestávající z jednoho řádku jako předešlý zůstává stále v paměti C 128. Nyní vymažeme obrazovku - stiskněte současně tlačítka SHIFT a CLR/HOME. Obrazovka je prázdná. Jazyk BASIC obsahuje příkaz pro zobrazení programu, jestliže je nutné prověřit zda se ještě nalézá v paměti počítače - příkaz LIST.

Napiště LIST a stiskněte RETURN. C 128 zobrazí/
10 PRINT "COMMODORE 128"
READY
Pokaždé, když chcete zobrazit všechny řádky nějakého programu, napište LIST. Tato operace se používá především když provádíme změny v programu a chceme zjistit, zda nové řádky byly uloženy v paměti počítače. Jako odpověď na tento příkaz počítač zobrazíz změněnou verzi řádku, řádky nebo program. Následují pravidla pro použití příkazu LIST:

- Abychom zobrazili samotný řádek N, natiskněte LIST N a stiskněte RETURN. Nahraďte N číslem požadovaného řádku.
- Abychom zobrazili program od řádku N až do konce, napište LIST N- a stiskněte RETURN
- Abychom zobrazili řádky od začátku programu až po řádek N, napište LIST -N a stiskněte RETURN
- Abychom zobrazili program od řádky N1 do řádky N2 včetně, napište LIST N1-N2 a stiskněte RETURN

Jednoduchá smyčka - příkaz GOTO

Čísla řádků programu mají i další účel, než pouze systematizovat příkazy ve správném pořadí pro počítač. Slouží také jako odvolávací bod pro počítač v případě, že je požadováno provést tento řádek opakovaně v programu.

Použijte příkaz GOTO /jdi na/, abyste sdělili počítači, že má přeskočit na určitý řádek a provést příkaz /nebo příkazy/ na něm obsažené. Nyní napište:
20 GOTO 10
Stisknutí RETURN po dopsání řádku 20 způsobí, že řádek se připojí k programu v paměti počítače.
Vidíme, že první řádek má číslo 10 a druhý 20. Skutečně je užitečné číslovat řádky programu po 10 /tj. 10, 20, 30, 40 atd/, abychom později mohli připojit další řádky programu mezi existující řádky. Tyto přidané řádky pak lze číslovat násobky pěti /15, 20,..../, jedné /1, 2,.../ - nebo jakýmikoliv mezičísly - abychom dodrželi správné pořadí řádků /viz příkazy RENUMBER a AUTO v Encyklopedii BASICu/.
Napište RUN a stiskněte RETURN. Slovo COMMODORE 128 bude vytištěno na celé obrazovce. Abychom zastavili zobrazování tohoto hesla na obrazovce, stiskněte tlačítko RUN/STOP umístěné v levé části klávesnice.
Dva řádky, které jsme napsali, tvoří jednoduchý program, opakující se do nekonečna, protože druhý řádek přikazuje počítači vrátit se na první řádek. Program pokračuje dokud není zastaven, nebo dokud nevypneme počítač.

Nyní napište LIST RETURN . Na obrazovce se objeví:

10 PRINT "COMMODORE 128"
20 GOTO 10
READY

Program se nalézá v paměti a může být proveden znovu. To je důležitý rozdíl programovacího modu proti přímému modu. Poté co je příkaz proveden v přímém modu, mizí z paměti počítače. Všimněte si také, že když použijeme symbol ? namísto příkazu PRINT, počítač jej nahradí úplným symbolem. To se stane vždy, když zobrazíme příkaz, který jsme původně zapsali zkráceně.

Vymazání paměti počítače - příkaz NEW

Jestliže je potřebí začít od začátku nebo vymazat program v BASICu v paměti počítače, napište NEW /nový/ a stiskněte RETURN. Tento příkaz vymaže paměť BASIC počítače, tedy tu její část, v níž se zapamatovávají programy.

Použití barev v programu

Abychom zvolili barvu v programu, musíme informaci o volbě barvy zahrnout do příkazu PRINT. Například, vymažte paměť počítače příkazem NEW a stisknutím tlačítka RETURN a dále vypište následující frázi a vynechte po jedné mezeře mezi všemi písmeny v uvozovkách:

10 PRINT " S P E K T R U M" RETURN

Dále napište znovu řádek 10, podržte však tlačítko CTRL a stiskněte tlačítko 1 ihned po vytištění prvních uvozovek. Pusťte tlačítko CTRL a napište "S". Poté přidržte znovu tlačítko CTRL a stiskněte tlačítko 2. Pusťte tlačítko CTRL a napište "P". Poté podržte tlačítko CTRL a stiskněte tlačítko 3 a tak pokračujte, dokud nevytiskněte všechna písmena hesla SPEKTRUM a nezvolíte barvy mezi všemi písmeny. Stiskněte tlačítko SHIFT a 2, abyste napsali závěrečné uvozovky, a stiskněte tlačítko RETURN. Dále napište RUN a stiskněte tlačítko RETURN. Počítač zobrazí heslo SPEKTRUM, každé písmeno v jiné barvě. Napište LIST a stiskněte tlačítko RETURN. Uvidíte grafické znaky, připojené k příkazu PRINT na 10. řádku. Tyto znaky ukazují počítači barvy požadované pro každé písmeno a neobjeví se, když Commodore 128 tiskne heslo SPEKTRUM různobarevně.

Znaky pro volbu barvy, nazývané řídící znaky, v příkazu PRINT na 10. řádku přikazují Commodoru 128 změnit barvu. Počítač tiskne následující znaky v nové barvě, dokud nenarazí na následující znak pro volbu barvy. Zatímco znaky mezi uvozovkami se objevují na obrazovce, řídící znaky samy se zobrazují pouze příkazem LIST.

Změny programu

Následující paragrafy pomáhají zavádět do programů opravy a spojovat vlastní programy.

Zrušení řádku v programu

Použijte příkaz LIST pro zobrazení dříve napsaného programu. Dále natiskněte 10 a stiskněte RETURN. Touto operací jste vymazali řádek 10 v programu. Znovu zobrazte program a přesvědčete se o tom. Jestliže starý řádek 10 zůstává zobrazen na obrazovce, přemístěte kursor do libovolného bodu tohoto řádku, stiskněte RETURN a 10. řádek je opět přenesen do paměti počítače.

Zdvojení řádku

Podržte tlačítko SHIFT a stlačte tlačítko CLR/HOME vpravo nahoře na klávesnici. Tato operace vymaže obrazovku. Dále zobrazte program. Přemístěte kursor nahoru, až ho umístíte na "0" 10. řádku. Stiskněte 5 a poté RETURN. Tímto způsobem se zdvojí /tedy zkopíruje/ řádek 10. Zduplikovaný řádek bude označen číslem 15. Napište LIST a stiskněte RETURN a zobrazíte program se zdvojeným řádkem.

NAHRAZENÍ ŘÁDKU

Jakýkoliv řádek lze zaměnit jestliže napíšete staré číslo řádku následované textem nového řádku a poté stisknete RETURN. Stará verze řádku bude vymazána v paměti a nahrazena novým řádkem okamžitě po stisknutí RETURN.

Změnění řádku

Jestliže je zapotřebí přidat jakýkoliv text uvnitř nějakého řádku, přemístěte kursor na znak nebo mezeru bezprostředně následující za bodem, v němž chcete provést vsuvku, poté stiskněte současně SHIFT A INST/DEL a držte je, dokud nezískáte prostor dostatečný pro vsunutí nových znaků.

Zkuste následující příklad. Vymažte paměť počítače pomocí NEW následovaného RETURN a potom napište:

10 ?"MYŠ 128 JE VYNIKAJICI" RETURN

Nyní vsuneme slovo COMMODORE před 128. Přemístíme kursor na číslo 1 v 128. Stiskneme SHIFT současně s tlačítkem INST/DEL dokud nezískáme dostatek místa pro slovo COMMODORE /včetně jedné mezery za E/. Poté natiskneme slovo COMMODORE. Jestliže je nutné zrušit nějaké znaky uvnitř určitého řádku /včetně prázdných mezer/, přemístěte kursor na znak, který následuje za částí, jež má být zrušena, poté podržte tlačítko INST/DEL. Kursor se pohybuje vlevo a znaky nebo mezery se vymazávají, dokud přidržujete INST/DEL.

Matematické operace

Příkaz PRINT lze používat pro provádění výpočtů, jako sčítání, odečítání, násobení, dělení a umocňování. Výpočet se píše za příkazem PRINT

Sčítání a odečítání

Zkuste následující příklady:

PRINT 6+4 RETURN

PRINT 50-20 RETURN

PRINT 10+15-5 RETURN

PRINT 75-100 RETURN

PRINT 30+40,55-25 RETURN

PRINT 30+40;55-25 RETURN

vidíte, že ve čtvrtém výpočtu je výsledek záporný, Mimoto vidíte, že je možné zadat počítači několik výpočtů v jediném příkazu PRINT. Příkaz může používat buď čárku nebo středník - pokud chceme tabelovat výsledky těsně vedle sebe.

Násobení a dělení

Hvězdička /*/ vpravo na klávesnici je symbolem používaným Commodorem 128 pro násobení. Tlačítko "příčka" / umístěná vedle tlačítka SHIFT se používá pro dělení.

Zkuste následující příklady:

PRINT 5*5 RETURN

PRINT 100/2 RETURN

Umocňování

Pro umocnění čísla na určitý mocnitel se používá tlačítko se šipkou směřující vzhůru /↑/, umístěné blízko hvězdičky na klávesnici. Jestliže chcete umocnit nějaké číslo na určitý mocnitel, použijte příkaz PRINT následovaný mocněncem, šipkou vzhůru a mocnitelem, přesně v tomto pořadí. Například, abychom dostali 3 na druhou, natiskněte:

PRINT 3↑2 RETURN

Pořadí operací

V předchozích příkladech jsme viděli, jak kombinovat sčítání a odečítání v témže příkazu PRINT. Když však kombinujeme operace násobení a dělení s operacemi sčítání a odečítání, výsledkem nemusí být to, co jsme chtěli. Napište například:

PRINT 4+6/2 RETURN

Jestliže naším cílem bylo vydělit 10 dvěma, budeme překvapeni až uvidíme, že počítač ohlásí jako výsledek 7. Důvodem tohoto výsledku je, že počítač provádí operace dělení a násobení dříve, než operace sčítání a odečítání. Násobení a dělení má prioritu /přednost/ před sčítáním a odečítáním. Pořadí, v jakém se operace provádějí, nehraje roli. Pořadí provádění matematických operací počítačem se řídí řečenou prioritou operací.

Umocňování čísel předchází všem ostatním čtyřem matematickým operacím. Napište například:

PRINT 16/4↑2 RETURN

Commodore 128 odpoví 1, protože nejprve provede druhou mocninu čtyř, dříve než vydělí 16.

Použití závorek pro stanovení priority operací

Commodoru 128 lze sdělit, které operace mají být provedeny především, tím, že tyto operace uzavřeme do závorek v příkazu PRINT. Vyšetříme první předchozí příklad za předpokladu, že je zapotřebí nejprve provést operaci sčítání a poté dělení - napište:

PRINT (4+6)/2 RETURN

Dostanete výsledek 5.

Jestliže požadujete, aby počítač provedl nejprve dělení a poté umocnění na druhou ve druhém příkladu, napište:

PRINT (16/4)↑2 RETURN

Výsledek bude 16.

Jestliže nepoužijete závorky, počítač provádí operace podle pravidel vysvětlených výše. Všechny operace stejné priority se provádějí zleva doprava. Napište například:

PRINT 4*5/10*6 RETURN

Výsledkem je 12 /4*5=20 .. 20/10=2..2*6=12/, protože operace se provádějí zleva doprava. Jestli chcete 4*5 vydělit 10*6 napište:

PRINT (4*5)/(10*6) RETURN

Výsledek bude .333333333.

Konstanty, proměnné a řetězce

Konstanty

Konstanty jsou numerické hodnoty stálé; tj jejich hodnota se nemění v rovnici nebo během programu. Například číslo 3 je konstanta, stejně jako jakékoliv jiné číslo. Následující příkaz ukazuje, jak počítač používá konstanty:

10 PRINT 3

Odpověď bude vždy 3, nezávisle na tom, kolikrát se tento řádek programu provede.

Proměnné

Proměnné jsou veličiny, které se mohou měnit v rovnici nebo v příkazu programu. V paměti BASIC počítače je jedna část věnována znakům /čísla, písmena a symboly/, které se používají v programu. Tento typ paměti je srovnatelný s tou částí paměti počítače, která je určena pro zapamatování informací o programu; tato část paměti se nazývá proměnná paměť. Napište následující program:

10 X=5

20 ?X

Nyní proveďte program a uvidíte, že počítač zobrazí na obrazovce 5. Řádek 10 oznamuje počítači, že písmeno X reprezentuje číslo 5. v pro-

gramu. Písmeno X je název proměnné, jejíž hodnota X se mění podle hodnoty, která se nalézá napravo od rovnítka. Toto je přiřazovací příkaz; v paměti počítače existuje část, určená pro zapamatování X a této části je přiřazeno číslo 5. Znak = oznamuje počítači, že vše, co se vyskytuje napravo od něho, bude uloženo v určité části paměti /přidělení paměti/, určené pro písmeno X. Jméno proměnné může sestávat z jednoho nebo dvou písmen, nebo z písmene a čísla /prvním znakem MUSÍ být písmeno/. Jméno může být i delší, ale počítač vezme do úvahy pouze první dva znaky. To znamená, že jménům PA a PARTE bude přiřazena stejná část paměti. Mimoto, slova používaná jako příkazy /LOAD, RUN, LIST atd/ nebo jako funkce /INT, ABS, SQR atd/ v BASICu nesmí být použita jako jména proměnných v programu. Obraťte se na Encyklopedii BASICu v Kapitole 5., abyste zkontrolovali, zda jméno proměnné není slovem používaným v BASICu. Povšimněte si, že znak = v přiřazovacím příkazu není ekvivalentní matematickému symbolu "rovná se", ale znamená umístění proměnné /část paměti/ a přiřazuje hodnotu této proměnné.

V programu uvedeném výše hodnota proměnné X je vždy 5. Je však možné přiřadit proměnné výsledek výpočetní operace, vypsané napravo od

znaku =. Abychom identifikovali konstanty můžeme použít text spolu s konstantami v příkazu PRINT. Napište NEW a stiskněte tlačítko RETURN, abyste vymazali obsah paměti Commodoru 128 a poté napište následující program:

```
10 A=3*100
20 B=3*20
30 ?"A SE ROVNA" A
40 ?"B SE ROVNA" B
```

V tomto okamžiku paměť počítače obsahuje dvě proměnné, označené jako A a B, které obsahují příslušně 300 a 60. Jestliže během programu chceme změnit hodnotu určité proměnné, stačí přidat do programu další přiřazovací příkaz. Připojte k předchozímu programu následující příkazy:

```
50 A=900*30/10
60 B=95+32+128
70 GOTO 30
```

Abyste zastavili program, stiskněte tlačítko STOP. Nyní zobrazte program a sledujte postup počítače. Za prvé počítač přiřadí proměnné A hodnotu uvedenou napravo od znaku = v řádku 10. Poté provede totéž pro písmeno B v řádku 20., natiskne hlášení obsažená v řádcích 30. a 40., které udávají hodnoty A a B. Nakonec přiřadí nové hodnoty A a B v řádcích 50 a 60. Staré

hodnoty budou změněny a nebude možné znovu je získat, pokud počítač znovu neprovede řádky 10. a 20. Když počítač pošleme na řádek 30 vytisknout hodnoty A a B, pak tiskne nové hodnoty, spočtené v řádcích 50. a 60. V těchto posledních dvou řádcích se přiřadí nové hodnoty A a B a řádek 70 vrátí počítač na řádek 30. Vznikne nekonečná smyčka, protože řádky od 30. do 70. se budou provádět stále znovu, dokud nestiskneme tlačítko RUN/STOP, abychom program zastavili.

Řetězce /stringy/

Řetězec je tvořen znakem nebo skupinou znaků, uzavřených v uvozovkách. Tyto znaky se zapamatují v paměti počítače podobně, jako proměnné, které jsme zavedli pro číselné hodnoty. Pro reprezentování řetězce můžeme použít jméno proměnné, přesně jako jsme je použili pro reprezentování čísla. Napište symbol dolaru /$/ za jménem proměnné a tím informujete počítač, že jméno proměnné přísluší řetězci a ne číselné proměnné.

Napište NEW a stiskněte RETURN, abyste vymazali obsah paměti počítače, dále napište následující program:

```
10 A$="POCITAC"
20 X=123
30 B$="COMMODORE"
40 Y=1
50 ? A$;X;B$"JE PRO MNE CISLO"Y
```

Jak vidíte, je možné tisknout číselné proměnné a řetězce ve společném příkazu. Procvičte se v práci s proměnnými tím, že si vymyslíte vlastní krátké programy.

Hodnota nějaké proměnné může být vytištěna v přímém modu po provedení programu. Napište ? A$;B$;X;Y po provedení předešlého progarmu a uvidíte, že hodnoty proměnných jsou zachovány v paměti počítače.

Abyste vymazali tuto oblast paměti BASIC, aniž byste přitom vymazali program, použijte příkaz CLR. Stiskněte CLR a RETURN. Všechny konstanty, proměnné a řetězce budou vymazány. Poté napište LIST a uvidíte, že program zůstal v paměti. Příkaz NEW naproti tomu vymaže jak proměnné, tak program.

Příklad programu

Uvedeme nyní příklad programu, který obsahuje většinu metod a příkazů uvedených v této části manuálu.

Tento program počítá střední hodnotu tří čísel /X,Y a Z/a tiskne na obrazovce tato čísla i průměr. Abyste změnili hodnoty proměnných, musíte změnit řádky 10. - 30. programu. V řádku 40. se proměnné sečtou a dělí třemi, abychom dostali průměr. Všimněte si zápisek, kterou

sdělují počítači, že nejprve má provést součet a teprve poté dělit.

RADA: Jestliže používáte v jednom příkazu více závorek, doporučujeme přepočítat otevírací i zavírací závorky a přesvědčit se, že celkem jich je sudý počet.

```
10 X=46
20 Y=72
30 Z=114
40 A=(X+Y+Z)/3
50 ?"PRUMER Z" X;Y;"A"Z;"JE"A;
60 END
```

Uložení a nové použití programu

Po vytvoření programu je možné zapamatovat si jej pro opakované přivolání a použití v budoucnu. Pro tyto operace slouží diskdrajv Commodoru nebo Dataset Commodore 1530 /magnetofon/. Vysvětlíme nyní různé příkazy, umožňující komunikovat s počítačem a diskdrajvem nebo magnetofonem. Tyto příkazy obsahují heslo příkazu, následované různými parametry. Parametry - písmena, slova nebo symboly v příkazu obsahují specifické informace pro počítač, jako například název fajlu nebo numerické proměnné, specifikují číslo periferijního zařízení. Každý příkaz má několik parametrů. Například parametry příkazu pro formátování disku zahrnují název disku a identifikační číslo

neboli kód, a navíc různé další parametry. Parametry se používají téměř ve všech příkazech BASIC; některé jsou proměnné, jiné konstanty. Parametry poskytující nezbytné informace C128 a diskdrajvu:

Parametry na řízení disku

jméno disku	- identifikační jméno o 16 libovolných znacích dané uživatelem
jméno fajlu	- identifikační jméno o 16 libovolných znacích dané uživatelem
kód id.	- identifikeční kód z 2 libovolných znaků daný uživatelem
číslo drajvu	- užívá se 0 pro jednoduchý drajv, 0 nebo 1 pro dvojitý drajv
číslo periferie	- číslo přiřazené periferijní jednotce. Například číslo diskové jednotky Commodoru je 8.

Formátování disku - příkaz HEADER

Pro uchování programu na novém /dosud nepoužitém/ disku musíme především připravit disk tak, aby byl schopen přijímat údaje. Taková operace se nazývá "formátování" disku. Poznámka: Ujistěte se, že máte přístup k diskdrajvu dříve než vložíte disk.

V procesu formátování se disk rozdělí na části /sekce/ nazývané stopy a sektory. Vytvoří se seznam /směrníky/. Pokaždé, když bude na disku ukládán program, jméno přiřazené programu bude připojeno k seznamu.

Commodore 128 má dva typy formátovacích příkazů. Jeden se používá pouze v modu C 128, druhý se používá jak v modu C 64, tak v C 128. Následující paragrafy popisují formátovací příkazy pro mod C 128. Kapitola 3., popisující mod C 64, obsahuje nejdůležitější podrobnosti o programování a řízení disků v C 64.

Příkazem, který formátuje disky je příkaz HEADER /hlavičkovač/ v rozšířeném nebo krátkém tvaru. Při formátování čisté /nové/ diskety se MUSÍ použít rozšířený tvar, jak ukazuje následující příklad:

HEADER "jméno disku",I id.[,Dčíslo drajvu]
[,[ON]Učíslo zařízení]

Za slovem HEADER napište zvolené jméno diskety uzavřené v uvozovkách. Toto jméno sestává maximálně z 16 znaků. Jméno diskety usnadní identifikování při ukládání na disketě.

Za jménem disku napište čárku a písmeno I, pak identifikátor ze dvou znaků následovaný čárkou. Identifikátor diskety nemusí sestávat nutně z číslic, je možné použít rovněž písmena. Pro více disků používejte pro kódování následující za sebou čísla, např. A1, A2, B1, B2.

Jstliže vlastníte jednoduchý diskdrajv, stiskněte v této chvíli RETURN. Commodore 128 bude automaticky předpokládat číslo drajvu 0 a číslo periferie 8. Jestliže vlastníte diskdrajv

dvojitý musíte tyto parametry specifikovat. Následující parametr příkazu volí číslo drajvu. Stiskněte tlačítko "D" a jestliže vlastníte jednoduchý diskdrajv stiskněte tlačítko nula následované čárkou. V dvojitém diskdrajvu budou drajvy označeny 0 a 1. Parametr - číslo periferie začíná písmenem U, takže stiskněte tlačítko "U" následované číslem periferie, určeným pro diskdrajv Commodoru, tedy 8.

Příklad rozšířeného tvaru příkazu HEADER:

HEADER "RESV",IA1,DC,U8 RETURN

Tento příkaz provede formátování: nazve disketu RESV, kód ident. A1, drajv č. 0, periferie č. 8.

Pro diskdrajv se použije číslo 0 a pro periferii č. 8 jestliže tyto parametry nejsou specifikovány jinak. Příklad:

HEADER "DISCC",51 RETURN

Příkaz HEADER se mimoto používá ke smazání všech údajů na použité disketě, načež se disketa může znovu použít jako nová. Dejte však pozor, abyste nevymazali disk, jehož obsah potřebujete pro budoucí použití.

Redukovaný tvar příkazu HEADER používajte, jestliže disketa byla již dříve formátována pomocí rozšířeného tvaru příkazu HEADER. Redukovaný tvar vymaže seznam, vymaže všechna data v modu rozšířeného tvaru, avšak zachová identifi-

který použité dříve. Redukovaný tvar příkazu HEADER:

HEADER "NOVE PROG" RETURN

Ukládání na disku

V modu C 128 lze programy ukládat na disku pomocí jednoho ze dvou následujících příkazů:

DSAVE "JMENO PROGRAMU" RETURN

SAVE "JMENO PROGRAMU",8 RETURN

Oba příkazy jsou ekvivalentní. Připomínáme, že posloupnost znaků DSAVE" se objeví na obrazovce po stisknutí funkčního tlačítka F5, nabo po vypsání této posloupnosti. Jméno programu může obsahovat do 16 znaků, které musí být uzavřeny v uvozovkách. Na disketě nesmějí být uloženy dva programy stejného jména. Jestliže se pokusíte uložit program pod stejným jménem, jako má již uložený program, pak si disk podrží starý program a nepřijme nově zaváděný program.

V druhém příkladu číslo 8 indikuje, že program se má uložit na periferijním zařízení číslo 8. V příkazu DSAVE není nutné specifikovat 8, protože počítač automaticky použije toto zařízení.

Ukládání na kazetě

Jestliže k ukládání programů používáte Dataset, vložte do magnetofonu prázdný pásek, pásek převiňte, pokud je to nutné, a napište:

SAVE"JMENO PROGRAMU" RETURN

Napíšete tedy slovo SAVE následované jménem programu, maximálně ze 16 znaků.

POZNÁMKA: Obrazovka o 40 sloupcích se během ukládání programu vyprázdní a vrátí se do normálního stavu po dovršení operace.

Na rozdíl od disku, na kazetě je možné uložit dva programy stejného jména. Přesto nedoporučujeme přiřadit totéž jméno dvěma programům, protože počítač pak provede vždy ten, který je první v pořadí na pásku. Po uložení může být program zapsán do paměti počítače pokaždé, když to potřebujeme.

Přepsání z disku

Přepsání znamená prostě zkopírování nějakého programu z disku do paměti počítače. Jestliže tato paměť již obsahuje nějaký program BASIC, pak přepsání jiného programu způsobí vymazání starého. Po přepsání programu BASIC z disku se používá v modu C 128 jeden z následujících dvou příkazů:

DLOAD"JMENO PROGRAMU" RETURN

LOAD"JMENO PROGRAMU",8 RETURN

Všimněte si, že v modu C 128 lze použít funkční tlačítko F2 /stiskněte SHIFT a F1/, aby se objevilo slovo DLOAD", nebo je možné slovo napsat pomocí klávesnice. V druhém příkazu je č. 8 oznamuje počítači, že má provést přepis

ze zařízení č. 8 /diskdrajvu/. Podobně jako DSAVE i DLOAD předpokládá číslo diskdrajvu, tj. 8. Vytiskněte jméno programu přesně jako když jste jej ukládali. V opačném případě počítač odpoví "FILE NOT FOUND" /fajl nebyl nalezen/.

Po dokončení přepisu napište příkaz RUN, aby se provedl. Commodore 128 má speciální formát pro příkaz RUN, používaný pro přepsání a provedení programu v modu C 128 pomocí jediného příkazu. Napište RUN následovené jménem programu /které také nazýváme jměnem fajlu/ v uvozovkách:

RUN"JMENO PROGRAMU" RETURN

Přepisování z kazety

Abyste přepsali program z kazety, napište:

LOAD"JMENO PROGRAMU" RETURN

Jestliže neznáte jméno programu na kazetě napište:

LOAD RETURN

Uvidíte, že naleznete první program na pásku. Během hledání programu na Datasetu je obrazovka 40 sloupců prázdná. Hakmile je program nalezen, na obrazovce se objeví:

FOUND JMENO PROGRAMU

Abyste přepsali program, stiskněte tlačítko Commodore /Cx/, nebo /v modu C 128/ stiskněte klávesu pro mezeru a začne se hledat následující

program na pásku.

Pro označení začátku programu použijte počítadlo Datasetu. Abyste potom přivolali program, posuňte pásek z COO na počátek programu a napište

LOAD RETURN

V tomto případu nemusíte specifikovat JMENO PROGRAMU; program se přepíše automaticky, protože se zpracuje první program na pásku.

Další příkazy pro disk

Kontrola programu

Abychom se přesvědčili, zda byl program správně uložen, použijeme v modu C 128 následující příkaz:

DVERIFY"JMENO PROGRAMU" RETURN

Jestliže je program v paměti počítače identický s programem na disku, na obrazovce se objeví "OK". Příkaz VERIFY pracuje rovněž pro programy na pásku. Napište:

VERIFY"JMENO PROGRAMU" RETURN

Nepište čárku a číslo zařízení.

Zobrazení obsahu diskety

V modu C 128 je možné zobrazit seznam nebo směrníky programů uložených na disketě pomocí následujícího příkazu:

DIRECTORY RETURN

Tímto příkazem se zobrazí obsah seznamu. Tuto operaci lze jednodušeji provést stisknutím tlačítka funkčního F3. Po stisknutí F3 C 128 zobrazí slovo DIRECTORY a provede příkaz.
Další podrobnosti o ukládání a přehrávání programů a informace týkající se disket naleznete v manuálech Datasetu a diskdrajvu. Viz rovněž popis příkazů LOAD a SAVE v Kapitole 5., Encyklopedie BASICu 7.0.

x x x x x x x x x x x x x x x x x x x x x x x x x x

V této části manuálu byly uvedeny první poznámky o jazyku BASIC a některé základní představy o programování. V dalších částech manuálu budou tyto základní poznatky prohloubeny a zavedeny další příkazy, funkce a metody používané při programování v BASICu.

3-44

ČÁST 4

Tato část manuálu popisuje použití řady příkazů, funkcí a programovacích metod v BASICu, které lze použít v modech C 128 a C 64. Tyto příkazy a funkce umožňují programovat opakující se operace pomocí smyček a bloků; zavádějí tabulky hodnot; umožňují přesunovat jednu část programu k jiné; přiřazovat proměnné hodnoty veličinám a řadu dalších operací. Uvedeme ukázky programů, abychom předvedli, jak tyto představy a pojmy BASICu aplikovat.

Rozhodování počítače - příkaz IF - THEN

Naučili jsme se, jak měnit hodnotu proměnných; nyní uvidíme, jak počítač přijímá rozhodnutí, založená na zpracování těchto hodnot pomocí příkazu IF-THEN /jestliže - pak/. Jestliže byste přikázali počítači provést samotný příkaz IF /jestliže/, pak by prověřil jednu určitou podmínku /např. IF X=5/. Příkaz, který počítač provede, jestliže je podmínka splněna, následuje za slovem THEN /pak/ příkazu. Vymažte obsah paměti počítače - napište NEW následované RETURN - a dále napište následující program:

```
10 J=0
20 J=J+
30 ?"J,"COMMODORE 128"
40 IF J=5 THEN GOTO 60
```

60 END

Tentokrát není nutné tisknout tlačítko STOP, abychom zastavili provádění programu ve smyčce. Příkaz IF-THEN přikazuje počítači tisknout COMMODORE 128 a zvětšovat J dokud nedojde ke splnění podmínky J=5. Jestliže podmínka v IF není pravdivá, počítač přejde na následující řádek programu nezávisle na tom, co stojí za slovem THEN. Všimněte si příkazu END /konec/ na řádku 60. Radíme vám, vždy psát příkaz END jako poslední řádek programu, čímž ukazujeme počítači, že má zastavit provádění příkazů.

Uvedeme nyní seznam symbolů, které lze použít v podmínce IF, a jejich význam:

SYMBOL	VÝZNAM
=	rovná se
>	větší než
<	je menší než
<>	nerovná se
>=	je větší nebo rovno
<=	je menší nebo rovno

Tyto symboly se řídí matematickými pravidly Odlišně budeme ovšem určovat, zda jeden řetězec je větší než, menší než nebo roven jinému. Viz Kapitolu 5., Encyklopedie BASICu 7.0, kde najdete další informace o operacích "zacházení s řetězci".

V části 5. budou popsány některé další možnosti, jak využít IF-THEN, založené na takových příkazech BASICu, jako jsou BEGIN, BEND a ELSE.

Používání dvojtečky

Dvoutečka se často používá při programování pro oddělení dvou /nebo více/ příkazů BASIC na témže řádku. Příkazy oddělené dvojtečkou na jednom řádku se provádějí v přirozeném pořadí, zleva doprava. Jeden řádek programu může obsahovat tolik příkazů, kolik se jich může vejít maximálně do 160 znaků, včetně čísla řádku. Je to ekvivalentní 4 úplným řádkům na obrazovce ve formátu 40 sloupců, a dvěma úplným řádkům na obrazovce ve formátu 80 sloupců. Tímto způsobem lze lépe využít část THEN příkazu IF-THEN. Je vskutku možné přikázat počítači provést více příkazů, jestliže je podmínka v IF pravdivá. Vymažte obsah paměti počítače a napište následující program:

```
10 N=1
20 IF N<5 THEN PRINT N;"MENSI NEZ 5":GOTO 40
30 ? N;"VETSI NEBO ROVNO 5"
40 END
```

Změňte v řádku 10 N=40 a proveďte program znovu. Vidíte, že počítači lze přikázat provést více příkazů, jestliže N je menší než 5. Za slovem THEN by mohl být zaveden libovolný příkaz. Všimněte si, že GOTO 40 se neprovádí, po-

kud není splněna podmínka N<5. Každý příkaz, který lze provést nezávisle na tom, že podmínka v IF není splněna, se musí objevit na vzláštním řádku.

Smyčka - příkaz FOR - NEXT

V programu použitém jako ilustrace IF-THEN se pštkrát tiskne COMMODORE a vždy se přikazuje počítači zvětšit o jedničku hodnotu J, až tato hodnota dosáhne pěti, kdy se dá příkaz ukončit program. V BASICu lze tyto operace provést jednodušeji za pomocí smyčky FOR-NEXT /pro-další/:

```
10 FOR J=1 TO 5
20 ?J; "COMMODORE 128"
30 NEXT J
40 END
```

Zaveďte a proveďte tento program a srovnejte výsledek s výsledkem programu IF-THEN. Výsledek je týž. Vskutku, stupně, kterými prochází počítač budou téměř identické pro oba programy. Smyčky FOR-NEXT se často používá při programování, protože umožňuje upřesnit, kolikrát se má provést stejná operace. Prozkoumáme nyní fáze provádění předchozího programu.

Nejprve počítač přiřadí proměnné J hodnotu 1. Číslice 5 v příkazu FOR na řádku 10. přikazuje počítači provést všechny příkazy nalézající

se mezi příkazy FOR a NEXT, dokud J nepřekročí hodnotu 5. V našem případě se provede jediný příkaz - PRINT.

Počítač nejprve přiřadí J hodnotu 1 a provede příkaz PRINT. Dále program dojde k příkazu NEXT J /další J), J se zvětší a porovná se s 5. Jestliže J nepřekročilo hodnotu 5, počítač se vrátí k příkazu PRINT. Po patinásobném provedení této smyčky hodnota J bude 5, program přeskočí na příkaz bezprostředně následující za NEXT a pokračuje odtud. V našem případě je následujícím příkazem END, takže program se zastaví.

Prázdná smyčka - přestávky v programu

Dříve než budeme pokračovat ukážeme si, jak lze v některých případech použít smyčku pro pozdržení programu, který počítač až do daného okamžiku prováděl velkou rychlostí. Pokuste se pochopit, jak pracuje následující program:

```
10 A$="COMMODORE C 128"
20 FOR J=1 TO 20
30 PRINT
40 FOR K=1 TO 1500
50 NEXT K
60 PRINT A$
70 NEXT J
80 END
```

Smyčka na řádcích 40 a 50 vyžaduje na počítači, aby počítal do 1500 dříve než bude provádět program dále. Jakmile se program octne uvnitř hlavní smyčky bude volat vnitřní smyčku.

V části 5, je popsán způsob, jak zavést přestávku pomocí nového příkazu BASICu 7.0 - SLEEP.

Příkaz STEP

Počítači lze přikázat, aby připočítával jiné určité množství (např. 10, 0.5 nebo jiné) pomocí příkazu STEP (krok) s příkazem FOR. Například, abychom počítali do 100 po 10 napíšeme:

```
10 FOR X=0 TO 100 STEP 10
20 ? X
30 NEXT
```

Jestliže se provádí jednoduchá smyčka, není nutné psát X za příkazem NEXT. Povšimněte si, že kromě přírustků při počítání ve smyčce, je možné i ubírat. Například změňte řádek 10 předchozího programu následovně:

```
10 FOR X=100 TO 0 STEP -10
```

Počítač bude počítat zpět od 100 do 0 po desítkách. Jestliže je přírustek 1, příkaz STEP v příkazu FOR se nemusí použít.

Komponenty příkazu FOR - NEXT jsou:

FOR - (pro) slovo označující začatek smyčky

X - odečítací proměnná; lze použít libovolnou číselnou proměnnou

1 - počáteční hodnota; libovolné číslo, kladné nebo záporné

TO - (do) spojuje počáteční a koncovou hodnotu

100 - koncová hodnota; libovolné číslo, kladné nebo záporné

STEP - (s krokem) jestliže se použije přírustek jiný než 1

2 - přírustek; libovolné číslo, kladné nebo záporné.

V části 5. popíšeme další možný příkaz BASICu 7.0, příkaz DO/LOOP, který provádí podobné operace jako příkaz STEP.

Zavádění dat

Příkaz INPUT

Přiřazení hodnoty proměnné

Vymažte paměť počítače příkazem NEW a RETURN, poté napište a proveďte následující program:

```
10 K=10
20 FOR I=1 TO K
30 ?"COMMODORE 128"
40 NEXT
```

V tomto programu lze měnit hodnotu K v řádku 10, abychom smyčku provedli opakovaně požadovaný počet krát. Tato operace se musí provést při psaní programu, dříve než se provede.

Nyní vysvětlíme, jak sdělit počítači během provádění programu kolikrát má smyčku provést. Jinými slovy, v tomto příkladu se požaduje změnit hodnotu K pokaždé, když se provádí program, aniž bychom museli měnit program sám. Znamená to mít možnost ovlivnit počítač. Je vskutku možné pomocí příkazu INPUT (vezmi) zařídit to tak, že se počítač zeptá, kolikrát má smyčku provést. Nahraďte například řádek 10 řádkem:

```
10 INPUT K
```

Při provedení programu počítač odpoví "?". V tomto okamžiku musíme udat hodnotu připisovanou K. Napište např. 15 a stiskněte RETURN. Počítač provede smyčku patnáctkrát.

Hlášení - vodítko

Je možné zařídit, aby počítač v příkazu INPUT uvedl, kterou proměnnou je třeba zadat. Nahraďte řádek 10:

```
10 INPUT "ZADEJ HODNOTU K";K
```

To co je mezi uvozovkami se vytiskne. Použijte středník mezi uvozovkami, končícími hlášení a písmenem K. V tisku lze použít libovolný nápis, pokud pouze příkaz INPUT nepřesáhne délku 160 znaků, což platí pro všechny příkazy BASICu.

Příkaz INPUT se používá i se stringovými proměnnými (pro řetězce) a budou pro ně platit stejná pravidla jako pro číselné proměnné. Nezapomeňte však použít symbol $, abyste naznačili, že jde o stringovou proměnnou. Vymažte paměť počítače příkazem NEW a RETURN a dále napište nasledující program:

```
10 INPUT "JAK SE JMENUJES";N$
20 ?"NAZDAR";N$
```

Poté proveďte program příkazem RUN. Po objevení se nápisu JAK SE JMENUJES? natiskněte vlastní jméno a nezapomeňte stisknout RETURN za jménem.

Po zavedení hodnoty určité proměnné (číselné nebo stringové) pomocí INPUT se lze na tuto proměnnou odvolávat v libovolném místě programu, jstliže použijeme jmého této proměnné. Stiskněte ?N$ RETURN. Počítač si vzpomene na vaše jméno.

Příkaz GET

Komunikovat s počítačem je možné i pomocí jiných příkazů BASICu. Jedním z nich je příkaz GET (dej), podobný příkazu INPUT. Abyste pochopili činnost příkazu GET vymažte paměť počítače a napište následující program:

```
10 GET A$
20 IF A$="" THEN 10
30 ? A$
40 END
```

Po napsání RUN následovaného RETURN se zdá, že počítač nereaguje. Je to proto, že počítač čeká na stisknutí nějakého tlačítka. Příkaz GET přikazuje počítači zkontrolovat stav klávesnice a nalézt znak (tlačítko), které je stisknuté. Požadavek počítače rovněž nebude uspokojen stisknutím znaku, uvedeného na řádku 20. Tento řádek přikazuje počítači vrátit se na řádek 10 v případě, že není stisknut žádný znak ("žádný znak" je označen dvěma uvozovkami bez mezery mezi nimi). Smyčka se opakuje, dokud nestiskněte nějaké tlačítko. Pak počítač přiřadí proměnné A$ znak stisknutého tlačítka.

Příkaz GET se často používá při programování tlačítek na klávesnici. V předchozím příkladu stisknutí Q má za následek, že se objeví příslušné hlášení na obrazovce. Nyní napište a proveďte program a poté stiskněte Q:

```
10 ?"STISKNI Q A ZOBRAZIS HESLO"
20 GET A$
30 IF A$="" THEN 20
40 IF A$="Q" THEN 60
50 GOTO 20
60 FOR I=1 TO 25
70 ? "UZ UMIS POUZIVAT PRIKAZ GET"
80 NEXT
90 END
```


Všimněte si, že při stisknutí jiného tlačítka než Q počítač nic nezobrazí a vrátí se na řádek 20 v očekávání jiného znaku.

V části 5. popíšeme, jak používat příkaz GETKEY, další účinný příkaz BASICu 7.0, používaný při podobných akcích jako popsané výše.

Ukázka programu

Abycho se naučili používat příkaz FOR-NEXT a příkaz INPUT, vymažte paměť počítače příkazem NEW RETURN a poté napište následující program:

```
10 T=0
20 INPUT "KOLIK CISEL";N
30 FOR J=1 TO N
40 INPUT "UVED CISLO";X
50 T=T+X
60 NEXT
70 A=T/N
80 PRINT
90 ?"MAME";N"CISEL A CELKEM";T
100 ?"PRUMER";A
110 END
```

S tímto programem lze počítat průměr libovolného počtu čísel. Čísla lze měnit pokaždé, když se program provádí, aniž bychom měnili sám program. Prozkoumáme nyní operace prováděné počítačem řádek po řádku:

řádek 10 přiřadí T hodnotu 0 (součet čísel)

řádek 20 umožňuje zadat počet čísel (proměnná N) pro výpočet průměru

řádek 30 říká počítači, aby provedl smyčku N-krát

řádek 40 umožňuje zadat čísla, jejichž průměr požadujeme

řádek 50 připočte každé číslo k celkovému součtu

řádek 60 říká počítači, aby zvětšil hodnotu (J) a vrátil se na řádek 30 pokud hodnota J je <=N

řádek 70 úhrnný součet zadaných čísel se dělí (N) po provedení smyčky N-krát

řádek 80 zobrazí prázdný řádek

řádek 90 tiskne hlášení udávající kolik čísel bylo zadáno a jejich součet

řádek 100 tiskne průměr těchto čísel

řádek 110 sděluje počítači, že program skončil

Příkaz READ - DATA

Existuje i další způsob, jak sdělit počítači znaky pro použití v programu: příkaz READ (čti) použitý v programu umožní počítači převzít jistý počet znaků z příkazu DATA (údaje). Například abychom nalezli průměr pěti čísel, použijeme příkazy READ a DATA následujícím způsobem:

```
10 T=0
20 FOR J=1 TO 5
30 READ X
40 T=T+X
```



```
50 NEXT
60 A=T/5
70 ?"PRUMER";A
80 END
90 DATA 5,12,1,34,18
```

Po provedení programu počítač vytiskne PRUMER 14. Program používá proměnnou T pro zachování součtu a počítá průměr stejně jako v předchozím příkladu. Program READ-DATA však nalézá čísla pro výpočet průměru v řádku DATA. Všimněte si řádku 30, READ X. Příkaz READ informuje počítač, že v programu se nachází také příkaz DATA. Počítač vyhledá řádek DATA a použije z něho první číslo jako běžnou hodnotu proměnné X. Když se smyčka provádí podruhé, jako hodnota X se použije druhé číslo z příkazu DATA a tak dále.

V příkazu DATA lze použít libovolné číslo, ne však aritmetický výraz. Příkaz DATA se může nacházet kdekoliv v programu, i za příkazem END. Je to proto, že počítač ve skutečnosti neprovádí nikdy příkaz DATA, ale obrací se k němu. Oddělujte čárkami různé prvky, avšak nevkládejte čárku mezi slovo DATA a první číslo seznamu.

Jestliže program obsahuje více příkazů DATA, počítač se obrátí na ten, který je nejblíže příkazu READ během provádění programu. Poč-

čítač si označí poslední přečtený údaj. Po přečtení prvního čísla příkazu DATA se značka přesune na následující číslo. Jestliže se počítač vrátí k příkazu READ, pak přiřadí proměnné v příkazu READ hodnotu, na níž ukazuje tato značka.
V programu lze použít tolik příkazů READ a DATA, kolik je nutné, aby bylo zajištěno, že příkazy DATA obsahují dostatečné množství údajů s ohledem na příslušné příkazy READ. Odejměte z příkazu DATA v posledním programu jedno z čísel a proveďte program. Počítač ohlásí ?OUT OF DATA ERROR IN 30 (nedostatek údajů v 30). V tomto případě je jasné, že při provádění smyčky popáté počítač nenalezl žádný údaj, který by mohl přečíst. Výsledkem je chybové hlášení. Naopak, zavedení příliš mnoha údajů do příkazu DATA nezpůsobí počítači problémy, protože přítomnost přebytečných údajů nezjistí.

Příkaz RESTORE

Příkaz RESTORE (navrátit se) se v programu používá, abychom uvedenou značku vrátili na první údaj. V předchozím programu nahraďte příkaz END (řádek 80) následujícím příkazem:

```
80 RESTORE
```

a přidejte

```
85 GOTO 10
```


Nyní proveďte program. Program se i nadále provádí za použití příkazu DATA. Poznámka: Jestliže počítač hlásí OUT OF DATA ERROR znamená to, že jsme zapomněli přidat nové číslo namísto čísla dříve zrušeného v příkazu DATA, takže všechna data jsou vyčerpána dříve než příkaz READ bude proveden tolikrát) kolikrát je požadováno.
Příkaz DATA je možné použít i pro přiřazení hodnoty stringové (řetězcové) proměnné. Vymažte paměť počítače a napište následující program:

```
10 FOR J=1 TO 3
20 READ A$
30 ? A$
40 NEXT
50 END
60 DATA POCITAC, COMMODORE, 128
```

Jestliže příkaz READ hledá proměnný řetězec, lze v příkazu DATA použít písmena nebo číslice. Všimněte si však, že jelikož počítač čte řetězec, číslice se rovněž ukládají v paměti jako obecné znaky a ne jako číselné hodnoty, které by mohly být modifikovány. Čísla uložená v paměti jako řetězec mohou být vytištěna, nelze je však použít ve výpočtech. Naopak, v příkazu DATA nelze používat písmena, jestliže příkaz READ hledá číselnou proměnnou.

Používání matic

Ukázali jsme si, jak používat READ-DATA při přiřazování mnoha hodnot jedné proměnné. Jále jsme vysvětlili jak si zapamatovat všechny údaje v jednom příkazu DATA, zatímco měníme hodnotu proměnné s novými daty, nebo jak například přivolat třetí číslo nebo druhý řetězec znaků.

Pokaždé, když přiřazujeme novou hodnotu nějaké proměnné, počítač "smaže" předchozí hodnotu v buňce paměti této proměnné a uloží na toto místo novou hodnotu. Lze však přinutit počítač, aby rezervoval celý blok paměťových buněk a uchoval všechny hodnoty přiřazované této proměnné v programu. Tento blok buněk se nazývá matice.

Indexované proměnné

Matici obsahující všechny hodnoty přiřazené proměnné X v příkazech READ-DATA budeme nazývat maticí X. První hodnotu přiřazenou programem X budeme nazývat X(1), druhou X(2) atd. To je tak zvaná indexovaná proměnná. Číslo v závorce se nazývá index; jako index lze použít i proměnnou nebo výraz. Nyní uvedeme druhou verzi programu pro výpočet průměru s použitím indexované proměnné:

```
5  DIM X(5)
10 T=0
20 FOR J=1 TO 5
30 READ X(J)
40 T=T+X(J)
50 NEXT
60 A=T/5
70 ? "PRUMER =";A
80 END
90 DATA 5,12,1,34,18
```

Vidíte, že nebylo nutné provést příliš mnoho změn. Jediným novým příkazem je řádek 5, který upozorňuje počítač, že musí rezervovat pět buněk paměti pro matici X. Řádek 30 je změněn tak, že počítač pokaždé, když provádí smyčku, přiřadí prvku matice X hodnotu z příkazu DATA, která odpovídá "počítadlu" smyček (J). V řádku 40 se počítá součet, přesně jako v předchozí variantě tohoto programu, avšak za použití indexované proměnné. Po provedení programu, jestliže si chceme vzpomenout například na třetí číslo, napište ? X(3) RETURN. Počítač odpoví tímto číslem z matice X. Tímtéž způsobem lze vytvořit i matice řetězců, abychom uchovali znaky stringových proměnných. Předělame program READ-DATA POCITAC COMMODORE 128 tak, že počítač bude moci uchovat příslušné prvky v matici A$:

```
 5 DIM A$(3)
10 FOR J=1 TO 3
20 READ A$(J)
30 ? A$(J)
40 NEXT
50 END
60 POCITAC, COMMODORE, 128
```

RADA: Program nemusí obsahovat příkaz DIM, ledaže použitá matice obsahuje více než 10 prvků. Viz Rozměry matic.

Rozměry matic

Matice použité spolu se smyčkou umožňují počítači zacházet efektivněji s daty. Předpokládejme nyní, že máme tabulku o 10 řádcích, s pěti čísly na každém řádku a chce spočíst průměr čísel z každého řádku. Museli bychom vytvořit 10 matic a říci počítači, aby spočetl průměr pěti čísel z každého řádku. To není nutné, protože je možné uspořádat čísla do jedné dvouměrné matice. Tato matice bude mít rozměry jako tabulka čísel s nimiž pracujeme - 10 řádků a 5 sloupců. Příkaz DIM pro takovou matici (nazveme ji opět matice X) bude:

```
10 DIM X(10,5)
```

Tímto žádáme počítač, aby rezervoval v paměti místo pro dvouměrnou matici pojmenovanou X. Počítač vyčlení dostatečné místo pro 60 čísel. Není přitom nutné zaplnit matici zcela čísly,

pro něž byla dimenzována, počítač však rezervuje místo pro všechny prvky matice.

Ukázka programu

Nyní se můžeme libovolně odvolávat na libovolné číslo tabulky, označené sloupcem a řádkem. Pohleďte na následující tabulku. Nalezněte třetí prvek v desátém řádku (1500). Program nazývá toto číslo X(10,3). Následující program načte čísla z tabulky do jedné dvouměrné matice (X) a spočte průměry čísel v každém řádku.

Řádek	Sloupec				
	1	2	3	4	5
1	1	3	5	7	9
2	2	4	6	8	10
3	5	10	15	20	25
4	10	20	30	40	50
5	20	40	60	80	100
6	30	60	90	120	150
7	40	80	120	160	200
8	50	100	150	200	250
9	100	200	300	400	500
10	500	1000	1500	2000	2500

Program:

```
10 DIM X(10,5),A(10)
20 FOR R=1 TO 10
30 T=0
40 FOR C=1 TO 5
```

```
 50 READ X(R,C)
 60 T=T+X(R,C)
 70 NEXT C
 80 A(R)=T/5
 90 NEXT R
100 FOR R=1 TO 10
110 PRINT "RADEK#";R
120 FOR C=1 TO 5
130 PRINT X(R,C)
140 NEXT C
150 PRINT "PRUMER=";A(R)
160 FOR D=1 TO 1000: NEXT
170 NEXT R
180 DATA 1,3,5,7,9
190 DATA 2,4,6,8,10
200 DATA 5,10,15,20,25
210 DATA 10,20,30,40,50
220 DATA 20,40,60,80,100
230 DATA 30,60,90,120,150
240 DATA 40,80,120,160,200
250 DATA 50,100,150,200,250
260 DATA 100,200,300,400,500
270 DATA 500,1000,1500,2000,2500
280 END
```


Podprogramy

Příkaz GOSUB - RETURN

Až dosud jediný způsob jak přikázat počítači přejít do jiné části programu byl příkaz GOTO. Nyní vysvětlíme, jak přejít do jiné části programu, provést příkazy obsažené v této části a pak se vrátit na místo, které jsme opustili a dále provádět program. Ta část programu, do níž se počítač přemístí a provede ji, se nazývá podprogram nebo sabrutina. Vymažte paměť počítače a napište následující program:

```
10 A$="PODPROGRAM": B$="PROGRAM"
20 FOR J=1 TO 5
30 INPUT "ZADEJ CISLO";X
40 GOSUB 100
50 PRINT B$:PRINT
60 NEXT
70 END
100 PRINT A$:PRINT
110 Z=X↑2: PRINT Z
120 RETURN
```

Tento program umožní umocnit požadované číslo na druhou a vytiskne výsledek. Ostatní tištěná hlášení indikují, zda počítač provádí podprogram nebo hlavní program. Řádek 40. přikazuje počítači přejít na řádek 100, provést následující příkazy až do posledního řádku, dokud

nenarazí na RETURN /vrať se/.Příkaz RETURN říká počítači, že se má vrátit na řádek bezprostředně následující za příkazem GOSUB /jdi do sabrutiny/ a pokračovat v provádění programu. Podprogram se může nalézat v libovolné části programu - i za příkazem END. Uvědomte si však, že příkazy GOSUB a RETURN musí vždy být používány ve spojení s jinými /jako FOR-NEXT nebo IF-THEN/, jinak počítač podá chybové hlášení.

Příkaz ON GOTO/GOSUB

Existuje i další metoda, nazývaná podmíněný skok, umožňující počítači přejít do jiné části programu pomocí příkazu ON /při,pokud/. Počítač se rozhoduje, do které části programu přeskočit na základě výpočtu nebo zásahu z klávesnice. Příkaz ON se používá spolu s příkazem GOTO nrbo příkazem GOSUB-RETURN, podle potřeby. Za příkazem ON musí následovat proměnná nebo výraz. Za příkazem GOTO nebo GOSUB musí následovat seznam čísel řádků. Napište následující program, abyste porozuměli činnosti příkazu ON:

```
10 ?"ZVOL CISLO MEZI JEDNICKOU A PETKOU"
20 INPUT X
30 ON X GOSUB 100,200,300,400,500
40 END
100 ?"CISLO JEDNA":RETURN
200 ?"CISLO DVA":RETURN
300 ?"CISLO TRI":RETURN
400 ?"CISLO CTYRI":RETURN
500 ?"CISLO PET":RETURN
```

Jestliže číslo X je 1, počítač přeskočí na první číslo řádku v seznamu /tj. 100/, Jestliže X je 2, počítač přeskočí na druhé číslo řádku v seznamu /200/ atd.

Použití rozdělení paměti

Použití PEEK a POKE pro přístu do RAM nebo ROM

Každá oblast paměti počítače má svůj zvláštní účel. Například rozsáhlá oblast paměti je určená na ukládání programů a spojených s nimi proměnných. Tato část paměti, nazývaná RAM, se vymže příkazem NEW. Jiné části, ne tak rozsáhlé, mají speciální účel. Například část paměti řídí hudební funkce počítače.

Pro přístup do paměti a její ovlivňování používáme dva příkazy BASICu: PEEK a POKE. Příkazy PEEK a POKE jsou velmi užitečné při programování, protože obsah paměti počítače přesně určuje operace prováděné počítačem v určitém okamžiku.

Použití PEEK

PEEK se používá, když chceme zjistit hodnotu, uloženou počítačem v určité buňce počítače /buňka počítače obsahuje číslo od 0 do 255/.

Můžeme odečíst hodnotu libovolné buňky paměti /RAM i ROM/ pomocí příkazu PEEK jak v přímém, tak v programovacím modu. Napište:

P=PEEK 2594 RETURN

? P RETURN

Při stisknutí RETURN po prvním řádku počítač přiřadí P hodnotu buňky paměti č. 2594. Po stisknutí RETURN za příkazem ?P se tato hodnota vytiskne. Buňka paměti č. 2594 určuje, jestli tlačítka jako klávesnice mezery a CRSR fungují opakovaně, když je držíme stisknuté. Hodnota 128 v buňce 2594 je vyvolána touto operací. Podržte stisknutou klávesu mezery a pozorujte, ja se kursor pohybuje po obrazovce.

Použití POKE

Abychom změnili obsah nějaké buňky RAM, používáme příkaz POKE. Napište

POKE 2594,96 RETURN

Počítač zaznamená hodnotu za čárkou /96/ do paměťové buňky před čárkou /2594/. Číslo 96 v paměťové buňce 2594 přikazuje počítači, aby neprováděl opakované akce tlačítek jako jsou klávesa mezery nebo tlačítko CRS pokud zůstávají stisknuté. Podržte tedy klávesnici mezery a pozorujte chování kursoru. Kursor se pohne o jedinou mezeru doprava. Abychom vrátili počítač do normálního stavu napište

POKE 2594,128 RETURN

Není možné měnit hodnoty všech paměťových buněk; hodnoty v ROM lze číst, ale ne změnit.

Poznámka: Tyto příklady předpokládají, že hledáme v paměťové bance č. 0. Viz popis příkazu BANK v Kapitole 5., Encyklopedie BASICu 7.0 s dalšími podrobnostmi o bankách paměti.

Základní funkce

Co je to funkce

Funkce je operace definovaná v jazyce BASIC, která obecně dodává jedinou hodnotu. Jestliže funkce dodá určitou hodnotu říkáme, že "vrací" hodnotu. Například funkce SQR /squer - druhá odmocnina/ je mstematická funkce, která vrací hodnotu čísla, jež je druhou odmocninou zadaného čísla.

Existují dva typy funkcí:

Numerické - vracejí výsledek ve tvaru čísla. Numerické funkce počítají matematické hodnoty z určité numerické hodnoty, uložené v dané paměťové buňce.

Stringové /řetězcové/ - vracejí výsledek ve tvaru řetězce znaků.

Nyní popíšeme některé nejčastěji používané funkce. Viz rovněž Kapitolu 5., Encyklopedie BASICu 7.0, obsahující úplný seznam funkcí BASICu 7.0.

Funkce celá část /INT/

Abychom zaokrouhlili nějaké číslo směrem dolů, používáme funkci INT, která odřízne všechny číslice za desetinnou tečkou. Napište následující příklady:

? INT (4.25) RETURN

? INT (4.75) RETURN

? INT (SQR (50)) RETURN

Jestliže potřebujeme zaokrouhlit běžným způsobem vyšetříme druhý případ, který by měl v takovém případě dát výsledek 5. Abycho číslo s desetinnou části převyšující 0.5 zaokrouhlili nahoru, přidáme k číslu 0.5 dříve než použijeme funkci INT. Napiště:

? INT (4.75+0.5) RETURN

Počítač sečte 4.75 a 0.5 dříve než provede funkci INT, takže obdrží 5.25, které pak zaokrouhlí na 5. Abychom viděli, jak se zaokrouhlije výsledek nějaké operace, napište:

? INT ((100/6)+0.5) RETURN

Dělení ve vnitřních závorkách lze nahradit libovolnou jinou operací.

Abychom zaokrouhlili číslo na 0.01 /bližší/, přidáme k číslu 0.005 místo 0.5 a vynásobíme stem. Například 2.876 zaokrouhlíme na setiny následovně:

? INT (2.876+0.005)*100 RETURN

Dále použijeme funkci INT, abychom odstranili vše za desetinnou tečkou /posunuli jsme se o dvě místa doprava po násobení stem/. Operace:

? INT ((2.876+0.005)*100) RETURN

dá celkem 288. Dělíme nyní stem a obdržíme hodnotu 2.88 - správnou odpověď. Tímto způsobem lze na 0.01 zaokrouhlovat i výsledky výpočtů, jako například:

? INT ((2.876+1.29+16.1-9.534 +0.005)*100)/100 RETURN

Generování náhodných čísel - funkce RND

Funkce RND /od random - náhodný/ přikazuje počítači vytvořit náhodné číslo. Tato čísla se používají při simulování hazardních her a při tvorbě grafických a hudebních programů. Všechna náhodná čísla obsahují 9 číslic, v desetinném tvaru od 0.000000001 do 0.999999999. Napište

? RND(0) RETURN

Jestliže generované náhodné číslo vynásobíme šesti, obdržíme číslo s hodnotou větší než 0 a menší než 6. Abychom zahrnuli číslo 6 mezi generovaná čísla, potřebujeme přidat jedničku k výsledku RND(0)*6 a dostaneme $1 < x < 7$. Jestliže aplikujeme funkci INT, abychom odstranili desetiny, příkaz bude generovat celá čísla od 1 do 6. Tímto způsobem můžeme simulovat vrhy kostkou:

```
10 R=INT(RND(1)*6+1)
20 ? R
30 GOTO 10
```

Každé generované číslo reprezentuje vrh jednou kostkou. Jastliže chceme simulovat dvě kostky, použijeme dva příkazy tohoto typu. Každé číslo se bude generovat zvlášť a součet dvou čísel reprezentuje úhrn bodů na dvou kostkách.

Funkce ASC a CHR$

Každému znaku /včetně grafických znaků/, který lze zobrazit na Commodore 128, odpovídá určitá číselná hodnota. Toto číslo nazýváme kódem řetězce znaků /CHR$/ a na Commodoru 128 nepřesahuje 255. S tímto pojmem jsou spojeny dvě funkce: První je funkce ASC. Napište:

? ASC("Q") RETURN

Počítač zobrazí 81, což je kód znaku tlačítka Q. Zaměňte Q v předchozím příkladu libovolným jiným znakem, a zjistíte číselný kód ASCII kteréhokoliv jednoduchého znaku na Commodoru. Druhou funkcí je CHR$, Napište:

? CHR$(81) RETURN

Počítač odpoví Q. Funkce CHR$ je přesným opakem funkce ASC a obě mají vztah k tabulce kódů znaků v paměti počítače. Hodnoty CHR$ můžeme používat při programování funkčních tlačítek. Další podrobnosti o použití CHR$ naleznete v části 5. Viz Dolněk E tohoto manuálu, kde najdete

úplný seznam kódů ASC a CHR$.

Konverze řetězců a čísel

Často je nutné provést výpočet s číselnými znaky, uloženými jako stringové proměnné v programu, nebo provést stringové operace s čísly. Z tohoto důvodu existují v BASICu dvě funkce, používané pro převádění číselných proměnných na řetězce a naopak.

Funkce VAL

Funkce VAL /od value - hodnota/ vrací číselnou hodnotu, jestliže jejím argumentem je řetězec. Vymažte paměť počítače a napište následující program:

```
10 A$="64"
20 A=VAL(A$)
30 ? "HODNOTA";A$;"JE";A
40 END
```

Funkce STR$

Funkce STR$ vrací stringovou reprezentaci číselné hodnoty. Vymažte paměť počítače a napište následující program:

```
10 A=65
20 A$=STR$(A)
30 ? A"JE HODNOTOU";A$
```

Druhá odmocnina - SQR

Funkce SQR se používá pro výpočet druhé odmocniny. Například, abyste nalezli druhou odmoc-

ninu z 50, napište:

? SQR(50) RETURN

Tímto způsobem lze nalézt druhou odmocninu libovolného čísla.

Absolutní hodnota (ABS)

Funkce absolutní hodnota (ABS) se často používá jestliže pracujeme se zápornými čísly. Tato funkce se používá pro obdržení kladné hodnoty libovolného čísla, kladného nebo záporného. Napište následující příklady:

? ABS(-10) RETURN

? ABS(5) "SE ROVNA" ABS(-5) RETURN

Příkazy STOP a CONT

Provádění programu lze přerušit a pak opět pokračovat v bodě, kde došlo k přerušení, jestliže do programu zahrneme příkaz STOP. Příkaz STOP lze použít v libovolném místě programu. Jestliže se počítač zastavil (přerušil provádění programu), lze použít různé příkazy v přímém modu a zobrazit různé komponenty programu. Například lze zjistit jakou hodnotu má "počítač" smyček, nebo libovolná proměnná. Tento rys má mnoho aplikací při ladění programu. Vymažte paměť počítače a napište následující program:

```
10 X=INT(SQR(630))
20 Y=(.025*80)↑2
30 Z=INT(X*Y)
40 STOP
```



```
50 FOR J=0 TO Z STEP Y
60 ?"ZASTAV A POKRACUJ"
70 NEXT
80 END
```

Nyní proveďte program. Počítač napíše "BREAK IN 40" (přerušení na 40). V tomto okamžiku počítač spočetl hodnoty X, Y a Z. Abychom zjistili jak probíhá program, napište PRINT X;Y;Z Používá se to při ladění dlouhého programu (nebo krátkého, ale složitého), abychom se dozvěděli hodnotu nějaké proměnné v určitém místě během provádění programu.

Program bude opět pokračovat po napsání CONT (od continue - pokračuj) a stisknutí RETURN, aniž by se program změnil. Počítač bude pokračovat v provádění příkazu, který se nachází bezprostředně za příkazem STOP.

-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-

Tuto část manuálu a části předchozí jste studovali, abyste se seznámili s programovacím jazykem BASIC a s jeho zvláštnostmi. Následující části této kapitoly popisují vlastní příkazy modu Commodore 128. Některé příkazy modu Commodore 128 nejsou k dispozici v modu C 64. Jiné příkazy Commodore 128 fungují jako některé příkazy C 64, avšak zjednodušeným způsobem. Syntaxi všech příkazů Commodore 7.0 naleznete v Kapitole 5., Encyklopedie BASICu 7.0.

ČÁST 5

Úvod

V této sekci se zavádějí do používánímdalší významné příkazy a instrukce BASICu,.které možná neznají ani zkušení programátoři v BASICu. Při programování v jazyce BASIC narazíme na jisté situace, v nichž tyto příkazy mohou pomoci. V dané části manuálu se vysvětlují postupy, využívané při tvorbě těchto příkazů a uvádějí se příklady na použití každého z těchto příkazů v programu (viz Kapitola 5., Encyklopedie BASICu 7.0, která obsahuje úplný seznam a vysvětlení těchto příkazů a instrukcí). Mimoto v této části manuálu vysvětlíme, jak používat speciální tlačítka, která jsou k dispozici v modu C 128.

Pokročilé využití smyček

Příkaz DO/LOOP

Příkaz DO/LOOP (udělej/smyčka) nabízí další způsob, jak tvořit smyčky, kromě příkazů GOTO, GOSUB a FOR/NEXT. V příkazu DO/LOOP nabízí jazyk BASIC silný a mnohostranný prostředek, který je běžně k dispozici pouze v jazycích pro strukturní programování. V této části manuálu ukážeme některé možnosti použití DO/LOOP. Abychom vytvořili nekonečnou smyčku počínající příkazem DO, zavedeme řádek nebo řádky, specifikující operace, které má počítač provést.

Na konci doplníme příkaz LOOP jako v následující ukázce:

```
100 DO
110 PRINT "OPAKOVANI"
120 LOOP
```

Stiskněte RUN/STOP, aby se program zastavil. Operace uvedené za příkazem DO se budou provádět až do příkazu LOOP (řádek 120); poté se řízení opět předá příkazu DO (řádek 100). Tedy všechny příkazy, nacházející se mezi DO a LOOP, se budou provádět do nekonečna.

Until

Další často používanou metodou je kombinování příkazu UNTIL (až do, dokud) s DO/LOOP. Příkaz UNTIL určuje podmínku pro provádění smyčky. Smyčka se provádí až do okamžiku, kdy dojde ke splnění podmínky UNTIL:

```
100 DO: INPUT "LIBI SE TI TVUJ POCITAC";A$
110 LOOP UNTIL A$="ANO"
120 PRINT "DIKY"
```

Příkaz DO/LOOP se často používá pro opakování vnitřní částu programu, jako v následující ukázce:

```
10 PRINT "PROGRAM POKRACUJE DOKUD NENATISKNES
   STUJ"
20 DO UNTIL A$="STUJ"
30 INPUT "STUPNE FAHRENHEITA";F
```

```
40 C=(5/9)*(F-32)
50 PRINT F;"STUPNU FAHRENHEITA SE ROVNA";C;
   "STUPNU CELSIA"
60 INPUT "POKRACOVAT NEBO SKONCIT";A$
70 LOOP
80 END
```

Příkaz DO/LOOP se rovněž používá jako počítadlo, přičemž příkaz UNTIL určuje počet opakování:

```
10 N=2*2
20 PRINT "DVE KRAT DVE SE ROVNA";N
30 DO UNTIL X=25
40 X=X+1
50 N=N*2
60 PRINT "KRAT DVE";X+1;"KRAT...";N
70 LOOP
80 END
```

Všimněte si, že při vynechání počítací podmínky (UNTIL X=25 v řádku 30) se číslo bude zdvojovat do nekonečna, pokud nedojde k chybě OVERFLOW (příliš velké číslo).

WHILE

Příkaz WHILE (pokud) působí podobně jako UNTIL, avšak smyčka se bude opakovat pokud uvedená podmínka platí, jako v následující ukázce:

```
100 DO: INPUT"LIBI SE TI TVUJ POCITAC";A$
110 LOOP WHILE A$<>"ANO"
120 PRINT "DIKY"
```


Exit

Příkaz EXIT (vyjdi) se používá uvnitř DO/LOOP. Jestliže narazí na příkaz EXIT, program přeskočí na příkaz bezprostředně následující za LOOP.

Výhrada ELSE s IF-THEN

Výhrada ELSE (jinak, v opačném případě) říká počítači co má dělat v případě, že podmínka IF-THEN není splněna. Místo aby začal provádět následující řádek,počítač provede příkaz- nebo přeskok na jiný řádek programu, jak to přikazuje ELSE. Například při tisku čtverce čísla použijte ELSE následujícím způsobem:

```
10 INPUT "UDEJ CISLO PRO UMOCNENI NA DRUHOU";N
20 IF N<100 THEN PRINT N*N: ELSE 40
30 END
40 ?"CISLO MUSI BYT MENSI NEZ 100": GOTO 10
```

Mezi příkazem ELSE a IF-THEN musí být dvojtečka.

Posloupnost BEGIN/BEND s IF-THEN

BASIC 7.0 umožňuje rozšířit možnosti IF-THEN Posloupnost BEGIN/BEND (začátek/konec) umožňuje zahrnout celou řadu řádků programu, které; se provedou, jestliže je splněna podmínka IF, tj. nejen provést jednoduchou operaci či GOTO. Příkaz má následující strukturu:

```
IF podmínka THEN BEGIN:
(řádky programu):
BEND: ELSE atd.
```

Nezapomeňte na dvojtečky mezi BEGIN a jakýmikoliv dalšími příkazy a mezi posledním příkazem bloku a slovem BEND. BEGIN/BEND lze použít bez klausule ELSE nebo s následným použitím ELSE, jako když za THEN následoval jediný příkaz. Napište následující program:

```
10 INPUT A
20 IF A<100 THEN BEGIN: ?"CISLO JE";A
30 SLEEP 2:          REM PAUZA
40 FOR X=1 TO A
50 ?"TOTO JE PRIKLAD BEGIN/BEND"
60 NEXT X
70 ?"FAJN": BEND: ELSE ?"PRILIS VELKE"
80 END
```

Program požaduje zadat číslo. Jestliže je číslo menší než 100, provedou se příkazy uzavřené mezi slovy BEGIN a BEND, včetně příkazů nacházejících se na řádku s BEND (mimo ELSE). Na obrazovce se objeví hlášení "CISLO JE N". Na řádku 30 je smyčka pro pozdržení hlášení na obrazovce po dobu postačující k jeho přečtení. Dále se použije smyčka FOR/NEXT, která zobrazí daný nápis tolikrát, kolikrát požaduje uživatel. Jestliže je tento počet opakování větší než sto, podmínak IF není splněna a provede se příkaz za ELSE (tisk "PRILIS VELKE"). Slovo ELSE <u>se musí</u> nalézat na stejném řádku jako BEND.

<u>Příkaz SLEEP</u>

Všimněte si použití příkazu SLEEP na 30. řádku v uvedeném programu. Příkazem SLEEP (spi) lze jednoduše a přesně zavést pauzu do provádění programu. Příkaz SLEEP má následující tvar:

```
SLEEP n
```

kde n určuje čas ve vteřinách, od 0 do 65535, požadovaného pozdržení. V příkazu na řádku 30. číslice 2 znamená dvouvteřinovou pauzu.

<u>Formátování výstupu</u>

<u>Příkaz PRINT USING</u>

Předpokládejme, že musíme napsat komerční program, který počítá ceny v dolarech. Celkový prodej dělený počtem prodavačů dává průměrnou tržbu. Po provedení tohoto výpočtu obdržíme výsledek se čtyřmi nebo pěti desetinnými místy. Abychom dostali pouze dvě desetinná místa, můžeme "tvarovat" výsledek, pokud počítač tiskne pomocí příkazu PRINT USING. PRINT USING umožňuje zadat tvar výstupu za použití mezer, čárek, desetinných teček a symbolu dolaru $. Symboly čísel (#) reprezentují mezery nebo znaky zobrazeného výsledku. Například:

```
PRINT USING "#$ ######.##";A
```

ukazuje počítači, že při tisknutí A musí použít daný tvar maximálně s 6 číslicemi nalevo

od desetinné tečky a dvěma číslicemi za desetinnou tečkou. Symbol čísla před symbolem dolaru indikuje, že $ se bude pohybovat, takže vždy bude vedle první číslice na levé straně formátu.

Pro zobrazení čárky označující tisíce, jako v $1,000.00 přidejte tuto čárku do příkazu PRINT USING. Další podrobnosti o použití jiných speciálních znaků v PRINT USING najdete v Encyklopedii BASICu.

Příkaz PUDEF

Jestliže je požádována forma výstupu reprezentující rovněž jiné symboly, mimo dolarů a centů, použijte příkaz PUDEF. Čtyři znaky formátu mohou být nahrazeny jedním libovolným znakem na klávesnici.

Příkaz PUDEF má čtyři pozice, ale není nutné predefinovat všechny. Příkaz vypadá následovně:

```
PUDEF "    ,.$"
       1234
```

Zde

- pozice 1 je znak vyplňující. Jestliže tato pozice není definována, objeví se prázdná mezera
- pozice 2 je znak čárky
- pozice 3 je desetinná tečka
- pozice 4 je symbol dolaru.

Například, abychom napsali program, který změní částku v dolarech na libry, musíme formátovat výstup pomocí následujících příkazů:

```
10 PUDEF " ,.£"
20 PRINT USING "# £###### .## ";X
```

Ukázka programu

Tento program počítá úroky a splátky půjčky za použití všech příkazů a instrukcí, které jsme viděli. Zavedeme minimální výši půjčky s použitím příkazu ELSE spolu s příkazem IF-THEN a zavedeme formát dolarů a centů pomocí PRINT USING.

```
10 INPUT "PUJCKA V DOLARECH";A
20 IF A<100 THEN 70: ELSE P=.15
30 I=A*P
40 ?"CELKOVE SPLATKY CINI";
50 PRINT USING "# $ ###### .## ";A+I
60 GOTO 80
70 ?"PUJCKY MENSI NEZ 100 $ NEPOSKYTUJEME"
80 END
```

Zavádění dat příkazem GETKEY

Viděli jsme, jak používat příkazy INPUT a GET při zavádění dat během programu. Další způsob jak zavádět data během provádění programu spočívá v použití příkazu GETKEY. Příkaz GETKEY přijímá pokaždé samotné tlačítko. GETKEY je obecně jedna stringová proměnná, např. A$, jíž bude přiřazeno stisknuté tlačítko. GETKEY se používá, protože umožňuje zavádět data znak po

znaku, aniž by se musel tisknout RETURN za každým znakem. Příkaz GETKEY se používá výlučně v programech.

Příklad na použití GETKEY v programu:

```
1000 PRINT "ZVOLTE A,B,C,D,E NEBO F"
1010 GETKEY A$
1020 PRINT A$;"JE STISKNUTE TLACITKO"
```

Počítač čeká na stisknutí jednoho tlačítka a v tomto okamžiku znak bude přiřazen proměnné A$ a zobrazen řádkem 1020. Následující program ukáže složitější využití a užitečnost GETKEY: pro zvolení jedné z možných odpovědí na jedinou otázku a rovněž pro žádost opakovat otázku. Jestliže daná odpověď není správná, uživatel se může znovu pokusit stisknutím tlačítka "A" (řádek 80.). Tlačítko stisknuté jako odpověď bude přiřazeno proměnné A$, kdežto odpověď "zkus znovu" je přiřazena B$ příkazy GETKEY na řádkách 60. a 90. Příkazy IF/THEN jsou použity ve smyčce v programu, aby bylo dosaženo vhodné reakce počítače v závislosti na různém vstupu z počítače.

```
10 PRINT"KDO NAPSAL HAVRAN?"
20 PRINT "A. EDGAR ELLEN POE"
30 PRINT "B. EDGAR ALLAN POE"
40 PRINT "C. IGOR ALLEN POE"
50 PRINT "D. ROB HAVRAN"
60 GETKEY A$
```



```
70 IF A$="B" THEN 150
80 PRINT "OMYL. ZKUSIS ZNOVU? (A NEBO N)"
90 GETKEY B$
100 IF B$="A" THEN PRINT "A,B,C NEBO D?":GOTO 60
110 IF B$="N" THEN 140
120 PRINT "STISKNI A NEBO N - NOVY POKUS"
130 GOTO 90
140 PRINT "SPRAVNA ODPOVED ZNI B"
145 GOTO 160
150?PRINT "SPRAVNE"
160 END
```

GETKEY se hodně podobá GET až na to, že GETKEY automaticky čeká na stisknutí tlačítka.

Rady jak programovat

V předchozích částech jsme viděli, jak provádět změny v programu, jak opravovat tiskové chyby pomocí INST/DEL. Mimoto je BASIC vybaven dalšími příkazy a funkcemi, které umožňují lokalizovat chyby v programu a příkazy, které usnadňují programování.

Zavádění programů do počítače

Auto

BASIC je vybaven automatickým číslováním, umožňujícím zadat přírustek při číslování řádků. Například, abychom číslovali řádky normálně po desítkách, napište v přímém modu:

AUTO 10 RETURN

Počítač bude automaticky číslovat řádky programu po desítkách. Po stisknutí RETURN se vždy objeví číslo následujícího řádku, přičemž kursor bude na místě, kde se začíná tisknout následující příkaz. Je možné zvolit číslování s libovolným přírustkem, například po 5 po 50 atd. Stačí napsat požadované číslo za slovem AUTO a stisknout RETURN. Abychom zrušili automatické číslování, stačí napsat AUTO bez následujícího čísla a stisknout RETURN.

RENUMBER

Jestliže přidáme kprogramu nějaké příkazy, v číslování řádků může nastat zmatek. Příkazem RENUMBER (přečísluj) můžeme změnit čísla řádků na konstantní přírustek jak v celém programu, tak v jeho části. Příkaz RENUMBER obsahuje různé nepovinné parametry, jak je vidět z následujícího:

RENUMBER [[nový počáteční řádek] [,[přírustek] přírustek] [,starý počáteční řádek]]]]

Nový počáteční řádek bude číslován, jako by to byl řádek programu po zadání příkazu RENUMBER. Jestliže není uvedeno jinak, předpokládá se hodnota 10. Přírustek je hodnota mezi číslem řádku a jiným a rovněž jeho předpokládaná hodnota je 10. Staré číslo počátečního řádku je číslo řádku, od něhož začíná přečíslování. To umožňuje přečíslovat pouze určitou

část programu, aniž by se přečíslovával celý. Bez jeho udání začíná přečíslování prvním řádkem programu. Například

RENUMBER 40,,80

přikazuje počítači přečíslovat program od řádku 80 s přírustkem 10. Řádek 80 se stává řádkem 40.

Všimněte si, že tento příkaz, podobně jako příkaz AUTO, se provádí pouze v přímém modu.

DELETE

Dříve jsme viděli, jak zrušit řádek programu tím, že natiskneme číslo řádku následované RETURN. Taková operace bude zdlouhavá, jestliže potřebujeme zrušit více řádků programu. Příkaz DELETE (vymaž) umožňuje ušetřit čas, protože umožňuje uvést více řádků, které mají být zrušeny najednou. Například

DELETE 10-50

zruší řádky od 10. do 50. DELETE se používá podobně jako LIST; příkaz umožňuje určit pokud až nebo odkud do konce je třeba zrušit řádky programu, případně číslo jediného řádku, který má být zrušen, jako v následujících příkladech:

DELETE -120

ruší všechny řádky do 120. včetně

DELETE 120-

ruší všechny řádky počínaje 120.

DELETE 120

ruší pouze řádek 120.

Identifikování chyb v programu

Jestliže program nepracuje jak by měl, bude obecně hlášena chyba. Protože hlášení jsou poněkud neurčitá, neumožňují identifikovat chybu. Počítač Commodore 128 umožňuje problém lokalizovat.

HELP

Commodore 128 má k dispozici příkaz HELP (pomoc), specifikující řádek, v němž byla objevena chyba. Příkaz HELP se aktivuje tlačítkem HELP v řadě tlačítek nad hlavní klávesnicí. Natiskněte následující příkaz, s obsaženou v něm úmyslnou chybou:

10 ?3;4:5;6

Při provádění tohoto jednořádkového programu počítač vytiskne 3 a 4 následované hlášením "SYNTAX ERROR IN 10" (syntaktická chyba v 10). Předpokládejme, že nejsme sto identifikovat chybu (dvojtečka namísto středníku mezi 4 a 5). Stiskněte tlačítko HELP (nebo napište HELP a stiskněte RETURN). Počítač zobrazí řádek, zdůrazní 5;6 aby indikoval, že chyba byla zjištěna v tomto řádku.

Rozlišení chyb - příkaz TRAP

Normálně jestliže počítač zjistí chybu, program se zastaví. V tomto okamžiku lze stisknout tlačítko HELP, abychom vypátrali chybu, nebo použít příkaz BASICu 7.0 TRAP. Tímto příkazem se do programu zavádí jistá funkce pro zachycení chyb. Příkaz TRAP lokalizuje chybu a oznámí opravu, program pak pokračuje od bodu, kde byl přerušen. Normálně se funkce pro rozlišování chyb zavádí jako první řádek programu:

5 TRAP 100

Tento příkaz sděluje počítači, že má přeskočit na určitý řádek programu (v daném případě řádek 100), jestliže narazí na chybu. Řádek 100 se nachází na konci programu a stanoví jistou podmínku. Tento řádek se neprovádí, pokud se nenarazí na chybu. Po nalezení chyby se příkaz TRAP provede a řízení se předá jiné části programu. Tento příkaz se může použít pro lokalizaci chyb během zavádění dat, pro pokračování v provádění programu nebo pro přechod z grafického modu do textového modu a podobně. Jestliže se provádí například DO/LOOP, který zdvojnásobuje nějaké číslo, aniž by se použil omezující příkaz UNTIL, objeví se nakonec chybové hlášení OVERFLOW (příliš velké

číslo) a program se zastaví. Aby se tomu předešlo, postačí přidat dvě řádky, jeden na začátku a jeden na konci programu, například:

```
  5 TRAP 100
100 IF N>1 THEN END
```

Ačkoliv hodnota N se stává vždy mnohem větší než 1 v celém programu, příkaz se nebere do úvahy dokud se nenarazí na chybu. Jestliže je číslo příliš velké, tj. převýší kapacitu počítače, příkaz TRAP se provede. Jestliže je N větší než 1, program se uzavře (spíše než by se přerušil).

Nyní uvedeme příklad, v němž identifikování chyby bude zobrazeno, aby se zabránilo dělení nulou:

```
10  TRAP 1000
100 INPUT "DELENI LIBOVOLNYM CISLEM. ZADEJ
    DELENE CISLO";D
110 INPUT "KTRYM CISLEM JE TREBA DELIT";B
120 A=D/B
130 PRINT D;"DELENE";B;"SE ROVNA";A
140 END
1000 IF B=0 THEN PRINT "NEMOHU PROVEST TUTO
     OPERACI"
1100 INPUT "ZVOL JINE CISLO";B;RESUME 120
```

Všimněte si příkazu RESUME na řádku 1100, k rý příkazuje počítači vrátit se na uvedený (v daném případě 120.) řádek a pokračovat

V závislosti na nalezené chybě, nové zahájení programu může nebo nemusí být možné.

Další informace e nacházení chyb viz chybové funkce ERR$, EL a ER popsané v Kapitole 5., Encyklopedie BASICu 7.0.

Trasování programu - příkazy TRON a TROFF

Jestliže v programu narazíme na nějaké problémy, nebo jestliže nedostaneme očekávané výsledky, bývá užitečné provést program metodicky - provést přesně ty operace, které provádí počítač. Tento proces se nazývá trasování, stopování. Označte si proměnné a zpracujte hodnoty v souladu s příkazy programu. Proveďte výpočet a vytiskněte výsledek každého příkazu. Trasování může například odhalit, že jste použili příkaz GOTO s chybným číslem řádku, nebo že jste sice spočetli výsledek, ale neuložili jej do správné proměnné. Řada chyb v programu může být zjištěna tak, že počítač bude provádět program po jednom příkazu. C 128 provádí takovou analýzu pomocí příkazů TRON a TROFF. Během provádění programu počítač pomocí příkazu TRON tiskne čísla řádků v pořadí, v němž se provádějí. Tímto způsobem lze nalézt příčinu proč program nedává očekávaný výsledek. Napište libovolný z dosud prováděných programů, nebo váš vlastní výtvor. Abyste aktivovali trasovací mod, napište a proveďte TRON v přímém modu.

Při provádění programu si všimněte, jak se objevují čísla řádků v závorkách dříve než se zobrazí příslušné výsledky. Snažte se sledovat čísla řádků a odhalit, v kterém okamžiku počítač dochází do určitého místa programu. TRON nejzajímavěji pracuje spolu s programem, obsahujícím více skoků, jako jsou řádky obsahující GOTO, GOSUB a IF-THEN. Dříve než budete pokračovat natiskněte TROFF, čímž dezaktivujete trasovací mod.

Není nutné trasovat celý program a je možné zahrnout TRON přímo do programu jako řádek bezprostředně předcházející té části programu, která působí problémy, a TROFF jako řádek za koncem této části. Při provádění se budou objevovat v závorkách psané pouze čísla řádků programu mezi TRON a TROFF.

Vytváření oken

Okno je specifikovaná část obrazovky, kterou uživatel definuje jako pracovní prostor, oblast. Všechno, co bude tištěno (řádky, programy atp) po zavedení okna se objeví uvnitř tohoto okna, aniž by to vyšlo z jeho mezí. Commodore 128 poskytuje dvě možnosti vytvářet okna: příkaz WINDOWN(okno) a funkce tlačítka ESC.

Využití příkazu WINDOW k vytvoření okna

V jazyku BASIC 7.0 Commodoru 128 je k dispo-

zici příkaz, umožňující vytvářet a ovládat okna::příkaz WINDOW. Tento příkaz mán následující tvar:

WINDOW sloupec nahoře vlevo, řádek nahoře vlevo, sloupec dole vpravo, řádek dole vpravo [,mazání - nepovinné]

Prvá dvě čísla za WINDOW udávají číslo sloupce a řádku vytvářejících levý horní roh okna; následující dvě čísla jsou souřadnice pravého dolního rohu. Uvědomte si, že tyto souřadnice musí souhlasit s typem formátu obrazovky (40 nebo 80 sloupců). Mimoto lze přidat nepovinný parametr pro mazání. Jestliže připojíte na konci příkazu 1, plocha okna na obrazovce bude vymazána, jako v následující ukázce:

WINDOW 10,10,20,20,1

V následujícím programu bude na obrazovce vytvořena čtyři okna, jak ve formátu 40 sloupců, tak ve formátu 80 sloupců:

```
10 SCNCLR: COLOR 5,5 :REM CI TI OBRAZOVKU A VOLI
PURPUROVY TEXT
20 COLOR 5,5
30 WINDOW 0,0,39,24 :REM OKNO NA CELE OBRAZOVCE
40 COLOR 0,13: COLOR 4,13 :REM SVETLE ZELENA
OBRAZOVKA 40 SL
50 A$="ABCDEFGHIJKLMNOPQRST"
60 COLOR 6,6 :REM VOLI PURPUROVY TEXT
```

```
70 FOR I=1 TO 25  :REM ZAPLNI OBRAZOVKU TEXTEM
80 PRINT A$;A$: NEXT I
90 WINDOW 1,1,7,20  :REM DEFINUJE OKNO 1
100 COLOR 5,3  :REMTISKNE AS INVERSNIM RUDYM
TEXTEM
110 PRINT CHR$(18);AS
120 WINOOW 15,15,39,20,1 :REM DEFINUJE OKNO 2
130 COLOR 5,7  :REM TISKNE A$ MODRYM TEXTEM
140 FOR I=1 TO 6: PRINT A$;: NEXT I
150 WINDOW 30,1,39,22,1: REM DEFINUJE OKNO 3
160 COLOR 5,8: LIST  :REM LIST ZLUTYM TEXTEM
170 WINDOW 5,5,33,18,1 :REM DEFINUJE OKNO 4
180 COLOR 5,2  :REM TISKNE A$ A LIST BILE
190 PRINT A$: LIST
200 END
```

Použití zlačítka ESC k vytvoření okna

Abycho zřídili okno pomocí tlačítka ESC, řiďte se následujícím:

1. Přeneste kurcor do bodu obrazovky, kde bude horní levý roh okna
2. Stiskněte a pusťte tlačítko ESC, dále stiskněte T
3. Přeneste kurcor do bodu obrazovky, kde bude dolní pravý roh okna
4. Stiskněte a pusťte ESC, dále stiskněte B. Okno je otevřeno.

Tlačítko ESC se používá na změnění okna a taxtu

v něm obsaženého. Modifikování obrazovka, jako například zavedení a zrušení textu, posun a změny rozměrů okna, mohou být provedeny stisknutím ESC a některého dalšího tlačítka. Stiskněte a pusťte ESC a poté stiskněte jedno z uvedených níže tlačítek:

@ Vymaže všechno to, co se nalézá za kursorem do konce okna

A Mod automatického vkládání

B Definuje pravý dolní roh okna (v okamžité poloze kursoru)

C Ruší mod automatického vkládání

D Ruší běžný řádek

E Uvede kursor do modu bez blikání

F Uvede kursor do modu s blikáním

G Aktivuje akuetickou signalizaci (tlačítka Control-G)

H Deaktivuje akuatickou aignalizaci

I Vkládá řádek

J Vetup začátku běžného řádku

K Vetup konečného běžného řádku

L Aktivuje "ekluz" obrazovky

M Deaktivuje "skluz" obrazovky

N Vrací k normálnímu zobrazení (neinverení obrazovka)(pouze v modu 80 sloupců)

Ó Deaktivuje mod vkládání v "podílu"

P Vymaže vše počínaje začátkem řádku a konče polohou kursoru

Q Vymaže vše počínaje kursorem do konce řádku

R Inverze obrazovky (pouze v modu 80 sloupců)

S Přemění kursor na čtvereček (■)

T Definuje horní levý okraj okna (v okamžité poloze kursoru)

U Přemění kursor na podtržení (_)

V Probíhá obrazovku ve směru horního řádku

W Probíhá obrazovku ve směru dolního řádku

X Přechod od 40 k 80 sloupcům

Y Obnoví tabulační body

Z Zruší tabulační body

Zkuste použít funkce tlačítka ESC. Pravděpodobně zjistíte, že některé funkce jsou užitečnější než jiné. Pamatujte, že lze rovněž použít normální tlačítko INST/DEL k modifikování textu uvnitř okna.

Po zavedení okna bude celá obrazovka obsahovat obdélník, který jsme definovali. Abyste vymazali plochu za oknem, stiskněte SHIFT současně s CLEAR/HOME . Okno vymažete stisknutím CLEAR/HOME dvakrát. Okno tak bude vymazáno a kursor se bude nacházet v levém horním rohu obrazovky. Okno se používá obzvláště, jestliže zapisujeme, prohlížíme a provádíme programy, v nichž je dovoleno pracovat v určité části obrazovky, ponechávajíc beze změny její ostatní části.

Operace na 2 MHz

Příkazy FAST a SLOW

Operační mod 2 MHz umožňuje provádět negrafické programy ve formátu 80 sloupců dvojnásobkem normální rychlosti. Abychom přešli od normálního k rychlému modu, používáme příkazy FAST (rychlý) a SLOW (pomalý).

Příkaz FAST převede Commodore 128 do modu 2 MHz. Příkaz má tvar

FAST

Příkaz SLOW převádí Commodore 128 do modu 1 MHz. Příkaz má tvar

SLOW

Tlačítka používaná samostatně v modu C 128

Funkční tlačítka

Čtyři tlačítka napravo nahoře na klávesnici Commodoru 128, nad k číselnou klávesničkou, jsou speciální funkční tlačítka, umožňující šetřit časem při provádění často opakovaných úkonů, za použití jediného tlačítka. První tlačítko je označeno F1/F2, druhé F3/F4, třetí F5/F6 a poslední F7/F8. Při použití funkcí F1, F3, F5 a F7 stiskneme tlačítko samotné, při použití funkčních tlačítek 2,4,6 a 8 podržte SHIFT současně s funkčním tlačítkem.

Přehled standartních funkcí všech tlačítek:

F1	F2	F3	F4
GRAPHIC	DLOAD"	DIRECTORY	SCNCLR
F5	F6	F7	F8
DSAVE"	RUN	LIST	MONITOR

Vysvětlíme nyní funkci každého tlačítka:

Tlačítko 1 aktivuje grafický mod, jestliže uvedeme číslo grafické oblasti a stiskneme RETURN. Příkaz GRAPHIC je nutný, aby bylo možné udělovat grafické příkazy, například CIRCLE nebo PAINT. Další podrobnosti viz Část 6.

Tlačítko 2 tiskne DLOAD" na obrazovce. Pro vyvolání programu, místo abyste vyťukávali DLOAD" celé, stačí stisknout tlačítko F2, napsat jméno programu" a stisknout RETURN

Tlačítko 3 natiskne seznam fajlů na disku, který je vložen do diskdrajvu

Tlačítko 4 vymaže obrazovku pomocí příkazu SCNCLR

Tlačítko 5 natiskne DSAVE" na obrazovce. Pro uložení programu na disk stačí doplnit jméno programu a stisknout RETURN

Tlačítko 6 provede program nacházející se v počítači

Tlačítko 7 zobrazí program nacházející se v počítači

Tlačítko 8 umožňuje přístup k monitoru strojového kódu. Viz Doplněk J, popisující monitor.

Předefinování funkčních tlačítek

Libovolné z těchto tlačítek lze předefinovat nebo naprogramovat tak, aby provádělo požadované operace. Předefinování lze jednoduše provést pomocí příkazu KEY. Tlačítko lze předefinovat v jazyce BASIC v programu nebo modifikovat v libovolném okamžiku v přímém modu. Příkladem, kdy je užitečné předefinování, je časté používání nějakého příkazu, který bychom jinak museli pokaždé vyťukávat celý. Nová definice anuluje v počítači starou. Lze předefinovat tolik tlačítek, kolik je zapotřebí a tolikrát, kolikrát je to nutné.

Abychom přeprogramovali např. funkční tlačítko F7 tak, aby provádělo například přechod z textového modu do grafického modu s vysokým rozlišením nebo multibarevné grafiky, použijeme následující příkaz:

```
KEY 7,"GRAPHIC0"+CHR$(13)
```

CHR$(13) je znak (v kódu ASCII) RETURN. Po předefinování se po stisknutí tlačítka F7 dostane automaticky nápis příkazu GRAPHIC 0 a a bude automaticky zaveden do počítače příkazem RETURN. Jedinému tlačítku lze tedy přiřadit celý příkaz, nebo dokonce řadu příkazů.

Další tlačítka používaná samostatně v modu C 128

Help

Jak jsme již uvedli, jestliže dojde v programu k chybě, počítač odpoví chybovým hlášením. Tato chybová hlášení jsou podrobně vysvětlena v Doplňku A tohoto manuálu. Abycho obdrželi pomoc při identifikování chyby, použijeme tlačítko HELP (pomoc!). Po zobrazení chybového hlášení stiskneme tlačítko HELP, abychom přesně lokalizovali, kde došlo k chybě.
Po stisknutí HELP se na obrazovce objeví řádek, obsahující chybu v inverzním modu (formát 40 sloupců) nebo podtržený (formát 80 sloupců).
Například:
? SYNTAX ERROR IN LINE 10 hlášení na obrazovce
HELP stiskněte HELP
10 PRONZ "COMMODORE COMPUTER"
Řádek obsahující chybu je zobrazen inverzně ve formátu 40 sloupců nebo podtrženě ve formátu 80 sloupců.

No Scroll

Stiskněte toto tlačítko, abyste zastavili svislý pohyb textu, jestliže kurzor dosáhl spodní části obrazovky. Tím zastavíte "klouzání" textu až do opětovného stisknutí tlačítka NO SCROLL.

Caps Lock

Pomocí tohoto tlačítka lze tisknout všechna velká písmena, aniž bychom museli pokaždé používat tlačítko SHIFT. Tlačítko CAPS LOCK aktivuje velká písmena (nebo horní polohu) jestliže je stiskneme a deaktivuje je jestliže je stiskneme podruhé. Toto tlačítko se používá pouze s abecedními tlačítky.

Zobrazení 40/80 sloupců

Tlačítko 40/80 volí formát hlavní obrazovky: 40 nebo 80 sloupců. Zvolený formát zobrazuje všechna hlášení a výstupy od okamžiku zapnutí nebo jestliže stiskneme tlačítko RESET nebo RUN/STOP/RESTORE. Toto tlačítko se používá pro stanovení formátu zobrazení pouze před zapnutím nebo při novém uvedení počítače do chodu. Tímto tlačítkem není možné měnit mod po zapnutí počítače, ledaže se použije tlačítko RUN/STOP/RESET nebo RUN/STOP/RESTORE.
Část 8. obsahuje podrobné vysvětlení modu 40/80 sloupců.

Alt

Tlačítko ALT umožňuje přiřadit část programu jedné speciální funkci některého tlačítka nebo řady tlačítek.
Jestliže tlačítko ALT bude stisknuto spolu s nějakým jiným tlačítkem, nebude to mít žádný následek , ledaže toto tlačítko bylo

předefinováno prostřednictvím specifického aplikačního programu.

Tab

Toto tlačítko funguje podobně, jako tabulátor psacího stroje a používá se k zavedení nebo zrušení tabulačních bodů na obrazovce a pro přemístění kurzoru na sloupec, na němž byl zaveden tabulační bod.

Line Feed

Toto tlačítko umožňuje přemístit kurzor na následující řádek, jako se používá tlačítko kursoru pro pohyb dolů.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

V této části manuálu jsme vysvětlili pouze některé koncepce, tlačítka a příkazy, které dělají z Commodoru 128 speciální prostředek. Další podrobnosti a vysvětlivky k jazyku BASIC naleznete v Encyklopedii BASICu 7.0 v Kapitole 5.

ČÁST 6

GRAFIKA

V modu C 128 jazyk BASIC 7.0 Commodoru 128 poskytuje nové a účinné příkazy a instrukce, usnadňující grafické programování. Oba formáty, které jsou k dispozici v modu C 128 /40 a 80 sloupců/, jsou řízeny čipem zvláštního mikroprocesoru. Čip pro 40 sloupců budeme nazývat dohližitelem videointerfejsu neboli VIC /z anglického Video Interface Controller/. Čip pro 80 sloupců nazýváme 8563. Čip VIC disponuje 16 barvami a umožňuje realizovat grafiku s vysokou přesností, nazývanou bodová matrice. Čip 8563 pro 80 sloupců pracuje rovněž s 16 barvami, avšak zobrazuje pouze znaky a grafické znaky. Z tohoto důvodu všechny grafické programy s vysokou přesností v modu C 128 musíme provádět ve formátu 80 sloupců.

Grafické prostředky

Commodore 128 poskytuje následující grafické prostředky v modu C 128:

- 13 příkazů pro speciální grafiku
- 16 barev
- šest různých způsobů zobrazení
- osm pohyblivých programovatelných obrazců, nazývaných sprajt /SPRITE - skřítek, víla/
- kombinované zobrazení grafika/text

Všechny tyto prostředky umožňují vytvořit grafický systém, který je mnohostranný a snadno

se používá.

Přehled příkazů

Nyní stručně vysvětlíme grafické příkazy:

BOX : kreslí obdélníky na obrazovce s bodovou matricí

CHAR : zobrazuje znaky na obrazovce s bodovou matricí

CIRCLE : kreslí kruhy, elipsy a jiné geometrické obrazce

COLOR : volí barvu rámu, popředí, pozadí a znaků na obrazovce

DRAW : zobrazuje čáry a body na obrazovce s bodovou matricí

GRAPHIC : volí typ zobrazení /text, bodová matrice na celé nebo na části obrazovky/

PAINT : vybarvuje plochy na obrazovce s bodovou matricí

SCALE : zavádí relativní rozměry obrazců na obrazovce s bodovou matricí

SPRDEF : aktivuje mod pro definování sprajtů, aby bylo možné měnit tvar sprajtu

SPRITE : aktivuje, vybarvuje a určuje prioritu obrazovky pro sprajty a mění rozměry sprajtů

SPRSAV : ukládá stringovou proměnnou do části paměti určené pro sprajty

SSHAPE : ukládá do paměti obraz části obrazovky s bodovou matricí do stringové proměnné.

Většina těchto příkazů je pojasněna na příkladech v této části manuálu. Další informace o tvaru a funkci grafických příkazů, včetně těch,

o nichž daná část manuálu nepojednává, naleznete v kapitole 5., Encyklopedie BASICu 7,0.

Grafické programování na C 128

V následující části manuálu se fáze za fází popisují příklady grafického programování. Během seznamování s každým grafickým příkazem doporučujeme psát i program,který budeme postupně vytvářet v této části manuálu. Na konci dané části bude vytvořen úplný grafický program.

Volba barev

Prvním krokem každé části grafiského programu je volba barev pozadí, popředí a rámu obrazovky. Abyste zvolili barvu napište:

COLOR zdroj, barva

kde zdroj je část obrazovyky, která má být vybarvena /pozadí, popředí, rám atd/ a barva je kód barvy zdroje. V tabulce 6-1 jsou uvedena čísla zdrojů, v tabulce 6-2 čísla barev ve formátu 40 sloupců a v tabulce 6-3 čísla barev ve formátu 80 sloupců.

Číslo	Zdroj
0	Barva pozadí ve 40 sloupcích
1	Popředí grafické obrazovky /VIC/
2	Barva 1. popředí multibarevné obrazovky /VIC/
3	Barva 2. popředí multibarevné obrazovky /VIC/

4 Rám 40 sloupců /VIC//v modu textovém nebo grafickém/

5 Barva znaků na obrazovce textové s 40 nebo 80 sloupci

6 Barva pozadí v 80 sloupcích /8563/

Tabulka 6-1. Čísla zdrojů

Kód barvy	Barva	Kód barvy	Barva
1	černá	9	oranžová
2	bílá	10	hnědá
3	červená	11	světlečervená
4	tmavomodrá	12	tmavošedá
5	purpurová	13	šedá
6	zelená	14	světlezelená
7	modrá	15	světlemodrá
8	žlutá	16	světlešedá

Tabulka 6-2. Čísla barev ve formátu 40 sl.

Kód barvy	Barva	Kód barvy	Barva
1	černá	9	tmavopurpurová
2	bílá	10	kaštanová
3	tmavočervená	11	světlečervená
4	světlemodrá	12	tmavomodrá
5	světlepurpurová	13	šedá
6	tmavozelená	14	světlezelená
7	tmavomodrá	15	světlemodrá
8	světležlutá	16	světlešedá

Tabulka 6-3. Čísla barev ve formátu 80 sl.

Typy zobrazení na obrazovce

C 128 poskytuje různé typy zobrazení informací na obrazovce; parametr "zdroj" příkazu COLOR se vztahuje na různé typy zobrazení. Typy zobrazení na obrazovce se podrozdělují na tři kategorie:

Do první kategorie patří zobrazení textu, které zobrazuje jednotlivé znaky, jako písmena, čísla, zvláštní znaky a grafické znaky, uvedené na přední straně téměř všech tlačítek C 128. C 128 zobrazuje text jak ve formátu 40 sloupců, tak ve formátu 80 sloupců.

Druhá kategorie zobrazení se používá pro grafiku s vysokým rozlišením, jako jsou ilustrace a komplikované kresby. Tento typ zobrazení zahrnuje standartní mod bodové matrice a mnohobarevný mod bodové matrice. Mod bodové matrice umožňuje ovládat libovolný bod obrazovky, naboli pixel /prvek zobrazení/. To umožňuje dosáhnout extrémní přesnosti ilustrací a jiných grafických prací prováděných s počítačem ve formátu 80 sloupců.

Rozdíl mezi textovým modem a modem bodové matrice spočívá ve způsobu, jak obrazovka posílá a ukládá informace. Textová obrazovka řídí celé znaky, každý z nichž zaujímá plochu 8 x 8 pixelů na obrazovce. V modu bodové metrice na-

opak ovládáme každý jednotlivý pixel na obrazovce.

Třetí typ zobrazení, členěná obrazovka, je kombinací prvních dvou modů. V zobrazení členěné obrazovky se část obrazovky zobrazuje v textovém modu a část v modu bodové matrice /standartní nebo multibarevné/. To je možné díky tomu, že C 128 používá dvě rozdílné části paměti počítače pro uchování dvou obrazovek: jedna část je určena protext a druhá pro grafiku.

Napište následující krátký program:

```
10 COLOR C,1   : REM BARVA VLOZENEHO TEXTU=CERNA
20 COLOR 1,3   : REM BARVA POPREDI OBRAZOVKY
                 S BODOVOU MATRICI = CERVENA
30 COLOR 4,1   : REM BARVA RAMECKU = CERNA
```

V tomto případě se vybarví černě vložky, červeně popředí a černě rámeček.

Volba grafického modu

Další fází grafického programování je volba vhodného grafického modu. Ta se volí pomocí příkazu GRAPHIC, který má následující tvar:

GRAPHIC <mod[,c][,s]/CLR>

kde mod je číslo od 0 do 5, c je 0 nebo 1 a s má hodnotu od 0 do 25. V tabulce 6-4 jsou uvedeny hodnoty, odpovídající grafickému modu.

Mod	Popis
0	Standartní text, 40 sloupců
1	Standartní bodová matrice

Mod	Popis
2	Standartní bodová matrice /dělená obrazovka/
3	Bodová matrice multibarevná
4	Bodová matrice multibarevná /dělená obrazovka/
5	Text o 80 sloupcích

Tabulka 6-4. Grafické mody

Parametr CRL znamená vymazání. V tabulce 6-5 jsou vysvětleny hodnoty spojené s CRL.

Hodnota C	Popis
0	Nemaže grafickou obrazovku
1	Maže grafickou obrazovku

Tabulka 6-5. Parametry CLR

Při prvním provedení programu, jestliže je nutné vymazat předem grafickou obrazovku, uveďte jako hodnotu parametru c v příkazu GRAPHIC číslo 1. Při druhém provedení, pro udržení obrazu na obrazovce, aniž byste museli pokaždé znovu kreslit, uveďte jako parametr c 0. Parametr s specifikuje počátek textové obrazovky v modu dělené obrazovky /textová obrazovka začíná číslem řádku následujícím za specifikovaným řádkem/. Vynechání parametru s znamená volbu grafické obrazovky s dělenou obrazovkou /čís. 2 nebo 4/, přičemž část obrazovky pro text se bude nacházet v části od 20. do 25. řádku, zbylá část obrazovky je zaplněna

bodovou matricí. Parametr s umožňuje změnit počáteční řádek textové obrazovky na libovolný řádek obrazovky, od 1 do 25. Nulová hodnota parametru s znamená, že obrazovka není dělená a je čistě textová,

Posledním parametrem příkazu GRAPHIC je CLR. Jestliže se grafický příkaz bodové matrice objeví poprvé, Commodore 128 rezervuje 9 K pro informace z obrazovky s bodovou matricí. 8 K je rezervováno pro údaje bodové matrice a 1 K pro údaje o barvě /videomatrice/. Poněvadž 9 K tvoří v paměti značně rozsáhlý blok, lze jej použít znovu pro jiné cíle programu. Účelem CLR tedy je reorganizovat paměť Commodoru 128 a dát znovu k dispozici 9 K paměti, která byla dříve použita pro obrazovku s bodovou matricí. Formát CLR je následující:

GRAPHIC CLR

Jestliže použijete tento tvar, vynechte všechny ostatní parametry příkazu GRAPHIC.

Přidejte k programu následující příkaz. Tento příkaz uvede C 128 do standartního modu bodové matrice a zajistí pro obrazovku s bodovou matricí 8 K /a 1 K pro údaje o barvě/ pro tvorbu grafiky.

40 GRAPHIC 1,1

Druhá 1 v tomto příkazu vymaže obrazovku bodové matrice. Aby se obrazovka nevymazala, zaměňte druhou 1 číslem 0./nebo ji úplně vynechte/.

POZNÁMKA: Jestliže se nedaří přejít z modu bodové matrice k textové obrazovce, stiskněte současně tlačítko RUN/STOP a RESTORE, nebo stiskněte tlačítko ESC následované X, abyste přešli k obrazovce s 80 sloupci. Ačkoliv s čipem VIC /40 sloupců/ je možné pouze zobrazovat grafiku, je nicméně možné psát grafické programy ve formátu 80 sloupců. Jestliže vlastníte duální monitor Commodore 1901 a chcete zobrazit grafický program ve fázi provádění, zvolte výstup 40 sloupců přenesením přerušovače kursoru na výstup o 40 sloupcích.

Zobrazování grafiky na obrazovce

Zvolili jsme tedy grafický mod a požadované vybarvení. Nyní začneme zobrazovat grafiku na obrazovce. Začneme zobrazením kruhu.

Kružnice - příkaz CIRCLE

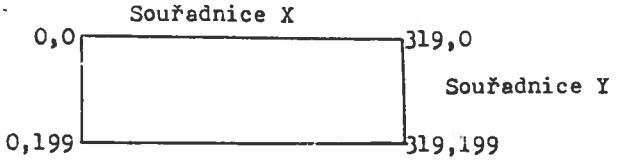
Abychom zobrazili kruh, použijeme příkaz CIRCLE jak následuje:

60 CIRCLE 1,160,100,40,40

Tímto příkazem zobrazíme kružnici ve středu obrazovky. Příkaz CIRCLE ovládají nové parametry, které se vplí, aby se vytvářely různé typy uzavřených křivek a geometrických obrazců. Například změnou čísel v příkazu na řádku 60. do-

staneme kružnice různého průměru nebo změněného tvaru /např. ovál/. Příkaz CIRCLE propůjčuje sílu a mnohostrannost programování grafiky v BASICu na Commodoru 128. Význam čísel v příkazu CIRCLE je vysvětlen v paragrafu věnovaném příkazu CIRCLE v Kapitole 5., Encyklopedie BASICu 7.0.

Na obrazovce Commodoru 128 se bod, v němž X=0 a Y=0 nachází v horním levém rohu obrazovky a nazývá se polohou HOME /doma/. V normální geometrii se obvykle za bod s X a Y rovným nule považuje levý dolní roh grafu. Na obr. 6-6 jsou zavedeny souřadnice obrazovky X /vodorovná/ a Y /svislá/ a čtyři rohové body obrazovky C 128.



Obr. 6-6. Uspořádání souřadnic X a Y

Význam číselných parametrů v příkazu CIRCLE:

- 1 zdrojová barva /zde popředí/
- 160 je počáteční souřadnice X /vodorovné/
- 110 je počáteční souřadnice Y /svislá/
- 40 je poloměr

Obdélníky - příkaz BOX

Nyní nakreslíme obdélník; napište:

```
80 BOX 1,20,60,100,140,0,1
```

Tento příkaz nakreslí nalevo od kruhu. Vysvětlení významu parametrů v příkazu pro vytváření obdélníků naleznete v Kapitole 5., Encyklopedie BASICu 7.0. Příkaz BOX obsahuje 7 parametrů, které lze volit a měnit tak, abychom vyrobili různé typy obdélníků. Změňte barvu popředí a nakreslete obrysy obdélníku napravo od kruhu pomocí následujícího příkazu:

```
90 COLOR 1,9 :REM MENI SE BARVA POPREDI
100 BOX 1,220,60,300,140,0,0
```

Příkaz box vytváří různé kombinace obdélníků a čtverců.

Čáry, body a různé obrazce - příkaz DRAW

Dosud jsme viděli, jak zvolit grafický mod a barvy a jak zobrazit na obrazovce kruhy a čtverce. Další grafický příkaz DRAW /kresli/ umožňuje kreslit čáry na obrazovce stejně, jako to děláme tužkou na listu papíru. Následující příkaz umožní nakreslit čáru pod čtverci a kruhem:

```
120 DRAW 1,20,180 TO 300,180
```

Abychom čáru zrušili změňte zdroj /1/ na 0. v příkazu DRAW. Na čáru se naloží barva pozadí a způsobí její zmizení. Zkuste použít jiné souřadnice a jiné zdroje, abyste se důvěrně

seznámili s příkazem DRAW.
Příkaz DRAW se může použít v jiném tvaru, v němž bude možné kreslit přímku a měnit její směr. Zkuste například následující příkaz:
130 DRAW 1,250,0 TO 30,0 TO 40,40 TO 250,0
Příkaz narýsuje trojúhelník v horní části obrazovky. Čtyři skupiny čísel reprezentují souřadnice X a Y tří vrcholů trojúhelníku. Všimněte si, že prvé a poslední souřadnice jsou stejné, aby se kresba trojúhelníku uzavřela v tomtéž bodě, kde začala. Tento typ příkazu DRAW umožňuje kreslit téměř všechny geometrické obrazce, například lichoběžníky, rovnoběžníky a mnohoúhelníky.
Příkaz DRAW se používá ve třech tvarech. Například abychom nakreslili bod, specifikujte pouze počáteční X a Y, například:
150 DRAW 1,160,160
Tento příkaz nakreslí bod pod kruhem.
Jak jsme viděli, příkaz DRAW má přizpůsobivý charakter, umožňující vytvářet na obrazovce obrazce, čáry, body a prakticky neomezený počet kreseb.

Vybarvení vymezených ploch - příkaz PAINT

Příkaz DRAW umožňuje kreslit kontury ploch na obrazovce. Abychom vybarvili plochu uvnitř nakreslené čáry, použijeme příkaz PAINT /maluj/.

Tento příkaz skutečně umožňuje vyplnit vymezenou plochu barvou. Stejně jako malíř natírá plátno, příkaz PAINT pokrývá plochy na obrazovce 16 možnými barvami. Napište například:
160 PAINT 1,150,97
Řádek 160 vybarví kruh, který byl nakreslen řádkem 50. Příkaz PAINT vyplní určitou plochu, dokud nenarazí na zvláštní hranici; v souladu s uvedenou zdrojovou barvou. Jestliže Commodore 128 přeruší barvení, ponechá bodový kursor v bodě, v němž vybarvování skončilo /tj. bod 150,97/. Dále uvedeme jiné příkazy PAINT:
180 PAINT 1,50,25
200 PAINT 1,225,125
Řádek 180 vybarví trojúhelník a řádek 200 vybarví čtverec.
Užitečná rada: Jestliže příkaz PAINT zvolil počáteční bod, který začne vybarvovat v souladu se zdrojovou barvou, Commodore 128 nenatře tuto plochu. Musí zvolit počáteční bod, který se nachází zcela uvnitř obvodu obrazce, který je zapotřebí vybarvit. Počáteční bod nesmí být na obvodové čáře. Čísla zdroje souřadnic obrazovky a souřadnice specifikované v příkazu PAINT se musí vzájemně lišit.

Zobrazení znaků na obrazovce s bodovou matricí - příkaz CHAR

Až dosud jsme používali standartní mod bodové matrice. Tento mod používá část paměti, která se zcela liší od té části paměti, která se používá pro textový mod (mod v němž se zavádějí programy a texty). Aktivovat mod bodové matrice a snažit se poté natisknout jiné znaky se zcela míjí účinkem. Je to proto, že znaky, které se snažíme vytisknout, musí být zobrazeny na textové obrazovce, ne na obrazovce s bodovou matricí, která je v daném okamžiku aktivována. Někdy však je nutné zobrazit znaky na obrazovce s bodovou matricí během vytváření diagramů a grafiky. Příkaz CHAR byl vytvořen výslovně pro tyto účely. Abychom zobrazili standartní znaky na obrazovce s bodovou mazricí, používáme příkaz CHAR následovně:

```
220 CHAR 1,11,24,"PRIKLAD GRAFIKY"
```

Příkaz zobrazí Text "PRIKLAD GRAFIKY" počínaje ne řádku 25 sloupcem 12. Příkaz CHAR se mimoto používá v textovém modu, přestože byl vymyšlen pro obrazovku s bodovou matricí.

Vytvoření grafického programu

Viděli jsme, jak používat různé grafické příkazy. Nyní napíšeme program a podíváme se, jak různé příkazy fungují. Příkazy barvy na řádcích 70, 110, 140, 170, 190 a 210 jsme přidali, aby každý obrazec měl jinou barvu.

```
10 COLOR 0,1   :REM VOLBA BARVY POZADI
20 COLOR 1,3   :REM VOLBA BARVY POPREDI
30 COLOR 4,1   :REM VOLBA BARVY RAMECKU
40 GRAPHIC 1,1:REM MOD BODOVE MATRICE
60 CIRCLE 1,160,100,40,40 :REM KRUZNICE
70 COLOR 1,6   :REM ZMENA BARVY POPREDI
80 BOX 1,20,60,100,140,0,1 :REM CTVEREC
90 COLOR 1,9   :REM ZMENA BARVY POPREDI
100 BOX 1,220,60,300,140,0,0 :REM CTVEREC
110 COLOR 1,8   :REM ZMENA BARVY POPREDI
120 DRAW 1,20,180 TO 300,180: REM USECKA
130 DRAW 1,250,0 TO 30,0 TO 40,40,TO 250,0:
        REM TROJUHELNIK
140 COLOR 1,15   :REM ZMENA BARVY POPREDI
150 DRAW 1,160,160   :REM KRESLI BOD
160 PAINT 1,150,97 :REM VYBARVI KRUH
170 COLOR 1,5   :REM ZMENA BARVY POPREDI
180 PAINT 1,50,25   :REM VYBARVI TROJUHELNIK
190 COLOR 1,7   :REM ZMENA BARVY POPREDI
200 PAINT 1,225,125: REM VYBARVI CTVEREC
210 COLOR 1,11   :REM ZMENA BARVY POPREDI
```

```
220 CHAR 1,11,24,"UKAZKA GRAFIKY" :REM TEXT
230 FOR I=1 TO 5000 : NEXT:GRAPHIC 0,1:
    COLOR 1,2
```

Popis jednotlivých fází programu:

- řádek 10 až 20 volí příslušně barvy pozadí, popředí a rámečku
- řádek 40 volí grafický mod
- řádek 60 zobrazí kruh
- řádek 80 nakreslí barevný čtverec
- řádek 100 kreslí obvod čtverce
- řádek 120 kreslí úsečku dole na obrazovce
- řádek 130 kreslí trojúhelník
- řádek 150 vytvoří bod pod kružnicí
- řádek 160 vybarví kruh
- řádek 180 vybarví trojúhelník
- řádek 200 vybarví čtverec
- řádek 220 natiskne "UKAZKA GRAFIKY" ve spodní části obrazovky
- řádek 230 zadrží počítač v modu, v němž obrázky zůstanou zobrazeny po určitou dobu na obrazovce a pak jej vrátí do textového modu s černou barvou znaků

Jestliže chcete, aby obrazce zůstaly na obrazovce, vynechte příkaz GRAPHIC na řádku 230.

Změna rozměrů grafických obrazců - příkaz SCALE

Commodore 128 ovládá další grafický příkaz, které poskytuje velké možnosti samotnému grafickému systému. Příkaz SCALE (měřítko) umožňuje zvětšovat nebo zmenšovat grafická zobrazení na obrazovce. Příkaz SCALE má i další účel, který bude vysvětlen níže.

Ve standartním modu bodové matrice má obrazovka o 40 sloupcích 320 vodorovných souřadnic a 200 svislých souřadnic. Ve vícebarevném modu bodové matrice má obrazovka o 40 sloupcích rozlišení pouze poloviční oproti standartnímu modu bodové matrice, tj. 160x200. Tato redukce rozlišení je kompenzována skutečností, že můžeme používat další barvy až do tří barev uvnitř matrice znaků 8x8. Standartní mod bodové matrice zobrazuje pouze dvě barvy v matrici znaků 8x8. Jestliže použijeme příkaz SCALE, standartní mod bodové matrice a vícebarevná bodová matrice mají vzájemně úměrné souřadnice. Měřítko jde od nuly do maximálně 1023 vodorovných souřadnic. To platí nezávisle na tom, zda se nalézáme v standartním nebo vícebarevném modu bodové matrice.

Funkci SCALE vyvoláme příkazem:

```
SCALE 1,x,y
```

Dostaneme tak souřadnice obrazovky od 0 do 65535 platné jak v standartním modu, tak ve větším vícebarevném rozlišení. Abychom deaktivovali SCALE, napište:

SCALE 0

Souřadnice se vrátí k normálním hodnotám.

Abyste viděli účinek SCALE v programu, přidejte k němu řádek:

50 SCALE 1,500,500

a proveďte program.

Další podrobnosti o příkazu SCALE naleznete v Kapitole 5.

Poznámka: SCALE se používá po GRAPHIC a nemá vliv na CHAR.

Nyní uvedeme několik ukázkových programů, používajících grafické příkazy:

```
10 COLOR 0,1
20 COLOR 1,8
30 COLOR 4,1
40 GRAPHIC 1,1
50 FOR I=80 TO 240 STEP 10
60 CIRCLE 1,I,100,75,75
70 NEXT I
80 COLOR 1,5
90 FOR I=80 TO 250 STEP 10
100 CIRCLE 1,I,100,50,50
110 NEXT I
```



```
120 COLOR 1,7
130 FOR I=50 TO 280 STEP 10
140 CIRCLE 1,I,100,25,25
150 NEXT I
160 FOR I=1 TO 7500: NEXT
170 GRAPHIC 0,1: COLOR 1,2
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
10 GRAPHIC 1,1
20 COLOR 0,1
30 COLOR 4,1
40 FOR I=1 TO 50
50 Z=INT(((RND(1))*16)+1)*1
60 COLOR 1,Z
70 X=INT(((RND(1))*30)+1)*10
80 Y=INT(((RND(1))*20)+1)*10
90 U=INT(((RND(1))*30)+1)*10
100 V=INT(((RND(1))*20)+1)*10
110 DRAW 1,X,Y TO U,V
120 NEXT I
130 SCNCLR
140 GOTO 40
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
10 COLOR 4,7: COLOR 0,7: COLOR 1,1
20 GRAPHIC 1,1
30 FOR I=400 TO 1 STEP -5
40 DRAW 1,150,100 TO I,1
50 NEXT I
```

```
60 FOR I=1 TO 400 STEP 5
70 DRAW 1,150,100 TO 1,I
80 NEXT I
90 FOR I=40 TO 320 STEP 5
100 DRAW 1,150,100 TO I,320
110 NEXT I
120 FOR I=320 TO 30 STEP -5
130 DRAW 1,150,100 T  320,I
140 NEXT I
150 FOR I=1 TO 7500: NEXT I
160 GRAPHIC 0,1: COLOR 1,1
```

Natiskněte tyto ukázky, proveďte je a uložte pro další použití. Jeden z nejlepších způsobů, jak se naučit programovat, je studium ukázkových programů, abyste pochopili, jak různé příkazy provádějí své operace. Tímto způsobem lze snadno použít grafické příkazy k vytváření vlastních obrázků na Commodore 128.

Další podrobnosti o instrukcích nebo příkazech BASIC naleznete v Encyklopedii BASICu 7.0 v Kapitole 5.

Nyní máme k dispozici řadu grafických příkazů, které umožňují vytvářet téměř neomezené množství grafických zobrazení. Grafické možnosti Commodoru 128 však tím nejsouv vyčerpány. Commodore 128 nabízí mimoto celou řadu příkazů, nazývaných grafické SPRITE (sprajty -

skřítci, víly), které umožňují rychlé, jednoduché a efektivní vytváření a ovládání grafických obrazců. Tyto příkazy umožňují vytvářet na vysoké úrovni "sprajty" - pohyblivé obrazce. C 128 obsahuje vhodné operace, včetně definování sprajtů (SPRDEF). Tyto příkazy představují zcela novou technologii vytváření a oživování sprajtů. Čtěte následující část, kde získáte základní ponětí o animování pomocí počítače.

Sprajty: programovatelné animované obrázky

Viděli jsme, jak používat grafické funkce Commodoru 128, jak používat nejprve grafické příkazy na vysoké úrovni ke kreslení kruhů, čtverců, přímek a bodů , a dále vybarvovat obrazovku, aktivovat grafický mod, vybarvovat předměty na obrazovce a měnit úměrně rozměry. Nyní se seznámíme s následující fází grafického programování: oživování sprajtů. Ti, kdo používali Commodore 64 jistě slyšeli mluvit o sprajtech; pro ty, kteří naopak nejsou důvěrně seznámeni s tímto termínem řekneme, že sprajt je oživlý obrázek vytvořený uživatelem. Sprajty mohou být vybarveny 16 různými barvami nebo mohou být vícebarevné. Tyto obrázky mohou být posouvány po obrazovce.

Uvedeme seznam příkazů, které vysvětlíme v zázo části manuálu:

MOVSPR
SPRDEF
SPRITE
SPRSAV
SSHAPE

Vytváření sprajtů

Nejprve musíme popsat tvar sprajtu. Například, předpokládejme, že chceme nakreslit vesmírnou loď nebo závodní automobil. Dříve než vybarvíme a oživíme sprajt, musíme definovat jeho obraz. V modu C 128 lze sprajty vytvářet třemi různými způsoby:

1. Za použití příkazu SPRITE v programu
2. Pomocí definováno sprajtu (SPRDEF)
3. Pomocí metod používaných v Commodoru 64

Použití příkazu SPRITE v programu

Tato metoda, na rozdíl od ostatních dvou, používá vestavěné příkazy a některé grafické příkazy uvedené v předchozí části, jak to ukáže následující obecná procedura. Další podrobnosti budou přibývat v průběhu této části manuálu.

1. Nakreslete obrázek pomocí grafických příkazů vysvětlených v předchozí části manuálu, jako například DRAW, CIRCLE, BOX a PAINT. Rozměry obrázku musí být 24 pixelů krát 21 pixelů výšky. v standartním modu bodové m matrice nebo 12 pixelů šířky krát 21 pixelů výšky ve vícebarevném modu bodové matrice.
2. Použijte příkaz SSHAPE pro uložení dat obrazce do stringové proměnné.
3. Přeneste data obrazce ze stringové proměnné do sprajtu pomocí příkazu SPRSAV.
4. Aktivujte sprajt, vybarvěte jej, zvolte standartní nebo vícebarevný mod a roztáhněte obrazec, to vše pomocí příkazu SPRITE.
5. Pohybujte sprajtem pomocí příkazu MOVSPR.

Kreslení sprajtů

Nyní popíšeme příkazy, které provádějí operace týkající se sprajtů. Než skončí tato část manuálu, budeme mít první program sprajtů, který můžeme provést a uložit pro další použití. Nejprve nakreslíme na obrazovce obrazec (24 krát 21 pixelů) pomocí DRAW, CIRCLE, BOX a PAINT. Tento příklad bude proveden ve standartním modu bodové matrice s pozadím v černé barvě. Příkazy pro zavedení grafického modu a černého zbarvení pozadí jsou následující:

```
 5 COLOR 0,1 :REM VYBARVI POZADI CERNE
10 GRAPHIC 1,1 :REM ZAVADI STANDARTNI MOD BO
     DOVE MATRICE
```

Následujícími příkazy se nakreslí obrázek závodního automobilu v levém horním rohu obrazovky. Příkazy byly vysvětleny v předchozí části manuálu.

```
 5 COLOR 0,1: COLOR 4,1: COLOR 1,2 :REM STANO
   VI SE BARVY
10 GRAPHIC 1,1 :REM GRAFICKY MOD S VYSOKYM ROZ
      LISENIM
15 BOX 1,2,2,45,45 :REM KRESBA RAMECKU
20 DRAW 1,17,10 TO 28,10 TO 26,30 :REM KORPUS
      VOZIDLA
22 DRAW TO 19,30 TO 17,10 :REM KORPUS VOZIDLA
24 BOX 1,11,10,15,18 :REM HORNI LEVE KOLO
26 BOX 1,30,10,34,18 :REM HORNI PRAVE KOLO
28 BOX 1,11,20,15,28 :REM DOLNI LEVE KOLO
30 BOX 1,30,20,34,28 :REM DOLNI PRAVE KOLO
32 DRAW 1,26,28 TO 19,28 :REM
34 BOX 1,20,14,26,18,90,1 :REM SEDADLO VOZU
36 BOX 1,150,35,195,40,90,1 :REM BILE CARY
38 BOX 1,150,135,195,140,90,1 :REM BILE CARY
40 BOX 1,150,215,195,220,90,1 :REM BILE CARY
42 BOX 1,50,180,300,195 :REM CILOVA PASKA
44 CHAR 1,18,23,"C I L" :REM ZOBRAZI CIL
```

Proveďte program. Nakreslí se bílý závodní automobil uzavřený v obdélníku v levém horním rohu obrazovky. Mimoto se zobrazí závodní dráha s cílem v dolní části obrazovky. V tomto okamžiku je vozidlo pouze nehybným obrazcem

na obrazovce a není ještě sprajtem. První fáze programování sprajtu skončila: obrázek je hotov.

<u>Uchování údajů o sprajtu pomocí SSHAPE</u>

Následující fází je "uložení" obrazce do stringového řetězce pomocí následujícího příkazu:

```
45 SSHAPE A$,11,10,34,30 :REM ULOZENI OBRAZKU
      DO STRINGU A$
```

Příkaz SSHAPE (tvar S(prajtu)) uloží do paměti počítače obrázek na obrazovce (seskupení bitů) - do stringové proměnné, která bude později použita podle určených souřadnic obrazovky. Čísla 11,10,34,30 jsou souřadnice obrázku. Jestliže by tyto souřadnice nebyly přesně uvedeny, příkaz SSHAPE by nemohl správně uložit údaje o obrázku do stringové proměnné A$. Jestliže příkaz SSHAPE bude umístěn do prázdného místa obrazovky, datový string zůstane prázdný, jak bychom okamžitě viděli při zpětné přeměně stringu na sprajt. Zajistíme se proti tomu tím, že použijeme příkaz SSHAPE s přesnými souřadnicemi. Mimoto, prověřte správnost rozměrů každého sprajtu, které musí být 24 pixelů šířky krát 21 pixelů výšky. Příkaz SSHAPE převede obrázek závodního vozidla na tvar datového řetězce (stringu), s nímž počítač zachází jako s údaji o tomto obrázku. Datový řetěz

nul a jedniček, které reprezentují obrázek na obrazovce. Jako v případě všech grafických funkcí, počítač má možnost reprezentovat vzezření grafiky pomocí bitů v paměti. Každému bodu obrazovky, nazývanému pixel, odpovídá v paměti počítače jeden bit, který tento bod ovládá. V standartním modu bodové matrice je bit v paměti počítače 1 (aktivní), jestliže je pixel na obrazovce aktivován. Jestliže se kontrolní bit v paměti rovná 0 (neaktivovaný) je i pixel neaktivovaný.

Uchování údajů o obrázku ve sprajtu

Obrázek je nyní uložen ve stringu. Následující fází programu bude převést údaje o obrázku z datového stringu (A$) do oblasti sprajtových dat tak, aby bylo možné sprajt aktivovat a oživit. Příkazem pro tuto operaci je:
SPRSAV (uložení sprajtu).

```
50 SPRSAV A$,1 :REM ULOZENI STRINGOVYCH DAT
     DO SPRAJTU CISLO 1
55 SPRSAV A$,2 :REM ULOZENI STRINGOVYCH DAT
     DO SPRAJTU CISLO 2
```

Údaje o obrázku byly převedeny do sprajtů č. a 2. Oba sprajty obsahují stejná data a mají tudíž přesně stejný vzhled. Sprajty nemohou být ještě zobrazeny, protože musí být teprve aktivovány.

Aktivování sprajtu

Příkaz SPRITE aktivuje jeden určitý sprajt (číslovaný od 1 do 8), vybarví jej, specifikuje jeho prioritu na obrazovce, změní rozměry a určí typ zobrazení sprajtu. Priorita obrazovky stanoví, zda sprajt prochází před nebo za předměty na obrazovce. Sprajty mohou být zvětšeny až na dvojnásobek svého původního rozměru jak v šířce, tak ve výšce. Typ zobrazení sprajtu určuje, jestli sprajt je standartní sprajt v bodové matrici nebo vícebarevný sprajt v bodové matrici. Nyní uvedeme dva příkazy, aktivující sprajty č. 1 a 2:

```
60 SPRITE 1,1,7,0,0,0,0 :REM AKTIVUJE SPRAJT 1

65 SPRITE 2,1,3,0,0,0,0 :REM AKTIVUJE SPRAJT 2
```

Význam čísel v příkazu SPRITE je následující:

SPRITE #,A,C,P,X,Y,M

#	číslo sprajtu (od 1 do 8)
A	aktivní (A=1) nebo neaktivní (A=0)
C	barva (od 1 do 16)
P	priorita, jestliže P=0 sprajt se pohybuje <u>před</u> předměty na obrazovce
X	X=1 zvětší se šířka sprajtu (X) X=0 sprajt zachová šířku (X)
Y	Y=1 zvětší se výška sprajtu (Y) Y=0 sprajt zůstane normálně vysoký
M	M=1 sprajt bude vícebarevný M=0 sprajt bude standartní

Jak uvidíme, příkaz SPRITE je velmi účinný, protože umožňuje ovládat většinu vlastností sprajtu.

Pohyb sprajtů pomocí MOVSPR

Nyní ukážeme jak pohybovat se sprajty po obrazovce. Příkaz MOVSPR (pohyb sprajtu) řídí pohyb sprajtu a umožňuje animování na obrazovce. Příkaz MOVSPR se používá dvěma způsoby. Za prvé, příkaz MOVSPR přesně umístí sprajt na obrazovce zadáním vodorovné a svislé souřadnice. Přidejte k programu následující příkazy:

```
70 MOVSPR 1,240,70 :REM UMISTI SPRAJT 1 -
        X=240, Y=70
80 MOVSPR 2,120,70 :REM UMISTI SPRAJT 2 -
        X=120, Y=70
```

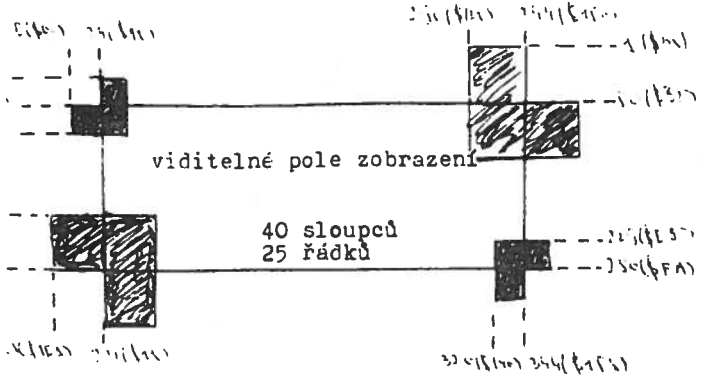
Řádek 70 umístí sprajt č. jedna v bodě 240,70. Řádek 80 umístí sprajt č. 2 v bodě 120,70.; Příkaz MOVSPR se rovněž používá pro přemísťování sprajtů z jejich původní polohy na obrazovce. Například umístíme sprajty 1 a 2 v bodech uvedených v řádcích 70 a 80. Abychom je přemístili z původní polohy do jiného bodu na obrazovce, použijeme následující příkazy:

```
85 MOVSPR 1,180#6 :REM SPRAJT SE POHYBUJE Z
      HORNI CASTI OBRAZOVKY DO DOLNI
87 MOVSPR 2,180#7 :REM SPRAJT SE POHYBUJE Z
      HORNI CASTI OBRAZOVKY DO DOLNI
```


Prvním číslem v těchto příkazech je číslo sprajtu. Druhé číslo je směr /ve směru hodinových ručiček/ ve stupních vzhledem k původní poloze sprajtu. Symbol # značí, že sprajt se pohybuje pod určitým úhlem a určitou rychlostí z původní polohy, až do absolutní polohy, uvedené v řádcích 70,a 80. Poslední číslo určuje rychlost, s níž se sprajt pohybuje po obrazovce. Tato rychlost se vyjadřuje čísly od 0 do 15. Příkaz MOVSPR má dva alternativní tvary. Podrobnosti naleznete v Encyklopedii BASICu v Kapitole 5. Sprajty používají souřadnice, zcela odlišné od souřadnic bodové matrice. Souřadnice bodové matrice se mění v rozmezí od 0,0 /levý horní roh/ do 319, 199 /pravý dolní roh/. Viditelné souřadnice sprajtu začínají bodem 50,24 a končí bodem 250, 344. Zbývající část souřadnic sprajtu jsou mimo obrazovku a nejsou vidět, a sprajty se proto pohybují rovněž v těchto souřadnicích. Místa vně pole, v němž je sprajtům dovoleno snadno se pohybovat uvnitř a vně obrazovky. Na obr. 6-7 jsou ukázány souřadnice sprajtu a viditelné polohy sprajtu.

Nyní proveďte celý program. Napsali jsme tedy první program: automobilové závody se dvěma vozidly. Zkuste přidat více automobilů a více obrázků na obrazovku. Nakreslete jiné sprajty a více obrázků na obrazovce. Zobrazte jinou

tvořit více "filmů" oživených počítačem.



Obrázek 6-7. Viditelné souřadnice sprajtů

Nyní zastavte sprajt tím, že stisknete současně RUN/STOP a RESTORE.

Vytváření programu sprajtů

Vytvořili jsme tedy následující program se sprajty:

```
5  COLOR 0,1: COLOR 4,1: COLOR 1,2 :REM BARVY
10 GRAPHIC 1,1  :REM VOLBA GRAFICKEHO MODU
15 BOX 1,2,2,45,45 :REM RAMECEK
20 DRAW 1,17,10 TO 28,10 TO 26,30 :REM SKELET
   AUTOMOBILU
22 DRAW 19,30 TO 17,10 :REM SKELET AUTOMOBILU
24 BOX 1,11,10,15,18 :REM  HORNI LEVE KOLO
26 BOX 1,30,10,34,18 :REM HORNI PRAVE KOLO
28 BOX 1,11,20,15,28 :REM DOLNI LEVE KOLO
```



```
32 DRAW 1,26,28 TO 19,28 :REM KRESBA
34 BOX 1,20,14,26,18,90,1 :REM SEDADLO
36 BOX 1,150,35,195,40,90,1 :REM BILE CARY
38 BOX 1,150,135,195,140,90,1 :REM BILE CARY
40 BOX 1,150,215,195,220,90,1 :REM BILE CARY
42 BOX 1,50,180,300,195 :REM CILOVA CARA
44 CHAR 1,18,23,"C I L" :REM NAPIS CIL
45 SSHAPE A$,11,10,34,30 :REM ULOZENI OBRAZKU
      DO A$
50 SPRSAV A$,1  :REM ULOZENI A$ DO SPRAJTU 1
55 SPRSAV A$,2  : REM ULOZENI A$ DO SPRAJTU 2
60 SPRITE 1,1,7,0,0,0,0  :REM SPUSTENI SPRAJTU1
65 SPRITE 2,1,3,0,0,0,0 :REM SPUSTENI SPRAJTU 2
70 MOVSPR 1,240,70  :REM SPRAJT 1 X=240,Y=70
80 MOVSPR 2,120,70  :REM SPRAJT 2 X=120,Y=70
85 MOVSPR 1,180#6  :REM POHYB SPR 1 DOLU
87 MOVSPR 2,180#7  :REM POHYB SPR 2 DOLU
90 FOR I=1 TO 5000: NEXT I
99 GRAPHIC 0,1
```

Následuje přehled různých fází programu:

- řádek 5 barví obrazovku načerno
- řádek 10 zavádí standartní grafický mod s vysokým rozlišením
- řádek 15 kreslí čtverec v horním levém rohu obrazovky
- řádek 20-32 kreslí závodní automobil
- řádek 35-40 kreslíddráhy a cíl
- řádek 45 přenáší údaje o obrázku vozidla do stringové proměnné

- řádek 50 a 55 přenášejí obsah stringové proměnné do sprajtů 1 a 2
- řádek 60 a 65 aktivují sprajty 1 a 2
- řádek 70 a 80 umístí sprajty v horní části obrazovky
- řádek 85 a 87 oživí sprajty - dva automobily se pohybují směrem k cíli

V této části manuálu jsme osvětlili, jak vytvářet sprajty pomocí grafických příkazů zabudovaných do C 128, jako například DRAW a BOX, a jak ovládat sprajty pomocí příkazů Commodoru 128. VCommodoru 128 jsou k dispozici ještě dvě jiné možnosti jak vytvářet sprajty: zaprvé zabudovaná funkce definování sprajtu /SPRDEF/, kterou popíšeme v následujících odstavcích; za druhé možnost využívaná v Commodoru 64. Obraťte se na Průvodní zprávu programátora C 64, kde naleznete podrobnosti o tomto způsobu vytváření sprajtů.

Definování sprajtů - příkaz SPRDEF

Commodore 128 má zabudovaný mod pro definování sprajtů, umožňující vytvářet sprajty na Commodoru 128. V Commodoru 128 například musíte mít na tuto operaci přídavný editor sprajtů, nebo kreslit sprajt na milimetrový papír a pak kodifikovaný sprajt zavádět do počítače pomocí příkazů READ, DATA a POKE. Nový příkaz Commodoru 128 SPRDEF definující sprajt umožňuje

vytvářet a měnit sprajty ve specifické pracovní části paměti.

Abyste aktivovali mod SPRDEF napište:

SPRDEF

a stiskněte RETURN. Commodore 128 zobrazí na obrazovce rastr /mřížku/ pro sprajt. Kromě toho počítač ohlásí:

SPRITE NUMBER? /číslo sprajtu?/

Stiskněte některé z čísel od 1 do 8. Počítač znovu vyplní rastr a zobrazí příslušný sprajt v pravém horním rohu obrazovky. Od této chvíle budeme rastr sprajtu nazývat pracovní oblastí. Pracovní oblast je široká 24 znaků a vysoká 21 znaků. Každé poloze znaku uvnitř pracovní oblasti odpovídá pixel sprajtu, protože sprajt sestává z 24 pixelů našíř a 21 pixelů na výšku. V pracovní oblasti jsou v modu SPRDEF k dispozici různé příkazy. Uvedeme souhrn těchto příkazů:

Seznam příkazů v modu definice sprajtu

Tlačítko CLR - vymaže vnitřek pracovní oblasti

Tlačítko M - aktivuje/deaktivuje vícebarevný sprajt

CTRL 1-8 - volba barvy popředí sprajtu /od 1 do 8/

Cx 1-8 - volba barvy popředí sprajtu /od 1 do 8/

Tlačítko 1 - aktivuje pixel v barvě pozadí

Tlačítko 2 - -"- -"- -"- popředí

Tlačítko 3 - aktivuje plochy v multikoloru 1

Tlačítko 4 - -"- -"- -"- 2

Tlačítko A - aktivuje/deaktivuje automatický pohyb kursoru

Tlačítka CRSR - pohybují kursorem /+/ uvnitř pracovní oblasti

RETURN - přesouvá kursor na začátek dalšího řádku

Tlačítko HOME - přemístí kursor do levého horního rohu pracovní oblasti

Tlačítko X -ovládá vodorovné rozšíření

Tlačítko Y - ovládá svislé rozšíření

Shift RETURN - ukládá do paměti sprajt v pracovní oblasti a zobrazí znovu nápis SPRITE NUMBER?

Tlačítko C - kopíruje jeden sprajt na jiný

Tlačítko STOP - deaktivuje zobrazený sprajt a zobrazí znovu nápis SPRITE NUMBER?, aniž by se přitom sprajt změnil

Tlačítko RETURN - /po zobrazení nápisu SPRITE NUMBER?/ - výstup z modu SPRDEF

<u>Procedura vytváření sprajtů v modu definice sprajtů /SPRDEF/</u>

Obecná procedura jak vytvářet sprajty v modu definice sprajtu, je následující:

1. Vymaž pracovní oblast současným stisknutím tlačítek SHIFT a CLR/HOME
2. Jesliže chceš mít vícebarevný sprajt, stiskni tlačítko M. Vedle normálního kursoru se objeví další kursor. Dvojice kursorů je dána tím, že vícebarevná modalita aktivuje dva pixely na každý jednotlivý pixel standartního modu. To je důvod, proč vícebarevný mod má pouze poloviční vodorovné rozlišení v porovnání s modem s vysokým rozlišením.
3. Zvol barvu sprajtu. U barev od 1 do 8 podrž tlačítko CONTROL současně s tlačítkem od 1 do 8. U barev od 9 do 16. podrž tlačítko Cx a stiskni jedno z tlačítek od 1 do 8.
4, Jsme připraveni začít tvořit tvar sprajtu. Číselná tlačítka od 1 do 4 vyplňují sprajt a dodávají mu tvar. Pro sprajt v jediné barvě používejte tlačítko 2, aby se zaplnila jedna pozice znakem uvnitř pracovní oblasti. Stiskněte tlačítko 1, jestliže chcete zrušit něco, co bylo načrtnuto pomocí tlačítka 2. Jestliže chcete pokaždé zaplnit znakem <u>jedinou</u> pozici, stiskněte tlačítko A. Nyní musíte pohybovat kursorem ručně, pomocí tlačítek pro pohyb kursoru. Jestliže chcete přemísťovat kursor automaticky směrem doprava, aniž byste tiskli tlačítko ovládající kursor, netiskněte tlačítko A, protože toto tlačítko je určeno pro automatický posun kursoru. Po zaplnění nějaké pozice v pracovní oblasti znakem vidíte, že odpovídající pixel zobrazovaného sprajtu je aktivován.

Ve vícebarevném modu tlačítko 3 zaplňuje dvě znakové pozice v pracovní oblasti barvou multikolor 1 a tlačítko 4 zaplňuje dvě znakové pozice multikolorem 2.
Je možné deaktivovat pozici deaktivujíc pixel barvou pozadí plochu zaplněnou uvnitř pracovní oblasti stisknutím tlačítka 1. Ve vícebarevném modu tlačítko 1 deaktivuje dvě znakové pozice při jednom stisknutí.

5. Během fáze vytváření sprajtu je možné volně se pohybovat uvnitř pracovní oblasti, aniž by se přitom aktivovaly nebo deaktivovaly pixely, pomocí tlačítek RETURN, HOME a tlačítek pro přemístění kursoru.
6. Sprajty je možné roztahovat v libovolném okamžiku jak vodorovně, tak svisle. Pro svislé roztažení stiskněte tlačítko Y. Pro vodorovné - tlačítko X. Abyste se vrátili k normálním rozměrům zobrazení, stiskněte znovu X a Y.
Jestliže se totéž tlačítko používá pro aktivování a deaktivování nějaké operace, pak je nazýváme bistabilní. Tlačítka X a Y operují v tomto modu při vodorovném a svislém rozšiřování sprajtů.
7. Jestliže jsme skončili kreslit sprajt a je definitivně dokončen jeho tvar, potřebujeme sprajt uložit: podržte tlačítko SHIFT a

stiskněte tlačítko RETURN. Commodore 128 uloží údaje o sprajtu do vhodné oblasti paměti. Sprajt zobrazený v pravém horním rohu obrazovky bude deaktivován a dále bude zobrazen nápis SPRITE NUMBER? Jestliže chcete vytvořit další sprajt, uveďte jeho číslo a tvořte nový sprajt přesně stejně, jako v prvém případě. Abyste znovu dostali na obrazovku původní sprajt v pracovní oblasti, natiskněte číslo tohoto sprajtu. Abyste vyšli z modu definování sprajtu, stiskněte tlačítko RETURN jako odpověď na nápis SPRITE NUMBER?
8. Pro kopírování jednoho sprajtu do jiného použijte tlačítko C.
9. Jestliže nechcete uložit sprajt do paměti, stiskněte tlačítko STOP. Commodore 128 deaktivuje zobrazený sprajt a zobrazí hlášení SPRITE NUMBER?
10. Abyste vyšli z modu definování sprajtu, stiskněte tlačítko RETURN když je na obrazovce zobrazen nápis SPRITE NUMBER, bez čísla sprajtu za ním. Je možné vyjít z tohoto modu, jestliže se nacházíme v jedné z následujících situací:
okamžitě po uložení sprajtu /shift RETURN/
okamžitě po stlačení tlačítka STOP.

Po vytvoření sprajtu a vyjití z modu definování sprajtu jsou údaje o sprajtech uchovány v příslušné části paměti Commodoru 128. Abychom sprajt zobrazili, musíme jej aktivovat, protože se nacházíme v normálním BASICu. Abychom sprajt aktivovali, použijeme příkaz SPRITE předvedený dříve. Například, vytvořili jsme sprajt 1 v modu SPRDEF. Abycho tento sprajt aktivovali v modu BASIC, vybarvili jej modře a roztáhli jak vodorovně, tak svisle, použijeme následující příkaz:

SPRITE 1,1,7,0,1,1,0

Dále použijeme příkaz MOVSPR pro pohyb sprajtu:

MOVSPR 1,45#5

To je vše, co se týká modu SPRDEF. Především je nutné vytvořit sprajt, uložit údaje o sprajtu a přejít z modu SPRDEF do BASICu, dále aktivovat sprajt pomocí příkazu SPRITE a pro pohyb sprajtu použít příkaz MOVSPR. Po skončení programové fáze uložte data sprajtu do binárního fajlu příkazem BSAVE:

BSAVE "jmeno fajlu",B0,P3584 TO P4096

Pro nové přivolání dat sprajtu z disku přečteme binární fajl uložený pomocí BSAVE, pomocí

BLOAD "jméno fajlu"[,B0,P3584]

BLOAD uloží data na adrese, z níž byla uložena na disk.

Přiblížili jsme si metody tvorby sprajtů: 1/ SSHAPE, SPRSAV, SPRITE, MOVSPR, 2/ mod SPRDEF. Používejte tyto metody, abyste se zdokonalili v animování sprajtů.

Další informace naleznete v Ukládání dat sprajtů do binárních fajlů.

Spojování sprajtů

Až dosud jsme si přiblížili jak tvořit, vybarvovat, aktivovat a oživovat sprajty. Může se naskytnout potřeba vytvořit obrazec, který je příliš členitý nebo příliš velký, aby mohl být vměstnán do jediného sprajtu. V tomto případě je možné spojit dva nebo více sprajtů, takže obrazec bude větší a podrobnější, než by mohl být při použití jediného sprajtu. Sprajty se přitom mohou pohybovat také nezávisle jeden na druhém, umožňujíc tak lepší ovládání animace. V této části manuálu uvedeme příklad spojení dvou sprajtů. Obecná procedura /algoritmus/ napsání programu s dvěma spojenými sprajty je následující:

1. Nakreslete na obrazovce obrázek pomocí grafických příkazů Commodoru 128 jako jsou DRAW, BOX a PAINT, přesně jako jsme to provedli v programu automobilových závodů v

v předešlé části manuálu. Tentokrát však vytvoříme obrazec o dvojnásobném rozměru, než má jeden sprajt, tj, 48 x 21 pixelů.

2. Požijte dva příkazy SSHAPE a uložte sprajty do dvou zvláštních datových stringů. Souřadnice prvého příkazu SSHAPE zvolte v poli 24 x 21 pixelů v první půlce nakresleného obrazce. Poté souřadnice druhého příkazu SHAPE zvolte v druhém poli 24 x 21 pixelů. Dejte pozor, avyste uložili data dvou obrázků do dvou různých stringů. Například první příkaz SSHAPE uloží první půlku obrazce do A$, zatímco druhý příkaz SSHAPE uloží druhou půlku obrazce do B$.
3. Převeďte data obrazce z každého stringu do zvláštního sprajtu pomocí příkazu SPRSAV.
4. Aktivujte každý sprajt pomocí příkazu SPRITE.
5. Umístěte oba sprajty tak, aby první sprajt začínal pixelem sousedícím s pixelem jímž končí druhý sprajt. Tím budou oba sprajty spojeny. Například nakreslete obrázek 48 x 21 pixelů. Umístěte první sprajt /definovaný jako 1. například/ do souřadnice 10,10 následujícím příkazem:

   100 MOVSPR 1,10,10

   kde první číslo je číslem sprajtu, druhé číslo je vodorovná /X/ a třetí číslo svislá /Y/ souřadnice. Umístěte druhý sprajt o 24 pixelů vpravo od sprajtu č. 1:

   200 MOVSPR 2,34,10

   V tomto okamžiku budou oba sprajty zobrazeny těsně vedle sebe a budou mít vzhled obrazce původně nakresleného programem pomocí příkazů DRAW, BOX a PAINT.
6. Dále pomocí příkazu MOVSPR je možné pohybovat oběma sprajty společně nebo dvěma různými směry. Jak jsme ukázali v předchozí části manuálu, příkaz MOVSPR umožňuje pohybovat sprajty směrem k určitému bodu obrazovky nebo v poloze, úměrné původní poloze sprajtu.

Program, který následuje, je příkladem spojení dvou sprajtů. Program vytváří mimozemskou scenerii. Kreslí hvězdy, planetu a kosmickou loď podobnou Apollu. Loď je nakreslena a poté uložena do dvou ~~sprajtů~~ datových stringů A$ a B$. Přední část lodě, kabina, bude uložena do sprajtu 1, zatímco zadní část lodě, raketa, bude uložena do sprajtu 2. Loď překříží obrazovku dvakrát. Jelikož její rychlost je malá a loď je velmi vzdálena od Země, musí se uvést v činnost zadní rakety, aby loď zamířila směrem k Zemi. Po druhém překřížení obrazovky budou spuštěny zadní rakety, které budou

pohánět loď směrem k Zemi.

Výpis programu:

```
 5 COLOR 4,1: COLOR 0,1: COLOR 1,2 :REM VOLBA
       BAREV
10 GRAPHIC 1,1    : REM MOD S VYSOKYM ROZLISENIM
17 FOR I= 1 TO 40
18 X=INT(RND(1)*320)+1
19 Y=INT(RND(1)*200)+1
21 DRAW 1,X,Y : NEXT I :REM HVEZDY
22 BOX 0,0,5,70,40,,1 :REM VYMAZE OBDELNIK
23 BOX 1,1,5,70,40: COLOR 1,8 :REM OBDELNIK PRO
       KOSMICKOU LOD
24 CIRCLE 1,190,90,35,25: PAINT 1,190,95:REM
     KRESLI A VYBARVI PLANETU
25 FOR I=90 TO 96 STEP 3: CIRCLE 1,190,I,65,10:
     NEXT I
26 DRAW 1,10,17 TO 16,17 TO 32,10 TO33,20 TO 32,
     30 TO 16,23 TO 10,23 TO 10,17
28 DRAW 1,19,24 TO 20,21 TO 27,25 TO 26,28 :REM
    SPODNI OKNO
35 DRAW 1,20,19 TO 20,17 TO 29,13 TO30,18 TO28,
      23 TO20,19 :REM HORNI OKNO
38 PAINT 1,13,20 :REM VYBARVI KOSM LOD
40 DRAW 1,34,10 TO 36,20 TO 34,30 TO 45,30 TO 46-
      20 TO 45,10 TO 34,10
42 DRAW 1,45,10 TO 51,12 TO 57,10 TO 57,17 TO
      51,15 TO 46,17 :REM RAKETA 1
43 DRAW 1,46,22 TO 51,24 TO 57,22 TO 57,29 TO
      51,27 TO 45,29 :REM RAKETA 2
44 PAINT 1,40,15: PAINT 1,47,12: PAINT 1,47,26:
      DRAW 0,45,30 TO 46,20 TO 45,10
45 DRAW 0,34,14 TO 44,14: DRAW 0,34,21 TO 44,21:
      DRAW 0,34,28 TO 44,28
47 SSHAPE A$,10,10,33,32: REM ULOZI SPRITE A$
48 SSHAPE B$,34,10,57,32 :REM ULOZI SPR DO B$
50 SPRSAV A$,1  :REM DATA SPRAJTU 1
55 SPRSAV B$,2  :REM DATA SPRAJTU 2
60 SPRITE 1,1,3,0,0,0,0  :REM SPRAJT 1 CERVENY
65 SPRITE 2,1,7,0,0,0,0  :REM SPRAJT 2 MODRY
82 MOVSPR 1,150,150  :REM POLOHA SPRAJTU 1
83 MOVSPR 2,172,150  :REM POLOHA SPRAJTU 2
85 MOVSPR 1,270#5  :REM OZIVI SPRAJT 1
87 MOVSPR 2,270#5  :REM OZIVI SPRAJT 2
90 FOR I=1 TO 5000: NEXT I
92 MOVSPR 1,150,150  :REM NAVRAT POLOHY
93 MOVSPR 2,174,150
95 MOVSPR 1,270#10  :REM ROZDELI SPRAJTY 1 A 2
96 MOVSPR 2,90#5
97 FOR I=1 TO 1200: NEXT I
98 SPRITE 2,0
99 FOR I=1 TO 5000: NEXT I
100 GRAPHIC 0,1  :REM NAVRAT K TEXTOVEMU MODU
```

Přehled jednotlivých fází programu:

- řádek 5 vybarví černě pozadí a bíle popředí
- řádek 10 volí standartní mod s vysokým rozlišením a vymaže obrazovku s vysokým rozlišením
- řádek 23 zaplní torné pole obrazovky obrázkem lodě v levém horním rohu obrazovky
- řádek 17-21 kreslí hvězdy
- řádek 24 kreslí a vybarví planetu
- řádek 25 kreslí prsteny kolem planety
- řádek 26 kreslí kontur kabiny lodě
- řádek 28 kreslí dolní okno kosmické kabiny
- řádek 35 kreslí horní okno kosmické kabiny
- řádek 38 vybarví kosmickou kabinu bíle
- řádek 40 kreslí kontury zadních raket lodě
- řádek 42 a 43 kreslí zadní raketové motory pro návrat lodě
- řádek 44 vybarví zadní raketové motory a kreslí obvod dna zadních raket v barvě pozadí
- řádek 45 načrtne přímky za zadními raketami v barvě pozadí /až dosud vidíme na obrazovce jediný obrázek bez použití příkazů pro sprajty/
- řádek 47 umístí souřadnice SSHAPE do prvé půlky /24 x 21 pixelů/, do kabiny lodě a uloží data do stringu A$
- řádek 48 umístí souřadnice SSHAPE do druhé půlky /24 x 21 pixelů/ lodě a uloží data do stringu B$

- řádek 50 přenese data A$ do sprajtu 1
- řádek 55 přenese data B$ do sprajtu 2
- řádek 60 aktivuje sprajt 1 a vybarví jej červeně
- řádek 65 aktivuje sprajt 2 a vybarví jej modře
- řádek 82 umístí sprajt 1 na souřadnice 150,150
- řádek 83 umístí sprajt 2 vpravo o 24 pixelů od počátečních souřadnic sprajtu 1
- řádky 82 a 83 spojí fakticky sprajty
- řádek 85 a 87 pohybuje spojenými sprajty po obrazovce
- řádek 90 pozastaví program. Zdržení je nutné, aby oba sprajty překřížili obrazovku dvakrát. Při vynechání tohoto zdržení by sprajty neměly dostatek času přemístit se po obrazovce.
- řádek 92 a 93 umístí sprajty do středu obrazovky a připraví loď k zapnutí zadních raket
- řádek 95 popožene kabinu kupředu /sprajt 1/. Číslo 10 v řádku 95 určuje rychlost sprajtu. Rychlost se mění od 1 /zastavení/ do 15 /maximální rychlost/
- řádek 96 oddělí koncové rakety lodě a ty vystoupí z obrazovky
- řádek 97 způsobí další zdržení, aby koncové rakety /sprajt 2/ stačili opustit obrazovku
- řádek 98 deaktivuje sprajt 2, jelikož opustil obrazovku
- řádek 99 další zádrž, během níž se kabina pohybuje po obrazovke
- řádek 100 se vrací do textového modu.

sledky než použití jediného sprajtu. Nejdůležitějšími body, které je třeba připomenout, jsou 1/ Přesvědčete se, že jste zvolili polohu souřadnic SSHAPE ve správném bodu obrazovky, abyste správně uložili data obrázku; 2/ ujistěte se, že jste správně zvolili souřadnice, když provádíte operaci spojení sprajtů příkazem MOVSPR. V našem příkladu jsme museli umístit sprajt 2 o 24 pixelů vpravo vzhledem k sprajtu 1.

Poté, co jsme vysvětlili, jak spojovat dva sprajty, zkuste spojit sami více než dva sprajty. Více sprajtů se používá, když chceme přidat více podrobností k obrázku.

C 128 ovládá ještě dva další příkazy pro sprajty: SPRCOLOR a COLLISION, které v této kapitole nebudeme vysvětlovat. Informace o těchto příkazech naleznete v Encyklopedii BASICu 7.0 v Kapitole 5.

Ukládání dat sprajtů do binárních fajlů

POZNÁMKA: Výklad, který následuje, předpokládá znalost strojového kódu, rozdělení paměti binárních fajlů a fajlů účelových kódů. Commodore 128 ovládá dva nové příkazy BLOAD a BSAVE, umožňující jednoduše zacházet s daty sprajtů. Písmeno B v BLOAD a BSAVE znamená Binární. Příkazy BLOAD a BSAVE ukládají a sní- [illegible] fajly z disku. Binární fajl obsahuje část programu ve strojovém kodu nebo soubor dat uvnitř stupnice specifikovaných adres. Pravděpodobně znáte příkaz SAVE zabudovaného monitoru strojového kódu. Jestliže se používá tento příkaz SAVE, výsledný fajl na disku bude binárním fajlem. Binární fajly se používají účinněji než fajly kódů, protože se binární fajl čte bez dalších úprav. Naopak fajl kódů se musí načíst pomocí čtecího programu, jak nás učí Rozvinutý Assembler Commodoru 64, proto pro provedení tohoto fajlu musíme použít systémový příkaz /SYS/. Abyste zaznamenali binární fajl, natiskněte jeden z následujících dvou příkazů:

LOAD "název binárního fajlu",8,1

nebo

BLOAD "název binárního fajlu", B0, P počáteční

kde P počáteční je 3584 v případě záznamu dat sprajtů.

Při použití prvého příkazu musíme specifikovat na konci "1", jinak počítač nanese na počátku text BASIC, jako když zpracovává programy BASIC. Číslo "1" přikazuje počítači uložit binární fajl do stejného místa, kde byl uložen původně.

Abycho porozuměli tomu, co se děje se sprajtem,

všimněte si, že v Commodore 128 je jedna část paměti, s decimálními adresami od 3584 /$CE00/ do 4095 /$0FFF/, vyčleněna pro ukládání dat sprajtů. Tato část paměti zabírá 512 bytů.
Jak bylo řečeno, jeden sprajt sestává z 24 pixelů na šířku a 21 pixelu na výšku. Každý pixel zaplní jeden bit paměti. Jestliže bit ve sprajtu je prázdný /rovný 0/, pak odpovídající pixel na obrazovce bude rovněž prázdný a bude vybarven barvou pozadí. Jestliže bit spritu je zaplněn /rovný 1/, odpovídající pixel na obrazovce bude aktivován v barvě popředí. Kombinace "0" a "1" vytváří tak obraz na obrazovce. Jelikož ve spritu 24 x 21 pixelů každý pixel zaujímá jeden bit paměti, sprajt použije 63 bytů paměti. Z obrázku 6-8 porozumíte kolik prostoru v paměti zabírají data jednoho sprajtu.

```
     12345678  12345678  12345678
 1   ........  ........  ........
22   ........  ........  ........
 3   ........  ........  ........
. . . . . . . . . . . . . . . . . . . .
19   ........  ........  ........
20   ........  ........  ........
21   ........  ........  ........
        Řádek = 24 bitů = 3 byty
```

Obr. 6-8. Paměť obsazená daty sprajtu

Jeden sprajt zaujímá 63 bytů dat. Každý blok sprajtu sestává z 63 bytů dat; dodatečný bajt se nepoužije. Jelikož Commodore 128 může používat 8 sprajtů a každý z nich zaujímá sprajtový blok o 64 bytech, počítač potřebuje 512 /8 x 64/ bytů, aby mohl uchovat data všech osmi obrazců.
Oblast obsahující osm bloků sprajtů začíná místem s adresou 3584 /$0E00/ a končí v místě s adresou 4095 /$0FFF/. Na obr. 6-9 jsou uvedeny skupiny hodnot adres paměti, v nichž jsou uloženy vlastní data každého sprajtu.

```
$0FFF  /4095 decimálních/
       ] sprajt č. 8
$0FC0
       ] sprajt č. 7
$0F80
       ] sprajt č. 6
$0F40
       ] sprajt č. 5
$0FC0
       ] sprajt č. 4
$0EC0
       ] sprajt č. 3
$0E80
       ] sprajt č. 2
$CE40
       ] sprajt č. 1
$CE00  /3584 decimálních/
```

Obr. 6-9. Paměťové adresy pro uchovávání sprajtů

BSAVE

Po opuštění modu SPRDEF musíme uložit data binárního fajlu spritu. Tímto způsobem je

možné znovu zapsat souhrn bloků sprajtu do Commodoru 128 snadno a bez problémů. Abychom uložili data sprajtů do binárního fajlu, použijeme následující příkaz:

BSAVE "jméno binárního fajlu",B0,P3584 TO P4096

"B0" určuje, že se ukládají data sprajtů z banky 0. Parametry P3584 TO P4096 znamenají, že se ukládají adresy od 3584 /$0E00/ do 4095 /$0FFF/, které tvoří rozsah, v němž jsou uloženy v paměti všechna data sprajtů.
Není nutné definovat všechny sprajty, jestliže mají být zaznamenány pomocí příkazu BSAVE. Definované sprajty budou skutečně zaznamenány /příkazem BSAVE/ jako bloky správných sprajtů. Sprajty, které nebyly definovány, budou rovněž zaznamenány v binárním fajlu jako příslušné sprajtové bloky, ale počítač na ně nebude brát zřetel. Operace záznamu /BSAVE/ všech 512 bytů bami sprajtů, nehledě na to, že nebylo použito všech sprajtů, je jednodušší než zaznamenávat pouze bloky jednotlivých sprajtů.

BLOAD

Od této chvíle, když chceme použít znovu sprajty zaznamenané v 512 bytech všech sprajtů od počáteční hodnoty adresy 3584 /$0E00/ do konečné 4095 /$0FFF/, napíšeme následující příkaz:

BLOAD "jméno binárního fajlu" [,B0,P3584]

Musíte samozřejmě použít totéž jméno fajlu, které jste použili při záznamu dat původních sprajtů. "B0" značí banku číslo 0 a P3584 určuje počáteční polohu, od níž bude binární fajl sprajtových dat zaznamenán v paměti počítače. Poslední dva parametry nejsou povinné.

x x x x x x x x x x x x x x x x x x x x x x x x

V této části manuálu jsme viděli, jak mohou nové příkazy BASICu 7.0 Commodoru 128 zjednodušit proces tvorby a oživování grafických obrazců, které bývají komplikované. Následující část manuálu popisuje další nové příkazy BASICu 7,0, které zjednodušují vytváření zvuků a hudby.

Č Á S T 7

Ú V O D

Commodore 128 je vybaven jedním z nejúčinnějších zvukových syntetizátorů, zabudovávaných do mikropočítačů. Syntetizátor, nyzývaný Sound Interface Device (SID), je čip určený výlučně ke generování zvuků a hudby. SID může produkovat současně tři různé hlasy (zvuky). Každý hlas zní v jednom ze čtyř typů zvuků, nazývaných tvar vlny. SID je mimoto vybaven programovatelnými parametry Attacco, Decadimento, Sostegno a Rilascio (ADSR). Tyto parametry určují kvalitu zvuku. Mimoto, syntetizátor je opatřen filtrem, který se používá při volbě zvláštních zvuků, vyloučení zvuků nebo nebo změně charakteru zvuku nebo více zvuků.V této části manuálu se naučíme, jak zacházet s těmito parametry, abychom byli schopni vyprodukovat zvuk jakéhokoliv druhu.

Aby se usnadnila volba a používání většiny funkcí SIDu vyvinul Commodore nové a účinné příkazy hudební v jazyce BASIC.

Na Commodore 128 jsou k dispozici následující nové příkazy pro zvuk a hudbu:

SOUND

ENVELOPE

VOL

TEMPO

PLAY

FILTER

Tři zvukové příkazy budou popsány jeden po druhém v této části manuálu, během níž rovněž vytvoříme ukázkový hudební program. Ke konci této části se naučíme, jak psát hudební programy. Mimoto budeme sto rozšířit uvedenou ukázku a psát programy pro složitější hudební kompozice. To nám umožní programovat vlastní hudební partitury, vytvářet speciální efekty a hrát hudbu velkých mistrů jako Bethoven i současných hudebníků, jako například Beatles. Mimoto budeme sto přidávat hudební složku ke grafickým programům a vytvářet tak vlastní video.

Příkaz SOUND

Příkaz SOUND (ZVUK) byl vytvořen pro jednoduché a rychle generování speciálních efektů v programech.V této části manuálu se naučíme, jak ozvučit celá hudební sramžmá za použití zvukových příkazů.

Příkaz SOUND má následující tvar:

SOUND VC,FREQ,DUR [,DIR [,MIN [,GL [,FO [,LA]]]]]

Parametry mají následující význam:

VC - volba hlasu 1., 2. nebo 3.

FREQ- kmitočet zvuku (od 0 do 65535)

DUR - doba trvání zvuku (v šedesátinách vteřiny)

DIR - stanoví směr růstu/úbytku zvuku
0= kmitočet se zvyšuje
1= kmitočet se snižuje
2= vyvolává kalísání kmitočtu

MIN - volí minimální kmitočet (od 0 do 65535), jestliže je požadováno glisando (DIR)

GL - volba veličiny kroku glisanda (od 0 do 32767)

FO - volba tvaru vlny (od 0 do 3)
0 = trojúhelníkový
1 = pilovitý
2 = proměnný impuls
3 = bílý šum

LI - stanoví délku impulsu, délku tvaru vlny proměnného impulsu

Poznamenáme, že parametry DIR, MIN, GL, FO; a LI není povinné uvádět.

První parametr (VC) v příkazu SOUND stanoví, kolikátým hlasem zvuk bude. Druhý parametr (FREQ) určuje kmitočet zvuku, který nabývá hodnot od 0 do 65535. Třetí parametr (DUR) určuje dobu trvání zvuku, která se měří v šedesátinách vteřiny. Abychom získali zvuk trvající jednu vteřinu, musíme požadovat trvání 60, protože 1/60 krát 60 se rovná 1. Zvuku dvouvteřinového dosáhneme při 120, zvuk desetivteřinový při 600 atd. Čtvrtý

parametr (DIR) určuje směr vzrůstání nebo snižování kmitočtu zvuku. Tato funkce se nazývá glidando. Pátý parametr (MIN) určuje minimální kmitočet v bodu, kde začíná glisando. Šestý (GL) je hodnota kroku glisanda, podobná hodnotě kroku ve smyčce FOR-NEXT. Jestliže jsou uvedeny hodnoty DIR, MIN a GL v příkazu SOUND, zvuk bude nejprve znít na původní úrovni specifikované parametrem FREQ.
Dále syntetizátor produkuje glisando zvuku na každé hodnotě celé stupnice kmitočtů, počínaje kmitočtem MIN. Glisando bude mít přírustek nebo zmenšení určené hodnotou kroku, sledujíc směr specifikovaný parametrem DIR, takže kmitočet bude znít na nové úrovni.
Sedmý parametr (FO) příkazu SOUND volí tvar zvokové vlny (o tvaru vlny pojednáme podrobněji v paragrafu nazvaném Pokročilé zvuky a hudba v modu C128).
Poslední parametr příkazu SOUND určuje délku impulsu tvaru vlny, jestliže je zvolen jako parametr tvaru vlny (viz odpovídající část v pokročilých zvucích - podrobnosti o tvaru vlny a délce impulsu).

Formulování programu SOUND

Nyní ukážeme, jak napsat první program SOUND. Uvedeme příklad příkazu SOUND:

```
10 VOL 5
20 SOUND 1,4096,60
```

Proveďte tento program. Commodore 128 vydá krátký a pronikavý zvuk. Dříve než se provede zvukový příkaz, stanoví se síla zvuku. Toho je dosaženo příkazem na řádku 10, který stanoví zvukovému čipu sílu zvuku. Řádek 20. zahraje první hlas s kmitočtem 4096 v trvání 1 vteřina (1/60 krát 60).
Změňte kmitočet:

```
30 SOUND 1,8192,60
```

Všimněte si, že řádek 30. produkuje vyšší tón, než řádek 20. To ukazuje na přímou závislost mezi parametrem kmitočtu a efektivním kmitočtem zvuku. Zvýšení kmitočtu Commodoru 128 zvýší výšku tónu. Napište nyní následující příkaz:

```
40 SOUND 1,0,60
```

Tento příklad ukazuje, že hodnota FREQ=0 vydává basový kmitočet (do té míry basový, že není slyšitelný). Hodnota FREQ=65535 vyvolá nejvyšší možný kmitočet.

Nyní umístíme zvukový příkaz do smyčky FOR-NEXT. Umožní nám to vyprodukovat celou stupnici kmitočtů uvnitř smyčky. Přidejte k programu následující příkazy:

```
50 FOR I=1 TO 65535 STEP 100
60 SOUND 1,I,1
70 NEXT
```

Tato část programu hraje zvuky s tvarem vlyny s proměnným impulsem v celé stupnici kmitočtů od 1 do 65535, s přírůstkem 100, počínajíc basovými kmitočty až po nejvyšší. Jestliže tvar vlny není specifikován, počítač zvolí jako neurčený parametr tvaru vlny číslo 2 - vlna s proměnným impulsem.
Nyní změníme tvar vlny.- napište následující (60) řádek a program znovu proveďte:

```
60 SOUND 1,I,1,0,0,0,0,0
```

Nyní prpgram hraje první hlas, používá trojúhelníkový tvar vlny, mění kmitočet od 1 do 65535 s přírůstkem 100. Vyvolá to podobný zvuk, jaký se používá ve hře vtipná hra.
Zkuste tvar vlny 1 - pilovitý tvar a poslechněte si zvuk vytvořený při provedení následujícího příkazu:

```
60 SOUND 1,I,1,0,0,0,1,0
```

Pilovitý tvar vlny má zvuk jako u trojúhelníkové, ale méně bzučí. Nakonec zkuste tvar vlny bílého šumu (3). Zaměňte řádek 60. následujícím:

```
60 SOUND 1,I,1,0,0,0,3,0
```

Nyní programová smyčka hraje generátor bílého šumu v celé stupnici kmitočtů. Nejprve bude slyšet dunění, zatímco s růstem kmitočtu poroste až nakonec bude reprodukovat hluk při startu rakety.
Až dosud se nestalo, že bychom specifikovali všechny parametry příkazu SOUND. Například v řádku:

```
60 SOUND 1,I,1,0,0,0,3,0
```

tři nuly za 1,I,1 přísluší parametrům glisando v příkazu SOUND. Poněvadž tyto parametry nejsou specifikovány, zvuk nemá glisando.
Připojte k programu následující řádek:

```
100 SOUND 1,49152,240,1,0,100,1,0
```

hlas
kmitočet
trvání
směr glisanda
minimální kmitočet glisanda
veličina kroku glisanda
tvar vlny
délka impulsu tvaru vlny s prom. imp.

Řádek 100 určuje kmitočet glisanda 49152 a snižuje glisando po stovkách až do dosažení minimálního kmitočtu glisanda (0). Hlas 1. používající pilovitý tvar vlny (1) vyvolává zvuk trvající čtyři vteřiny (240*1/60 vteřin). Řádek 100 produkuje zvuk padající bomby, jako je tomu v mnoha videohrách.
Nyní měňte některé parametry v řádku 100. Například určete směr glisanda 2 (kolísání); změňte nejmenší kmitočet glisanda na 32768 a zvyšte krok na 3000. Příkaz má tvar:

```
110 SOUND 1,49152,240,2,32768,3000,1
```

Řádek 110 produkuje zvuk sirény, jeko když je policie již nablízku. Abyste uslyšeli příjemnější zvuk, napište:

```
110 SOUND 1,65535,250,0,32768,3000,2,2600
```

Obdržíte zvuk, jako když kosmická flotila pálí po nepříteli.
Dosud jsme programovali jednohlasé zvuky.,je však možné produkovat zajímavé zvukové efekty pomocí příkazu SOUND za použití až tří hlasů. Napíšeme nyní program používající tři hlasy. Uvedeme nyní program, který ukáže, jak programovat čip syntetizátoru Commodoru 128. Program při provádění vyžaduje zadání některých parametrů a potom generuje příslušný

zvuk. Výpis programu je uveden níže. Natiskněte program a proveďte:

```
10 SCNRL: PRINT "          HRAC ZVUKU": PRINT
   PRINT:PRINT
20 PRINT "ZAVED PARAMETRY ZVUKU":PRINT:PRINT
30 INPUT "HLAS (1-3)";V
40 INPUT "KMITOCET (0-65535)";F
50 INPUT "DOBA TRVANI (0-32767)";D : PRINT
60 INPUT "CHCES ZADAT DALSI PARAMETRY? A/N";
     B$: PRINT
70 IF B$="N" THEN 130
80 INPUT "SMER GLISANDA (0=NAHORU,1=DOLU,2=KO
   LISA)";DIR
90 INPUT "NEJMENSI KMITOCET GLISANDA (0-65535)
        ;M
100 INPUT "KROK GLISANDA (0-32767)";S
110 INPUT "TVAR VLNY (0=TROJ, 1=PIL,2=PROMIMP,
     3=SUM)";W
120 IF W=2 THEN INPUT "SIRKA IMPULSU (0-4095)"
       ;P
130 SOUND V,F,D,DIR,M,S,W,P
140 INPUT "CHCES SLYSET ZVUK JESTE JEDNOU?
       A/N";A$
150 IF A$="A" THEN 130
160 RUN
```

Vysvětlíme program: řádek 10. a 20. ukáže na obrazovce počáteční hlášení. Řádky 30 - 50 zavádějí do počítače parametry hlasu, kmitočtu

a délky zvuku.
Řádek 60. se ptá, zda chcete zavést nepovinné parametry SOUNDu, jako například zavést glisando a tvar vlny. Jestliže tyto parametry nechcete spedifikovat, stiskněte tlačítko "N" (ne). Program přeskočí na řádek 130 a vyprodukuje zvuk. Jestliže naopak nepovinné parametry chcete specifikovat, stiskněte tlačítko "A" (ano). Program pokračuje na řádku 80. Řádky 80 - 110 vyžadují zadání směru glisanda, nejmenšího kmitočtu glisanda, hodnoty kroku glisanda a tvaru vlny. Řádek 120 zavádí délku impulsu tvaru vlny s variabilním impulsem, jestliže byl zvolen tvar vlny čís. 2 (proměnný impuls). Nakonec řádek 130. generuje zvuk v souladu s parametry specifikovanými dříve v programu. Řádek 140. se ptá, zda chcete znovu slyšet tentýž zvuk. Jestliže je odpověď kladná, stiskněte tlačítko "A", jestliže je záporná stiskněte tlačítko "N". Řádek 150 prověřuje, zda jste stiskli "A"; v kladném případě se řízení programu vrací na řádek 130. a program znovu generuje zvuk. Jestliže jste nestiskli "A", program provede řádek 160 a celý program se opakuje. Abyste zastavili program, stiskněte současně tlačítka RUN/STOP a RESTORE.

Náhodné zvuky

Následující program generuje náhodné zvuky pomocí funkce RND (generátor náhodných čísel). Každý parametr SOUNDu se počítá jako náhodné číslo. Zavěďte program do počítače, uložte jej a proveďte. Program předvede tisíce zvuků, které jen mohou být produkovány pro nejrůznější kombinace specifikovaných parametrů SOUNDu.
Výpis programu:

```
10 PRINT "VC  FRQ  DIR  MIN  SV  WF  PW":
       VOL 5
20 PRINT "----------------------------------"
30 V=INT(RND(1)*3)+1          :REM HLAS
40 F=INT(RND(1)*65535)        :REM KMITOCET
50 D=INT(RND(1)*240)          :REM DELKA
60 DIR=INT(RND(1)*3)          :REM SMER KROKU
70 M=INT(RND(1)*65535)        :REM MIN KMITOCET
80 S=INT(RND(1)*32767)        :REM HODNOTA KROKU
90 W=INT(RND(1)*4)            :REM TVAR VLNY
100 P=INT(RND(1)*4095)        :REM SIRKA IMPULSU
110 PRINT V;F;DIR;M;S;W;P:PRINT:PRINT
       :REM zobrazi hodnoty parametru
120 SOUND V,F,D,DIR,M,S,W,P :REM HRAJE
130 SLEEP 4                 :REM CHVILI CEKA
140 SOUND V,0,0,DIR,0,0,W,P :REM VYPNE ZVUK
150 GOTO 10
```

Řádky 10. a 20. tisknou záhlaví sloupců parametrů a podtržení. Řádky 30.-100. počítají každý zvukový parametr uvnitř určeného rozpětí hodnot. Například řádek 30. počítá pořadové číslo hlasu:

```
30 V=INT(RND(1)*3)+1
```

Označení RND(1) specifikuje generační hodnotu pro náhodné číslo. Generační číslo je číslo startující počítač. Hodnota 1 přikazuje počítači přikazuje počítači generovat nové generační číslo pokaždé, když narazí na příkaz. V Commodoru 128 máme k dispozici 3 hlasy, značení *3 přikazuje počítači generovat náhodné číslo v rozmezí od 0 do 3. Všimněte si, že jelikož neexistuje hlas 0, hodnota +1 v řádku 30. přikazuje počítači generovat náhodné číslo $1 \leq N < 4$. Procedura generování náhpdného čísla ve specifikovaném rozmezí; hodnot znamená násobit určité náhodné číslo maximální hodnotou parametru (v našem případě 3). Jestliže je minimální hodnota parametru větší než nula, připočteme k náhodnému číslu číslo určující minimální hodnotu rozmezí generovaných čísel (v daném případě 1). Například řádek 40 generuje náhodná čísla v rozmezí $0 \leq N < 65535$. V tomto případě je minimální hodnota nula, takže není nutné přidávat nic k generovanému náhodnému číslu.

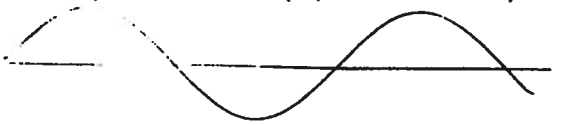
Řádek 110 tiskne hodnoty parametrů. Řádek 120 zahraje zvuk, určený náhodnými čísly z řádků 30-100. Řádek 130 zastaví program na 4 vteřiny. Nakonec řádek 150 předá řízení na řádek 10 a proces se opakuje, dokud nestisknete současně tlačítka RUN/STOP a RESTORE.

Až dosud jsme prováděli programy používající pouze příkaz SOUND. Příkaz SOUND se může použít pro přehrávání hudebních děl podle partitury, avšak jeho hlavní funkcí je generovat rychle a účinně zvukové efekty. Commodore 128 je vybaven speciálními příkazy pro provádění hudebních děl. Následující paragrafy popisují účinnější příkazy pro přehrávání hudby, umožňující hrát hudební partitury a složitá aranžmá pomocí syntetizátoru Commodoru 128.

Pokročilé zvuky a hudba v modu C 128

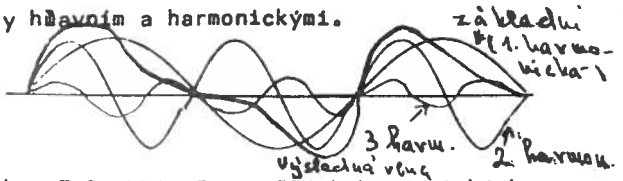
Stručné vysvětlení charakteristik zvuků

Každý zvuk je ve skutečnosti zvuková vlna šířící se ve vzduchu. Jako jakákoliv vlna, i zvukovou vlnu (sinusoidu) lze představit graficky a matematicky (viz. Obr. 7-1).



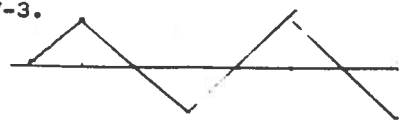
Obr. 7-1. Sinusoidální vlna

Zvuková vlna se pohybuje, mění (osciluje) určitou rychlostí (kmitočet), která určuje výšku zvuku (altový nebo basový).
Zvuk mimoto obsahuje vyšší harmonické, kterými jsou vlny - násobky kmitočtu nebo noty. Kombinace těchto harminických zvukových vln určuje kvalitu zvuku, nazývanou zabarvení.
Na obr. 7-2. je zobrazen vztah mezi kmitočty hlavním a harmonickými.

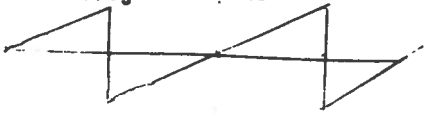


Obr. 7-2. Kmitočty vyšších harmonických

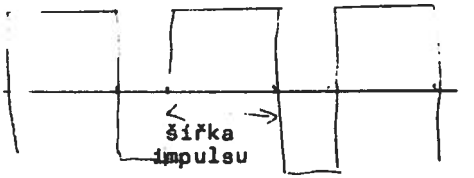
Zabarvení hudebního tónu (tj. zvučení tónu) se určuje tvarem vlny noty. Commodore 128 generuje čtyři typy tvarů vlny.: trojúhelníkový, pilovitý, variabilní impuls a šum. Abyste získali grafickou představu o těchto čtyřech tvarech vln, podívejte se na obr. 7-3.



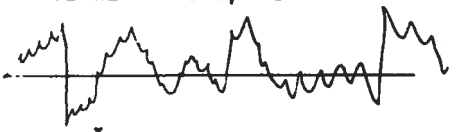
Trojúhelníková



Pilovitá



Variabilní impuls



Šum

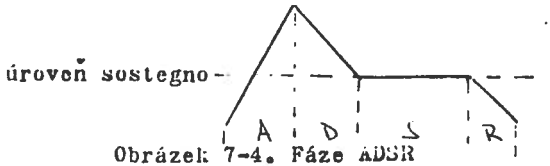
Obr. 7-3. Typy tvarů zvukových vln

Hudba na Commodoru 128

Příkaz ENVELOPE

Síla určitého zvuku se mění během trvání noty, od okamžiku, kdy začne být vnímán, do okamžiku, kdy přestane být slyšitelný.
Tato vlastnost síly zvuku se nazývá attacco (úhoz), decadimento (úpadek), sostegno (opora) a rilascio (popuštění) (ADSR). Attacco je okamžik, v němž hudební nota dosahuje největší síly. Decadimento je doba v níž hudební nota ubývá od maximální hodnoty až do hodnoty sostegno. Sostegno je hladina, na níž nota zní se střední silou. Rilascio

je perioda během níž hudební nota klesá od úrovně sostegno až do nulové síly zvuku. Generátor ENVELOPE řídí parametry ADSR zvuku. Grafické znázornění ADSR je na obr. 7-4.



Obrázek 7-4. Fáze ADSR

Commodore 128 může měnit všechny parametry ADSR, maximálně 16 různých hodnot. Umožňuje to dosáhnout velké flexibility generátoru ENVELOPE a vlastností síly generovaného zvuku.

Jedním z nejúčinnějších příkazů Commodoru 128 ovládajícím tvar ADSR a tvar zvukové vlny je příkaz ENVELOPE. Tento příkaz dává různé instrukce čipu syntetizátoru, který poskytne jakýkoliv unikální zvuk. ENVELOPE umožňuje ovládat syntetizátor SID. Pomocí tohoto příkazu je možné volit tvar akustické vlny hudby a zvukových efektů. Příkaz ENVELOPE má následující tvar:

ENVELOPE e[,a[,d[,s[,r[,[,fo[,li]]]]]]]

Význam jednotlivých písmen je následující:

e - číslo obálky, zabarvení (od 0 do 9)
a - hodnota attacco (od 0 do 15)
d- hodnota decadimento (od 0 do 15)
s - hodnota sostegno (od 0 do 15)
r - hodnota rilascio (od 0 do 15)
fo- tvar vlny - 0=trojúhelníkový
1=pilovitý
2=impulsní (hranatý)
3=šum
4=kruhová modulace
li- délka impulsu (od 0 do 4095)

Vysvětlíme význam dosud nedefinovaných parametrů:

"Obálka" - (zabarvení) vlastnost hudebního tónu, specifikovaná tvarem vlny a hodnotou attacco, decadimento, sostegno a rilascio tónu. Například zabarvení tónu kytary má ADSR a tvar vlny odlišný od flétny.

Tvar vlny: Akustickou vlnu vytváří kombinace vyšších harmonických frekvencí, doprovázejících základní tón. Akustická vlna sestává z harmoniky a vyšších harmonických, naložených na základní kmitočet tónu a založených na něm. Kvalita tónů generovaných různým tvarem vlny se vzájemně liší; různé tvary vlny byly ukázány na obr. 7-3.

Délka impulsu - interval mezi notami, generovanými impulsním tvarem vlny

Příkaz ENVELOPE je velice účinný: určuje z větší části kvalitu tónu, generovaného zvukovým syntetizátorem Commodoru 128, který ovládá 10 "zabarvení", predefinovaných pro 10 různých hudebních nástrojů. Při používání těchto "zabarvení" není nutné specifikovat parametry ADSR, tvar vlny a délku impulsu; jediné, co se musí specifikovat, je číslo "zabarvení". Všechny ostatní parametry specifikije automaticky Commodore 128 sám. Předem zvolitelná "zabarvení" pro různé typy hudebních nástrojů jsou následující:

číslo	Nástroj	A	D	S	R	tvar vlny	délka impulsu
0	Klavír	0	9	0	0	2	1536
1	Harmonika	12	0	12	0	1	
2	Calliope	0	0	25	0	0	
3	Buben	0	5	5	0	3	
4	Flétna	9	4	4	0	0	
5	Kytara	0	9	2	1	1	
6	Klavičembalo	0	9	0	0	2	512
7	Varhany	0	9	9	0	2	2048
8	Trubka	8	9	4	1	2	512
9	Xylofon	0	9	0	0	0	

Obr. 7-5. Nepovinné parametry příkazu ENVELOPE

Poté, co jsme vysvětlili, jak pracuje příkaz ENVELOPE, začneme psát další program, počínajíc příkazem:

```
10 ENVELOPE 0,5,9,2,2,2,1700
```

Tento příkaz ENVELOPE redefinuje "zabarvení" (0) pro klavír následujícím způsobem: atttacco = 5, decadimento=9, sostegno=2, rilascio=2, tvar vlny zůstává týž (2) a délka impulsu tvaru vlny s variabilním impulsem trvá 1700. "Zabarvení" pro klavír nepocítí tyto změny, po kud se nevyvolá příkaz PLAY, který uvedeme poz ději v této části manuálu.

Další fází hudebního programování je určení síly zvuku (volume) zvukpvého čipu následujícím přílazem:

```
20 VOL 8
```

Příkaz VOL určuje zvukovému čipu hodnotu od 0 do 15, kde 15 je maximální hodnota a 0 je vypnutá síla zvuku.

Příkaz TEMPO

Následující fází hudebního programu na Commodoru 128 je řízení tempa, rychlosti provádění hudebního díla. Pro tento účel se používá příkaz TEMPO v následujícím tvaru:

```
TEMPO n
```

kde n je číslo od 0 do 255 (255 je maximální rychlost). Jestliže příkaz TEMPO není v programu uveden, Commodore 128 dosadí automaticky hodnotu 8. Přidejte k ukázce hudebního programu následující příkaz:

30 TEMPO 10

Příkaz PLAY

Podíváme se jak zní jeden tón písně. Noty se ozvučují podobně, jako když je řetězec textu zobrazován na obrazovce pomocí příkazu PRINT. Jediný rozdíl je, že místo PRINT se používá příkaz PLAY (hraj). Print zobrazuje text, zatímco PLAY generuje hudební tóny. Příkaz PLAY má následující tvar:

PLAY "řetězec znaků řídících syntetizátor a hudebních not"

Příkaz PLAY může obsahovat maximálně 255 znaků (zahrnujících hudební noty a řídící znaky syntetizátoru). Poněvadž tato hodnota je větší, než maximální počet znaků (160) dovolených v jednom řádku programu BASIC 7.0, musíme sdružit alespoň dva řetězce, abychom dosáhli této hodnoty, nebo, pokud se chceme vyhnout sdružování řetězců, nesmíme překročit 160 znaků, tedy jeden řádek programu (ten je ekvivalentní čtyřem řádkům na obrazovce se 40 sloupci nebo dvěma řádkům v modu 80 sloupců). Tímto způsobem se produkují stringy pro příkaz PLAY, jednoduché pro chápání i použití. Abychom vyznačili hudební noty, napište v uvozovkách písmena odpovídající notám, které chcete zahrát (tj. C, D, E, F, G, A, B (=H)). Například hudební stupnice:

40 PLAY "CDEFGAB"

Tímto způsobem přehrajeme noty CDEFGAB se "zabarvením" klavíru, tedy číslo 0. Po každém provedení vytvořeného ukázkového hudebního programu podržte tlačítko RUN/STOP a stiskněte tlačítko RESTORE, abyste čip syntetizátoru uvedli do počátečního stavu.
Nyní můžeme specifikovat délku noty tím, že ji předešleme jedno z následujících písmen ve výrazu v uvozovkách:

W - celá nota
H - poloviční nota
Q - čtvrtinová nota
I - osminová nota
S - šestnáctinová nota

Jestliže délka noty není specifikována, předpokládá se W (celá nota).
Lze také zahrnout mezi noty pauzu tak, že v řetězci PLAY napíšeme:

R - pauza

Abychom sdělili počítači, že má čekat, dokud všechny hrané hlasy nedojdou na konec taktu, zahrňte do uvozovek:

M - čekej na konec taktu

Commodore 128 používá také další řídící znaky syntetizátoru, které lze zahrnout do řetězce PLAY v uvozovkách. Umožňují získat plnou kontrolu nad každým tónem a měnit ovládání syntetizátoru uvnitř řetězce not. Za každým řídícím znakem uveďte číslo z rozmezí, dovoleného pro daný znak. Řídící znaky a rozmezí číselných hodnot pro každý znak jsou uvedeny v tabulce 7-6. Písmeno n za řídícím znakem znamená, že je zde třeba uvést požadované číslo.

Řídící znak	Popis	Rozmezí	Pokud není n uvedeno
Vn	Hlas	1 - 3	1
On	Oktáva	0-6	4
Tn	Zabarvení	0 - 9	0
Un	Hlasitost	0 - 15	9
Xn	Filtr	0=vypnut 1= zapnut	0

Tab. 7-6. Řídící znaky zvukového syntetizátoru

Ačkoliv čip SID může zpracpvat tyto řídící znaky v libovolném pořadí, doporučujeme zavádět řídící znaky do řetězce v tom pořadí, jak jsou uvedeny v tab. 7-6.
Specifikovat řídící znaky není nutné, avšak doporučujeme dělat to, abyse využily možnosti syntetizátoru. Commodore 128 sám automaticky přiřadí řídícím znakům syntetizátoru hodnoty uvedené v posledním sloupci tab. 7-6. Jestliže speciální řídící znaky nejsou uvedeny, čip SID bude hrát s jediným zabarvením, jeden hlas a v určité oktávě bez použití filtru. Specifikujte tyto znaky, abyste plně kontrolovali tóny uvnitř řetězce PLAY.
Jestliže specifikujete příkaz ENVELOPE a jestliže požadované parametry zvolíte, místo abyste použili jejich implicitní hodnoty uvedené v tabulce 7-5., pak hodnota řídícího znaku "zabarvení" v řetězci PLAY musí být stejná jako číslo "zabarvení" uvedené v příkazu ENVELOPE, aby počítač uvedené parametry ENVE/-LOPE akceptoval. Jestliže chcete použít implicitní parametry "zabarvení" uvedené v tabulce 7-5, pak příkaz ENVELOPE nesmíte použít. V takovém případu zvolte jednoduše číslo zabarvení pomocí řídícího znaku (T) v příkazu PLAY.

Nyní uvedeme příkaz PLAY s použitím řídících znaků čipu SID uvnitř řetězce. Přidejte k programu následující řádek a všimněte si rozdílu mezi tímto příkazem a příkazem PLAY na řádku 40.

```
50 PLAY "MV2 O5 T7 U5 X0 CDEFGAB"
```

Tento příkaz zahraje stejné tóny jako řádek 40. s tím rozdílem, že byl zvolen druhý hlas, tóny budou hrány o oktávu výše (5) než tóny na řádku 40., hlasitost bude 5 a filtr je vypnut. Dosud jsme ponechávali filtr vypnutý. Až v následující části manuálu vysvetlíme jak používat filtr, můžeme se vrátit k tomuto řádku programu a aktivovat filtr, abychom viděli, jak ovlivňuje tóny. Všimněte si, že v řádku 50. jsme zvolili nový nástroj - varhany - řídícím znakem T7. Nyní program používá dva nástroje ve dvou nezávislých hlasech. Přidejte následující příkaz s třetím hlasem:

```
60 PLAY "MV3 O6 U7 T6 X0 CDEFGAB"
```

Podíváme se nyní jak řádek 60. ovládá syntetizátor. V3 vybere třetí hlas, O6 zvedne o jednu oktávu (6) třetí hlas vzhledem k druhému hlasu, T6 vybere klavičembalo, V7 určí hlasitost 7 a X0 deaktivuje filtr pro všechny tři hlasy. Nyní program hraje trojhlasně, každý hlas o oktávu výš než předchozí, s třemi různými nástroji: klavír, varhany a klavičembalo.

Až dosud jsme používali příkaz PLAY aby hrál celé tóny. Přidáme noty různé délky: přidejte do řetězce PLAY řídící znaky délky tónů:

```
70 PLAY "MV2 O6 T0 U7 X0 H C D Q E F IGA SB"
```

Řádek 70. hraje druhý hlas v oktávě 6 s hlasitostí 7 na klavíru (0) a s vypnutým filtrem. Tento příkaz zahraje tóny C a D jako poloviční noty, E a F jako čtvrtinové noty, G a A jako osminové a B jako šestnáctinovou. Všimněte si rozdílu mezi "zabarvením" klavíru v řádku 40. a předefinovaným zabarvením klavíru na řádku 70. Řádek 40. dává tón podobnější tónu klavíru než řádek 70.

Je možná hrát i zvýšené (křížkované), snížené a prodloužené noty tak, že předešleme zvoleným notám následující znaky:

- # - zvýšení (křížek)
- $ - snižovací znaménko
- . - prodloužená nota

Prodloužená nota je jedenapůlkrát delší než neprodloužená.

Nyní přidejte do programu křížky, snižovací znaménka a prodloužené noty:

```
80 PLAY "MV1 O4 T4 U8 XO .HCD Q*EF I$GA.S*B"
```

Řádek 80. hraje první hlas ve 4. oktávě hlasitostí 8, nástroj flétna a s vypnutým filtrem. Přitom zahraje C a D jako prodloužené poloviční noty, E a F jako zvýšené čtvrtinové noty, G a A jako snížené osminové a B jako prodlouženou zvýšenou šestnáctinovou notu. V libovolném místě řetězce PLAY je možné přidat pauzu (R).

Až dosud byl ve všech příkladech filtr zvukového syntetizátoru vypnut a tudíž nebyla příležitost vyzkoušet jeho možnosti. NynPoté, co jsme objasnili většinu zvukových a hudebních příkazů a řídících znaků SIDu, přejdeme k následující části manuálu, kde vysvětlíme, jak zdokonalit kvalitu hrané hudby pomocí příkazu FILTER.

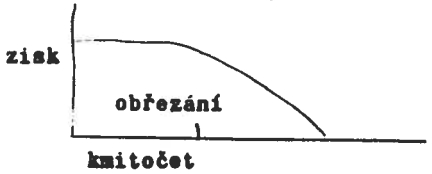
Filtr SIDu

Pote, co jsme zvolili zabarvení, ADSR, hlasitost a tempo, použijeme filtr, abychom zdokonalili syntetizované zvuky. V programu příkaz FILTR musí předcházet příkazu PLAY. Dříve než použijeme filtr se ujistíme, že dobře chápeme, jak generovat zvuky. Čip SID obsluhuje jediný filtr, který se používá pro všechny tři hlasy. Produkovaná hudba by zněla i bez filtru, avšak pro využití všech potenciálních možností syntetizátoru doporučujeme využít i příkaz FILTR, který zvyšuje kvalitu a zřetelnost zvuků.

V prvém paragrafu této části manuálu, týkajícím se charakteristik zvuku, byl zvuk definován jako akustická vlna, která se šíří (osciluje) vzduchem určitou rychlostí, určující kmitočet vlny. Akustickou vlnu tvoří základní kmitočet a vyšší harmonické, jež jsou násobkem základního kmitočtu. Viz obr. 7-2. Vyšší harmonické dávají zvuku zabarvení, tj. kvalita zvuku je určována tvarem vlny. Filtr v čipu SID umožňuje zdůraznit nebo vyloučit vyšší harmonické z určitého tvaru vlny a změnit její zabarvení.

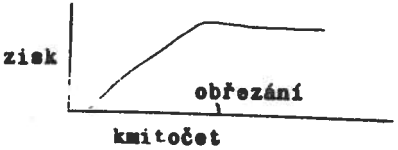
Filtr čipu SID filtruje zvuky třemi způsoby: ze strany basů, pásmově, ze strany výšek (aktů). Tyto typy filtrů jsou aditivní, tj. mohou se používat současně. Podrobnosti naleznete v následující části manuálu. Filtr basový uřezává kmitočty nad určitou hodnotou (obřezávací kmitočet). Obřezávací kmitočet je hranice, která odděluje kmitočty, které budou znít od kmitočtů, které nebudou znít. S filtrem basového typu čip SID přehrává všechny kmitočty pod obřezávacím kmitočtem a vyloučí kmitočty nad ním. Dá se říci, že basové kmitočty procházejí filtrem, zatímco vysoké se uřezávají. Basový filtr produ-

kuje zvuky syté a stejnorodé. Viz obr. 7-7.



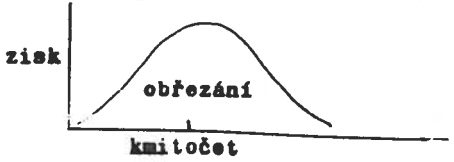
Obr. 7-7. Basový filtr

Naopak, altový filtr všechny kmitočty nad obřezávacím propouští čipem. Všechny kmitočty pod ním se uřezávají. Viz obr. 7-8. Altový filtr produkuje zvuky kovově zabarvené a temné.



Obr. 7-8. Altový filtr

Oboustranný filtr propouští čipem SID kmitočty v určitém rozmezí pod a nad obřezávacím kmitočtem. Všechny kmitočty pod a nad pásmem kmitočtů kolem obřezávacího kmitočtu budou potlačeny. Viz obr. 7-9.



Obr. 7-9. Pásmový filtr

Příkaz FILTER

Příkaz FILTER specifikuje obřezávací kmitočet, typ použitého filtru a rezonanci. Rezonance je efekt soustředění kmitočtů zvukové vlny, když se přibližujeme k obřezávacímu kmitočtu. Rezonance určuje přesnost a zřetelnost zvuku; čím je rezonance vyšší, tím je zvuk zřetelnější. Příkaz FILTER má následující tvar:

```
FILTER cf,lp,bp,hp,res
```

Parametry mají následující význam:

cf - obřezávací kmitočet (od 0 do 2047)
lp - basový filtr, 0=vypnut, 1= zapnut
bp= - pásmový filtr, 0=vypnut, 1=zapnut
hp - altový filtr, 0=vypnut, 1=zapnut
res- rezonance (od 0 do 15)

Obřezávací kmitočet se musí nacházet v rozmezí od 0 do 2047. Avychom aktivovali basový filtr, specifikujte 1 jako druhý parametr a abychom aktivovali altový filtr, apecifikujte 1 jako čtvrtý parametr. Abychom vypnuli filtr, specifikujte 0 v poloze příslušející vypínanému filtru. Je možné zapnout nebo vypnout jeden, dva nebo všechny tři filtry spučasně.

Poté co jsme vysvětlili, jak používat příkaz FILTER, přidejte k zvukovému programu násle-

dující příkaz, ale program zatím neprovádějte

```
45 FILTER 1200,1,0,0,10
```

Řádek 45, určuje obřezávací kmitočet 1200, aktivuje basový filtr, ponechává vypnutými pásmový a altový filtr a přřazuje rezonanci hodnotu 10. Dále aktivujte filtr v příkazu PLAY tím, že řídící znak X0 nahradíte znakem X1. Uveďte zvukový čip do výchozí polohy stisknutím tlačítek RUN/STOP a RESTORE a proveďte zvukový program znovu. Všimněte si rozdílu mezi zvukem tónů nyní a když byly pro dukovány bez použití filtru. Změnte řádek 45 následovně:

```
45 FILTER 1200,0,1,0,10
```

nový řádek vypne bysový filtr a aktivuje pásmový filtr. Stiskněte tlačítka RUN/STOP a RESTORE a proveďte znovu zvukový program. Všimněte si rozdílu mezi basovým a pásmovým filtrem. Změnte ještě jednou řádek 45:

```
45 FILTER 1200,0,0,1,10
```

Uveďte čip do počátečního stavu a proveďte znovu ukázku programu. Povšimněte si rozdílu mezi altovým filtrem a filtry basovým a pásmovým. Zkuste použít jiný obřezávací kmitočet, úroveň rezonance a různé filtry,

abyste vylepšily hudbu a zvuky ve vašich vlastních programech.

Spojování příkazů v hudebním programu

Váš první hudební program je hotov. Nyní můžete programovat vaše vlastní oblíbené motivy. Všechny složky programu jsou nyní spojeny. Výpis programu je uveden níže. Zbývá pouze připojit k programu příkazy pro tisk, které ukazují, co se v programu právě provádí.

```
10 ENVELOPE 0,5,9,2,2,2,1700
15 VOL 8
20 TEMPO 10
25 PRINT "RADEK 30"
30 PLAY "CDEFGAB"
35 FILTER 1200,0,0,1,10
40 PRINT "FILTR VYPNUT"
45 PLAY "V2 O5 T7 U5 X0 CDEFGAB"
50 PRINT "TOTEZ - FILTR ZAPNUT"
55 PLAY "V2 O5 T7 U5 X1 CDEFGAB"
60 PRINT "FILTR VYPNUT - RADEK 65"
65 PLAY "V3 O6 U7 T6 X0 CDEGGAB"
70 PRINT "TOTEZ - FILTR ZAPNUT"
75 PLAY "V3 O6 U7 T6 X1 CDEFGAB"
80 PRINT "RADEK 85 - FILTR VYPNUT"
85 PLAY "V2 O6 T0 U7 X0 HCD QEF IGA SB"
90 PRINT "TOTEZ - FILTR ZAPNUT"
```

```
95 PLAY "V2 O6 T0 U7 X1 HCD QEF IGA SB"
100 PRINT "RADEK 105 - FILTR VYPNUT"
105 PLAY "V1 O4 T4 U8 X0 H.CD Q$EFI$GA S.B"
110 PRINT "TOTEZ - FILTR ZAPNUT"
115 PLAY "V1 O4 T4 U8 X1 H.CD Q$EF I$GA S.B"
```

Na 10. řádku příkaz ENVELOPE specifikuje klavír (0), který stanoví attacco 0, decadimento 9, sostegno 0 a rilascio 0. Mimoto tento příkaz volí tvar vlny - proměnlivý impuls s šířkou impulsu 1700. Řádek 15. určuje hlasitost 8. Řádek 20. určuje tempo 10. Řádek 35. filtruje zvuky hrané na řádcích od 30 do 115 a určuje obřezávací kmitočet filtru 1200. Mimoto řádek 35 vypíná basový a pásmový filtr dvěma 0 za obřezávacím kmitočtem (1200). Altový filtr je zapnut zavedením 1 za dvěma 0. Rezonanci je přiřazena hodnota 10 - poslední parametr příkazu FILTER.
Řádek 30. hraje noty CDEFGAB v daném pořadí. Řádek 45. hraje stejné noty jako řádek 30. se specifikovanými hodnotami řídících znaků SID . hlasitost 5, V1 - první hlas, O5 - pátá oktáva. Připomeneme, že řídící znaky SID umožňují měnit řídící znaky syntetizátoru i uvnitř řetězce, čímž se dosahuje maximální kontrola syntetizátoru. Řádek 65. specifikuje řídící znaky U7 pro hlasitost 7,

V3 pro třetí hlas, O6 pro šestou oktávu, a X0 pro vypnutí filtru. Řádek 65. hraje stejné noty jako řádky 30. a 45., ale s odlišnou hlasitostí, jiným hlasem a oktávou.
Řádek 85. má stejnou hlasitost, hlas a oktávu jako řádek 65, ale zavádí půlnoty pro C a D, čtvrtinové noty pro E a F, osminové pro G a A a šestnáctinovou pro B. Řádek 105. stanoví hlasitost 7, 1. hlas, 4. oktávu a vypnutý filtr. Mimoto řádek 105. stanoví, že C je prodloužená půlnota, E je zvýšená čtvrtinová nota, G a A jsou snížené osminové noty a B je zvýšená psodloužená šestnáctinová nota.

Filtr v pokročilejším programování

V předchozích příkladech se používal vždy pouze jeden filtr. Je však možné kombinovat tři filtry čipu SID, abychom dostali různé efekty, například je možné zapnout současně basový a altový filtr tak, že dostaneme blokový filtr rychlosti čela. Tento typ filtru propouští kmitočty pod a nad obřezávacím kmitočtem čipem SID, zatímco kmitočty blízké obřezávacímu kmitočtu budou potlačeny. Na obrázku 7-10. máme graficky znázorněnu činnost blokovacího filtru.

Obr. 7-10. Blokovací filtr rychlosti čela

Pro obdržení speciálních efektů je možné přidat k pásmovému filtru jak basový, tak altový filtr. Kombinace pásmového a basového filtru vybírá frekvenční pásmo pod obřezávacím kmitočtem. Jiné kmitočty jsou uřezány.
Kombinace pásmového a altového filtru vybírá frekvenční pásmo nad obřezacím kmitočtem. Všechny kmitočty pod ním budou potlačeny.
Zkuste různé kombinace filtrů, abyste položili různý důraz na hudební noty a jiné zvukové efekty. Zkoumejte filtry, abyste zdokonalili zvuky produkované jinými částmi zvukového čipu SID. Poté, co jste vyprodukovali hudební tóny nebo zvukové efekty pomocí čipu SID, vraťte se k předchozím příkladům a zapněte filtr, abyste obdrželi plastičtější a zřetelnější hudbu.
Podali jsme vám všechny informace použitelné při psaní vlastních hudebních programů v jazyce BASIC Commodoru 128. Používejte různé tvary vlny, parametry ADRS, příkazy TEMPO

a FILTER. Obraťte se o pomoc ke knihám, obsahujícím hudební partitury a tiskněte noty hudební stupnice jako posloupnost znaků řetězce. Spojte noty v řetězci používaném v hudebním syntetizátoru Commodoru 128 s grafikou v modu C 128 a vytvářejte vlastní audovizuální programy.

Zakódování hudebního díla podle partitury

V této části manuálu vám předložíme ukázku hudební partitury a vysvětlíme, jak zakódovat noty hudební osnovy a jak je převést do tvaru, pochopitelného pro Commodore 128.
Toto cvičení nejrychleji a nejsnadněji prospěje těm, kdo snadno harají hudbu. Není však nutné být hudebníkem, abyste mohli hrát na Commodore 128.
Pro ty, kdo sami nehrají, obrázek 7-11. ukazuje hudební linky a nad notami odpovídající klávesy klavíru.

střední C

Obr. 7-11. Hudební linky

Na obr. 7-12. je uveden výňatek z díla Invence č. 13 Johanna Sebastiana Bacha. Ačkoliv tato hudba byla napsána před stovkami let, můžeme ji zahrát na takovém moderním syntetizátoru, jako je čip SID Commodoru 128. Uvedeme jednotlivé takty ouvertury Invence č. 13

Obr. 7-12. Výňatek z Invence č. 13 J.S.Bacha

Nejlepší způsob, jak začít kódovat hudební dílo pro Commodore 128, je přepsat noty do přechodného kódu. Přepište noty první hudební linky na kus papíru. Poté přepište noty spodní linky. Před značky not napište jejich délku. Například před osminovou notu napište 8, před šestnáctinovou 16 atd. Nyní oddělte noty tak, že noty jednoho taktu horní linky byly úměrné, pokud jde o tempo, no-

tám jednoho taktu spodní linky.
Jestliže má hudební kompozice třetí linku, oddělte její noty tak, že její trvání bude úměrné dvěma předchozím pentagramům. Jakmile budou noty všech pentagramů rozmístěny podle stejné délky trvání, připíšeme každé lince jeden zvláštní hlas. Například první hlas hraje první linku, druhý hlas druhou, třetí hlas - třetí (pokud je).
Předpokládejme, že první pentagram začíná řetězcem čtyř osminových not a druhá linka skupinou osmi šestnáctinových not. Poněvadž osminová nota má délku dvou šestnáctinových, rozdělíme noty jako na obr. 7-13.

V1 =	8A	8B	8C	8D
V2 =	16D16E	16F16G	16A16B	16C16D

Obr. 7-13. Synchronizace not pro dva hlasy

Poněvadž v hudební kompozici je synchronizace a tempo velice důležitá, ujistěte se, že například noty horní linky pro první hlas jsou v souladu (z hlediska tempa) s notami následující linky pro druhý hlas.Prvá nota prvé linky reprodukované na obr. 7-12. je osmanové A. Prvé dvě noty druhého hlasu jsou šestnáctinové D a E. V tomto případě musíme za prvé zavést osminové noty pro prvý hlas v řetězci PLAY a poté zavést šestnácti-

nové pro druhý hlas. Pokračujeme v příkladu: druhá nota na obr. 7-12. pro prvý hlas je osminové B. Tato osminová nota se rovná (z hlediska tempa) dvěma šestnáctinovým (F a G), které se nacházejí na následující lince pro druhý hlas. Aby bylo koordinované tempo, zaveďte osminové B do řetězce pro prvý hlas a dvě šestnáctinové (F a G) pro druhý hlas.

Začínejte vždy notou, která má nejdelší trvání. Například, jestliže nějaká linka začíná dvěma šestnáctinovými notami na spodní lince pro druhý hlas a horní lonka začíná osminovou notou pro prvý hlas, zaveďte do řetězce nejprve osminovou notu, která musí proznít celá během taktu dvou šestnáctinových not. Musíte poskytnout počítači dostatek času, aby mohl zahrát nejdelší notu a poté notu kratší, jinak kompozice nebude synchronní. Uvedeme program, který zahraje Invenci č. 13. Natiskněte a uložte pro další použití následující program:

```
10 REM INVENCE 13 J S BACHA
20 TEMPO 6
30 A$="V104T7U8X0 V204T0U8X0" :REM V1=VARHANY
     V2=KLAVIR
40 DO
50 PLAY A$
```



```
60 READ A$
70 LOOP UNTIL A$="KONEC HUDBY"
80 END
90 REM   PRVNI TAKT
100 DATA V201IA V103IE V202QA V103SA04C03BEM
110 DATA V202I♯G V103SB04D04IC V202SAEM
120 DATA V104IE V202SA03C V103I♯G V202SBEM
130 DATA V104IE V202SB03DM
140 REM  DRUHY TAKT
150 DATA V203IC V103SAE V202IA V103SA04CM
160 DATA V202I♯G V103SBE V202IE V103SB04DM
170 DATA V104IC V202SAE V103IA V202SA03CM
180 DATA V104QR V202SBEB03DM
190 REM   TRETI TAKT
200 DATA V203IC V104SRE V202IA V104SCEM
210 DATA V203IC V103SA04C V202IA V102SEGM
220 DATA V103IF V203SD02A V103IA V202SFAM
230 DATA V104ID V202SDF V104IF V201SA02CM
240 REM  CTVRTY TAKT
250 DATA V201IB V104SFD V202ID V103SB04DM
260 DATA V202IG V103SGB V202IB V103SDFM
270 DATA V103IE V202SGE V103IG V202SEGM
280 DATA V104IC V202SCE V104IE V201SGBM
290 REM   PATY TAKT
300 DATA V201IA V104SEC V202IC V103SA04CM
310 DATA V103IF V202SDF V104ID V201SB02DM
320 DATA V201IG V103SDB V201IB V103SGBM
```

```
340 REM   SESTY TAKT
350 DATA V201IF V104SC03A V201ID V103SFAM
360 DATA V103ID V201SG02G V103IB V202SFGM
370 DATA V201IA V104SC03A V202I#F V104SCEM
380 DATA V201IB V104SD03B V202I#G V104SDFM
390 REM  SEDMY TAKT
400 DATA V202IC V104SEC V202IA V104SEGM
410 DATA V202ID V104SFE V202ISB V104SDCM
420 DATA V202I#G V103SB04C V202IF V104SDEM
430 DATA V202ID V104SFD V201IB V104S#GDM
440 REM  OSMY TAKT
450 DATA V202I#G V104SBD V202IA V104SCAM
460 DATA V202ID V104SFD V202IE V103SB04DM
470 DATA V202IF V103S#GB V202I#D V104SC03AM
480 DATA V202IE V103SEA V202IE V103SB#GM
490 REM   DEVATY TAKT
500 DATA V201HA V103SAECE02QAM
510 REM  KONEC
520 DATA KONEC HUDBY
```

Při kódování oblíbených hudebních děl pro xxx hraní na Commodore 128 používejte metody popsané v této části manuálu.

x x x x x x x x x x x x

V předchozí části manuálu jsme viděli, jak používat nové účinné příkazy v jazyku BASIC 7.0 pro mod C 128. V následující části vysvětlíme, jak používat zobrazení se 40 a 80 sloupci na Commodoru 128.

Č Á S T 8

Úvod

V modech C 128 a CP/M je možné použít jak mod se 40 sloupci na obrazovce, tak mod s 80 sloupci. V jediném programu lze použít oba mody. Každý tvar obrazovky se používá pro jiné účely. Obrazovka se 40 sloupci je typ obrazovky, používaný Commodorem 64. V zobrazení se 40 sloupci je možné používat grafické funkce Commodoru 128 pro kreslení kružnic, grafiky, sprajtů, čtverců a jiných obrazců s vysokým rozlišením nebo multibarevném grafickém modu. Mimoto je možné používat sprajty.

V modu 80 sloupců je k dispozici dvojnásobek znaků než v modu se 40 sloupci v každém programovém řádku. V modu 80 sloupců je možné volit pomocí klávesnice grafické znaky a možné barvy.

Abychom zcela využili možnosti obou formátů obrazovky, můžeme psát programy s použitím dvou monitorů, každý z nichž provádí část programu. Například textové výstupy mohou být vyvedeny na monitor s 80 sloupci, zatímco grafické výstupy - na monitor o 40 sloupcích.

Tlačítko 40/80

Pro definování délky obrazovky se 40 nebo

80 sloupci se používá tlačítko 40/80.
Toto tlačítko je účinné pouze když se provede jedna z následujících operací:

1. Počítač se zapojuje
2. Je stlačeno tlačítko RESET
3. Tlačítka RUN/STOP a RESTORE jsou stlačena současně.

Tlačítko 40/80 funguje podobně jako tlačítko SHIFT/LOCK, tj. bude aktivováno po stlačení a zůstane aktivováno dokud je nestiskneme podruhé. Stiskněte tlačítko poté, co byla provedena jedna ze tří uvedených operací - obrazovka bude aktivována v modu 40 sloupců. Jestliže naopak bude stlačeno dříve než zapneme počítač, přestože byla splněna jedna z předchozích podmínek, obrazovka bude uvedena do modu 80 sloupců. Není možné aktivovat formát odlišný od běžně používaného (40 nebo 80 sloupců) pomocí tlačítka 40/80. Jestliže musíte přezkoušet tuto nutnost, je nutné stisknout a pustit tlačítko ESC následované tlačítkem X.

Použití již hotového softvéru v modu 80 sloupců

Většina programů CP/M používá zobrazení s 80 sloupci; tak byly aplikační soubory připraveny pro použití na Commodoru 128.

Poněvadž šířka normálního listu papíru je pro tiskárnu 80 znaků, může normální word-processor pro 80 sloupců zobrazit informace na obrazovce přesně tak, jak se objeví na papíru. Rovněž programy pro elektronické karty jsou často specifikovány pro formát 80 sloupců, aby bylo dost místa na různé kolonky a kategorie informací. Rovněž mnoho databankových souborů a telekomunikačních programů vyžadují nebo mohou používat zobrazení s 80 sloupci.

Tvorba programů s 80 sloupci

Mimo provádění programů již připravených k použití může šířka obrazovky s 80 sloupci sloužit k vytváření vlastních programů uživatele. Již jsme ukázali, že při psaní řádku delšího než 40 sloupců na obrazovce se 40 sloupci, se po dokončení řádku pokračuje na následujícím řádku. To by mohlo způsobit zmatek při psaní řádků a rovněž programové chyby. Zobrazení s 80 sloupci dovoluje vyhnout se chybám tohoto typu, protože poskytuje více místa pro řádky programu.

Současné používání 40 a 80 sloupců

Výhodou výstupu se 40 sloupci je možnost použít grafiku s bodovou matricí, zatímco zobrazení s 80 sloupci umožňuje obdržet

výstupy složitých textů a jiné komerční aplikace. Jestliže vlastníte dva monitory, můžete psát "dělené" programy a používat pro text zobrazení s 80 sloupci a pro grafiku zobrazení se 40 sloupci. V programu lze použít specialní příkazy (GRAPHIC 1,1) pro přechod k provádění grafických příkazů ve zobrazení se 40 sloupci. Při použití duálního monitoru (který zobrazuje formáty se 40 i 80 sloupci) se příkaz GRAPHIC 1,1 vkládá do programu, abychom obdrželi grafiku ve formátu 40 sloupců. Abychom však zobrazili tuto grafiku, musíme do programu zahrnout "přepínač" na monitor se 40 sloupci. Jestliže napíšete program tohoto typu, je radno zahrnou do něj informaci, která se zobrazí na obrazovce a připomene uživateli, aby použil přepínač režimu monitoru. Například, je třeba napsat program, který žádá uživatele, aby zadal data a poté na základě tohoto vstupu kreslí grafy. Nápis "PREPNI NA 40 SLOUPCU PRO ZOBRAZENI GRAFIKY" upozorňuje uživatele, aby změnil způsob zobrazení výsledků.

Jak již bylo řečeno, je možné přejít z formátu 80 sloupců do formátu 40 sloupců po zapnutí počítače pomocí příkazu ESC X.

Následující příklad ukáže, jak je možné v programu použít duální obrazovku:

```
10 GRAPHIC 5,1: SCNCLR: REM PREPNE NA 80 SLOU
     PCU A VYCISTI JE
20 PRINT "ZACNI V 40 SLOUPCICH VOLBOU KOMPO
     ZITNIHO VIDEA"
30 PRINT "VSTUP VASEHO DUALNIHO MONITORU"
40 PRINT
50 PRINT "STISKNI RETURN AZ BUDES HOTOV"
60 GETKEY A$: IF A$ <> CHR$(13) THEN 60
70 GRAPHIC 2,1: REM MOD DELENE OBRAZOVKY
80 CHAR 1,8,18,"BIT MAP/TEXT DELENA OBRAZOVKA"
90 FOR I=70 TO 220 STEP 20
95 CIRCLE 1,I,50,30,30: NEXT I
100 PRINT
110 PRINT "PREPOJ NA 80 SLOUPCU. PREPNI NA
     SVEM"
120 PRINT "DUALNIM MONITORU NA RGBI VIDEOVS
     TUP"
130 PRINT "POTE STISKNI RETURN KDYZ JE VSE
     PRIPRAVENO"
140 GETKEY A$: IF A$ <> CHR$(13) THEN 140
150 GRAPHIC 5,1: REM VSTUP PREPNUT NA 80
     SLOUPCU
160 FOR J=1 TO 10
170 PRINT "JSES NYNI V MODU 80 SLOUPCU"
180 NEXT J: PRINT
190 PRINT "NYNI PREPNI ZPET NA 40 SLOUPCU":
     PRINT
200 PRINT "STISKNI RETURN AZ BUDES HOTOV"
```

```
210 GETKEY A$: IF A$ <> CHR$(13) THEN 210
220 GRAPHIC 0,1: REM VYSTUP PREPNUT NA 40
      SLOUPCU
230 PRINT
240 FOR J=1 TO 10
250 PRINT "NYNI JSI V TEXTOVEM VYSTUPU SE
      40 SLOUPCI"
260 NEXT J
270 END
```

Každý formát obrazovky má své výhody; oba typy zobrazení lze kombinovat v jednom programu, aby se doplňovaly. Zobrazení o 40 sloupcích nabízí více místa pro samy programy. Mimoto, toto zobrazení umožňuje provádět širokou škálu programů vyvinutých pro mod s 80 sloupci.

x x x x x x x x x x x x x x x x x x x x x x

Paragrafy této kapitoly pojednávaly o řadě vlastností a funkcí Commodoru 128 v modu C 128. Následující kapitola ukáže, jak používat Commodore 128 v modu C 64.