SKALKA

5. MODUS C 64

5. M O D U S C 64

Obsluha klávesnice v módu C 64 je podrobně popsána v kap.3.1., proto se k ní nebudeme znovu blíže vracet.

POZOR!

Modus C 64 podporuje v každém případě jen klávesnici ASCII. Přepne-li se do módu C 64, měla by se nejdříve uvolnit klávesa ASCII/DIN, tedy je třeba ji odblokovat, jinak by nesouhlasilo obsazení klávesnice s označením kláves.

5.1. BASIC 2.0 v módu C 64

Tato část není vodítkem k naučení se všeobecných technik programování nebo programového jazyka BASIC, ale příručkou pro uživatele obeznámené s všeobecnými koncepty programování, kteří na následujících stranách najdou důležitá fakta k sestavení efektivních programů Basicu. Začátečníkům doporučujeme prostudovat základní lekce kursu Basicu.

V této části je popsán jazykový rozsah základní verze 2.0.- Commodore-Basicu. Tato verze je platná pro modus C 64, tak i pro modu C 128. Kapitola 4 /modus C 128/ obsahuje popis rozšíření Commodore-Basicu do verze 7.0 a je platný pouze pro modus C 128.

Vzhledem k tomu, že byly struktura jazyka a syntaxe následujícího výkladu probrány již v kap. 4.1 až 4.4., je zde od toho upušteno.

5.2. POPIS JEDNOTLIVÝCH PŘÍKAZŮ, INSTRUKCÍ, FUNKCÍ A PROMĚNNÝCH

K ulehčení vyhledávání určitých Basic-příkazů, instrukcí, funkcí a proměnných, nejsou následující popisy rozčleněny podle popsané struktury jazyka Basicu v kap. 4.4, nýbrž jsou všechny jazykové částice shrnuty a alfabeticky uspořádány.

F U N K C E ABS

Formát: v=ABS(x)

Účel: Poskytuje absolutní hodnotu výrazu x.

Příklad:

```
PRINT ABS(7*(-5))
35
READY.
```

F U N K C E ASC

Formát: v=ASC(x$)

Účel: Poskytuje numerickou, celočíselnou hodnotu mezi 0 a 255, která představuje ASCII-kód prvního znaku řetězce x$. Má-li řetězec x$ délku 0 /prázdný řetězec/, vydá se chybové hlášení ILLEGAL QUANTITY.

Příklad:

```
10 A$="TEST"
20 PRINT ASC(A$)
RUN
84
READY.
```

FUNKCE ATN

Formát: v=ATN(x)

Účel: Poskytuje tangens velikosti úhlu x vyjádřené v obloukové míře v rozsahu -PI/2 až PI/2. Výraz x může být jakýmkoliv numerickým typem; výpočet ATN je v každém případě prováděn binárně ve formátu s pohyblivou řádovou čárkou.

Příklad:

```
10 INPUT A
20 PRINT ATN(A)
RUN
?3
1.24904577
READY.
```

F U N K C E CHR$

Formát: v$ = CHR$(n)

Účel: Poskytuje jedno-byteový řetězec, jehož znak má ASCII-kód n. Proto musí n ležet v rozsahu mezi 0 a 255. CHR$ se používá k vytváření speciálních znaků nebo řídících kódů.

Příklady:

```
PRINT CHR$(147)      Maže obrazovku
PRINT CHR$(66)       Vytiskne B na pozici kurzoru.
```

I N S T R U K C E CLOSE

Formát: CLOSE číslo souboru

Účel: Ukončuje vstup nebo výstup přes vstupní/výstupní kanál.

Poznámky: Číslo souboru je číslo mezi 1 a 255, pod kterým byl instrukcí OPEN otevřen údaj.

Souvislost mezi určitým datem a číslem souboru se instrukcí CLOSE ruší. Údaj pak může být otevřen instrukcí OPEN pod tím samým nebo jiným číslem souboru, nebo také může být pod tímto číslem souboru otevřen jiný libovolný údaj.

CLOSE použitá na sekvenčním výstupním údaji, zapíše do data poslední vyrovnávací paměť pro data /datenpuffer/ a ukončí jej koncovou značkou data.

Příklad:
```
10 OPEN 4,4
20 PRINT#4,"TOTO JSOU TISKOVA DATA"
30 CLOSE 4
```

P Ř Í K A Z CLR

Formát: CLR

Účel: Nastavuje všechny numerické proměnné na nulu, všechny řetězcové proměnné na délku 0, vyprazdňuje zásobníkovou /sklípkovou/ paměť a paměť pro pole a nastavuje ukazatel volného paměťového místa zpět na hodnotu, která vyplývá z velikosti BASIC-programu bez všech proměnných.

Poznámky: CLR může být prováděn také uvnitř BASIC-programu. Program potom může pokračovat, přihlédne-li se ke shora uvedeným podmínkám, zejména k těm, které se vztahují na GOSUB.

Příklad:
```
X=25
CLR
PRINT X
0
READY.
```

P Ř Í K A Z CMD

Formát: CMD číslo souboru[,seznam výrazů]

Účel: Adresuje přístroj na spojovací místo vstupu/výstupu a nechává tento přístroj po výstupní operaci v adresovaném stavu.

Poznámky: CMD má stejný seznam parametrů jako instr. PRINT#.

Příkaz - CMD pokračování

Příklad:

```
REM SEZNAM PROGRAMU VYDAT NA TISKARNU
OPEN 4,4
CMD4,"SEZNAM PROGRAMU"
LIST
PRINT# 4,"CMD-MODUS SE UKONCI"
CLOSE4
```

P Ř Í K A Z CONT

Formát: CONT

Účel: Pokračuje v programu, který může být přerušen nebo ukončen stisknutím klávesy STOP nebo instrukcemi STOP nebo END.

Poznámky: Provádění programu bude pokračovat přesně na tom místě, na kterém byl program přerušen. Nastane-li přerušení po indikaci textu instrukcí INPUT stiskem klávesy RETURN bez předchozího zadání textu, bude program pokračovat opakováním této indikace /?nebo text/.

CONT se používá běžně ve spojení s instrukcí STOP k hledání chyb v programech. Po přerušení programu mohou být indikovány pomocné výsledky /mezisoučty/ ev. mohou být změněny instrukcemi v přímém módu. Provádění programu bude pokračovat zadáním CONT nebo GOTO společně s určitým číslem řádku v přímém módu.

CONT je neplatný, byl-li program přerušen chybovým hlášením nebo byl-li během přerušení pozměněn.

Příklad: viz příklad u instrukce STOP.

F U N K C E COS

Formát: v=COS(x)

Účel: Poskytuje kosinus hodnoty x v obloukové míře/v radianech/ Výpočet COS(x) probíhá binárně ve formátu s pohyblivou řádovou čárkou.

Příklad:

```
10 X=2*COS(.4)
20 PRINT X
RUN
1.84212199
READY.
```

I N S T R U K C E DATA

Formát: DATA seznam konstant

Účel: Ukládá numerické a/nebo řetězcové konstanty, které mohou být vybaveny instrukcí READ /viz tam/.

Poznámky: Instrukce DATA nejsou instrukce způsobilé k tomu, aby mohly stát na libovolném místě v programu. Každá DATA-instrukce může obsahovat tolik konstant, kolik se vejde /rozdělených čárkami/ do příkazového řádku /80 znaků/. Počet DATA-instrukcí je libovolný. Instrukce READ čte jednotlivé DATA-řádky v pořadí jejich čísel řádků. Data obsažená v těchto řádcích jsou chápána, nezávisle na jejich čísle a umístění v programu, jako kontinuální seznam elementů.

Seznam konstant: může obsahovat řetězcové a/nebo numerické konstanty každého formátu, tzn. řetězce, s pohyblivou řádovou čárkou nebo celá čísla. Numerické výrazy nejsou dovoleny. Řetězcové konstanty v instrukcích DATA musí být umístěny v uvozovkách jen tehdy, když obsahují čárky, dvojtečky, nebo statisticky významná prázdná místa, umístěná před a/nebo po.

Typ proměnné /numerický nebo řetězec/ deklarovaný v instrukci READ musí souhlasit s typem konstanty obsaženým v příslušných instrukcích DATA.

Ukazatel čtení může být umístěn instrukcí RESTORE /viz tam/ na začátek první DATA-instrukce-.

Příklad:
```
10 DATA"TO","POCASI","JE","DNES","HEZKE!"
20 FOR I=1 TO 5
30 READ A$
40 PRINT A$;"";
50 NEXT
RUN
TO POCASI JE DNES HEZKE!
READY
```

I N S T R U K C E DEF FN

Formát: DEF FN jméno [(seznam argumentů)]
= definice funkce

Účel: Definuje a pojmenovává uživatelem naprogramovanou BASIC-funkci

INSTRUKCE DEF FN - pokračování

Poznámky:

Jméno musí být dovolené jméno proměnné. Toto jméno, uvedené před FN je považováno jako jméno funkce.

Seznam argumentů je argument funkce, který je označen v definici funkce jednou nebo více proměnnými s pohyblivou řádovou čárkou. Poslední jsou potom při vyvolání funkce nahrazeny aktuálními parametry.

Definice funkce je libovolný výraz, který obsahuje operaci, která má funkci provést. Délka výrazu je omezena instrukčním řádkem Basicu /80 znaků/. Jména proměnných, použité v tomto výrazu slouží pouze formální definici funkce a nejsou zaměnitelná s proměnnými programu stejného jména. Jméno proměnné, použité v difinici funkce může být uvedeno jako parametr, není však podmínkou. Je-li parametrem, je jeho hodnota při vyvolání funkce nahrazena; v opačném případě se použije nynější hodnota proměnné.

Pomocí DEF FN nelze definovat uživatelsky specifické řetězcové funkce.

Je-li ve jménu funkce specifikován typ proměnné, potom se přizpůsobí hodnota výrazu tomuto typu dřív, než bude předán vyvolané instrukci. Je-li deklarován typ proměnné ve jménu funkce, který nepatří k typu, jenž výraz poskytuje, vydá se chybové hlášení TYPE MISMATCH.

Instrukce DEF FN musí být provedena dřív, než je tímto definovaná funkce poprvé vyvolána, jinak dojde k vydání chybového hlášení UNDEFINED FUNCTION.

DEF FN nemůže být použita v přímém módu.

Příklad:
```
410 DEF FNAB(X) =X↑3/Y↑3
420 T=FNAB(I)
```

Řádek 410 definuje funkci, která je v řádku 420 vyvolána. Přitom je proměnná X nahrazena aktuální hodnotou I. Proměnná Y si podrží hodnotu jí přiřazenou v momentě vyvolání funkce.

I N S T R U K C E DIM

Formát: DIM seznam indikovaných proměnných

Účel: Definuje maximální počet prvků proměnných pole a rezervuje paměť pro proměnné pole.

Poznámky: Je-li použito jméno proměnné bez předchozí instrukce DIM, je povolen maximální index 10. Zadá-li se index vyšší hodnoty, vydá se chybové hlášení BAD SUBSCRIPT.

Nejmenší index pole je stále 0. Indicie musí být celočíselné.

Instrukce DIM - pokračování

Instrukce DIM umísťuje všechny prvky specifikovaného pole na počáteční nulu, popř. prázdný řetězec.

Pole mohou být deklarována až 255 dimenzemi, ze kterých může každá obsahovat max. 32767 elementů/prvků/. V každém případě je velikost pole omezena použitelnou pamětí.

Příklady:

```
10 DIM A( 20)
20 FORI=0 TO 20
30 READ A(I)
40 NEXT
50 DATA 1,2,3...
10 DIM R3( 5,5):REM 36 ELEMENTY
10 DIM DØ( 2,2,2):REM 27 ELEMENTU
```

I N S T R U K C E END

Formát: END

Účel: Ukončuje průběh programu, uzavírá všechna otevřená data a umísťuje interpret do přímého módu.

Poznámky: Instrukce END k dokončování programu mohou stát na jakémkoliv místě programu.

Instrukce END nevytváří při instr. STOP obrazovkové hlášení /např. BREAK/

Na konci programu /poslední řádek/ není instrukce END povinná.

Příklad: 520 IF K> 1000 THEN END

F U N K C E EXP

Formát: v=EXP(x)

Účel: Poskytuje x-tou mocninu čísla e. x musí být menší nebo rovno 88.02969191, jinak se vytvoří chybové hlášení OVERFLOW.

Příklad:

```
10 X=5
20 PRINT EXP( 5-1)
RUN
54.5981501
READY.
```

I N S T R U K C E FOR...NEXT

Formát: FOR num prom = x TO y [STEP z]
.
.
.
NEXT [num prom][,num prom...]

Účel: Umožňuje vícenásobné zpracování sledu příkazů,instrukcí, a/nebo funkcí v programové smyčce s definovaným číslem průchodů.

Poznámky:

num prom se použije jako čítač průchodů a musí být proměnnou s pohyblivou řádovou čárkou. První numerický výraz ~~je~~ x je počáteční hodnota, druhý numerický výraz y je koncová hodnota počitadla.

Všechny instrukce a programové řádky po instrukci FOR až k první instrukci NEXT se provedou. Potom se zvýší počítadlo o hodnotu z a vyzkouší se, zvýšila-li se koncová hodnota y. Pokud se hodnota nezvětšila, rozvětví interpret zpět k instrukci po instrukci FOR a průběh se zopakuje. Je-li počítadlo vyšší než y, pokračuje program po instrukci NEXT dále.
Toto se rozumí pod pojmem FOR...NEXT- smyčka.

Zadá-li se za z negativní /minusová/ hodnota, musí být konečná hodnota y menší než hodnota počáteční x. y se v tomto případě při každém průběhu zmenší o hodnotu z, až je počítadlo nižší než konečná hodnota y.

Nezadá-li se STEP z, zvýší se počítadlo při každém průběhu o 1.

FOR...NEXT-smyčky mohou být i rozčleněny,tzn. jedna smyčka může být přiřazena uvnitř smyčky jiné. Každý smyčkový čítač pak musí obdržet své vlastní jméno.
Všem čítačům v rozčleněných smyčkách stačí jedna instrukce NEXT, následovaná jednotlivými proměnnými čítačů ve správném pořadí a rozdělenými čárkami, mají-li j dnotlivé instrukce NEXT následovat bezprostředně po sobě.

Proměnné v instrukci NEXT mohou být vypuštěny. V tomto případě se vztahuje každá NEXT-instrukce k naposledy insterpretované instrukci FOR. Nalezne-li interpret NEXT-instrukci bez předcházející instrukce FOR, vydá chybové hlášení NEXT WITHOUT FOR a přeruší program.

Vzhledem k omezené /ohraničené/ sklípkové paměti, lze dohromady rozčlenit max. 9FOR...NEXT smyček

Příklady na následující straně vysvětlují všechno řečené blíže:

Instrukce FOR...NEXT - pokračování

Příklad 1:

```
10 REM ROZCLENENE SMYCKY
20 FOR I=1 TO 3:FOR J=1 TO 3:PRINT I;J
30 NEXT J,I
RUN
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
READY:
```

Příklad 2:

```
10 REM ZMENA PROMENNYCH PO UMISTENI SMYCKY
20 K=10
30 FOR I=1 TO K STEP2
40 PRINT I;
50 K=K+10
60 PRINT K
70 NEXT
RUN
1 20
3 30
5 40
7 50
9 60
READY.
```

Příklad 3:

```
10 REM DRUHA HODNOTA JE MENSI NEZ PRVNI
20 J=0
30 FOR I=1 TO J
40 PRINT I
50 NEXT
RUN
1
READY.
```

V tomto případě proběhne smyčka jen jednou, protože je počáteční hodnota větší než hodnota konečná, což však bude přezkoušeno teprve při dosažení instrukce NEXT.

Příklad 4:

```
10 REM CELOCISELNA PROMENNA JAKO CITAC
20 FOR I%=1 TO 10:PRINT I%:NEXT
RUN
?SYNTAX ERROR IN 20
READY:
```

POZOR!
Celočíselné proměnná jako smyčkové čítače jsou zakázány.

F U N K C E FRE

Formát: v=FRE(x)

Účel: Poskytuje počet ještě nepoužitých Byte v programové paměti Basicu. Za x lze zadat libovolný argument, protože nemá ani v nejmenším účinek. Musí však být k dispozici jako jalový argument.

Poznámky: Protože je předávané číslo zpracováno interpretem jako číslo celé, jehož rozsah je definován pouze mezi -32768 a 32767, může ?FRE(0), činí li volná paměť více než 32767 Byte, poskytovat také negativní /záporné/hodnoty.
V tomto případě se vypočítá skutečně volná paměť ze součtu 65536 a indikovaného negativního čísla.

Příklad:

```
PRINT FRE(0)
1433
READY.
```

I N S T R U K C E GET a GET#

Formát: GET [#log číslo data,] seznam proměnných

Účel: Čte znak z klávesnice nebo z data a přiděluje tento znak další proměnné v seznamu proměnných.

Log číslo data
Celočíselná hodnota mezi 1 a 255, pod kterým byl údaj otevřen.

Poznámky:
GET bez zadání logického čísla data přečte z vyrovnávací paměti klávesnice /klávesnicový puffer/ kód poslední stisknuté klávesy a přiřadí jej následující proměnné /numerické nebo řetězcové/ v zadaném seznamu proměnných. Nebyla-li předtím stisknuta žádná klávesa, poskytne GET hodnotu 0 ev. řetězec o délce 0 /prázdný řetězec/.

GET# čte znak z data, otevřeného pod logickým číslem data. Byl-li otevřen údaj s přístrojovou adresou 0, potom je GET# identická s GET, poněvadž je klávesnici přiřazena přístrojová adresa 0.

Příklad:

```
10 PRINT"CEKAT NA STISKNUTOU KLAVESU"
20 GET A$:IF A$="THEN 20
```

I N S T R U K C E GOSUB a RETURN

Formát: GOSUB číslo řádku
.
.
.
RETURN

Účel: Odskakuje do podprogramu, který začíná číslem řádku a po provedení podprogramu se vrací zpět do hlavního programu.

Poznámky:

Instrukce RETURN v podprogramu způsobí zpětný skok k instrukci, která následuje po naposledy interpretované instrukci GOSUB.

Podprogram může obsahovat více instrukcí RETURN, má-li být skok zpět vázán na různé podmínky.

Podprogram může stát na libovolném místě programu, musí však být interpretem rozlišitelný od částí hlavního programu. K zamezení nechtěného průběhu podprogramu, by měla před podprogramem stát instrukce STOP, END nebo GOTO, která provede programové řízení okolo podprogramu. Stojí-li podprogramy na začátku programu, provedou se rychleji.

Podprogramy mohou být rozděleny až do 23 úrovní.

Příklad:

```
10 GOSUB 40
20 PRINT"Z PODPROGRAMU ZPET"
30 END
40 PRINT"DO PODPROGRAMU"
50 RETURN
RUN
DO PODPROGRAMU
Z PODPROGRAMU ZPET
READY:
```

I N S T R U K C E GOTO

Formát: GOTO číslo řádku

Účel: Za všech podmínek odskakuje z normálního programového sledu ke specifikovanému číslu řádku.

Poznámky:

Označuje-li číslo řádku řádek s proveditelnou instrukcí, bude tento,i potom následující řádky zpracovány.

Neexistuje-li specifikovaný řádek, indikuje se chybové hlášení UNDEF'D STATEMENT ERROR.

Příklad:

```
10 READ R
20 PRINT"R=";R,
30 A=3.14*R↑2
40 PRINT"PLOCHA=";A
50 GOTO 10
60 DATA 5,7,12
```

```
RUN
R=5  PLOCHA=78.5
R=7  PLOCHA=153.86
R=12 PLOCHA=452.16
?OUT OF DATA ERROR IN 10
READY
```

I N S T R U K C E IF

Formát: IF výraz THEN instrukce
IF výraz GOTO číslo řádku

Účel: Umožňuje odskok k jiným instrukcím nebo do různých částí programu, v závislosti na logickém obsahu pravdy numerického výrazu.

Poznámky:

Je-li výsledek výrazu logická"pravda", tedy různý od 0, provedou se klausule THEN nebo GOTO.

Po THEN může následovat číslo řádku ke skoku nebo více instrukcí.
Po GOTO může následovat jenom číslo řádku.Je-li logický výsledek Výrazu "nepravda", tedy nula, budou klausule THEN nebo GOTO ignirivány a program bude pokračovat následujícím příkazovým řádkem.

IF...THEN -instrukce mohou být také rozčleněny, přičemž je členění omezeno jen délkou příkazového řádku /80 znaků/:

```
IF A=B THEN B=C THEN PRINT"A=C"
```

Je-li instrukce IF...THEN, zadaná v přímém módu,následovaná číslem řádku, generuje interpret chybové hlášení UNDEFINED STATEMENT, zvláště byl-li předtím zadán řádek s tímto číslem.

Při použití IF k testování hodnoty, která byla složena z výpočtu s pohyblivou řádovou čárkou, je třeba brát v potaz to, že interní záznam hodnoty nemusí být přesný. Proto by měl být test dělán v rozsahu, ve kterém se přednost obměňuje.
Tedy: při testování proměnné na vypočtenou hodnotu 1.0 by se mělo postupovat následovně:

```
IF ABS(A-1.0)< =1.0E-6 THEN...
```

Tento test vydá výsledek "pravda", je-li hodnota A rovna 1.0 s relativní chybou menší než 1.06E-6.

Příklad 1:

```
100 IF I THEN GET I
```

Tato instrukce zkouší na stlačené klávese, když hodnota I není nulová.

Příklad 2:

```
100 IF(I> 10) AND ( I< 20) THEN
    DB=1979-1:GOTO 300
110 PRINT"PREKROCENI ROZSAHU"
```

Je-li I větší než 10 a menší než 20, bude DB vypočítána a program bude pokračovat řádkem 300. Jinak bude proveden řádek 110.

I N S T R U K C E INPUT

Formát: INPUT ["text";] seznam proměnných

Účel: Umožňuje datový vstup přes klávesnici během provádění programu. Datový vstup se ukončuje stiskem klávesy RETURN.

Poznámky:

Narazí-li interpret na instrukci INPUT, pozastaví se průběh programu, na obrazovce se vydá otazník, který ukazuje, že program očekává vstup dat; kurzor bliká.
Byl-li text specifikován, zobrazí se před otazníkem. Potřebná data pak mohou být zadána přes klávesnici. Tato data jsou pak přiřazena proměnné/ým/ v seznamu proměnných ; proto musí souhlasit číslo základní informace /rozdělené čárkami/ s číslem proměnné v seznamu.

Proměnné v seznamu mohou být jména numerických a řetězcových proměnných /také proměnná pole/. Typ každé udané základní informace musí souhlasit s typem korespondující proměnné.

Zadané řetězce nemusí být umístěny v uvozovkách, protože obsahují čárky a/nebo dvojtečky.

Vstup dat je omezen na délku logického obrazovkového řádku /80 znaků/. Kvůli otazníku může být tedy zadáno pouze 78 znaků nejvíc. Překročí-li se toto číslo, vezme INPUT poslední zadaný obrazovkový řádek jako celý vstup. Za logický obrazovkový řádek budou pokládána data obrazovky až do 80 znaků, počítáno od začátku řádku až po kód zpětného chodu vozíku /klávesa RETURN/.

Zadá-li se při INPUT chybný typ data, např. řetězcová data namísto numerických/, vydá se hlášení ?REDO FROM START a čeká se na správné zadání /vstup/.

INPUT není dovolen v přímém módu. V tomto případě se vydá chybové hlášení ILLEGAL DIRECT.

Bylo-li zadáno příliš mnoho základních informací, vydá se hlášení EXTRA IGNORED. Při malém počtu základních informací se nahradí chybějící indikací ???.

Příklad 1:

```
10 REM ZADANIPRILIS MNOHA ZNAKU
20 INPUT A$
30 PRINT:PRINT A$
RUN
?DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD

****

****
```

Bylo zadáno 78 krát D a 4 krát *. Protože si INPUT při překročení max. délky všímá pouze posledního logického řádku obrazovky, přiřadí se řetězcové proměnné A$ pouze **** .

Instrukce INPUT - pokračování

Příklad 2:

```
10 INPUT X  NA DRUHOU
20 PRINT X"DRUHA MOCNINA JE"X↑2
RUN
?5
5 NA DRUHOU JE 25
READY.
```

Příklad 3:

```
10 PI=3.14159265
20 INPUT"ZADEJ POLOMER";R
30 A=PI*R↑2
40 PRINT"PLOCHA KRUHU JE";A
50 PRINT
60 GOTO 20
RUN
ZADEJ POLOMER?7.4
PLOCHA KRUHU JE 172.033614
ZADEJ POLOMER? atd.
```

I N S T R U K C E INPUT#

Formát: INPUT# log číslo data, seznam proměnných

Účel: Pročítá informace ze sekvenčních nebo relativních dat a přiděluje je proměnným programu.

Poznámky:

log číslo data je číslo, pod kterým byl údaj instrukcí OPEN při vstupu otevřen.

seznam proměnných obsahuje jména proměnných, jimž byly přiděleny údajové prvky z data. Typ data musí odpovídat typu proměnné.

INPUT# nevydává jako indikaci ?, byla-li jako vstupní přístroj zvolena klávesnice.

Základní informace v údaji musí být následně seřazeny tak, jako by byly zadány z klávesnice. Předcházející a následující prázdná místa, kódy zpětného chodu vozíku a kódy posuvu řádků budou ignorovány. Rozdělovacími znaménky mezi obsahy proměnných mohou být čárky nebo dvojtečky; kód zpětného návratu rozděluje v každém případě od sebe jednotlivé elementy data.
Při nepřizpůsobení dat a typů proměnných se vydá chybové hlášení FILE DATA ERROR. Při pokusu číst údaje na konci data nebo při překročení času na spojovacím místě vstupu /viz systémová proměnná STATUS/ poskytuje INPUT# kód zpětného návratu /CHR$/13//.

INPUT# může číst maximálně 80 znaků.

Instrukce INPUT#- pokračování

Příklad:

```
10 REM CTENI Z KAZETY AZ DO KONCE SOUBORU DAT
20 OPEN 3,1,0
30 INPUT# 3,A$
40 IF STATUS AND 64 THEN FE=1
50 PRINT A$
60 IF FE THEN CLOSE 3:END
70 GOTO 30
```

Dotaz počítačového slova Status /zde řádek 40/ je podrobně popsán v této kap. u systémové proměnné STATUS.

F U N K C E INT

Formát: v=INT(x)

Účel: Poskytuje nejvyšší celé číslo, které je menší nebo rovno x.

Příklad:

```
PRINT INT(99.89),INT(-12.11)
99            -13
READY.
```

F U N K C E LEFT$

Formát: v$=LEFT$(x$,n)

Účel: Předává řetězec, který se skládá z n levých znaků x$. n musí lešet v rozsahu mezi 0 a 255. Je-li n větší, než délka x$, vydá se společný znakový řetězec x$. Je-li n rovno nule, předá se znakový řetězec o délce 0 /prázdný znakový řetězec/.

Příklad:

```
10 A$="COMMODORE KANCELARSKE STROJE"
20 B$=LEFT$(A$,9)
30 PRINT B$
RUN
COMMODORE
READY
```

F U N K C E LEN

Formát: v=LEN(x$)

Účel: Předává počet znaků ve znakovém řetězci x$. Počítají se všechny znaky, tedy i netisknutelné a prázdné.

Příklad:

```
10X$="COMMODORE BRAUNSCHWEIG"+CHR$(13)
20 PRINT LEN(X$)
RUN
23
READY.
```

I N S T R U K C E LET

Formát: [LET] proměnná=výraz

Účel: Přiřazuje hodnotu výrazu proměnné.

Poznámky: Slovo LET není povinné, tzn. při přiřazování jedné hodnoty jedné proměnné stačí znaménko rovnosti.

Příklad:
```
110 LET D=12:LET E=12*2
120 LET F=144/12
130 LET SUM=D+E+F
```

má stejný význam jako:
```
110 D=12:E=12* 2
120 F=144/2
130 SUM=D+E+F
```

P Ř Í K A Z LIST

Formát: LIST [řádek1][- [řádek2]]

Účel: Listuje část nebo celý program na současně aktivovaném výstupním přístroji /obrazovka, tiskárna, kazeta, floppy-disk/.

Poznámky: Interpret Basicu se po provedení LIST vždy přesune do přímého módu.

Příkazem LIST bez zadání čísel řádků se prolistuje celý program počínaje nejnižším číslem řádku. Listování se ukončí buďto koncem programu nebo stiskem klávesy STOP.
Zadá-li se pouze řádek1, bude nalistován pouze tento jeden řádek.

Příklady:

LIST	Listuje celý program nacházející se v paměti.
LIST 500	Listuje řádek 500.
LIST 150-	Listuje všechny řádky od řádku 150 až do konce programu.
LIST -1000	Listuje všechny řádky od začátku programu až do řádku 1000 včetně.
LIST 150-700	Listuje všechny řádky od řádku 150 až po řádek 700 včetně.

P Ř Í K A Z LOAD

Formát: LOAD"jméno data"[,přístr.adresa]
[posuv]

Účel: Zavádí programový údaj z externího paměťového přístroje /kazeta,floppy-disk/ do paměti počítače.

Poznámky:
Jméno data je to jméno, pod kterým byl uložen program s instrukcí SAVE /viz tam/ do externího paměťového přístroje.

Přístrojová adresa musí být celočíselná hodnota mezi 1 a15. Vynechá-li se přístrojová adresa, bude se zavádět program z přístroje 1 /kazetové umístění/.

Posuv Celočíselná hodnota, která udává má-li být program zaveden na začátek Basic-programové paměti/0/ nebo na adresu, se kterou bylo ukládáno i na kazetu nebo na floppy-disk/1/. Přednastavena je zde 0. Hodnota tohoto parametru musí být vždy 1, mají-li být zaváděny programy strojového jazyka.
Příkazem LOAD se všechna otevřená data uzavřou stejně jako všechny umístěné proměnné a případný program nacházející se v paměti budou vymazány dřív, než bude zaveden specifikovaný program.
Provede-li se LOAD uvnitř programu, spustí se tím uložený Basic-program, přičemž všechna otevřená data zůstanou otevřená. Tím lze spolu sřetězit více programů nebo programových segmentů.Umístěné proměnné zůstanou sřetězením zachovány.

Zaváděný program však smí být ~~nejvíc~~ maximálně tak velký,jako vyvolávaný.

Příklady: Zavádění programu z kazety:

```
LOAD"TESTPRG"
PRESS PLAY ON TAPE
```

Zavádění programu ve strojovém jazyce z diskety:

```
LOAD"MULT",8,1
```

F U N K C E LOG

Formát: v=LOG (x)

Účel: Poskytuje přirozený logaritmus x.
x musí být větší než 0; jinak se vydá chybové hlášení ILLEGAL QUANTITY.

Příklad:

```
PRINT LOG( 45/7)
1.86075234
READY
```

F U N K C E MID$

Formát: v$=MID$(x$,n[,m])

Účel: Poskytuje část řetězce x$ o m znacích počínaje n-tým znakem x$. n a m musí ležet v rozsahu mezi 0 a 255. Vypustí-li se m, nebo je-li od napravo od n-tého znaku k dispozici méně znaků než m znaků v x$, vydají se od n-tého znaku všechny pravé znaky x$. Je-li n větší, než délka x$, předá se znakový řetězec o délce 0 /prázdný string/.

Příklad:
```
10 A$="DOBRY"
20 B$="JITRO VECER DEN"
30 PRINT A$;MID$( B$,8,5)
RUN
DOBRY VECER
READY
```

P Ř Í K A Z NEW

Formát: NEW

Účel: Maže stávající program, nacházející se v paměti, stejně jako všechny proměnné.

Poznámky: NEW se zadává zpravidla v přímém módu, dřív než bude vydán nový program. Interpretační program uvede počítač /v každém případě/po provedení NEW do přímého módu.

I N S T R U K C E ON...GOSUB a ON...GOTO

Formát: ON výraz GOTO seznam čísel řádků
O ON výraz GOSUB seznam čísel řádků

Účel: Odskakuje k jednomu z mnoha specifikovaných čísel řádků v závislosti na hodnotě, která je poskytována výrazem.

Poznámky: Hodnota výrazu určuje, ke kterému číslu řádku ze seznamu program odskočí. Je-li hodnota např. 4, zobrazí 4.číslo řádku v seznamu cíl skoku.

Před skokem se v každém případě přemění hodnota na celé číslo, tzn. případná desetinná místa se odříznou.

U instrukce ON...GOSUB musí každé specifikované číslo řádku označit začátek řádku podprogramu.

Je-li hodnota výrazu negativní, vydá se chybové hlášení ILLEGAL QUANTITY. Je-li rovna nule, nebo je větší, než počet čísel řádků udaných v sez amu, bude pokračovat program řádkem, následujícím po této instrukci.

Instrukce ON...GOSUB a ON...GOTO - pokračování

Příklad: 100 ON L-1 GOTO 150,300,320,220

Pro L=2 se odskočí na řádek 150, pro L=3 na řádek 300 atd.

I N S T R U K C E OPEN

Formát: OPEN log.číslo data[,přístrojová adresa
[, sekundární adr[,"jméno data]]]

Účel: Otevírá vstup/výstup data ev. vstupní/výstupní kanál přístroje.

Poznámky: log.číslo data musí ležet mezi 1 a 255.

Není-li zadána přístrojová adresa, vezme se 1 /kazetová stanice/.

Za jméno data lze zadat až 16 znaků dlouhý znakový řetězec.

V případě vstupu/výstupu přes spojovací místo IEEE 488 /sériově nebo paralelně/ se každou instrukcí GET#, INPUT# nebo PRINT# vyšle přístrojová ev. specifikovaná adresa sekundární přes spojovací místo.

U floppy-diskových dat bude typ data přijatý a P /programový údaj/, nebude-li připojen S nebo U /sekvenční údaj/ rozdělené od sebe čárkami, ke jménu data.
Zadá-li se po další čárce W, otevře se specifikovaný údaj k psaní / k záznamu/, v opačném případě ke čtení-.

U dat z kazety označuje sekundární adresa 0 vstupní údaj, 1 výstupní údaj a 2 výstupní údaj, u kterého bude instrukcí CLOSE za datem napsána koncová značka pásky.

Data ev. datové kanály mohou být otevřeny pro klávesnici /přístr.adr.0/, kazetový přístroj/přístr. adr. 1/ nebo obrazovku /přístr. adr. 3/.

Přístrojové adresy větší než 3,se vztahují na přístroje, které mohou být připojeny na spojovací místo IEEE488 /sériově nebo paralelně/ /např. 4 pro tiskárnu, 8 pro floppy-disk/.

Příklad:

```
10 REM OTEVRENI KAZETY
15 REM REGISTRACNI SOUBOR DAT
20 REM PRILOZENA ZNACKA KONCE PASKY
30 OPEN 6,1,2,"SOUBOR DAT"
40 FOR I=1 TO 10
50 PRINT# 6,CHR$( I)
60 NEXT
70 CLOSE6
```

Další příklady otevírání dat v provozu floppy-disku jsou popsány v kap. 6.6.

F U N K C E PEEK

Formát: v=PEEK(n)

Účel: Poskytuje obsah paměťového místa s adresou n jako celočíselnou hodnotu mezi 0 a 255. Za n musí být zadány celočíselné hodnoty mezi 0 a 65535. PEEK je protějšek instrukce POKE /viz tam/.

Příklad: A=PEEK(53281) AND 15

Hodnota proměnné A je dle této instrukce kód pro nyní nastavenou barvu pozadí na obrazovce /C 64 modus/.

I N S T R U K C E POKE

Formát: POKE n,m

Účel: Zapisuje 8-bitovou-binární informaci do specifikovaného paměťového místa /buňka s daty/

Poznámky: Hodnota celočíselného výrazu n označuje paměťové místo určené k zápisu; hodnota celočíselného výrazu m označuje datum,ulkádané do paměti.

n musí ležet v rozsahu mezi 0 a 65535 a m v rozsahu mezi 0 a 255.

Protějškem instrukce POKE je funkce PEEK, jejíž argument označuje paměťové místo, jehož obsah má být vybrán.

POKE aPEEK jsou efektivními pomůckami pro ukládání dat, zavádění podprogramu ve strojovém jazyce i pro předávání parametrů a výsledků mezi Basic-hlavními programy a podprogramy ve strojovém jazyce.

Příklad 1:
```
10 REM RIZENI BARVY A ZNAKU
20 POKE 1024,1:POKE 55296,6
```

Zde se zobrazí písmeno A v modré barvě na HOME-pozici obrazovky /modus C 64/

Příklad 2:
```
10 REM VSECHNY KLAVESY MAJI OPAKOVACI FUNKCE
20 POKE 650,128
30 REM POUZE RIDICI KLAVESY KURZORU MAJI
40 REM OPAKOVACI FUNKCI
50 POKE 650,0
```

F U N K C E POS

Formát: v=POS(n)

Účel: Poskytuje momentální sloupcovou pozici kurzoru na obrazovce. Krajní levý sloupec je pozice 0.

n je jalový argument, který však z formálních důvodů musí být zadávám.

Příklad: IF POS(0)> 20 THEN PRINT CHRØ(13)

I N S T R U K C E PRINT a PRINT#

Formát: PRINT [#log.číslo data,][seznam výrazů]

Účel: Vydává data na obrazovku, nebo přes specifikovaný výstupní kanál.

Poznámky: Nežadá-li se seznam výrazů, vytiskne se prázdný řádek. V opačném případě budou výrazy vyhodnoceny a jejich hodnoty se vydají v příslušné instrukci OPEN pod log.číslem data na specifikovaném výstupním přístroji nebo výstupní údaj.

Jsou povoleny numerické a/nebo řetězcové výrazy.

Znakové řetězcové konstanty musí být vždy uvedeny v uvozovkách.

Pozice každého tištěného data se určuje interpunkcí, která od sebe dělí data na seznamu. Interpretační program Basicu člení tiskový řádek do tiskových zón po 10 prázných místech. Čárka v seznamu výrazů způsobuje, že hodnota následujícího výrazu vytiskne od začátku další tiskové zóny, naproti tomu střední ~~zvojtečka~~ způsobí, že se další hodnota vytiskne hned za předcházející hodnotu.

Čárka nebo středník na konci seznamu výrazů značí, že budou hodnoty další instrukce PRINT vytištěny v další tiskové zóně toho samého řádku nebo v bezprostřední souvislosti toho samého řádku. V obou případech bude potlačen návrat vozíku.

Je-li tisknutý řádek delší než 40 znaků /nebo 80 u módu C 128/, pokračuje výstup na dalším fyzikálním řádku.
Po vytištěných číselných hodnotách následuje vždy prázdné místo. Před pozitivními číselnými hodnotami je postaveno prázdné místo, před negativními hodnotami znak minus. Každé číslo mezi 0 a 0.01 se reprodukuje ve vědeckém exponenciálním znázornění /viz kap.2.4/ Instrukce PRINT /ne PRINT#/ může být zkrácena otazníkem/?/.

Instrukce PRINT a PRINT# -pokračování

Příklad 1:

```
10X=5
20 PRINT X+5,X-5,X*( -5),X↑5
RUN
10          0          -25        3125
READY.
```

Čárky mezi výrazy na řádku 20 způsobí, že se každá hodnota vytiskne na začátek 10-místné šířky tiskové zóny.

Příklad 2:

```
10 INPUT X
20 PRINT X"MOCNITEL 2 JE"X↑2"A";
30 PRINT X"MOCNITEL 3 JE"X↑3
40 PRINT
50 GOTO 10
RUN
?9
9 MOCNITEL 2 JE 81 A 9 MOCNITEL 3 JE 729

? atd.
```

Zde způsobí středník na konci řádku 20 to, že hodnoty obou instrukcí PRINT v řádcích 20 a 30 budou vytištěny na jednom stejném řádku. Řádek 40 způsobí výtisk prázdného řádku.

Příklad3:

```
10 FOR X=1 TO 5
20 J=J+5
30 K=K+10
40?J;K;
50 NEXT
RUN
5 10 10 20 15 30 20 40 25 50
READY.
```

U tohoto příkladu byl v řádku 40 zvolen za instrukci PRINT znak?. Středníky způsobí výtisk jednotlivých hodnot bezprostředně po sobě, oddělených od sebe 2 prázdnými místy /každé číslo je následováno jedním prázdným místem,u kladných čísel je jedno prázdné místo přednastaveno/. Prolistovává-li se program s LIST, nahradí se znak ? v řádku 40 slovem PRINT.

I N S T R U K C E READ

Formát: READ seznam proměnných

Účel: Čte údaje z instrukce DATA /viz tam/ a přiřazuje je obsazuje je proměnnými.

Poznámky: Instrukce READ může být použita jen v souvislosti s instrukcí DATA.

Každé proměnné ze seznamu, která může být numerickou nebo řetězcovou proměnnou, se vždy přiřadí jen jedna hodnota z instrukce DATA. Typy dat a proměnných musí souhlasit. V opačném případě se vydá chybové hlášení SYNTAX ERROR.

Instrukce READ - pokračování

Jedna jediná instrukce READ může sekvenčně vyvolat více instrukcí DATA, stejně tak více READ-instrukcí může vyvolávat jednu DATA-instrukci.

Je-li počet proměnných v seznamu proměnných větší, než počet elementů v instrukci/ích/ DATA, vydá se chybové hlášení OUT OF DATA. Je-li v seznamu specifikováno méně proměnných, než částic, které jsou k dispozici v instrukci /ích/ DATA, čtou následující instrukce READ dopusud nečtené elementy. Nenásledují-li v takovém případě žádné další instrukce READ, zůstanou nadpočetné datové elementy nepovšimnuty. K opakovanému čtení instrukcí DATA od začátku může být použita instrukce RESTORE /viz tam/.

Příklad 1:

```
80 FOR I=1 TO 10
90 READ A(I)
100 NEXT
110 DATA 3.08,5.19,3.12,3.98,4.24
120 DATA 5.08,5.55,4.00,3.16,3.37
.
.
.
```

V této výseči programu se budou číst elementy instrukcí DATA řádků 110 a 120 v poli A.

Příklad 2:

```
10 PRINT"PLZ","MESTO","ZEME"
20 READ PZ,SZ,LZ
30 DATA 6000,"FRANKFURT","HESSEN"
40 PRINT PZ,SZ,LZ
RUN
PLZ        MESTO        ZEME
6000            FRANKFURT   HESSEN
READY.
```

Zde byla přečtena a vytištěna numerická a řetězcová data z instrukce DATA v řádku 30.

I N S T R U K C E REM

Formát: REM [komentář]

Účel: Touto instrukcí mohou být do programu vloženy vysvětlené komentáře.

Poznámky: Bude-li program listován, nebudou instrukce REM provedeny, ale přesně reprodukovány.

Od instrukce GOTO nebo GOSUB lze odskočit k instrukci REM. Program pak bude pokračovat další proveditelnou instrukcí, následující po instrukci REM.

Instrukce REM - pokračování

Text,následující po REM, nemůže obsahovat žádné znaky, které byly zadány při stisknuté klávese SHIFT, protože jsou při LISTování interpretovány a vytištěny jako klíčová slova Basicu.

Příklad:

```
10 REM VYPOCET STREDNI
15 REM RYCHLOSTI
20 FOR I=1 TO 20
30 SUM=SUM+V(I)
40 NEXT
50 VM=SUM/(I-1)
```

I N S T R U K C E RESTORE

Formát: RESTORE

Účel: Umísťuje snímací ukazatel instrukce READ na začátek první DATA-instrukce v programu.

Poznámky: po provedení instrukce RESTORE vyvolá v paměti další READ instrukce první element data v první DATA instrukci v programu.

Příklad:

```
10 READ A,B,C
20 PRINT A,B,C
30 RESTORE
40 READ D,E,F
50 PRINT D,E,F
60 DATA 57,68,79
RUN
57          68          79
57          68          79
READY.
```

F U N K C E RIGHT$

Formát: v$=RIGHT$(x$,n)

Účel: Poskytuje pravé n znaky z řetězce x$.

Je-li n rovno nebo větší, než délka x$, předá se x$.
Pro n=0 bude předán řetězec o délce 0 - prázdný řetězec.
n musí mít hodnotu mezi 0 a 255.

Příklad:

```
10 A$="COMMODORE PSACISTROJE"
20 PRINT RIGHT$(A$,14)
RUN
PSACISTROJE
READY.
```

F U N K C E RND

Formát: v=RND(x)

Účel: Poskytuje náhodné číslo, které se produkuje různě, v závislosti na argumentu x.

x> 0: Bude poskytována vždy další hodnota z řady náhodných čísel, která se vypočítají numerickým algoritmem v interpretačním programu BASICu. Řada je nezávislá na hodnotě argumentu x a při zapojení počítače bude inicializovaná náhodnou počáteční hodnotou.

x< 0: Každý argument x inicializuje novou řadu náhodných čísel. Stejné argumenty vedou ke stejným řadám náhodných čísel.

x=0: Různými, na sobě nezávislými ~~algoritmy~~ časovými spínači, se algoritmem vytvoří náhodné číslo.

Příklad:

```
10 FOR I=1 TO 5
20 PRINT INT(RND(1)* 100);
30 NEXT
RUN
24  30  83  45  1
READY.
```

P Ř Í K A Z RUN

Formát: RUN[číslo řádku]

Účel: Spouští přítomný BASIC - program, nacházející se v programové paměti.

Poznámky: Zadá-li se číslo řádku, spustí se ~~tím~~ ~~označ~~ program, označený tím řádkem. V opačném případě začne provádění programu nejnižším číslem řádku.

Před spuštěním programu se provede nejprve CLR skrze RUN.

Program se ukončí vrácením interpretačního programu do přímého módu, když:

1. již nejsou k dispozici žádné proveditelné řádky.
2. se provede instrukce END nebo STOP.
3. vznikne-li během provádění programu chyba.

P Ř Í K A Z SAVE

Formát: SAVE["jméno data"[,přístr.adr [,opce]]]

Účel: Ukládá programový údaj BASICu na specifikovaný výstupní přístroj.

Poznámky: Nezadá-li se přístrojová adresa, uloží se BASIC-program, nacházející se v programové paměti, na kazetu /přístrojová adresa 1/.

Při ukládání na kazetu se může zadat za opci nula / bez koncové značky pásky/ nebo hodnota různá od nuly /koncová značka pásky po programovém údaji/.

Nezadá-li se jméno data /povoleno pouze při ukládání na kazetu/, neuloží se vůbec žádné jméno. Dosadí-li se za jméno data řetězcová proměnná, pak musí být umístěna v závorkách.

Příklady:
```
SAVE
SAVE"TESTPRG"
SAVE(A$),1,0
```

Další příklady a detailnější popis instrukce SAVE v provozu floppy-disku, najdete v kap. 6.2.

F U N K C E SGN

Formát: v=SGN(x)

Účel: Poskytuje znaménko argumentu x v následujícím kódovaném tvaru:

X>0	poskytuje	1
X=0	-"-	0
X<0	-"-	-1

Příklad: ON SGN(A)+2 GOTO 100,200,300

Program se rozvětví /odskočí/ na řádek 100, je-li hodnota A negativní, na řádek 200, rovná-li se nule, a na řádek 300, je-li hodnota A kladná.

F U N K C E SIN

Formát: v=SIN(x)

Účel: Poskytuje sinus x v radianech /obloukova míra/. Výpočet SIN(x) probíhá binárně ve znázornění s pohyblivou řádovou čárkou. Mezi SIN(x) a COS(x) existuje souvislost

COS(x)=SIN(x+3.14159265/2)

Příklad:
```
PRINT SIN(1.5)
.997494987
READY
```

F U N K C E SPC

Formát: SPC(n)

Účel: Poskytuje n prázdných míst. SPC lze použít ve spojení s instrukcí PRINT nebo PRINT#. n musí mít hodnotu mezi 0 a 255.

Příklad:
```
PRINT"ZDE"SPC(15)"TU"
ZDE              TU
READY.
```

F U N K C E SQR

Formát: v=SQR(x)

Účel: Poskytuje druhou odmocninu x. x musí být větší nebo rovno nule.

Příklad:
```
10 FOR X=10 TO 25 STEP 5
20 PRINT X,SQR(X)
30 NEXT
RUN
10                3.16227766
15                3.87298335
20                4.47213595
25                5
READY.
```

S Y S T É M O V Á P R O M Ě N N Á STATUS

Formát: v=STATUS
v=ST

Účinek: Poskytuje stavový byte počítače, jehož obsah je postaven na základě poslední vstupní/výstupní operace. Přitom platí dle použitého vstup/výstupního přístroje následující stavové hodnoty:

ST-Bit	ST-hodnota	kazeta	ser.IEC-Bus
0	1		časový průběh při záznamu
1	2		časový průběh při čtení/snímání/
2	4	krátký blok	
3	8	dlouhý blok	
4	16	fatální chyba	
5	32	chyba v kontrolním součtu	
6	64	konec data	konec souboru
7	-128	konec pásky	přístroj nezapojen

Systémová proměnná STATUS- pokračování

Příklad:
```
10 OPEN 6,1,2,"MASTER FILE"
20 GET# 6,AZ
30 IF ST AND 64 THEN 60
40 ?AZ
50 GOTO 20
60 ?AZ:CLOSE6
```

Údaj MASTERFILE bude čten a indikován až do konce souboru kazety.

I N S T R U K C E STOP

Formát: STOP

Účel: Přerušuje probíhající program a umísťuje interpretační program do přímého módu.

Poznámky: Instrukce STOP mohou stát na libovolném místě v programu. Provede-li se STOP, ohlásí toto interpret hlášením:
BREAK IN nnnnn

přičemž nnnnn je číslo řádku, u kterého byl program přerušen.
Instrukce STOP neuzavírá otevřená data, jako instrukce END.
Po instrukci STOP může program pokračovat zadáním CONT/viz tam/ v přímém módu.

Příklad:
```
10 INPUT A,B,C
20 K=(A+3)/2:L=B*3
30 STOP
40 M=C*K+100:PRINTM
RUN
?1,2,3
BREAK IN 30
READY.
CONT
106
READY.
```

F U N K C E STRØ

Formát: vØ=STRØ(x)

Účel: Poskytuje řetězcové znázornění x.

Příklad:
```
10 INPUT"ZADEJ PROSIM CISLO";N
20 PRINT N,LEN(STRØ(N))
30 GOTO 10
RUN
ZADEJ PROSIM CISLO?-124
-124        4
ZADEJ PROSIM CISLO?2
2           2
ZADEJ PROSIM CISLO atd.
```

Ve druhém případě je délka STRØ(2) proto 2, protože je v řetězcovém znázornění kladných čísel předinstaveno číslu vždy prázné místo.

I N S T R U K C E SYS

Formát: SYS výraz[,seznam parametrů]

Účel: Předává řízení programu na podprogram ve strojovém jazyce, který začíná u specifikované adresy / viz funkce USR/.

Poznámky:
Hodnota výrazu musí být celé číslo mezi 0 a 65535. Označuje adresu v paměti programu C 128, u které začíná podprogram ve strojovém jazyce.
Návrat do hlavního programu BASICu probíhá asemberovým příkazem RTS.

V seznamu parametrů mohou být zadány parametry, které mají být předány podprogramu ve strojovém jazyce.

Vyhodnocení těchto parametrů však musí proběhnout podprogramem ve strojovém jazyce.

Příklad: SYS 7+2↑ 12,X,Y

Sled znaků ,X,Y musí být vyhodnocen podprogramem strojového jazyka, který začíná u adresy 28676, tak, že textový ukazatel interpretačního programu ukazuje po ukončení podprogramu na Y.

F U N K C E TAB

Formát: TAB(n)

Účel: Tabeluje přes n sloupců do aktuálního řádku na obrazovce. Stojí-li kurzor před provedením TAB(n) napravo od n-tého sloupce, přetabeluje se n sloupců následujícího řádku.

TAB se vztahuje vždy na začátek řádku v krajně levém sloupci obrazovky /sloupec 0/. Nejkrajnější pravé pozice v řádku je potom sloupec 39 /ev. sl.79 v módu C128/
n musí ležet mezi 0 a 255.
TAB lze použít jen ve spojení s PRINT nebo PRINT#.

Příklad:
```
10 PRINT"ZBOZI"TAB(15)"CENA":PRINT
20 READ A$,B$
30 PRINT A$TAB(15)B$
40 DATA"MASLO","KCS 10"
RUN
ZBOZI            CENA
MASLO            KCS 10
READY.
```

F U N K C E TAN

Formát: v=TAN(x)

Účel: Poskytuje tangens x v radianech. Výpočet TAN(x) probíhá binárně ve znázornění s pohyblivou řádovou čárkou.

Příklad:
```
PRINT TAN(5)/2
-1.6902575
READY.
```

S Y S T É M O V Á PROMĚNNÁ TIME

Formát: v=TIME
v=TI

Účel: Poskytuje momentální stav interních systémových hodin, které vypisují čas každou 1/60 sekundy. Systémové hodiny nejsou hodiny reálného času. Inicializují se zapojením počítače.

Příklad:
```
PRINT TI
154788
READY.
```

I N S T R U K C E TIME$
SYSTÉM.PROMĚNNÁ TIME$

Formát: TIME$=x$ jako instrukce
TI$=x$

v$=TIME$ jako systémová proměnná
v$=TI$

Účel: Jako instrukce se systémové hodiny nastaví na zvolený čas ve tvaru 6-Byte/ového/ řetězce v uspořádání:

hhmmss /hodiny,minuty,sekundy/

Tento čas /průběh času/ je hlídán systémem a může být kdykoliv vyvolán a snímán jako systémová proměnná.

Příklad:
```
10 INPUT"PROSIMCAS/HHMMSS/
         ZADAT";TI$
20 FOR I=1 TO 1000:NEXT
30 PRINT TI$
RUN
PROSIMCAS/HHMMSS/ ZADAT?141223
141234
READY.
```

F U N K C E USR

Formát: v=USR(x)

Účel: Odskakuje k podprogramu ve strojovém jazyce, jehož startovací adresa musí být napřed uložena do buněk s adresou 85 a 86 nulové strany - zero page /řídící paměť interpretačního programu, viz dodatek H/. Buňka 85 obsahuje nízkohodnotovou a buňka 86 vysokohodnotovou část adresy. Argument x bude předán v prvním akumulátoru pohyblivé řádové čárky interpretačního programu, ve kterém také musí být uložen výsledek podprogramu, aby mohl BASIC-hlavní program přiřadit tento výsledek ihned jedné proměnné.

Příklad:
```
10 B=T*SIN(Y)
20 C=USR(B/2)
30 D=USR(B/3)
```

F U N K C E VAL

Formát: v=VAL(xØ)

Účel: Poskytuje numerickou hodnotu řetězce, který se skládá z číslic. Mimo to jsou dovoleny na správném místě také znaky . + - a E. Začíná-li řetězec jiným znakem, než číslicí, tečkou, znaménkem plus nebo minus, vydá VAL(xØ) nulu.

Příklad:

```
10XØ=".0053"
20 PRINT VAL(XØ)
RUN
5.3E-03
READY.
```

P Ř Í K A Z VERIFY

Formát: VERIFY["jméno data"[,přístrojová adresa]]
[,posuv]

Účel: Srovnává přítomný program, nacházející se v programové paměti, s programem, uloženým na specifikovaném výstupním přístroji podle byte-ů a hlásí případné rozdíly.

Poznámky:

Přístrojová adresa - přednastavena je 1 /kazetový přístr./

Nezadá-li se jméno data /dovoleno pouze u souboru na kazetě/ nastane srovnávání s prvním programem, nalezeným na kazetě.

Posuv Celočíselná hodnota 0 nebo 1, která udává, ~~jestli~~ začíná-li srovnávaný program na začátku programové paměti BASICu /0/, nebo u jiné adresy /1//např. podprogram ve strojovém jazyce/. Přednastavena je zde hodnota 0.

Příklady:

```
VERIFY"PRGFILE"
PRESS PLAY ON TAPE
OK
FOUND PRGFILE
VERIFYING
VERIFY OK nebo VERIFYING ERROR
READY.
```

VERIFY "MULT",8,1

Program strojového jazyka MULT na disketě se bude srovnávat s rezidentním pamětovým programem, který nezačíná na začátku programové paměti BASICu.

I N S T R U K C E WAIT

Formát: WAIT adresa,n[,m]

Účel: Pozastaví provádění programu, dokud zadaná paměťová buňka C 128 nepřijme specifikovaný vzor bit-u.

Poznámky: Ke kontrole specifikovaného vzoru bit-u se mezi obsahem, označeným adresou paměťové buňky a celočíselným výrazem m, vytvoří výhradné /exkluzivní/ spojení NEBO. Tento výsledek se sloučí logickým A s celočíselným výrazem n. Je-li výsledek nula, otestuje se znovu vzor bitu specifikované paměťové buňky. Je-li výsledek různý od nuly, provede se následující BASIC-instrukce. Když se m vynechá, přejme se jeho hodnota s nulou.

POZOR!
Instrukce WAIT nemůže být přerušena klávesou STOP.

Příklady: WAIT 1,16,16
Provádění programu se pozastaví tak dlouho, dokud se se nestiskne klávesa kazetové stanice.

WAIT A,2↑n s n=0,1,....,7

Provádění programu se pozastaví tak dlouho, ~~dokud~~ až je bit n, ~~specifikovaný~~ paměťové buňky, specifikované skrze A, logicky 1.

5.3. B A R V Y A G R A F I K A V MÓDU C 64.

Nyní jsme se již naučili jazykový rozsah BASICu v módu C 64. Jednou z vynikajících vlastností tohoto módu je také možnost vytvářet barevnou grafiku. Jedním z nejjednodušších příkladů je, v průběhu tohoto oddílu popsaný program pružného míče. Možnosti módu C 64 jsou však nesrovnatelně větší.

V této části Vám ukážeme, jak můžete vytvářet barevné grafiky a jak je nasadit např. ve hrách.

Vycházíme přitom ze standartních barev obrazovky /světlemodré písmo na tmavě modrém pozadí se světle modrým okrajem/ a chceme Vám ukázat, jak můžete tyto barvy měnit a jak nasadit mnoho grafických symbolů v módu C 64.

Stanovení barev příkazy PRINT

Nastavení barev probíhá podle vzoru, který lze vytvořit jednoduchými prostředky na obrazovce TV přijímače nebo monitoru.

Použijete k tomu klávesu CTRL na levé straně klávesnice. Klávesa CTRL se používá stále dohromady s jinou klávesou tak, že se nejprve přidrží klávesa CTRL a potom se stiskne zvolená jiná klávesa.

Máte k dispozici škálu 16 barevných znaků. Použitím klávesy CTRL a číselných kláves,můžete vytvořit následující barevné kombinace:

1	2	3	4
černá	bílá	červená	zelená
5	6	7	8
fialová	tmavě zelená	modrá	žlutá

Použitím klávesy Commodore/úplně dole vlevo na klávesnici/ společně s číselnými klávesami obdržíte následující barvy:

1	2	3	4
světle hnědá	hnědá	světle červená	tmavě šedá
5	6	7	8
šedá	světle zelená	světle modrá	světle šedá

Nyní zadejte NEW a udělejte následující experiment:

Po začátku řádku 10 PRINT" stiskněte společně klávesu CTRL a klávesu 1. Poté pusťte klávesu CTRL a stiskněte klávesu S. Nyní stiskněte opět klávesu CTRL společně s klávesou 2 a poté samostatně klávesu P atd.

Tímto způsobem volte jednu barvu za druhou a písmeny udejte slovo Spektrum.

```
10 PRINT"  S P E K T R U M
           ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
CTRL       1 2 3 4 5 6 7 8
```

Tak jako u řízení kurzoru se také barevné řídící znaky znázorní jako grafické znaky.

Současným stiskem klávesy CTRL a klávesy 3 se objeví znak anglické libry a stiskem CTRL a klávesy 7 - šipka doleva.

Následující tabulka udává přehled těchto barevných kódů:

Tabulka:

klávesa	barva	výstup	klávesa	barva	výstup
CTRL 1	černá		C= 1	oranž	
CTRL 2	bílá	E	C= 2	hnědá	
CTRL 3	červená		C= 3	sv.červená	
CTRL 4	tyrkysová		C= 4	šedá 1	
CTRL 5	fialová		C= 5	šedá 2	
CTRL 6	zelená		C= 6	sv.zelená	
CTRL 7	modrá		C= 7	sv.modrá	
CTRL 8	žlutá		C= 8	šedá 3	

výstup-znaky: viz originál str. 5-76
klávesa C= znamená Commodore

Jak jste již určitě zjistili, jsou řídící znaky viditelné jen při listování programu, při provádění programu, tzn. při tisku slova Spektrum se již neobjeví.

Vyzkoušejte si hrou možnosti, abyste se lépe seznámili s řízením barev. Nezapomeňte přitom, že v souvislosti s klávesou Commodore máte ještě další možnosti kombinace barev.

. Poznámka: Po ukončení programu, při kterém jste se zabývali řízením barev, zůstane C 128 v tom módu, který jste naposled zapojili. Stiskem klávesy RUN/STOP a RESTORE přejdete zpět do módu normální barvy.

B A R E V N É KÓDY CHR$

Některé barvy mohou být přímo vyvolány funkcí CHR$. Mají stejný účinek, jako stisk klávesy CTRL a odpovídající číslicové klávesy.

Vyzkoušejte následující příklad:

```
10 PRINT CHR$(147):REM CLR
20 PRINT CHR$(30);"CHR$(30) VYBARVETE MNE"
```

Písmo by mělo být teď zelené. V mnoha případech je použití funkce CHR$ k řízení barev jednodušší, než použití barevných kláves.

Následující program vytvoří na obrazovce barevné pruhy: viz tabulka str. 5-77 /originál příručky k obsluze C 128/

legenda: AUTOMATISCHE FARBBALKEN = automatické barevné pruhy /řádek 10/

řádek 30 REVERSER BALKEN = reverzní pruhy

Udání barev-funkcí PEEK a instrukcí POKE

Nyní se naučte metodu, jak se pohybovat v módu C 64, jak zanést informace na Vámi zvolené místo tak, abyste mohli počítač řídit.

Obsahy jednotlivých paměťových buněk mají zcela určitý význam. Existují paměťové buňky, ve kterých je v módu C 64 pevně stanoveno, jaká barva byla zvolena pro obrazovku, nebo rámeček, které znaky na obrazovce mají být indikovány v jaké barvě a kde mají být umístěny.

Změní-li se obsah těchto paměťových míst, může se tím změnit také parametr.

Barvy mohou být různě obměňovány. Objekty se mohou na obrazovce objevovat a pohybovat.

Zapisujeme-li hodnoty přímo do paměti počítače, děje se tak za pomoci příkazu POKE /viz tam v kap. 5.2/.
Říká se proto taky, že byla hodnota"zapokeovaná" do paměťové buňky.

Paměťová místa je možno vyvolat jejich adresou:
např: 53280 nebo 53281

Zadali jsme zde dvě paměťová místa, jejichž obsahy určují barvu obrazovky a pozadí.

Zadejte, prosím, následující:

POKE 53281,7

Po stisknutí klávesy RETURN obdržíte žlutou obrazovku, protože hodnota 7 definuje žlutou barvu, a zapoke-ovali jsme ji do paměťové buňky, která volí barvu pozadí na obrazovce. Vyzkoušejte to samé s jinými číselnými hodnotami.

Můžete použít každé číslo mezi 0 a 255, smysl však mají jen čísla mezi 0 a 15.

Z následující tabulky je zřejmé, jak jsou přiřazeny číselné hodnoty barvám:

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

Tyto barvy mohou být u různých barevných monitorů úplně odchylné.

Ukažme si nyní různé kombinace barev pozadí a rámečku.

K tomu nám pomůže následující program:

```
10 FOR BA=0 TO 15
20 FOR BO=15 TO 1 STEP -1
30 POKE 53280,BA
40 POKE 53281,BO
50 FOR X=1 TO 2000:NEXT X
60 NEXT BO:NEXT BA
```

Dvě smyčky se dohromady zařadí, aby mohly zachytit všechny kombinace. Dodatečná smyčka v řádku 50 celý proces je trochu zpomalí.
Chcete-li během programu přenosu barev zviditelnit na obrazovce hodnoty, které zodpovídají za právě platný sled barev, připojte následující programový řádek:

```
25 PRINT CHR$(147) ;"RAMECEK=";PEEK(53280)AND 15,
        "POZADI=";PEEK(53281)AND 15
```

GRAFIKA NA OBRAZOVCE

Doposud jste řídili kurzor pouze příkazy PRINT.

Tím lze dosáhnout každého bodu na obrazovce; tato metoda je však velmi pomalá a spotřebuje cenné paměťové místo.
V módu C 64 existují však také paměťová místa, která určují barvy obrazovky.

PAMĚŤ OBRAZOVKY

Obrazovka v módu C 64 se skládá z 25 řádků po 40 znacích. Tím na ni můžete zanést až 1000 znaků k čemuž potřebujete stejné množství paměťových míst, které obsahují informace o tom, které znaky se nacházejí v jednotlivých pozicích na obrazovce. Představte si obrazovku jako pravoúhlou mřížku, přičemž každý malý čtvereček v této mřížce odpovídá jedné paměťové bunce.
Do těchto paměťových buněk můžete vepsat hodnoty mezi 0 a 255 /poke-ovat/.

viz nákres mřížky, originál, str. 5-80
/Spalte= sloupec, Zeile= řádek/

Paměť obrazovky v módu C 64 začíná u adresy 1024 a sahá až k adrese 2023.

Paměťová buňka 1024 přitom odpovídá levému hornímu rohu obrazovky, paměťová buňka 2023 právému dolnímu rohu obrazovky.

Ke získání určité plochy na obrazovce, můžete postupovat podle následujícího početního schematu:
Adresa obrazovky = 1024 + sloupec + 40* řádek

Předpokládejme, že chcete zobrazit cca uprostřed obrazovky /sloupec 20, řádek 12/ míč. Adresa paměti se vypočítá následovně:

1024 + 20 + 40* 12 = 1524

Nyní vymažte obrazovku současným stiskem kláves SHIFT a CLR/HOME a potom zadejte:

POKE 1524,81

Přitom čísla znamenají:

1524 = adresa středu obrazovky
81 = kód znaku /míč/

PAMĚŤ BAREV

Vytvořili jste právě míč ve středu obrazovky. Dosáhli jste toho i bez použití příkazu PRINT. Hodnotu jste zapsali přímo do paměti obrazovky. Míč však zatím nemůžete vidět, protože má stejnou barvu, jako pozadí.

V módu C 64 však ještě existuje další paměť, kterou můžete, změnou obsahu paměti, určit barvy jednotlivých znaků na obrazovce.
Zadejte následující:

POKE 55796,2

Jednotlivá čísla znamenají:
55796 = adresa barevné buňky pro střed obrazovky
2 = barevný kód /červená/

Poté bude míč červený. Protože potřebujete mimo informace o znaku na určité pozici na obrazovce také informace o barvě, náleží ke každé pozici na obrazovce dvě paměťové místa. Paměť barev začíná u adresy 55296 /levý horní roh/ a má, jako paměť obrazovky, 1000 paměťových buněk.

viz obr. na str. 5 - 82 /originál/

Barevné kódy leží mezi 0 a 15 a odpovídají těm, které jsme použili k určení pozadí a rámečku / viz dříve uvedená tabulka barev/

Vzorec k výpočtu adresy paměti barev odpovídá vzorci k výpočtu adresy na obrazovce:

adresa paměti barev = 55296 + sloupec + 40* řádek

Hra s pružným míčem:

```
10 PRINT CHRØ(147): REM CLR/HOME
20 POKE53280,7:POKE53281,0
30 X=1:Y=1
40 DX=1:DY=1
50 POKE 1024+X+39*Y,81
55 POKE 55296+X+39*Y,1
60 FOR T=0 TO 10:NEXT
70 POKE 1024+X+39*Y,32
80 X=X+DX
90 IF X=0 OR X=39 THEN DX=-DX
100 Y=Y+DY
110 IF Y=0 OR Y=24 THEN DY=-DY
120 GOTO 50
```

Poté, co se řádkem 10 vymazala obrazovka, zvolí se v řádku 20 pozadí černé a barva rámečku žlutá.

Proměnné X a Y v řádku 30 jsou řádek a sloupec, kde se míč momentálně nachází.

Proměnné DX a DY řádku 40 udávají horizontální a vertikální směr pohybu míče. DX=+1 odpovídá pohybu směrem do prava. DX=-1 odpovídá pohybu do leva. Analogicky k tomu odpovídají DX a DY pohybu vzhůru ev. dolů.

V řádku 50 se zobrazí míč na určité pozici, určené číslem řádku a sloupce.

V řádku 60 je vložena zpožďovací smyčka.

Na řádku 70 se míč vymaže přepisem s prázdným znakem

V řádku 80 se míč pohybuje správným směrem, sčítáním DX.

Znaménko před DX se obrátí, pokud je v řádku 90 určeno, že se míč dotkne pravého nebo levého okraje.

V řádku 100 a 110 proběhne to samé pro horní a dolní okraj.

Řádek 120 způsobí skok na řádek 50, kde se míč objeví v nově vypočítané pozici na obrazovce.

Změníte-li v řádku 50 číslo 81 za jiné číslo, můžete nahradit míč libovolným znakem.

Následujícím doplněním můžeme udělat program ještě trošku inteligentnější:

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

Řádky 21-27 obsahují náhodně zvolené pozice na obrazovce překážkami. V řádku 115 se vyzkouší funkcí PEEK, narazí-li míč do překážky. Je-li to tento případ, změní se směr pohybu.

5.3.1.

SPRITE - GRAFIKA V MÓDU C 64

Úvod

V předcházejících kapitolách jste viděli, jak lze formátovat příkazem PRINT obrazovku jako tabulku a jak je možno vytisknout znaky na libovolném místě na obrazovce za pomoci příkazu POKE.
Konstrukce pohybujících se obrazů způsobem oběma metodami potíže, protože musí být objekty složeny z předem hotových symbolů.

Dále sebou pohyb a kontrola objektů přináší veliké plýtvání příkazy. Ohraničeným počtem grafických symbolů jste silně omezeni při formování objektů.
Použití Sprites odstraní většinu výše uvedených problémů. SPRITE zobrazí volně programovaný objekt ve vysokorozlišitelné grafice, kterému lze dát libovolný tvar příkazy BASICu. Jednoduchým udáním pozice může být SPRITE na obrazovce posunován. Potřebné výpočty se v módu C 64 vyřídí interně. SPRITEs však mají ještě více předností. Může být měněna jejich barva a srážka dvou Sprite může být registrována jednoduchým způsobem.
Sprite se může pohybovat před nebo za jiným a příkazem lze také měnit jeho velikost.

Díky všem těmto přednostem nastanou při programování jen nepatrné těžkosti. Dodáváme ještě, že se musíte naučit některé ze základů, jak pracovat v módu C 64 a to, jak se interně zpracovávají čísla. Je to koneckonců velmi zajímavé a vůbec ne složité.

NÁVRH SPRITE/ů/

Sprite jsou podloženy speciálním grafickým základem, VIC /modulovou součástkou /Video Interface Chip/ . Práci, která dělá náčrt Sprite, kontrolování jejich pohybu a pozic a udání barvy, přebírá z větší části tento čip.
Rozsah, ve kterém se budou Sprites generovat, se skládá z 64 paměťových míst druhu, které jste poznali u zpracovávání paměti obrazovky a barev.
Každé z těchto paměťových míst si lze představit rozdělené na 8 malých částí, tzv. bity, které mohou být jednotlivě zapojovány a odpojovány a tímto způsobem určují tvar Spritu.
V jakém stavu se jednotlivé bity nacházejí /vyp. nebo zap./, se určí číslem, které se zaznamená do jmenovaného registru.

Dodatečně k těmto speciálním registrům použijeme také paměť v módu C 64, k ukládání informací přes tvar spritů.

V 8 paměťových buňkách /bezprostředně za pamětí obrazovky/ se ukládají data, která sdělují počítači, v jakém paměťovém rozsahu jsou uložena data pro Sprity.

Jaká je struktura Spritů?

Jak již víte, skládá se obrazovka z 25 řádků po 40 znacích. Každou z těchto vzniklých 1000 obrazovkových pozic, můžete obsadit příkazy POKE jedním znakem.

Při navrhování Spritu můžete každý z bodů této matice jednotlivě vyvolat a obdržet tím vyřešení 320x200 /horizontální x vertikální/ bodů pro celou obrazovku.

Objekt, který tímto způsobem vytvoříte /sestavíte/ může mít max. šířku 24 a výšku 21 bodů.

Jako příklad jsme zkonstruovali z tohoto 24 x 21 bodového pole míč, který je níže zobrazen.

Nejlépe bude, načrtneme-li si objekt na rastrovém papíře /např. na milimetrovém papíře/, kde vyznačíte pole o velikosti 24 - šířka krát 21 - výška čtverečků.
Nakreslete nejdříve tvar, který Vám napadne a potom vyplňte políčka, které jste si ohraničili.
Tímto způsobem jste určili tvar Spritů. Musíte je však ještě přeměnit v data, která může počítač zpracovat.

Za tímto účelem nadepište na horní okraj bodového pole 24x21 třikrát za sebou řadu čísel 128, 64, 32, 16, 8, 4, 2, 1. Řádky pole očíslujte od 1 do 21.

Pořiďte si nyní pro každý řádek tři hodnotové tabulky tak, jak je dále uvedeno.
Tabulka 1 odpovídá prvním 8 pozicím jednoho řádku, tabulky 2 a 3 pozicím 9 - 16 ev. 17 - 24.
Přes celá pole napište opět výše uvedenou číselnou řadu 128-1.
V našem náčrtku Spritu obsadíme každé vyplněné políčko číslicí 1, každé prázdné políčko číslicí 0.
Potom přiložíme každému řádku tři tabulky a vypíšeme odpovídající hodnoty do jednotlivých políček.

viz obr. str. 5 - 87 v originále příručky k obsluze C 128

(Gruppe = skupina, Spalte = sloupec, Zeile - řádek)

Jako příklad vezměme řádek 1 výše zobrazeného Spritu. Prvních osm políček je prázdných, tzn. naše tabulka č. 1 obdrží jen nuly. K výpočtu čísla, které může počítač zpracovat, násobíme obsah hodnotou, která je nad jednotlivými políčky uvedena a výsledky pak sečteme. Protože všech 8 políček obsahuje nulu, obdržíme jako první údaj součtu nulu.

Druhá tabulka je následně znázorněna.

128	64	32	16	8	4	2	1
0	1	1	1	1	1	1	1

0 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127

Třetí tabulka první řady obdrží opět nuly. Součet je tedy nula. První řada našeho Spritu bude popsána čísly 0, 127, 0.

Abychom je mohli lehce zhodnotit v programu, napišme je jako DATA - řádek:

DATA 0, 127,0

Jako další příklad údaje druhé řady našeho Spritu:

0 0 0 0 0 0 0 0
1 = 1

1 1 1 1 1 1 1 1

128 + 64 +32 + 16 + 8 + 4 + 2 + 1 = 255

1 1 0 0 0 0 0 0

128+64 = 192

Za řadu 2 obdržíme tedy DATA-řádek:

DATA 1,255,192

Abyste se s popsanou metodou lépe sžili, měli byste si zbývajících 19 DATA-řádků vypočítat sami.

Poté, co tak učiníte, vyzkoušejte následující program:

```
10 REM BEH MICE
20 PRINT CHR$(147):V=53248:REM BASISADR. VIC
30 POKEV+21,4:REM SPRITE 2 AKTIVOVAT
40 POKE2042,13:REM DATA PRO SPRITE 2
50 FORN=0 TO 62:READQ:POKE832+N,Q:NEXT
60 FORX=0 TO 200
70 POKEV+4,X:REM NOVA X-SOURADNICE
80 POKEV+5,X:REM NOVA Y-SOURADNICE
90NEXTX:GOTO60
100 DATA 0,127,0,1,255,192,3,255,224,3,231,244
110 DATA 7,217,240,7,223,240,7,217,240,3,231,224
120 DATA 3,255,224,3,255,224,2,255,160,1,127,64
130 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0,0
140 DATA 62,0,0,62,0,0,62,0,0,28,0
```

/omyl/

Pokud jste všechno zadali správně, letí míč přes obrazovku. Abyste program pochopili, musíte vědět, která paměťová místa Sprite-u /registr/ kontrolují které funkce.

To můžete vyrozumět z následující tabulky:

Registr	Výklad
0	X-ová souřadnice Spritu 0
1	Y-ová souřadnice Spritu 0
2-15	Význam dvojmo jako 0 a 1 pro Sprity 1-7
16	Nejvyšší bit-x-ové souřadnice
21	1=Sprite aktivován, 0=Sprite dezaktivován
23	Sprite se zvětší v Y-ovém směru
29	Sprite se zvětší v X-ovém směru
39-46	Barvy Spritů 0-7

Mimo to musíte vědět, ve kterém z 64 bloků jsou uložena data určitého Spritu.

Tato data stojí v 8 registrech hned za pamětí obrazovky.

2040	41	42	43	44	45	46	2047
Sprite 0	1	2	3	4	5	6	7

Nyní si probereme krok po kroku, jak lze zanést naše Sprity na obrazovku:

- Příkazem POKE udejte do paměťového místa správnou hodnotu /viz dále/, aby se Vámi zvolený Sprite objevil na obrazovce.
- Nechejte ukázat ukazatel-Sprite na to paměťové místo, ze kterého mají být čtena/snímána/ data Spritu.
- S POKE zapište data do tohoto paměťového místa.
- Smyčkou zkonstruujte Xovou a Yovou souřadnici pro pohyb Spritu.
- Dodatečně můžete změnit barvu Spritu nebo jeho velikost /X nebo/a Y-ový směr/. Parametry pro změnu velikosti jsou v registrech 23 a 29

Jednotlivé body programu ještě blíže objasníme.

Řádek 20: V=53248

První adresa paměti VIC se uloží pod V. K této hodnotě potřebujeme už jenom sečíst čísla registru k dosažení absolutní adresy paměti.

Řádek 30: POKEV+21,4

Tento příkaz umožní Spritu 2 objevit se na obrazovce. Podíváte-li se na následující tabulku, můžete rozeznat sounáležitost jednotlivých Spritu:

	128	64	32	16	8	4	2	1	náležející hodnoty
	7	6	5	4	3	2	1	0	čísla Sprite
21	0	0	0	0	0	1	0	0	=4

1 pro zvolený Sprite

Každý Sprite je v registru 21 zastoupen. Tak odpovídá hodnota 4 Spritu 2. Sprite 7 odpovídá hodnotě 128 a oba dohromady 132(128+4). Chcete-li zapojit Sprite 2 a 7, musíte zadat Poke V+21,132.

Řádek 40: POKE2042,13

Počítači bylo uloženo přečíst data pro Sprite 2/odpovídá paměťovému místu 2042/ z paměťového bloku 13 /1 paměťový blok = 64 Byte, tzn. 13*64 = 832/.
Druhé číslo instrukce POKE je počáteční adresa čteného Spritu.

Jeden Sprite spotřebuje 63 Byte. Obsah tabulky, kterou jsme sestavili ke zjišťování datových řádků, odpovídá jednomu Byte. Proto musí být snímáno 63 Byte k zaplnění jednoho Spritu.

Děje se tak následujícím způsobem:

```
50 FORN=0 TO 62:READQ:POKE832+N,Q:NEXT
```

Touto smyčkou budou čteny /snímány/ 63 databyte ve 13.bloku. který začíná u adresy 832 /13x64/.

```
60 FORX=0 TO 200
70 POKEV+4,X
80 POKEV+5,X
```

Protože registry 4 a 5 obsahují x-ové a y-ové souřadnice Spritu 2, způsobí tato část programu /samozřejmě společně s NEXT/, že se Sprite 2 pohybuje diagonálně přes obrazovku.

Protože počátek souřadnic leží v levém horním rohu obrazovky, probíhá pohyb shora na levo směrem vpravo dolů. Počítač čte data dostatečně rychle, aby mohl pohyb kontinuálně zobrazit.

Má- li se přes obrazovku pohybovat více Spritů, přidělí se každému objektu vlastní paměťový rozsah. Řádek 90 způsobí zpětný skok k řádku 60, přičemž se celý proces bude opakovat.

Zbytek programu se skládá z DATA-řádků, které obsahují informace o tvaru míče.

Připojte k programu následující řádek a znovu jej spusťte.

```
55 POKE V+23,4:POKE V+29,4:REM SPRITE ZVETSIT
```

Míč je teď v x-ovém i y-ovém směru dvakrát tak větší, než předtím. Základ tkví v tom, že do registrů 23 a 29 bylo zapsáno číslo 4.

Důležité je dbát na to, aby levý horní roh Spritu zůstal na svém místě.

K vytvoření programu ještě zajímavějšího, připojte ještě následující řádky:

```
30 POKEV+21,12
40 POKE 2042,13:POKE 2043,13
60 FOR X=1 TO 190
75 POKEV+6,X
85 POKEV+7,190-X
```

Na obrazovce se objevil další Sprite /č.3/, protože jsme do registru 21"zapokeovali" 12. 12-ka zapojuje Sprite 2 a 3.

Řádky 75 a 85 způsobí pohyb 3.Spritu po obrazovce.Paměťová místa V+6 a V+7 obsahují x-ové a y-ové souřadnice objektu. Pokud chcete, aby se dělo "ještě něco více",přidejte do programu ještě následující dodatek:

```
30 POKE V+21,28
40 POKE 2042,13:POKE2043,13:POKE2044,13
55 POKE V+23,12:POKE V+29,12
78 POKE V+8,X
88 POKE V+9,100
```

Příkazem POKE na řádku 30, se objeví další Sprite.

V řádku 40 je určeno, že si všechny tři Sprite vzaly své data v bloku 13.

V řádku 55 se Sprite 2 a 3 dvojnásobně zvětšily ve směru x a y.

Řádek 78 způsobí, že se Sprite3pohybuje ve směru x-ové osy.

Protože je v řádku 88 zadána hodnota 100 pro y-ovou souřadnici, pohybuje se Sprite 3 jen horizontálně.

Zadání barvy Sprite

Z tabulky barev znaků si můžete vybrat zvolené kódování barev /0-15/.

Chcete-li dát např. Spritu 1 barvu bledě zelenou, musíte zadat následující příkaz:

POKE V+40,13

Možná, že jste při zkoušení příkladového programu spozorovali, že Sprity nikdy nedosáhly pravý okraj obrazu. Je to proto, že šířka obrazovky odpovídá 320 bodům, x-ový registr však může obsahovat pouze hodnoty mezi 0 a 255. Jak tedy přimějeme objekt k tomu, aby se pohyboval přes celou obrazovku? K tomuto účelu se použije registr č.16, který obsahuje bit s nejvyšší hodnotou /MSB=Most Sigfificant Bit/. Dá se tomu rozumět takto:

Stanoví-li se n-tý bit tohoto registru, nachází se n-tý Sprite v té pozici na obrazovce, která odpovídá X-té hodnotě, větší než 255. Aktuální X-ová hodnota potom vyplyne, připočítá-li se 256 k hodnotě, která je v X-ovém registru,

Má-li např. Sprite 2 zaujmout X-ovou polohu mezi 256 a 320, musí být v registru 16 umístěn druhý bit. Musíte tedy do tohoto registru "zapokeovat" 4/čtyřku/.

POKE V+16,4

Nyní umocněme obsah X-registru, který patří ke druhému Sprite /to je registr č.4/ od 0 do 63. Tímto způsobem získáte zbývající X-ové hodnoty od 256 do 319.

Nejlépe pochopíte tento koncept, budete-li analyzovat následující program, který znázorňuje mírně pozměněnou verzi našeho dosavadního Sprite-programu:

viz další strana

```
10 V=53248:POKE V+21,4:POKE 2042,13
20 FOR N=0 TO 62:READ Q:POKE 832+N,Q:NEXT
25 POKE V+5,100
30 FOR X=0 TO 255
40 POKE V+4,X
50 NEXT
60 POKE V+16,4
70 FOR X=0 TO 63
80 POKE V+4,X
90 NEXT
95 POKE V+16,0
98 GOTO 30
```

Řádek 10 obsahuje počáteční adresu VIC. Sprite 2 se zaktivuje a blok, ze kterého bude čteno, se určí.

V řádku 20 se z DATA-řádků vstoupí do bloku 13.

Řádkem 25 se určí Y-ová souřadnice.
Na řádku 30 se založí smyčka pro X souřadnic 0-255 a vydána bude v řádku 40.

Na řádek 60 se umístí MSB pro Sprite č.2.

V řádku 70 začíná smyčka, která umožní postup Spritu k pravému kraji obrazovky.

Řádek 95 je také tak důležitý, protože je zde opět vypnut MSB, takže může začít pohyb opět na levém okraji obrazu.

5.4. H U D B A V MÓDU C 64.

Vytváření tónů v módu C 64 má dvě hlavní oblasti použití:

Hra hudebních skladeb a vytváření zvukových efektů.

Nyní přistoupíme ke krátkému objasnění toho, jak se všeobecně vytváří program hudby, a vytvoříme si hudební program, se kterým můžete experimentovat.

STRUKTURA HUDEBNÍHO PROGRAMU

Zvuk jednoho tónu je určen čtyřmi hodnotami:

- výška tónu
- hlasitost
- barva zvuku
- úder

Dvě poslední hodnoty způsobí, že můžete sluchem rozlišit různé instrumenty.
Právě těmito důležitými vlastnostmi pak můžete ovlivnit svůj program.
K tomuto účelu se v módu C 64 používá elektronická součástka SID /SOUND INTERFACE DEVICE/. V SIDu je obsažena řada paměťových míst pro parametry, které skládají zvolený zvukový obraz.

Již víte, že je možné v módu C 64 současně vytvořit tři hlasy. Podívejme se nejprve na první z těchto hlasů.
Základní adresu SID zkrátíme proměnnou SI:

SI = 54272

Výška tónu se určí frekvencí. Frekvence je v SID uložena parametrem, který může přijímat hodnoty mezi 0 a 65535. Již v předchozí části jste se naučili, že tak vysoké čísla nelze do paměťové buňky uložit. Proto musíme rozložit frekvenční parametr na vysoko hodnotné a méně hodnotné Byty.

Velmi hodnotný Byte se nazve Hi-Byte, méně hodnotný Lo-Byte. Oba tyto Byte obsadí první dva registry v SID.

FL = SI /Frekvence, Lo-Byte/
FH = SI + 1 /Frekvence, Hi-Byte/

Hlasitosti je v SID určeno 16 stupňů od 0 /vypnuto/ do 15 /plná hlasitost/.
Odpovídající parametr se uloží v registru 24.

L = SI + 24 /hlasitost/

Nyní přijde na řadu barva zvuku:

V podstatě je určena druhem vln, které vytvářejí příslušný hudební nástroj.

V módu C 64 jsou k dispozici čtyři základní tvary:

- trojúhelník
- pilovitý
- obdélníkový
- šum

V následujících příkladech programování se naučíte některým příkladům, jak lze tyto tvary měnit a ovlivnovat je filtrem.

Nejprve nám postačí základní tvary: každý z nich je kontrolován bit-em v registru 4:

W = SI + 4 /tvar vlny/

V tomto registru zapište k výběru výše jmenovaných základních tvarů, parametr 17,33,65 a 129. Zvolte 65 /obdélníková vlna/, dodatečně musíte ještě určit parametr mezi 0 a 4095 pro tzv. klíčovací poměr /poměr mezi zap. a vyp. obdélníku/.

Oba byty tohoto parametru přijdou do registru 2 a 3:

TL = SI + 2 /klíčovací poměr Lo-Byte/
TH = SI + 3 /klíčovací poměr Hi-Byte/

Úder a průběh tónu se nastaví v registrech 5 a 6. Tím se nastavená hlasitost v registru 24 nejprve zvýší, potom opět trochu sníží.
Nyní dosažená hlasitost zůstane zachována tak dlouho, dokud bude tón zapnut; potom tón dozní:

A = SI + 5 /úder/
H = SI + 6 /držení/

Každý z těchto registrů je rozdělen do dvou částí:

Parametr ve čtyřech velmi hodnotných bit-ech A řídí náběh tónu, ve čtyřech méně hodnotových bit-ech,doznění. Malé hodnoty znamenají tvrdě; vysoké hodnoty pomalu, měkce.

Toto platí pro čtyři méně hodnotné Bit-y H, které kontrolují doznívání tónu po jeho odpojení.. Vyšší čtyři Bit-y H určují hlasitost, se kterou má být tón podržen.

Nejvyšší hodnota udává předem nastavenou hlasitost v registru 24, nižší hodnoty tuto hlasitost více nebo méně oslabují.

Příklad programu

Nejprve se musíte rozhodnout, jaké hlasy /tónové generátory/ chcete používat.

Pro každý z těchto hlasů musíte určit čtyři výše zmíněná nastavení /hlasitost, tvar vlny atd./. Můžete použít až tři tónové generátory, náš příklad však užívá pouze hlas č.1.

```
10 SI=54272:FL=SI:FH=SI+1:W=SI+4:A=SI+5:H=SI+6:L=SI+24
20 POKE L,15
30 POKE A,16+9
40 POKE H,4*16+4
50 POKE FH,29:POKE FL,69
60 POKE W,17
70 FORT=1TO500:NEXT
80 POKEW,0:POKEA,0:POKE H,0
```

V řádku 10 jsou definovány jednotlivé adresy registru.
V řádku 20 je nastavena plná hlasitost
V řádku 30 je nastaven úder.
V řádku 40 se kontroluje držení a doznívání tónu.
V řádku 50 je nastavena Hi- a Lo-Byte frekvence, zde pro komorní "a"/440 Hz/- jiné hodnoty viz dodatek F.
Na řádku 60 se reguluje tvar vln. Tento POKE-příkaz musí být nastaven vždy jako poslední, protože nejnižší Bit v tomto registru zapíná ev. odpojuje tónový generátor
V řádku 70 je nastavena smyčka k určení trvání tónu.

V řádku 80 se odpojí nastavení tvaru vln a obalové křivky.

Po zadání RUN můžete uslyšet tón/notu/, která byla tímto proramem vytvořena.

MELODIE v MÓDU C 64

K vytvoření melodie v módu C 64, nemusíte být muzikantem. Uvádíme příklad programu, na kterém je ukázáno, jak na to. Použijeme opět pouze jeden hlas ze tří, které máme k dispozici.

Vymažte pomocí NEW předcházející program a zadejte následující:

```
10 REM STUPNICE
20 SI=54272:FL=SI:FH=SI+1:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 POKE L,15
40 POKE A,9
50 READ X:READY
60 IF Y=-1THEN POKEW,0:END
70 POKE FH,X:POKE FL,Y
80 POKE W,17
90 FOR T=1TO100:NEXT
100 POKE W,0
110 FORT=1TO50:NEXT
120 GOTO 40
130 DATA 17,103,19,137,21,237,23,59,26,20,29,69,
         32,219,34,207
140 DATA-1,-1
```

Chcete-li vytvořit tóny, které se podobají cembalu, musíte pozměnit následujícím způsobem řádek 80:

80 POKE W,33

Tímto POKE-příkazem zvolíte jako tvar vlny pilovitý tvar, čímž obdržíte vyšší harmonicí zvuk, než jak to bylo u doposud používaného trojúhelníkového tvaru vlny.

Volba tvaru vlny je však jen jedna z možností, jak určit charakter zvuku. Speciální volbou číderových hodnot můžete vytvořit z cembala banjo. Dojde k tomu pozměněním řádku 40:
40 POKE A,3

Můžete tedy napodobit zvuk různých nástrojů, jako s pravým syntetizátorem. Nyní probereme to, jakým způsobem k tomuto účelu měnit příslušné obsahy registrů.

DULEŽITÁ VYLADĚNÍ ZVUKU

1. Hlasitost:

Volba hlasitost platí pro všechny tři tónové generátory. Registr,který připadá v úvahu má adresu 54296. Dostanete max. hlasitost, udáte-li do tohoto registru příkazem POKE hodnotu 15:

POKE L,15 nebo POKE 54296,15

K vypnutí tónových generátorů, zapište do registru nulu/0/

POKE L,0 nebo POKE 54296,0

Určete hlasitost na začátku hudebního programu. Programovatelnou změnou hlasitosti můžete však dosáhnout zajímavých efektů.

2. Tvar vlny:

Jak jste viděli již ve výše uvedeném příkladě, určuje tvar vlny charakter zvuku tónu. V módu C 64 můžete každému hlasu nastavit odděleně tvar vlny. K tomu můžete volit z:

- tvaru trojúhelníku
- tvaru pilovitého
- obdélníkového tvaru
- a šumu

Sestavení příslušných adres a jejích obsahů, které odpovídají různým hlasům a tvarům vln, udává následující tabulka. Chcete-li pro první hlas zvolit tvar vlny trojúhelníkový, musíte použít následující příkaz:

POKE W,17 nebo POKE 54276,17

První číslo /adresa/ stojí místo registru. Druhé číslo /obsah adresy nebo registru/ udává momentální tvar vlny.

NASTAVENÍ TVARU VLNY

Registr				Obsah			
Hlas	1	2	3	šum	obdél.	pilovitý	trojúh.
	4	11	18	129	65	33	17

Tuto tabulku jsme použili v řádku 30, v programu Stupnice. S POKE SI+4,17 jsme jako tvar vlny zvolili trojúhelník. Tento tvar jsme potom ke změně charakteru zvuku nahradili pilovitým tvarem - zaměnili jsme 17 za 33.

Za další chceme vidět, jak můžeme měnit obalové křivky, které určují průběh hlasitosti uvnitř tónu.
Povšimněte si, pokud ~~xxxxx~~ určíte hlasitost a tvar vlny, jak je výše popsáno, že obdržíte pouze jeden tón.

3. Obalové křivky - nastavení

Hodnoty úderu a doznívání, které můžeme volit jako tvar vlny pro každý hlas odděleně, jsou zaznamovány společně jednou číselnou hodnotou.

Parametr úderu udává čas, ve kterém stoupá tón až do maximální /předem nastavené/ hlasitosti.

Parametr doznívání je míra toho, jak rychle opadává hlasitost na přídržné hladině. Zvolí-li se přídržná hladina 0, vydá parametr doznívání /opadávání/ čas doznění až po hlasitost 0 a určí tím trvání tónu.

Adresy, přiřazené jednotlivým hlasům a hodnoty odpovídající různým nastavením úderu lze čerpat z následující tabulky:

Nastavení úderu

Registr				Obsah	
Hlas	1	2	3	úder	opadávání
	5	12	19	15*16/měkce/..0*16 (tvrdě)	15/měkce/..0 tvrdě/

Zadáte-li pouze čas úderu, např. příkazem

POKE 54277,64
nastaví se 4as opadávání automaticky na nulů.

POKE 54277,66

Nastavuje úder na střední hodnotu /64=4*16/ a opadávání na nízkou hodnotu /2/. Hodnota 66 pak vyplývá jako součet 64+2. Souvislosti rozpoznáte nejlépe tak, když namísto

POKE 54277,66
napíšete
POKE 54277,4*16+2.

Nyní jste dospěli k bodu, kdy bude nejlepší ze všech poznatků sestavit program:

```
10 REM EXPERIMENTALNI PROGRAM
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:
   A=SI+5:H=SI+6:L=SI+24
30 PRINT"STISKNI JEDNU KLAVESU"
40 GET ZØ:IF ZØ=""THEN 40
50 POKE L+15
60 POKE A,1*16+5
70 POKE H,0*16+0
80 POKE TH,8:POKE TL,0
90 POKE FH,14:POKE FL,162
100 POKE W,65
110 FOR T=1 TO 200:NEXT
120 POKE W=0
130 GOTO 40
```

Řádek 50 - nastavuje hlasitost
Řádek 60 - určuje úder a opadávání
Řádek 70 - zde je nastaveno držení a doznívání
Řádek 80 - nastavuje klíčovací poměr /střídu/
Řádek 90 - určuje frekvenci
Řádek100 - zapojí se generátor a tvar vlny
Řádek110 - doba trvání tónu
Řádek120 - odpojí se generátor
Řádek130 - odskakuje na řádek 40 a čeká na stisk klávesy.

Použijeme hlas 1 k vytvoření tónu s krátkým úderem a krátkým opadnutím po dosažení maximální hlasitosti /řádek 60/.

Hlas, který přitom vyjde zní jako míč, který skáče po plechovém sudu. K vytvoření jiného zvuku, změníme tento řádek: K tomu zastavíme program a následovně změníme řádek:

60 POKE A,11*16+14

Tón, který tímto nastavením získáme, má zvuk hoboje, nebo jiného dřevěného dechového nástroje.

Vyzkoušejte si změnit tvar vlny a obalovou křivku, abyste zjistili, jak různé hodnoty tohoto parametru mění charakter tónu.

Přídržným nastavením můžete určit, jakou hlasitost si podrží tón po úderu. Trvání tónu se nastavuje smyčkou FOR...NEXT.

Podobně, jako u předchozího registru, se stanoví držení a doznívání tónu číselnou hodnotou, která se vypočítá součtem hodnot, uvedených v následující tabulce:

Držení a doznívání

Registr				Obsah	
Hlas	1	2	3	držení	doznívání
	6	13	20	15*16/hlasitě/ 0 *16 /nezněle/	15 /pomalu/ 0 /rychle/

Nahraďte hodnoty 0 v řádku 70 jakýmikoliv jinými hodnotami až max. 15, a poslechnete si, jaký zvuk přitom vyjde.

4. VOLBA HLASŮ A NOT

Jak jsme již dříve uvedli, je nutné při vytváření tónu, zadat dvě hodnoty, které jsme nazvali Hi-Byte a Lo-Byte frekvence.

Protože jsou hlasům přiřazeny rozdílné hodnoty /viz následující tabulka/, můžete v módu C 64 naprogramovat nezávisle na sobě tři hlasy a tím vytvořit trojhlasné hudební skladby.

Adresy tří tónových generátorů a POKE-hodnoty /Hi-Byte aLo-Byte/ tónů 5. oktávy.

Registr				Obsah pro noty 5. oktávy												
HLAS	1	2	3	C	C#	D	D#	E	F	F#	G	G#	A	A#	H	C
HI-BYTE	1	8	15	35	37	39	41	44	46	49	52	55	58	62	66	70
LO-BYTE	0	7	14	3	24	77	163	29	188	132	117	148	226	98	24	6

K vytvoření tónu C s hlasem 1, musíte zadat následující POKE-příkazy:

```
      POKE 54273,35:POKE 54272,3
nebo: POKE SI+1,35:POKE SI,3
```

Ten samý tón s 2.hlasem obdržíte zadáním:

```
      POKE 54280,35:POKE 54279,3
nebo: POKE SI+8,35:POKE SI+7,3
```

PROGRAMOVÁNÍ TÍSNIČKY V MÓDU C 64

S následujícím příkladem programu lze zkomponovat a reprodukovat písně. Počítač k tomu používá 1. hlasu. Povšimněte si, prosím, že jsou v programovém řádku 110 přiřazeny adresy často používaných registrů numerickým proměnným, a tím mohou být pohodlněji v programu použity.

Má-li být zvolena tvar vlny, postačí, zadá-li se v příslušném POKE-příkazu písmeno W namísto čísla 54276.
Dále byste si měli pro použití ve vlastním programu zaznamenat, jak se pracuje s DATA-řádky. V našem programu jsou v DATA-řádcích po sobě uložena tři čísla, která jsou nutná k označení /popisu/ jednoho tónu.
Jedná se zde o Hi-Byte a Lo-Byte frekvence a o trvání tónu. Trvání tónu je určeno smyčkou, která probíhá od 1 až po určitou hodnotu. Tato hodnota je v DATA-řádku vždy umístěna na 3. místě.

Přitom 125 odpovídá osminové notě, 250 čtvrteční notě, 375 čtvrtkové notě s tečkou, 500 poloviční a 1000 celé notě.

Podívejme se nyní na řádek 110: 17 a 103 jsou Hi-Byte a Lo-Byte noty "C". Číslo 250 na třetím místě způsobí, že nota bude čtvrteční. Také druhá nota je čtvrtkové, tentokrát je to však "E"atd.

Do DATA-řádků můžete zanést samostatně zvolené hodnoty, které sestavíte za pomocí tabulky not v dodatku F. Svoji melodii můžete volit tak dlouho, dokud je k dispozici paměťové místo v módu C 64. Musíte mít však na zřeteli, aby poslední programový řádek obsahoval výraz DATA-1,-1,0. Řádek 130 se postará o to, aby program při dosažení tohoto řádku skončil.

```
10 REM PRIKLADNA PISEN
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:
   A=SI+5:H=SI+6:L=SI+24
30 POKE L,15:POKE TH,13:POKE TL,15:POKE A,3*16+15:
   POKE H,9
40 READ X:READ Y: READ D
50 IF X=-1 THEN END
60 POKE FH,X:POKE FL,Y
70 POKE W,65
80 FOR T=1 TO D:NEXT
90 POKE W,0
100GOTO40  100GOTO 40
110DATA 17,103,250,21,237,250,26,20,400,21,237,100,
        26,20,250,29,69,250
120 DATA 26,20,250,0,0,250,21,237,250,26,20,250,29,69,
    1000,26,20,250,0,0,250
130 DATA-1,-1,0
```

ZVUKOVÉ EFEKTY

Na rozdíl od hudby ~~zobrazují~~ podmalovávají zvukové efekty události, které se dějí na obrazovce /explose lodi atd./, nebo informují uživatele programu-ev. jej varují /např. že je právě ve stavu, kdy může vymazat disketu/.

Uvádíme několik návrhů, které by měly vyprovokovat k experimentování:

- V průběhu znění tónu změňte hlasitost, docílíte tak efektů ozvěny.
- Vbíhejte sem a tam mezi dva tóny - docílíte chvějivého zvuku.
- Vyzkoušejte různé tvary vln.
- Zabývejte se důkladně obalovou křivkou.
- Rozdílným programováním tří hlasů /např. přidržením jednoho tónu v jednom hlase déle, než ve druhém/ lze docílit překvapivých efektů.
- Užijte obdélníkové vlny a měňte šířku impulsů.
- Experimentujte s generátorem šumu k vytvoření hluku z výbuchu, palby z pušek, kroků apod.
- Měňte rychlejším sledem frekvenci přes více oktáv.

PŘÍKLADY ZVUKOVÝCH EFEKTŮ

Zmíněné příklady programu můžete vsadit do přímého nebo pozměněného tvaru vlastního Basicového programu. Měly by Vás vyprovokovat k experimentování a ukázat Vám, jaké možnosti zvukových efektů jsou v módu C 64 skryty.

```
10 REM LOUTKA
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:
   A=SI+5:H=SI+6:L=SI+24
30 POKE L,15:POKE TH,15:POKE TL,15:POKE A,0*16+0:
   POKE H,15*16
40 POKE W,65
50 FOR X=250 TO 0 STEP-2:POKE FH,40:POKE FL,X:NEXT
60 FOR X=150 TO 0 STEP-4:POKE FH,40:POKE FL,X:NEXT
70 POKE W,0
```

```
10 REM VYSTREL Z PUSKY
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:
   A=SI+5:H=SI+6.L=SI+24
30 FOR x=15 TO 0 STEP-1
40 POKE L,X:POKE A,15:POKE H,0:POKE FH,40:POKE FL,200:
   POKE W,129
50 NEXT
60 POKE W,0:POKE A,0
```

```
10 REM MOTORY
20 SI=54272
30 FOR K=0 TO 24:READ X: POKE SI+K,X:NEXT
40 DATA 9,2,0,3,0,0,240
50 DATA 12,2,0,4,0,0,192
60 DATA 16,2,0,6,0,0,64
70 DATA 0,30,243,31:REM FILTR
80 POKE SI+4,65:POKE SI+11,65:POKE SI+18,65
```